Universidad de las Ciencias Informáticas

Facultad de Ciencias y Tecnologías Computacionales

Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA

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

Autora: Tania Elena Castro Aguilera

Tutores: Msc. Dionis Pérez Pérez

Ing. Bárbara Sepúlveda Mateo

Ing. Bernardo Hernández González

La Habana, julio de 2017

"Año 59 de la Revolución"

Declaración de Autoría

Declaro por este medio que yo, Tania Elena Castro Aguilera, soy la autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría en La Habana a los 6 días del mes de julio del año 2017.

_____

Tania Elena Castro Aguilera
Autora

_____

Msc. Dionis Pérez Pérez
Tutor

_____

Ing. Bárbara Sepúlveda Mateo
Tutora

_____

Ing. Bernardo Hernández González
Tutor

Datos de Contacto

Tania Elena Castro Aguilera

Grupo AZCUBA, La Habana, Cuba.

E-mail: taniaec@uci.cu

Dionis Pérez Pérez

Grupo AZCUBA, La Habana, Cuba.

E-mail: dionis.perez@azcuba.cu

Bárbara Sepúlveda Mateo

Grupo AZCUBA, La Habana, Cuba.

E-mail: barbara.sepulveda@azcuba.cu

Bernardo Hernández González

Universidad de Ciencias Informáticas UCI

E-mail: b.hernandez@uci.cu

AGRADECIMIENTOS

A mi hijo por todos los días que triste o feliz estuvo conmigo dando clases, para que hoy yo este donde estoy, gracias mi niño.

A mis padres por su apoyo.

A mimi Vicenta, porque ha sido otra madre para mí.

A mis maestros durante toda la carrera, que me ayudaron y enseñaron, gracias por toda su paciencia.

A Yanelis Benítez por ser la madre de todos los poco inteligentes del CPE, y confiar siempre que lo lograríamos, aunque no supiéramos que $\star = \,$ *, gracias.*

A Yadira que además de buena jefa, ha sido una magnífica amiga.

A mis compañeros de aula durante toda la carrera, la loca de Arlen, el puntualito de Andry, Maribel, Karen, Yohana la seria, el travieso Akell que fue el primero en irse, Yaima, mis negros Ernesto y Roberto, esta carrera no habría sido lo mismo sin ustedes, a Pedro que no por haber llegado de ultimo a mi grupo de amigos ha sido menos. Agradecida por todo su apoyo siempre que el diablito mío estuvo en el aula dando clases con nosotros, para poder estar aquí hoy, sin dudas se lo agradezco de corazón.

A mis amigas de la secretaria, Fifi, Isbel, Yuliet, Yeni por todas las risas compartidas.

A todas las personas que he conocido en estos largos años de la carrera por aportar siempre un poco en mi crecimiento como mejor persona.

A todos los que por mala memoria que tengo no menciono, gracias por estar para mí cuando lo necesité.

DEDICATORIA

A mi hijo, por ser mi motor impulsor en esta etapa de estudios.

A mis padres, por estar siempre conmigo.

A mis hermanas y hermano.

RESUMEN

Con el creciente desarrollo de las tecnologías de la información y las comunicaciones, diversos procesos que se llevan a cabo en las organizaciones se han favorecido sustancialmente, al punto de ser la automatización sinónimo de eficiencia. En la actualidad, el control de incidencias en el Grupo Azucarero AZCUBA, se realiza de forma manual. Esto genera demora en la gestión, pérdida de información y falta de claridad en el análisis de los informes, influyendo negativamente en las producciones azucarera y agroindustrial a nivel nacional. En el siguiente trabajo se describe un sistema informático que permitirá la gestión de incidencias del Grupo Azucarero AZCUBA. El sistema será desarrollado con tecnologías libres: Javascript ES 6 como lenguaje de desarrollo, SailJS y AngularJSv1.5.4 como marcos de trabajo, NodeJS v7 como entorno de ejecución y PostgreSQL v9.3 como gestor de bases de datos. Como herramientas serán utilizadas: Visual Paradigm 8.0 para la construcción de los artefactos generados por la metodología y NetBeans v8.0 como entorno integrado de desarrollo. La metodología OpenUP será utilizada para guiar el proceso de desarrollo. Además, se hacen un conjunto de pruebas al sistema para validar su correcto funcionamiento.

PALABRAS CLAVE: control de incidencias, Grupo Azucarero AZCUBA, sistema informático de gestión.

Abstract

With the growing development of the technologies of information and communication, various processes carried out in organizations have been favored substantially, to the point of being automation synonymous with efficiency. Nowadays, the control of incidents in the Sugar Group AZCUBA, is done manually. This leads to delays in management, loss of information and lack of clarity in the analysis of reports, negatively influencing sugar and agroindustrial production at national level. The following work describes a software that will allow incident management for the Sugar Group AZCUBA. The software will be developed with free technologies: Javascript ES 6 as development language, SailJS and AngularJS v1.5.4 as frameworks, NodeJS v7 as runtime environment and PostgreSQL v9.3 as database manager. Visual Paradigm 8.0 for the construction of the artifacts generated by the methodology and NetBeans v8.0 as an integrated development environment, will be used. The OpenUP methodology will guide the development process. In addition, a set of tests are done to the system to validate its correct functioning.

KEY WORDS: control of incidents, Sugar Group AZCUBA, management software.

Índice de contenido

Índice de Tablas

No se encuentran elementos de tabla de ilustraciones.

