Facultad de Ciencias y Tecnologías Computacionales

“Sistema Informático para la gestión de incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas”

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Frank Michel Rodríguez Palenzuela

Tutor:

Ing.Willian Simón Grass

La Habana, 30 de junio de 2017

“Año 59 de la Revolución”

"El secreto del éxito es la constancia en el propósito".

Benjamín Disraeli

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año 2017.

_____	_____
Firma del tutor	Firma del autor
Ing. Willian Simón Grass	Frank Michel Rodríguez Palenzuela

DATOS DE CONTACTO

Tutor:

Ing. Willian Simón Grass: Graduado en el año 2013 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Se desempeña actualmente como profesor del departamento de ISW y Practica Profesional de la Facultad de Ciencias y Tecnologías computacionales. Presenta experiencia como tutor y oponente en pregrado.

Correo electrónico: wgrass@uci.cu

AGRADECIMIENTOS

A Dios.
Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mi madre Verónica.
Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A mi padre Alfredo.
Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante y por su amor.

A mi hermana Jennifer.
Por preocuparse tanto por mí y ser tan buena hermana.

A mis hijos.
Por ser el mayor regalo que me dio la vida y para que vean en mi un ejemplo a seguir.

A mi tutor.
Ing. Willian Simón Grass por su paciencia, dedicación, empeño y apoyo ofrecido en este trabajo, que sin su ayuda no hubiera sido posible.

A mis maestros.
Ing. Yadira Barroso Rodríguez por su gran apoyo y motivación para la culminación de mis estudios profesionales; a la Ing. Bárbara Bron Fonseca por su entera disposición e impulsar el desarrollo de mi formación profesional; Ing. Luis Enrique Argota Vega por su tiempo compartido y el apoyo brindado para la elaboración de esta tesis.

A mis amigos.
Que nos apoyamos mutuamente en nuestra formación profesional y que hasta ahora, seguimos siendo amigos: Lizy, Ariadna, a Eddy por haberme ayudado a realizar este trabajo.

Finalmente, a los maestros, aquellos que marcaron cada etapa de nuestro camino universitario, y que me ayudaron en asesorías y dudas presentadas en la elaboración de la tesis.

RESUMEN

En la Dirección de Servicios Generales de la Universidad de las Ciencias Informáticas se llevan a cabo numerosos procesos que sustentan los servicios de la comunidad universitaria. Entre los procesos que se lleva a cabo se encuentran la recogida de desechos sólidos, el mantenimiento y cuidado de áreas verdes, se vela por el buen servicio y distribución de agua potable y la distribución del gas a la comunidad. Procesos que hoy debido a su demanda generan un grupo de incidencias, que el no conocimiento oportuno de las mismas por parte de los operarios del área, afecta la calidad del servicio prestado a los usuarios. La presente investigación describe una solución a este problema con la implementación de un Sistema informático para la gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas. El cual permite agilizar e informatizar dicho proceso de manera centralizada. Para guiar el desarrollo se utilizó como metodología *OpenUP*, como lenguajes de programación *PHP* y *Java Script; Symfony* como marco de trabajo, *NetBeans* como entorno de desarrollo integrado, *Apache* como servidor web y *MySQL* como sistema gestor de base de datos. El sistema obtenido permite la gestión de incidencias e información georreferenciada en la Universidad de las Ciencias Informáticas, dándole al proceso una mayor rapidez, organización y control; a su vez garantiza mayor disponibilidad de la información, además de elevar la precisión en la toma de decisiones por parte de la Dirección de Servicios Generales de la universidad.

Palabras Claves: *información georreferenciada, incidencias, sistema de gestión, gestión de incidencias.*

ABSTRACT

In the General Services Direction of the Informatics Science University, several processes are carried out that support the services of the university community. Among those processes that are carried out are solid waste collection, maintenance and care of green areas, ensuring good service and distribution of drinking water and distribution of gas to the community. Processes that today due to their demand generate a group of incidents, which it does not know about them on the part of the operators of the area, affects the quality of the service provided to the users. This research describes a solution to this problem with the implementation of a computer system for the managing of Incidents and georeferenced information for the Informatics Science University. This allows to streamline and computerize this process centrally. To guide the process OpenUP as development methodology, PHP and Java Script were used as programming languages; Symfony as a framework, NetBeans as an integrated development environment, Apache as a web server and MySQL as a database management system. The system obtained allows the management of incidents and georeferenced information in the Informatics Science University, giving this process a greater speed, organization and control, also guaranteeing a greater availability of information by increasing the precision in decision making by the General Services Direction of the university.

Keywords: *georeferenced information, incidents, management system, incident management.*

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Históricamente, el hombre, a través de sus prácticas diarias de tipo doméstico, comercial, industrial, etc., ha requerido de procesos sencillos o complejos que generan una diversidad de productos e igualmente de desechos. Dentro de estos encontramos diferentes tipos, clasificados de acuerdo a su estado (liquido, sólido o gaseoso), a su origen (residencial, comercial, industrial, etc.), a su manejo (peligrosos e inertes) y por último a su composición (orgánicos e inorgánicos).

Cada día que pasa, la producción de residuos va creciendo exageradamente, originando una problemática ambiental como la contaminación de recursos naturales, la contaminación visual, entre otros. Todo esto ocurre debido a que son arrojados a fuentes hídricas, terrenos no poblados, o simplemente en lugares no apropiados, causando la alteración paisajística del ecosistema y, en consecuencia, afectando a la salud, lo que provoca un deterioro en la calidad de vida de las comunidades y la alteración de los recursos naturales. Por lo que en determinados sitios habitables del planeta se toman medidas para la recolección y selección de estos desechos y a su vez se contribuye con la salud ambiental.

Nuestro país no se encuentra exento de estas medidas, habilitándose así el servicio de Salud Pública Nacional del Ministerio de Salud, que cuenta entre sus funciones con la recogida de desechos sólidos por parte de la corporación Aurora perteneciente a la empresa de Comunales a través de colectores de basura ubicados en diferentes puntos estratégicos de la ciudad para evitar la contaminación ambiental y visual de áreas urbanas, así como su sistemática revisión y vaciado, con el fin de transportar estos desechos a los denominados vertederos para ser tratados en dependencia de su clasificación. Sin estos colectores se formarían acumulaciones de desechos en cada esquina de las localidades, lo que trae consigo una epidemia masiva y el costo de numerosas vidas dejando secuelas en la salud de toda la población sin contar la magnitud del deterioro del medio ambiente.

El principal problema radica en que no existe manera rápida y confiable de realizar un diagnóstico a todos los colectores de un área determinada que consistiría en conocer el estado del colector y si se encuentra en condiciones para seguir su función de recolección. A no ser los reportes que realiza el departamento de Salud Pública desde su función de recogida, pero al estar programado cada determinado tiempo, puede darse el caso de que se llene un colector y no sea vaciado hasta la próxima recogida programada.

En la Universidad de Ciencias Informáticas (UCI) la entidad encargada de brindar este servicio es la Dirección de Servicios Generales (DSG), que entre otras funciones vela por el estado y cuidado de los colectores de desechos y por un buen método de vaciado sistemático de los mismos. Entre las estrategias

de la universidad, se encuentran la creación y conservación de las áreas verdes. Estos espacios son indispensables por los múltiples servicios ambientales y sociales que prestan dentro del ambiente universitario.

Entre los servicios ambientales que las áreas verdes prestan hoy a la universidad tenemos: la generación de oxígeno; la disminución de los niveles de contaminantes en el aire; la disminución de los efectos de las llamadas “islas de calor”; el amortiguamiento de los niveles de ruido; la disminución de la erosión del suelo; además de representar sitios de refugio, protección y alimentación de fauna silvestre. En cuanto a los servicios sociales, las áreas verdes de la universidad representan los espacios favoritos para el esparcimiento, recreación y deporte de la comunidad, además del realce de la imagen urbana, haciendo de ella una institución universitaria más agradable y con una identidad propia.

Con el auge de las nuevas Tecnologías de la información y las comunicaciones (TIC) y el nivel de competitividad en la industria del *software*, numerosas organizaciones en función de ganar en eficiencia y control en los procesos que llevan a cabo, optan por la creación y utilización de productos informáticos. Estos favorecen en gran medida el desarrollo y vigencia de la industria del *software* en la automatización de sus principales procesos, siendo la misma la principal filosofía de la presente investigación teniendo en cuenta el aprovechamiento de recursos existentes en nuestra universidad y un grupo de dificultades que han surgido en la DSG de la misma, que atentan contra el buen servicio de recogida de desechos sólidos, mantenimiento y cuidado de áreas verdes, y la distribución de agua en la universidad. Quedando evidenciado que:

- La difícil gestión y generación de reportes de incidencias e insuficiente control y aprovechamiento de recursos asociados a los colectores de desechos sólidos ha posibilitado la proliferación de tiraderos clandestinos, como: generación y propagación de malos olores producto de la degradación de la materia orgánica expuesta al ambiente; contaminación potencial de las aguas superficiales y subterráneas; dispersión de residuos en las proximidades del tiradero, que afectan a la estética de la residencia universitaria; el riesgo de la propagación de enfermedades gastrointestinales provocada por vectores provenientes de los sitios actuales de disposición.
- La posible rotura de líneas hidráulicas de la red de la universidad y el no conocimiento oportuno de estas averías atentan contra el buen funcionamiento y distribución del agua por cada una de las áreas, además del derroche que puede existir y trayendo esto consigo la creación de focos de vectores y propagación de enfermedades en la comunidad universitaria.

- El difícil control y programación de limpieza de áreas verdes, así como el reporte de áreas que por sus características del terreno necesitan de un mantenimiento en un corto período de tiempo.
- La no informatización de la gestión de los procesos llevados a cabo por parte del operario del gas y desconocimiento de condiciones establecidas para la distribución del mismo al usuario teniendo en cuenta la cantidad de consumidores que requieren el servicio. Produce la duplicidad de información al no contar con una Base de datos lo que no existe un control total de la gestión oportuna del recurso, además de generar inconformidades en la comunidad.

Atendiendo a la situación antes planteada se define como **problema a resolver**: ¿Cómo contribuir al control de las incidencias en la Dirección de Servicios Generales de la Universidad de las Ciencias Informáticas?

Para dar solución al problema identificado se traza como **objetivo general**: Desarrollar un sistema informático que contribuya al control y monitoreo de la información geográfica de las incidencias reportadas a la Dirección de Servicios Generales en la Universidad de las Ciencias Informáticas, así como su visualización en tiempo real de las modificaciones realizadas

Una vez planteado el objetivo general se determina como **objeto de estudio:** Los sistemas de incidencias, enmarcado en el **campo de acción:** El sistema de incidencia e información georreferenciada de la Universidad de las Ciencias Informáticas.

La **idea a defender** de la presente investigación sustenta que: Con el desarrollo de una aplicación informática, se contribuirá a la disminución y control de incidencias en la Universidad de las Ciencias Informáticas.

Para guiar el proceso investigativo del presente trabajo se definen las siguientes **tareas investigativas:**

1. Análisis de los conceptos asociados a la gestión de incidencias e información georreferenciada para la construcción del marco teórico de la investigación.
2. Caracterización de sistemas similares para gestionar incidencias y sistemas de información georreferenciada.
3. Selección de la metodología, tecnologías y herramientas a utilizar para el desarrollo del sistema.
4. Identificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del sistema.

5. Implementación de los requisitos identificados luego de aplicada la ingeniería de requisitos, el análisis y diseño para el desarrollo del sistema.
6. Realización de las pruebas de *software* para comprobar el correcto funcionamiento del sistema propuesto.

Para obtener los conocimientos necesarios con la finalidad de hacer posible el cumplimiento del objetivo trazado en la presente investigación, se llevó a cabo un estudio en el que se utilizaron los siguientes métodos de la investigación científica:

Métodos empíricos usados para poder determinar el grado de complejidad que adquiere el problema y para identificar si quedaron o no satisfechas todas las necesidades previstas a través de:

- **Entrevistas:** Aplicada a directivos, técnicos y especialistas de la Dirección de Servicios Generales, con el objetivo de obtener información sobre el proceso de gestión de incidencias en la Universidad de la Ciencias Informáticas, y a partir de las respuestas obtenidas identificar las irregularidades y deficiencias que se cometen en dicho proceso.

Métodos teóricos: Usados para estudiar las características del objeto de investigación que no son observables directamente, para apoyar el análisis, la síntesis y su relación con otros fenómenos:

- **Analítico-Sintético**: Aplicado por el autor para descomponer el problema de la investigación en elementos por separados y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta. Entre los elementos identificados se encuentran el procedimiento creado por la Dirección de Servicios Generales de la Universidad encargados de la gestión y control de incidencias en las áreas de la Universidad, de manera que permita construir correctamente el marco teórico de la investigación.

- **Histórico-Lógico**: Aplicado por el autor para constatar teóricamente cómo han evolucionado los sistemas de gestión de incidencias e información georreferenciada en el ámbito laboral, los cuales varían en dependencia de las necesidades que requieran los usuarios y las posibilidades reales de suplirlas de cada institución en dependencia del momento histórico vigente de forma tal que permita un análisis real a la hora de identificar los problemas que existen en la gestión y control de incidencias en la UCI.

El presente trabajo de diploma se ha estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica.

Se realiza el estudio del arte de las diferentes soluciones existentes, así como lo principales conceptos asociados al dominio del problema de las metodologías, tecnologías y herramientas empleadas para el desarrollo de la solución propuesta.

Capítulo 2: Desarrollo de la Solución.

Se definen las características y se realiza el diseño del sistema a desarrollar. Incluye el modelo de dominio, la definición de los requisitos funcionales y no funcionales. Contiene además los actores y casos de uso representados en el diagrama de casos de uso del sistema y la descripción textual de uno de sus casos de uso arquitectónicamente significativos. En el proceso del diseño se brinda un acercamiento a la implementación del sistema, construyendo para ello los diagramas de interacción del mismo y el modelo de clases del diseño siguiendo el estilo arquitectónico y los patrones de diseño seleccionados, proporcionando la base para proceder a la etapa de implementación. Además, se muestra el modelo de datos con la descripción de las tablas más significativas.

Capítulo 3: Validación de la Solución.

Se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y se muestra una descripción detallada de los paquetes de implementación. También se presenta el mapa de navegación del sistema desarrollado y algunas imágenes para una mejor comprensión del mismo. Se realiza la validación del sistema mediante las pruebas de software, para comprobar la operatividad de las principales funcionalidades. Y finalmente se muestra el diagrama de despliegue, con una breve descripción de los nodos que lo conforman.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se presenta una descripción sobre los principales conceptos asociados al marco referencial de la investigación. Se realiza un análisis acerca de las principales características y ventajas de las diferentes soluciones existentes que facilitan la gestión de incidencias basándose además en sistemas de información georreferenciada. Se fundamenta la metodología, herramientas y tecnologías a utilizar para el desarrollo de la propuesta de solución.

1.1. Conceptos asociados al sistema informático para la gestión de incidencias e información georreferenciada

Sistema de gestión

Un sistema de gestión (SG) es una herramienta o aplicación informática que permite controlar todos y cada uno de los aspectos de una empresa (pedidos, producción, control de presencia, facturación, ventas, administración, etc.), un SG se hace imprescindible ya que, no solamente permite un control exhaustivo del ciclo de producción, sino que además permite la interacción con los dispositivos.

Un SG no solamente está indicado para grandes empresas. Permite a la pequeña y mediana empresa aprovechar sus características para conseguir un perfecto conocimiento del funcionamiento interno, detectando los posibles puntos débiles en la gestión y facilitando la labor de corrección de los mismos.

