Universidad de las Ciencias Informáticas

Facultad 1

Módulo de gestión de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga (ASIC)

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

Autor: Rosalba Rodríguez Pérez

Tutores:

Ing. Mariela Milagros Bony Fernández

Ing. Gregorio Ferrer Córdova

La Habana, junio de 2017

Año 59 de la Revolución

Declaración de Autoría

Declaro por este medio que yo Rosalba Rodríguez Pérez, con carnet de identidad 93040310999, soy la principal autora del trabajo titulado “Módulo de gestión de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga” y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso de la misma en su beneficio, así como los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____

Firma del Autor
Rosalba Rodríguez Pérez

_____

Firma del Tutor
Ing. Mariela Milagros Bony Fernández

_____

Firma del Tutor
Ing. Gregorio Ferrer Córdova

Datos del contacto

Tutor: Ing. Mariela Milagros Bony Fernández

Graduada en la Universidad de las Ciencias Informáticas en el año 2009. Durante su vida laboral se ha desempeñado en las siguientes líneas de desarrollo: diagnósticos, revisiones y auditorías de calidad a proyectos de producción de software; estudio de normas, estándares y modelos de calidad de software; desarrollo e implantación de sistemas de gestión de la calidad; gestión de recursos humanos y educación comparada. Ha realizado varias publicaciones en revistas y eventos científicos y también se ha desempeñado como profesora de Práctica Profesional. Además, ha tutorado varias tesis de diploma y trabajos presentados en eventos estudiantiles.

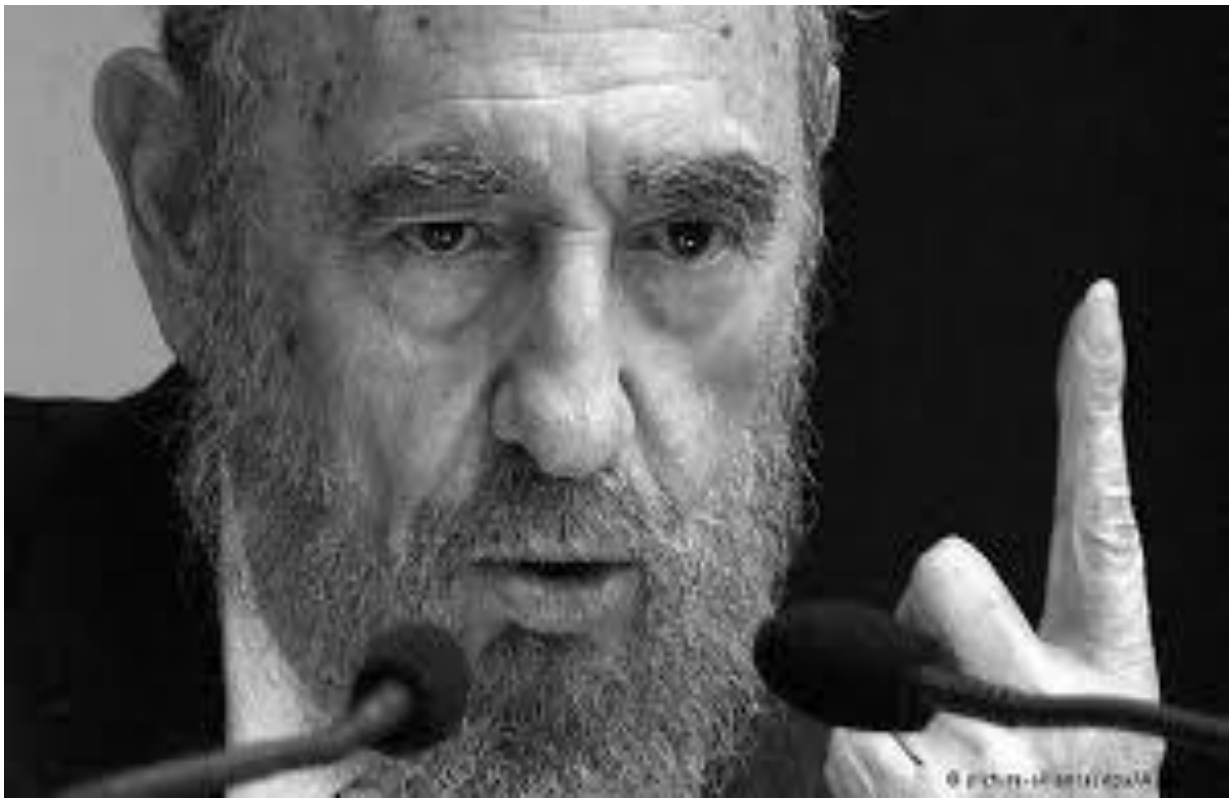
Correo electrónico: mmbony@uci.cu

Tutor: Ing. Gregorio Ferrer Córdova

Graduado en la Universidad de las Ciencias Informáticas en el año 2014. Desde su graduación se desempeña como especialista del Centro de Identificación y Seguridad Digital (CISED), vinculado al desarrollo de componentes para la identificación y reconocimiento de personas, a través de características biométricas como la huella dactilar y la imagen facial. Además, forma parte del equipo de entrenadores del movimiento de programación competitiva “Tomás López Jiménez” y pertenece al Comité de Jueces del Caribe en competiciones del ACM-ICPC (Competición Internacional Universitaria ACM de Programación, en inglés, *ACM International Collegiate Programming Contest*).

Correo electrónico: gcordova@uci.cu

En homenaje



«No. Nosotros no perseguimos ningún interés material, y es lógico que los imperialistas no lo entiendan, porque se guían por criterios exclusivamente chovinistas, nacionalistas, egoístas».

Discurso pronunciado en la Clausura del Primer Congreso del Partido Comunista de Cuba, celebrado en el Teatro Carlos Marx, el 22 de diciembre de 1975.

Agradecimientos

A mi familia que me ha apoyado sobremanera en estos últimos cinco años, a mi niña por esperarme, por quererme y por ser la cosa más bella que me da fuerza todos los días; a mi querido amigo, compañero, amor, que me ha hecho a crecer en estos años y que me ha reclamado más que nadie por mis desvaríos ocasionales; a mis padres por guiar mis primeros pasos, por hacer de mí una mujer emprendedora, luchadora, persistente, caprichosa con la vida y soñadora; a mi abuela por ser apoyo incondicional, por ser ejemplo de mujer para mí, por ser madre ante todo y abuela querida; a mi hermano, mi tata, por ser quien es, sostén de mi madre en mi ausencia, querido hermano mío; a Arbenia por cuidarme a mi hija durante estos años y por apoyarme en mi carrera; a mis suegros por su apoyo; a mi tía postiza Concha por su experiencia en la vida que me ayudó mucho; a mis primos Marién y Nestor por su amistad sincera y ayuda en estos años que nos conocemos; a mis compañeros en estos últimos años, a los que están ahora, a los que faltan y ya no están entre nosotros, a mi buen amigo Ortelio, que me regañó bastante en estos meses, a Yesenia, Abel, Carlos Yordan, Aimet, Osmel, Arlene, María, Lázaro, Rolando, Jorge Ignacio, Leydis, Alejandro, Alexander, Daynis y Marian, todos marcan esta etapa de mi vida que finaliza en breve. A mis tutores, maestros y amigos, guías del desarrollo de esta investigación en estos meses que han transcurrido; a Bartolo por apoyarme y ser un magnífico maestro no obstante mis ausencias en ocasiones; a Mayleidis por brindarme apoyo, a Alien por enseñarme a programar o intentar programar al menos; a Yari por ser como es, tan cándida y amigable, dispuesta a ayudar, a Delly por su comprensión e innata vocación de ayudar, a los maestros que han pasado por mi vida en esta universidad, los guardo fraternalmente en mi corazón; a los del centro CISED, a José, Joel, Yoermis, Dayneris, Vladimir, que aunque los conocí por poco tiempo, exceptuando a Yoel y Jose, les tengo un gran aprecio; a todos y cada uno de ellos porque de una manera u otra han marcado de forma significativa mis pasos en esta universidad.

Dedicatoria

Recuerdo que de niña quería ser alguien impresionante, que fuera orgullo de mis padres, que honrara a la Revolución, alguien que dejara huella. Hoy no estoy segura si dejaré huella en mi país, pero sí sé que en mi familia dejaré todo el amor que por ella siento. Por eso, el presente trabajo lo dedico a mi hermosa niña que llena todos los espacios de mi vida; a mi compañero en estos últimos cinco años, Alejandro; a mi madre querida que supo amarme en todos los momentos de mi vida; a mi padre que estuvo a mi lado en todos mis grandes momentos; a mi abuela que es pilar de mi familia, quien como nadie apoya, guía y sufre junto con sus hijos, a todo aquel que ha hecho de mí la persona que soy hoy.

Resumen

La Agencia de Supervisión e Inspección de Carga (ASIC) ofrece servicios de inspección y evaluación de cargas en varios lugares del país. En cada instancia de la agencia los informes y evidencias de los servicios se elaboran de forma manual, por lo que se genera un gran cúmulo de documentación que debe ser manejada en formato duro, aumentando así el riesgo de deterioro y/o pérdida de algún documento. Para solucionar la problemática planteada, se definió como objetivo general desarrollar un módulo para la gestión de informes e incidencias en el Sistema Integral de Administración que contribuyera a la calidad de los servicios brindados por la Agencia de Supervisión e Inspección de Carga. Con el fin de alcanzar el objetivo propuesto se estudiaron varios conceptos relacionados con la gestión documental, así como algunos sistemas nacionales e internacionales referentes a la gestión de reportes, indicadores e incidencias. También se definió el ambiente de desarrollo, los requisitos del software requeridos por el cliente y los artefactos para el análisis y diseño de la solución. Finalmente, se obtuvo un módulo que permite gestionar informes de inspección a contenedores y vehículos, incidencias de inspecciones a bultos, unidades de mercancía, inventarios y vehículos, así como cualquier hecho extraordinario ocurrido; también se gestionan los sellos emitidos para cada servicio, los certificados de inspección y se evalúa la eficacia del servicio. Para garantizar la calidad del software se ejecutaron pruebas funcionales, de carga y estrés y de integración y se solucionaron todas las no conformidades detectadas.

Palabras clave: servicio, inspección y evaluación de cargas, informes, incidencias.

Índice

Índice de figuras

Índice de tablas

Introducción

El hombre, partícipe y protagonista de la misma historia, factor indisoluble de cada uno de los eventos ocasionados en ella ha sido parte de su desarrollo, de su constante evolución, revolución e inigualable avance para con la misma humanidad y el entorno que la rodea. Tanto es así que es notable destacar que en su afán de mejorar la calidad en la vida cotidiana de las personas se han logrado maravillosos descubrimientos. Válido es concretar estos avances en el campo de la tecnología, donde lo que es novedoso hoy puede ser noticia vieja mañana y siempre se ha de necesitar un buen software que respalde la calidad de dicha tecnología o viceversa.

En el área empresarial ha resultado indispensable el uso de las Tecnologías de la Información y la Comunicación (TIC) para la realización de los procesos que se gestionan en sus diferentes áreas. Esta práctica agiliza y garantiza el objetivo que se quiere alcanzar acortando tiempo y costos para la misma empresa. Literalmente, lo que hace unos años requería de mano de obra de entre tres o seis empleados, actualmente puede ser gestionado por una sola persona, la cual se apoya en los conocimientos adquiridos en el manejo del software del cual dispone para realizar dichas tareas.

Cuba es un país que se caracteriza por el buen servicio que ofertan las empresas que radican en él, las cuales llevan como premisas la calidad en el servicio ofertado y la buena comunicación con el cliente. Mas, dichos centros no siempre cumplen con los estándares de gestión empresarial que son seguidos actualmente a nivel mundial, los cuales necesitan un total desenvolvimiento informático en todas sus esferas. Para ello, muchas de estas empresas se han propuesto informatizar completamente las áreas que involucran la gestión del personal, los procesos, entre otros.

La Agencia de Supervisión e Inspección de Carga (ASIC) es una empresa que ofrece servicios de inspección y evaluación de cargas en varios lugares del país, lo cual conlleva una alta demanda de sus esfuerzos, incluida la gestión de los servicios que brindan. La gestión de los informes e incidencias resultantes de los servicios brindados no se realiza de forma eficiente, por lo que se genera un gran cúmulo de documentación que debe ser manejada en formato duro, aumentando el riesgo de deterioro y/o pérdida de algún documento. Lo mismo sucede con la gestión de los hechos extraordinarios, los certificados de inspección

entregados a los clientes y los sellos. Esta misma problemática se refleja en las demás instancias de la agencia distribuidas a lo largo del país, por lo que otra de las necesidades de la agencia es la unificación del proceso de contratación a nivel nacional.

Ante estas dificultades se plantea como **problema de investigación** ¿Cómo contribuir a la gestión de los informes e incidencias resultantes de los servicios brindados en la Agencia de Supervisión e Inspección de Carga? Se define como **objeto de estudio** los procesos de gestión empresarial asociados a los servicios.

Como una alternativa de solución al problema planteado, se define como **objetivo general** de la investigación: desarrollar un módulo para la gestión de informes e incidencias en el Sistema Integral de Administración, que contribuya a la calidad de los servicios brindados por la Agencia de Supervisión e Inspección de Carga. Por ello, se define como **campo de acción** los procesos de gestión de informes e incidencias en las agencias de supervisión.

Las **preguntas científicas** derivadas del objetivo general son las siguientes:

- ¿Cuáles son los referentes teóricos fundamentales que sustentan la investigación relacionados con la gestión de servicios?
- ¿Cuál es el estado actual de las herramientas, tecnologías y metodologías para el desarrollo del módulo de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga?
- ¿Qué funcionalidades del módulo son necesarias definir e implementar para la gestión de informes e incidencias?
- ¿Cómo validar y probar las funcionalidades del módulo de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga?

Las **tareas de investigación** definidas para darle respuesta a las preguntas científicas son:

- Análisis de las tendencias actuales relacionadas con los sistemas de gestión empresarial.

- Selección de las tecnologías, las herramientas y la metodología necesarias para implementar el módulo de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga.

- Diseño de los artefactos requeridos por la metodología de desarrollo seleccionada.

- Implementación del módulo de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga.

- Ejecución de pruebas para validar el módulo desarrollado.

El **método científico** de investigación es la forma de abordar la realidad, es una regularidad interna del pensamiento humano, de estudiar los fenómenos de la naturaleza, la sociedad y el pensamiento, se emplea de forma planificada y consciente para explicar y transformar al mundo. En la presente investigación se utilizarán algunos de ellos según su clasificación: teóricos y empíricos, o métodos particulares. [1]

Los **métodos teóricos** que serán utilizados en el análisis e interpretación de la información obtenida son:

- Análisis-Síntesis: en el análisis de los procedimientos referentes a la gestión empresarial, para identificar estándares, herramientas y tecnologías idóneas para el desarrollo de la aplicación en cuestión.

- Sistémico: para lograr un mayor entendimiento de las características intrínsecas del módulo, a través de la realización de diagramas.

- Modelación: para modelar cada uno de los procesos de gestión, este método abstrae de la realidad al modelo informático para así buscar la manera de implementarlo, detallando las características de cada una de las variables seleccionadas.

Los **métodos empíricos** a utilizar durante la investigación son los siguientes:

- Recopilación de información:

- ✓ Revisión documental: al realizar un análisis de los documentos facilitados por la ASIC para llegar a una mejor comprensión del entorno de trabajo necesario.

El presente trabajo de diploma tiene estructurado el contenido en tres capítulos de la siguiente manera:

Capítulo 1: Fundamentación teórica. En este capítulo se describen los principales conceptos asociados al problema de investigación planteado. Se hace un estudio sobre las principales tendencias a nivel nacional e internacional de los sistemas de gestión empresarial, así como de métodos tradicionales para la gestión empresarial de informes e incidencias. Finalmente, se seleccionan las tecnologías, herramientas y metodología adecuadas para el desarrollo del módulo.

Capítulo 2: Propuesta de solución. Se hace referencia a la propuesta de solución, haciendo énfasis en la evolución de dicha solución a través de las fases iniciales: Inicio y Elaboración, de la metodología de desarrollo de software AUP[1], y se generan diferentes artefactos que permiten una mejor comprensión de lo que se desea implementar.

Capítulo 3: Implementación y prueba. En este capítulo se les da cumplimiento a los planes trazados mediante la implementación de la propuesta de solución y finalmente se examina la calidad del módulo a través de pruebas de software.

[1] *Agile Unified Process*

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El presente capítulo consolida las bases de la investigación al realizar una búsqueda de los conceptos fundamentales que respaldan esta investigación, para así poder enfocar en qué entorno se desarrolla. De manera concreta, sería de utilidad conocer acerca de aplicaciones web que presentan funcionalidades similares a los establecidos por la ASIC; de esta forma se seleccionarían eficazmente las metodologías, herramientas y tecnologías acordes para el diseño e implementación del módulo deseado.