Índice de Figuras

Introducción

Con el triunfo de la revolución cubana en 1959, se produjeron grandes cambios en la economía del país, y una de las industrias que garantizó el comercio de Cuba con el mundo fue la industria azucarera, la cual es de las más importantes de la producción agroindustrial cubana de todos los tiempos, además de ser una de las principales fuentes de ingreso a la economía del país. En el año 2011 a raíz del proceso de perfeccionamiento empresarial se disuelve el Ministerio del Azúcar y se crea la Organización Superior de Dirección para la Agroindustria Azucarera en forma abreviada AZCUBA, la cual se crea con los bienes y recursos del extinto Ministerio del Azúcar, además de los que integran su patrimonio como son sus empresas agroindustriales, productivas, de servicios, comerciales y de proyecto, contando también con los centros de Investigación de la Caña de Azúcar (INICA), el centro de Investigación de los Derivados de la Caña de Azúcar (ICIDCA) y el Centro de Capacitación (CNCA). El Grupo Azucarero AZCUBA forma parte de la administración central del estado, por su importancia económica y productiva. Su misión es la producción de azúcar y derivados de la caña, energía eléctrica y alimento animal, con calidad y costos competitivos, aplicando la ciencia y la técnica y protegiendo el medio ambiente (AZCUBA 2014).

Las Tecnologías de la Información y las Comunicaciones (TICs) están cada vez más presentes en el mundo moderno con implicaciones en cada una de las ramas de la sociedad y el sector azucarero no está exento de este cambio, con lo que se hace más fácil en muchas de las tantas áreas que tiene este sector el trabajo a realizar, al estar las empresas sumidas en estos cambios tecnológicos pues se tiene que priorizar su correcto estado utilizable, por lo que un mal funcionamiento o la interrupción de estos servicios pueden llegar a tener importantes consecuencias en la obtención de los objetivos de la empresa.

En el Grupo Azucarero AZCUBA, entre las distintas direcciones que tiene se encuentra la Dirección de Informática, Comunicaciones y Análisis, constituida por el conjunto de salas de control y análisis existentes en el grupo, sus empresas y Unidades Empresariales de Base (UEB) y constituye el centro de dirección, información y análisis que mediante la información operativa y estadística a tiempo real, controla las 24 horas del día el flujo de los procesos, la ejecución de la zafra azucarera y las labores agropecuarias, los hechos extraordinarios y las situaciones de excepción, llamándolas incidencias en general, generando los análisis que faciliten la toma de decisiones a la dirección, y rectorando la actividad de informática, comunicaciones y archivo (AZCUBA 2012).

En estos momentos para llevar a cabo la recepción de las incidencias se utiliza una plantilla en formato Word que la envían las direcciones que trabajan directamente con el grupo, detallando la información de lo ocurrido, con otras informaciones como quien lo reporta, fecha, hora, cargo del encargado de reportar el

hecho, que no siempre es quien lo informa, y un número consecutivo, para poder darle seguimiento. Luego esa información se transcribe hacia el sistema actual creado en Visual FoxPro lo que provoca una brecha al dar la posibilidad de introducir errores al copiar y pegar, además este sistema utilizado por los directivos del nivel central solamente permite, registrar la información y generar reportes sencillos, no siendo la solución más eficiente para esta problemática porque no permite que en las unidades subordinadas se puedan crear los reportes mediante el mismo sistema, hay que hacerlo manualmente, por lo que ocurren errores humanos, como falta de información o perdida de la misma, inconsistencia, falta de claridad en los datos, por lo que se hace difícil el correcto manejo de este tipo de información en algunos casos de gran importancia para el país, ya sea en costo monetario o de recursos materiales, ejemplo de esto la quema de la caña de azúcar, influyendo negativamente en los análisis que podrían realizar los especialistas para tomar alguna decisión.

Por lo antes planteado se deriva el **problema a resolver**:

¿Cómo contribuir al control de las incidencias en el Grupo Azucarero AZCUBA?

Por lo que el **objeto de estudio** lo constituyen los sistemas para la gestión de incidencias, enmarcado en el **campo de acción** los sistemas para la gestión de incidencias para la industria azucarera.

Se define como **objetivo general** desarrollar un sistema informático para la gestión de las incidencias en el Grupo Azucarero AZCUBA. De este se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico conceptual relacionado con la gestión de incidencias en la industria azucarera.
- Realizar el análisis y diseño del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.
- Implementar el Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.
- Aplicar pruebas para comprobar las funcionalidades de la solución.

Es preciso lograr el cumplimiento de las siguientes **tareas de investigación**, en aras de lograr materializar los objetivos planteados.

- Análisis de los conceptos asociados a la gestión de incidencias.
- Caracterización de sistemas similares para gestionar las incidencias.
- Selección de las tecnologías, metodología y herramientas a utilizar en el desarrollo del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.

- Definir métodos teóricos y empíricos que serán utilizados en el diseño metodológico de la investigación.
- Identificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.
- Realización del análisis y el diseño del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.
- Implementación de las funcionalidades del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.
- Validar el sistema mediante pruebas de aceptación.

En el transcurso de la investigación se utilizaron distintos métodos que entran dentro de la clasificación de empíricos, destacándose entre ellos la observación. Debido a su utilidad en la comprensión del flujo de información recibida es en definitiva este método el que permitió conocer los aspectos en los que esta actualización tecnológica permitiría mejorar el proceso de recepción y uso de la información en general.

