Universidad de las Ciencias Informáticas

Facultad 1

Sistema de gestión de reportes sobre la actividad del centro de soporte de la Universidad de las Ciencias Informáticas.

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

Autor: Jorge Alonso Podadera

Tutores:
MsC: Neybis Lago Clara
Ing. Nelson Sánchez Álvarez

Habana, julio 2020

“Estamos ante un entorno laboral muy volátil. Aprende cada día a reinventarte, sé creativo, nunca dejes que el cortoplacismo apague tus grandes ideas”

Susan Bayle

DECLARACIÓN DE AUTORÍA

Yo Jorge Alonso Podadera, con CI: 97070407706 declaro ser el único autor de la presente tesis que tiene por título: Sistema de gestión de reportes sobre la actividad del Centro de Soporte de la Universidad de las Ciencias Informáticas y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firman la presente a los ____ días del mes de _____ del año _____.

Jorge Alonso Podadera

_____

Firma del autor

Neybis Lago Clara

_____

Firma del tutor

Nelson Sánchez Álvarez

_____

Firma del tutor

AGRADECIMIENTOS

Agradezco primeramente a mis padres Ernesto Alonso Rodríguez y Aida R. Podadera López, sin ellos no sería quien soy, no sería capaz de estar donde estoy. A mis amigos que me han apoyado en todo momento, a Mariana Machado Lemos por toda su ayuda. A todos los profesores y compañeros que de distintas formas contribuyeron al desarrollo de este trabajo y sin ellos no hubiese obtenido este resultado. A mis tutores que me apoyaron desde mis inicios en el Centro. A todos muchísimas gracias por su apoyo en todo este proceso.

DEDICATORIA

A mis padres, a mi familia y amigos, por todo su apoyo.

RESUMEN

En la Universidad de las Ciencias Informáticas, el Centro de Soporte brinda servicios para las entidades que contratan el soporte técnico a los productos de software. De las dudas e inquietudes de estas entidades se generan incidencias, las cuales necesitan ser atendidas rápidamente por los especialistas. Este trabajo actualmente se realiza sobre una plataforma útil pero deficiente, ya que no notifica sobre asignación de incidencias a especialista, no exporta reportes a pdf, entre otros procesos necesarios. Lo que trae como consecuencia la no disponibilidad o entrega tardía de la información a los clientes, comprometiéndose así la calidad del servicio de soporte. Para dar solución a esta problemática se propone la creación de un sistema de gestión de reportes sobre la actividad del centro de soporte de la Universidad de las Ciencias Informáticas. El desarrollo de la solución propuesta estuvo guiado por la metodología AUP-UCI empleando la arquitectura Modelo-Vista-Controlador para una mejor organización y comunicación entre los componentes. Se expone un estudio de sistemas homólogos que afirma la necesidad de implementar un sistema que reúna las funcionalidades necesarias ya implementadas en otros y aquellas no presentes en ellos, obteniéndose una aplicación que cumpla con todos los requisitos funcionales y no funcionales descritos en las historias de usuario. Para garantizar la calidad de la aplicación desarrollada se realizaron pruebas de caja blanca y caja negra, comprobando que esta cumple con el objetivo propuesto.

PALABRAS CLAVE: centro de soporte, entidades, incidencias, reporte, sistema de gestión.

Índice de contenidos

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las tecnologías de la información y las comunicaciones (TICs) se han constituido en elementos sustantivos inherentes al desarrollo de todas las esferas de la vida. A raíz de esto, ha cambiado la forma en que las empresas gestionan sus negocios, con el objetivo de mantenerse a un nivel competitivo. Es por ello que el crecimiento de las organizaciones es directamente proporcional al buen uso que hagan estas de las TICs para automatizar procesos claves y costosos dentro de su funcionamiento (Sánchez y otros, 2020).

Definidas por diferentes autores, las TICs se conocen como aquellas “tecnologías que permiten la adquisición, almacenamiento, procesamiento, evaluación, transmisión, distribución y difusión de la información. Dichas TIC son desarrolladas mediante la convergencia de la informática, las telecomunicaciones, la electrónica y la microelectrónica” (Sánchez y otros, 2020).

En el ámbito del procesamiento y tratamiento de la información se encuentran las TI (Tecnologías de la Información) como un subconjunto altamente utilizado en diferentes áreas del conocimiento. Un ejemplo específico de áreas donde las TI son un factor clave es en los servicios y la forma en que estos son gestionados. El concepto de gestión se comprende como “la capacidad de alcanzar lo propuesto, ejecutando acciones y haciendo uso de los recursos técnicos, financieros y humanos disponibles” (Botero y otros, 2009).

La incorporación de las TICs permite a una empresa competir eficientemente y tener la información disponible en el lugar y momento en el que se necesita por lo que éstas se han convertido en arma esencial para ofrecer productos y servicios con mayor calidad.

La calidad en el servicio al cliente no es un tema reciente dentro de las empresas, pues son la clave del éxito en la ejecución de cualquier servicio. Para lograr estar presente en un mercado que es cada vez más competitivo las empresas no son viables sin una apropiada atención al cliente y sin brindar un soporte adecuado a los servicios que ofrece. Las entidades u organizaciones tienen que apostar por la calidad de los servicios, debido a que es muy importante la forma en que el cliente percibe la calidad y los medios que existen para mantenerlo satisfecho. Estas prácticas se han acrecentado y diversificado llegando a cada lugar del mundo donde exista una empresa u organización, que comprenda entre sus objetivos el de brindar este tipo de servicios, teniendo una amplia representación incluso en América Latina.

Cuba no se ha quedado al margen de estos significativos cambios tecnológicos. Con la aprobación, en 1996, de los Lineamientos Generales para la Informatización de la Sociedad, el país se ha enmarcado en la ardua tarea de incorporar las TICs en todas las esferas de la sociedad.

"es imposible concebir el desarrollo de los pueblos al margen de las TICs aunque su avance debe tener en cuenta los principios de justicia, equidad e inclusión social" (Ramos, 2018).

La Universidad de las Ciencias Informáticas (UCI) es una de las instituciones encargadas de este proceso de informatización de la sociedad. Tiene como objetivo principal el "Formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos a partir del vínculo estudio – trabajo como modelo de formación – investigación - producción, sirviendo de soporte a la industria cubana de la Informática." (UCI, 2019).

Entre sus principales objetivos, la UCI destaca la producción de software a partir de su modelo de formación que incluye la vinculación estudio-trabajo. Para esto se apoya en diferentes centros creados por la Universidad, entre los que se encuentra el centro de soporte. Este centro se encarga de "brindar el servicio de soporte técnico a las aplicaciones y servicios informáticos desarrollados por la UCI, con calidad y eficiencia, a partir de su correcta gestión de los mismo y garantizando elevados niveles de satisfacción de sus clientes" (UCI, 2019).

El centro de soporte rige su trabajo por medio de una plataforma tipo HelpDesk[1], la cual controla todos los procesos del mismo. Zammad, la plataforma actual es un producto de Open Source[2], el cual es altamente eficiente para la gestión de incidentes tipo tickets[3], aunque no es un producto a la medida de las necesidades del centro.

Entre las funciones del centro se encuentra la gestión de incidencias[4], tener el control de su actividad en un proceso vital en su ciclo de vida, para ello es preciso la gestión de reportes, que garantiza la constante actualización del estado de las mismas para los clientes y los especialistas. En la plataforma no es posible efectuar reportes programados para notificar a los especialistas del estado y

[1] Helpdesk: Tipo de software donde se ofrecen servicios acerca de soporte técnico (errores, consultas, etc…).
[2] Open Source: "Código abierto" es un término para denominar a cierto tipo de software que se distribuye mediante una licencia que le permite al usuario final utilizar el código fuente del programa para estudiarlo, modificarlo y realizar mejoras en el mismo.
[3] Tickets: Un ticket es donde se encuentra toda la información relevante del incidente o petición como: los datos del cliente, los detalles de queja del cliente, cuándo ocurrió, etc.
[4] Gestión de incidencias: Una incidencia es toda interrupción o reducción de calidad no planificada del servicio. El objetivo de la gestión de incidencias es restaurar cuanto antes la operativa normal del servicio.

seguimiento de una incidencia, provocando como consecuencia una respuesta tardía por parte del personal encargado de darle solución a la incidencia en cuestión, comprometiéndose así la calidad del servicio de soporte.

Otro punto a destacar es la importancia de la existencia de reportes para la facturación, que requieren evidencias del trabajo durante períodos determinados. Con Zammad es posible realizar reportes con determinados intervalos de tiempo, pero el formato obtenido no es configurable, por lo que no es un producto presentable, provocando inconformidades al ser recibidos por los clientes.

También se debe medir la actividad laboral de los especialistas del centro y su interacción con el cliente, por lo que es imprescindible tener el control total de las acciones realizadas. Zammad no notifica a los especialistas sobre la asignación de incidencias, provocando una demora en el proceso de resolución de las mismas. Esta plataforma no cuenta con la posibilidad de incluir el historial de comunicación entre especialista y cliente dentro de los reportes.

Actualmente, todo este proceso se realiza manualmente por los especialistas, lo que trae consigo la posibilidad de errores humanos y empleo de mayor tiempo para la solución de problemas. Estas consecuencias inciden, además, en la disponibilidad de recursos o información para los clientes, comprometiéndose así la calidad de este servicio.

Tras analizar varios autores como Pilar Baptista Lucio y Carlos Álvarez de Zayas, el autor definió que para la presente investigación se utiliza lo expuesto por Carlos Fernández Collado y Roberto Hernández Sampieri, que plantea que en la elaboración del marco teórico queda definido de la siguiente manera (Hernández Sampieri, Fernández Collado, 2010):

Por lo anteriormente planteado se presenta como **problema a resolver:** ¿Cómo contribuir a la gestión de reportes en el Centro de Soporte? Considerando como **objeto de estudio** la gestión de reportes.

Se define como **campo de acción:** La gestión de reportes sobre la actividad del centro de soporte.

Para resolver este problema se plantea como **objetivo general** de la investigación: desarrollar una herramienta que permita la gestión de reportes sobre la actividad del Centro de Soporte de la Universidad de las Ciencias Informáticas.

Para guiar la investigación se definen las siguientes **preguntas científicas:**

1. ¿Cuáles son las bases teóricas que sustentan el desarrollo de los sistemas de gestión?

2. ¿Qué elementos asociados al desarrollo de software son necesarios para obtener un sistema de gestión de reportes sobre la actividad de soporte?
3. ¿Cómo verificar el funcionamiento del sistema de gestión de reportes propuesto?

Para dar cumplimiento al objetivo general se definen las siguientes **tareas de la investigación:**

1. Elaboración del marco teórico para orientar la investigación de los sistemas de gestión y obtener una conceptualización adecuada de los términos a utilizar.
2. Elaboración del estado del arte asociado a los sistemas de gestión de reportes para demostrar la novedad y originalidad del sistema a realizar.
3. Análisis de las tecnologías y metodologías de desarrollo para el diseño de sistemas de gestión.
4. Análisis de las herramientas de desarrollo para la implementación del sistema de gestión.
5. Implementación del sistema de gestión, de acuerdo a la metodología AUP-UCI.
6. Realización de las pruebas de caja blanca y caja negra para verificar el correcto funcionamiento del sistema de gestión.

Métodos Teóricos:

- **Analítico-Sintético:** Permitió la descomposición de un todo complejo en sus partes y cualidades. La síntesis, por su parte, estableció la unión entre las partes, previamente analizadas y posibilitó descubrir relaciones y características generales entre los elementos de la realidad. Este método se utilizó para seleccionar las herramientas y tecnologías utilizadas durante el desarrollo del sistema de gestión, además permitió la evaluación de otras soluciones que respondieran al problema, lo que permitió realizar una valoración crítica y detallada de cada una de ellas, aportando nuevas funcionalidades al sistema de gestión de reportes.

Métodos Empíricos:

- **Observación:** Se utilizó para apreciar los resultados obtenidos. Es uno de los métodos más utilizados en la investigación científica, debido a que es un procedimiento fácil de llevar a cabo y que exige técnicas de tabulación muy sencillas. De la misma forma permitió percibir

directamente, sin intermediarios que deformen la percepción, los hechos de la realidad objetiva, con lo cual se eliminaron las deformaciones de otros métodos indirectos. Fue utilizado principalmente para ver cómo se hace el proceso de gestión de reportes en el centro.

- **Entrevista:** Se utilizó para realizar recopilación de información mediante una conversación profesional dando respuesta a un cuestionario, de esta forma además se obtuvo información acerca del proceso de gestión de reportes en el Centro de Soporte, teniendo importancia educativa; y dependiendo en gran medida del nivel de comunicación entre el investigador del centro y los participantes en la misma. Se realizó con el propósito de obtener información, puntos de vistas e ideas que contribuyeran al desarrollo de la investigación y aportaran conocimientos específicos del centro.

La **estructuración del contenido** queda dividida en tres capítulos:

Capítulo 1. Fundamentación teórica del sistema de gestión de reportes.

En este capítulo se exponen los antecedentes que permitieron la comprensión del estudio del problema, teniendo en cuenta para esto el estudio de sistemas de gestión homólogos y las definiciones de interés relacionadas con la investigación. Además, se realiza un análisis sobre las principales tecnologías, herramientas y metodologías que se aplican en el desarrollo de los sistemas de gestión de reportes.

Capítulo 2. Análisis y diseño del sistema de gestión de reportes.

En este capítulo se presentará el modelo de dominio, se describirá la solución propuesta teniendo en cuenta la concepción del sistema de gestión, la visión y el alcance del mismo. Se especificarán los requisitos funcionales y no funcionales que debe cumplir el sistema, realizando una descripción de estos a través de las historias de usuario. Además, se definen los patrones de diseño y se elabora el diagrama de entidad-relación correspondiente al sistema de gestión de reporte.

Capítulo 3. Implementación y pruebas realizadas al sistema de gestión de reportes.

En este capítulo se muestra el modelo de implementación, así como el diagrama de componentes del sistema de gestión. Se dan a conocer los estándares de codificación por los que se regirá el código y se describen las pruebas de caja blanca y caja negra que permiten comprobar el funcionamiento correcto del sistema de gestión, incluyendo en estas las pruebas de verificación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DEL SISTEMA DE GESTIÓN DE REPORTES.

En el presente capítulo se exponen los aspectos teóricos sobre los sistemas de gestión de reportes. También se abordan aspectos importantes de soluciones existentes a nivel nacional e internacional, que se pudieran tener en cuenta para un mejor desarrollo de la aplicación. Y por último se describen y caracterizan la metodología, las herramientas y los lenguajes utilizados en la presente investigación.

1.1 Elementos asociados al dominio del problema

A continuación, se relacionan los principales conceptos o temáticas que están asociados al desarrollo de la investigación.

Aplicaciones Web

En la Ingeniería de Software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un Servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación (software) que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador (Talledo, 2015).

(Ramos, 2014) las define como: Herramientas de software que se utilizan para crear y editar textos, hojas de cálculo y presentaciones de manera online a través de la red, utilizando un navegador de internet y sin necesidad de instalar las aplicaciones ofimáticas habituales de escritorio.

Por su parte (Berzal y otros, 2011) plantean que las aplicaciones web permiten generar dinámicamente el contenido que finalmente se les ofrece a los usuarios, por lo que es posible seleccionar, filtrar, ordenar y presentar la información de la forma más adecuada en función de las necesidades de cada momento.

Al analizar los conceptos expuestos anteriormente, el autor de la presente investigación considera que el que más se ajusta a la investigación y sus objetivos es la definición propuesta por (Berzal y otros, 2011).

Sistema de gestión

Un sistema de gestión es una herramienta que permitirá optimizar recursos, reducir costes y mejorar la productividad en una empresa. Este instrumento de gestión reportará datos en tiempo real que permitirán tomar decisiones para corregir fallos y prevenir la aparición de gastos innecesarios (De la Peña, 2015).