1.2 Conceptos fundamentales

- Documentación: En sentido restringido, la documentación como ciencia documental se podría definir como la ciencia del procesamiento de información, que proporciona un compendio de datos con un fin determinado, de ámbito multidisciplinar o interdisciplinar. [2]

  La documentación es como una ciencia auxiliar e instrumental. También es una ciencia en sí misma y una de las finalidades primordiales de la misma es informar; en sentido general, las ciencias de la documentación y la documentación como sinónimos, si el contexto no perturba la intención del emisor, es decir, si no se distorsiona el mensaje del interlocutor porque no se dé ambigüedad semántica. [3]

  Este trabajo maneja la documentación de las gestiones que se ponen de manifiesto en los procesos realizados en la ASIC. Todo el cúmulo de documentación generado de los servicios que la empresa presta se recoge en el sistema a implementar, sosteniendo el propósito de aligerar los gastos y el tiempo en las tareas comprendidas en la entidad.
- Administración: es una ciencia social que persigue la satisfacción de objetivos institucionales por medio de una estructura y a través del esfuerzo humano coordinado [4]; consiste en lograr un objetivo predeterminado, mediante el esfuerzo ajeno. [5]

  Para tener una mayor comprensión del entorno empresarial que envuelve la síntesis de la investigación, el autor de esta investigación realizó un bosquejo por los documentos que explican el sistema administrativo de la ASIC, llegando a la conclusión que este trabajo se desarrolla en el área de la producción u operaciones, que es una de las cuatro áreas funcionales existentes en una

empresa, cualquiera sea esta. En este caso, al área mencionada se puede asociar el departamento de administración de las tecnologías de la información.

- Administración de las tecnologías de la información: tiene como objetivo el desarrollo de sistemas de información que ayudan a resolver problemas de la administración. Las organizaciones tanto lucrativas como no lucrativas deben mantenerse a la vanguardia en sus diferentes campos de acción, y para poder realizar esto deben contar con lo último en sistemas de información que puedan cubrir las necesidades tanto de su entorno interior como de su entorno exterior. El rediseño de una organización basado en la adquisición de nuevas tecnologías de información que den paso a un nuevo sistema de información no es tarea fácil, se tienen que tomar en cuenta muchos aspectos de la organización (recursos humanos, económicos y operativos) y se debe seguir un proceso previamente definido para poder renacer y hacer que este rediseño sea todo un éxito. [6]

  Según el propio Chekland "lo primero que se debe realizar en el rediseño de una organización con sistemas de información es el análisis de las necesidades", todo ello para realizar el cambio estructural deseado para la empresa los cuales pueden ser Automatización[2], Racionalización[3], Reingeniería[4] y Cambios del paradigma.

- Informe: El concepto de informe, como derivado del verbo informar, consiste en un texto o una declaración que describe las cualidades de un hecho y de los eventos que lo rodean. El informe, por lo tanto, es el resultado o la consecuencia de la acción de informar (difundir, anoticiar). [7]

  Refiere hechos obtenidos o verificados por el autor (reconocimientos, investigaciones, estudios, o trabajos). Además, aporta los datos necesarios para una cabal comprensión del caso, explica los métodos empleados y propone o recomienda la mejor solución para el hecho tratado; es la exposición de los datos obtenidos en una investigación de campo o bibliografía de un determinado tema; por eso, su propósito es principalmente informativo.

  Los informes técnicos, en el caso que compete a este trabajo, es la constancia de una labor realizada por individuos pertenecientes a una institución a otra. Son la exposición por escrito de las

[2] *Consiste en el uso de computadoras para acelerar el desempeño de tareas existentes*.
[3] *Consiste en la agilización de los procedimientos operativos estándar eliminando cuellos de botella obvios, de modo que la automatización haga más eficientes los procedimientos operativos.*
[4] *Rediseñar radicalmente el flujo de trabajo y los procesos de negocios que se siguen para generar productos y servicios, con el objeto de reducir radicalmente los costos del negocio*

circunstancias observadas en el examen de la cuestión que se considera, con explicaciones detalladas que certifiquen lo dicho.

Se trata de una exposición de datos o hechos dirigidos a alguien, respecto a una cuestión o un asunto, o a lo que conviene hacer del mismo. Es, en otras palabras, un documento que describe el estado de un problema científico. Suele prepararse a solicitud de una persona, una empresa o una organización. [8] En estos se ven reflejados cada uno de los servicios prestados con sus respectivas observaciones. La ASIC, como agencia de supervisión, genera estos documentos para validar los servicios desempeñados por sus trabajadores, no obstante, suelen ocurrir situaciones fuera de lo normal que quedan recogidas dentro de las incidencias, estrechamente vinculadas al informe del servicio al cual pertenece.

- Incidencia: Puede referirse a un hecho que acontece mientras está ocurriendo un negocio u otra situación, relacionada con ello. [9] Es cualquier evento que no forma parte del desarrollo habitual del servicio y que causa, o puede causar, una interrupción del mismo o una reducción de la calidad de dicho servicio. Una incidencia es cualquier comportamiento inesperado de un servicio que afecta negativamente a la calidad del mismo.

  Estas tienen lugar, como bien se dijo anteriormente, cuando sucede algo imprevisto, que debe archivarse para que quede salvada la información referente al hecho. Es una extensión del informe, que pretende dejar explícito las especificaciones de la situación surgida.

1.3 Estudio de sistemas homólogos

El análisis de las particularidades de los sistemas que realizan la gestión empresarial, permite conocer características que, por su relevancia, pueden ser fundamentales para la propuesta a desarrollar, o servir como objeto de estudio para realizar una selección acertada de estándares, herramientas y tecnologías a utilizar para su construcción.

1.3.1 Principales alternativas identificadas a nivel internacional

OpenBizview es un software libre de reportes e indicadores de gestión, diseñado para ayudar a las organizaciones y personas a obtener información de manera inmediata y sencilla. **OpenBizView** fue concebida como repositorio de reportes e indicadores de gestión desarrollados con Birt Report, engloba su *runtime* y permite una sencilla administración desde un solo lugar. El acceso a los mismos es distribuido de

manera sencilla desde la plataforma, permitiendo una seguridad robusta al momento de acceder a la información. Desarrollado con tecnología Java Server Faces permite que sea una aplicación portable independiente de su estructura tecnológica, adaptándose a cualquier requerimiento.

Este software, muestra las oportunidades accesibles para la gestión empresarial, tiene facilidades para las empresas que carecen de un sistema de gestión para sus procesos, sin embargo, está ajustado de acuerdo a las tendencias actuales en el exterior del país, donde la mayoría de la información que maneja el usuario está guardada en un servidor web, es decir, es un software anclado en la red al cual se accede vía internet. Esto no significa que no pueda ser de utilidad en las empresas cubanas, donde muchas empresas mayoristas tienen acceso a "la red de redes", pero para su uso se ha de pagar una cuota cada cierto tiempo, a diferencia del propuesto en esta investigación que se ha de poner en manos de la institución con toda la documentación que respecta a este, para una posible recodificación en el futuro. En fin, este software no cumple con todos los requerimientos necesarios para la implementación del sistema, sea así la creación de cuantiosas y detalladas tablas que precisan los servicios que efectúa la ASIC, quedando una brecha de información en el momento de generar reportes que necesitan datos específicos de la base de datos.

Mantis es una aplicación de software libre multiplataforma que permite gestionar las incidencias en una empresa, en un sistema o en un proyecto. Es un sistema fácil de usar y adaptable a muchos escenarios. Además, cuenta con diferentes *plugins* que aumentan la capacidad de trabajo con la herramienta. Cuenta con una gran variedad de funcionalidades que permiten que todos los objetivos insertados por el cliente queden cubiertos completamente; entre estas cabe destacar el reporte de incidencias, que permite a los distintos usuarios reportar tickets de cualquier tipo, ya sean incidencias técnicas, peticiones de soporte o bugs de un sistema; un sistema de permisos de usuario, que permite identificar a los distintos usuarios que acceden al sistema; completa descripción y estado de incidencias, el usuario cuenta con una gran cantidad de opciones y campos a rellenar, con el fin de hacer más fácil el trabajo a los responsables de resolverlas; notificaciones de usuario por los cuales Mantis permite al usuario recibir notificaciones mediante correos electrónicos.

Es un software que sería una buena opción para las gestiones de las incidencias de la ASIC, mas, no cuenta con todas las funcionalidades exigidas por esta para con los servicios que le es imprescindible manejar. La ASIC tiene un sistema de gestión de informes e incidencias relacionados entre sí ya que cada uno de estos

documentos se realizan al culminar el operativo, y las incidencias son detalles precisos de servicios de inspección realizados. De cierta manera, esta relación de dependencia se pone de manifiesto en los campos existentes en cada informe, que recoge los parámetros esenciales del servicio realizado.

1.3.2 Principales alternativas identificadas a nivel nacional

En Cuba, existen aplicaciones web que gestionan los procesos empresariales, pero no hay constancia de procesos de gestión de informes e incidencias en estos. Sin embargo, existen empresas como Cubacontrol e Intermar, empresas cubanas que se rigen por las condiciones generales y normas internacionales que establecen las relaciones y el código de práctica, entre las entidades que prestan el servicio de supervisión comercial asociado al ejercicio de compra-venta internacional. Al igual que la ASIC, estas empresas organizan su trabajo con un importante número de agencias de supervisión a nivel mundial y cuentan con Unidades Empresariales de Base en todo el territorio nacional; realizan la gestión de los servicios a partir de una solicitud de servicio que representa la constancia de los requerimientos del cliente, de esta manera, a medida que van realizando el trabajo se van generando informes que respaldan el quehacer de los trabajadores que desempeñan sus funciones en la empresa.

1.4 Metodologías, Herramientas y Tecnologías

1.4.1 Metodología de desarrollo de software

La metodología de desarrollo de software, en la ingeniería de software, tiene como función separar el proceso en distintas fases o etapas, que contienen actividades enfocadas a una mejor planificación y administración del mismo. Puede incluir la definición previa de una serie de artefactos que son creados y completados con el equipo que desarrolla o mantiene la aplicación. [10]

Existen numerosas propuestas metodológicas que inciden en diferentes dimensiones del proceso. Hay propuestas más tradicionales que se centran especialmente en el control, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben generar, así como las herramientas a utilizar. Otras se enfocan en el factor humano o el producto de software. Esta filosofía de las metodologías ágiles da mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. [11]

AUP es una versión simplificada de RUP[5]; este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de aplicaciones de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad. [12]

Consta de cuatro fases: inicio, elaboración, construcción y transición, así como siete disciplinas las cuáles son: modelo, implementación, prueba, despliegue, gestión de configuración, gestión de proyectos y entorno.

Se realiza una variación de esta metodología para adaptarla al ciclo de vida definido para la actividad productiva de la UCI ante lo cual surge AUP-UCI como metodología oficial a utilizar en la universidad. De las cuatro fases que propone AUP 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 tres fases de AUP en una sola llamada Ejecución y se agrega la fase de Cierre.

Por estos motivos, la metodología utilizada en esta investigación es AUP-UCI, ya que esta es la definida por la UCI para el desarrollo de todos sus proyectos, por lo que se rige por las regulaciones y estándares predefinidos por dicha institución para la producción en sus centros de desarrollo.

1.4.2 Modelado

El lenguaje de modelado UML[6] tiene una notación gráfica muy expresiva, que permite representar en mayor o menor medida todas las fases de un proyecto informático. UML provee de un mayor rigor en la especificación, permite realizar una verificación y validación del modelo realizado, además de automatizar determinados procesos, generar código a partir de los modelos y a la inversa (a partir del código fuente

[5] *Proceso Unificado de Rational*
[6] *Lenguaje Unificado de Modelamiento*

generar los modelos). [13] Permite documentar todos los artefactos de un proceso de desarrollo (requerimientos, arquitectura, pruebas, versiones). Una de sus principales características está relacionada a la posibilidad de poder conectarse con lenguajes de programación (ingeniería directa e inversa).

Existen en el mercado una gran variedad de herramientas para modelado con UML de software como son: ArgoUML, Poseidon for UML, OpenAmeos, Visual Paradigm for UML, StarUML, Rational Software Modeler (sucesor de Rational Rose), Enterprise Architect, Umbrello UML Modeler, UML Designer, entre otras.

Según un estudio realizado por José Arturo Mora Soto, doctor en *Computer Science* y profesor en universidades de México, Argentina y España, donde tomó en cuenta características que satisfacían tanto para la enseñanza como para uso comercial, las cuáles fueron las siguientes: código abierto, tipo de licencia (considerando de importancia un licenciamiento académico a un coste accesible), lenguaje de programación utilizado, integración con entornos de desarrollo, coste, versión de UML, diagramas que soporta, soporte para MDA, soporte para XMI, generación de código, capacidades de ingeniería inversa, sistema operativo y requisitos de reinstalación; llegó a la conclusión que solo dos de ellas satisficieron la calidad global deseada: Rational Software Modeler y Visual Paradigm for UML. En las propias palabras del doctor "…ambas herramientas son muy potentes, soportan todos los diagramas de UML y además ayudan para la gestión de requisitos de software…"

Con el fin de ahondar en la investigación la autora de esta investigación se propuso exponer algunas características de ambas herramientas para decidir luego a partir de ellas cual ha de utilizar.

IBM Rational Software Architect Designer (RSAD, y antes denominado RSA) es una completa herramienta de diseño, modelado y desarrollo para la entrega de software de extremo a extremo. Utiliza lenguaje de modelado unificado (UML) para diseñar aplicaciones Java de empresa y servicios web. Rational Software Architect Designer se basa en la infraestructura de software de código abierto Eclipse y se puede ampliar con varios *plugins* de Eclipse. También puede mejorar la funcionalidad adecuándola a sus requisitos específicos con ampliaciones de Rational adquiridas por separado. [14]

Visual Paradigm es una herramienta para el desarrollo de aplicaciones utilizando modelado UML. Es ideal para ingenieros de software, así como para analistas y arquitectos de sistemas que estén interesados en

construir software a gran escala y necesiten confiabilidad y estabilidad en el desarrollo orientado a objetos. Brinda navegación intuitiva entre la escritura del código, su visualización y un ambiente visualmente superior de modelado.

Se decidió utilizar Visual Paradigm por su estabilidad de ejecución en diferentes sistemas operativos y la facilidad de abrir y trabajar con un modelo UML utilizando el mismo programa sin importar el sistema operativo y sin afectar en absoluto el trabajo hecho, es un potente generador de informes en formato PDF[7]/HTML[8], permite realizar ingeniería inversa y directa, soporta varios usuarios trabajando sobre un mismo proyecto y permite agilidad en el trabajo del analista; además destacar que esta herramienta guarda todo el modelo en un solo fichero, bastando con copiarse solo ese fichero para estar seguro de que todo el trabajo está encapsulado en él. Finalmente, Visual Paradigm es una herramienta que trabaja muy bien en ordenadores "poco potentes". Además, se cuenta con experiencia trabajando en la misma.

1.4.3 Lenguajes de programación

Para el desarrollo de la aplicación se realizó un análisis teniendo presente su objetivo intrínseco: el trabajo con la documentación. Para ello se tomaron en cuenta dos lenguajes de programación: PHP Y Python. Para tomar una decisión al respecto la autora de esta investigación realizó una búsqueda *online* con el fin de distinguir cuál de ellos es el más adecuado para el objetivo trazado. Estos son los resultados:

PHP es un lenguaje fácil de aprender, se puede desarrollar a través de decenas de *frameworks* como son Symfony, Laravel, Zend, Cake, Codelgniter, entre otros; tiene una gran demanda laboral ya que de todas las ofertas webs el 90 % pide este lenguaje, así como una amplia comunidad, es compatible con casi todos los *Hostings*. Lo utilizan grandes empresas como Wikipedia y Facebook; entre muchas otras, Wordpress está realizado en este lenguaje. Sin embargo, este lenguaje es lento comparado con sus competidores y más si se utiliza un *framework* como Larevel, esto se debe principalmente a que no compila los archivos, no genera un *bytecode* intermedio ya que son texto plano que debe interpretar en cada ejecución, tiene más

[7] *Por sus siglas en inglés: Portable Document Format.*

[8] *Lenguaje de programación web, sus siglas en inglés significan: HyperText Markup Language.*

líneas de código que Python, es decir, para hacer la misma tarea que Python se necesitarían más líneas de código. [15]

Python es un lenguaje agradable a la vista ya que su sintaxis es bella y fácil de escribir. Es posible ejecutarlo en cualquier sitio, funciona en escritorio, web y servidor, además de contar con un rendimiento excelente. Lo utilizan grandes empresas como Google y Youtube, además de ser muy usado en el software libre, sobretodo en Linux. La forma más fácil de trabajar es por medio de Django, una buena opción para proyectos grandes, y Flask, para pequeños. Es un lenguaje rápido, fácil de desarrollar ya que no se necesita un servidor web para que funcione, solo Python, es un gestor interno de paquetes, es decir, instala y borra todos los módulos, además de contar con una comunidad muy buena que provee de todos los tutoriales y paquetes necesarios. Sin embargo, tiene pocos *Hostings* compatibles, necesita de un *framework* para desarrollar web porque es muy complejo de otra manera, para desplegar en un servidor web es necesario poseer buenos conocimientos. [15]

Para programar web ambos lenguajes son excelentes, sus diferencias radican en muy pocos elementos, en sí es una elección de apego al lenguaje por comodidad o por experiencia en su desarrollo que en características rotundas. De esta manera se seleccionó Python, lenguaje de programación poderoso y fácil de aprender, el cual cuenta con una variedad de librerías que permiten crear y generar documentos, entre otras funcionalidades, así como herramientas para procesamiento de lenguaje natural, como NLTK[9].

Python en su versión 3.4.1 cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su elegante sintaxis y tipado dinámico junto con su naturaleza interpretada, hacen de este un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. [16]

[9] *Del inglés Natural Languaje Toolkit.*

Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA[10], Industrias Light & Magic y todas las distribuciones Linux, en las que cada vez representa un tanto por ciento mayor de los programas disponibles. [17] Es administrado por la Python Software Foundation, una organización sin fines de lucro orientada al avance de tecnologías de código abierto relacionadas con dicho lenguaje de programación. Posee una licencia de código abierto, denominada Python Software Foundation License.