De acuerdo con las necesidades enfrentadas en la fase investigativa se acudió también a métodos que entran en la clasificación de teóricos como el inductivo-deductivo para comprender cómo el framework de desarrollo, utilizado en el lenguaje de programación realiza el procesamiento de las peticiones. Además, se utilizó el método analítico-sintético para esgrimir las funcionalidades necesarias para cumplimentar los objetivos y satisfacer las necesidades del cliente.

En aras de mostrar de manera clara y concisa todo el proceso de investigación, desarrollo y pruebas, el documento queda estructurado de la siguiente manera:

Capítulo 1: Fundamentos teóricos de la gestión de incidencias en la industria azucarera en Cuba.

En este capítulo se exponen los principales conceptos asociados a la gestión de información, gestión de incidencias y toma de decisiones. Además se realiza un análisis de las principales soluciones existentes, con el objetivo de identificar las tendencias y las características de estas que pudieran aportar al diseño de la solución. Finalmente se explica la selección de la metodología, herramientas y tecnologías empleadas para el desarrollo de la solución.

Capítulo 2: Análisis y diseño del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.

En este capítulo se realiza el modelo de negocio, el cual constituye el punto de partida para el desarrollo de la propuesta de solución. Se definen los requisitos funcionales y no funcionales. Se desarrolla el

modelo de casos de uso, donde se exponen también los diferentes patrones utilizados. Se especifican los elementos fundamentales de la arquitectura y el diseño, los patrones de diseño empleados y el modelo de datos del sistema. Al concluir este capítulo se contará con una comprensión de los problemas actuales del negocio y una idea pertinente del sistema a desarrollar.

Capítulo 3: Implementación y pruebas del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.

En este capítulo se abordan los principales aspectos referentes a las fases de implementación y pruebas. Se describe la distribución física del sistema propuesto a través del modelo de despliegue y se representa el sistema dividido en componentes y dependencias entre estos mediante el diagrama de componentes. Se especifican las pruebas realizadas a la solución y sus resultados, con el objetivo de determinar no conformidades, erradicarlas y probar que el sistema satisface sus requisitos funcionales y no funcionales.

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

Capítulo 1: Fundamentos teóricos de la gestión de incidencias en la industria azucarera en Cuba.

En este capítulo se exponen los principales conceptos asociados a la gestión de información, gestión de incidencias y toma de decisiones. Además se realiza un análisis de las principales soluciones existentes, con el objetivo de identificar las tendencias y las características de estas que pudieran aportar al diseño de la solución. Finalmente se define la metodología de desarrollo de software y las principales tecnologías y herramientas para su implementación y despliegue. El objetivo de este trabajo de diploma es la implementación de un sistema de gestión de incidencias, por lo que se hace necesario definir qué se entiende por gestión de incidencias.

1.1 Conceptos asociados

1.1.1 Gestión de Información

La Gestión de la Información es un proceso mediatizado por un conjunto de actividades que permiten la obtención de información, lo más pertinente, relevante y económica posible, para ser usada en el desarrollo y el éxito de una organización. Genera nuevos conocimientos, es ir en busca de nuevos significados, análisis, aplicar el principio de que el todo, es más que la suma de las partes. Es producir un impacto en el ambiente de cualquier organización.

Es un proceso que debe estar presente en cada uno de los pasos de la organización, es un proceso y a la vez un subproceso. Requiere acción, decisión y evaluación.

Mediante la gestión se proporcionan los recursos de información necesarios para una buena toma de decisiones, se desarrollan nuevos conocimientos que posibilitan calidad y eficiencia en los servicios y productos de las organizaciones.

La Dra. Gloria Ponjuán define que cuando se menciona gestión de información se refiere a la gestión que se desarrolla en un Sistema de Información, si se trata de que el sistema tenga como propósito obtener salidas informacionales. Además la define como el proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales), para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico la gestión del ciclo de vida de este recurso y ocurre en cualquier organización. Es propia también de unidades especializadas que manejan este recurso en forma intensiva, llamadas unidades de información (PONJUÁN DANTE 1998).

1.1.2 Sistema de Gestión de Información

Davis y Olson conceptualizan sistema de gestión de información como “un sistema integrado y automatizado para proveer la información que sostenga las funciones de operatividad, gestión y toma de decisiones en una organización” (DAVIS and OLSON 1984).

Un sistema de gestión de información permite la gestión de los recursos de información tanto internos como externos. Su finalidad es generar servicios y productos que respondan a las necesidades y sobrepasen las expectativas de los usuarios, posibilitando que el sistema trabaje eficientemente y económicamente a la vez. Es una herramienta que permite optimizar recursos, reducir costes y mejorar la productividad de una empresa, permitiendo tomar decisiones para corregir fallos y prevenir la aparición de gastos innecesarios.

1.1.3 Gestión de incidencias

La Gestión de Incidencias tiene como objetivo resolver, de la manera más rápida y eficaz posible, cualquier incidente que cause una interrupción en el servicio. La Gestión de Incidencias no debe confundirse con la Gestión de Problemas, pues a diferencia de esta última, no se preocupa de encontrar y analizar las causas subyacentes a un determinado incidente sino exclusivamente a restaurar el servicio. Sin embargo, es obvio, que existe una fuerte interrelación entre ambas.

Gestión de incidencias es cualquier evento que no forma parte de la operación estándar de un servicio y que causa, o puede causar, una interrupción o una reducción de calidad del mismo.