FUNDAMENTACIÓN TEÓRICA DEL SISTEMA DE GESTIÓN DE REPORTES

Los sistemas de gestión están basados en normas internacionales que permiten controlar distintas facetas en una empresa, como la calidad de su producto o servicio, los impactos ambientales que pueda ocasionar, la seguridad y salud de los trabajadores, la responsabilidad social o la innovación (De la Peña, 2015).

El Modelo Europeo de Excelencia Empresarial (EFQM) los define como un "Esquema general de procesos y procedimientos que se emplean para garantizar que la organización realiza todas las tareas necesarias para alcanzar sus objetivos. Mientas que (Ogalla, 2005) lo define como un conjunto de procesos, comportamientos y herramientas.

Teniendo en cuenta los conceptos analizados anteriormente y las características de la investigación, el autor de este documento considera que la que más se ajusta es la definición dada por (De la Peña, 2015).

Reporte

Un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital, audiovisual, etc.) pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y de otros tipos. El reporte puede ser la conclusión de una investigación previa o adoptar una estructura de problema-solución en base a una serie de preguntas. En el caso de los informes impresos, el texto suele ir acompañado por gráficos, diagramas, tablas de contenido y notas al pie de página (Pérez, 2010).

En el ámbito de la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios (Pérez, 2010).

El autor de este documento considera que la definición dada por (Pérez, 2010) se ajusta completamente a las características y objetivos de la investigación y decidió solamente referirse a ella.

Cliente

Es un concepto que procede del latín '*cliens*' y que refiere a la persona que accede a un determinado producto o servicio tras concretar un pago. Pese a que existen los clientes ocasionales, el término suele aplicarse a aquellos que acceden al producto o servicio con asiduidad. Un cliente, por lo tanto, puede ser sinónimo de comprador (la persona que compra el producto), usuario (la persona que usa el servicio) o consumidor (quien consume un producto o servicio) (Paz, 2007).

Un cliente es alguien que ha sido impactado por un producto, es el destinatario de un servicio o producto ofrecido por un suministrador, en una situación contractual, el cliente se denomina comprador, pero a la vez puede ser consumidor final, usuario o beneficiario; según plantea (Juran, 1996).

Al analizar los conceptos citados anteriormente, el autor considera que el que más se ajusta a la investigación es el dado por (Juran, 1996).

Soporte técnico

La noción de soporte se utiliza para nombrar a algo que brinda un respaldo, que puede ser físico o simbólico. Lo técnico, por otra parte, se asocia a aquello que se aplica en la ciencia o una disciplina artística (Znet,2020).

El soporte técnico, por lo tanto, es una asistencia que brindan las empresas para que sus clientes puedan hacer uso de sus productos o servicios. La finalidad del soporte técnico es ayudar a los usuarios para que puedan resolver ciertos problemas (Znet,2020).

Al analizar el concepto aportado por (Znet,2020), el autor considera que se ajusta completamente a los objetivos de su investigación por lo que decidió solo referirse a este.

Actividad

Se conoce actividad como las acciones que desarrolla un individuo o una institución de manera cotidiana, como parte de sus obligaciones, tareas o funciones (Pérez, 2015).

(Bastida, 2009) la define como: Facultad de obrar. Conjunto de tareas propias de una persona o entidad. Capacidad de hacer o actuar sobre algo.

Teniendo en cuenta los conceptos anteriores el autor del documento considera que el dado por (Pérez, 2015) es el que más se ajusta a los objetivos de su investigación.

1.2 **Estudio de soluciones existentes**

En el presente epígrafe se realiza una descripción de los sistemas HelpDesk que existen actualmente a nivel nacional e internacional. Para ello se tienen en cuenta las ventajas y desventajas de cada uno de ellos, aportando así requisitos indispensables a la propuesta de solución.

Sistema gestión de reportes en Zammad

Es un sistema gratuito de código abierto de tickets basado en web para el servicio de asistencia al cliente. Se envía con una multitud de funciones para manejar la comunicación con el cliente a través de varios canales, como redes sociales (**Facebook** y **Twitter**), chat en vivo, correos electrónicos y

teléfono. Tiene una API para integrar su sistema telefónico en las llamadas entrantes y salientes. Puede atender fácilmente las consultas y quejas de los clientes. Zammad viene con un amplio conjunto de características que incluyen búsqueda de texto completo, guardado automático, auditable, admite autenticación externa, admite varios idiomas y mucho más. Se distribuye bajo la Licencia pública general de GNU AFFERO (AGPL) y se puede instalar en múltiples plataformas como Linux, AIX, FreeBSD, OpenBSD y MacOSX. Está disponible en Github y se puede instalar gratis en su propio servidor (Zepeta, 2017).

Zammad trabaja en conjunto con Elasticsearch el cual es un servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON (Zepeta, 2017).

Sus desventajas residen en que:

- No es posible efectuar reportes programados para notificar a los especialistas del estado y seguimiento de una incidencia.
- Formato de reportes no configurable (los campos están en inglés, falta la descripción del problema, entre otros).
- No exporta reportes a pdf.
- No notifica sobre asignación de incidencias.
- No cuenta con la posibilidad de incluir historial de comunicación en los reportes.

Al analizar estas desventajas se puede apreciar que el sistema Zammad carece de funcionalidades necesarias pues en este no es posible efectuar reportes programados, el formato de reportes que ofrece no es confiable y no notifica sobre la asignación de incidencias.

Sistema gestión de reportes en ServiceDesk

Es una división del fabricante norteamericano Zoho Corp y uno de los líderes del mercado en software para la monitorización de infraestructuras. Los productos ManageEngine siempre ofrecen funcionalidad de alta calidad, a precios muy asequibles. Su estrategia de bajos costes y amplia funcionalidad ha convertido ManageEngine en uno de los fabricantes de mayor crecimiento en los últimos años (Orueta, 2010).

ManageEngine ServiceDesk Plus, anteriormente utilizado en el centro, es revolucionario porque cambia el modo de trabajo de los equipos de TI: de ser bomberos para solucionar los problemas del día a día a ser los responsables de entregar un increíble servicio al cliente. Brinda una gran visibilidad y un control central al tratar con los problemas de TI para garantizar que el negocio no sufra momentos de inactividad. Entre sus funciones está la gestión de incidentes, la gestión de problemas,

la gestión del cambio, opciones de creación de catálogos de servicios, monitorización y administración de todos los elementos de configuración y la gestión de proyectos TI. (Orueta, 2010)

Este sistema cumple con las necesidades del centro de soporte técnico de la UCI, pero es privativo por lo que resulta muy difícil obtener y renovar la licencia, el código no puede ser adaptado a las necesidades del centro de soporte y presenta altos costos, por lo tanto, no es viable.

Sistema gestión de reportes en Odoo Helpdesk

Odoo es un conjunto de aplicaciones empresariales de código abierto que ayuda a las empresas a hacer crecer su negocio. Más de 2 millones de personas usan Odoo para aumentar sus ventas, administrar sus operaciones, organizar actividades de marketing, aumentar la productividad y potenciar sus recursos humanos. Odoo proporciona una solución de software completa para cada nivel de su negocio (Sembiring, 2016).

Su conjunto de aplicaciones empresariales integradas es: CRM, POS, creador de sitios web, comercio electrónico, ventas, facturación, contabilidad, fabricación, almacén, recursos humanos, proyectos y herramientas de marketing.

Cuenta con un módulo para brindar un servicio de asistencia ágil con una excelente interfaz de usuario con posibilidad de monitorear, priorizar y resolver solicitudes de asistencia de los clientes. Posee múltiples canales, email, formularios de sitio web, twitter y chat directo, asegurando así la disponibilidad (Sembiring, 2016).

El servicio de asistencia de Odoo está optimizado para la productividad su equipo:

- Cree sus normas de acuerdo al nivel de servicio y permita que Odoo actúe automáticamente.
- Automatice los correos electrónicos o las acciones en diferentes etapas de la resolución de las solicitudes de asistencia.
- Eleve las solicitudes de asistencia a su gerente con solo un clic.
- Utilice respuestas prediseñadas en el chat en vivo para respuestas instantáneas.
- Define dinámicas plantillas de reportes para automatizar las respuestas más comunes.
- Puede comentar cada reporte para imprimirlos y reportar a su asesor.
- Exporta reportes a xls para extra análisis.

Su desventaja reside en que, aunque el software en su versión community (con ciertas limitaciones) es gratis, la implementación no lo es, por lo que si hay que invertir. Sus precios varían en dependencia de los servicios y la cantidad de usuarios, un usuario con servicio helpdesk debería pagar 18 euros mensuales. Otras versiones como Odoo Enterprise (privativas) presentan beneficios

notables como un soporte funcional ilimitado, actualizaciones de versiones, etc. Además, la compañía puede dejar de mantener Odoo en su versión comunitaria o cambiar la licencia (Sembiring, 2016).

Este producto no cumple con los requerimientos del centro dada su inestabilidad en el tema de la licencia. Además, se requiere de pagos para acceder a algunos de sus servicios. Por lo que no es una solución viable.

Sistema gestión de reportes en NgDesk

Es un software gratuito de gestión de tickets que no ofrece planes de pago. Todas sus funciones se encuentran disponibles de manera gratuita para un número ilimitado de agentes. Ofrece todo lo que necesario e incluye respuestas predefinidas, chat privado entre agentes y horarios de guardia. Las políticas personalizadas de derivación automática permiten decidir de antemano cuándo escalar los tickets, así como los intervalos de tiempo entre derivaciones, para garantizar que los casos no resueltos sigan derivándose hasta que alguien del equipo responda (Valenzuela, 2019).

El panel de ngDesk muestra la carga de trabajo, los informes, los días de mayor actividad y el estado de los clientes más importantes en tiempo real.

Los agentes pueden trabajar sobre la marcha gracias a las aplicaciones móviles y recibir notificaciones y actualizaciones de tickets por teléfono, correo electrónico o SMS (Valenzuela, 2019).

NgDesk también permite definir las incidencias y supervisiones que se ejecutan en todos los tickets del centro de ayuda, compararlos con los de la empresa a intervalos predefinidos y recibir notificaciones sobre incidencias en riesgo de incumplimiento o vencimiento (Valenzuela, 2019).

El mayor inconveniente es el desconocimiento. Su página web no ofrece mucha información sobre la atención al cliente que puede esperar el usuario. Según usuarios, NgDesk se considera perfecto por contar con tantas funciones gratuitas, pero a largo plazo llega a saturar tanta información y puede ser algo confuso.

Sistema gestión de reportes en Freshdesk

Ofreciendo soporte multicanal, reúne todas las conversaciones de los clientes en una interfaz centralizada. Esto ayuda a los agentes de soporte a abordar y resolver los tickets eficazmente. Tanto si la interacción se produce por teléfono, como si se realiza por correo electrónico, chat, web o a través de redes sociales como Facebook y Twitter, Freshdesk se asegura de que las incidencias del cliente lleguen al miembro de soporte que corresponda (Gómez, 2013).

Debido a que el sistema helpdesk de freshdesk está totalmente basado en la nube, ayuda a administrar múltiples grupos, flujos de trabajo y procedimientos de escalamiento de incidencia. También es compatible con múltiples productos y marcas e incluye varios idiomas y zonas horarias.

Automatiza múltiples procesos. El sistema enruta el ticket al agente adecuado. Mientras, el supervisor puede controlar el estado de todos los tickets abiertos en la plataforma y, en caso de ser necesario, avisar a gerencia de que no se está cumpliendo el nivel de servicio establecido (Gómez, 2013).

Freshdesk también ofrece funciones orientadas al cliente como un portal de autoservicio, una base de conocimientos y foros para que los clientes puedan ver el estado de sus tickets, encontrar soluciones en línea y compartir o incluso votar ideas (Gómez, 2013).

Al igual que **ManageEngine,** el sistema cumple con las necesidades del centro de soporte técnico de la UCI, pero es privativo, presentando altos costos, resulta muy difícil conseguir y renovar la licencia y el código no puede ser adaptado a las necesidades del centro de soporte, por lo tanto, no es viable.

Luego de analizar cada uno de los sistemas homólogos, se hace evidente la necesidad de la implementación de un nuevo sistema capaz de satisfacer las necesidades del centro de soporte pues a pesar de que estos permiten realizar búsquedas de texto completo, guardar automáticamente, admiten autentificación externa y otras funcionalidades de gran utilidad, carecen de otras que no permiten que se satisfagan completamente los requisitos del cliente: no permiten exportar a pdf y el formato de reporte no es configurable, entre otras. Las funcionalidades y características presentes en estos sistemas que resultan provechosas se tendrán en cuenta en el diseño e implementación del nuevo sistema a desarrollar.

1.3 Tecnologías y herramientas

En este epígrafe se analizarán algunas de las tecnologías y herramientas que se utilizarán en la construcción de la aplicación para que esta cumpla con todos los requisitos necesarios. Luego de estudiar estas herramientas y metodologías se llegó a una propuesta final, las cuales se mostrarán a continuación.

El usuario al ingresar a una dirección en línea realiza una petición que viaja por el internet hacia el servidor en donde está alojada la plataforma que desea visitar.

Al haber interacción entre los equipos de los usuarios y los servidores en internet, se debe tomar en cuenta las distintas tecnologías para ambos extremos, el lado del usuario se le conoce como "f*rontend*" y el lado del servidor se le conoce como "*backend*" (Platzi,2017).

Tecnologías utilizadas en el *backend*

Python

Python es un lenguaje de programación interpretado de tipado dinámico[5] cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas (Foundation,2017).

Dicho de otro modo, Python es (Covantec., 2018):

- Interpretado: Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
- Multiparadigma: Soporta programación funcional, programación imperativa y programación orientada a objetos.
- Tipado dinámico: Las variables se comprueban en tiempo de ejecución.
- Multiplataforma: disponible para plataformas de Windows, Linux o MAC.
- Gratuito: No dispone de licencia para programar.

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero (Foundation,2017).

Django

Django es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. Un framework web es un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente (Holovaty, 2009).

Cuando se construye un sitio web, siempre se necesita un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para tu sitio web, formularios, una forma de subir archivos, entre otros.

Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio “No te repitas” (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos (Holovaty, 2009).

Tecnologías utilizadas en el frontend

JavaScript

[5] Tipado dinámico: Lenguaje de programación que realiza la comprobación de tipificación durante su ejecución en vez de durante la compilación.
Tipificación: Clasificación en tipos o clases.

JavaScript, es uno de los más potentes e importantes lenguajes de programación en la actualidad, por tres enfoques claros: es útil, práctico y está disponible en cualquier navegador web (Mohedano, 2012)

El poder de JavaScript está disponible principalmente en lado frontend, agregando mayor interactividad a la web, también puedes usar las librerías y framework como: jquery, angular, backbone, react y demás, escritas sobre JavaScript, y que te ayudan a crear una mejor experiencia de usuario en nuestros sitios web (Mohedano, 2012).

Las características de JavaScript que haces de este lenguaje, uno de los más populares en la actualidad son:

1. Es Liviano.
2. Multiplataforma, ya que se puede utilizar en Windows, Linux o Mac o en el navegador de tu preferencia.
3. Es Imperativo y estructurado, mediante un conjunto de instrucciones indica al computador qué tarea debe realizar.
4. Prototipado, debido a que usa prototipos en vez de clases para el uso de herencia.
5. Orientado a objetos y eventos.
6. Es Interpretado, no se compila para poder ejecutarse.

Angular

Angular, ubicado en el *frontend*, es un framework mantenido por Google para el desarrollo de aplicaciones web del lado del cliente, su estructura está hecha en base a TypeScript, el cual es una versión avanzada de JavaScript (lenguaje principal para el desarrollo *frontend*) (Basalo, 2016).