Existen varios *frameworks* para desarrollar código en Python, como son Bottle, Cyclone.io, CherryPy, Tornado, Flask, Klein, ObjectWeb, Pecan, Django, Pyramid, entre otros. Haciendo una comparación entre estos *frameworks* mencionados se llegó a la conclusión que de ellos el más completo es Django, el cuál es un *framework* de alto nivel que promueve el desarrollo rápido, el cuál toma especial importancia a los temas de seguridad, de modo que ayuda a los desarrolladores a detectar fallos en su código que podrían comprometer la seguridad de la aplicación, además está construido sobre Python, fomenta el desarrollo ágil permitiendo la construcción de aplicaciones de alta calidad en un breve período de tiempo. Utiliza una modificación de la arquitectura MVC[11] llamada Modelo-Plantilla-Vista (MTV, por sus siglas en inglés).

Django impulsa el desarrollo de código limpio y promueve la utilización de buenas prácticas de desarrollo web como el principio DRY[12]. Entre algunos casos de éxito se encuentran: Instagram[13], Pinterest[14] y The New York Times [18].

Lo anteriormente expuesto no significa que los demás frameworks no son de utilidad para los desarrolladores, por ejemplo, CherryPy permite crear aplicaciones web de la misma manera que es posible hacerlo en otro programa orientado a objetos, por lo que su código fuente se reduce y se obtiene en poco tiempo; ObjectWeb es un *framework* rápido que no depende de librerías externas y trata a Python como es

[10] *Agencia del gobierno estadounidense responsable del programa espacial civil, investigación aeronáutica y aeroespacial.*

[11] *Modelo-Vista-Controlador*
[12] *Principio orientado a reducir la repetición de información de todo tipo, especialmente útil en arquitecturas de múltiples niveles.*
[13] *Red social y aplicación para subir fotos y videos. Disponible en: https://www.instagram.com/*
[14] *Plataforma para compartir imágenes. Disponible en https://es.pinterest.com/*

en su esencia, un lenguaje orientado a objetos; finalmente, Pyramid es un *framework* de código abierto, uno de los primeros en ser compatibles con Python 3 y ofrece una completa documentación que permite a los usuarios familiarizarse con la interfaz rápidamente.

Para la selección de un *framework* de desarrollo web del lado del cliente se observaron opciones como son Boostrap, Foundation y Materialize. A continuación, se exponen varias de las características de estos a fin de escoger el más acertado para esta investigación.

Boostrap se utiliza principalmente para desarrollo de aplicaciones web, de modo sencillo y ligero, ya que puede bastar con un fichero CSS y uno Java Script. Está basado en los últimos estándares de desarrollo de Web: HTML5, CSS3 y Java Script/JQuery. Utiliza plugins de jQuery para validar entradas de datos, visualización de tablas, grafos, entre otros. Presenta una curva de aprendizaje baja y es compatible con todos los navegadores habituales. Su arquitectura está basada en LESS y Reset CSS basado en Normalize.css. El diseño de las páginas web se enfoca en permitir al usuario visualizarlas perfectamente en un amplio rango de dispositivos, como son: navegadores en el PC, tabletas y *smartphones*. [19]

Foundation es una de las mayores competidoras de Twitter Bootstrap ya que es un *framework* flexible y estable que posee una serie de herramientas para crear sitios web *responsive*, teniendo en cuenta el enfoque de diseño Mobile First. Es compatible con los navegadores más populares y tiene una extensa documentación. Este *framework* no sólo permite crear sitios web, sino también aplicaciones y correos; incluye plantillas HTML para empezar con un proyecto rápidamente e incluso usarlo en conjunto con Sass y Rails. [20]

Materialize es un moderno *framework* basada en los principios del Material Design creado por Google. Como su nombre indica, Materialize es apropiada si se quiere aplicar tanto los principios como el diseño del Material Design. Este *framework* permite implementar el Material Design a un sitio web sin mayores dificultades. Al especializarse en esta técnica de diseño, posee una serie de componentes únicos como diseño basado en tarjetas, efectos animados, menú flotante para móviles, entre otros. También posee los componentes usuales de todo *framework* como botones, barra de navegación, entre otros. [20]

Se selecciónó Boostrap, que en su versión 3.0 es rápido y fácil realizar con él una página web funcional, lo diseñado con Boostrap es posible verlo bien en cualquier navegador, es extensible, es decir, brinda la posibilidad de adaptarlo a los requerimientos que el programador desee. De manera general, Boostrap es una buena opción para esta investigación por las características anteriormente expuestas, de esa manera se armoniza el trabajo deviniendo en buenos resultados.

1.4.4 Sistema gestor de base de datos

Los gestores de base de datos han sido, desde los inicios de la programación de gestión hasta nuestros días, clave para ayudar al desarrollo eficiente de empresas, entes públicos y cualquier organismo que utilice datos, siguiendo en constante avance y mejora. Como tal, estos son un programa o grupo de programas informáticos que permiten trabajar con bases de datos, ya sea en su creación o con su mantenimiento. Destaca el papel del administrador de estas, pudiendo realizar copias de seguridad, modificar, eliminar, clonar, y más, respecto a las bases de datos que administra.

Existen distintos tipos de lenguajes de base de datos, pero el principal y más utilizado, desde el origen de la programación de gestión, es el lenguaje SQL (Structured Query Language). Es un lenguaje de consulta estructurada que permite acceder a la gestión de las bases de datos relacionales y, por consiguiente, realizar tareas en ellas, permitiendo realizar consultas para recoger, eliminar, agregar o modificar información.

Para poder desarrollar este lenguaje, hace falta un gestor de base de datos. Existen muchos, algunos de acceso libre y otros de pago; los mejores de ellos y más utilizados son Oracle, SQL Server, MySQL, FireBird, PostgreSQL, entre otros.

Oracle es uno de los sistemas de gestión de base de datos relacional más fiable y usado. Pertenece a Oracle Corporation y se desarrolló en 1977. Está construido alrededor de un marco en el que se puede acceder directamente a los objetos a partir del lenguaje de consulta SQL. Oracle es una arquitectura escalable y muy utilizada por las empresas. Tiene su propio componente de red para que pueda haber comunicación a través de las redes. Se ejecuta en casi todas las plataformas (Windows, Unix, Linux, MAC OS…). La principal y peculiar característica de Oracle es que, su arquitectura, se divide entre la lógica y la

física. A grandes rasgos, esto permite una mayor flexibilidad en las redes de datos y, a la vez, robustez en la estructura de los datos. [21]

MySQL es un gestor de simple instalación que actúa del lado del cliente (servidor) y de código abierto con licencia comercial disponible. Actualmente, pertenece a Oracle Corporation. Gestiona bases de datos relacionales, es multiusuario y el más usado dentro del software libre. Destaca por requerir de poca memoria y procesador para funcionar, dando lugar además a una mayor velocidad en sus operaciones. Es usado principalmente para el desarrollo web. [21]

PostgreSQL en su versión 9.4.1, es un Sistema Gestor de Bases de Datos (SGBD) objeto-relacional que se encuentra distribuido bajo la licencia de software libre BSD. Es mantenido por la organización PostgreSQL Global Development Team y cuenta con una amplia comunidad de usuarios y programadores que colaboran activamente. Se centra en que el software sea robusto, de calidad, fácil de mantener y se destaca por su estabilidad, potencia, robustez y facilidad de administración. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios, accediendo paralelamente al sistema. [22]

Lo que diferencia a PostgreSQL de otras bases de datos es la facilidad con que se puede extender, por lo general, sin compilar ningún código. Este SGBD no sólo incluye características avanzadas como funciones de ventana SQL y replicación de *streaming*, raramente encontradas en otras bases de datos de código abierto, sino que también las ejecuta rápidamente y supera en muchas ocasiones a otras bases de datos incluso propietarias, como es el caso de Oracle, SQL Server y DB2. [23]

1.4.5 Entorno de desarrollo

Para programar en Python basta con instalar Python y utilizar el IDE que viene con la instalación, pero existen muchos más IDEs que probablemente son de mayor comodidad para el programador a la hora de desarrollar el proyecto. De otra manera, no se puede concluir que un IDE es mejor que otro ya que esta conclusión depende de muchos factores, no obstante, lo más importante es que el IDE utilizado resulte cómodo y satisfaga las necesidades de quien lo utiliza. Existen varios de estos como son Pycharm IDE, PyDev IDE, Sublime Text 3 IDE, Wing IDE, Vim IDE, entre otros.

El IDE Pycharm es muy completo, creado por JetBrains. Este IDE es profesional y viene en dos modalidades: una edición Free y otra muy completa privada que apunta a empresas de desarrollo de software. La popularidad del IDE Pycharm se puede medir a partir de que grandes empresas como Twitter, Groupon, Spotify, *ebay* y telefónica, han utilizado éste para su trabajo. La mayoría de sus características están disponibles en la versión gratuita, se integra con IPython, soporta Anaconda, así como otros paquetes científicos como *matplotlib* y NumPy. Características como desarrollo remoto, soporte de bases de datos, soporte de frameworks de desarrollo web, entre otras, están disponibles solo para la edición profesional de PyCharm. Algo muy útil de Pycharm es su compatibilidad con múltiples marcos de desarrollo web de terceros como Django, Pyramid, web2py, motor de aplicaciones Google y Flask, lo que lo convierte en un competo IDE de desarrollo de aplicaciones rápidas. [24]

PyDev es libre de costo y está lleno de características poderosas para programar de manera eficiente en Python. Es un *plugin* de código abierto y se ejecuta en Eclipse. Está integrado con Django, completa el código de manera automática, soporta multilenguaje; tiene plantillas de código, análisis de código, marcado de errores y mucho más. Se mantiene siempre actualizado y contiene una gran comunidad de usuarios y empresas de patrocinio como Liclipse, Squish, TraceTronic y alguna más. Aunque PyDev califica como uno de los mejores IDE de Python de código abierto, también viene empaquetado junto con otro producto llamado Liclipse, un producto comercial construido sobre Eclipse que proporciona mejoras en la usabilidad y temas adicionales. [24]

Es seleccionado Pycharm en su versión PyCharm Professional-2016.2.3, ya que es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Provee funcionalidades que permiten una experiencia única y aumentan en gran medida la productividad:

- Completamiento de código de manera inteligente y señalamiento de errores con reparación de los mismos de forma automatizada.
- Integración con marcos de trabajo de desarrollo web modernos como Django.
- Depurador integrado y herramientas de pruebas.
- Soporte para bases de datos e integración con sistemas de control de versiones.

En adición a Python, PyCharm soporta JavaScript, CoffeeScript, TypeScript, HTML/CSS, Cython, lenguajes de plantilla, AngularJS, Node.js y más. [25]

1.5 Conclusiones parciales

El análisis y profundización de los conceptos fundamentales y las tendencias actuales, posibilitó conocer el entorno del sistema sobre el cual se trabajará. La metodología de desarrollo a utilizar fue AUP-UCI pues es la seleccionada a nivel de universidad para el ciclo de vida del sistema a implementar. Además, se seleccionaron como lenguaje de modelado a UML y como herramienta a Visual Paradigm 8.0, debido a que juntas proporcionan un ambiente de diseño óptimo para el analista, brindando alta escalabilidad, confiabilidad y estabilidad. Para el desarrollo de la aplicación se utilizará como lenguaje de programación Python 3.4.1 y como marcos de trabajo para el desarrollo de aplicaciones Web se seleccionaron a Boostrap en su versión 3.0 del lado del cliente y Django del lado del servidor, demostró el potencial de estas modernas tecnologías, a fin de logar la creación de una aplicación de alta calidad guiada por la metodología de desarrollo seleccionada.

Capítulo 2: Propuesta de Solución

El presente capítulo abordará las características de la propuesta de solución. Se desarrolla un modelo conceptual que agrupa los principales conceptos que se manifestarán.

2.1 Descripción de la propuesta de solución

Para dar solución a la problemática planteada se implementará un módulo para el Sistema Integral de Administración para la ASIC, que permitirá realizar los procesos de gestión de los informes e incidencias generados por los servicios que brinda la empresa, de manera que se evite la documentación cuantiosa que se acumula, así como la pérdida o el deterioro de la misma.

El diseño del sistema se realizará en la herramienta de modelado Visual Paradigm, la cual provee los diagramas de diseño necesarios para la comprensión del sistema que se desea desarrollar. Mediante estas opciones que brinda la herramienta será sencillo visualizar las relaciones existentes entre los conceptos fundamentales que deciden el flujo de trabajo en la agencia, así como las variables características de cada una de las entidades imprescindibles en la realización del módulo. De esta manera, quedarán definidas las relaciones entre los Servicios y los Contratos realizados en la agencia.

El módulo se realizará en el marco de trabajo Django, el cual cuenta con disímiles librerías que facilitan la labor del desarrollador, además de facilidades a la hora de implementar ya que es posible utilizar vistas genéricas que simplifican el código a su mínima expresión; estas vistas realizan tareas “sencillas” y comunes, muestran páginas de “listado” (ListView) y “detalle” (DetailView) para un solo objeto, entre otros.

De esta manera, se gestionarán cada uno de los documentos en el módulo a través de una interfaz amigable que propicie al entendimiento del sistema por todos los usuarios. Desde cualquier sucursal de la agencia se podrá acceder al sistema y gestionar la información requerida, ya que se contará con una única base de datos, además, se podrá ejecutar desde cualquier plataforma al tener la condición de aplicación web.

2.2 Modelo de dominio

“Es el artefacto más importante que se crea durante el análisis orientado a objetos, permite la representación visual de las clases conceptuales u objetos significativos de un dominio de interés. En la realización de este modelo se debe capturar las abstracciones e informaciones necesarias para entender el dominio en el

contexto de los requisitos actuales, permitiendo a las personas a comprender el negocio, sus conceptos, terminología y relaciones". [13]

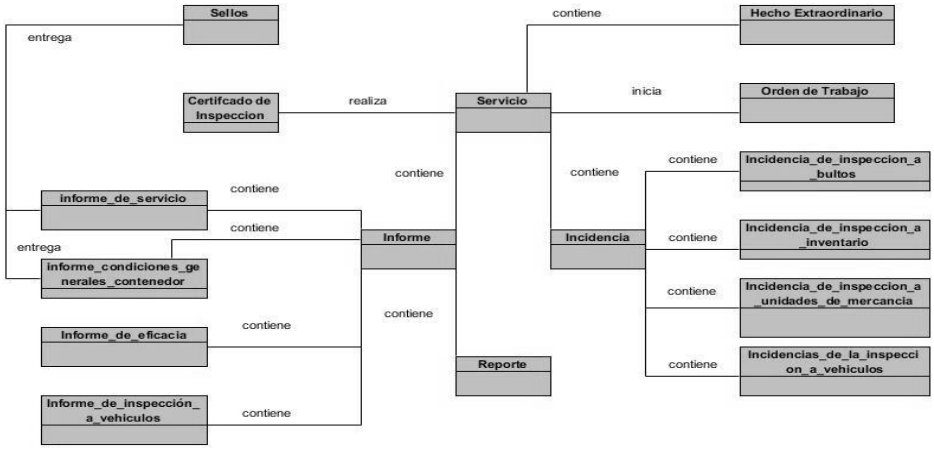


Figure 1 Modelo de dominio

2.2 Especificación de requisitos

2.3.1 Listado de requisitos funcionales:

Para definir las funcionalidades del sistema se tuvieron en cuenta los requerimientos recogidos en la ASIC para proceder con un adecuado funcionamiento del sistema y dar cumplimiento al objetivo planteado. A continuación, se expone la relación con las funcionalidades definidas:

RF01: Insertar informe de servicio.
RF02: Modificar informe de servicio.

RF03: Eliminar informe de servicio.
RF04: Visualizar informe de servicio.
RF05: Insertar incidencias de la inspección de bultos.
RF06: Modificar incidencias de la inspección de bultos.
RF07: Eliminar incidencias de la inspección de bultos.
RF08: Visualizar incidencias de la inspección de bultos.
RF09: Insertar incidencias de la inspección a unidades de mercancía.

RF10: Modificar incidencias de la inspección a unidades de mercancía.

RF11: Eliminar incidencias de la inspección a unidades de mercancía.

RF12: Visualizar incidencias de la inspección a unidades de mercancía.

RF13: Insertar incidencias de la inspección a inventario.

RF14: Modificar incidencias de la inspección a inventario.

RF15: Eliminar incidencias de la inspección a inventario.

RF16: Visualizar incidencias de la inspección a inventario.

RF17: Insertar informe de inspección a condiciones generales del contenedor.

RF18: Modificar informe de inspección a condiciones generales del contenedor.

RF19: Eliminar informe de inspección a condiciones generales del contenedor.

RF20: Visualizar informe de inspección a condiciones generales del contenedor.

RF21: Insertar informe de inspección a vehículos.

RF22: Modificar informe de inspección a vehículos.

RF23: Eliminar informe de inspección a vehículos.

RF24: Visualizar informe de inspección a vehículos.

RF25: Insertar incidencias de la inspección a vehículos.

RF26: Modificar incidencias de la inspección a vehículos.

RF27: Eliminar incidencias de la inspección a vehículos.

RF28: Visualizar incidencias de la inspección a vehículos.

RF29: Insertar hecho extraordinario.

RF30: Modificar hecho extraordinario.

RF31: Eliminar hecho extraordinario.

RF32: Visualizar hecho extraordinario.

RF33: Insertar certificado de inspección.

RF34: Modificar certificado de inspección.

RF35: Eliminar certificado de inspección.

RF36: Visualizar certificado de inspección.

RF37: Generar certificado de inspección.

RF38: Registrar entrega de sellos.