Por lo mismo, tenemos que un sistema de seguimiento de incidentes es un paquete de software que administra y mantiene listas de incidentes, conforme son requeridos por una institución. Típicamente el ticket tiene un número único de referencia, también conocido como un número de caso, incidente o reporte de llamada, el cual es usado para permitir al cliente o al personal de soporte localizar, añadir o comunicar el estado del incidente o requerimiento.(4)

Un sistema de gestión sin importar el entorno en que se utilice es sin duda una herramienta que se hace indispensable para mantener un buen funcionamiento del negocio. Sin embargo es el propio sistema de gestión por donde debe siempre comenzar el proceso de mejora, asegurando de esta forma que el control de la actividad que se realiza no derive en costos innecesarios, ya sea de tiempo o monetarios. Es por esto que la informatización de estos sistemas ha presentado un impacto positivo debido a que ha ofrecido la posibilidad de llegar, incluso en los procesos más complejos, a permitir controles en tiempo real con una precisión muchas veces mayor que los antiguos sistemas pese a que estos últimos conllevaban más

tiempo de análisis y cálculos. Ante una necesidad de este tipo la industria del software no tardó en dar respuesta, por lo que en la actualidad existen varios productos informáticos con este objetivo.

1.1.4 Toma de decisiones

Una decisión es la elección de la alternativa más adecuada de entre varias posibilidades con el fin de alcanzar un estado deseado, considerando la limitación de recursos. La palabra decisión deriva del término decido que significa cortar; referido al concepto actual, se entiende que se "corta" una alternativa finalmente elegida.

La toma de decisiones a nivel individual se caracteriza por el hecho de que una persona haga uso de su razonamiento y pensamiento para elegir una solución a un problema que se le presente en la vida; es decir, si una persona tiene un problema, deberá ser capaz de resolverlo individualmente tomando decisiones con ese específico motivo. También, la toma de decisiones es considerada como una de las etapas de la dirección.

En la toma de decisiones importa la elección de un camino a seguir, por lo que en un estado anterior deben evaluarse alternativas de acción. Si estas últimas no están presentes, no existirá decisión. Para tomar una decisión, cualquiera que sea su naturaleza, es necesario conocer, comprender, analizar un problema, para así poder darle solución. En algunos casos, por ser tan simples y cotidianos, este proceso se realiza de forma implícita y se soluciona muy rápidamente, pero existen otros casos en los cuales las consecuencias de una mala o buena elección pueden tener repercusiones en la vida y si es en un contexto laboral en el éxito o fracaso de la organización, para los cuales es necesario realizar un proceso más estructurado que puede dar más seguridad e información para resolver el problema.

1.2 Soluciones Similares

Los sistemas informáticos facilitan las actividades de cualquier empresa o compañía, estos contribuyen a la optimización de los resultados y mejor aprovechamiento de los recursos, buscando mayor eficiencia en los procesos que se realizan y mejores resultados en la producción. En busca de una solución al problema de la investigación planteado se realizó un estudio de sistemas de gestión para incidencias, en el ámbito internacional y nacional. A continuación, se detallan algunos de los sistemas analizados:

1.2.1 Sistemas existentes en el ámbito internacional

Artologik Help Desk

El software Artologik HelpDesk es un sistema de soporte basado en web que le permite centralizar la gestión de incidencias internas y externas. Es un sistema privativo, que facilita a los clientes registrar sus solicitudes en cualquier momento, permitiéndole reducir considerablemente la carga de trabajo de su personal de soporte. Se adapta a cada organización con múltiples ajustes, pero que necesita del acceso a internet todo el tiempo para su uso, por lo que se hace imposible trabajar con este sistema en el país (ARTOLOGIK 2017).

Mantis

Es una aplicación de software libre multiplataforma que permite gestionar las incidencias en una empresa, en un sistema o en un proyecto. Es un sistema fácil de usar y adaptable a varios escenarios. Además cuenta con diferentes plugins que aumentarán la capacidad de trabajo con la herramienta. Cuenta con una gran variedad de funcionalidades que permitirán que todos los objetivos queden cubiertos completamente (MANTISBT 2017).

Jira

Jira es una aplicación comercial basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos. Jira también se utiliza en áreas no técnicas para la administración de tareas. La herramienta fue desarrollada por la empresa australiana Atlassian. Inicialmente Jira se utilizó para el desarrollo de software, sirviendo de apoyo para la gestión de requisitos, seguimiento del estatus y más tarde para el seguimiento de errores. Jira puede ser utilizado para la gestión de procesos y para la mejora de procesos, gracias a sus funciones para la organización de flujos de trabajo (ATLASSIAN 2017).

1.2.2 Sistemas existentes en el ámbito nacional

MainPack 10.0

Es una aplicación informática para la gestión de la actividad de mantenimiento en la industria azucarera. Permite la planificación, organización, control del mantenimiento industrial, las reparaciones y tener un registro de incidencias. Proporciona informaciones que permiten obtener un aumento de la rentabilidad de la empresa, una utilización más eficiente del factor humano, materiales disponibles, y mejoras en el desempeño y fiabilidad de los equipos industriales. Este sistema se encuentra en empresas vinculadas al Grupo Azucarero AZCUBA pero está relacionado solamente con la actividad de mantenimiento. El sistema Mainpack 10.0 fue desarrollado para la plataforma Windows, usando el lenguaje de programación orientado a objetos Object Pascal, de Delphi y el sistema de gestión de base de datos MySQL (ESTUPIÑÁN DÍAZ and VARGAS VARGAS 2015).