Uno de los objetivos de Angular es fortalecer la estructura MVC(Modelo-Vista-Controlador) en el desarrollo web y las SPA (Single Page Application). El MVC es una arquitectura que se encarga de separar la lógica del código en una aplicación, en donde la vista es la interfaz gráfica de usuario y esta cambia de acuerdo a un controlador que gestiona el contenido en ella. Las Single Page Application son páginas que cargan únicamente un archivo HTML en el cliente y que su contenido, la navegación entre secciones y páginas de la aplicación, así como la carga de datos, se realiza de manera dinámica, casi instantánea, asíncronamente haciendo llamadas al servidor ("*backend*" con un API REST) y sobre todo sin refrescar la página en ningún momento (Basalo, 2016).

Es decir, las aplicaciones web que podemos hacer con Angular son reactivas y no recargan el navegador, todo es muy dinámico.

Ventajas de usar Angular:

- Lenguaje Typescript, tiene una sintaxis muy parecida a Java, con tipado estático.
- Sigue el patrón MVC, con la vista separada de los controladores.
- Basado en componentes, es decir, podemos escribir componentes web con vista y lógica para después reutilizarlos en otras páginas.
- Comunidad muy grande con multitud de tutoriales y librerías.
- Inyección de dependencias, un patrón de diseño que se basa en pasar las dependencias directamente a los objetos en lugar de crearlas localmente.
- Programación reactiva, la vista se actualiza automáticamente a los cambios.
- Se integra bien con Ionic, para adaptar aplicaciones web a dispositivos móviles.

HTML 5

El HyperText Markup Language o lenguaje sencillo para formato de documentos de hipertexto (HTML). Es un lenguaje de marcas, diseñado para estructurar textos y presentarlos en forma de hipertexto y multimedia. Es un lenguaje que se escribe mediante etiquetas, que aparecen especificadas por corchetes angulares (< >). Permite aglutinar textos, sonidos e imágenes y posibilita la introducción de referencias a otras páginas por medio de los enlaces hipertexto, facilita además la inclusión de código JavaScript en su código. Los documentos que son creados con extensión HTML, pueden ser visualizados mediante navegadores como Explorer, Mozilla, Firefox o Netscape. Entre los componentes del HTML, aparecen los elementos y sus atributos, los tipos de data y la declaración de tipo de documento. Los elementos son la estructura básica de este lenguaje, ya que tienen dos propiedades: atributos y contenido. Es considerado uno de los formatos más populares que existen para la construcción de documentos (Gauchat, 2012).

Con HTML5, se tienen otras posibilidades para explotar usando menos recursos. Entra en desuso el formato XHTML, dado que ya no sería necesaria su implementación. Se trata de un sistema para formatear el layout de nuestras páginas, así como hacer algunos ajustes a su aspecto. Los navegadores pueden saber cómo mostrar una determinada página web, saber dónde están los elementos, dónde poner las imágenes, dónde ubicar el texto (Gauchat, 2012).

CSS 3

Una página web es realmente un documento de texto. En dicho documento se escribe código HTML, con el que se crea el contenido de una web. Por otro lado, existe el código CSS, que unido al código HTML permite darle forma, color, posición (y otras características visuales) a una página.

Las siglas CSS (Cascading Style Sheets) significan «Hojas de estilo en cascada» y parten de un concepto simple pero muy potente: aplicar estilos (colores, formas, márgenes, etc.) a uno o varios

documentos (generalmente documentos HTML, páginas webs) de forma masiva. Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla (Aubry, 2014).

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan sólo información y datos, relativos al significado de la información a transmitir (el contenido), y todos los aspectos relacionados con el estilo (diseño, colores, formas, etc.) se encuentren en un documento CSS independiente (la presentación) (Aubry, 2014).

De esta forma, se puede unificar todo lo relativo al diseño visual en un solo documento CSS, y con ello, varias ventajas:

- Si necesitamos hacer modificaciones visuales lo hacemos en un sólo lugar y no tenemos que editar todos los documentos HTML en cuestión por separado.
- Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil de organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).
- Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, entre otros.

Boostrap 4

Es un framework CSS de código abierto que favorece el desarrollo web de un modo más sencillo y rápido. Incluye plantillas de diseño basadas en HTML y CSS con la que es posible modificar tipografías, formularios, botones, tablas, navegaciones, menús desplegables, etc. También existe la posibilidad de utilizar extensiones de Javascript adicionales (Spurlock, 2013).

Fue desarrollado inicialmente por Twitter en 2011 y permite crear interfaces de usuario limpias y compatibles con todo tipo de dispositivos. Entre las ventajas que tiene Bootstrap es que favorece el design responsive, el cual se utiliza para mejorar la experiencia de los usuarios en el sitio web y en consecuencia el posicionamiento (Spurlock, 2013).

Entre las ventajas de utilizar bootstrap se pueden enumerar las siguientes:

- Es de código abierto, y todo su código actualizado se encuentra en un repositorio de Github.
- Está mantenido y actualizado por Twitter.
- Es compatible con la mayoría de navegadores (Chrome, Safari, Mozilla, entre otros).
- Dispone de gran cantidad de documentación, tanto en su portal como en páginas web especializadas.

- Utiliza componentes vitales para los desarrolladores (HTML5, CSS3, jQuery o GitHub, entre otros).
- Incluye Grid system para maquetar por columnas.
- Sus plantillas son de sencilla adaptación responsive
- Dispone de un conjunto de elementos web personalizables.
- Se integra con librerías JavaScript.
- Usa Less: un lenguaje de las hojas de estilo CSS preparado para enriquecer los estilos de la web.

PostgreSQL 12.1

PostgreSQL es un avanzado sistema de bases de datos relacionales basado en Open Source. Esto quiere decir que el código fuente del programa está disponible a cualquier persona libre de cargos directos, permitiendo a cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades. PostgreSQL está bajo licencia BSD. Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas del´algebra relacional. Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras (Pérez, 2008).

Se caracteriza por ser un sistema estable, de alto rendimiento, gran flexibilidad ya que funcionar la mayoría de los sistemas Unix, además tiene características que permiten extender fácilmente el sistema. PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes. Permite desarrollar o migrar aplicaciones desde Access, Visual Basic, Foxpro, VisualFoxpro, C/C++ Visual C/C++, Delphi, etc., para que utilicen a PostgreSQL como servidor de BD; Por lo expuesto PostgreSQL se convierte en una gran alternativa al momento de decidirse por un sistema de bases de datos (Postgresql, 2015).

Se decidió la utilización de estas tecnologías siguiendo la directiva del centro de soporte técnico, facilitando, en un futuro, la fusión de la presente aplicación con otras desarrolladas en el centro para crear un sistema de mayor nivel.

Herramientas

Visual Studio Code 1.40.2

Es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS . Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo

que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto (Microsoft, 2020).

Visual Studio Code es una aplicación basada en Electron. Electron es un framework para programar aplicaciones gráficas de escritorio utilizando tecnologías web, e incluye Chromium (la versión libre de Google Chrome) como motor gráfico y el entorno de Node.js para ejecutar JavaScript (Node.js a su vez utiliza V8, el compilador de JavaScript de Chrome) (Microsoft, 2020).

El desarrollo de Visual Studio Code se lleva a cabo en GitHub. GitHub permite a cualquier usuario crear notificaciones (denominadas issues) dirigidas a los programadores, para informar de fallos o hacer peticiones. Cada issue permite hacer un seguimiento de su evolución: si es tomada en consideración, si es rechazada, comentarios de otros usuarios, su proceso de resolución y resolución final (Microsoft, 2020).

Ventajas:

- Se puede utilizar como lenguajes de programación.
- Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros.
- Posibilidad de configurar la vista a nuestro gusto. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas de nuestro proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades.

Pycharm 2019.2.3

Es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web con varios precios (Negrete, 2020).

Entre sus características cabe mencionar.

- Asistencia y análisis de codificación, con finalización de código, sintaxis y resaltado de errores, integración delinter y arreglos rápidos.
- Navegación de proyectos y códigos: vistas de proyectos especializados, vistas de estructura de archivos y saltos rápidos entre archivos, clases, métodos y usos.

- PythonRefactoring: incluye renombrar, extraer método, introducir variable, introducir constante y otros.
- Soporte para frameworks web: Django, web2py y Flask.
- Depurador integrado de Python.
- Pruebas unitarias integradas, con cobertura line-by-line (línea por línea).

PgAdmin 3

Es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. La aplicación se puede utilizar para manejar postgreSQL 7.3 y superiores y funciona sobre casi todas las plataformas. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de la sintaxis SQL, un editor de código del lado del servidor, un agente para la programación de tareas «SQL/batch/shell», soporte para el motor de replicación Slony-I y mucho más. La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor.

Una característica interesante de pgAdmin3 es que, cada vez que se realiza alguna modificación en un objeto, escribe la(s) sentencia(s) SQL correspondiente(s), lo que hace que, además de una herramienta muy útil, sea a la vez didáctica. También incorpora funcionalidades para realizar consultas, examinar su ejecución (como el comando "*explain*") y trabajar con los datos (PGAdmin, 2016).

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Paradigm, 2020).

Se caracteriza por (Ortiz, 2014):

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan 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 e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, con diferentes especificaciones.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web.

1.4 Metodología de desarrollo del software

Una metodología es un conjunto de actividades necesarias para llegar a un fin, con el propósito de transformar los requisitos de un usuario en un sistema de software. Esta surge ante la necesidad de trabajar mediante el uso de procedimientos, técnicas, herramientas y documentos durante el desarrollo de software. Entre sus funciones está, guiar, planificar, estructurar, controlar, manipular y dirigir el proceso de desarrollo de sistemas de información (Alonso, 2005).

El objetivo de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software.

Existen diversas metodologías, citando algunos ejemplos se puede mencionar Microsoft Solution Framework (MSF), Extreme Programming (XP) y Rational Unified Process (RUP), respondiendo cada una de ellas a diferentes clasificaciones, agiles, flexibles y robustas.

AUP-UCI es una variación de la metodología “Proceso Unificado Ágil” (AUP por sus siglas en inglés) en unión con el modelo CMMI-DEV v 1.3 desarrollada en la UCI. Es flexible y no requiere de una gran cantidad de desarrolladores, concisa en el aspecto de la documentación, permitiendo generar solo la necesaria y no la especificada para cada flujo de trabajo como lo hace RUP. Está diseñada para trabajar en proyectos pequeños donde la atención se centra en las actividades que realmente son importantes. Permite el uso de herramientas de cualquier tipo, incluyendo aquí las de código abierto. Es fácil de manejar a través de herramientas de edición HTML sin necesidad de ser adaptada y es una metodología que se ajusta y aprovecha las ventajas que brindan las metodologías ágiles.

Dadas las características anteriores se comprende que estas se ajustan perfectamente al trabajo a realizar, pues es idónea para la entrega de productos en cortos períodos de tiempo y satisface las necesidades del proceso por lo que se escoge AUP-UCI como metodología de desarrollo para guiar la investigación.

Fases Variación UCI-AUP:

1. **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización que

permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

2. **Ejecución:** En esta fase se realizan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
3. **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades de cierre del proyecto.

1.5 Conclusiones del capítulo

Al analizar aspectos teóricos sobre los sistemas de gestión de reportes se puede concluir que estos constituyen una aplicación que en esencia permita crear, modificar o eliminar un grupo de incidencias o reportes, respondiendo a los requerimientos del cliente. Con el estudio de sistemas homólogos se pudo apreciar que estos presentan características y funcionalidades de gran utilidad, pero no cumplen con todos los requerimientos del cliente, evidenciando la necesidad de la implementación de un nuevo sistema para la gestión de los reportes. Luego de analizar las características de la metodología AUP-UCI se decidió utilizar esta para guiar el proceso de desarrollo garantizando la eficacia y eficiencia en el proceso de generación del software. El análisis de las herramientas permitió seleccionar las que brindaban la estabilidad y robustez necesaria para la realización del sistema.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA DE GESTIÓN DE REPORTES.

Cuando se inicia el desarrollo de un sistema informático es necesario realizar el análisis de las particularidades del producto y diseñar previamente las funcionalidades más importantes. En este proceso se identifican varias etapas, que van desde la declaración del problema y los requisitos del sistema, hasta las pruebas y la liberación del mismo. En el presente capítulo se define la propuesta del sistema de gestión de reportes sobre la actividad de soporte, con el propósito de satisfacer el objetivo de la presente investigación. En el caso de la solución propuesta se utiliza la variante ágil de la metodología AUP, en su variación UCI; a partir de la cual se realiza la descripción de los requisitos funcionales y no funcionales. Los requisitos funcionales son ampliados en las Historias de Usuario, siendo organizados en la línea de tiempo del desarrollo del producto. Además, se diseñan las clases y relaciones más importantes dentro del sistema de gestión de reportes, y se identifican los principales patrones de diseño mediante los cuales se garantiza la presencia de buenas prácticas de programación. Como parte del diseño del sistema de gestión de reportes, se desarrollan los diagramas de clases y de secuencia, que constituyen los artefactos principales generados para esta etapa.

2.1 Modelo de Dominio

El modelo de dominio es una representación visual de los principales conceptos u objetos del mundo real, significativos para un problema o área de interés. Este es de gran ayuda para desarrolladores y usuarios, de esta forma se utiliza un vocabulario común y pueden entender el contexto en que se enmarca el sistema.

Se realiza cuando los procesos a ejecutar no son del todo visibles, ya que no se pueden identificar con claridad los actores y trabajadores del negocio en un problema específico. En él se describen las distintas entidades del problema, en forma de conceptos modelados por objetos y sus relaciones, además de las restricciones que rigen el dominio del problema.

En el sistema de gestión de reportes sobre la actividad del centro de soporte se brinda soporte técnico a las aplicaciones desarrolladas por la UCI que necesitan de este servicio, las cuales son vendidas a varios clientes. Las dudas e inquietudes de los clientes son atendidas como incidencias por los especialistas del centro de soporte. Es importante destacar que para que exista la incidencia debe estar asociada obligatoriamente a un cliente que haya contratado los servicios del centro de soporte. El centro a su vez genera uno o varios reportes con respecto al estado de las incidencias.

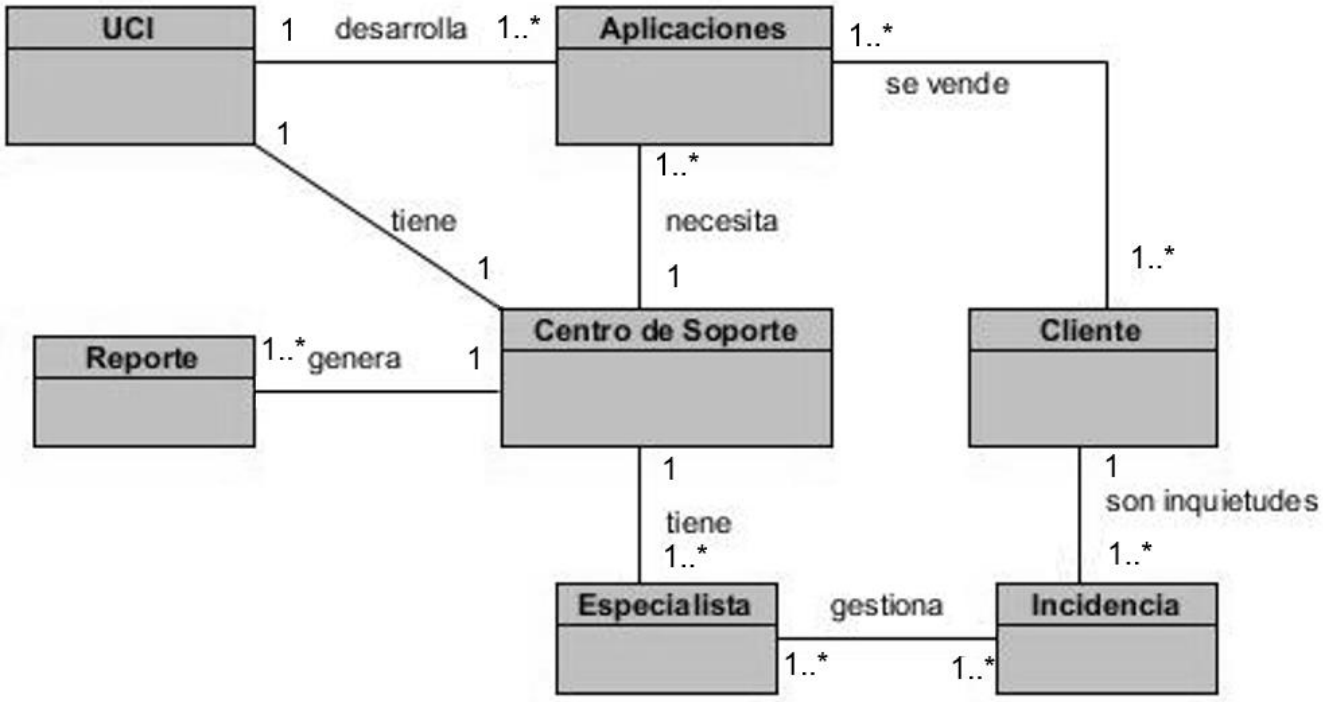


Ilustración1: Diagrama del Dominio del sistema.