En resumidas cuentas, un SG es una herramienta fundamental para conseguir unos elevados niveles de control y calidad (CRISTALDO y KLOSTER, 2016).

Sistema de Información

Un Sistema de Información está constituido por los métodos y procedimientos establecidos para registrar, procesar, resumir e informar sobre las operaciones de una entidad. La calidad de la información que brinda el sistema afecta la capacidad de los directivos y ejecutivos para adoptar decisiones adecuadas que permitan controlar las actividades de la entidad (CONTRALORÍA GENERAL DE LA REPÚBLICA, 2009).

Sistemas de información geográfica

Es una integración organizada de *hardware, software* y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión (SANTOVENIA, TARRAGÓ Y CAÑEDO, 2009).

Información geográfica

Se le denomina información geográfica al conjunto organizado de datos espaciales georreferenciada, que mediante símbolos y códigos genera el conocimiento acerca de las condiciones físico - ambientales, de los recursos naturales y de las obras de naturaleza entrópica del territorio nacional. La información es el resultado de un dato y una interpretación (JAVIER, 2015).

Datos geográficos

Entidades, espacio-temporales que describen o cuantifican la distribución, el estado y los vínculos de los distintos fenómenos naturales y sociales (INSTITUTO GEOGRÁFICO AGUSTÍN CODAZZI, 2016).

Objetos geográficos

Los objetos geográficos son abstracciones de elementos del mundo real que están asociadas a una posición geográfica y temporal definida, respectivamente, en un sistema de referencia espacial y temporal (SUÁREZ, 2014).

Monitoreo

Enmarcado en esta investigación monitorear es la observación del proceso de gestión de la información geográfica con el fin de detectar las modificaciones realizadas por los usuarios, capturar los datos referentes a estas modificaciones y transmitirlos para su posterior visualización.

Sistemas de incidencias

La Gestión de Incidentes tiene como objetivo resolver cualquier incidente que cause una interrupción en el servicio de la manera más rápida y eficaz posible. A su vez, “la Gestión de Incidentes no debe confundirse con la Gestión de Problemas, pues a diferencia de esta última, no se preocupa de encontrar y analizar las causas subyacentes a un determinado incidente sino exclusivamente a restaurar el servicio.” (LOAYZA UYEHARA ALEXANDER ALBERTO 2015)

Por otro lado, tenemos que una incidencia se puede definir como “Cualquier evento que no forma parte de la operación estándar de un servicio y que causa, o puede causar, una interrupción o una reducción de calidad del mismo” (GESTIÓN DE INCIDENTES ITIL, 2015).

1.2. Sistemas de gestión de incidencias

A continuación, se expone un análisis sobre algunos de los sistemas dedicados a la gestión de incidencias, incluyendo soluciones tanto privativas como de código abierto. Se hace referencia primeramente a las soluciones de código abierto.

BugTracker.NET

Se presenta como una aplicación web donde datos y web están alojados en sus propios servidores. Algunas de sus características más interesantes son: la capacidad de crear informes propios, permitir definir el flujo de trabajo a seguir e importación de *emails* a “tickets” entre otros. No se ha referencia a que la asignación de incidencias se pueda establecer de forma automática.

Mantis Bug Tracker

Aplicación en PHP. También puede ser usada como herramienta para gestión de proyectos. De esta solución se puede destacar que permite configurar la transición de estados de las incidencias y el soporte en varios idiomas. Pero como en el caso anterior parece que la asignación de incidencias es de forma manual o estática dependiendo del proyecto al que se refiera la incidencia.

Request Tracker

De esta aplicación cabe señalar que permite a los usuarios configurar los campos de sus incidencias, así como crear incidencias a partir de emails y generar respuestas automáticas.

Sistema de Gestión de Incidencias

Este es un sistema de tickets desarrollado en la UCI con el objetivo de crear un sitio donde se puedan organizar todos los reportes de mantenimiento, redes y servicios telemáticos y tecnología de la universidad. Para acceder al sistema es necesario poseer un usuario uci válido para de esta forma identificarse. Una vez dentro del sistema, el mismo ofrece la posibilidad de crear un nuevo ticket, de mostrar todas las incidencias realizadas por el usuario registrado, de buscar incidencias con diferentes filtros y una ventana para configuración. Utiliza código abierto y fácil de manejar.

A continuación, se hace referencia a las soluciones privativas.

JIRA

Al igual que *Matis Bug Tracker* también posee características para ser usado como gestor de proyectos. Es altamente configurable, permite integración con otros sistemas y ofrece estadísticas e informes bastante completos. Uno de los principales inconvenientes del producto es que si lo que pretende obtener sería un gestor de incidencias, las tareas y opciones que lo hacen versátil también como gestor de proyectos pueden dificultar su usabilidad.

ServiceDesk Plus

La funcionalidad de esta solución es muy completa. Posee reglas de negocio para la asignación automática de solicitudes, base de conocimientos, gestión de tareas programadas, integración con correo electrónico. No es sólo un gestor de incidencias sino también posee características de un gestor de cambios y problemas.

I-Solver

Es una solución muy completa. Al igual que el anterior programa permite la asignación automática, creación de incidencias por *email* y creación de incidencias programadas. Además de todo lo anterior, añade la posibilidad de que el cliente valore el grado de satisfacción.

SysAid

Es uno de los programas más clásicos y conocidos para la gestión informática de grandes empresas u organizaciones. Es una solución de *software* integral de servicio de asistencia que ofrece las herramientas necesarias para resolver cualquier desafío de Tecnología Informática (TI). Con *SysAid* es sencillo automatizar la gestión de las llamadas al servicio de asistencia, gestionar y realizar un seguimiento de su *hardware* y *software*, y resolver rápidamente los problemas de TI, tanto en la oficina como en los equipos de tecnología móvil.

Zendesk

Es un sistema de tickets que incorpora un gran número de funciones que lo hacen atractivo y llaman su atención. Ofrece a los clientes una vía rápida para las respuestas que necesitan, con una base de conocimientos, una comunidad, y el portal del cliente todo en un solo paquete. Construye una comunidad haciendo crecer raíces más profundas en la comunidad de negocios, abriendo un diálogo y construyendo

relaciones fuera del ticket de soporte. Proporciona al equipo de trabajo una visión integral de sus clientes y sus problemas de soporte. Al trabajar en un ticket, se tendrá acceso inmediato a la información unificada del cliente, pudiendo hacer búsquedas en la base de conocimientos de agente y obtener información pertinente de otras áreas del negocio.

1.3. Sistemas de información georreferenciada

A continuación, se presentan un conjunto de soluciones existentes tanto a nivel nacional como internacional de sistemas de información georreferenciada.

ArcView

Es el *software* más usado de los SIG porque posibilita de una forma fácil el trabajo en datos geográficos. Tiene una interfaz gráfica amigable, la cual permite desplegar de manera rápida la información geográfica. Es una herramienta con la que se puede visualizar, analizar, crear y gestionar información geográfica. La mayoría de la información posee un componente que puede relacionarse con un lugar geográfico: direcciones, códigos postales, posiciones de GPS, ciudades, regiones, países u otro tipo de localizaciones en terrenos determinados. Permite visualizar, explorar y analizar estos datos revelando patrones, relaciones y tendencias que no se aprecian bien en base de datos, hojas de cálculo o conjuntos estadísticos (ESRI ESPAÑA GEOSISTEMAS, S.A., 1991)

USGS Global Gis

Atlas Digital del planeta tierra. Provee mapas en formato SIG con escala 1:1 millón. Se puede usar con *ArcView* (*ArcVIEW* no requiere si viene con el visor). La precisión de la coordenada es de 2 decimales. Provee coberturas globales de elevaciones, sismicidad, recursos minerales y energéticos.

SiG UCI

Posee información sobre la localización de lugares de interés dentro de la UCI, lo que propicia orientar a los usuarios a la hora de dirigirse hacia alguno de estos. Cuenta con numerosas funcionalidades que permiten realizar cálculos de distancia, elemento de importancia para la planificación del tiempo de traslado de un lugar a otro, así como el cálculo del área que abarca determinado terreno, el cual puede ser empleado para la toma de decisiones en aspectos vinculados a la realización de obras de construcción. Además, posee información asociada al lugar de residencia de estudiantes y trabajadores internos, esta

funcionalidad constituye una forma de localizar a determinada persona y de identificar el camino que debe recorrerse para contactar con esta.

Una vez analizadas cada una de las soluciones existentes se procede a realizar una comparación entre ellas, con el objetivo de determinar si pueden servir para dar solución a la problemática planteada o pueden aportar algo que contribuya en la realización de un nuevo sistema.

Tabla 1 Comparación entre las soluciones existentes

Solución	Propietario	Plataforma	Reportes	Incidentes Georreferenciada
BugTracker.NET	No	Multiplataforma	Si	No
Mantis Bug Tracker	No	Multiplataforma	Si	No
Request Tracker	No	Multiplataforma	Si	No
Sistema de Gestión de Indicidencias	No	Multiplataforma	Si	No
JIRA	Si	Multiplataforma	Si	No
I-Solver	Si	Multiplataforma	Si	No
SysAid	Si	Multiplataforma	Si	No
Zendesk	Si	Multiplataforma	Si	No
ArcView	Si	Multiplataforma	No	No
USGS Global GIS	Si	Multiplataforma	No	No
SiG UCI	No	Multiplataforma	Si	No

Al concluir la comparación y su posterior análisis se constató que las soluciones existentes, aunque presentan funcionalidades útiles para el proceso de gestión de incidencias de la DSG de la UCI no satisfacen todas las necesidades de la investigación ya que de forma general:

- Ninguna de estas aplicaciones está asociadas a suplir ambos procesos, el de aportar mediante información georreferenciada las incidencias en cada una de las áreas.

- No tienen mecanismos de asignación para el control de recursos, operarios y respuestas a soluciones inmediatas emitidas por la DSG.

Dentro de las principales funcionalidades que se consideran útiles y que de manera independiente presentan estas soluciones, se identificaron las siguientes:

- Acceso a la información sobre la localización de lugares de interés dentro de la Universidad, lo que propicia orientar a los usuarios a la hora de dirigirse hacia alguno de estos.
- Cuenta con numerosas funcionalidades que permiten realizar cálculos de distancia, elemento de importancia para la planificación del tiempo de traslado de un lugar a otro, así como el cálculo del área que abarca determinado terreno, el cual puede ser empleado para la toma de decisiones en aspectos vinculados a la realización de obras de construcción, además posee información asociada al lugar de residencia de estudiantes y trabajadores internos, esta funcionalidad constituye una forma de localizar a determinada persona y de identificar el camino que debe recorrerse para contactar con esta.

A partir de los análisis realizados se concluye con la necesidad de desarrollar un nuevo sistema para informatizar el proceso de gestión de reportes de incidencias para la DSG de la UCI. La solución propuesta debe permitir añadir recursos existentes, los cuales son controlados por la DSG, así como generar listados clasificados por áreas y por tipos y con la posibilidad de realizarle modificaciones y todos estos datos mostrarlos geográficamente en un mapa de la universidad. Además, la solución debe ser multiplataforma y debe permitir la generación de reportes que permitan la toma de decisiones de manera eficiente, para contribuir al mejor control las áreas.

1.4. Metodologías, Tecnologías y Herramientas a considerar

Las tecnologías, herramientas y metodologías escogidas para el desarrollo del sistema se exponen a continuación, para lo cual se analizaron las distintas alternativas y las necesidades del proyecto. El principal elemento que se tuvo en consideración fue que las herramientas debían ser compatibles, basadas en *software* libre, gratuitas y sin restricciones de uso.

1.4.1 Metodología de Desarrollo de Software

Se entiende por metodología de desarrollo de *software* a una colección de pasos a seguir los cuales guían el trabajo referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del *software*. Una metodología define con precisión los artefactos, roles y actividades presentes en el proceso de desarrollo de *software*. La finalidad de una metodología de desarrollo es garantizar la calidad del trabajo, logrando con ello el cumplimiento de los requisitos iniciales, minimizando las pérdidas de tiempo en el proceso de generación de *software*. (BEDINI, 2009)

No existe una metodología de *software* universal. El hecho de que cada *software* presente necesidades y entornos diferentes, que deben tenerse en consideración, hace que el proceso deba ser adaptable a sus características. Por otra parte cada metodología tiene características propias del ambiente de desarrollo, constituyendo estos los principales aspectos que rigen el proceso de selección de la metodología adecuada para la construcción de un *software* (JACOBSON et al. 2000). Las metodologías de desarrollo de *software*, por sus características se pueden clasificar en tradicionales y ágiles.

Las metodologías tradicionales centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto que se define completamente en la fase inicial del proceso de desarrollo. Otras características importantes son el alto costo al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno puede ser volátil, se considera a la arquitectura del *software* como esencial y se orienta hacia el trabajo en grandes grupos que pueden ser incluso distribuidos (FIGUEROA, SOLÍS Y CABRERA, 2014).

Por su parte las metodologías ágiles se basan en dos aspectos fundamentales, el retrasar las decisiones y la planificación adaptativa. Esto posibilita que la documentación necesaria para el proceso de desarrollo se elabore en etapas que pueden ser posteriores a la fase inicial y puedan ser adaptadas a cambios que se presenten en el proceso (FIGUEROA et al. 2011).

A partir del análisis de estos dos tipos de metodologías de desarrollo, se realiza la siguiente comparación entre ambas:

Tabla 2: Comparación entre las metodologías de desarrollo de software

Metodología Tradicional	**Metodología Ágil**
La arquitectura del *software* es esencial y se	Concede menos importancia a la arquitectura

expresa mediante modelos.	del software.
Se enfoca hacia el trabajo de grupos grandes que pueden incluso estar distribuidos.	Se enfoca hacia el trabajo en grupos pequeños.
El cliente no forma parte del equipo de desarrollo. Solamente interactúa con este a través de reuniones definidas u oficiales, en algunos momentos del proceso de desarrollo.	El cliente es parte del equipo de desarrollo e interactúa con este constantemente mediante reuniones que pueden ser hasta cierto punto informales.
Genera muchos artefactos y presenta numerosos roles en el equipo de desarrollo.	Genera pocos artefactos y presenta pocos roles en el equipo de desarrollo.
Presenta cierta resistencia a los cambios.	Está especialmente preparada para enfrentar cambios que surgen durante el desarrollo del proceso.

A partir de la comparación realizada se define como metodología de desarrollo para la propuesta de solución una metodología de clasificación ágil. Esta decisión se debe a que el equipo de desarrollo está formado por pocos miembros; el director de la DSG, quien es considerado el cliente principal, tiene la disposición de interactuar con el equipo mediante reuniones en el momento en que sea necesario. Las metodologías ágiles generan pocos artefactos y presentan pocos roles, lo que posibilita dedicar más tiempo al proceso de implementación y terminar la solución con mayor rapidez. Otro factor importante es que durante el proceso de desarrollo existe una importante probabilidad de que sucedan cambios en la concepción de algunos elementos relacionados con la propuesta de solución, situación para la cual las metodologías ágiles están especialmente preparadas.

Dentro de las metodologías ágiles más utilizadas y mejor documentadas está *OpenUP* (*Open Unified Process*), la cual conserva las características principales del modelo de desarrollo de la metodología tradicional *RUP* (*Rational Unified Process*), incluye el desarrollo iterativo, permite identificar los requisitos

operacionales del sistema, prever las interacciones con los usuarios y prevenir los posibles riesgos en el desarrollo del sistema.