Sistema de gestión de incidencias de la UCI

Aplicación web que permite insertar incidencias de mantenimiento, tecnología, redes y servicios telemáticos. Se especifican los detalles de la incidencia y los estados por los cuales transita, pero no permite al usuario realizar alguna modificación. Además no se organizan por el orden de prioridad, requisito necesario para dar respuesta y mejor seguimiento a los reportes realizados.

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

1.2.3 Comparación de las soluciones similares

Se realizaron análisis de varios sistemas de gestión de incidencias que actualmente se encuentran en explotación tanto en el ámbito internacional como nacional. En la tabla 1 se muestra una comparación entre las soluciones analizadas.

Tabla 1. Comparación entre las soluciones existentes.

Solución	Propietario	Plataforma	Reportes	Soporte
Artologik Help Desk	Si	Multiplataforma	Si	Si
Mantis	No	Multiplataforma	Si	Si
Jira	Si	Multiplataforma	No	Si
MainPack	Si	No	Si	Si
Sistema de Gestión de Incidencias de la UCI	No	Multiplataforma	No	No

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

- Son sistemas propietarios, que necesitan de una licencia para su uso.
- Necesitan acceso a internet para su funcionamiento.
- Cuenta con funcionalidades que no se ajustan a las necesidades de la entidad.

A partir de los análisis realizados se concluye con la necesidad de crear un nuevo sistema que permita la gestión de las incidencias en el Grupo Azucarero AZCUBA, debe ser una solución multiplataforma, desarrollarse sobre la base del software libre y debe permitir identificar tendencias o comportamientos relacionados con las incidencias, para contribuir al mejor control de estas.

1.3 Metodología, herramientas y tecnologías

Para desarrollar la solución propuesta en la presente investigación, se hace necesario indagar sobre la metodología, herramientas y tecnologías a ser utilizados. A continuación se procede con el estudio y selección de las mismas.

1.3.1 Metodología de desarrollo

Dentro de las metodologías existen fundamentalmente dos tipos que contribuyen al desarrollo de software, asegurando en un caso y otro la calidad del producto terminado, estas son: las tradicionales y las ágiles. Independientemente de las especificidades de una u otra, las metodologías de desarrollo pudieran definirse de la siguiente manera:

La metodología de desarrollo de software es un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (SOMMERVILLE 2005).

Por su parte las metodologías tradicionales se definen de la siguiente manera: “Las tradicionales están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo; ofrecen por lo general cierta resistencia a los cambios que puedan producirse durante el ciclo de desarrollo del producto, y es un proceso muy bien controlado con muchas políticas y normas” (FIGUEROA *et al.* 2008).

Por otro lado, se pueden encontrar las metodologías ágiles que se basan en dos aspectos fundamentales, el retrasar las decisiones y la planificación adaptativa. Esto posibilita que la documentación necesaria para el proceso de desarrollo se elabore en etapas que pueden ser posteriores a la fase inicial y puedan ser adaptadas a cambios que se presenten en el proceso (FIGUEROA *et al.* 2008).

Las metodologías para el desarrollo del software imponen un proceso sobre el desarrollo de aplicaciones informáticas con el fin de hacerlo más predecible y eficiente. Tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, haciendo énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo “producir lo esperado en el tiempo esperado y con el coste esperado” (PRESSMAN, ROGER S. 2005).

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

Tabla 2 Comparación entre las metodologías de desarrollo de software.

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

A partir de la comparación realizada se define como metodología de desarrollo para la propuesta de solución una metodología de clasificación ágil. Esta decisión se debe a que el equipo de desarrollo está formado por pocos miembros, el director de la dirección de informática, comunicaciones y análisis, quien es considerado el cliente principal, tiene la disposición de interactuar con el equipo mediante reuniones en el momento en que sea necesario.

Las metodologías ágiles generan pocos artefactos y presentan pocos roles, lo que posibilita dedicar más tiempo al proceso de implementación y terminar la solución con mayor rapidez. Otro factor importante es que durante el proceso de desarrollo existe una importante probabilidad de que sucedan cambios en la concepción de algunos elementos relacionados con la propuesta de solución, situación para la cual las metodologías ágiles están especialmente preparadas.

Dentro de las metodologías ágiles más utilizadas y mejor documentadas está OpenUp, la cual conserva las características principales del modelo de desarrollo de la metodología tradicional RUP, incluye el desarrollo iterativo, permite identificar los requisitos operacionales del sistema, prever las interacciones

con los usuarios y prevenir los posibles riesgos en el desarrollo del sistema. Open UP se señala apropiado no solo para proyectos pequeños sino también de bajos recursos, permite disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito y ha sido especialmente diseñada para pequeños equipos (FOWLER 2002).

OpenUP es una forma de desarrollo más ágil y ligera, consiste en equipos a los cuales se les asigna una fase del desarrollo que tienen que complementarse entre sí para obtener un buen producto final, no puede ser una sola persona la que realice todo el trabajo pues esto podría ocasionar que se pierdan de vista ciertas características importantes, por ejemplo para un proyecto pequeño se constituyen equipos de hasta seis personas e implican hasta seis meses de esfuerzo del desarrollo (SAAVEDRA LÓPEZ *et al.* 2013).

La mayoría de los elementos de OpenUP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances. Open UP adopta un enfoque pragmático, con una filosofía ágil que se centra en la naturaleza colaborativa de desarrollo de software. Es una herramienta que diagnóstica, procesos de baja formalidad que puede ser usado tal cual o ampliarse para hacer frente a una amplia variedad de proyectos.