2.2 Solución propuesta

El Centro de Soporte de la Universidad de la Ciencias Informáticas tiene como misión: “Brindar el servicio de soporte técnico a las aplicaciones y servicios informáticos desarrollados por la Universidad de las Ciencias Informáticas con calidad y eficiencia a partir de una correcta gestión de los mismos y garantizando elevados niveles de satisfacción de sus clientes” (UCI, 2019). Es por ello que la atención a clientes juega un papel primordial en su funcionamiento diario, lo cual incluye la gestión de incidencias reportadas por las entidades que contratan el servicio de soporte.

Al observar la forma en la que se trabajan las incidencias, se detectó que el sistema informático que controla este proceso no cumple los requerimientos necesarios para una gestión óptima del mismo. Lo que trae como consecuencia la no disponibilidad de recursos o información para los clientes, comprometiéndose así la calidad de este servicio. Para dar solución a estas deficiencias se propone crear una aplicación web que, en primer orden, esté acorde a la tendencia de la universidad de desarrollar sobre tecnologías libres. Además, se informatizarán procesos de facturación que requieren evidencias del trabajo durante periodos determinados, también permitirá medir la actividad laboral de los especialistas del centro, por tanto, se podrá tener un mayor control de la información.

Por medio de este sistema, el centro de soporte incrementará el valor agregado de sus servicios al ser más organizado y brindar un soporte in-situ con mayor calidad.

2.3 Concepción del sistema

La estructura, el diseño y las funcionalidades que debe ejecutar el sistema para alcanzar su objetivo, son vitales para entender el curso de la investigación. Es asociado a esta idea que se hace preciso la gestión documental de la concepción del sistema, mostrando los principales detalles del producto de software que se propone como solución. Esto permitirá un mayor entendimiento del proyecto, no solo para su desarrollo, sino también para el cliente y la creación de futuras versiones.

2.3.1 Visión y alcance del sistema

El sistema de gestión de reportes contendrá como objetivos fundamentales:

- El control del estado de las incidencias en cada momento.
- La formulación de reportes asociados a las incidencias.

Para el cumplimento de estos objetivos se implementan diversas funcionalidades desde el punto de vista del código fuente, que permitan una fácil interacción y una respuesta rápida a las peticiones de los usuarios. En caso de ser necesario las funciones del producto pueden ampliarse o modificarse, para responder a las necesidades del centro o entidad.

2.4 Especificación de requisitos

Cuando se elabora un producto de software, es preciso satisfacer un grupo de necesidades que sirven como objeto de su creación. Estas dificultades encarecen o dificultan el trabajo de un cliente, por lo que es preciso ser muy cuidadoso cuando se elabora un producto, ya que sus funcionalidades son en concreto la respuesta a esas necesidades, las cuales son plasmadas por las metodologías bajo el nombre de requisitos o requerimientos (Turner, 2005).

Un requisito es una condición o capacidad que debe tener un sistema para satisfacer las necesidades de un cliente, con el fin de resolver un problema planteado por el mismo en forma de un documento formal (IEEE.R., 2019).

Para el ciclo de vida de la metodología AUP-UCI los requisitos pueden ser encapsulados dependiendo de las características del proyecto. Esta facilitará que una vez definidos los requisitos, se realice una estimación previa del costo en tiempo que implicará cada requisito para el desarrollador (Brito, 2019).

2.4.1 Requisitos funcionales

Los requisitos funcionales(RF) definen las funciones que es capaz de realizar el sistema, es decir, describen las funcionalidades o los servicios que se espera que este provea (IEEE.R., 2019).

Se seleccionaron en total 29 requisitos funcionales, estimando el tiempo de realización en días tomará aproximadamente 84 días realizar la aplicación, definiendo las prioridades según las necesidades del cliente.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad (RF.RNF, 2016).

Tabla1: Tabla de requisitos funcionales y no funcionales.

Prioridad	Item*	Descripción	Estimación	Estimado por
Alta	RF1	Autenticar.	2	Analista
Alta	RF2	Obtener reporte de incidencias por rango de fecha.	3	Analista
Alta	RF3	Obtener reporte de incidencias por mes.	3	Analista
Alta	RF4	Obtener reporte de incidencias por especialista.	3	Analista
Alta	RF5	Obtener reporte de incidencias por sistema.	3	Analista
Alta	RF6	Obtener reporte de incidencias en espera.	3	Analista
Alta	RF7	Crear reporte personalizado.	4	Analista
Alta	RF8	Modificar reporte personalizado.	4	Analista
Alta	RF9	Eliminar reporte personalizado.	4	Analista
Media	RF10	Exportar a pdf reporte de incidencias por rango de fecha.	2	Analista
Media	RF11	Exportar a pdf reporte de incidencias por mes.	2	Analista
Media	RF12	Exportar a pdf reporte de incidencias por especialista.	2	Analista
Media	RF13	Exportar a pdf reporte de incidencias por sistema.	2	Analista

Media	RF14	Exportar a pdf reporte de incidencias en espera.	2	Analista
Media	RF15	Exportar a Excel reporte de incidencias por Rango de fecha.	2	Analista
Media	RF16	Exportar a Excel reporte de incidencias por mes.	2	Analista
Media	RF17	Exportar a Excel reporte de incidencias por especialista.	2	Analista
Media	RF18	Exportar a Excel reporte de incidencias por sistema.	2	Analista
Media	RF19	Exportar a Excel reporte de incidencias en espera.	2	Analista
Alta	RF20	Graficar reporte de incidencias por rango de fecha.	4	Analista
Alta	RF21	Graficar reporte de incidencias por mes.	4	Analista
Alta	RF22	Graficar reporte de incidencias por especialista.	4	Analista
Alta	RF23	Graficar reporte de incidencias por sistema.	4	Analista
Alta	RF24	Graficar reporte de incidencias en espera.	4	Analista
Alta	RF25	Análisis estadístico para reportes por tipo por rango de fecha.	3	Analista
Alta	RF26	Análisis estadístico para reportes por tipo por mes.	3	Analista
Alta	RF27	Análisis estadístico para reportes por tipo por especialista.	3	Analista
Alta	RF28	Análisis estadístico para reportes por tipo por sistema.	3	Analista
Alta	RF29	Análisis estadístico para reportes por tipo por incidencias en espera.	3	Analista

Baja			
Requisitos No Funcionales			
Usabilidad	• La terminología del menú debe ser constante en todo el sitio. • El sistema será utilizado por todo el personal del centro que tenga acceso a la red del centro y necesite hacer uso del mismo.		
Rendimiento	• El sistema deberá soportar una conexión simultánea de al menos 100 usuarios.		
Disponibilidad	• Los usuarios deben tener acceso a la información desde cualquier dispositivo sin que los mecanismos utilizados para la seguridad de los datos retrasen la obtención de los mismos.		
Restricciones de diseño	• Lenguajes de desarrollo: Python, Django, CSS 3, HTML 5, AngularJs 1.5 como framework de JavaScript • Como IDE se empleará Visual Studio Code 1.40.2 • Como servidor web Apache 2.4 • El sistema gestor de base de datos PostgreSQL 12.1 • El sistema operativo a usar en el entorno de desarrollo: Nova 5. • Los artefactos de análisis se realizarán con Visual Paradigm 8.0		
Software	• En los ordenadores donde se encuentre instalado se recomienda		

	como servidor de aplicación Debian 6 o superior. No obstante, debe tener un rendimiento eficiente en otros sistemas operativos (ejemplo: Windows). • A los clientes que acceden al sistema se les recomienda para el acceso Mozilla Firefox 30 en adelante. • El servidor de base de datos puede estar alojado en el mismo servidor de aplicación, contando con espacio disponible para almacenar los datos. (50GB)		
Hardware	• En una PC física al menos dual core a 2.8GHZ con 2GB de RAM. • Monitor VGA o superior. • Espacio en disco duro de 10GB		
Portabilidad	• El sistema debe ser multiplataforma		

2.4.2 Historias de Usuario

Una vez que los requisitos son establecidos y estimados es necesario que sean descritos minuciosamente para facilitar su desarrollo. Las metodologías ágiles, generalmente, proponen las Historias de Usuarios (HU) como el mecanismo para gestionar los requisitos, aplicándose de manera similar en AUP UCI en su versión ágil. Las HU "se escriben desde la perspectiva del cliente, aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas pocas semanas" (Guevara, 2014).

En el capítulo se muestran 6 Historias de Usuario, el resto se encuentra en la sección de anexos (ver Anexo 2).

Tabla2: Historia de usuario "Autenticar usuario".

Historia de Usuario

Número: HU_1	**Nombre Historia de Usuario:** Autenticar usuario
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Permite el acceso de los usuarios del centro de soporte al sistema de planificación de visitas.	
Observaciones: • **El sistema debe estar iniciado y el cliente desde el que se accede estar conectado a la red.** • **El usuario autenticado debe ser usuario del sistema.**	
Prototipo de interfaz: 	

Tabla3: Historia de usuario “Obtener reporte de incidencias por rango de fecha.”

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Obtener reporte de incidencias por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Muestra todas las incidencias dado un rango de fecha determinada.	
Observaciones: • **Se debe introducir al sistema un rango de fecha determinado.**	
Prototipo de interfaz:	

Tabla4: Historia de usuario “Obtener reporte de incidencias por mes”.

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Obtener reporte de incidencias por mes.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Muestra todas las incidencias dado un mes determinado.	
Observaciones: • **Se debe introducir al sistema un mes determinado**	

Prototipo de interfaz:

Tabla5: Historia de Usuario "Exportar a pdf incidencias por mes".

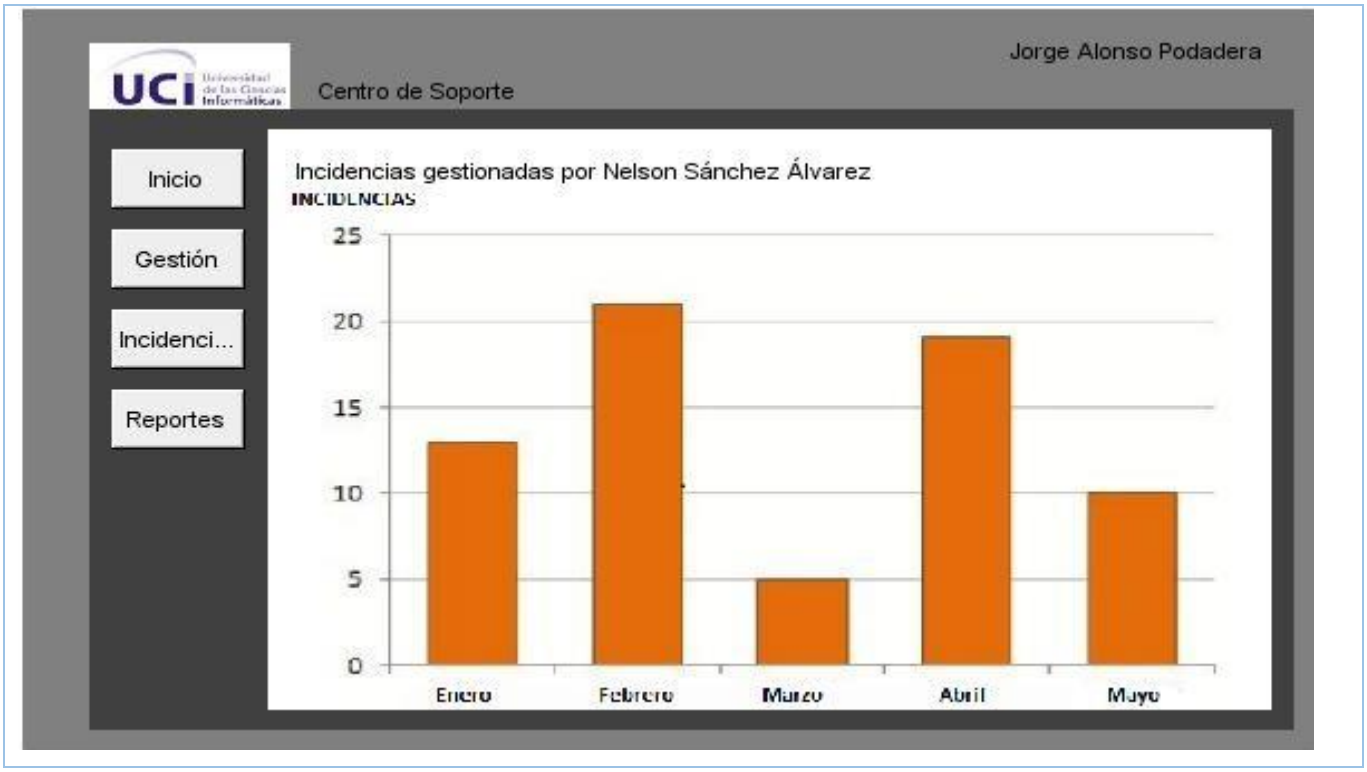
Historia de Usuario	
Número: HU_11	**Nombre Historia de Usuario:** Exportar a pdf reporte de incidencias por mes.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 1.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema un mes determinado.**	
Prototipo de interfaz:	

Tabla6: Historia de Usuario "Exportar a Excel incidencias por rango d fecha".

Historia de Usuario	
Número: HU_15	**Nombre Historia de Usuario:** Exportar a Excel reporte de incidencias por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 1.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema un rango de fecha determinada**	
Prototipo de interfaz:	

Tabla7: Historia de Usuario "Graficar incidencias por especialista".

Historia de Usuario	
Número: HU_22	**Nombre Historia de Usuario:** Graficar reporte de incidencias por especialista.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite visualizar la información mediante gráficos.	
Observaciones: • **Se debe introducir al sistema el nombre o identificador de un especialista.**	
Prototipo de interfaz:	



2.5 Arquitectura del sistema y Diagrama de clases

Cuando se utiliza el framework Angular, se utiliza necesariamente la arquitectura Modelo-Vista-Control (MVC), por ser sobre la cual se encuentra desarrollado el marco de trabajo, aunque no sea obligatoria durante su utilización (Jaramillo, 2008).

MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- El **modelo** es el objeto de la aplicación, o sea es la representación específica de la información con la cual el sistema opera.
- La **vista** es la presentación en pantalla. Este presenta el modelo en un formato adecuado para interactuar con los usuarios.
- El **controlador** responde a eventos o acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

MVC permite reflejar un diseño que separa vistas de modelos. Pero el diseño de MVC es aplicable a un problema más general: separar objetos, por lo que el cambio de uno de ellos puede afectar a cualquier número de los otros, sin que se requiera del objeto cambiado para conocer los detalles de otros (Jaramillo, 2008).