Metodología de Desarrollo de Software (*OpenUP*)

OpenUP es una metodología de desarrollo de *software*, basada en *RUP* (*Rational Unified Process*), que contiene el conjunto mínimo de prácticas que ayudan a un equipo de desarrollo de *software* a realizar un producto de alta calidad, de una forma eficiente. *OpenUP*, es un proceso unificado, iterativo e incremental, que se centra en el desarrollo colaborativo de *software* para generar sistemas de calidad. Los elementos que forman *OpenUP* son tareas, disciplinas, artefactos y procesos. El ciclo de vida de un proyecto, según la metodología *OpenUP*, permite que los integrantes del equipo de desarrollo aporten con micro-incrementos, que pueden ser el resultado del trabajo de unas pocas horas o unos pocos días. El progreso se puede visualizar diariamente, ya que la aplicación va evolucionando en función de este micro-incremento.

El objetivo de *OpenUP* es ayudar al equipo de desarrollo, a lo largo de todo el ciclo de vida de las iteraciones, para que sea capaz de añadir valor de negocio a los clientes, de una forma predecible, con la entrega de un *software* operativo y funcional al final de cada iteración. El ciclo de vida del proyecto provee a los clientes de: una visión del proyecto, transparencia y los medios para que controlen la financiación, el riesgo, el ámbito, el valor de retorno esperado (INFANTE A., 2013). Todo proyecto en *OpenUP* consta de cuatro fases: inicio, elaboración, construcción y transición. Cada una de estas fases se divide a su vez en iteraciones.

La metodología de desarrollo *OpenUP* está caracterizado por cuatro principios básicos interrelacionados (BALDUINO 2007):

- Colaboración para unificar intereses y compartir conocimientos.
- Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- Enfoque en la articulación de la arquitectura.
- Desarrollo continuo para obtener realimentación y realizar las mejoras respectivas.

Se escoge como metodología de desarrollo de *software* a *OpenUP*, que permite al sistema integrarse a requerimientos cambiantes que se pueden dar durante el desarrollo del proyecto, es una metodología ágil, que presenta documentación, muy favorable para proyectos que presentan pocos integrantes y es aplicable a sistemas de bajo costo.

1.4.2 Lenguajes de modelado de sistemas

El lenguaje de modelado pretende dar apoyo a la mayoría de los procesos de desarrollo de *software*, son desarrollados con el objetivo de simplificar y consolidar el gran número de métodos de desarrollo que han surgido. (JACOBSON, 2000).

Lenguaje Unificado de Modelado 2.0 (UML)

Una exigencia de las instituciones es que los desarrollos de *software* se formalicen con un lenguaje estándar y unificado. Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común (CORNEJO, 2008). Un lenguaje unificado que cumple con estos requisitos, es ciertamente el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*).

Lo fundamental de UML es la capacidad de diagramación y los diferentes tipos de diagramas que soporta (CORNEJO, 2009). UML 2.0 presenta como característica que es un lenguaje de modelado orientado a objetos y que contiene una viabilidad en la corrección de errores. Permite documentar todos los artefactos de un proceso de desarrollo: requisitos, arquitectura pruebas y versiones. Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

De acuerdo a esto se decide utilizar UML 2.0 como lenguaje de modelado, ya que permite modelar el análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos; además de ser flexible para admitir cambios no previstos durante el diseño o el rediseño.

1.4.3 Herramienta CASE de modelado

La Ingeniería de Software Asistida por Computadora (*CASE* por sus siglas en inglés, *Computer Aided Software Engineering)* constituye la aplicación de métodos y técnicas utilizadas para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelo de sistemas, la

depuración y las pruebas. Las herramientas CASE representan una forma que permite modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información.

Visual Paradigm for UML 8.0

La herramienta de modelado *Visual Paradigm for UML* 8.0 es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue; y no emplea una metodología en específico. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones de calidad (PRESSMAN, 2008). Esta herramienta ofrece las siguientes características:

- Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales Entornos de desarrollo integrado como: *Eclipse/IBM webSphere, NetBeans IDE, Oracle JDeveloper*, entre otras (S. 2002).

Además de las ventajas expuestas, es válido resaltar que se selecciona *Visual Paradigm for UML* 8.0 debido a que es una herramienta que tiene disponibilidad en múltiples plataformas y es capaz de, a partir de un modelo relacional desplegar todas las entidades asociadas a las tablas para diferentes sistemas gestores de base de datos, aspecto importante en la realización de la presente investigación.

1.4.4 Lenguajes de Programación

En la actualidad existen diferentes lenguajes de programación, estos han surgido debido a las tendencias y necesidades de las plataformas existentes.

Desde el comienzo de Internet, surgieron diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. Con el transcurso del tiempo, las tecnologías se desarrollaron y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación para la Web dinámica, que permitieran interactuar con los usuarios y utilizaran sistemas de bases de datos.

Lenguajes de Programación para el lado del servidor PHP3 5.5.1

PHP es un lenguaje script que corre del lado del servidor en la Arquitectura Cliente – Servidor. Es utilizado para generar páginas Web dinámicas. Su programación es segura y confiable ya que de ninguna manera en el navegador se accede al código fuente en PHP, sino solo a su resultado en HTML4 (GONZÁLEZ ENRIQUE, 2010). Con PHP se puede procesar la información de formularios, generar páginas con contenidos dinámicos, entre muchas más cosas. Permite aplicar técnicas de programación orientada a objetos.

Como se ha diseñado para su uso en la Web, incorpora gran cantidad de funciones integradas para realizar útiles tareas relacionadas con la Web. Puede generar imágenes GIF en un instante, establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con cookies y generar documentos PDF (GONZÁLEZ ENRIQUE, 2010).

Una de sus características es su portabilidad ya que está disponible para diferentes sistemas operativos. Dispone de una conexión propia a todos los sistemas de base de datos. Además de MySQL, puede conectarse directamente a bases de datos como PostgreSQL, Microsoft SQL, Oracle, entre otras. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos, no limita su distribución y se puede ampliar con nuevas funcionalidades si se desea. Se puede utilizar como módulo de Apache, lo que lo hace extremadamente veloz. Por estar completamente escrito en C, se ejecuta rápidamente utilizando poca memoria.

Motor de plantillas de PHP: Twig

Es un motor de plantillas para el lenguaje de programación PHP. Su sintaxis se origina a partir de Jinja y plantillas de Django. El producto de código abierto se distribuye bajo licencia BSD y desarrollado por Fabien Potencier. El *framework* Symfony2 viene con un soporte incluido para Twig como su motor de plantillas por defecto.

Las características clave son:

- **Rápido:** Twig compila las plantillas hasta código PHP regular optimizado. El costo general en comparación con código PHP regular se ha reducido al mínimo.

- **Seguro:** Twig tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Esto te permite utilizar Twig como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- **Flexible:** Twig es alimentado por flexibles analizadores léxico y sintáctico. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio DSL.

Leguajes de Programación para el lado del Cliente JavaScript

Es uno de los lenguajes más utilizados en la creación de aplicaciones web presentando funcionalidades para la creación de contenidos dinámicos. Es un lenguaje de programación para el lado del cliente debido a que la mayor carga de procesamiento la soporta el navegador. Su uso está destinado principalmente para la creación de efectos visuales, así como para la creación de interfaces de usuario. Las aplicaciones que utilizan este lenguaje casi siempre están incrustadas en los códigos HTML de las aplicaciones web y están destinadas para realizar las acciones con el cliente.

CSS 3.0

CSS 3 (Cascading style sheets) es un lenguaje que permite la definición de hojas de estilos diseñadas para el control de las interfaces de usuario definidas en los códigos HTML. Separa los contenidos de la presentación de las aplicaciones lo que permite el diseño de documentos bien definido y muy complejo. Cuando son creados los contenidos de estilos es permitido definir los diferentes tamaños, colores y tipos de fuentes para las letras, así como la gestión de imágenes.

Los lenguajes de programación del lado del cliente escogidos fueron CSS y JavaScript, este último se escogió ya que los programas escritos en él no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos y es independiente de la plataforma, hardware o sistema operativo y funciona correctamente siempre y cuando exista un navegador que lo soporte.

El lenguaje escogido para el lado del servidor fue PHP por ser un lenguaje multiplataforma que posee capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL. Este lenguaje es libre y la amplísima biblioteca que posee simplifica el trabajo de los desarrolladores.

1.4.5 Marco de Trabajo (Framework)

Los *frameworks* o marcos de trabajo son aplicaciones compuestas por componentes personalizables que permiten el desarrollo de aplicaciones. Las funcionalidades principales de estas aplicaciones es aligerar de carga a los desarrolladores realizando de manera automática e interactiva tareas comunes dentro de la programación, lo que permite reducir considerablemente el tiempo de desarrollo de aplicaciones complejas.

Framework Symfony 2.8

Symfony es un poderoso *Framework* orientado para la optimización del desarrollo de aplicaciones web basado en el patrón de diseño MVC. El diseño del mismo separa la lógica del negocio, del servidor y la presentación; además proporciona herramientas para reducir el tiempo de desarrollo de aplicaciones complejas. Realiza de forma automatizada las tareas básicas durante el desarrollo permitiendo al desarrollador centrarse en las funciones más específicas de la aplicación. El *Framework* está implementado parcialmente en PHP 5.3, es compatible con los Sistemas Gestores de Base de Datos más reconocidos del mundo y es independiente al sistema operativo del servidor, brindando así características muy poderosas para el desarrollo de aplicaciones web (POTENCIER y ZANINOTTO 2010). Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio

Symfony es el *framework* que se escoge para la realización del sistema porque presenta múltiples posibilidades en el trabajo con las validaciones, integración con otras herramientas y es muy factible para realizar pruebas y depuración del código.

1.4.6 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE) es una herramienta de *software* dedicada exclusivamente al desarrollo de programas informáticos, ofrece una serie de complementos que facilitan el desarrollo ágil de *software*. Esto simplifica el trabajo y ahorra tiempo de desarrollo.

NetBeans 8.1

Netbeans es un entorno de desarrollo gratuito y de código abierto que en el momento de escribir este artículo está en su versión 7.4. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: Java, PHP, Groovy, C/C++, HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. (GENBETA:dev, 2014)

Para el desarrollo de la aplicación informática se seleccionó NetBeans, por permitir escribir, compilar, ensamblar y desplegar aplicaciones, además de brindar soporte para varios lenguajes de programación mediante el empleo de plugins. Facilita el completamiento para los lenguajes de programación HTML, CSS, JavaScript y PHP permitiendo mapear este último.

1.4.7 Sistemas Gestores de Base de Datos (SGBD)

Un SGBD es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Permite almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas, además de ser ampliamente utilizado en entornos científicos con el objeto de almacenar la información experimental.

PostgreSQL 9.3

PostgreSQL constituye un potente gestor de BD de código abierto muy avanzado, ofrece un control de concurrencia multiversión lo que mejora las operaciones de bloqueo y las transacciones en sistemas multiusuario. Soporta casi toda la sintaxis SQL (del inglés, *Structured Query Language*), incluyendo sub-consultas, transacciones, tipos y funciones definidas (MARTÍNEZ, 2010). Además de la conectividad, velocidad, seguridad e interacción con el lenguaje de programación a utilizar para el desarrollo del módulo propuesto.

Con PostgreSQL 9.3 como SGBD debido a la estabilidad, potencia, robustez y facilidad de administración que provee. Además, funciona con grandes cantidades de datos y una alta concurrencia de usuarios, lo que se traduce en una garantía de eficiencia, dado que el módulo en desarrollo debe ser capaz de enfrentar múltiples conexiones simultáneas.

PostGIS 2.0

PostGIS es una extensión del sistema de base de datos objeto-relacional PostgreSQL que permite almacenar objetos GIS (Sistemas de Información Geográfica) en la base de datos. PostGIS incluye soporte para los índices espaciales R-Tree basados en GiST y funciones para el análisis y procesamiento de objetos GIS. PostGIS fue desarrollado por *Refractions Research Inc*, como un proyecto de investigación de tecnología de bases de datos espaciales. *Refractions* es una empresa de consultoría de

GIS y bases de datos en Victoria, British Columbia, Canadá, especializada en integración de datos y desarrollo de software personalizado. Planeamos apoyar y desarrollar PostGIS para soportar una gama de funcionalidades SIG importantes, incluyendo soporte OpenGIS completo, construcciones topológicas avanzadas (coberturas, superficies, redes), herramientas de interfaz de usuario de escritorio para ver y editar datos GIS y herramientas de acceso basadas en la web. (CARTOSIG 2017)

Después de analizar las características y prestaciones de los SGBD mencionados anteriormente, se aprecian las ventajas que brinda PostgreSQL. Este sistema provee de gran capacidad de almacenamiento, consistencia, escalabilidad y rendimiento bajo grandes cargas de trabajo. Es un SGBD objeto – relacional donde su código fuente se encuentra disponible libremente. Por lo tanto, se decide desarrollar la solución propuesta en esta investigación haciendo uso del SGBD PostgreSQL en su versión 9.3, y PostGIS como extensión del SGBD objeto - relacional PostgreSQL permitiendo almacenar objetos de sistemas de información geográfica en el mismo.

1.4.8 Servidor Web

Los servicios web tal como lo define el consorcio *World Wide Web* (WC3) son un conjunto de aplicaciones o de tecnologías con capacidad para inter-operar en la web (WC3, 2014) estas aplicaciones o tecnologías intercambian datos entre sí, con el objetivo de ofrecer servicios.

Servidor Web Apache

El servidor Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, entre sus principales ventajas se encuentran: posee gran cantidad de extensiones para diversas tecnologías, además de una amplia documentación, es libre, modular y multiplataforma. Se señala como desventaja que no posee interfaz gráfica que facilite su configuración.

Este servidor Web es otro de los más ligeros que hay en el mercado. Está especialmente pensado para hacer cargas pesadas sin perder balance, utilizando poca RAM y poca de CPU. Algunas páginas populares que lo usan son YouTube, Wikipedia y otras que soportan gran tráfico diariamente. Es gratuito y se distribuye bajo Licencia BSD.

Se seleccionó el servidor web Apache 2.4.4 debido a que consume pocos recursos, es gratuito, es uno de los más usados y presenta una amplia documentación.

Conclusiones del capítulo

Después de realizar el estudio de la bibliografía relacionada con los sistemas para la gestión de información georreferenciadas, se pudo identificar que las soluciones existentes no satisfacen las condiciones deseadas. Por esta razón se propone la implementación de un sistema que sea capaz de ajustarse a las necesidades existentes y que se nutra de las principales ventajas que muestran algunas de las soluciones analizadas como por ejemplo la capacidad de generación de reportes, la autenticación mediante la cuenta institucional de la universidad y la característica de ser multiplataforma.

Del análisis sobre las tendencias actuales en el desarrollo del software, se seleccionó la metodología de desarrollo y un grupo de tecnologías y herramientas, las adecuadas para realizar el sistema informático que se requiere: como metodología de desarrollo, la metodología ágil *OpenUP*; y como sistema de gestión de base de datos PostgresSQL 9.3 y PostGIS 2.0 como extensión de PostgreSQL para el almacenamiento de objetos de sistemas de información geográfica en el mismo; como lenguaje de modelado UML 2.0 y *Visual Paradigm for UML* 8.0 como herramienta CASE; PHP como lenguaje de programación, haciendo uso de JavaScript y CSS 3, además de Symfony 2.8 como marco de trabajo y NetBeans 8.1 como IDE de desarrollo.

CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN

Introducción

En este capítulo se define el modelo de dominio y se da una propuesta de solución al problema planteado. Se identifican, especifican y describen los requisitos no funcionales y funcionales, agrupando en casos de uso estos últimos. Se brinda un acercamiento a la implementación a través de la etapa de diseño, construyéndose para ello los diagramas de interacción y el modelo de clases del diseño. Además, se selecciona el estilo arquitectónico y los patrones de diseño a utilizar y se muestra el modelo de datos con la descripción de las tablas de la base de datos.

2.1. Modelo de Dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Se realiza cuando no se logra determinar el proceso del negocio con fronteras bien establecidas y donde los flujos de información son difusos (múltiples orígenes, sólo eventos, sucesos), cuando existe solapamiento de responsabilidades, así como múltiples responsabilidades. Se representa con un conjunto de diagramas de clases UML en los que no se define ninguna operación (LARMAN, 2009). En la *Figura 1* se muestra el Modelo de Domino del Sistema de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas.

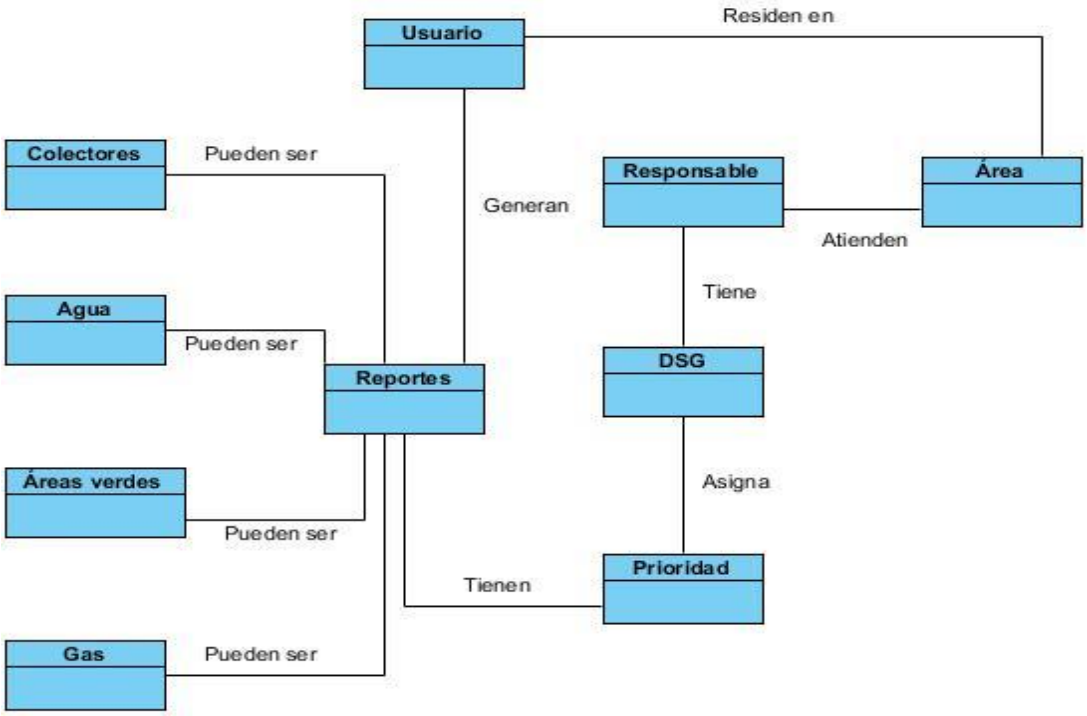

Figura 1 Modelo de dominio del sistema

Definición de clases del modelo del dominio

Usuarios: Son las distintos usuarios que residen en las distintas áreas de la Universidad de las Ciencias Informáticas.

Responsables: Son los encargados de las distintas áreas.

Prioridad: Está dada por el grado de urgencia del reporte.

Dirección de Servicios Generales (DSG): Es la encargada de tramitar los reportes y asignar las prioridades.

Áreas: Hace referencia a las distintas áreas de la universidad.

Reportes: Avisos elaborado por los usuarios de la universidad sobre las incidencias referentes a los servicios de **gas**, **agua**, **áreas verdes** o **colectores de basura**.

Descripción del modelo de dominio

La DSG de la UCI es la encargada de garantizar la recolección, vaciado y mantenimiento de los colectores de basura, así como de la buena distribución del gas licuado y el agua potable, además del mantenimiento y limpieza de las áreas verdes de la universidad.

Los usuarios de estos servicios muchas veces realizan reportes solicitando atención o respuestas de parte de la DSG por disímiles situaciones que se presentan cotidianamente, para esto hay un encargado de área que revisa los reportes que se reciben y determina cuál de ellos atender primero según el grado de prioridad asignado.

2.2. Extracción de Requisitos

La especificación de los requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Contiene requisitos funcionales que son las capacidades o condiciones que el sistema debe cumplir, en el que se incluye un conjunto de casos de uso que describen las interacciones que tendrán los usuarios con el software y contiene, además, requisitos no funcionales que son las propiedades o cualidades que el producto debe tener, que imponen restricciones en el diseño o la implementación. En esta etapa se debe generar una información clara y precisa de los aspectos más relevantes del producto, ya que esta actividad es el hilo conductor de todo el desarrollo del software.

2.2.1 Requisitos funcionales

Los Requisitos Funcionales (RF) definen las condiciones y funciones que el sistema será capaz de realizar. Luego de efectuar la modelación del dominio se obtuvieron los siguientes requisitos funcionales:

Autenticar usuario.

RF 1. Autenticar usuario.

Gestionar usuarios.

RF 2. Adicionar usuario.

RF 3. Eliminar usuario.

RF 4. Mostrar usuario.

RF 5. Modificar usuario.

RF 6. Listar usuario.

RF 7. Asignar Tareas a usuario.

RF 8. Visualizar Tareas de usuario.

Gestionar reportes.

RF 9. Adicionar reporte.

RF 10. Eliminar reporte.

RF 11. Modificar reporte.

RF 12. Mostrar reporte.

RF 13. Listar reporte.

RF 14. Aprobar reporte.

RF 15. Denegar reporte.

RF 16. Atender reporte.

RF 17. Buscar reporte.

Visualizar reportes.

RF 18. Visualizar reporte por área.

RF 19. Visualizar reporte por usuario.

RF 20. Visualizar reporte de solicitudes atendidas.

RF 21. Visualizar reporte de solicitudes por atender.

RF 22. Generar reportes en pdf.

Gestionar recursos.

RF 23. Actualizar recurso.

RF 24. Adicionar recurso.

RF 25. Buscar recurso.

RF 26. Eliminar recurso.

RF 27. Listar recurso.

Gestionar Áreas.

RF 28. Actualizar área.

RF 29. Buscar área.

RF 30. Adicionar área.

RF 31. Eliminar área.

Gestionar puntos en el mapa.

RF 32. Adicionar un punto en el mapa.

RF 33. Modificar un punto en el mapa.

RF 34. Eliminar un punto en el mapa.

RF 35. Mostrar un punto en el mapa.

RF 36. Generar mapas de incidencias por áreas.

Notificar a terceros.

RF 37. Enviar notificación de servicios.

2.2.2 Requisitos no funcionales

Los Requisitos No Funcionales (RNF) son propiedades o cualidades, que pueden usarse para juzgar la operación de un sistema en lugar de su funcionalidad. El sistema debe satisfacer las siguientes condiciones o cualidades que el producto debe tener.

Requisitos de Software.

Para obtener un óptimo funcionamiento del sistema se requiere la existencia de los siguientes requisitos en el servidor y las máquinas clientes que harán uso de la aplicación.

Estaciones de Trabajo

- **RNF 1.** La solución se debe ejecutar sobre cualquier plataforma, la PC cliente debe contar con un navegador Web.
- **RNF 2.** Sistema operativo Ubuntu Server 14.04 o cualquier versión LTS (Long Time Suport, en español, Largo Tiempo de Soporte) superior o Windows Seven o superior.
- **RNF 3.** En los clientes debe estar instalado un navegador web que cumpla con los estándares de la W3C2, preferiblemente Mozilla Firefox 3.0 o superior.

Servidor

- **RNF 4.** Servidor de aplicaciones Apache.
- **RNF 5.** Lenguaje de programación PHP y librerías.
- **RNF 6.** Sistema operativo Ubuntu Server 14.04 o cualquier versión LTS (Long Time Suport, en español, Largo Tiempo de Soporte) superior o Windows Seven o superior.
- **RNF 7.** SGBD PostgreSQL versión 9.3 o superior.

Requisitos de Hardware.

El buen estado de las conexiones de red es imprescindible y de suma importancia para la recogida y entrega de la información que será almacenada en el servidor y posteriormente solicitada por los usuarios.

Estaciones de Trabajo:

- **RNF 8.** Se requiere tarjeta de red a 100Mb o superior.

RNF 9. Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.

RNF 10. Memoria RAM mínimo 512 MB.

Servidor

RNF 11. Ordenador Dual Core o superior, con 3.0 GHz de velocidad de microprocesador.

RNF 12. Memoria RAM mínimo 2 GB.

RNF 13. Disco Duro con 250Gb de capacidad.

RNF 14. Requiere tarjeta de red a 100Mb o superior.

Requisitos de Apariencia o interfaz externa.

RNF 15. Diseño encuadrado para resoluciones de 800x600, pero preparado para verse en otras resoluciones.

RNF 16. El diseño debe permitir el uso de los colores azul y blanco. Para la realización del diseño se tendrá en cuenta el control y la transparencia como eje principal en torno al cual girará el sistema.

Requisitos de Seguridad Confidencialidad.

Confidencialidad

RNF 17. La información manejada por el sistema deberá estar protegida de acceso no autorizado y divulgación.

RNF 18. El sistema debe ser accedido solo mediante comunicación segura a través de la red, haciendo uso del protocolo *http*.

RNF 19. El acceso a las funcionalidades de la solución propuesta debe estar mediado por una jerarquía de roles cumpliendo el principio de mínimo privilegio.

Integridad

RNF 20. La información manejada por el sistema será objeto de cuidadosa protección contra estados inconsistentes y corrupción., a través de roles y permisos de los usuarios autenticados (R-Back).

Disponibilidad

RNF 21. A los usuarios autorizados se les deberá garantizar el acceso a la información solicitada en todo momento.

2.3. Modelo de casos de uso del sistema

El modelo de casos de uso del sistema sirve para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Permite que desarrolladores de software y clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. Está compuesto por un diagrama que muestra las relaciones existentes entre actores y casos de uso del sistema (CUS) y por la especificación de dichos CUS (LARMAN, 2004).

Los CUS representan fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un CUS describe una secuencia de acciones que el sistema debe realizar interactuando con sus actores, incluyendo alternativas dentro de la secuencia (LARMAN, 2004).

Se considera un actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Por lo general estimula al sistema con eventos de entrada o recibe algo de él. Los actores representan a terceros fuera del sistema que colaboran con el mismo (LARMAN, 2004). En la tabla 3 se describen los actores que interactúan con el sistema de información georreferenciada.

Tabla 3: Actores del sistema.

Actor	Descripción
Usuario UCI.	Usuario UCI son todos los usuarios del dominio de la universidad (estudiantes, trabajadores, especialistas), se autentica en el sistema y realiza un reporte.
Administrador.	Es el administrador del sistema. Encargado de la gestión de los usuarios del sistema, de los roles y de gestionar la información georreferenciada.
Técnico de Área.	Atiende la solicitud de los reportes que se realizan.
Asistente de DSG	Asigna las prioridades de los reportes y gestiona las solicitudes a atender.

2.3.1 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema (DCU) representa gráficamente a los procesos y su interacción con los actores. Además, facilita el entendimiento de los procesos realizados por el sistema para el desarrollador.

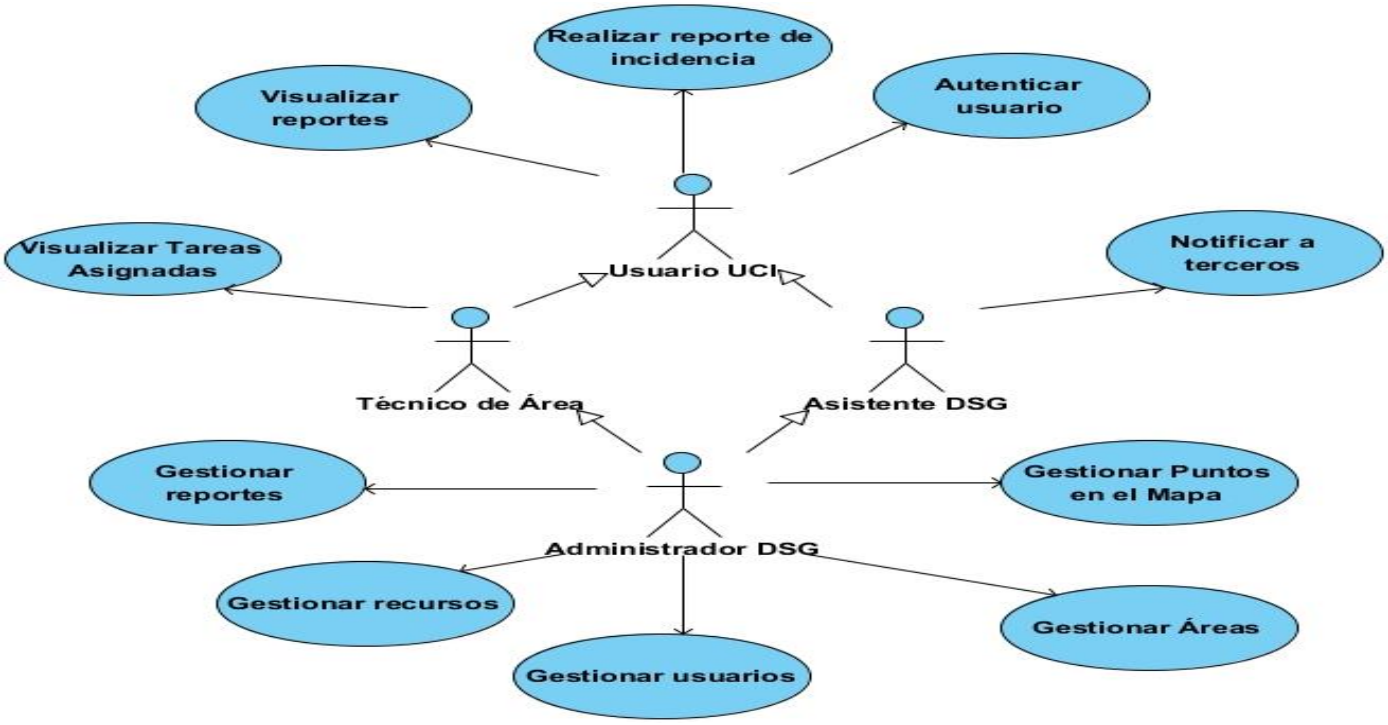

Figura 2 Diagrama de caso de uso del sistema

2.3.2 Descripción de casos de uso del sistema

Cada CUS se detalla habitualmente mediante una descripción textual que describe la funcionalidad que se construirá en el sistema. En la tabla 4 se muestra la descripción textual del CUS Gestionar recurso:

Tabla 4 Descripción de caso de uso

Objetivo	Añadir Reporte por el Usuario
Actores	Usuario
Resumen	El usuario accede a la página de inicio del sistema, donde se muestra una lista de las solicitudes realizadas recientemente y debajo un

	formulario para realizar el nuevo reporte.
Complejidad	Alta.
Prioridad	Crítica.
Precondiciones	
Postcondiciones	Se inserta el nuevo reporte.