OpenUP es un proceso de desarrollo de *software* mínimamente suficiente, esto quiere decir que incluye solo el contenido fundamental, esto es que no provee orientación sobre temas en los que el proyecto tiene que lidiar, como son: el tamaño del equipo, el cumplimiento, seguridad, orientación tecnológica entre otras. Sin embargo, OpenUP es completa en el sentido de que manifiesta por completo el proceso de construir un sistema(ECLIPSE 2011).

Para atender las necesidades que no están cubiertas en su contenido OpenUp es extensible a ser utilizado como base sobre la cual se pueden añadir o adaptarse a contenido de otro proceso que sea necesario. Open UP nos ofrece una metodología ágil y flexible, que se puede acoplar a la mayoría de proyectos, además que cubre aspectos como la seguridad y contratación de personal, también incluye a otras personas interesadas en el proyecto o parte de, tiene como desventajas que a veces omite contenido que puede ser de interés en el proyecto, además que se espera que cubra un amplio sistema de necesidades para los proyectos de desarrollo en un plazo muy corto(BALDUINO 2007).

Open UP es un proceso completo, flexible y corto, fomenta el uso de técnicas agiles y principios, mientras que tiene un ciclo de vida estructurado y probado que hace referencia en la continua entrega de software de calidad.

Todo proyecto en OpenUP consta de cuatro fases: inicio, elaboración, construcción y transición. Cada una de estas fases se divide a su vez en iteraciones, lo cual tiene como ventaja que permite a los integrantes del equipo de desarrollo aportar con micro-incrementos, que pueden ser el resultado del trabajo de unas pocas horas o unos pocos días. Además este ciclo de vida provee a los clientes de una visión del proyecto, transparencia y los medios para que controlen la financiación, el riesgo, el ámbito y el valor de retorno esperado(ECLIPSE 2011). A continuación en la figura 1 se muestran los detalles de estas cuatro fases:

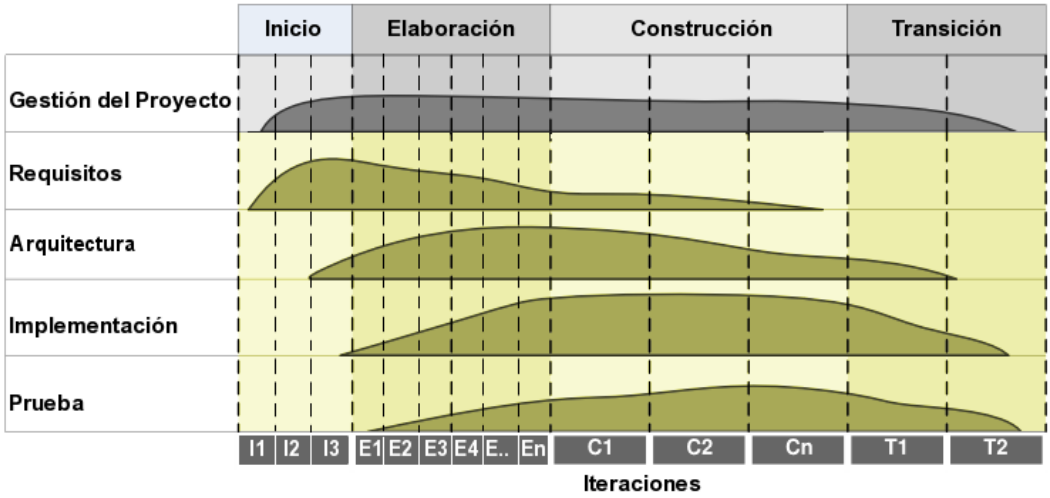


Figura 1 Ciclo de vida de la metodología Open Up(ECLIPSE 2011).

1. **Fase de inicio**: En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los Casos de Uso críticos, una estimación inicial del coste y un boceto de la planificación.
2. **Fase de elaboración**: En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo, unos requisitos y una arquitectura estables, se especifican, además, las herramientas, la infraestructura a utilizar y el entorno de desarrollo.
3. **Fase de construcción**: Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados en esta fase.
4. **Fase de transición**: Esta fase corresponde a la introducción del producto en la comunidad de usuarios. Consta de las sub-fases de pruebas de versiones beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema (BALDUINO 2007).

Para guiar la implementación del sistema se decide emplear la metodología Open Up por ser este un proyecto pequeño donde todo el trabajo es realizado por un programador, el cliente forma parte del equipo de desarrollo, de esta manera se logra una mejor retroalimentación, corrección de errores y se garantiza la entrega de un producto final con la calidad requerida.

1.3.2 Lenguaje Unificado de Modelado: UML 2.0

El Lenguaje Unificado de Modelado fue creado con el objetivo de llenar la brecha existente entre la idea de un proyecto y su implementación, permitiendo modelar uno o varios sistemas de manera que no sean necesarios procesos engorrosos para expresar de manera clara y concisa lo que deberían ser. Basado en tecnología orientada a objetos este es sin duda alguna una herramienta de uso obligatorio al menos en los proyectos de gran alcance, aunque los pequeños proyectos también podrían obtener grandes ventajas de este lenguaje. Es válido decir que involucra todo el ciclo de vida del proyecto y está pensado para varios lenguajes y plataformas (RUMBAUGH *et al.* 2000).