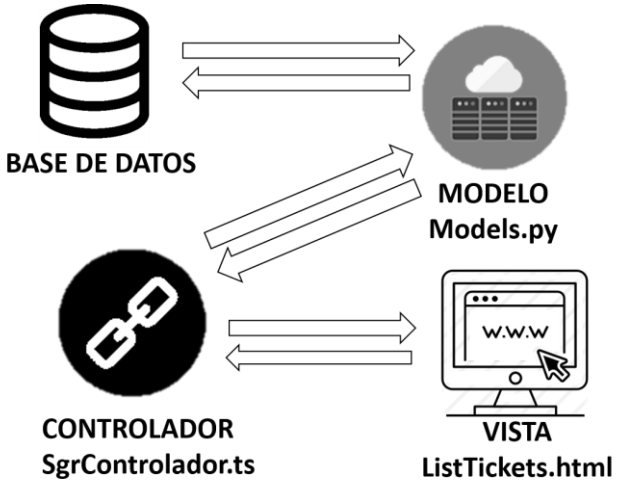


Ilustración 2: Arquitectura Modelo-Vista-Controlador.

En esta arquitectura la interacción entre las capas lógicas del sistema puede realizarse en cualquier dirección, permitiendo un acceso más rápido y eficiente. Se modelaron las clases y funcionalidades del sistema, mediante el diseño del Diagrama de Clases. De esta forma es posible comprender los elementos de implementación esenciales en el producto y la relación que se establece entre ellos.

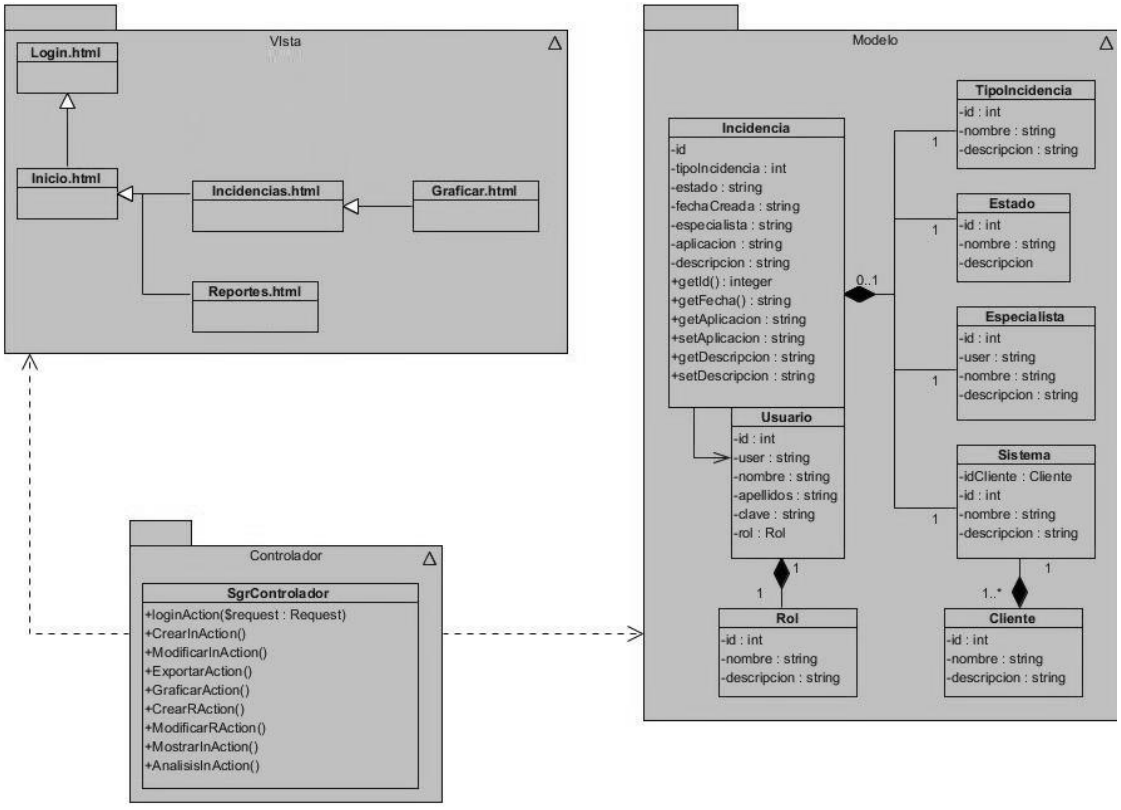


Ilustración 3: Diagrama de Clases (MVC).

2.6 Patrones del Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema/solución con un nombre, codifican buenos principios y sugerencias relacionados con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de software (Larman, 2015).

2.6.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Software,2011).

Experto:

El patrón experto en información es el principio básico de asignación de responsabilidades. Consiste en asignar la responsabilidad de creación de un objeto o la implementación de un método a la clase que contiene la información específica. La utilización de este patrón evita que deban solicitarse datos entre las clases de forma innecesaria (Fuente, 2014).

A través del uso de este patrón se conserva el encapsulamiento, ya que, los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

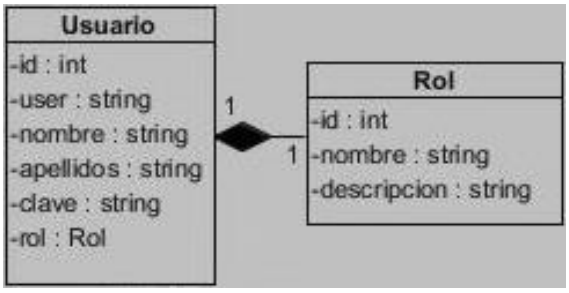


Ilustración 4: Patrón Experto**.**

Creador:

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, propiciando una mayor claridad, el encapsulamiento y la reutilización (Carmona, 2012).

Se evidencia, principalmente, en la clase SgrControlador, ya que es quien crea las instancias necesarias para hacer funcional el sistema de la forma más óptima posible.

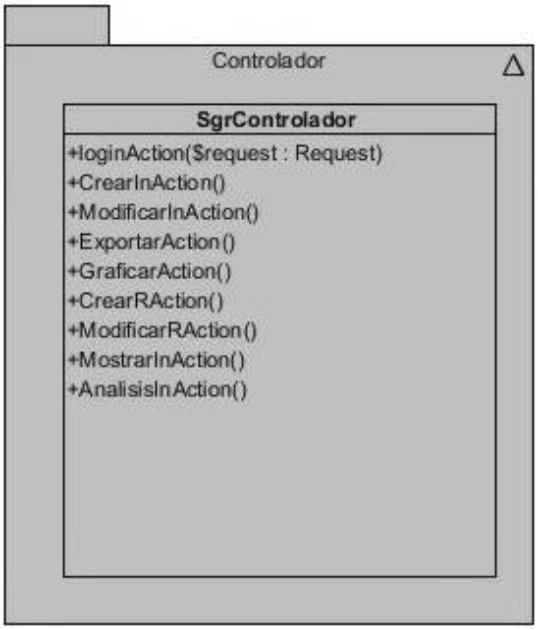


Ilustración 5: Patrón Creador.

Controlador:

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define, además, el método de su operación. Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente, un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad (Ingeniería, 2015).

Este patrón se evidencia, principalmente, en la clase SgrControlador que controla la información desde y hacia las vistas, siendo un controlador único en la gestión del sistema.

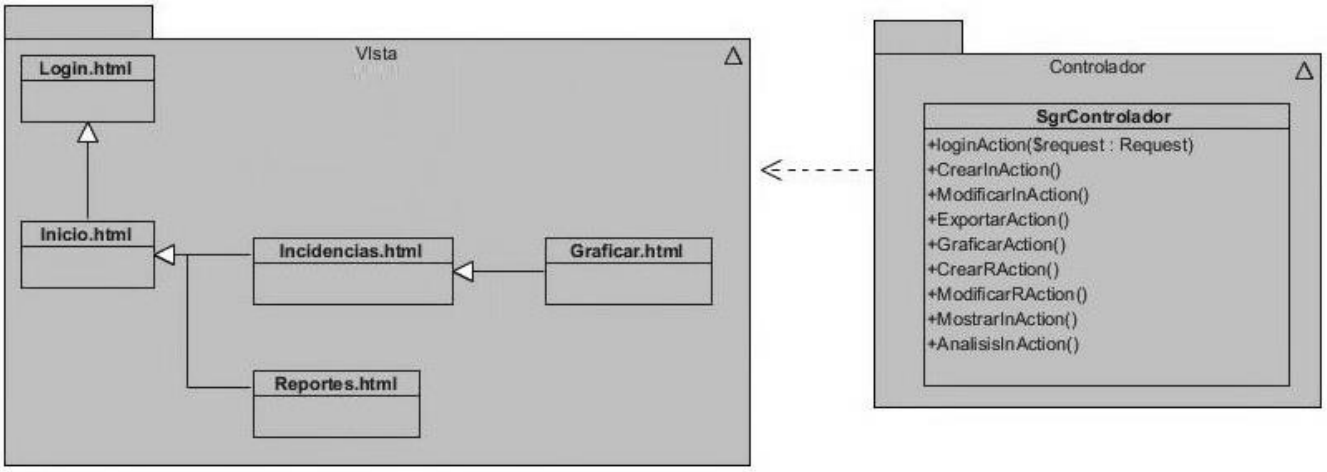


Ilustración 6: Patrón Controlador.

Bajo acoplamiento:

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. El grado de acoplamiento no puede considerarse aisladamente de otros principios como experto y alta cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño (Stevens, y otros, 1974).
En el SGR es posible evidenciar el Bajo Acoplamiento en la clase Incidencia, ya que una modificación en la clase Estado no afectaría su estabilidad y usabilidad.

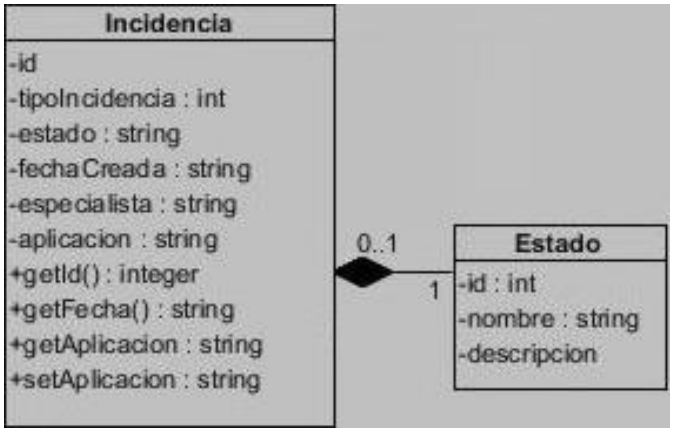


Ilustración 7: Patrón Bajo acoplamiento

Alta cohesión:

“En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas

que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo" (Craig, 1999).

La clase controladora contiene las diferentes funcionalidades que se encuentran estrechamente relacionadas, posibilitando que el software sea flexible frente a grandes cambios.

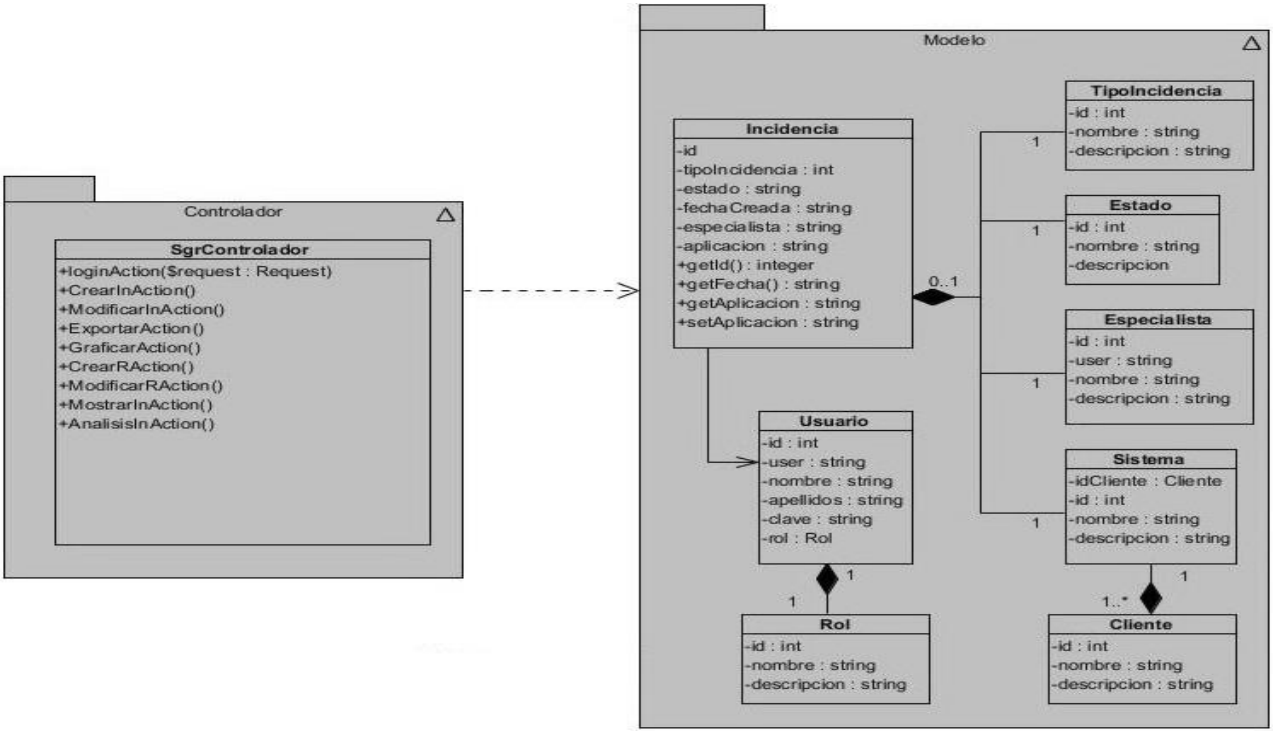


Ilustración 8: Patrón Alta cohesión.

2.6.2 Patrones GOF

Los patrones "Banda de los Cuatro" (Gang-of-Four) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas (Craig, 2015).

Instancia única (Singleton): Es un patrón diseñado para restringir la creación de objetos pertenecientes a una clase u objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Este patrón se refleja en las clases controladoras que son instancias únicas para la interacción entre componentes. A continuación, se muestra el método changeMessage(), este es el encargado de enviar los datos de las incidencias a la gráfica verificando si los datos son nuevos. En caso de no ser nuevos lanza una alerta que dice "Ya se está mostrando la gráfica para estas incidencias"

```
changeMessage(graf: Ticket[]){
  if(this.editMessage === this.dataTickets.data){
    alert("Ya se está mostrando la gráfica para estas incidencias");
  }else{
  this.editMessage = this.dataTickets.data;
  this.helper.changeMessage(graf);
  this.grafica.fecha();
 }
}
```

Ilustración 9: Patrón Instancia única.

Mediador (Mediator): Define un objeto que coordine la comunicación entre objetos de distintas clases. Se refleja en las librerías que funcionan como mediadoras entre las clases controladoras y las modelos de acceso a datos. A continuación, se muestra el objeto changeMessage(), el cual permite la comunicación entre objetos de las clases Listtickets y Graficar.

```
public changeMessage(msg: Ticket[]): void {
    this.message.next(msg);
}
```

Ilustración 10: Patrón Mediador.

Observador (Observer): Este patrón se encarga de definir dependencia entre objetos, de forma que, si alguno cambia su estado, automáticamente se notifica y actualizan todos los objetos que dependen de él. A continuación, se muestra el método listtickets, el cual permite que una vez cambiado el valor de la tabla de tickets, envía los datos a la gráfica para que esta se actualice.

```
listtickets(){

  this.serviceticket.getTickets().subscribe((result : Ticket[])=>{
    this.dataTickets = new MatTableDataSource(result);
    const graf = this.dataTickets.data;
    this.changeMessage(graf);
  });
}
```

Ilustración 11: Patrón Mediador.