RF39: Generar informe de eficacia.

RF40: Generar reporte.

2.3.2 Listado de requisitos no funcionales

Los RNF, como su nombre lo sugiere, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de los datos, así como la protección, disponibilidad, entre otras propiedades emergentes. [26]

Atributo de calidad: Eficiencia

RNF01: El sistema debe permitir que las peticiones al servidor no excedan los cinco segundos.

Atributo de calidad: Confiabilidad

RNF02: El sistema debe mostrar mensajes de error al usuario ante cualquier error en el sistema, estos mensajes deben ser en español y en un lenguaje comprensible para el usuario.

Atributo de calidad: Usabilidad

RNF03: El sistema debe tener un diseño aceptable para los usuarios que accedan a ella, fácil de manejar por los usuarios con conocimientos mínimos en computación.

RNF04: El sistema debe poseer una interfaz acorde con la identidad de la ASIC.

Atributo de calidad: Portabilidad

RNF05: El sistema debe ser compatible con los navegadores web Chrome, Firefox, Internet Explore (Microsoft Edge en su versión más actualizada), Opera, Safari.

Atributo de calidad: Rendimiento

RNF06: El sistema debe procesar la información y responder a las búsquedas en un tiempo menor de 5 segundos.

RNF07: El sistema debe ser escalable, permitiendo incorporarle nuevas funcionalidades sin afectar las existentes.

Seguridad:

RNF08: El sistema ante cualquier error del sistema no se deben mostrar detalles de información que comprometan su integridad y seguridad.

RNF09: El sistema debe ser utilizado solo por los usuarios autorizados y el administrador.

RNF10: El sistema debe cerrar la sesión del usuario ante un tiempo de inactividad mayor a los treinta minutos.

Software:

RNF11: El sistema debe estar soportado por:

- Python en su versión 3.4.1 como lenguaje de programación.
- Django en su versión 1.8.5 como *framework* web.
- PostgreSQL 9.4.1 como base de datos.

Hardware:

RNF12: Para los servidores de aplicación y de base de datos se necesitará como mínimo CPU Core i3 Tercera generación 2.5 GHz con 16 GB de RAM.

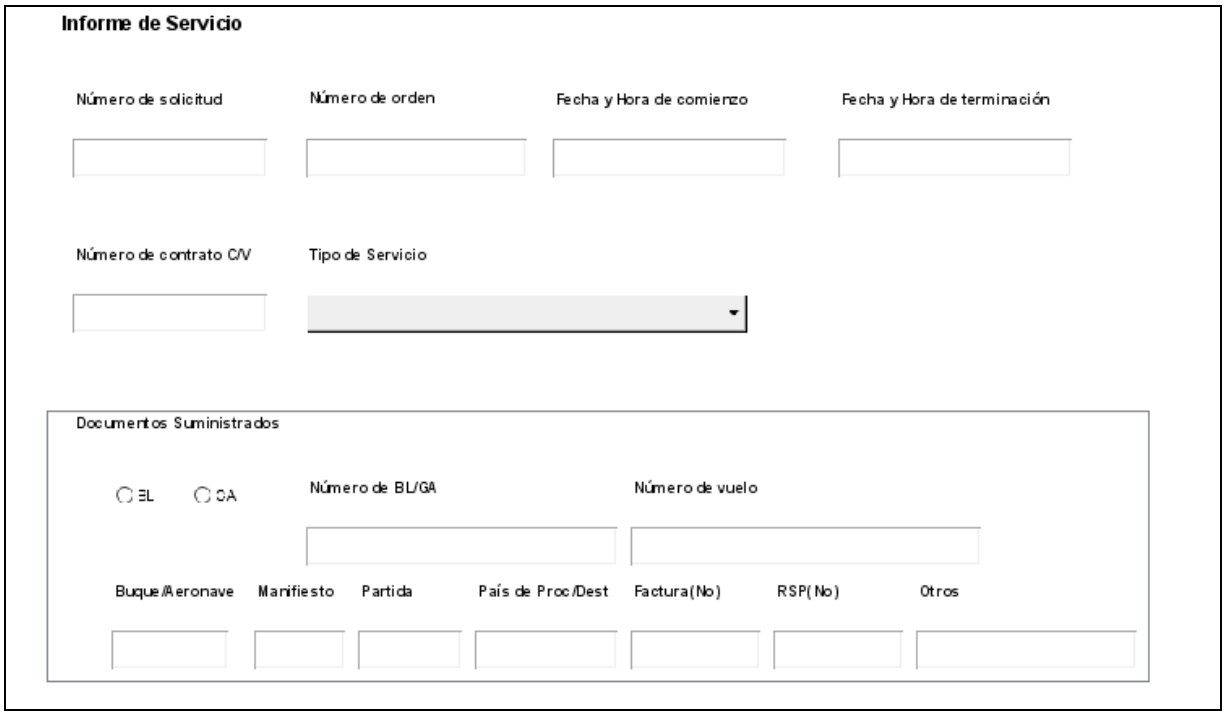
RNF13: Para el servidor de base de datos se necesitará como mínimo para una capacidad de almacenamiento de 5 GB y para el servidor de aplicaciones 10 GB.

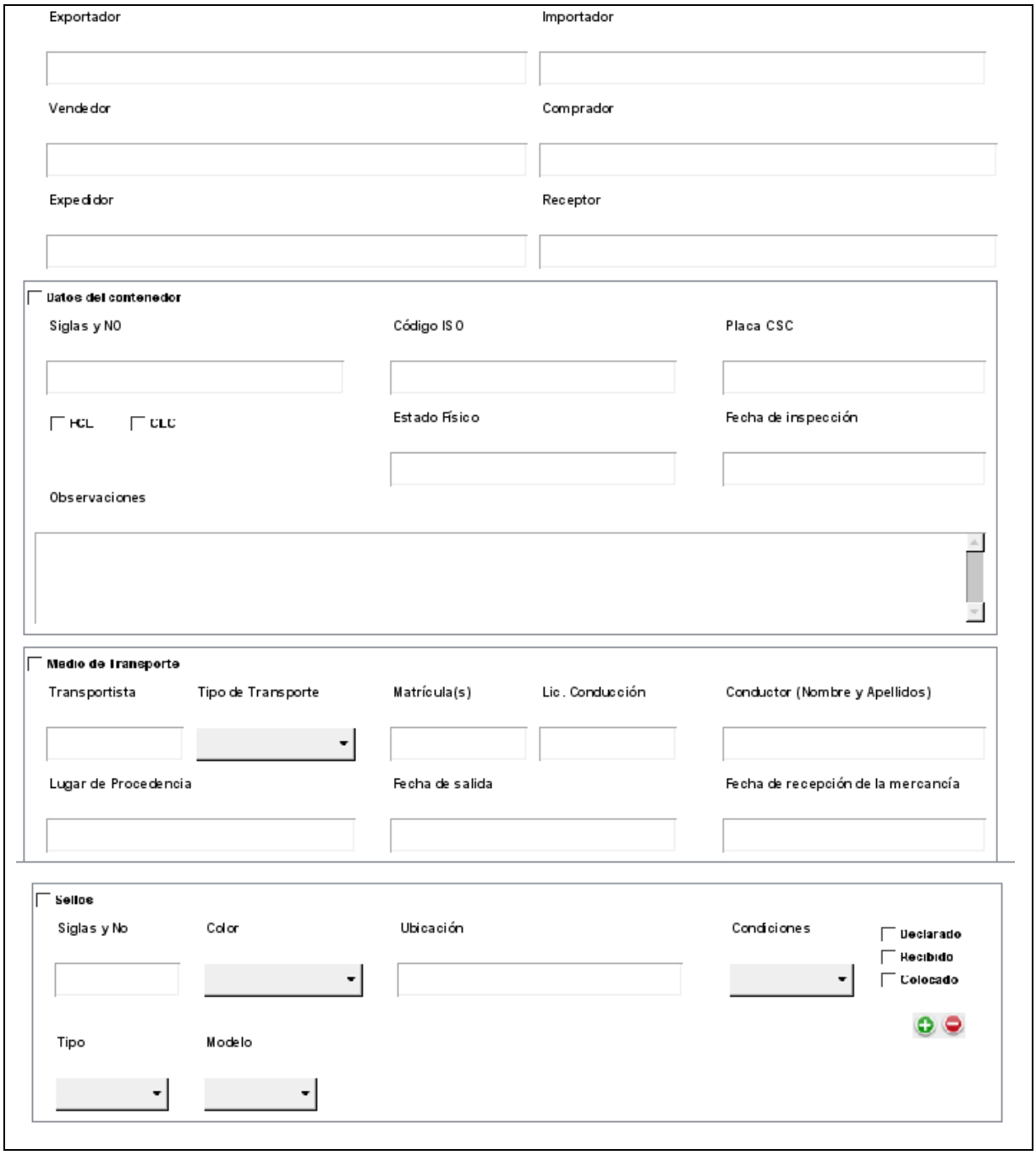
2.3.3 Historias de usuario

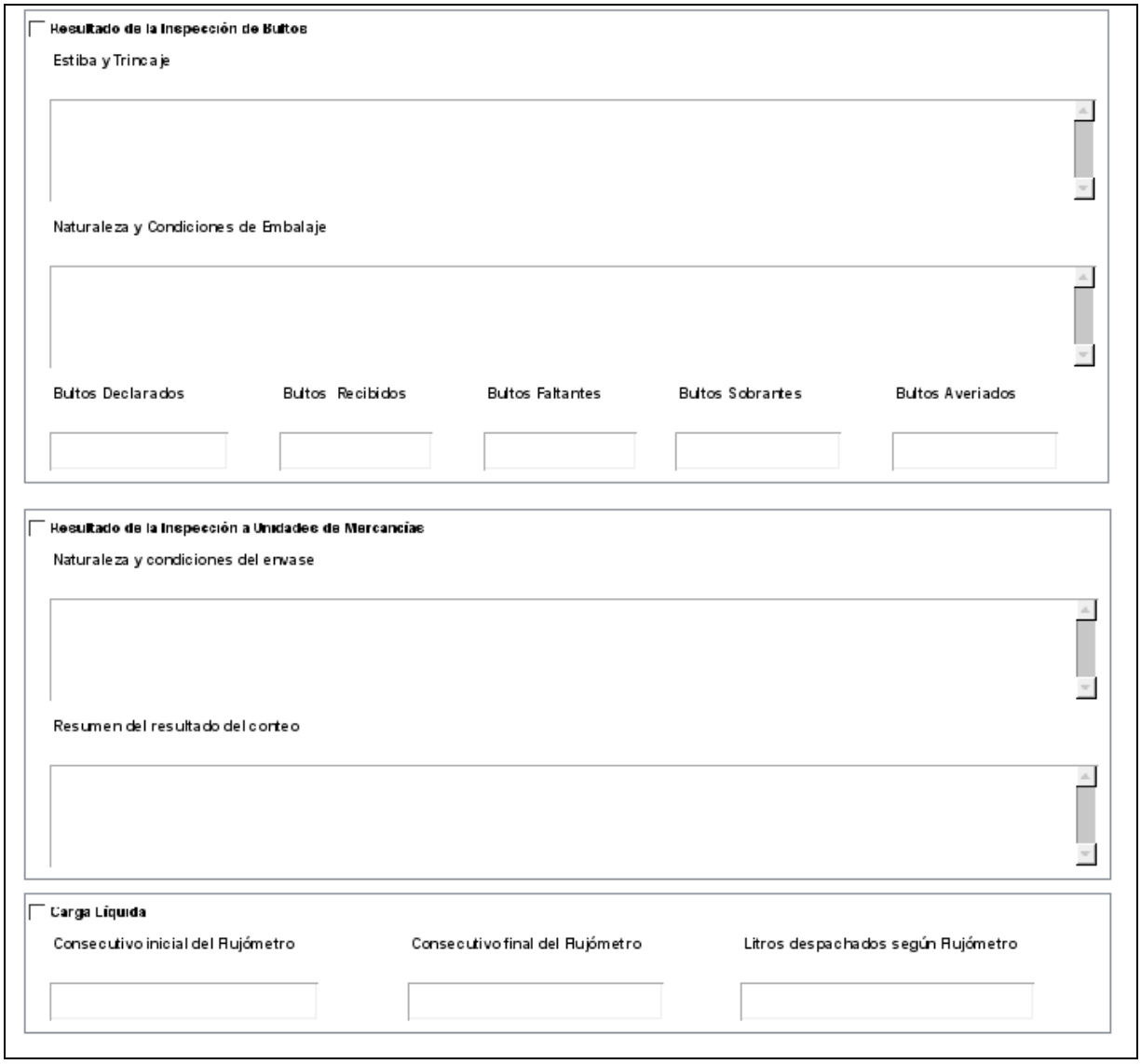
Las historias de usuario (en lo adelante HU) son descripciones cortas de una necesidad del cliente del software, definen lo que se debe construir en el proyecto de software. Tienen una prioridad asociada, definida por el cliente con el objetivo de indicar cuáles son las más importantes para el resultado final. Se encuentran divididas en tareas y su tiempo será estimado por los desarrolladores. La estimación de puntos es de acuerdo al tiempo que toma la tarea en ser desarrollada, donde un punto equivale a una semana ideal de programación. Las HU generalmente valen de 1 a 3 puntos. A continuación, se describen algunas de las historias de usuario de mayor prioridad, las restantes se encuentran descritas en el Anexo A:

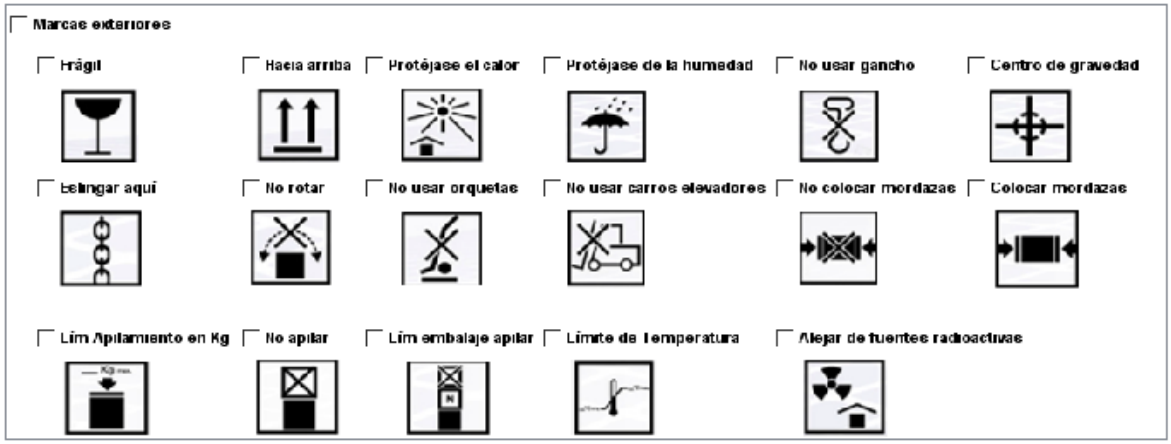
Tabla 1 HU Insertar informe de servicio

	Historia de usuario
Número: HU1	**Usuario:** Todos
Nombre de historia: Insertar informe de servicio	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 1
Programador responsable: Rosalba Rodríguez Pérez	
Descripción: El usuario presiona el botón de insertar informe de servicio, se despliega el formulario e inserta los datos, luego pulsa sobre el botón “Enviar”	
Observaciones: Observaciones: El sistema debe notificar al usuario cuando la operación termine.	
Prototipo de interfaz	


Informe de Servicio
Número de solicitud
Número de orden
Fecha y Hora de comienzo
Fecha y Hora de terminación
Número de contrato C/V
Tipo de Servicio
Documentos Suministrados
EL
CA
Número de BL/GA
Número de vuelo
Buque/Aeronave
Manifiesto
Partida
País de Proc/Dest
Factura(No)
RSP(No)
Otros


Exportador
Importador
Vendedor
Comprador
Expedidor
Receptor
Datos del contenedor
Siglas y NO
Código ISO
Placa CSC
HCL
CLC
Estado Físico
Fecha de inspección
Observaciones
Medio de Transporte
Transportista
Tipo de Transporte
Matrícula(s)
Lic. Conducción
Conductor (Nombre y Apellidos)
Lugar de Procedencia
Fecha de salida
Fecha de recepción de la mercancía
Sellos
Siglas y No
Color
Ubicación
Condiciones
Declarado
Recibido
Colocado
Tipo
Modelo