Flujo de eventos		
Flujo básico		
	Actor	**Sistema**
1.	Accede a la página de inicio del sistema.	
2.		Muestra un listado con los reportes almacenados y posibilita al usuario realizar el nuevo reporte a través de un formulario con los campos para añadir usuario, ubicación, solicitar seguimiento, urgencia, categoría, descripción y un campo para seleccionar la categoría: Categoría: Desechos Sólidos. **Ver Sección 1.** Categoría: Agua. **Ver Sección 2.** Categoría: Gas. **Ver Sección 3.** Categoría: Áreas Verdes. **Ver sección**
3.	El usuario rellena el formulario seleccionando una de las cuatro categorías de reporte.	
4.		Finaliza el caso de uso.
Flujo alterno		

Nº 1 "Escoge otra opción."		
	Actor	**Sistema**
1	El usuario decide iniciar sesión como administrador del sistema.	
Sección 1: "Añadir reporte de categoría: desechos sólidos**"**		
Flujo básico		
	Actor	**Sistema**
1	Selecciona la categoría "Desechos Sólidos" del campo "Categoría" del formulario de reporte.	
2		Muestra el formulario del nuevo reporte y se le añade un nuevo campo a rellenar con el listado de los colectores existentes en el sistema.
3	Adiciona los datos y da clic en el botón "Guardar".	
4		Se actualiza la tabla de reportes con el nuevo reporte adicionado.
Flujo alterno		
3.1	El usuario decide cerrar sesión.	
3.2		Se muestra la página de login de administrador.
Sección 2: "Añadir reporte de categoría: "Agua"**"**		
Flujo básico		
	Actor	**Sistema**

1	Selecciona la categoría “Agua” del campo “Categoría” del formulario de reporte.	
2		Muestra el formulario del nuevo reporte y se le añade un nuevo campo a rellenar con el listado de los recursos hidráulicos existentes en el sistema
3	Adiciona los datos y da clic en el botón “Guardar”.	
4		Se actualiza la tabla de reportes con el nuevo reporte adicionado.
Flujo alterno		
5.1	El usuario decide cerrar sesión.	
5.2		Se muestra la página de login de administrador.
Sección 3: “Añadir reporte de categoría: “Gas”**”**		
Flujo básico		
	Actor	**Sistema**
1	Selecciona la categoría “Gas” del campo “Categoría” del formulario de reporte.	
2		Muestra el formulario del nuevo reporte y se le añade un nuevo campo a rellenar con el listado de los recursos de gas existentes en el sistema
3	Adiciona los datos y da clic en el botón “Guardar”.	
4		Se actualiza la tabla de reportes con el nuevo reporte adicionado.

Flujo alterno		
3.1	El usuario decide cerrar sesión.	
3.2		Se muestra la página de login de administrador.
Sección 4: "Añadir reporte de categoría: "Áreas Verdes"**"**		
Flujo básico		
	Actor	**Sistema**
1	Selecciona la categoría "Áreas Verdes" del campo "Categoría" del formulario de reporte.	
2		Muestra el formulario del nuevo reporte y se le añade un nuevo campo a rellenar con el listado de las áreas verdes existentes en el sistema
3	Adiciona los datos y da clic en el botón "Guardar".	
4		Se actualiza la tabla de reportes con el nuevo reporte adicionado.
Flujo alterno		
4.1	El usuario decide cerrar sesión.	
4.2		Se muestra la página de login de administrador.

Objetivo	Editar Reporte
Actores	Administrador (inicia).
Resumen	El administrador solicita la edición de un reporte, el sistema muestra una un vínculo sobre el identificador del reporte en la parte izquierda de la

	tabla para la edición del reporte.
Complejidad	Alta.
Prioridad	Crítica.
Precondiciones	Debe existir al menos un recurso para poder modificarlo o listarlo.
Postcondiciones	Se inserta, elimina, lista y modifica recursos.

Flujo de eventos

Flujo básico

	Actor	**Sistema**
1.	Selecciona el reporte que desea editar.	
2.		Muestra una nueva ventana con el mismo formulario de añadir reporte, pero relleno con la información del reporte seleccionado y da al administrador la posibilidad de modificar los parámetros.
3.	El administrador modifica los datos y da clic en el botón “Guardar”.	
4.		Finaliza el caso de uso.

Flujo alterno

Nº 1 “Cancela la operación”

	Actor	**Sistema**
1	El administrador da clic en el botón “Cancelar”	
		El sistema regresa a la página de inicio de administrador y muestra la tabla de

		los reportes recientes.

2.4. Diseño del sistema

Está relacionado con el diseño del sistema; se describen los patrones de diseño empleados. Se presentan los diagramas de clases del diseño, diagramas de interacción y diagrama de despliegue del sistema con el objetivo de tener una idea de cómo quedará.

Con el objetivo de crear una representación de *software* que proporcione detalles acerca de las estructuras de los datos, las arquitecturas, las interfaces y los componentes del *software* que son necesarios para implementar el sistema se elaboró un modelo del diseño siguiendo los principios definidos para realizar el diseño por la literatura científica.

2.4.1 Patrón arquitectónico

Los patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de software, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad. A continuación, se describen los utilizados en el desarrollo de la propuesta de solución.

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Propone que son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una y tienen un impacto en la arquitectura de subsistemas. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software (BUSCHMANN F et al. 1996).

Los patrones arquitectónicos:

- Definen la estructura básica de una aplicación.
- Pueden contener o estar contenidos en otros patrones.
- Proveen un subconjunto de subsistemas predefinidos, incluyendo reglas y pautas para su organización.
- Son una plantilla de construcción.

Patrón arquitectónico Modelo – Vista – Controlador

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.

La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como solución para un problema determinado. En la medida que los sistemas de software crecen en complejidad, dado por el número de requerimientos o por el impacto de los mismos, se hace necesario establecer medios para el manejo de esta complejidad. En general, la técnica es descomponer el sistema en componentes que agrupan aspectos específicos del mismo y que al organizarse, de cierta manera constituyen la base de la solución de un problema en particular.

MVC son las siglas de Modelo Vista Controlador, que es un patrón de arquitectura de software cuya función es subdividir una aplicación en tres módulos que corresponden a la vista del usuario (la interfaz a la que accede el usuario), una lógica de control para captar los eventos que el usuario ha generado a través de la interfaz, y un modelo que gestiona los datos según le indique la lógica de control.

Modelo: Esta es la representación específica de la información con la cual el sistema opera y se compone por el Sistema de Gestión de Base de Datos y la lógica de negocio. La lógica de negocio asegura la integridad de estos y permite derivar nuevos datos. El Sistema de Gestión de Base de Datos (SGBD) será el encargado de almacenar los cambios en los datos (agregar datos, editarlos o borrarlos) producidos por la lógica de negocio; ejemplos de SGBD son MySQL, Oracle... Es recomendable una capa de abstracción extra denominada *Data Access Object* (DAO), que es un componente de software que suministra una interfaz común entre la lógica de negocio y el SGBD.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Por lo tanto, la vista es la encargada de presentar los datos al usuario y la interfaz necesaria para modificarlos. Un ejemplo de tecnología podría ser las JSP que, mediante el servidor, muestra los datos y los formularios que constituyen la vista para que pueda interactuar con la aplicación.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Por lo general, el controlador sería la unidad central que comunica la vista con el modelo y viceversa, asociando los eventos del usuario con los cambios que se producirán en el modelo y devolviendo los datos resultantes que genere el modelo a la vista que corresponda.

El patrón arquitectónico Modelo – Vista – Controlador (MVC) divide una aplicación interactiva en tres componentes. El “modelo” contiene la información central y los datos. Las “vistas” despliegan información al usuario. Los “controladores” capturan la entrada del usuario.

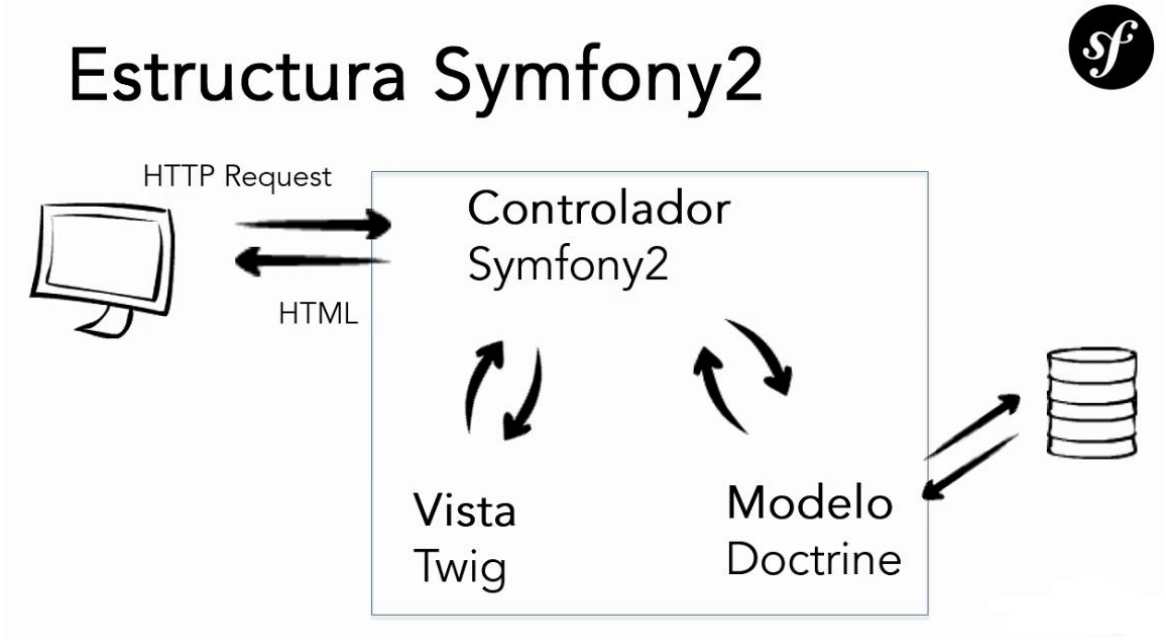

Figura 3 Patrón arquitectónico Modelo Vista Controlador

La implementación de MVC evidenciada en el sistema sigue el flujo de control siguiente:

- El usuario interactúa con la interfaz de usuario.
- El controlador recibe (a través de la interfaz) la notificación de la acción solicitada por el usuario. Es decir, el controlador gestiona el evento que llega desde la vista producida por un usuario.
- El controlador accede al modelo, ya sea con el fin de consultar datos o actualizarlos, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Por lo general, el controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar la orden a la vista para que se actualice. Sin embargo, en algunas implementaciones, la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.4.2 Diagrama de clases del diseño

"Los diagramas de clases del diseño (DCD) son los más utilizados en el modelado de sistemas orientados a objetos. Estos muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema y se obtiene como resultado del refinamiento del modelo conceptual" (JACOBSON, I. 2000). Los diagramas de clases muestran cómo se lleva a cabo la colaboración entre las clases para dar cumplimiento a un requisito determinado. Para la elaboración de estos diagramas se hace uso de estereotipos que ayudan a representar de manera más fácil la función y el carácter de las clases dentro de la realización del requisito. La Figura 4 muestra el diagrama de clases del diseño del caso de uso del sistema añadir reporte.

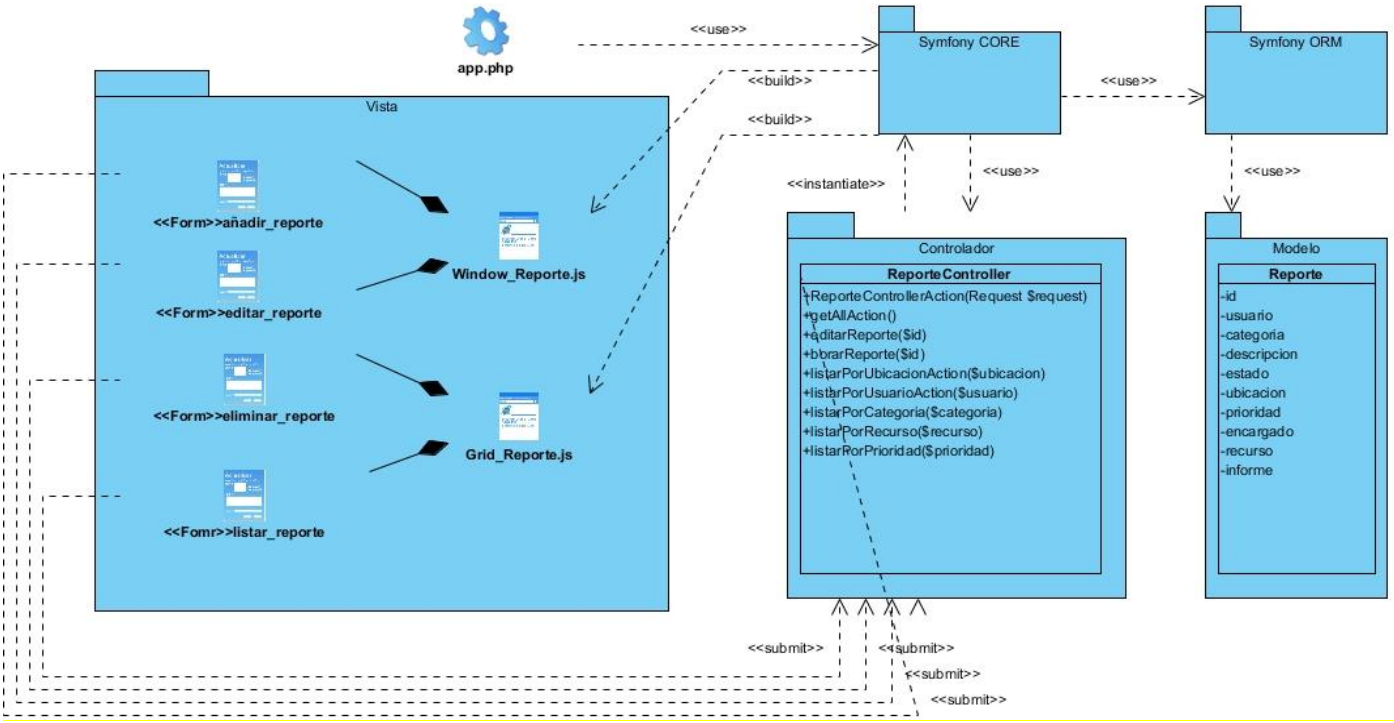

Figura 4 Diagrama de clases del diseño

2.4.3 Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de *software*, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes

en comunicación, que resuelve un problema general de diseño en un contexto particular (BUSCHMANN F ET AL. 1996).

Patrones GRASP

Los Patrones de Principios Generales para Asignar Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (CARDOSO E, 2004).

- **Controlador:** Este patrón queda evidenciado en el sistema haciendo uso de las clases controladoras para atender las peticiones realizadas por el usuario. Cada una de ellas se encarga de procesar la petición de su entidad correspondiente de forma independiente. A través del DCD de la Figura 4 se puede observar la evidencia de lo anteriormente planteado.