2.7 Diagrama Entidad-Relación

Teniendo en cuenta que la solución propuesta se basa en el desarrollo sobre tecnologías web es preciso contar con una Base de Datos que permita almacenar la información referente a las visitas gestionadas y demás funcionalidades del sistema. El diseño de la base de datos es el proceso mediante el cual se toman los elementos conceptuales del sistema y se materializan en la base de

datos Estos elementos son representados mediante un diagrama de entidad-relación, definido como “una herramienta para el modelado de datos de un sistema de información. Estos modelos están basados en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos Objetos (Quintana, 2011).

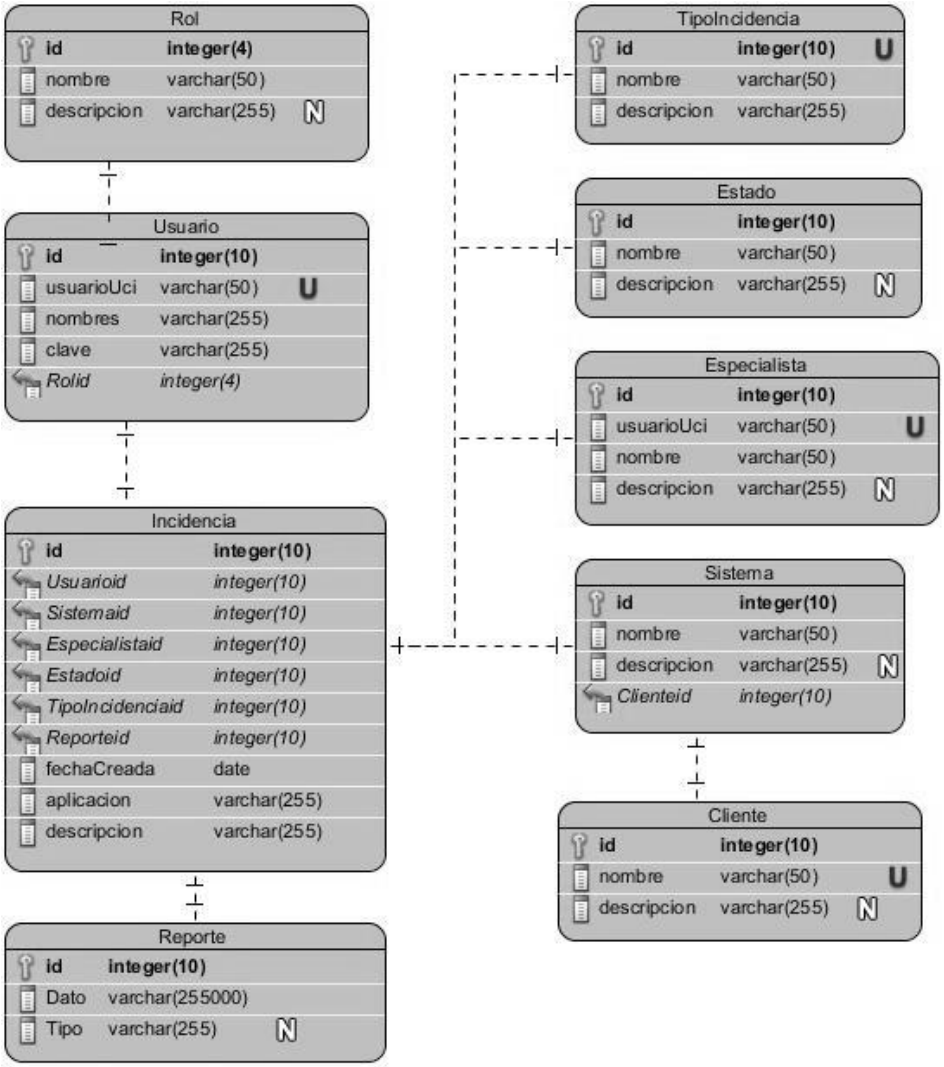


Ilustración 12: Diagrama Entidad-Relación.

2.8 Conclusiones del capítulo

La realización del modelo de dominio permitió entender el contexto en el que se enmarca el sistema, definiendo las características que debe cumplir el nuevo sistema, basado en tecnologías libres. La realización del análisis y diseño correspondiente, permitió identificar veintisiete (27) requisitos funcionales, que se estima deben ser realizados en un período de setenta y siete (77) días a partir del proceso de obtención de los requisitos. Los requisitos definidos en conjunto con las historias de

usuario generadas, constituyeron una guía fundamental para la construcción del sistema de gestión de reportes. La utilización de la arquitectura Modelo-Vista-Controlador y el uso de los patrones de diseño y de base de datos propuestos, garantizaron una mayor organización, reutilización de funciones y la obtención de un código más legible.

CAPÍTULO 3: IMPLEMENTACION Y PRUEBAS REALIZADAS AL SISTEMA DE GESTIÓN DE REPORTES

En el presente capítulo se definen los estándares de codificación que debe seguir el equipo de trabajo para desarrollar el sistema de gestión de reportes, así como los métodos y técnicas para la realización de las pruebas, con el propósito de validar la solución. Se describe la implementación del software, fase donde finalmente se materializa el sistema de gestión y cumple con los requisitos obtenidos al inicio de la investigación. El proceso de pruebas está dirigido a componentes del software, con el objetivo de medir el grado en que se cumplen los requisitos exigidos por el cliente y detectar la mayor cantidad de errores en el sistema de gestión para lograr su corrección.

3.1 Implementación del sistema de gestión de reportes

El modelo de implementación está comprendido por un conjunto de componentes y subsistemas que constituyen la parte física de la implementación del sistema. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes entre los que se encuentran datos, archivos, ejecutables, código fuente y directorios. Principalmente se describe la relación existente entre estos y cómo unos dependen de otros. Luego de esta fase se procede a probar la nueva herramienta (Pressman, 2013).

3.1.1 Diagrama de Componentes

Un diagrama de componentes es un diagrama tipo del Lenguaje Unificado de Modelado; representa cómo un sistema de software es dividido en componentes (código fuente, binario o ejecutable) y muestra la dependencia entre estos componentes. Los elementos modelados dentro de un diagrama de componentes serán componentes y paquetes. Proporciona una visión física de la construcción del sistema de información y pueden ser usados para modelar o documentar cualquier arquitectura de sistema (Microsoft, 2020).

En el diagrama de componentes de la Figura 7, se evidencian las interacciones entre los componentes de Angular y Django, dando lugar al sistema de gestión de reportes. Angular cuenta con los componentes .html (para las vistas), .ts (para la lógica), entre otros y Django cuenta tanto con componentes controladores y modeladores que permiten una correcta comunicación con la base de datos. Estos componentes permiten un adecuado manejo de la seguridad, las configuraciones, las excepciones y errores del subsistema.

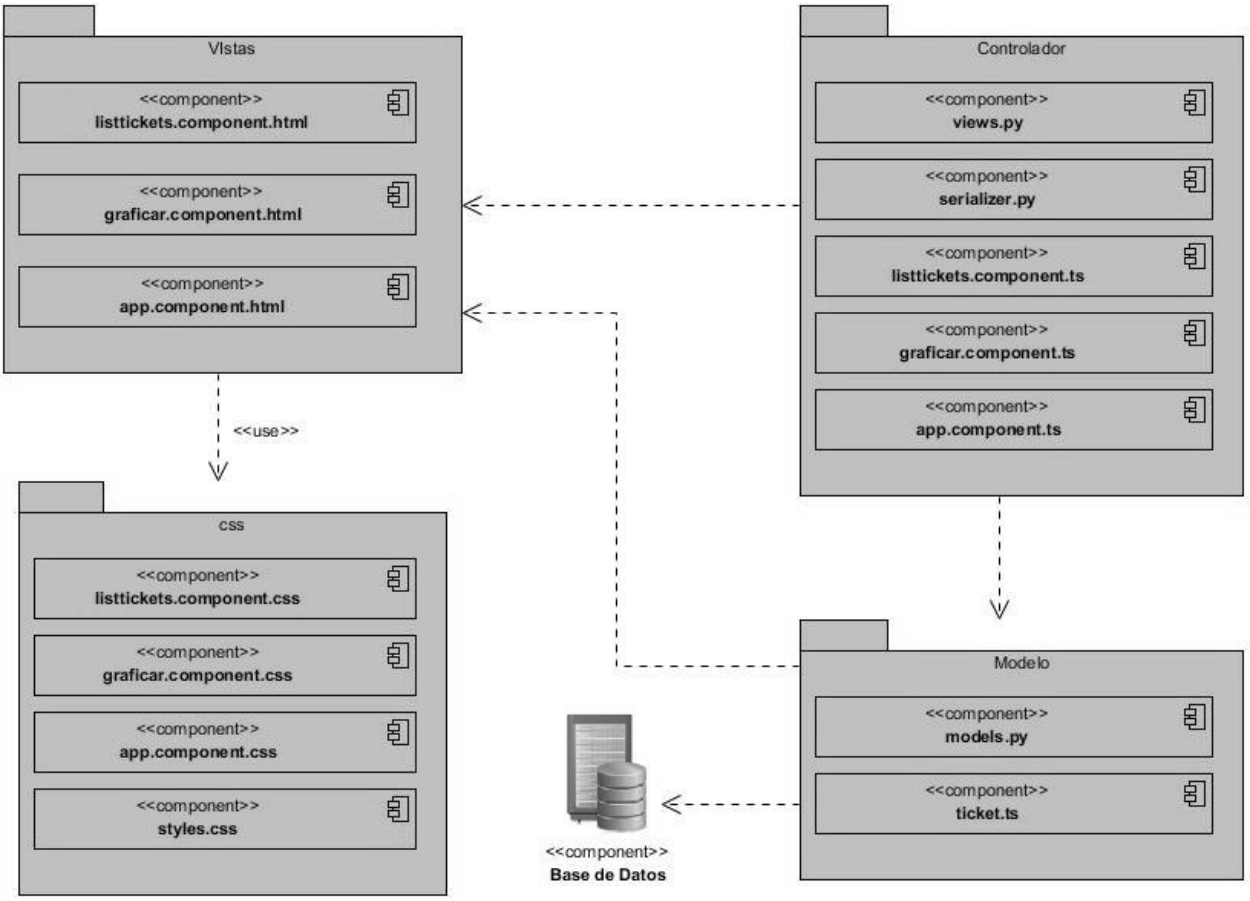


Ilustración 13: Diagrama de Componentes**.**

3.1.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema. El modelo de despliegue define la arquitectura física del sistema. Se usa para modelar de manera detallada los nodos físicos y las asociaciones de comunicación que existen entre ellos. Del mismo modo queda especificado qué hardware, sistemas operativos, software de interfaces y soporte. A continuación, se muestra dicho diagrama, el mismo está compuesto por una computadora que utilizará el usuario para acceder al sistema y deberá tener instalado un navegador web. Este ordenador se comunica con el servidor de aplicación a través del protocolo HTTP; el cual a su vez utilizará una conexión TCP-IP para la comunicación con el servidor de base de datos. Además, el servidor de aplicaciones consume los servicios web que brindan la plataforma de Soporte.

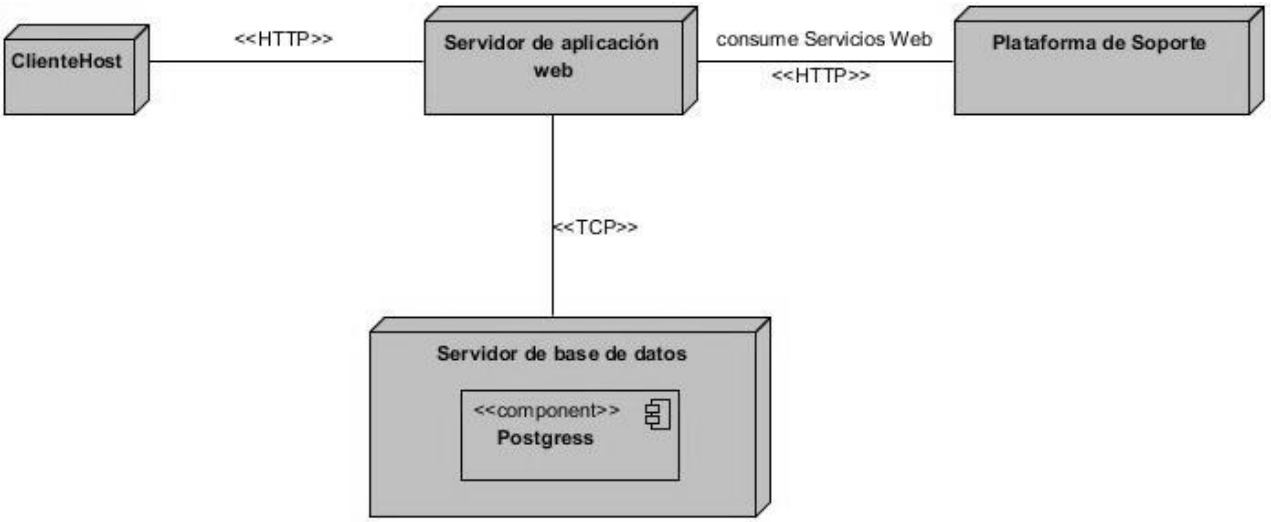


Ilustración 14: Diagrama de despliegue.

3.1.3 Estándares de codificación

Por código fuente se entiende todo texto legible por un ser humano y redactado en un lenguaje de programación determinado.

Los estándares de codificación son un elemento fundamental en la implementación de proyectos, permitiendo que el código fuente generado sea fácil de leer y modificar independientemente de quien ha sido su autor. Son una guía para el equipo de desarrollo, permiten asegurar que el código presente alta calidad y no contenga errores.

JavaScript:

Declaración de variables:

- Los nombres de variables deben ser escritos con notación camelCase.
- Los espacios entre el nombre y el valor de una variable son muy importantes puesto que mejoran la legibilidad del código.
- Cuando se declaran múltiples variables deben declararse todas con la misma sentencia “var”, una variable por línea y con (,) al final a menos que sea la última variable en cuyo caso será un (;) al final de la línea. A partir de la segunda línea debe usarse cuatro espacios antes del nombre de variable para mejorar la legibilidad del código fuente. El orden de las variables debe ser alfabético de ser posible. Es aconsejable que cada variable tenga un comentario acerca de su objetivo, de manera que aumenta la comprensibilidad del código.

- Cuando la variable a declarar contiene un elemento de tipo colección (arreglo y objeto) y su contenido excede los 70 caracteres o tiene más de dos elementos debe tener cada ítem de la colección en una línea diferente para mejorar la legibilidad del código (**Crockford, 2008**).

```
30
31  public barChartData: ChartDataSets[] = [
32    { data: [65, 59, 80, 81, 56, 55, 40], label: 'Sistema A' },
33    { data: [28, 48, 40, 19, 86, 27, 90], label: 'Sistema B' }
34  ];
35
```

Ilustración 15: Declaración de arreglo**.**

Declaración de funciones:

- Los nombres de funciones deben ser escritos con notación camelCase.
- Las llaves de las funciones deben empezar en l misma línea que se declara la función y debe haber un espacio entre el paréntesis de cierre de argumentos y la llave de inicio de cuerpo de función.
- Las funciones pueden ser funciones como tal o funciones como variables. En caso de que sean funciones como variables es necesario usar (;) después del cuerpo de la función para finalizar la sentencia.
- Los argumentos de funciones deben ser nombres con notación camelCase, si son varios deben estar separados por (,) (Crockford, 2008).

```
38
39  ngOnInit() {
40
41    this.helper.custonMessage.subscribe(msg => this.message = msg);
42    this.helper.custonMessage2.subscribe(msg => this.message2 = msg);
43
44  }
45
```

Ilustración 16: Declaración de funciones**.**

Construcción de cadenas:

- Deben ser usadas comillas simples (‘) puesto de esta manera se reducen los problemas a la hora de escapar las comillas dobles (“) usadas cuando se genera HTML y mejora la legibilidad (Crockford, 2008).

```
58    public contador = 'vacio';
```

Ilustración 17: Construcción de cadenas.