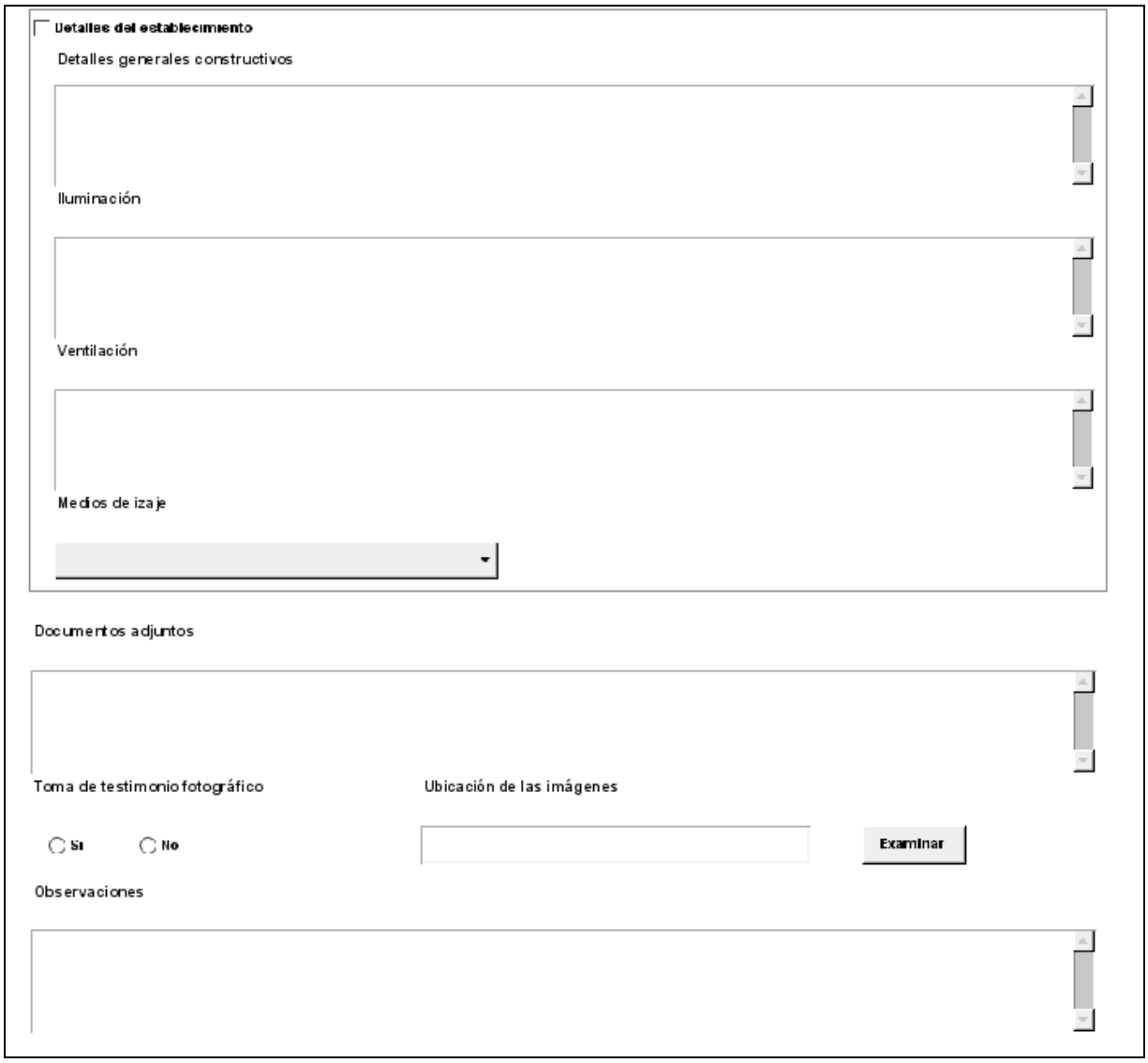

Resultado de la Inspección de Bultos
Estiba y Trincaje
Naturaleza y Condiciones de Embalaje
Bultos Declarados
Bultos Recibidos
Bultos Faltantes
Bultos Sobrantes
Bultos Averiados
Resultado de la Inspección a Unidades de Mercancías
Naturaleza y condiciones del envase
Resumen del resultado del conteo
Carga Líquida
Consecutivo inicial del Flujómetro
Consecutivo final del Flujómetro
Litros despachados según Flujómetro



Descripción de la cantidad de bultos en buen estado y con su contenido completo

Descripción de los faltantes y los sobrantes y las evidencias que lo sustentan

Descripción de las averías, detallando naturaleza, causas, alcance y evidencias que lo sustentan


Detalles del establecimiento
Detalles generales constructivos
Iluminación
Ventilación
Medios de izaje
Documentos adjuntos
Toma de testimonio fotográfico
Ubicación de las imágenes
Si
No
Examinar
Observaciones

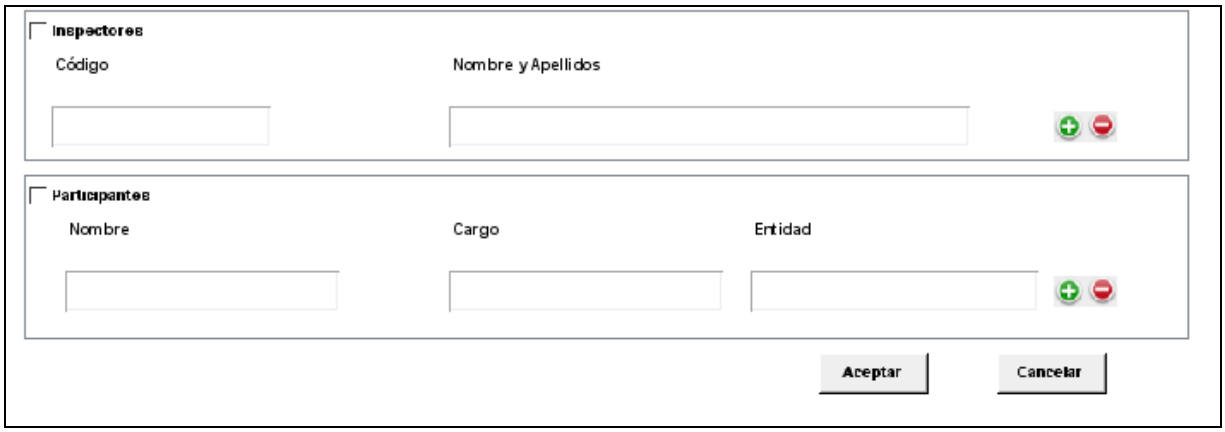
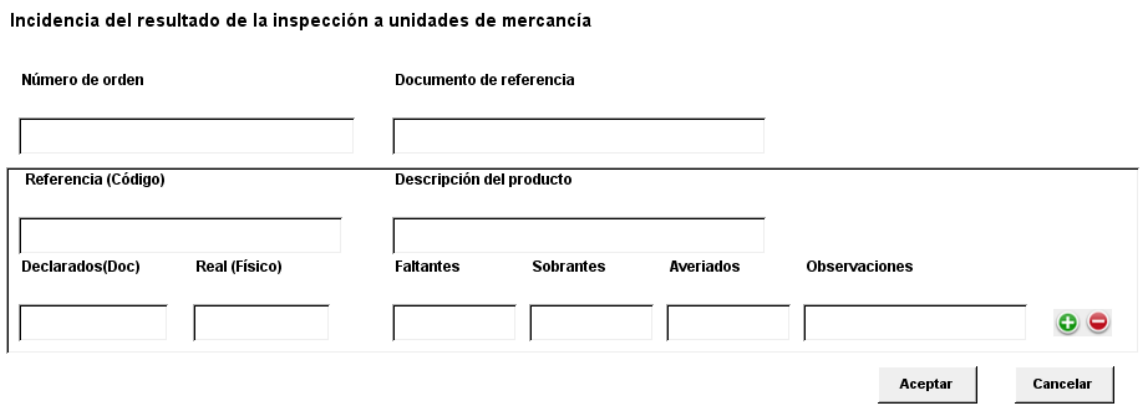


Tabla 2 HU Insertar incidencias de la inspección de bultos

Historia de usuario	
Número: HU2	**Usuario:** Todos
Nombre de historia: Insertar incidencias de la inspección de bultos	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 1
Programador responsable: Rosalba Rodríguez Pérez	
Descripción: El usuario presiona el botón de insertar incidencia, se despliega el formulario e inserta los datos, luego pulsa sobre el botón “Enviar”.	
Observaciones: Observaciones: El sistema debe notificar al usuario cuando la operación termine.	
Prototipo de interfaz 	

Tabla 3 HU Insertar incidencia de la inspección a unidades de mercancía

	Historia de usuario
Número: HU3	**Usuario:** Todos
Nombre de historia: Insertar incidencia de la inspección a unidades de mercancía	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 1
Programador responsable: Rosalba Rodríguez Pérez	
Descripción: El usuario presiona el botón de insertar incidencia, se despliega el formulario e inserta los datos, luego pulsa sobre el botón “Enviar”.	
Observaciones: Observaciones: El sistema debe notificar al usuario cuando la operación termine.	
Prototipo de interfaz 	

2.4 Arquitectura del Software

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. [27]

2.4.1 Patrón Modelo-Vista-Plantilla

Django fue diseñado para promover el acoplamiento débil y la estricta separación entre las piezas de una aplicación. Si se sigue esta filosofía, es fácil hacer cambios en un lugar particular de la aplicación sin afectar otras piezas. En las funciones de vista, por ejemplo, se discute la importancia de separar la lógica de negocios de la lógica de presentación usando un sistema de plantillas. Con la capa de la base de datos, se

aplica esa misma filosofía para el acceso lógico a los datos. Estas tres piezas juntas, la lógica de acceso a la base de datos, la lógica de negocios, y la lógica de presentación, comprenden un concepto que a veces es llamado el patrón de arquitectura de software MVC. En este patrón, el "Modelo" hace referencia al acceso a la capa de datos, la "Vista" se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el "Controlador" implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario. [28]

Django sigue el patrón MVC tan al pie de la letra que puede ser llamado un *framework* MVC. Someramente, la M, V y C se separan en Django de la siguiente manera:

- *M*, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django.
- *V*, la porción que selecciona qué datos mostrar y cómo mostrarlos, es manejada por la vista y las plantillas.
- *C*, la porción que delega a la vista dependiendo de la entrada del usuario, es manejada por el *framework* mismo siguiendo la URLconf y llamando a la función apropiada de Python para la URL[15] obtenida. [28]

Debido a que la "C" es manejada por el mismo *framework* y la parte más emocionante se produce en los modelos, las plantillas y las vistas, Django es conocido como un *Framework* MTV. En el patrón de diseño MTV,

- M significa "*Model*" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T significa "*Template*" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

[15] *Uniform Resource Locator*

- V significa "*View*" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.

A continuación, se muestra un ejemplo del patrón en la implementación del sistema:

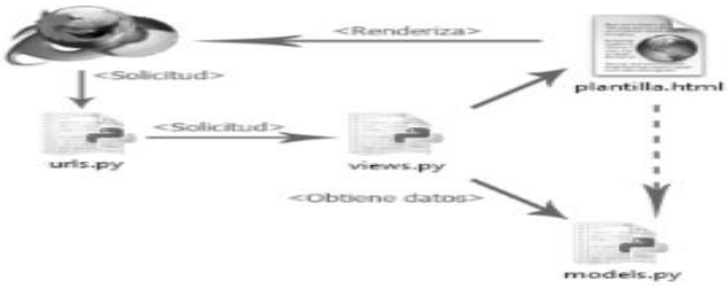


Figure 2 Ejemplo de patrón MTV

El módulo a desarrollar se comportará de manera similar a la observada en la Figura 2, cada uno de sus paquetes contará con un fichero models.py donde serán definidos todos los modelos, views.py que es donde se definirán los controladores y una serie de archivos HTML, agrupados en el fichero templates.py, que representarán las vistas. Por ejemplo, para modelar la funcionalidad Modificar informe de inspección a vehículos (ver Anexo C) se crearían en la herramienta de modelado tres paquetes correspondientes a las plantillas, las vistas y el modelo, respectivamente. El paquete de las vistas, estas van a atender la solicitud del navegador, que no son más que las páginas en el paquete Plantilla; a su vez interactúan con el modelo para obtener los datos y luego devolverlos a la plantilla para que se visualice en el navegador.

2.4.2 Diagramas de clases del diseño

El diagrama de clases del diseño, permite describir la estructura de la solución mostrando sus clases, atributos y las relaciones entre ellos; en él se evidencia la arquitectura de software seleccionada en el proyecto. Las figuras mostradas posteriormente exponen el patrón MTV a través de las relaciones que existen entre sus paquetes.

La figura que se muestra a continuación corresponde al diagrama de clases del diseño de la HU Insertar informe de servicio, donde se expone la arquitectura seleccionada aplicada a la entidad Informe de Servicio. En este sentido, se muestran las relaciones entre los tres componentes: modelo, vista y plantilla, en la carpeta que contiene a las plantillas se encuentran la página principal, la página del informe de servicio y la página del formulario de informe de servicio, cada una de ellas asociadas a la vista donde se relacionan para realizar las funciones de la clase Informe de Servicio que lleva los cambios efectuados al modelo para que persistan en la base de datos. De esta manera se procederá para cada una de las clases definidas.

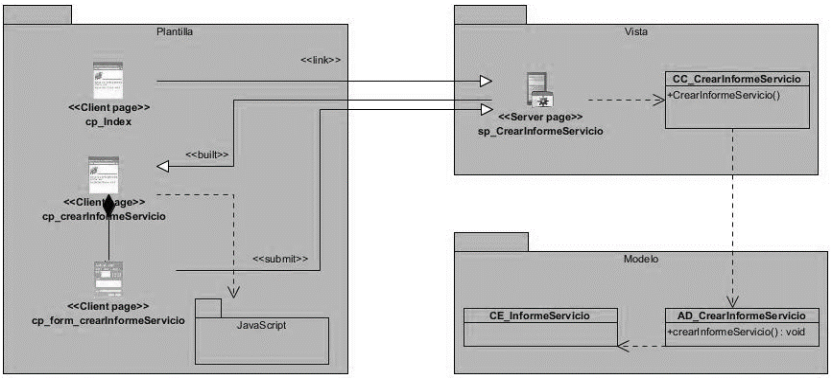


Figure 3 Diagrama de clases del diseño de la HU InsertarInformeServicio

El diagrama de clases del diseño de la HU Modificar informe de servicio se muestra seguidamente exponiendo las relaciones existentes entre los componentes: modelo, vista y plantilla; estas relaciones no varían con respecto a las presentes en la HU Insertar informe de servicio:

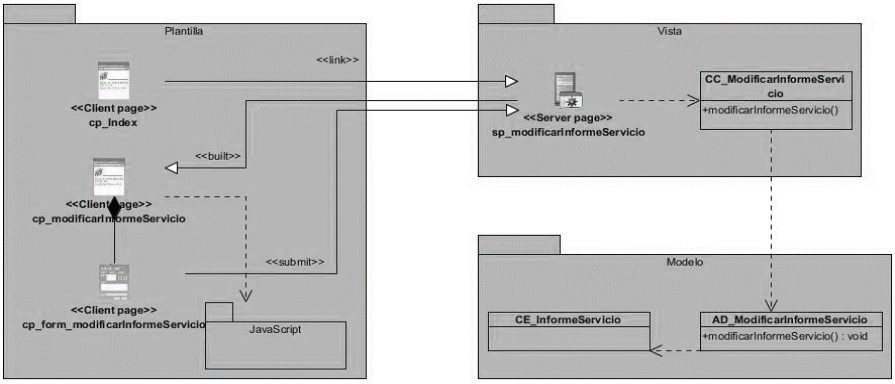


Figure 4 Diagrama de clases del diseño de la HU Modificar Informe Servicio

Sin embargo, el diagrama de clases del diseño correspondiente a la HU Generar Reporte no se desenvuelve como en las figurar anteriores, ya que al solo necesitar mostrar los datos ya en la base de datos solo pide información de la base de datos y la devuelve a través de la página por la cual responde:

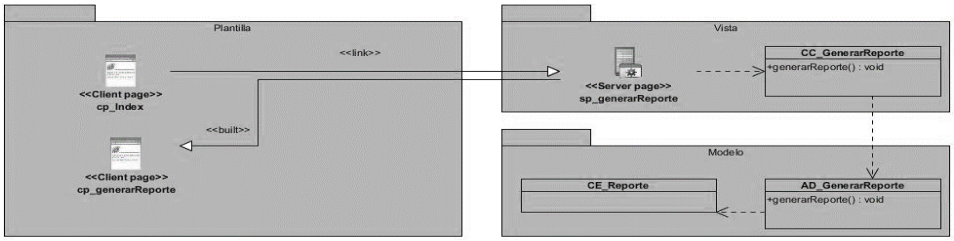


Figure 5 Diagrama de clases del diseño de la HU Generar Reporte

2.4.3 Patrones de diseño

Patrones GRASP (*Responsability Assignment Software Patterns*): Según Grosso [29], son patrones basados en la asignación de responsabilidades a objetos. Es una buena práctica para el desarrollo eficaz de la POO[16], a continuación se muestran los utilizados en esta investigación:

- **Experto en información:** consiste en asignar una responsabilidad al experto en información, es decir, a la clase que contiene toda la información necesaria para desempeñar una responsabilidad [30].

  Ejemplo de ello es en la clase InformeServicioCreate (ver figura 3), la cual es la encargada de crear informes de servicios con la utilización de la clase Informe_Servicio creada en el modelo.

```
class InformeServicioCreate(CreateView):
    model = Informe_Servicio
    template_name = 'Contenido/nuevo_informe_servicio.html'
    form_class = InformeServicioForm
    success_url = reverse_lazy('solicitud_listar')
```

Figure 6 Ejemplo de aplicación del patrón Experto

[16] *Programación Orientada a Objetos*

- **Creador:** consiste en asignar a una determinada clase B la responsabilidad de crear una instancia de la clase A al ocurrir alguna de las siguientes circunstancias: B agrega a A, B tiene los datos de inicialización de A, B registra a A o B utiliza estrechamente a A. [30]

  Este patrón es utilizado en CE_HechoExtraordinario ya que, para crear un hecho extraordinario se utiliza una instancia de la clase Cliente

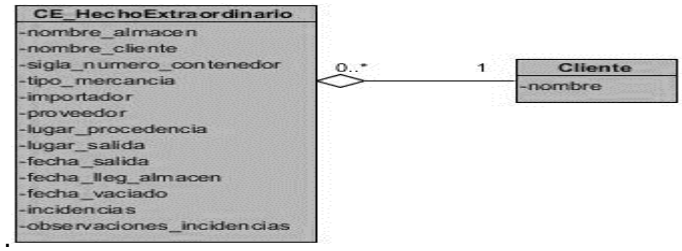


Figure 7 Clase controladora Insertar hecho extraordinario

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

```
def inicio(request):
    return render_to_response('home.html', RequestContext(request))

def bloqueo(request):
    return render_to_response('Seguridad/bloqueo.html', locals(), RequestContext(request))
```

Figure 8 Ejemplo del patrón Alta cohesión

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

  Este patrón se evidencia en la CC_GestionarInforme debido a que se encarga de recibir los datos del usuario y la que los envía a las distintas clases según el método al que se llame.