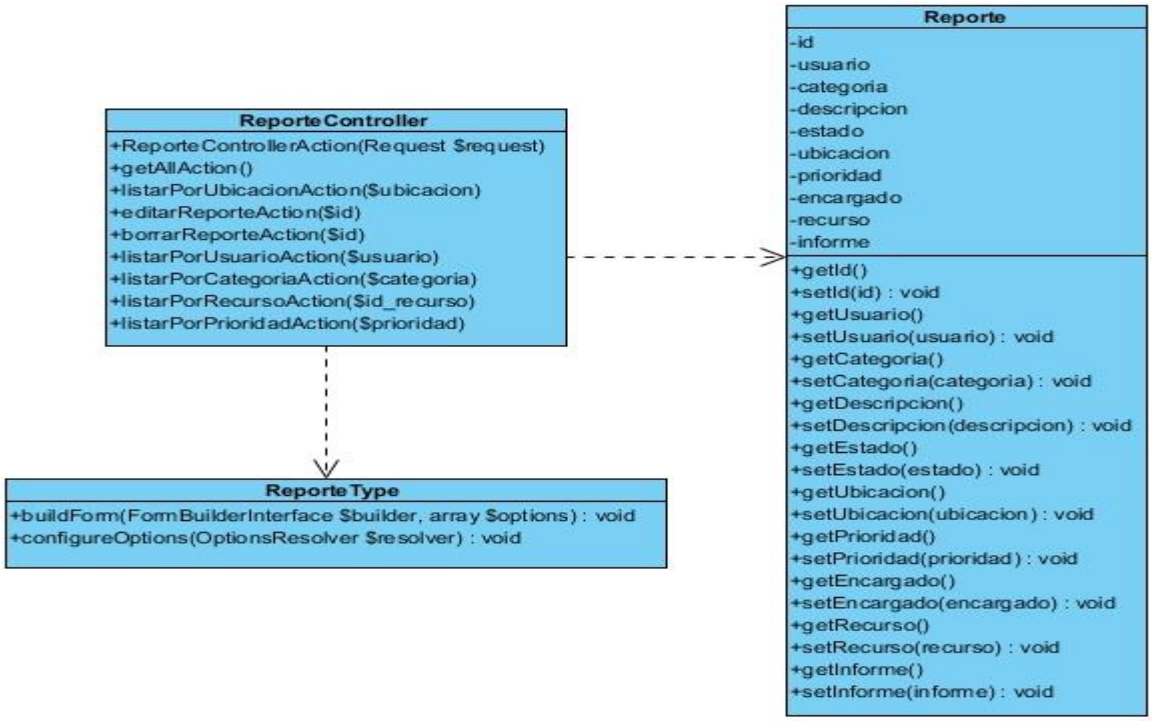

Figura 5 Comportamiento del patrón Controlador en el modelado del diseño

- **Experto:** Es uno de los patrones que más se utiliza cuando se trabaja con Symfony 2.8, con la inclusión de la librería Doctrine para mapear la Base de Datos. Symfony 2.8 utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos

poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

- **Alta Cohesión:** Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *Actions*, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las *properties*.

- **Bajo Acoplamiento:** La clase *Actions* hereda únicamente de *sfActions* para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Patrones Gof

Los patrones GoF (*Gang of Four* o “Pandilla de los Cuatro” en español), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

- **Instancia Única** (*Singleton*): Clase *sfRouting* – método *getInstance* Esta clase la utiliza el controlador frontal (*sfWebFrontController*) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. El *singleton sfRouting* precisa otros métodos muy útiles para la gestión manual de las rutas: *ClearRoutes* (), *hasRoutes* (), *getRoutesByName* ().
- **Decorador** (*Decorator*): Este método pertenece a la clase abstracta *sfView*, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado *layout.html.twig* es el que contiene el *layout* de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla. Este procedimiento es una implementación del patrón *Decorator*.

2.4.4 Diagrama entidad relación

El diagrama entidad relación muestra las relaciones existentes entre las clases del sistema, es una representación de las entidades de la base de datos.

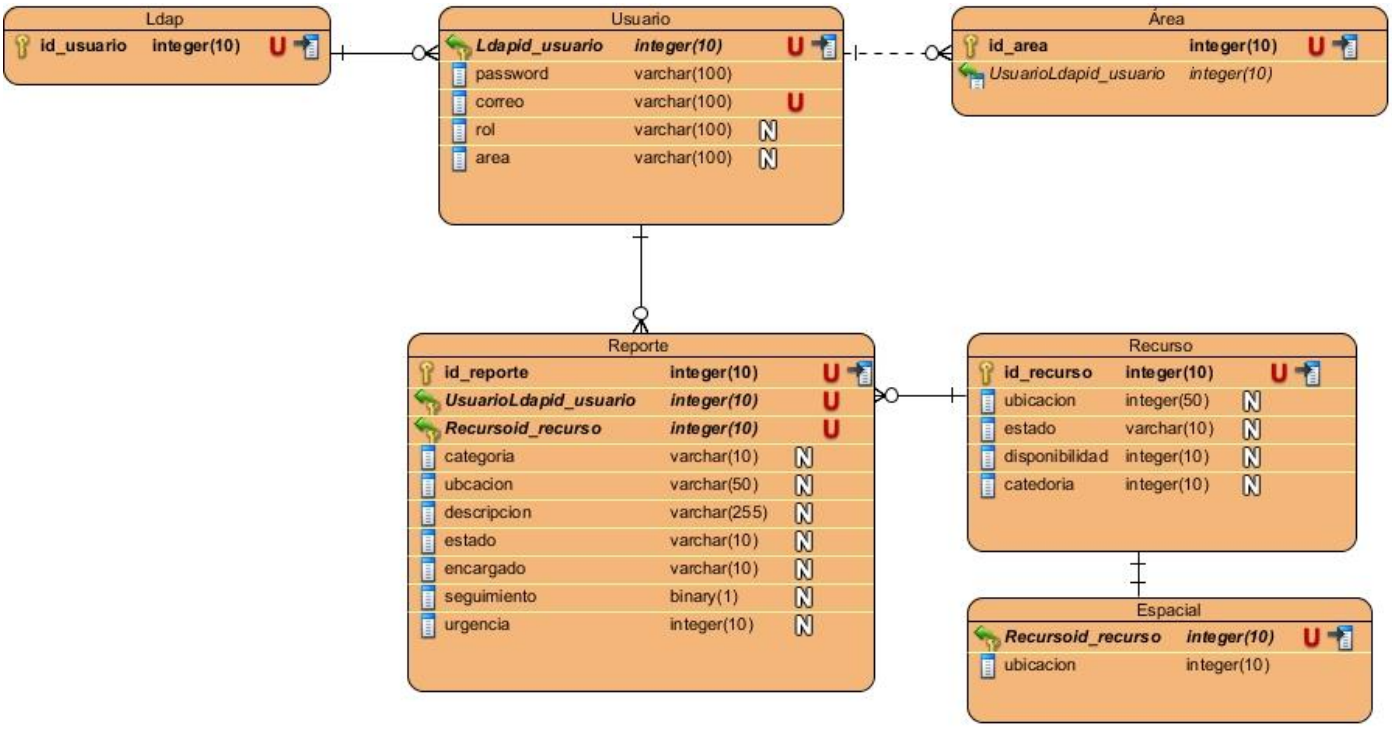

Figura 6 Diagrama Entidad Relación

2.4.5 Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema.

PC_Cliente: Representa las computadoras clientes que se conectan al servidor de aplicaciones, las mismas se comunican con el servidor a través del protocolo seguro HTTP.

Servidor web: Representa el servidor donde se encuentra instalada la aplicación web. Este accede al servidor de Base de Datos para el manejo de la información mediante el protocolo TCP/IP.

Servidor de Base de datos: Es donde se almacena toda la información de la aplicación.

TCP/IP: Protocolo para conectar el servidor de aplicaciones con las bases de datos.

HTTP: Protocolo de transferencia utilizado para conectar la computadora del cliente con el servidor donde está el sistema y este último con el servidor de los usuarios de la universidad (LDAP).

Servidor LDAP: Base de Datos donde se encuentran todos los usuarios de la universidad.

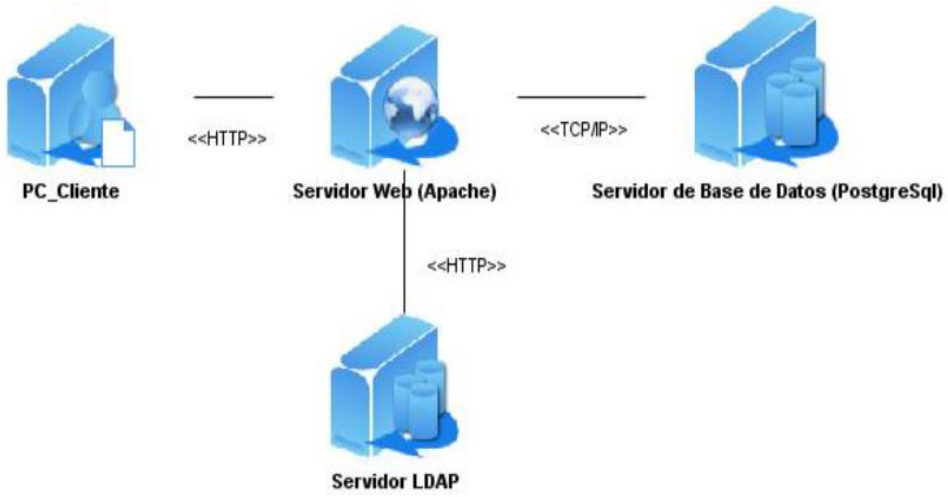

Figura 7 Diagrama de despliegue

2.4.6 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Se realizan con el objetivo de poseer una vista de forma general del sistema a partir de las dependencias e integraciones de los componentes y módulos.

- En el paquete controlador se encuentran las clases controladoras, encargadas de manejar las peticiones de los usuarios a través de los métodos que tienen implementados.
- El paquete modelo agrupa las entidades del sistema, a través de las cuales se realiza el acceso a la base de datos y el paquete vista, agrupa los archivos que permiten visualizar las respuestas que devuelven los controladores al usuario.

A continuación, se muestra el diagrama de componentes para el *DSGMainBundle*, donde se encuentran los principales componentes del sistema a desarrollar.

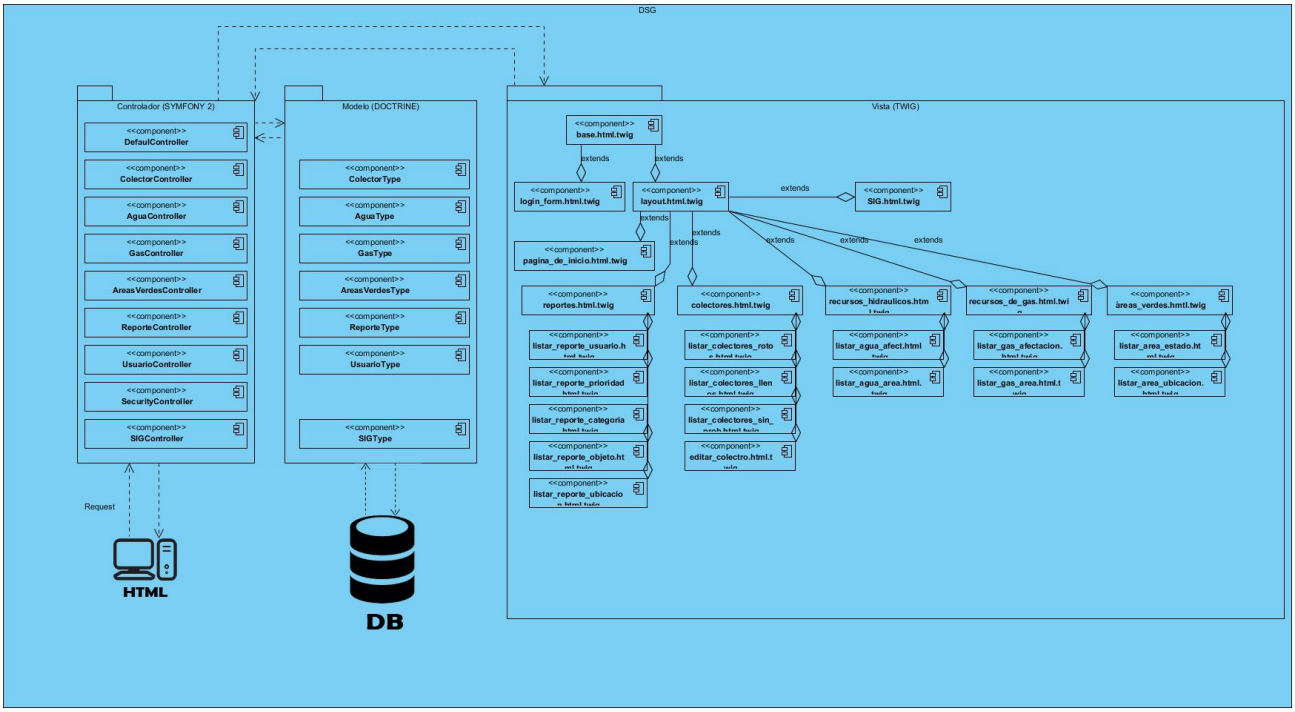

Figura 8 Diagrama de componentes

Conclusiones del capítulo

La utilización de un modelo de negocio bien definido y la generación de los artefactos del mismo, permitieron conocer, el funcionamiento de todos los procesos que se llevan a cabo en la DSG de la UCI. La realización del análisis y diseño del sistema a implementar permitió obtener una representación visual de las principales relaciones entre los conceptos asociados a la problemática a través del modelo de dominio.

Se identificaron 37 requisitos funcionales que fueron agrupados en 8 casos de uso que el sistema debe cumplir para su correcto funcionamiento; y 21 requisitos no funcionales que le aportan cualidades significativas al producto en cuanto a rendimiento, disponibilidad, accesibilidad, usabilidad, interfaz, entre otras. Para estructurar el sistema se utilizó el patrón arquitectónico MVC evidenciado en la estructura de clases, lo que facilitó la escalabilidad, adaptabilidad y el mantenimiento del mismo. La utilización de los patrones GRASP: Controlador, Creador y Bajo Acoplamiento; y GoF: Observador y Decorador; permitió obtener diseños de clases robusto al estructurarlos adecuadamente. El diseño del modelo entidad relación representado evidencia la descripción de la estructura de datos y sus restricciones.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

Introducción

Una de las últimas fases del ciclo de vida antes de entregar un software para su explotación es la fase de pruebas, cuyo objetivo es comprobar si este cumple sus requisitos. Dentro de la fase de pruebas pueden desarrollarse varios tipos de pruebas en función de los objetivos de las mismas y de su importancia.

3.1 Código fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido, como resultado se obtienen archivos que contienen el código fuente de la aplicación. El código fuente de un software es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está escrito su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos (LASSO 2008).

3.1.1 Estándares de codificación

Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Para facilitar el entendimiento del código y fijar un modelo a seguir, se establecieron estándares de codificación. A continuación, se muestran algunos de estos estándares utilizados en la implementación del sistema.

Nombres de clases y métodos

Para la definición de las clases y métodos en el código del sistema informático fueron utilizados los estándares de codificación *CamelCase* y *UpperCamelCase*. Estos son estilos de escritura que se aplica a frases o palabras compuestas.

Estructura

- El código debe usar cuatro espacios en vez de usar el tabulado. Esto minimiza problemas con otras herramientas de desarrollo.
- Las líneas deben tener 80 caracteres o menos, evitando tener más de 120 caracteres.