Operaciones:

- Todas deben tener espacios entre los operadores y los operandos, si se están usando paréntesis no es necesario usar espacio entre los paréntesis y los operandos.
- Deben usarse los comparadores estrictos (aquellos que comparan valor y tipo “===”, “!==”) para todas las operaciones lógicas de comparación (Crockford, 2008).

```
874     if((this.contador === "vacio")||(this.contador === "FechaI")){
875       if(this.ayudafecha2.length === 0){
876         this.ayudafecha2 = ddatos;
877       }
878     }
```

Ilustración 18: Operaciones.

3.2 Verificación del sistema

Las pruebas de verificación son el proceso de revisión que verifica que el sistema de software cumple con las especificaciones y logra su cometido. Utiliza técnicas tales como evaluaciones, inspecciones y tutoriales. La verificación es el proceso que permite comprobar que **lo** que se ha especificado cumple con las peticiones del usuario. Se trata de evaluar el sistema o parte de este durante o al final de desarrollo para determinar si satisface los requisitos iniciales (Kendall, 2005).

Entre los elementos que permiten concluir que el objetivo general se cumple completamente, se encuentran los siguientes:

- Constituye un sistema que facilita el trabajo de los especialistas, ya que tiene toda la información sobre el uso y gestión de reportes de una forma más organizada.
- El sistema recopila todo lo referente a incidencias y reportes realizados, de una manera estructurada, permitiendo generar nuevos reportes al ritmo que se desee, dándole la posibilidad de realizar las modificaciones en caso que lo desee, lo que apoya al control de la información.
- La información generada durante el proceso de gestión cuenta con una seguridad adecuada, garantizada por los niveles de acceso que brinda el sistema, ejemplo de esto lo evidencia la necesidad de autenticación para acceder a cualquier información del sistema.

3.3 Pruebas aplicadas al sistema.

Cualquier pieza de software completo desarrollado o adquirido puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, analizar defectos globales o estudiar

aspectos específicos de su comportamiento (seguridad y rendimiento). Este grupo de pruebas se conoce como Pruebas de sistema (Pressman,2013).

Las pruebas del sistema tienen como objetivo ejercitar profundamente el sistema para comprobar su integración de información globalmente, verificando el funcionamiento correcto de los interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con que se comunica. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.

Cuando se realizan pruebas debe mantenerse un enfoque sistémico, teniendo en cuenta que:

- Todo sistema tiene una serie de objetivos que le dan sentido, los cuales están asociados con indicadores de éxito que permiten determinar si los objetivos se cumplen y en qué medida lo hacen.
- Todo sistema tiene una serie de elementos que lo forman y la interacción de estos elementos se orienta a satisfacer los objetivos.
- Ningún sistema existe en aislamiento; siempre interaccionan con otros sistemas constituyendo un sistema mayor.

El objetivo de estas pruebas es descubrir errores y luego proporcionar a los programadores la información que necesitan para corregirlos mediante una serie de pasos de prueba (Pressman, 2013). A continuación, se describen los tipos de pruebas de software aplicadas, así como los métodos y técnicas empleadas para la evaluación de la propuesta de solución.

Prueba de caja blanca

Las pruebas de caja blanca se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente, y se establecen por medio del diseño de casos que se usan como base la estructura de control de flujo. Son aplicables a varios niveles (unidad, integración y sistema), pero habitualmente se aplican a las unidades de software; su objetivo es comprobar los flujos dentro de cada unidad (función, clase, módulo), también pueden probar el flujo entre unidades durante la integración, e incluso entre subsistemas. Se considera como una de los tipos de prueba más importante aplicadas al software, logrando como resultado una disminución apreciable en el número de errores existentes en los sistemas y por tanto el sistema ofrece una mayor calidad y confiabilidad.

La prueba de caja blanca permite:

- Que se ejecute por lo menos una vez cada instrucción del programa.
- Garantizar que todas las condiciones se comprueben como verdaderas y falsas.

- Que se ejecuten los bucles, probando el caso general y los casos extremos.

Las pruebas de caja blanca son pruebas estructurales, en las que conociendo el código y siguiendo su estructura lógica permite comprobar que dicho código cumple correctamente lo que el diseño indica (L., 2009).

Las principales técnicas de diseño de pruebas de caja blanca son:

- Pruebas de Flujo de Control
- Prueba de Datos
- Prueba de Bifurcación
- Prueba de Caminos Básicos

Para la realización de esta prueba en el sistema de gestión de reportes se utilizó la técnica de Caminos Básicos. Esta técnica permite al diseñador de casos de prueba obtener una medida de la complejidad lógica del diseño procedimental y usarla para la definición de caminos de prueba derivados del conjunto básico, garantizando que se ejecute al menos una vez cada sentencia del programa.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los que puede circular el flujo de control, para obtener dichos caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.

Los pasos seguidos para aplicar esta técnica son:

- Partir del diseño, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se separan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

```
imprimirEspecialista(ticket: Ticket){
  const y = ticket.toString().toLowerCase();
1 let nombre: String;
  let limite = this.dataUsers.data.length;
  const num1 = Number(y);

2 for(let k=0; k < limite; k++){
    const id = this.dataUsers.data[k].id;
    const idd = id.toString().toLowerCase();
3   const num2 = Number(idd);
4   if( num1 === num2){
5     nombre = this.dataUsers.data[k].firstname;
      k = limite;
    }
  }
6 return nombre;
  }
```

Ilustración 19: Técnica de Camino Básico a la funcionalidad Imprimir especialista en la tabla principal**.**

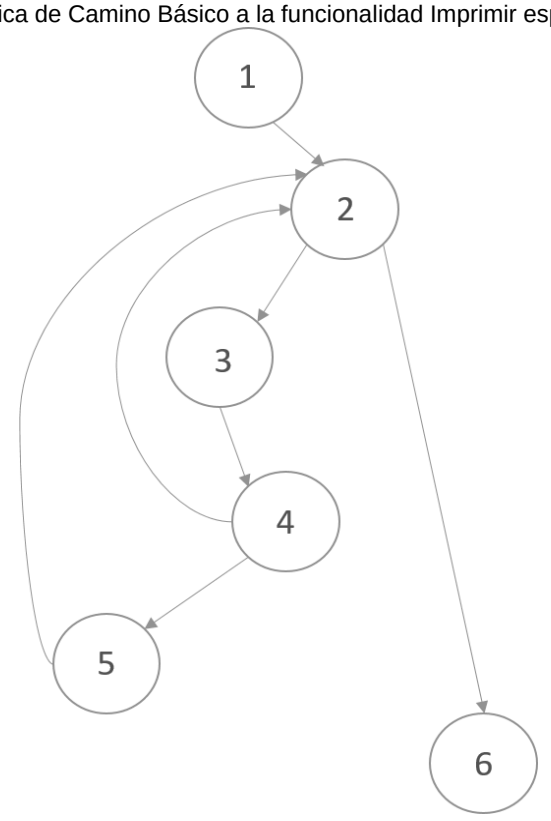


Ilustración 20: Grafo de flujo.

A partir del grafo obtenido con 6 nodos y 7 aristas se calcula la complejidad ciclomática V(G), la cual constituye una métrica de software que proporciona una medida cuantitativa de la complejidad lógica del programa (Pressman, 2013).

V(G) = cantidad_aristas - cantidad_nodos + 2
V(G) = 7 – 6 + 2 = 3
El valor V(G) ofrece un límite superior del número de pruebas que deben diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones (Pressman, 2013). Por esta razón, se diseñan casos de prueba para ser aplicados a cada camino básico.
Una ruta independiente o camino básico es cualquiera que introduce al menos una nueva condición en el programa. Este debe moverse a lo largo de al menos una arista que no se haya recorrido antes de definir el camino (Pressman, 2013). En la Tabla 9 se muestran los caminos básicos obtenidos a través del cálculo de la complejidad ciclomática.

Tabla8: Listado de caminos básicos.

No. Ruta	Camino básico
1	1-2-6
2	1-2-3-4-2-6
3	1-2-3-4-5-2-6

A continuación, se muestran los diseños de casos de prueba realizados a cada camino básico.

Tabla9: Diseño de caso de prueba para el camino 1 de la funcionalidad Imprimir especialista en la tabla principal.

Diseño de caso de prueba para el camino 1	
Nombre de la persona que realiza la prueba	Jorge Alonso Podadera.
Descripción de la prueba	Dado un identificador de usuario por parámetro mostrar en la tabla el nombre del Especialista.
Entrada	La tabla de usuario debe estar vacía.
Resultados esperados	El sistema no debe recorrer la tabla de usuarios, por tanto debe devolver la variable vacía
Evaluación de la prueba	Satisfactoria. Se devuelve la variable vacía

Tabla10: Diseño de caso de prueba para el camino 2 de la funcionalidad Imprimir especialista en la tabla principal.

Diseño de caso de prueba para el camino 2	
Nombre de la persona que realiza la prueba	Jorge Alonso Podadera.
Descripción	Dado un identificador de usuario por parámetro mostrar en la tabla el nombre del Especialista.
Entrada	La tabla de usuario debe contener elementos pero esta no contiene el identificador dado por parámetro.
Resultados esperados	El sistema debe recorrer la tabla de usuarios y comprobar que exista un usuario con identificador igual al identificador de usuario dado. Almacenar en una variable el nombre de este usuario y luego devolverlo. En caso de no existir un usuario con identificador igual al identificador dado, devolverá la variable vacía.
Evaluación de la prueba	Satisfactoria. Se devuelve la variable vacía

Tabla11: Diseño de caso de prueba para el camino 3 de la funcionalidad Imprimir especialista en la tabla principal.

Diseño de caso de prueba para el camino 3	
Nombre de la persona que realiza la prueba	Jorge Alonso Podadera.
Descripción	Dado un identificador de usuario por parámetro mostrar en la tabla el nombre del Especialista.
Entrada	La tabla de usuario debe contener elementos, entre estos el identificador dado por parámetros.
Resultados esperados	El sistema debe recorrer la tabla de usuarios y comprobar que exista un usuario con identificador igual al identificador de usuario dado. Almacenar en una variable el nombre de este usuario y luego devolverlo.
Evaluación de la prueba	Satisfactoria. Se devuelve el nombre del Especialista.

Se realizaron tres iteraciones de la prueba de caja blanca. A continuación, se identifican los resultados arrojados.

- En la primera iteración se obtuvo como resultado un total de 3 No Conformidades(NC), de ellas 1 Significativa(S), 1 No Significativa(NS) y 1 Recomendación(R).
- En la segunda iteración de pruebas arrojó como resultado 2 NC, de ellas 1 NS y 1R.
- En la tercera y última iteración de pruebas se obtuvo como resultado 1R solamente.

Prueba de caja negra:

La prueba de caja negra es una técnica de pruebas de software que verifica la funcionalidad sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software. Su objetivo es demostrar que las funciones del software son operativas, las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Esta prueba solo se enfoca en las entradas y salidas del sistema, basándose en los requerimientos del software y las especificaciones funcionales. Permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Alonso, 2005).

Estas pruebas permiten encontrar:

- Funciones incorrectas.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

A continuación, se muestra un caso de prueba realizado a la Historia de Usuario Gestionar reporte personalizado. Se entiende por Historia de Usuario, aquellas que son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos (acompañadas de las discusiones con los usuarios). Cada historia de usuario debe ser limitada, ésta debería poderse escribir sobre una nota adhesiva pequeña.

Tabla12: Caso de prueba # 2 utilizando método de caja negra.

Historia de usuario	**Gestionar reporte personalizado**

Caso de Prueba	**1**
Entrada	Estado de incidencia: Cerrado Especialista: Nelson Fecha por rango - Inicio: 8/15/2019 Fecha por rango - Fin: 1/9/2020 Fecha por mes: -
Salida	Todos los datos de los atributos son válidos y se muestran 17 incidencias en la tabla cumpliendo con los parámetros establecidos.

Tabla13: Caso de prueba # 2 utilizando método de caja negra.

Historia de usuario	**Gestionar reporte personalizado**
Caso de Prueba	2
Entrada	Estado de incidencia: Todas Especialista: Nelson Fecha por rango - Inicio: 3/11/2019 Fecha por rango - Fin: 3/15/2019 Fecha por mes: 3/14/2019
Salida	Todos los datos de los atributos son válidos y se muestran 59 incidencias en la tabla cumpliendo con los parámetros establecidos.

Tabla14: Caso de prueba # 2 utilizando método de caja negra.

Historia de usuario	**Gestionar reporte personalizado**
Caso de Prueba	3

Entrada	Estado de incidencia: Abierto Especialista: Nelson Fecha por rango - Inicio: 3/11/2019 Fecha por rango - Fin: 3/15/2019 Fecha por mes: 3/14/2019
Salida	Todos los datos de los atributos son válidos y se muestran 0 incidencias en la tabla. El especialista no tiene incidencias asignadas que cumplan con los parámetros establecidos.

3.4 Conclusiones del capítulo

En este capítulo se detallaron los estándares de codificación empleados en la implementación del sistema de gestión de reportes lo que permitió obtener una mejor organización y con ello una mejor comprensión del código, a la vez que posibilita su reutilizado por desarrolladores de otros proyectos. Se obtuvo un diagrama de componentes a partir del cual se logró definir la estructura general del sistema y el comportamiento de los componentes.

La descripción y realización de las pruebas posibilitó asegurar la calidad del software, obtener resultados satisfactorios y encontrar los errores existentes para poder solucionarlos, permitiendo así que el sistema implementado tenga la calidad y robustez requerida.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se cumplieron cada uno de los objetivos trazados, distinguiéndose de manera general los siguientes aspectos:

- El estudio de los elementos teóricos vinculados a la problemática a resolver, en conjunto con el estudio de los sistemas homólogos permitieron sentar las bases teóricas, definiendo funcionalidades y características que resultaron provechosas para el desarrollo del sistema de gestión de reportes, definiéndose este como un conjunto de procesos, comportamientos y herramientas que permiten organizar y exhibir la información contenida en una base de datos.
- El análisis de las tecnologías, herramientas y metodologías utilizadas en la actualidad, así como la selección de los requisitos funcionales y no funcionales que debían ser implementados constituyeron elementos necesarios asociados al desarrollo del sistema de gestión de reporte sobre la actividad de soporte, pudiendo definir cuáles se adecuaban mejor a los procesos productivos de la universidad y cumplían con las necesidades del centro de soporte.
- Las pruebas realizadas al sistema de gestión de reportes implementado permitieron verificar el funcionamiento y garantizaron la usabilidad del mismo, evidenciaron que cumple con los requerimientos definidos, estas permitieron la corrección de las no conformidades encontradas.

RECOMENDACIONES

Concluida la investigación, con el objetivo de perfeccionar la aplicación y lograr una mayor explotación de sus potencialidades se recomienda:

- Implementar una funcionalidad automatizada que notifique vía correo cuando se asigne una incidencia a un especialista.
- Implementar plantillas para la creación de reportes con diferentes diseños.

REFERENCIAS BIBLIOGRÁFICAS

Aubry C. HTML5 y CSS3: para sitios con diseño web responsive [Libro]. - [s.l.] : Ediciones ENI, 2014.

Ávila Negrete E.A. Manual de Instalacion de Pycharm [En línea]. - 2020. - 21 de marzo de 2020. - https://es.scribd.com/doc/300360377/Manual-de-Instalacion-de-Pycharm.

Basalo Alberto Manual de AngularJS [Libro]. - 2016.

Bastida Lugones Lázara Criterios epistemológicos acerca de la actividad pedagógica profesional [Libro]. - La Habana : Pueblo y educación , 2009.

Berzal Fernando y Cortijo Francisco José y Cubero, Juan Carlos Desarrollo profesional de aplicaciones web [Libro]. - 2011.

Botero Chica Carlos Alberto Cinco tendencias de la gestión educativa [Publicación periódica] // Revista Iberoamericano de Educación. - 2009. - ISSN: 1681-5653.

Craig Lerman UML y patrones:una introducción al análisis y diseño orientado a objetos y al proceso unificado [Libro]. - La Habana : [s.n.], 2015.