Figure 9 Función Insertar informe de servicio

Bajo acoplamiento: plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas [31] (Figura 2.6). Django tiene incluido este patrón ya que este *framework* permite un bajo acoplamiento entre las piezas evitando de esta manera las dependencias. Esto se pone de manifiesto cuando se le realizan cambios en las configuraciones de las URL, plantillas HTML, en la Base de Datos, etc., es suficiente realizarlo solo una vez.

```
url(r'^inf_vehic/listar$', login_required(InformeInspVehiculosList.as_view()),
    name='informe_insp_vehiculos_listar'),
url(r'^inf_vehic/nueva$', login_required(InformeInspVehiculosCreate.as_view()),
    name='informe_insp_vehiculos_crear'),
url(r'^inf_vehic/editar/(?P<pk>\d+)$', login_required(InformeInspVehiculosUpdate.as_view()),
    name='informe_insp_vehiculos_editar'),
url(r'^inf_vehic/eliminar/(?P<pk>\d+)$', login_required(InformeInspVehiculosDelete.as_view()),
    name='informe_insp_vehiculos_eliminar'),
```

Figure 10 Ejemplo de patrón Bajo acoplamiento

Patrones GoF (*Gang of Four*): Representan patrones que representan soluciones técnicas basadas en POO que favorecen la reutilización del código, forma parte de esta definición el patrón a continuación:

- **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. [25] Como ejemplo de este patrón se evidencia la utilización de los decoradores de Django (@login_required, @permission_required, etc.).

```
url(r'^inf_vehic/listar$', login_required(InformeInspVehiculosList.as_view()),
    name='informe_insp_vehiculos_listar'),
url(r'^inf_vehic/nueva$', login_required(InformeInspVehiculosCreate.as_view()),
    name='informe_insp_vehiculos_crear'),
url(r'^inf_vehic/editar/(?P<pk>\d+)$', login_required(InformeInspVehiculosUpdate.as_view()),
    name='informe_insp_vehiculos_editar'),
url(r'^inf_vehic/eliminar/(?P<pk>\d+)$', login_required(InformeInspVehiculosDelete.as_view()),
    name='informe_insp_vehiculos_eliminar'),
```

Figure 11 Ejemplo del patrón Decorador

2.4.4 Diagrama de clases del sistema

El diagrama de clases del sistema, como bien sugiere el nombre, muestra las relaciones entre las clases del sistema, además de la multiplicidad entre ellas para mayor comprensión del proyecto en desarrollo. En sí, guarda relación con el diagrama entidad-relación, ya que este expone todos los atributos, sus propiedades y las relaciones entre las tablas que persisten en la base de datos. La diferencia entre uno y el otro, se puede decir que es que el diagrama de clases relacional solo tiene intención de mostrar la relación entre las clases y la multiplicidad en estas, obviando las propiedades que puedan tener los atributos que contiene.

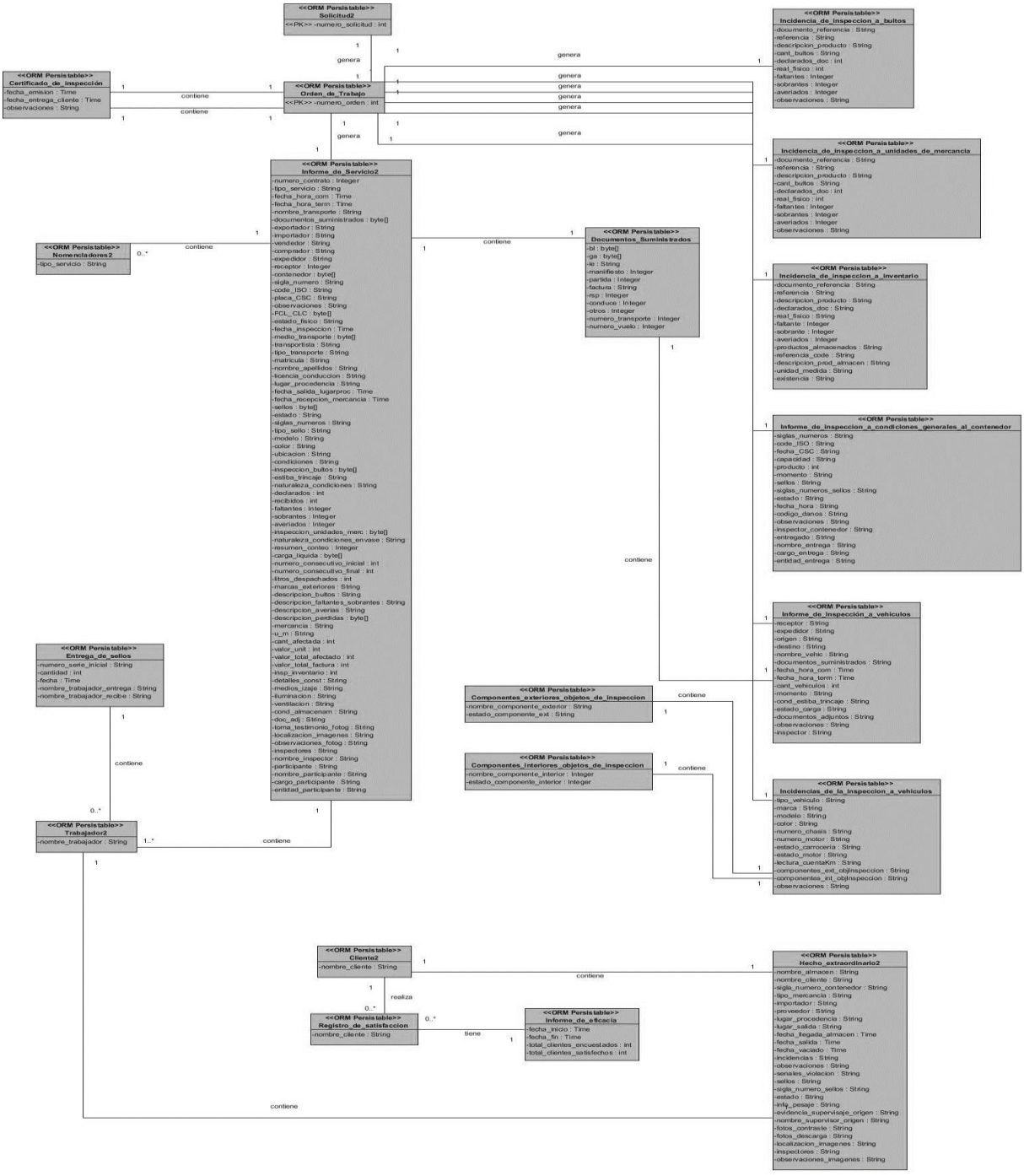


Figure 12 Diagrama de clases

2.4.5 Diagrama entidad-relación

Para el modelado de datos, se propone el diagrama entidad-relación, que refleja la interrelación entre las tablas de la base de datos, así como sus propiedades.

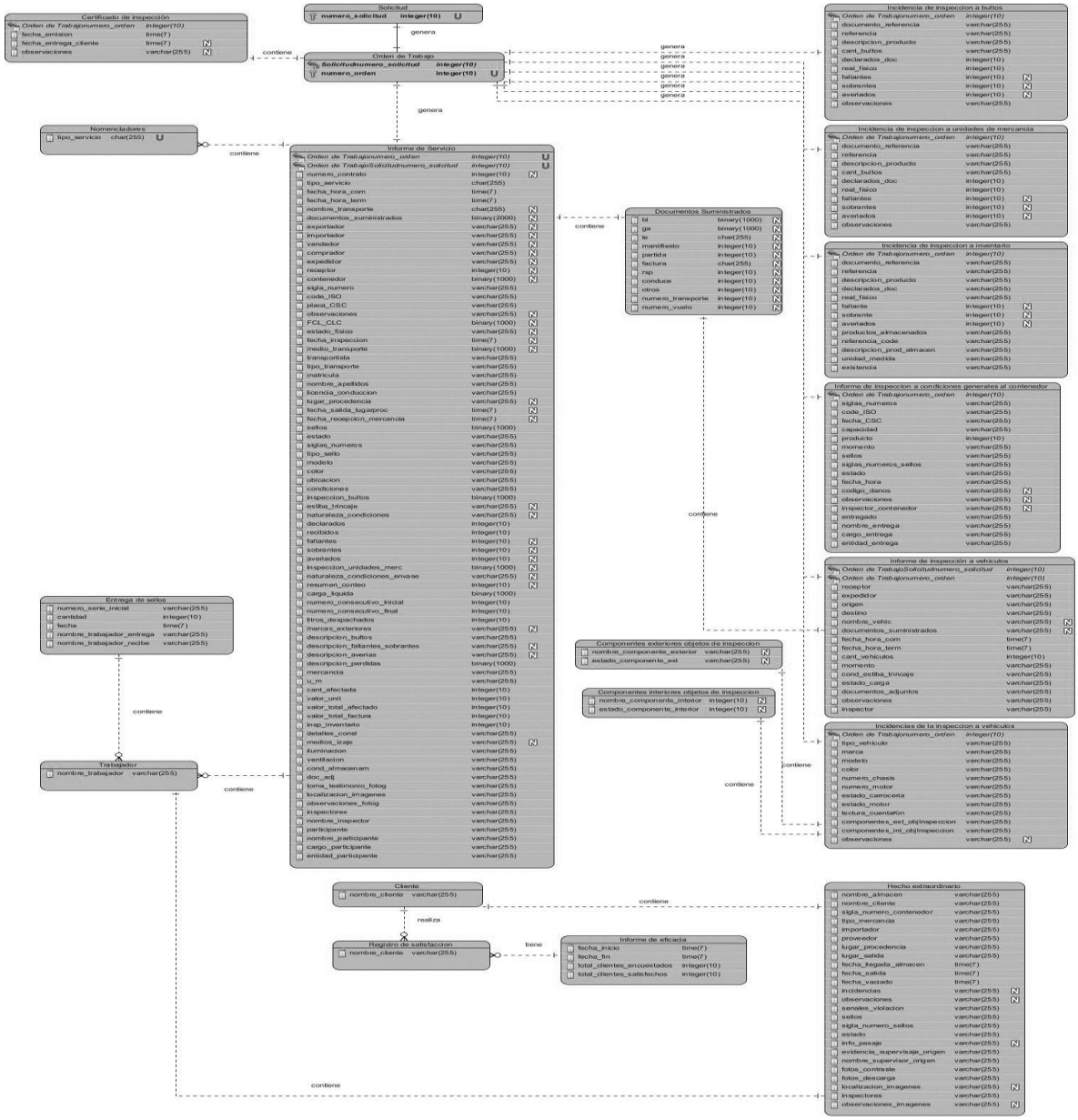


Figure 13 Diagrama entidad-relación

2.5 Conclusiones parciales

Las historias de usuarios generadas a partir de la especificación de requisitos recogidos en el levantamiento de información realizado a la ASIC, así como la realización de los artefactos definidos durante el proceso de análisis permitieron diseñar una propuesta de solución que incluye una serie de patrones GRASP y GoF a realizar en la aplicación para lograr una buena reutilización del código en el futuro.

Capítulo 3: Implementación y Prueba

En este capítulo se plasma como objetivo materializar la solución propuesta y cumplir con los requisitos obtenidos al inicio de la investigación. La fase de implementación y prueba recoge los estándares de codificación que deben ser empleados en el desarrollo del entorno colaborativo, los diagramas de componentes y despliegue donde se observan las dependencias lógicas que existen entre los componentes del software y los nodos necesarios para la puesta en marcha del sistema. Al culminar se muestran las pruebas realizadas para validar el correcto funcionamiento de la propuesta de solución.

3.1 Estándar de codificación

Indentación:

- Las líneas de continuación han de alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).
- Es necesario el uso de una indentación de una tabulación para cada línea exceptuando la primera.
- La indentación se ha de realizar solamente con tabulaciones, no deben utilizarse nunca los cuatro espacios.

Máxima longitud de líneas:

- Todas las líneas deben estar limitadas a un máximo de setenta y nueve caracteres.
- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas.
- En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas.

Líneas en blanco:

- Separar las funciones de alto nivel y definiciones de clases con dos líneas en blanco.
- Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco.
- Se puede utilizar líneas en blanco escasamente para separar secciones lógicas.

Codificaciones:

- Utilizar la codificación UTF-8.

- Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”.

Importaciones:

- Las importaciones deben estar en líneas separadas.
- Siempre deben colocarse al comienzo del archivo.
- Deben quedar agrupadas de la siguiente forma:
  - Importaciones de la librería estándar.
  - Importaciones terceras relacionadas.
  - Importaciones locales de la aplicación / librerías.
- Cada grupo de importaciones debe estar separado por una línea en blanco.

Espacios en blanco en expresiones y sentencias:

- 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.
- Deben rodearse con exactamente un espacio los siguientes operadores binarios:
  - Asignación (=).
  - Asignación de aumentación (+=, -=, etc.).
  - Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, is not).
  - Expresiones lógicas (and, or, not).
- Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.
- No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

Comentarios:

- Los comentarios deben ser oraciones completas.
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.
- Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc.
- Si un comentario es corto el punto final puede omitirse.

Comentarios en bloque:

- Deben estar indentados al mismo nivel que el código a comentar.
- Cada línea de un comentario en bloque comienza con un numeral (#) y un espacio en blanco.

Comentarios en la misma línea:

- Se recomienda utilizarlos escasamente.
- Se debe definir comenzando por un numeral (#) seguido de un espacio en blanco.
- Deben ubicarse en la misma línea que se desea comentar.

Cadenas de documentación:

- Deben quedar documentados todos los módulos, funciones, clases y métodos públicos.
- Para definir una cadena de documentación debe quedar encerrada dentro de (“””).
- Los (“””) que finalizan una cadena de documentación deben quedar en una línea a no ser que la cadena sea de una sola línea.

Convenciones de nombramiento:

- Nunca se deben utilizar como simple caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0).
- Los módulos deben tener un nombre corto y en minúscula.
- Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas).
- Los nombres de las excepciones deben estar escrito también en la convención “CapWords” utilizando el sufijo “Error”.

- Los nombres de las funciones deben estar escrito en minúscula separando las palabras con un guión bajo "_".
- Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (_). [32]

3.2 Modelos de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo éstos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. La implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes.

Los artefactos principales generados en esta etapa, son el diagrama de despliegue y el de componentes, ambos conforman lo que se conoce como el modelo de implementación, donde se describen los componentes, su organización y dependencias entre los nodos físicos en los que funcionará la aplicación. [33]

3.2.1 Diagrama de componentes

Representa la estructura física de implementación; un componente es una unidad modular que puede reemplazarse en su propio entorno (ejemplo: librería, ejecutable, dll, documento). Sus elementos internos quedan ocultos, pero tiene una o varias interfaces proporcionadas bien definidas a través de las cuales se puede obtener acceso a sus funciones. Un componente también puede tener interfaces necesarias. En una interfaz necesaria, se definen las funciones o servicios de otros componentes que son necesarios. Mediante la conexión de las interfaces proporcionadas y las interfaces necesarias de distintos componentes, puede construirse un componente mayor. Un sistema de software completo se puede concebir como un componente.

Un diagrama de componentes es utilizado para organizar el modelo de implementación, mostrando la organización y las dependencias entre un conjunto de componentes. Es capaz de concebir el diseño atendiendo a los bloques principales ayudando de esa manera al equipo de desarrollo a entender un diseño existente y a crear uno nuevo. Al pensar en el sistema como una colección de componentes con interfaces proporcionadas y necesarias bien definidas, se mejora la separación entre los componentes, facilitando la comprensión y los cambios cuando se modifican los requisitos.