- Las llaves de apertura en las clases deben ir en la siguiente línea y la llave de cierre debe ir en la siguiente línea después del cuerpo.
- Las llaves de apertura, en los métodos de las estructuras de control debe ir en la misma línea y las llaves de cierre deben de ir después del cuerpo.
- Los paréntesis en las estructuras de control no deben usar espacios antes o después.
- Añadir un solo espacio después de cada limitador de coma.
- Añadir un solo espacio alrededor de los operadores (==, &&, …).
- Usa llaves para indicar el control de la estructura sin tener en cuenta el número de declaraciones que el grupo pueda contener.
- Definir una clase por fichero.
- Declarar las propiedades de clase antes que los propios métodos de clase.

3.2 Pantallas principales del sistema

La interfaz de una aplicación permite el flujo de información entre el usuario y el sistema. A continuación, se muestran algunos ejemplos de las interfaces del sistema desarrollado:

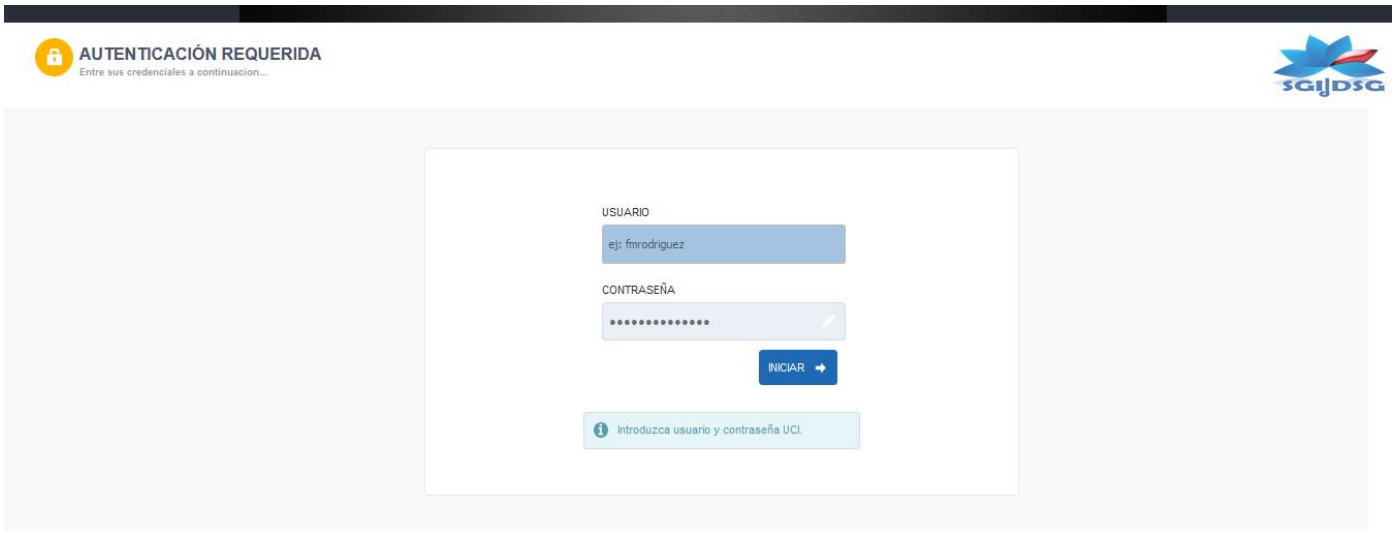

Figura 9 Página de acceso del SGIDSG

La Figura 9 muestra la página de acceso al Sistema de Gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas, donde le usuario UCI debe introducir sus credenciales y el sistema mostrará la pagina correspondiente al rol de usuario autenticado.

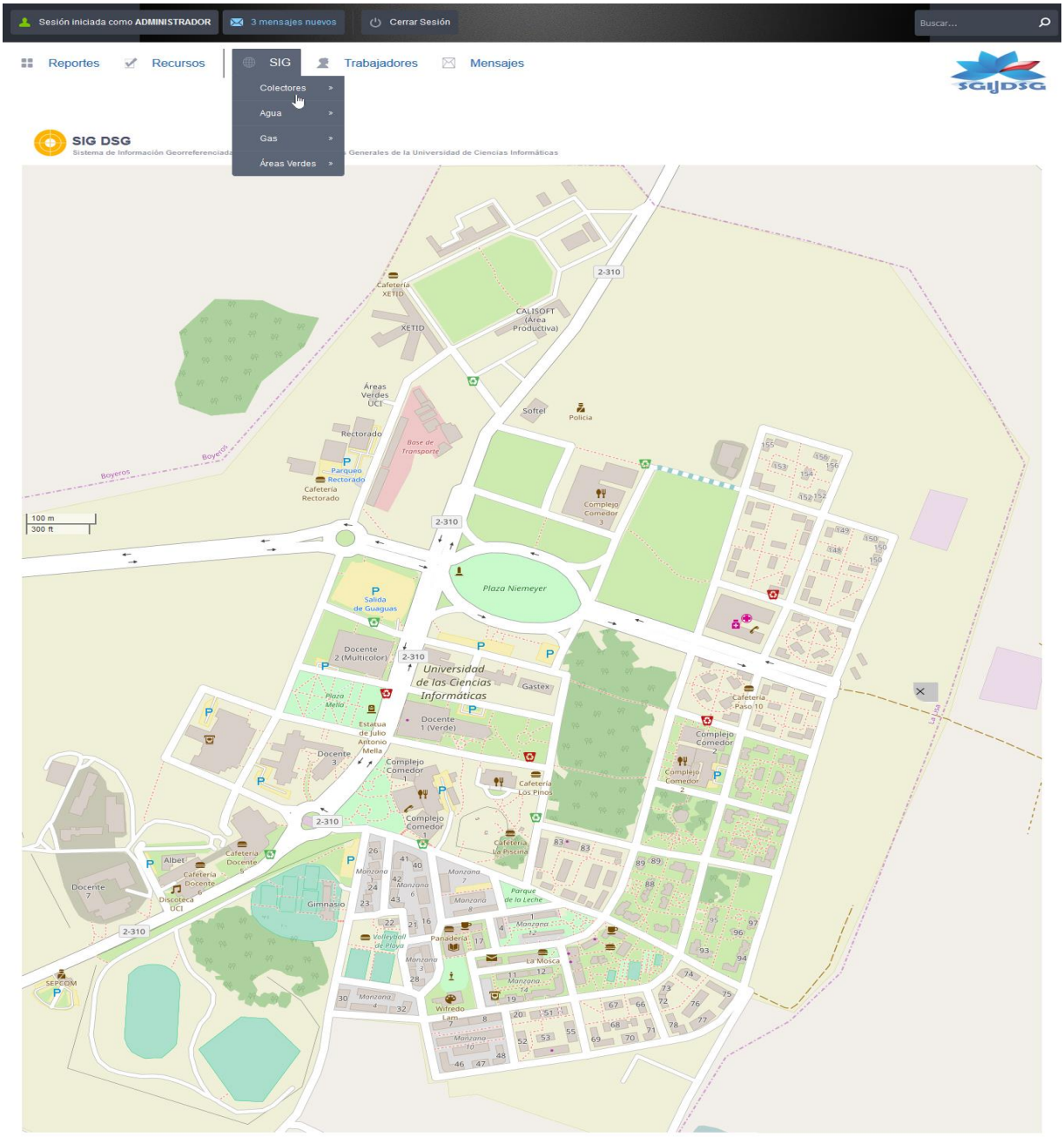

Figura 10 Página del SIG

La Figura 10 muestra la sección del SIG del Sistema de Gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas, donde este brinda la información georreferenciada de los recursos existentes en el sistema, así como la posibilidad de interactuar directamente con ellos, entre otras funcionalidades.

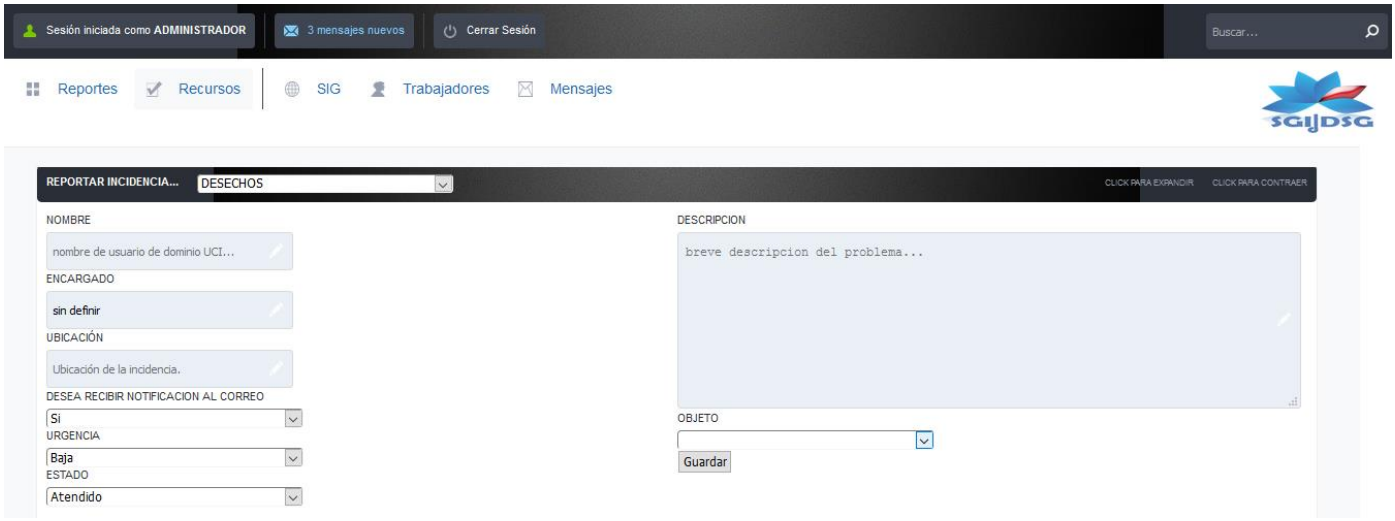

Figura 11 Página principal Añadir un nuevo reporte

La Figura 11 muestra la página para reportar incidencias del Sistema de Gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas, donde este brinda la posibilidad de añadir un nuevo reporte, entre otras funcionalidades.

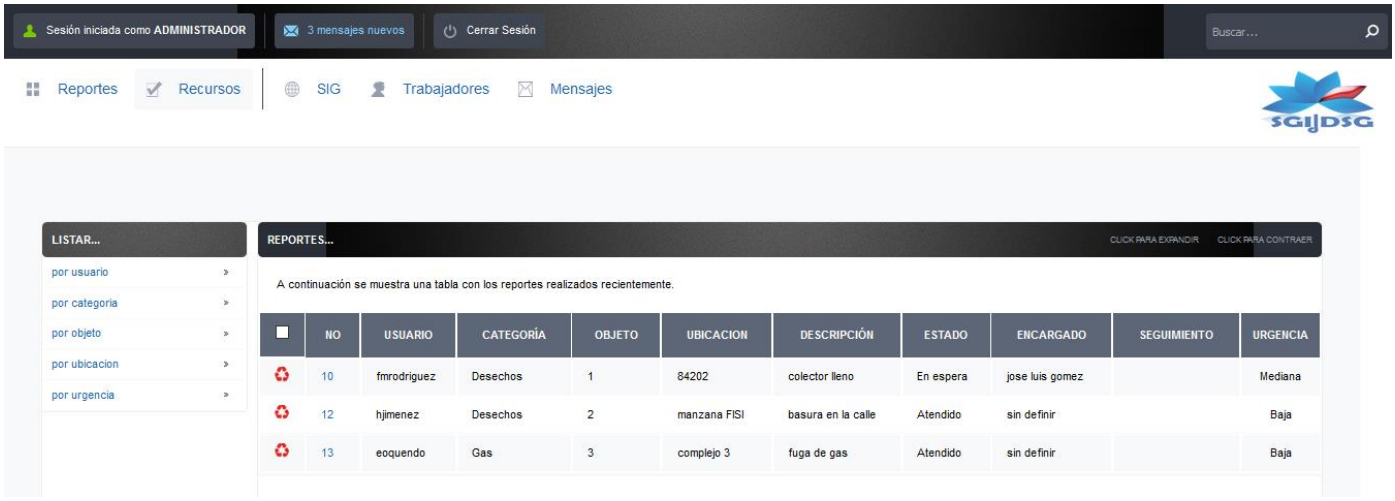

Figura 12 Página principal de Listar Reportes

La Figura 12 muestra la página principal que se observa en el Sistema de Gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas, al autenticarse un usuario

con rol de administrador de sistema, donde este brinda la información de los reportes añadidos recientemente y la posibilidad de listarlos de acuerdo a determinados parámetros.

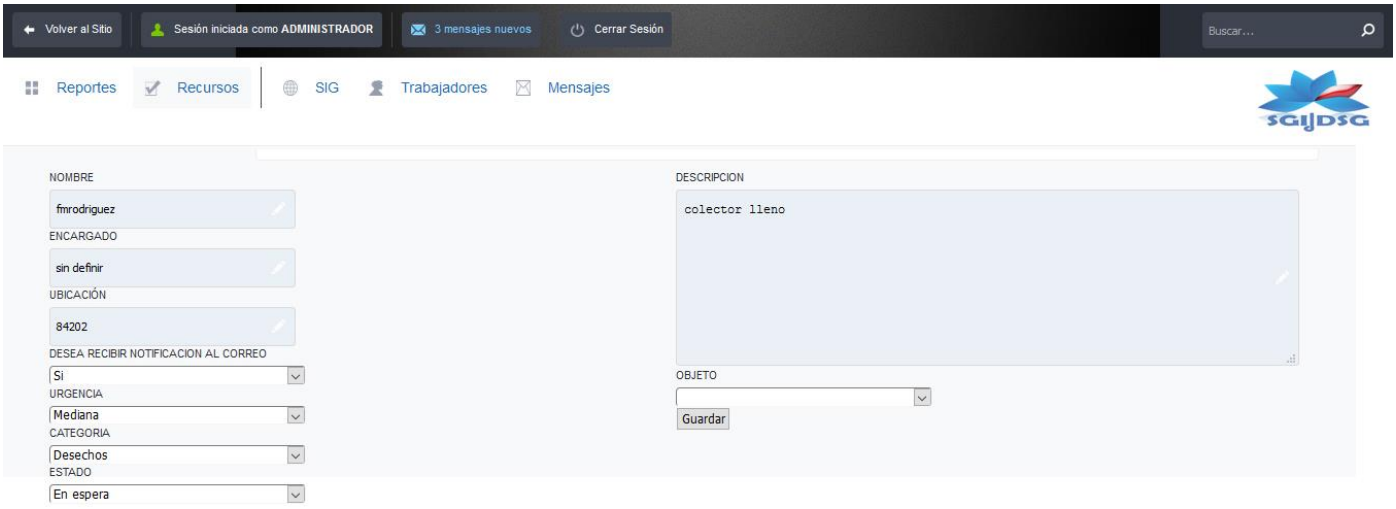

Figura 13 Página principal de Editar un Reporte

La Figura 13 muestra la página principal que se observa en el Sistema de Gestión de Incidencias e información georreferenciada para la Universidad de las Ciencias Informáticas, donde este brinda la posibilidad de editar un determinado reporte, entre otras funcionalidades.

3.3 Validación del sistema

Una vez terminada la implementación del producto que se requiere es necesario realizarle pruebas con el objetivo de detectar errores en la aplicación y la documentación; este proceso resulta de gran importancia ya que da una medida de la calidad del mismo, siempre que se lleve a cabo de la forma correcta. A continuación, se muestran las pruebas realizadas al sistema y los resultados obtenidos por cada una.