1.3.3 Visual Paradigm 8.0

Las herramientas CASE *(Computer Aided Software Engineering* por sus siglas en inglés), se desarrollan para automatizar procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software. Se enfoca en un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo.

Visual Paradigm soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue y contribuye a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Se caracteriza por el uso de un lenguaje estándar común al equipo de trabajo, que facilita la comunicación entre sus integrantes, es una herramienta fácil de instalar y actualizar, genera código para varios lenguajes de programación y exporta en formato HTML, es una tecnología libre y está disponible en varios idiomas. Uno de los elementos importantes que le aportan distinción es la posibilidad que brinda de soportar aplicaciones web (PRESSMAN, ROGER S. 2002a).

1.3.4 Frameworks

Para lograr un desarrollo acorde a las necesidades existentes en cuanto a tiempo y requisitos del sistema se hizo necesario el análisis de varios *frameworks* o bibliotecas de desarrollo tanto del lado del cliente como del servidor. En términos generales un framework es definido como un conjunto de códigos o

utilidades previamente implementadas siguiendo diversos patrones que permiten a los desarrolladores reutilizar código y como consecuencia directa, ahorrar ya sea tiempo de trabajo o esfuerzo.

Sails JS

Para el lado del servidor se utiliza Sails.js. Es un framework para Node.js. Está realizado bajo el framework Express, incluyendo varias capas de abstracción para hacer un desarrollo más fácil. Sails viene instalado con un potente ORM llamado Waterline, una herramienta de almacén de datos agnóstico que simplifica drásticamente la interacción con una o más bases de datos. Proporciona una capa de abstracción en la parte superior de la base de datos subyacente, lo que le permite consultar fácilmente y manipular sus datos sin necesidad de escribir código de integración específica del proveedor. La última versión de Sails viene preparado para trabajar Postgres, Mongo, MySQL, Redis, y en disco. Entre otras cosas, nos facilita en gran medida el desarrollo de APIs REST, servidores de archivos, seguridad y websockets para manejar y realizar peticiones en tiempo real gracias a Socket.io. Está creado con la filosofía "Convención sobre Configuración"; esto significa que nos permite enfocarnos en el desarrollo de la idea, ahorrándonos mucho tiempo en configuración inicial y es un complemento ideal para frameworks como AngularJS, BackboneJS o ReactJS. El frameworks Sails JS facilita el desarrollo de aplicaciones Node.js empresariales. Ha sido diseñado para imitar el patrón MVC de frameworks como Ruby on Rails, pero con soporte para los requisitos de aplicaciones modernas: data-driven APIs con una arquitectura escalable y service-oriented (DELGADO MAGDALENA 2015).

Angular JS v1.5.4

Angular JS es un framework de código abierto, sobre el lenguaje Javascript con programación del lado del cliente que contiene un conjunto de bibliotecas útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo (BASALO and ÁLVAREZ 2014).

El framework se basa en el patrón Modelo Vista Controlador (MVC), con el objetivo de separar las capas de presentación, lógica y componentes de una aplicación. Se concibió con los siguientes objetivos en mente:

- Desacoplar la manipulación del DOM de la lógica de aplicación
- Desacoplar el lado cliente del lado servidor en una aplicación
- Proveer estructura para el desarrollo de una aplicación

AngularJS se puede combinar con el entorno en tiempo de ejecución Node.js (BLASCO 2016).

1.3.5 Lenguaje de programación: Javascript ES6

En aras de desarrollar las funcionalidades necesarias para la aplicación se ha decidido utilizar el lenguaje de programación Javascript ES6.

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Actualmente es la tecnología más extendida en el enriquecimiento de páginas web del lado del cliente (EGUÍLUZ PÉREZ 2012).

Entre las acciones típicas que se pueden realizar en Javascript se tienen dos vertientes. Por un lado los efectos especiales sobre páginas web, y por el otro, Javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo (FLANAGAN 2002a; 2002b).

Este lenguaje será utilizado en la solución propuesta principalmente en el proceso de validación de datos entrados por los usuarios al sistema, ya que permite ejecutar instrucciones como respuesta a las acciones del usuario.

1.3.7 Entorno de ejecución: Node.JS v7

Node.js es un entorno de ejecución para Javascript construido con el motor de Javascript V8 de *Chrome*. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. Concebido como un entorno de ejecución de Javascript orientado a eventos asíncronos, Node.js está diseñado para construir aplicaciones en red escalables, perfecto para aplicaciones en tiempo real con gran cantidad de datos que se ejecutan en dispositivos distribuidos. El ecosistema de paquetes de Node.js, es el ecosistema más grande de librerías de código abierto en el mundo (NODE.JS 2017).

1.3.8 Entorno de desarrollo: NetBeans IDE 8.0

Un Entorno de Desarrollo Integrado (del inglés *Integrated Development Environment*, IDE) es un programa compuesto por un conjunto de herramientas para un programador. NetBeans IDE es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Es de código abierto, escrito completamente en Java, pero puede utilizarse para desarrollar en cualquier otro lenguaje de programación. Soporta el desarrollo de Aplicaciones empresariales con Java EE, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies, y modelado UML (NETBEANSIDE 2017).

NetBeans es “una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso” (PARRA 2011).

NetBeans tiene gran éxito con una gran base de usuarios y una comunidad en constante crecimiento. Provee una estructura para los proyectos, propone un esqueleto para organizar código fuente, el editor conjuntamente integra los lenguajes como PHP, HTML, Javascript y CSS. Cabe señalar que el referido entorno integrado de desarrollo brinda soporte a Symfony2, obteniendo un grupo de ventajas importantes para el desarrollador frente al framework (NETBEANSIDE 2017).