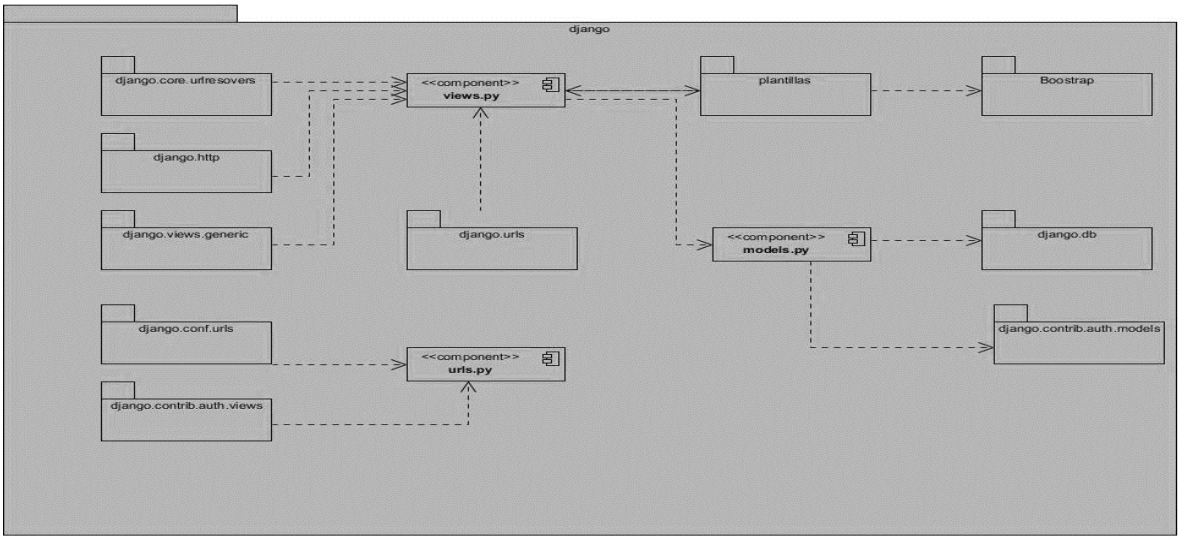


Figure 14 Diagrama de componentes

El componente models.py implementa cada una de las clases del módulo con sus respectivos atributos y funciones, utiliza la librería django.db, la cual importa la funcionalidad models que permite llamar a la característica Model para generar la base de datos a partir del componente. Un ejemplo de este componente se presenta en la siguiente figura:

```
from django.db import models
from django.contrib.auth.models import User
from ASIC.models import TipoNomencladores,User,Provincia,Municipio,Nomencladores,Contacto_Cliente,Cliente,\
    Trabajador,Contrato,Solicitud,Orden_Trabajo

class Documentos_Suministrados(models.Model):
    pk_doc = models.AutoField(primary_key=True)
    bl = models.BooleanField
    ga = models.BooleanField
    le = models.TextField(default='')
    manifiesto = models.IntegerField(default='')
    partida = models.IntegerField(default='')
    factura = models.TextField(default='')
    rsp = models.TextField(default='')
    conduce = models.TextField(default='')
    otros = models.TextField(default='')
    num_transp = models.TextField(default='')
    num_vuelo = models.TextField(default='')

    def __unicode__(self):
        return self.pk_doc
```

Figure 15 Fragmento del código del componente models.py

Luego de crear las clases en el componente models.py se crea forms.py, fichero que va a recoger todos los atributos declarados en el anterior y a los cuales se le agregan funcionalidades, es decir, el comportamiento que van adoptar en la aplicación. Por ejemplo, al atributo nombre_almacen es necesario insertarle el nombre del almacén por lo que ese campo en el HTML se va a comportar como un TextField (): (campo de texto).

Serán creados cada uno de los archivos HTML necesarios para las funcionalidades de las clases creadas en el modelo y se van guardar en templates.py. Por ejemplo, a clase Informe_Servicio tiene cuatro operaciones: insertar, editar, visualizar y eliminar, para cada una de estas se crea un HTML diferente, es preciso de esta manera ya que van a ser llamadas por el componente urls.py indistintamente. La figura siguiente muestra el archivo nuevo_informe_servicio.html:

```
{% extends 'plantilla.html' %}

{% block recoger %}page-sidebar-closed{% endblock %}

{% block a %}
    <li xmlns="http://www.w3.org/1999/html">
        <a href="/inicio">Inicio</a>
        <i class="fa fa-circle"></i>
    </li>
    <li>
        <a href="#">Nuevo Informe de Servicio</a>
    </li>
{% endblock %}

{% block Titulo_Pagina %}
    Nuevo Informe de Servicio
{% endblock %}

{% block css %}
    {% load staticfiles %}
    <link rel="stylesheet" type="text/css" href="{% static 'assets/global/plugins/clockface/css/clockface.css' %}"
          xmlns="http://www.w3.org/1999/html"/>
    <link rel="stylesheet" type="text/css"
          href="{% static 'assets/global/plugins/bootstrap-datepicker/css/bootstrap-datepicker3.min.css' %}"/>
    <link rel="stylesheet" type="text/css"
          href="{% static 'assets/global/plugins/bootstrap-timepicker/css/bootstrap-timepicker.min.css' %}"/>
    <link rel="stylesheet" type="text/css"
          href="{% static 'assets/global/plugins/bootstrap-colorpicker/css/colorpicker.css' %}"/>
```

Figure 16 Fragmento de nuevo_informe_servicio.html

Cada archivo HTML recibe la ruta que le corresponde mediante una URL, esto se puede ver para cada clase, a continuación, en la siguiente figura se observa este comportamiento en la clase Informe_Servicio:

```
urlpatterns = [
    #------------Informe de Servicio-------------#
    url(r'^informe/listar$', login_required(InformeServicioList.as_view()), name='informe_listar'),
    url(r'^informe/nueva$', login_required(InformeServicioCreate.as_view()), name='informe_crear'),
    url(r'^informe/editar/(?P<pk>\d+)$', login_required(InformeServicioUpdate.as_view()), name='informe_editar'),
    url(r'^informe/eliminar/(?P<pk>\d+)$', login_required(InformeServicioDelete.as_view()), name='informe_eliminar'),
    url(r'^informe/visualizar/(?P<pk>\d+)$', login_required(InformeServicioDetail.as_view()),
        name='informe_servicio_detail'),
```

Figure 17 URLs de la clase Informe_Servicio

A partir de ello, ya es posible dar paso a la definición de las clases en el componente views.py. De acuerdo con lo anterior, para la realización del módulo se utilizarán las CBV dada su sencilla implementación y comprensión, al poder realizar las operaciones más comunes en una aplicación web sin tener que implementar un extenso código como en las FBV. Las vistas genéricas llaman dentro de sí al modelo (model), al formulario (form_class), a la plantilla (template_name), además de success_url que indica la URL hacia la que va a redireccionar.

```
class InformeInspCondicGenContenedorCreate(CreateView):
    model = Informe_condic_generales_contenedor
    template_name = 'Contenido/nuevo_informe_insp_condic_gen_contenedor.html'
    form_class = InformeInspCondicGenContenedorForm
    success_url = reverse_lazy('informe_insp_cond_gen_contenedor_listar')


class InformeInspCondicGenContenedorUpdate(UpdateView):
    model = Informe_condic_generales_contenedor
    form_class = InformeInspCondicGenContenedorForm
    template_name = 'Contenido/nuevo_informe_insp_condic_gen_contenedor.html'
    success_url = reverse_lazy('informe_insp_cond_gen_contenedor_listar')

class InformeInspCondicGenContenedorDetail(DetailView):
    model = Informe_condic_generales_contenedor
    template_name = 'Contenido/visualizar_contenedores.html'
    success_url = reverse_lazy('informe_insp_cond_gen_contenedor_listar')


class InformeInspCondicGenContenedorDelete(DeleteView):
    model = Informe_condic_generales_contenedor
    template_name = 'Contenido/eliminar_informe_insp_condic_gen_contenedor.html'
    success_url = reverse_lazy('informe_insp_cond_gen_contenedor_listar')
```

Figure 18 Vistas genéricas de la clase Informe_condic_generales_contenedor

3.2.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el despliegue de los componentes sobre dichos nodos. Es el complemento del diagrama de componente, ya que unidos, proveen la vista de implementación del sistema. Describe la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Representa a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP.

Este diagrama muestra la configuración del funcionamiento del sistema incluyendo su software y su hardware. Para cada componente de un diagrama se deben documentar las características técnicas requeridas, el tráfico de la red y el tiempo de respuesta.

En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

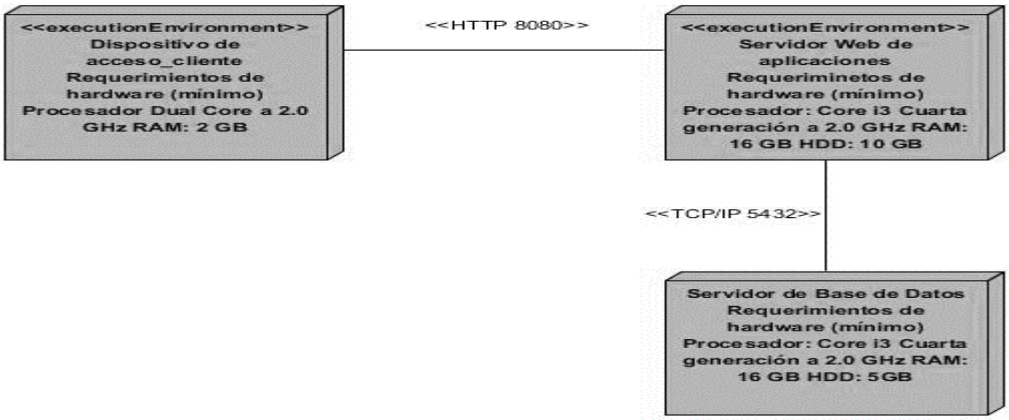


Figure 19 Diagrama de despliegue

3.2.3 Descripción de los nodos

Nodo PC Cliente: representa una computadora desde la cual el usuario podrá acceder a la aplicación.

Nodo Servidor WEB: representa una estación donde estará instalada el servidor Apache sobre el cual correrá la aplicación a la cual accederán los usuarios a través de la máquina cliente.

Nodo Servidor de Base de Datos: representa el servidor donde estará el sistema gestor de base de datos que dará respuesta a las peticiones hechas por la aplicación.

3.3 Pruebas de software

Previo a la utilización de cualquier software este debe someterse a una serie de pruebas que validarán su desempeño al verificar si cumplen con las funcionalidades requeridas. Cumpliendo de esta manera con el primer principio definido por Davis [Dav95b]: *Todas las pruebas deben poder rastrearse hasta los requerimientos del cliente.* El objetivo de las pruebas de software es descubrir errores. Entonces, los defectos más severos (desde el punto de vista del cliente) son aquellos que hacen que el programa no cumpla sus requerimientos. [34]

3.3.1 Selección de las pruebas

Se puede considerar el proceso de pruebas funcionales como un proceso donde se va probando inicialmente lo de más bajo nivel y se van integrando y probando paulatinamente componentes hasta lograr

un sistema completo totalmente probado. De ahí la existencia distintos niveles de prueba. El orden de las pruebas sería el siguiente: pruebas unitarias, pruebas de componentes, pruebas de integración, pruebas de sistema, pruebas de humo, pruebas alpha, pruebas beta y finalmente pruebas de aceptación.

Las pruebas de regresión se pueden considerar como la ejecución (normalmente automática) de las pruebas ya realizadas hasta el momento. [35]

Para este software en particular se le aplicarán las pruebas del sistema, este es un tipo de prueba que tiene como objetivo verificar el ingreso, procesamiento y recuperación apropiado de datos y la implementación apropiada de las reglas de negocios. Este tipo de pruebas se basan en técnicas de caja negra, es decir, verificar el sistema (y sus procesos internos), la interacción con las aplicaciones que lo usan vía GUI[17] y analizar las salidas o resultados. Esta prueba incluye la prueba de funcionalidad, usabilidad, performance, seguridad, esfuerzo (Stress), recuperación, entre otras.

Las pruebas de sistema incluyen técnicas del método de prueba de Caja Negra, estas pruebas a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software.

Se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca.

Los errores que se pueden encontrar con este método pueden ser de funciones incorrectas o faltantes, errores de interfaz, en las estructuras de datos o en el acceso a bases de datos externas, errores de comportamiento o rendimiento y en la inicialización y terminación. Para ello se aplica durante las últimas etapas de la prueba, además, la prueba de caja negra no considera la estructura de control y la atención se enfoca en el dominio de la información. La técnica de la partición equivalente pertenece a este método de prueba y puede ser muy útil cuando se quieren verificar si existen errores en las estructuras de los datos.

[17] *Interfaz gráfica de usuario (del inglés graphical user interface), programa o entorno que gestiona la interacción con el usuario basándose en relaciones visuales como iconos, menús o un puntero.*

3.3.2 Diseño de casos de prueba

La partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Un caso de prueba ideal descubre de primera mano una clase de errores (por ejemplo, procesamiento incorrecto de todos los datos carácter) que de otro modo podrían requerir la ejecución de muchos casos de prueba antes de observar el error general.

El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. De acuerdo con lo expuesto, si un conjunto de objetos puede vincularse mediante relaciones que son simétricas, transitivas y reflexivas, se presenta una clase de equivalencia [Lea Beizer]. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana.

Al aplicar esta técnica, pueden desarrollarse y ejecutarse los casos de prueba para cada ítem de datos del dominio de entrada. Los casos de prueba se seleccionan de modo que se revise a la vez el número más grande de atributos de una clase de equivalencia [36].

Tabla 4 Caso de prueba correspondiente a la HU Insertar certificado de inspección

Escenario	**Descripción**	**V1**	**V2**	**V3**	**V4**	**Respuesta del sistema**	**Flujo central**
EC1.1 Introducir datos correctamente	El usuario introduce los datos en el certificado de inspección	V	V	V	V	El usuario crea el certificado de inspección y redirecciona la lista de certificados de inspección	El usuario introduce los datos correctamente y pulsa en el botón Aceptar

		A1234	20/04/2017	24/04/2017	La inspección al buque “La maravilla” fue realizada por el inspector Miguel Ángel Fernández Meriño quien le realizó posteriormente peritaje por encontrar indicios de fuerza en las cerraduras de los contenedores a bordo.	el documento	
EC1.2 Campos vacíos	El usuario deja uno o más campos vacíos	V	I	I	V	El sistema muestra el mensaje “Debe llenar los campos obligatorios”	Se completa el formulario quedando campos vacíos
		A1340	“ ”	“ “	“ “		
		V	I	V	V	El sistema muestra el mensaje “Debe llenar los campos obligatorios”	
		A1340	“ “	20/04/2017	El avión comercial B12 tiene malas condiciones de almacenamiento, no cumple con los requisitos establecidos para transportar a mercancía		
		V	V	V	V	El usuario crea el certificado de inspección	
		A1340	20/04/2017	24/04/2017	“ “		

						y redireccio na la lista de certificado s de inspección el document o	

Tabla 5 Variables empleadas en el caso de prueba correspondiente a la HU Insertar certificado de inspección

No	**Nombre del campo**	**Clasificación**	**Valor nulo**	**Descripción**
1	Número de orden	Campo entero	No	Se inserta una cadena de texto con valores alfanuméricos.
2	Fecha de emisión	Campo fecha	No	Se selecciona la fecha de emisión, es decir, la fecha en la cual se emitió el certificado de inspección.
3	Fecha de entrega al cliente	Campo fecha	No	Se selecciona la fecha en la que se le hizo entrega al cliente del certificado.

4	Observaciones	Campo texto	Si	Se inserta una cadena de texto donde se esclarecen elementos no tomados en cuenta en los campos precedentes.

Resultados de las pruebas

Para validar los requisitos funcionales se realizaron tres iteraciones de pruebas al módulo para el Sistema Integral de Administración para la ASIC. A continuación, se muestran los resultados obtenidos por cada iteración:

Tabla 6 Resumen del resultado de las pruebas funcionales

No Conformidades	1ra iteración	2da iteración	3ra iteración
Detectadas	11	4	0
Resueltas	11	4	0
Pendientes	0	0	0

Entre las No Conformidades más significativas se destacan las siguientes:

Tabla 7 Resumen de las principales No Conformidades detectadas

No Conformidades	Resultado	Prioridad	Iteración
Insertar informe de servicio	En nuevo_informe_servicio.html no muestra el contenido	Media	1
Insertar incidencia de inspección de bultos	En nueva_incidencia_inspeccion_bultos.html no muestra el contenido	Media	1

Insertar informe de inspección a vehículos	En informe_inspeccion_vehiculos.html no muestra el contenido	Media	1
Insertar informe de eficacia	En nuevo_informe de eficacia.html no muestra el contenido	Media	1
Registrar entrega de sellos	En nueva_entrega_sellos.html no muestra el contenido	Media	1
Insertar incidencia de inspección a inventario	En nueva_incidencia_inspeccion_inventario.html no muestra el contenido	Media	1
Insertar incidencia de inspección a unidades de mercancía	En nueva_incidencia_inspeccion_unidades_mercancia.html no muestra el contenido	Media	1
Insertar incidencia de inspección a vehículos	En nueva_incidencia_inspeccion_vehiculos.html no muestra el contenido	Media	1
Insertar certificado de inspección	En nuevo_certificado_inspeccion.html no muestra el contenido	Media	1
Insertar informe de inspección a condiciones generales del contenedor	En nuevo_informe_insp:condic_gen_contenedor.html no muestra el contenido	Media	1
Insertar hecho extraordinario	En nuevo_hecho_extraordinario.html no muestra el contenido	Media	1
Insertar informe de servicio	La vista InformeServicioCreate no redirecciona correctamente para InformeServicioList	Alto	2
Modificar informe de servicio	La vista InformeServicioUpdate no redirecciona correctamente para InformeServicioList	Alto	2
Eliminar informe de servicio	La vista InformeServicioDelete no redirecciona correctamente para InformeServicioList	Alto	2
Visualizar informe de servicio	La vista InformeServicioDetail no redirecciona correctamente para InformeServicioList	Alto	2

3.3.3 Pruebas de rendimiento

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. La prueba del rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema.