El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores. Además son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa (PRESSMAN, 2010).

Para dar inicio a las pruebas de un *software* lo primero es describir una estrategia de prueba donde quede plasmado los niveles de prueba a tratar, así como los tipos de prueba a emplear en cada nivel, los

métodos de prueba a aplicar, así como las técnicas a utilizar para cada método. Las estrategias de pruebas describen y verifican el enfoque de la misma.

Los **niveles de prueba** son diferentes ángulos de verificar y validar en determinados momentos el ciclo de vida del software. Existen diferentes niveles de pruebas como: pruebas funcionales, de sistema y de aceptación. En el desarrollo de la fase de pruebas del sistema para determinar el índice de control organizacional se aplicarán las pruebas que abarcan el siguiente nivel:

- **Desarrollador:** Se realiza con el objetivo de detectar errores en la implementación de requerimientos.

Existen diferentes **tipos de pruebas** que se pueden aplicar para comprobar el correcto funcionamiento de la aplicación. A continuación, se muestra el tipo de prueba seleccionado:

- **Pruebas de Funcionalidad**: Se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

3.3.1 Métodos de Prueba

Pruebas funcionales de caja negra

Las pruebas de caja negra, también denominadas pruebas funcionales se centran en los requisitos funcionales del software, es decir, la prueba de caja negra permite al ingeniero de software obtener los conjuntos de condiciones de entrada, que ejerciten completamente todos los requisitos funcionales de un programa.

Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

- Las pruebas funcionales que se realizarán a la solución, estarán enfocadas o dirigidas a los casos de uso del sistema para verificar su correcto funcionamiento.

- En este tipo de pruebas se ejecutarán los distintos servicios prestados con datos correctos e incorrectos; en caso de que los datos sean incorrectos se verificará que los mensajes de error sean los deseados y en el caso opuesto que los resultados sean los esperados (PRESSMAN, 2010).

Resultados e interpretación de las pruebas

El sistema implementado cuenta con un total de 39 requisitos funcionales, para probar cada uno de estos requisitos fue necesario elaborar un diseño de casos de prueba. Cada uno de las No Conformidades (NC) detectadas fueron registradas en el registro de NC del expediente de proyecto de la UCI y fueron debidamente clasificadas.

Se realizaron un total de 3 iteraciones de pruebas. Una primera iteración arrojó un total de 16 no conformidades clasificadas como se muestra en el siguiente gráfico:

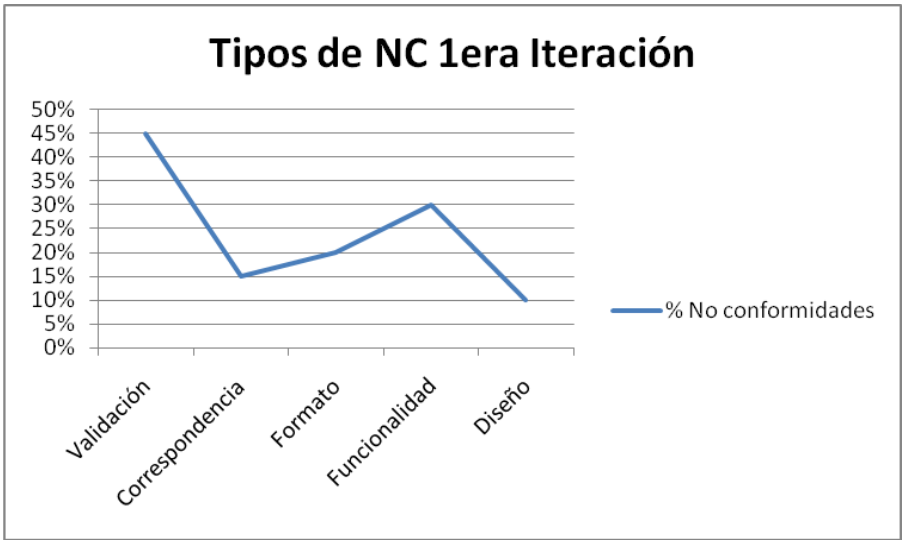

Figura 14 Tipos de NC 1era iteración

Las 16 NC detectadas fueron debidamente resueltas y se decidió hacer una segunda iteración de pruebas con el objetivo de verificar que no hayan quedado errores sin ser detectados y revisar que los cambios introducidos al resolver las NC de la primera iteración no habían introdujo nuevos errores.

En la segunda iteración se encontraron 7 NC clasificadas de la siguiente manera:

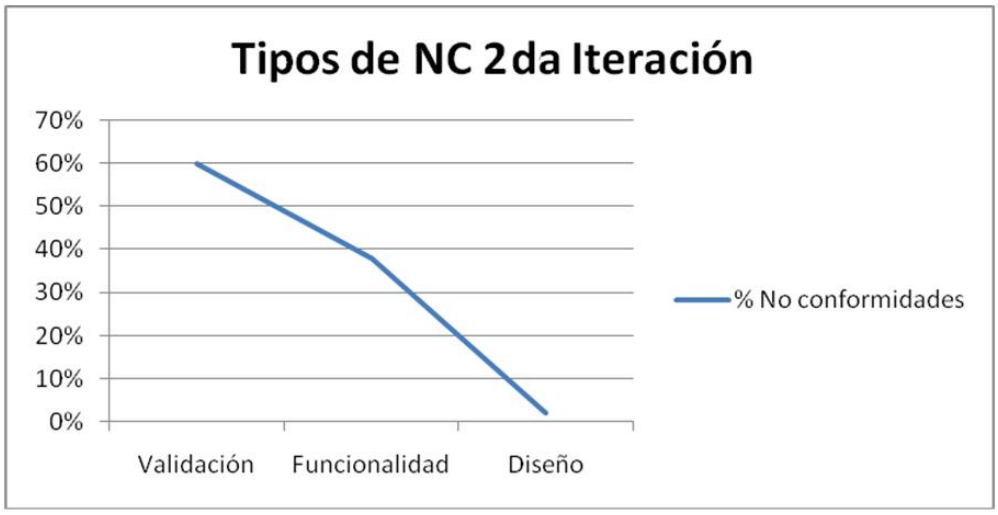

Figura 15 Tipos de NC 2da iteración

La tercera iteración de pruebas se realizó exitosamente y no se encontraron ninguna NC, garantizando de esta manera que fueran satisfechos el total de requisitos especificados por el cliente. Los resultados de las 3 iteraciones se muestran en el siguiente gráfico:

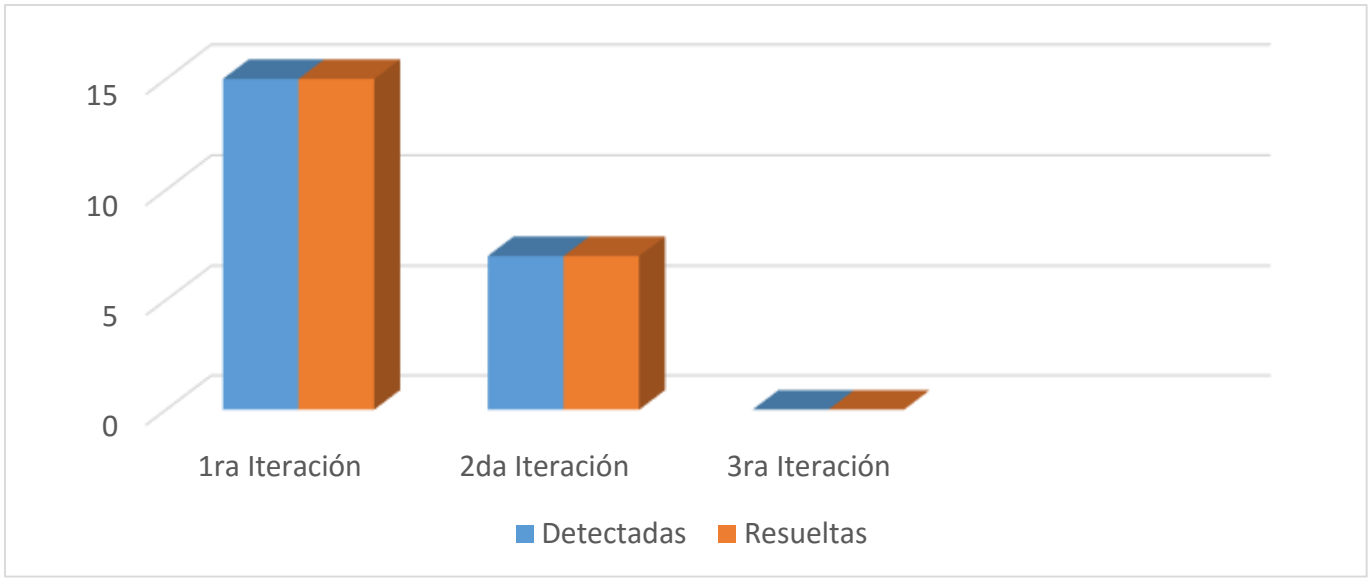

Figura 16 Iteraciones de las pruebas funcionales

Conclusiones del capítulo

Tras el flujo de implementación y prueba, el sistema quedó desarrollado. La descripción del empleo de los estándares de codificación y los estilos de programación permitieron a su vez hacer más fácil el entendimiento del código del programador y facilitar el mantenimiento futuro del sistema. Se cumplió con el objetivo planteado en el diseño teórico y se obtuvo como resultado un producto que cumple con las especificaciones emitidas en la DSG. Las pruebas realizadas permitieron elevar la calidad de la solución obtenida y corroboraron el correcto funcionamiento de la aplicación.

CONCLUSIONES GENERALES

Una vez concluida la investigación referida al mejoramiento de los procesos parar la gestión de incidencias en la Dirección de Servicios Generales de la Universidad de las Ciencias Informáticas, se obtuvo que:

- El estudio de diferentes fuentes bibliográficas, permitió obtener una visión de las principales ventajas y carencias que poseen los sistemas de gestión de incidencias con información georreferenciada y la necesidad de un sistema informático que garantice su correcto funcionamiento dando cumplimiento al objetivo de la investigación.

- Con la selección de la metodología desarrollo de software *OpenUP* se generaron varios artefactos, correspondientes a las fases: Modelado de Negocio, extracción de Requisitos, Análisis, Diseño e Implementación; permitió un mejor entendimiento de las funcionalidades a desarrollar, gracias a la gran documentación que presenta, sirviendo de base para la implementación del sistema informático.
- Las realizaciones de las pruebas de caja negra contribuyeron a garantizar que el sistema cumple con la calidad requerida para ser desplegado en la DSG. El sistema cumple adecuadamente con todos los requerimientos planteados por el cliente.

RECOMENDACIONES

El sistema para la gestión de incidencias en la Dirección de Servicios Generales de la Universidad de las Ciencias Informáticas requiere de seguimiento e incremento en las funcionalidades que puede brindar, se recomienda agregar nuevas funcionalidades al sistema como:

- ✓ Se recomienda realizar un estudio en el campo de la inteligencia artificial sobre la mejora de toma de decisiones en el sistema mediante juicio el de expertos.

REFERENCIAS BIBLIOGRÁFICAS

BALDUINO, R. 2007. Introduction to Open UP. Edtion ed.

BEDINI ALEJANDRO 2009. Metodología de Desarrollo de Software, Gestión de Proyectos de Software.

BUSCHMANN F, MEUNIER, R., ROHNERT, H., SOMMERLAND, P. y STAL M 1996. Pattern – Oriented Software Architecture. A System of Patterns. Londres.

CARDOSO E, CAMACHO F y NUÑEZ G 2004. Arquitecturas de software. Guía de estudio. S.l.

CARTOSIG 2017. GIS, Cartography and GeoSpatial Content. PostGIS 2. Análisis Espacial Avanzado.

CONTRALORÍA GENERAL DE LA REPÚBLICA 2009a. Folleto Sistema de Control Interno, Artículo 3. Disponible en: http://www.contraloria.gob.cu/index.php/documentos-rectores

CONTRALORÍA GENERAL DE LA REPÚBLICA 2009b. Folleto Sistema de Control Interno, Artículo 6. Disponible en: http://www.contraloria.gob.cu/index.php/documentos-rectores

CONTRALORÍA GENERAL DE LA REPÚBLICA 2009c. Folleto Sistema de Control Interno, Resolución No. 60/11. Disponible en: http://www.contraloria.gob.cu/index.php/documentos-rectores

CONTRALORÍA GENERAL DE LA REPÚBLICA 2009d. Reglamento de Ley 107 de la CGR, Artículo 83. Disponible en: http://www.contraloria.gob.cu/index.php/documentos-rectores

CORNEJO, J. E. G. 2008. ¿Qué es UML? El Lenguaje de Modelado Unificado. In.

CORNEJO, J. E. G. 2009. ¿Cuáles son las características que debe tener una herramienta UML? In.

CRISTALDO PATRICIA y KLOSTER MIRIAM 2016. Sistemas de Gestión.

ESRI ESPAÑA GEOSISTEMAS, S.A. 1991. ArcView.

FIGUEROA ROBERTH, SOLÍS CAMILO Y CABRERA ARMANDO 2014. Metodologías tradicionales vs. Metodologías ágiles.

GENBETA:dev 2014. NetBeans 8.1. Disponible en: http://genbeta.dev/tools/netbeans8.1

GONZALEZ ENRIQUE 2010. Aprender a Programar. PHP desde cero

INFANTE A. 2013. Desarrollo de un prototipo de software para el requerimiento de solicitudes de créditos. Universidad EAN facultad de ingeniería. Bogota.com.

INSTITUTO GEOGRÁFICO AGUSTÍN CODAZZI 2016. Fundamentos de Sistemas de Información Geográfica, Datos geográficos.

JACOBSON 2000. El Proceso Unificado de Desarrollo de Software. Madrid: Pearson Educación S.A. ISBN 8478290362.

JAVIER SILVA LUZ ALEXANDRA 2015. Sistemas de información geográfica y la localización óptima de instalaciones para residuos sólidos.

LARMAN, C. 2004. Modelo de caso de uso del sistema.

LARMAN, C. 2009. UML y patrones. Una introducción al análisis y diseño orientado a objetos. 2. S.l.: s.n.

LASSO IVAN 2008. Que es el código fuente. [en línea]. Disponible en: http://www.proyectoautodidacta.com/comics/que es el código fuente.

LOAYZA UYEHARA ALEXANDER ALBERTO 2015. Gestión de incidentes ITIL. Modelo de gestión de incidentes para una entidad estatal. Sistemas de Incidencias.

MARTÍNEZ, R. 2010. Sobre PostgreSQL. PostgreSQL-es. Disponibele en: http://www.postgresql.org.es/sobre_postgresql.POSTGIS 2.0

POTENCIER FABIEN y ZANINOTTO FRANCOISE 2010. The Definitive Guide to Symfony 2.8.

PRESSMAN, R.S. 2008. Ingeniería del Software. Un enfoque práctico. Sexta Edición. Edition ed.

PRESSMAN, R.S. 2010. Ingeniería del software, un enfoque práctico. 5. S.l.: Mc Graw Hill.

S., P. R., 2002. Ingeniería de Software, un enfoque práctico. 2002. Edition ed. McGraw-Hill Companies.

SANTOVENIA JAVIER, TARRAGÓ CONSUELO Y CAÑEDO RUBÉN 2009. Sistemas de información geográfica para la gestión de la información.

SUÁREZ NORBERTO 2014. Catálogo Objetos Geográficos y Símbolos del SGM (COGS-SGM).

WC3 2014. World Wide Web. Servidor Web. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.