Crockford Douglas JavaScript: The Good Parts [Libro]. - [s.l.] : O'Reilly Media, 2008. - 1st Edición.

De la Peña Natalia Calvo Gestión y control de los sistemas de información [Libro]. - España : [s.n.], 2015.

Foundation Python Software Introducción — Tutorial de Python 3.6.3 documentation. [En línea]. - 2017. - 15 de marzo de 2020. - http://docs.python.org.ar/tutorial/3/real-index.html..

Fuente Mario Luis y Hernández, Pedro Enrique Sistema integrado para la monitorización de MongoDB y CouchDB [Libro]. - La Habana : UCI, 2014.

Gauchat J.D. El gran libro de HTML5, CSS3 y Javascript [Libro]. - 2012.

Holovaty A. y Kaplan-Moss, J. The definitive guide to Django: Web development done right [Libro]. - 2009. - ISBN 1-4302-1937-8.

IEEE.R. Recommended Practice for Software Requirements Specifications [En línea]. - 2019. - http://www.ieee.org/index.html?WT.mc_id=hpf_logo..

Juran J M. Juran y la calidad por el diseño [Libro]. - España : [s.n.], 1996.

Lerman Craig UML y Patrones [Libro]. - México : [s.n.], 1999.

María L. J. Pruebas de caja blanca [Libro]. - 2009.

Merino Julián Pérez Porto y María Definición de actividad [En línea]. - 2015. - marzo de 2020. - https://definicion.de/actividad/.

Microsoft Remote Development FAQ [En línea]. - 2020. - https://code.visualstudio.com/docs/remote/faq/.

Microsoft What is Visual Studio Codespaces [En línea]. - 2020. - https:/docs.microsoft.com/en-us/visualstudio/codespaces/resources/faq/ .

Mohedano J., Saiz, J.M. y Román, P.S. Iniciación a Javascript. [Libro]. - [s.l.] : S.l.: Ministerio de Educación, 2012. - ISBN 978-84-369-5433-3.

Ogalla Francisco Segura Sistema de gestión, una guía práctica [Libro]. - España : [s.n.], 2005.

Ortiz Ricardo Iraini y Pantoja Guerrero, Yunior Red social de investigadores para la gestión del conocimiento en el contexto de la Universidad de las Ciencias Informáticas [Libro]. - La Habana : [s.n.], 2014.

Paradigm Visual isual Paradigm Product Overview [En línea]. - 2020. - 15 de marzo de 2020. - https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html. .

Paz Renata Causo Atención al cliente. Guías Prácticas de Técnicas y Estrategias [Libro]. - España : [s.n.], 2007.

Pérez G., Mario, E., Torres Gálvez, A., Acosta, M. y Ángel M. Administracion y optimizacion de un Sistema de Base de Datos Descentralizado, en PostgreSQL [En línea]. - 2008. - 28 de octubre de 2019. - http://repositorio.uci.cu/jspui/handle/ident/TD_1358_08.

Pérez Julián Porto y Merino, María Definición de reporte [En línea]. - 2010. - https://definición.de/reporte/.

PGAdmin PostgreSQL Global Development Group. [En línea]. - 2016. - http://www.pgadmin.org/..

Postgresql D. Installing PostgreSQL. , Qué son las historias de usuario y su función en Agilidad. Solving Ad Hoc - Resolviendo a medida tus necesidades de cambio [En línea]. - 2015. - 22 de marzo de 2020. - https://solvingadhoc.com/las-historias-usuario-funcion-agilidad/.

Pozo Guevara Macdemis Liliana Interfaces para el módulo de Administración del Sistema de Gestión de Servicios de Soporte del Centro de Soporte UCI [Libro]. - La Habana : UCI, 2014.

Pressman R.S. ngeniería del software: un enfoque práctico [En línea]. - 2013. - 15 de marzo de 2020. - http://www.ingebook.com/ib/NPcd/IB_BooksVis?cod_primaria=1000187&codigo_libro=4272.

Qué es el soporte técnico informático [En línea]. - 2020. - https://www.z-net.com.ar/blog-post/que-es-el-soporte-tecnico-informatico/ .

Qué es Frontend y Backend [En línea]. - 2017. - https://platzi.com/blog/que-es-frontend-y-backend/.

Quintana Rondón Yoandri, Camejo Domínguez, Lianet y Díaz Berenguer, Abel Diseño de la Base de Datos para Sistemas de Digitalización y Gestión de Medias [Publicación periódica] // Revista de Informática Educativa y Medios Audiovisuales. - Argentina : [s.n.], 2011. - Vol. 8(15). - págs. 17-25.

R.L. Covantec Ventajas y desventajas — Materiales del entrenamiento de programación en Python - Nivel básico. [En línea]. - 2018. - 15 de marzo de 2020. - https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/ventajas_desventajas.html. .

Ramos Alicia Martín y Ramos, Ma. Jesús Martín Aplicaciones Web [Libro]. - 2014. - 2da.

Ramos Maimir Mesa Inauguración Feria Internacional Informática [Entrevista]. - La Habana : [s.n.], 2018.

RF.RNF Requerimientos Funcionales y No Funcionales (RF/RNF) [En línea]. - 2016. - http://ingenieriadesoftware.bligoo.com.mx/requerimientosfuncionales-y-no-funcionales-rf-rnf.

Sánchez Torres, Jenny Maricela, González Zabala, Mayda Patricia y Sánchez Muñoz, María Paloma. Madrid: La Sociedad de la Información: Génesis, Iniciativas, Concepto y su relación con las TIC [Publicación periódica] // UIS Ingenierías. - 2012. - 1 : Vol. 11.

Software Prácticas de Patrones GRASP [En línea]. - 2011. - 2 de noviembre de 2019. - http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

Spurlock J. Bootstrap: responsive web development [Libro]. - [s.l.] : O'Reilly Media, 2013.

Talledo José San Miguel Certificado de profesionalidad IFCDO210 desarrollo de aplicaciones con tecnologías web MF0493_3 [Libro]. - España : [s.n.], 2015.

Turner Raymond "The Foundations of Specification" [Publicación periódica]. - octubre de 2005. - 5 : Vol. 15. - págs. 623-663.

UCI Sitio oficial [En línea]. - Universidad de las Ciencias Informáticas, 2019. - https://uci.cu/.

UCI Soporte Centro de Soporte [En línea]. - Universidad de las Ciencias Informáticas, 2019. - https://soporte.uci.cu/misión.

ANEXOS

Anexo 1: Entrevista a especialista del centro de soporte

Estimado especialista: Se necesita de su cooperación en una investigación para una tesis de pregrado. Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Cuáles son los procesos de la institución?
2. ¿Cuáles son las principales actividades que se realizan, cómo se relacionan estas actividades con las del resto de la organización?
3. ¿Existe algún mecanismo estructurado para la gestión de las incidencias?
4. ¿Cuáles son los problemas fundamentales que impiden el buen funcionamiento de la gestión de los reportes?
5. ¿Cuentan con alguna norma y\o resolución que incida en la gestión de reportes?
6. ¿Considera que se le deba mejorar algo al sistema actualmente implementado para la gestión de reportes?

Anexo 2: Historia de usuario

Tabla15: Historia de Usuario “Obtener incidencias por rango de fecha”.

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Obtener reporte de incidencias por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Muestra todas las incidencias dado un rango de fecha determinada.	
Observaciones: • **Se debe introducir al sistema un rango de fecha determinado.**	
Prototipo de interfaz:	

Tabla16: Historia de usuario “Obtener incidencias por especialista”.

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Obtener reporte de incidencias por especialista
Modificación de Historia de Usuario Número: ninguna	

Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 0.4semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Muestra todas las incidencias dado un especialista.	
Observaciones: • **Se debe introducir al sistema el nombre de un especialista.**	
Prototipo de interfaz:	

Tabla17: Historia de usuario ”Obtener reporte de incidencias por sistema”.

Historia de Usuario	
Número: HU_5	**Nombre Historia de Usuario:** Obtener reporte de incidencias por sistema
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Muestra todas las incidencias de un sistema.	
Observaciones: • **Se debe introducir el nombre del sistema.**	
Prototipo de interfaz:	

Tabla18: Historia de usuario ”Obtener reporte de incidencia en espera”.

Historia de Usuario	
Número: HU_6	**Nombre Historia de Usuario:** Obtener reportes de incidencias en espera
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 0.4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 0.4 semanas
Descripción: Permite obtener las incidencias que se encuentran en espera	
Observaciones: • **Se debe especificar como tipo de estado de incidencia: en espera.**	
Prototipo de interfaz:	

Tabla19: Historia de usuario ”Crear reporte personalizado”.

Historia de Usuario	
Número: HU_7	**Nombre Historia de Usuario:** Crear reporte personalizado.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite la creación de reportes.	
Observaciones: • **El usuario debe estar autenticado.** • **El usuario debe poseer los permisos necesarios para realizar la gestión de reportes.**	
Prototipo de interfaz:	

Tabla20: Historia de usuario ”Modificar reporte personalizado”.

Historia de Usuario	
Número: HU_8	**Nombre Historia de Usuario:** Modificar reporte personalizado.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite la modificación de reportes.	
Observaciones: • **El usuario debe estar autenticado.** • **El usuario debe poseer los permisos necesarios para realizar la gestión de reportes.**	
Prototipo de interfaz:	

Tabla21: Historia de usuario ”Eliminar reporte personalizado”.

Historia de Usuario	
Número: HU_9	**Nombre Historia de Usuario:** Eliminar reporte personalizado.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1

Prioridad en Negocio: Muy Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite eliminar reportes.	
Observaciones: • **El usuario debe estar autenticado.** • **El usuario debe poseer los permisos necesarios para realizar la gestión de reportes.**	
Prototipo de interfaz:	

Tabla22: Historia de usuario ”Exportar a pdf reporte de incidencias por rango de fecha”.

Historia de Usuario	
Número: HU_10	**Nombre Historia de Usuario:** Exportar a pdf reporte de incidencias por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 2 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema un rango de fecha determinado.**	
Prototipo de interfaz:	

Tabla 23: Historia de usuario “Exportar a pdf reporte de incidencias por especialista”.

Historia de Usuario	
Número: HU_12	**Nombre Historia de Usuario:** Exportar a pdf reporte de incidencias por especialista.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema el nombre de un especialista.**	
Prototipo de interfaz:	

Tabla 24: Historia de usuario “Exportar a pdf reporte de incidencias por sistema”.

Historia de Usuario	
Número: HU_13	**Nombre Historia de Usuario:** Exportar a pdf reporte de incidencias por sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir el nombre del sistema.**	
Prototipo de interfaz:	

Tabla 25: Historia de usuario “Exportar a pdf reporte de incidencias en espera”.

Historia de Usuario	
Número: HU_14	**Nombre Historia de Usuario:** Exportar a pdf reporte de incidencias en espera.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe especificar como estado de incidencia: en espera.**	
Prototipo de interfaz:	

Tabla 26: Historia de usuario “Exportar a excel reporte de incidencias por mes”.

Historia de Usuario	
Número: HU_16	**Nombre Historia de Usuario:** Exportar a excel reporte de incidencias por mes.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 1.0 semanas

Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema un mes determinado.**	
Prototipo de interfaz:	

Tabla 27: Historia de usuario "Exportar a excel reporte de incidencias por especialista".

Historia de Usuario	
Número: HU_17	**Nombre Historia de Usuario:** Exportar a excel reporte de incidencias por especialista.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 1.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir al sistema el nombre del especialista.**	
Prototipo de interfaz:	

Tabla 28: Historia de usuario "Exportar a excel reporte de incidencias por sistema".

Historia de Usuario	
Número: HU_18	**Nombre Historia de Usuario:** Exportar a excel reporte de incidencias por sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 1.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe introducir el nombre de un sistema.**	
Prototipo de interfaz:	

Tabla 29: Historia de usuario "Exportar a excel reporte de incidencias en espera".

Historia de Usuario	
Número: HU_19	**Nombre Historia de Usuario:** Exportar a excel reporte de incidencias en espera.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos estimados:** 1.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.0 semanas
Descripción: Permite almacenar las incidencias en un dispositivo externo al sistema.	
Observaciones: • **Se debe especificar como tipo de estado de incidencia en espera**	
Prototipo de interfaz:	

Tabla 30: Historia de usuario “Graficar reporte de incidencias por rango de fecha”.

Historia de Usuario	
Número: HU_20	**Nombre Historia de Usuario:** Graficar reporte de incidencias por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite visualizar la información mediante gráficos.	
Observaciones: • **Se debe introducir al sistema un rango de fechas determinado.**	
Prototipo de interfaz:	

Tabla 31: Historia de usuario “Graficar reporte de incidencias por mes”.

Historia de Usuario	
Número: HU_21	**Nombre Historia de Usuario:** Graficar reporte de incidencias por mes.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2.0 semanas

Descripción: Permite visualizar la información mediante gráficos.
Observaciones: • **Se debe introducir al sistema un mes determinado.**
Prototipo de interfaz:

Tabla 32: Historia de usuario “Graficar reporte de incidencias por sistema”.

Historia de Usuario	
Número: HU_23	**Nombre Historia de Usuario:** Graficar reporte de incidencias por sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite visualizar la información mediante gráficos.	
Observaciones: • **Se debe introducir el nombre del sistema.**	
Prototipo de interfaz:	

Tabla 33: Historia de usuario “Graficar reporte de incidencias en espera”.

Historia de Usuario	
Número: HU_24	**Nombre Historia de Usuario:** Graficar reporte de incidencias en espera.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 2.0 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2.0 semanas
Descripción: Permite visualizar la información mediante gráficos.	
Observaciones: • **Se debe especificar como estado de incidencia: en espera.**	
Prototipo de interfaz:	

Tabla 34: Historia de usuario “Análisis estadístico para reportes por rango de fecha”.

Historia de Usuario	
Número: HU_25	**Nombre Historia de Usuario:** Análisis estadístico para reportes por rango de fecha.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 3.0 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 3.0 semanas
Descripción: Permite estimar una probabilidad de próximas incidencias.	
Observaciones: • **Se debe introducir al sistema un rango de fechas determinado.**	
Prototipo de interfaz:	

Tabla 35: Historia de usuario “Análisis estadístico para reportes por mes”.

Historia de Usuario	
Número: HU_26	**Nombre Historia de Usuario:** Análisis estadístico para reportes por mes.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 3.0 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 3.0 semanas
Descripción: Permite estimar una probabilidad de próximas incidencias.	
Observaciones: • **Se debe introducir al sistema un mes determinado.**	
Prototipo de interfaz:	

Tabla 36: Historia de usuario “Análisis estadístico para reportes por tipo de especialista”.

Historia de Usuario	
Número: HU_27	**Nombre Historia de Usuario:** Análisis estadístico para reportes por especialista.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 3.0 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 3.0 semanas

Descripción: Permite estimar una probabilidad de próximas incidencias.
Observaciones: • **Se debe introducir al sistema el nombre de un especialista.**
Prototipo de interfaz:

Tabla 37: Historia de usuario “Análisis estadístico para reportes por tipo de sistema”.

Historia de Usuario	
Número: HU_28	**Nombre Historia de Usuario:** Análisis estadístico para reportes por sistema.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 3.0 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 3.0 semanas
Descripción: Permite estimar una probabilidad de próximas incidencias.	
Observaciones: • **Se debe introducir el nombre del sistema.**	
Prototipo de interfaz:	

Tabla 38: Historia de usuario “Análisis estadístico para reportes por incidencias en espera”.

Historia de Usuario	
Número: HU_29	**Nombre Historia de Usuario:** Análisis estadístico para reportes por incidencias en espera.
Modificación de Historia de Usuario Número: ninguna	
Usuario: Jorge Alonso Podadera	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos estimados:** 3.0 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 3.0 semanas
Descripción: Permite estimar una probabilidad de próximas incidencias.	
Observaciones: • **Se debe especificar como estado de incidencia: en espera.**	
Prototipo de interfaz:	