Las pruebas de rendimiento tienden a unirse con las pruebas de esfuerzo y por lo general requieren instrumentación de hardware y de software, es decir, con frecuencia es necesario medir la utilización de los recursos (por ejemplo, ciclos del procesador) en forma meticulosa. La instrumentación externa puede monitorear intervalos de ejecución y eventos de registro (por ejemplo, interrupciones) conforme ocurren, y los muestreos del estado de la máquina de manera regular. Con la instrumentación de un sistema, la persona que realiza la prueba puede descubrir situaciones que conduzcan a degradación y posibles fallas del sistema. [34]

- **Prueba de estrés:** Su objetivo es verificar el tiempo de respuesta del sistema para transacciones o casos de uso de negocios, bajo diferentes condiciones de carga.
  La meta de las pruebas de carga es determinar y asegurar que el sistema funciona apropiadamente aún más allá de la carga de trabajo máxima esperada. Adicionalmente, las pruebas de carga evalúan las características de desempeño (tiempos de respuesta, tasas de transacciones y otros aspectos sensibles al tiempo). [34]
- **Prueba de carga:** Su objetivo es verificar que el sistema funciona apropiadamente y sin errores, bajo estas condiciones de estrés:
  - Memoria baja o no disponible en el servidor.
  - Máximo número de clientes conectados o simulados (actuales o físicamente posibles)
  - Múltiples usuarios desempeñando la misma transacción con los mismos datos.
  - El peor caso de volumen de transacciones. [37]

Las pruebas de estrés se proponen encontrar errores debidos a recursos bajos o completitud de recursos. Poca memoria o espacio en disco puede revelar defectos en el sistema que no son aparentes bajo condiciones normales. Otros defectos pueden resultar de incluir recursos compartidos, como bloqueos de

base de datos o ancho de banda de la red. Las pruebas de estrés identifican la carga máxima que el sistema puede manejar.

Para la realización de las pruebas de carga y estrés se empleó la herramienta Apache JMeter, el cual es un es un software de código abierto, una aplicación diseñada totalmente en JAVA para medir el rendimiento y comportamiento de servidores mediante pruebas. Originalmente se diseñó para probar aplicaciones Web, pero se ha ampliado desde entonces a otras funciones.

Se utiliza para probar el rendimiento tanto de los recursos estáticos y dinámicos (archivos, Servlets, scripts de Perl, objetos Java, bases de datos - consultas, servidores FTP y mucho más). Se puede utilizar para simular una carga pesada en un servidor, la red o un objeto para poner a prueba su resistencia o para analizar el rendimiento global en diferentes tipos de carga. Puede usarse para hacer un análisis gráfico de rendimiento o para probar el comportamiento de diferentes elementos con un gran volumen de carga y concurrencia. [38]

El ambiente de prueba estuvo conformado por:

- Sistema Operativo: Windows 10
- Microprocesador: Intel(R) Core(TM) i3-3120M CPU @ 2.50 GHz 2.50 GHz
- Memoria RAM: 4GB
- Disco Duro: 320 GB
- Tipo de Conexión: Ethernet 10/100Mbps.

Resultados y análisis de las pruebas de carga y estrés

Para realizar la validación del sistema se definen una prueba de hasta 150 usuarios. Se simularon un total de 390 peticiones a cinco direcciones del módulo en el servidor. En la siguiente figura se observan los resultados obtenidos:

USUARIOS	# MUESTRAS	% ERROR	PROMEDIO RESPUESTA
20	1000	0,00 %	0,14 segundos
90	4500	0.36 %	0.42 segundos
250	**12500**	**2.54%**	**1.2 segundos**

Figure 20 Prueba de carga y estrés realizada en la herramienta JMeter Apache

Descripción de los campos observados en la figura:

- **Muestras**: cantidad de hilos utilizados para la URL.
- **%Error**: porciento de error de las respuestas de las peticiones.
- **Promedio de respuesta**: tiempo promedio en milisegundos para un conjunto de resultados.

Previamente a la realización de la prueba se le asigna la cantidad de usuarios que va a introducir en la petición, en este caso, se hizo una variación de 20 hasta 250 usuarios para determinar los cambios que pudieran efectuarse. Se procede a realizar la petición HTTP en donde se insertan las URLs a chequear, en un primer caso se introdujo http://127.0.0.1:8000/gestion_informes/informe/nueva, la cual es la dirección del RF1: Insertar informe de servicio, luego se introdujo http://127.0.0.1:8000/gestion_informes/inf_vehic/nueva que da respuesta al RF21: Insertar informe de inspección a vehículos, y finalmente http://127.0.0.1:8000/gestion_informes/inc_bult/nueva, la cual corresponde al RF5: Insertar incidencia de inspección a bultos.

En el número de muestras se observa el valor creciente que se le dio a los hilos, estos corresponden al número de usuarios que accede al sistema, en la primera iteración el porciento de error de las respuestas a las peticiones fue 0 y un promedio de respuesta de 0,14 segundos; en la segunda iteración el número de usuarios fue mayor y por consiguiente el número de muestras también resultando el porciento de error en 0,36 y el promedio de respuesta correspondiente a 0,42 segundos; en la tercera iteración el número de usuarios fue de 250 siendo así el número de muestras 12 500, el porciento de error 2,54 y el promedio de respuesta 1,2 segundos.

Con estos datos puede afirmarse que el servidor es capaz de responder rápidamente a las peticiones realizadas por los usuarios así su número sea cada vez mayor, el error se incrementa a la vez que aumenta el número de usuarios, pero su porcentaje es mínimo por lo que n afectaría en gran manera las funciones del módulo.

3.3.4 Pruebas de integración

Las pruebas de integración son aquellas que se le hace a un sistema cuando se integran dos o más módulos en él, para comprobar que todos los elementos funcionan correctamente. En el caso que comprende esta investigación el sistema está integrado con otro módulo implementado por el desarrollador Marbier Pérez,

este módulo comprende la sección administrativa como registrar a los nuevos usuarios y autenticarlos, además de gestionar varios servicios como es la entrega de los medios, las facturas, entre otros.

Para el Sistema Integral de Administración para la ASIC, ambos módulos se desarrollaron en un mismo entorno de desarrollo, con el objetivo de que al unirlos no sucedieran errores con respecto a la compatibilidad y las funciones que realizan las herramientas seleccionadas previamente. Además, las relaciones de dependencia dadas por las clases del módulo comercial realizado por el otro desarrollador y las del módulo de informes e incidencias realizado por la autora, fueron tomadas en cuenta al llamar los atributos en común.

Por ejemplo, el formulario del informe de servicio requiere datos de los trabajadores, los contratos y la orden de trabajo, pertenecientes estos datos al otro módulo desarrollado. Al llamar en las vistas a estos atributos no ocurrió ningún error por las precauciones tomadas por los desarrolladores del sistema previamente, es decir, los datos fueron llamados de igual manera para su efectiva búsqueda en el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga, de esta manera, al necesitarse en el formulario del informe de servicio el número de orden de servicio este es encontrado en las tablas concernientes a esta entidad sin ningún contratiempo.

Las pruebas de integración arrojaron el siguiente resultado:

- El módulo se integró correctamente con el Sistema Integral de Administración para la ASIC, logrando los procesos de gestión de informes e incidencias, sin afectar las funcionalidades del sistema.

3.4 Conclusiones parciales

Se definieron estándares de codificación que permitieron implementar códigos fáciles de entender por otros programadores para su futuro reutilización.

Se crearon los diagramas de componentes y de despliegue, los cuales permitieron conocer la separación lógica del código fuente, las relaciones entre los componentes y la distribución física del problema sobre una arquitectura de software.

Las pruebas realizadas al software validaron las cuarenta HU definidas a partir de los requerimientos especificados por el cliente, durante la etapa de recopilación de la información. Se realizaron de esta manera, pruebas de partición de equivalencia, de estrés y de carga. No se detectaron No Conformidades sin resolver.

Conclusiones

Esta investigación tuvo como base el desarrollo de un módulo para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga, para ello se dio cumplimiento a una serie de tareas de investigación, las cuales fueron cumplidas satisfactoriamente, por lo que se puede concluir que:

- El análisis de la situación actual de la gestión empresarial reafirmó que las soluciones existentes para los procesos de gestión de informes e incidencias en esta área no responden a las exigencias presentadas por la Agencia de Supervisión e Inspección de Carga, ya que los sistemas analizados no contaban con los requerimientos definidos por esta para con los servicios que le es imprescindible manejar.
- Las herramientas, tecnologías y estándares fueron guiados por la metodología de software seleccionada, la cual proveyó de los artefactos requeridos para la comprensión de la propuesta a desarrollar.
- La validación a través de la definición de una estrategia de prueba, permitió comprobar el funcionamiento correcto del módulo de informes e incidencias para el Sistema Integral de Administración para la Agencia de Supervisión e Inspección de Carga, a partir del cumplimiento de los requisitos definidos por el cliente.
- El desarrollo del módulo de gestión de informes e incidencias para el Sistema Integral de Administración, contribuye a la calidad de los servicios brindados en la Agencia de Supervisión e Inspección de Carga, por lo que se considera una herramienta de apoyo a los procesos de gestión empresarial asociados a los servicios.

Recomendaciones

Luego de cumplido el objetivo general y analizados los resultados obtenidos se recomienda:

- Agregar nuevas funcionalidades a la aplicación, como son filtros de búsqueda por provincia o municipio.

Bibliografía

1. Hosford, Rubén y Bayarre, Héctor. Métodos y técnicas aplicados a la Investigación en Atención Primaria de Salud .

2. Tramullas, Jesús. Tendencias en documentación Digital. 2006.

3. Pujol y Fuentes. Documentación y Periodismo. 1997.

4. Arenas, José A. Fernández.

5. Terry, George R.

6. Chekland, P. Information, systems, and information systems. Chichester, UK : John Wiley & Sons, 1998.

7. http://definition.de/informe/. [En línea]

8. https://definicion.mx/informe-tecnico. [En línea]

9. http://definicion.de/incidencia. [En línea]

10. CMS. CMS. Selecting a development approachhttp://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf. Selecting a development approach. [En línea] 27 de marzo de 2008.

11. Canos, J, Letelier, P y Penadés, M. C. Metodologías ágiles en el desarrollo de software. Valencia : Universidad Politécnica de Valencia, 2003.

12. UCI. Programa de mejora: Metodología de desarrollo para la Actividad productiva de la UCI .

13. Hernández Orallo, Enrique. El Lenguaje Unificado de Modelado (UML). s.l. : Addison Wesley, 2000.

14. Rational Software Architect Designer. [En línea] http://www-03.ibm.com/software/products/es/category/application-lifecycle-management/IBM%20-%20Rational%20Software%20Architect%20Designer.html.

15. PHP vs Python en desarrollo web. [En línea] https://programadorwebvalencia.com/php-vs-python-en-desarrollo-web/.

16. Rossum, G. El tutorial de Python. s.l. : Jr. Fred L. Drake, 2009.

17. González, R. Python para todos. [En línea] 2010. http://edge.launchpad.net/improvepython-spanish-doc/0.4/0.4/+download/Python%20para%20todos.pdf.

18. Python-Django. Framework de desarrollo web para perfeccionistas basado en el Modelo MTV. Condori, J.L. 2012.

19. Mestras, Juan Pavón. Boostrap 3.0. Aplicaciones Web/Sistemas Web.

20. Aula informative. Desarrollo web, Framework responsive alternativas a Boostrap. 2016.

21. Los mejores gestores de base de datos del mercado. [En línea] 2017. https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.

22. Martínez, R. Sobre PostgreSQL . [En línea] 2010. http://www.postgresql.org.es/sobre_postgresql.

23. Obe, R. O. y HSU, L. S. PostgreSQL: Up and Running: A practical Introduction to the Advanced Open Source Database. s.l. : O`Reilly Media, Inc, 2014.

24. Los cinco mejores IDE para Python . [En línea] 2016. http://www.pythondiario.com/2016/11/los-5-mejores-ide-para-python.html.

25. Jetbrains. Jetbrains. [En línea] http://www.jetbrains.com/pycharm.

26. Larman, C. UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2004.

27. Olivera , A. G. y Novelo, C. Reporte de instalación de apache. Escárcega : s.n., 2010.

28. ASPP, I. Multimedia Takes on Societal Challenges. EIC`s Message. 2015.

29. Holovaty, Adrian y Kaplan Moss, Jacob. La guía definitiva de Django.

30. Grosso, A. Prácticas de software, Experiencias sobre la Ingeniería y Management del Software. [En línea] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

31. Informáticos, D. Patrones de asignación de responsabilidades (GRASP). Universidad de Sevilla : s.n., 2005.

32. Peralta , C. y Durán, D. Módulos de edición de plantillas y recepción de órdenes de impresión para el Sistema de Personalización de Documentos de Identidad basado en nuevas tecnologías libres. La Habana, Cuba : s.n., 2014.

33. Guía de estilo para el código Python-PEP 8 . [En línea] www.recursospython.com.

34. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El proceso Unificado de Desarrollo de Software. España, Madrid : s.n., 2006.

35. Pressman, R. S. Software Engeneering a Practitioner`s Approach. 2010.

36. Barrientos, Pablo Andrés. Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. 2014.

37. Londoño, Jorge Hernan Abad. Tipos de prueba de software.

38. FAVA - Formación en Ambientes Virtuales de Aprendizaje, SENA - Servicio Nacional de Aprendizaje. Manual Apache Jmeter.

Bibliografía consultada

7 consejos sobre cómo escribir un plan de pruebas, 2013, obtenido de: https://testeandosoftware.com/7-consejos-sobre-como-escribir-un-plan-de-pruebas/

Alexander A.E PEP 8 – Guía de Estilo para Python, obtenido de https://alexanderae.com/pep8-guia-de-estilo-para-python.html, 2013

Asuni Nicola, TCPDF (2017), obtenido de https://tcpdf.org/

Boostrap get started, 2017, obtenido de: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

Boostrap vs Material Design, 2016. Disponible en: http://www.rafelsanso.com/bootstrap-vs-material-design/

Carbali, Mauricio, Pruebas de caja negra y blanca (2013), obtenido de: https://prezi.com/sjwfwmix7slk/pruebas-de-caja-negra-y-caja-blanca/?webgl=0

Cudra, Héctor, Diseño de sistemas de información (2010), obtenido de http://chitita.uta.cl/cursos/2010-2/0001282/recursos/r-7.pdf

Francisco Torregosa, J.,2007. Introducción a la documentación informativa.

Galindos, R. (2013). Guión Visual Paradigm for UML.

Guía de estilo para código en Python, Sloth's Lab, obtenido de http://www.slothslab.com/standards/python/2014/05/08/PEP-0008-Guia-de-estilo-para-codigo-python.html, 2014

Guía de estilo para el código Python - PEP 8 en español. (2013). Obtenido de www.recursospython.com

Guido Van Rossum, Barry Warsaw, Guía de estilo del código Python, obtenido de http://mundogeek.net/traducciones/guia-estilo-python.htm, 2007

http://www.academia.edu/7894130/METODOLOGIAS_AGILES_AUP

Ingeniería de software (2005), obtenido de http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html

Introducción a la teoría general de la administración.

Jacobson, I.; Booch, G.; Rumbaugh, J. El proceso unificado de desarrollo de software. Reading: Addison Wesley, 2000.

Jiménez Contreras, Evaristo y Moya Anegón, Félix de. “Análisis de la autoría en revistas españolas de biblioteconomía y documentación, 1975-1995”.

Lemus, Guillermo Tipos de prueba de software (2012), obtenido de https://es.slideshare.net/GuillermoLemus/tipos-de-pruebas-de-software

López, P., & Ruiz, F. (2011). Lenguaje Unificado de Modelado - UML.

Luis Fernando Ramos Simón, 2003. Introducción a la administración de la información.

Módulo de Reportes Estándares para SIGE, Víctor Manuel Estrada Grillo, Claudia GARCÍA DEL Villar, Héctor Luis Reyes Zaldívar, Elena Leonila Fernández García

Moya Anegón, Félix de, Jiménez Contreras, Evaristo y Moneda Corrochano, Mercedes de la. “Research fronts in library and information science in Spain (1985-94)”.

Niveles de pruebas de software (2013), obtenido de https://pruebasdelsoftware.wordpress.com/2013/01/21/niveles-de-prueba-del-software/

PEP 8 - La guía de estilo para Python, Bioinformatics at COMAV, obtenido de https://bioinf.comav.upv.es/courses/linux/python/estilo.html

Pérez, B. (4 de Julio de 2007). Gestión de las pruebas funcionales. Universidad de la República, Montevideo, Uruguay.

PMOinformatica.com La oficina de proyectos de informática, Tipos de pruebas de software definidos por el ISTQB (2014), obtenido de http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html

Pressman, R. S. (2010). Software Engeneering a Practitioner's Approach.
Pruebas de software (2015), obtenido de http://pruebasdesw.blogspot.com/2015/08/niveles-de-prueba.html

Pruebas de software, Niveles de pruebas (2011), obtenido de http://ingenieriadepruebasdelsoftware.blogspot.com/2011/01/niveles-de-prueba-del-software.html

PyCharm. (16 de diciembre de 2015). Obtenido de http://www.jetbrains.com/pycharm/
Python Software Foundation, obtenido de http://pyfound.blogspot.com/

Reportes en SYmfony (2009), obtenido de https://groups.google.com/forum/#!topic/symfony-es/iznMqpJG_gE

Salvador, J. (2015). Metodologías Ágiles AUP. Obtenido de
Sarmiento, J. (16 de febrero de 2016). UML: Diagrama de despliegue. Obtenido de Visión general de los diagramas de despliegue: http://umldiagramadespliegue.blogspot.com/
Software QA- ¿Cuáles son los tipos de prueba de software? (2015), obtenido de http://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/

Sommerville, I. (2005). Ingeniería de Software. 2, Séptima Edición, 712. Madrid, España: Pearson Education.
Template Monster, ¿Boostrap o Foundation? ¿Cuál HTML5 Framework es más sensible para el desarrollo de aplicaciones web? 2017. Disponible en: http://www.templatemonsterblog.es/2013/06/05/bootstrap-o-foundation-cual-html5-framework-es-mas-sensible-para-el-desarrollo-de-sitios-web/

Ureña, C. (2011). Lenguajes de Programación.
Van, G. (septiembre de 2009). El tutorial de Python.