1.3.9 Sistema gestor de Base de Datos: PostgreSQL v9.3

Sus características técnicas la hacen una de las bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, *estabilidad, potencia, robustez, facilidad de administración e implementación de estándares* han sido las características que más se han tenido en cuenta durante su desarrollo. *PostgreSQL* funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (GROUP 2017).

Mediante un sistema denominado MVCC, *PostgreSQL* permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit.* En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que *PostgreSQL* nunca ha presentado caídas en varios años de operación de alta actividad (MARTINEZ GUERRERO 2010).

1.3.10 PgAdmin III v1.18

Es una aplicación gráfica para gestionar el gestor de bases de datos *PostgreSQL*, siendo la más completa y popular con licencia *Open Source*. Está escrita en C++ usando la librería gráfica multiplataforma *wxWidgets*, lo que permite que se pueda usan en Linux, *FreeBSD*, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la *PostgreSQL* 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de *PostgreSQL* como *Pervasive Postgres, EnterpriseDB, Mammoth Replicator* y *SRA PowerGres* (PGADMIN 2017).

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de *PostgreSQL* y facilita enormemente la administración. La aplicación también incluye un

editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación *Slony-I* y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL para mayor seguridad (PGADMIN 2017).

1.4 Conclusiones Parciales

Después de realizar el estudio de la bibliografía relacionada con los sistemas para la gestión de incidencias, se pudo identificar que las soluciones existentes no satisfacen las necesidades que afronta el Grupo Azucarero AZCUBA. Por esta razón se propone la implementación de un sistema que sea capaz de ajustarse a estas necesidades y que se nutra de las principales ventajas que muestran algunas de las soluciones analizadas.

Las herramientas seleccionadas impulsan la soberanía tecnológica propuesta por el país. Se decidió la utilización de la metodología de desarrollo OpenUp, como lenguaje de modelado UML 2.0, como herramienta CASE Visual Paradigm en su versión 8.0, como *frameworks* de desarrollo Sail Js y Angular Js v 1.5.4, utilizando el lenguaje de programación Javascript ES6, como entorno de ejecución Node JS V7 y, como entorno de desarrollo NetBeans en su versión 8.0, PostgreSQL 9.3 como SGBD, así como la herramienta PgAdmin III 1.18 para su administración.

Capítulo 2: Análisis y diseño del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA.

En este capítulo se realiza el modelo de negocio, el cual constituye el punto de partida para el desarrollo de la propuesta de solución. Se definen los requisitos funcionales y no funcionales. Se desarrolla el modelo de casos de uso, donde se exponen también los diferentes patrones utilizados. Se especifican los elementos fundamentales de la arquitectura y el diseño, los patrones de diseño empleados y el modelo de datos del sistema. Al concluir este capítulo se contará con una comprensión de los problemas actuales del negocio y una idea pertinente del sistema a desarrollar.

2.1 Propuesta de solución

Se propone realizar un sistema informático que permita gestionar el proceso de incidencias que se lleva a cabo en el Grupo Azucarero AZCUBA. Este sistema permitirá tener mejor control de las incidencias en el Grupo Azucarero AZCUBA, mediante el cual los especialistas de cada área de trabajo involucrado podrán realizar análisis de datos específicos y necesarios por nivel organizativo lo que incidirá directamente en un mejor proceso de control y organización.

2.2 Modelo de Negocio

En todo proceso de informatización es necesario antes de desarrollar una solución para un escenario determinado, comprender de manera natural cómo funciona este y cuáles son sus principales características. Precisamente este es el objetivo principal de un modelo de negocio. Según Ivar Jacobson un modelo de negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente. Al igual que el modelo de casos de uso para un sistema software, el modelo de casos de uso del negocio presenta un sistema desde la perspectiva de su uso y esquematiza cómo proporciona valor a sus usuarios (JACOBSON, IVARBOOCH *et al.* 2000b).

2.2.1 Actores y trabajadores del negocio

Una vez identificados los procesos del negocio es posible determinar los actores y trabajadores involucrados en su realización. Los actores constituyen elementos externos que interactúan con los procesos del negocio, mientras que los trabajadores pueden ser abstracciones de personas, grupo de personas o un sistema automatizado que realiza una o varias operaciones en el negocio (ABRAN *et al.*

2004). A continuación, se muestran los actores y trabajadores que participan en el proceso de recepción y seguimiento de las incidencias en el Grupo azucarero AZCUBA:

Tabla 3 Actores y trabajadores del negocio.

Actores del Negocio	**Justificación**
Analista UEB	Es el encargado de crear la nueva incidencia e informarla.
Trabajadores del Negocio	**Justificación**
Jefe de sala de análisis	Es el encargado de revisar que todos los elementos de la planilla de la incidencia reportada estén bien para su posterior análisis.
Especialista superior	Es el encargado de analizar la relevancia de la incidencia, darle seguimiento y posible solución.
Jefe de la comisión de hechos extraordinarios	Es el encargado de analizar las incidencias de mayor relevancia y decidir su posible solución.

2.2.2 Diagrama de Casos de Uso del Negocio (CUN)

Un diagrama de casos de uso del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio (GONZÁLEZ, ANAISA HERNÁNDEZ 2004). A continuación, se muestra el diagrama de CUN correspondiente a la propuesta de solución:

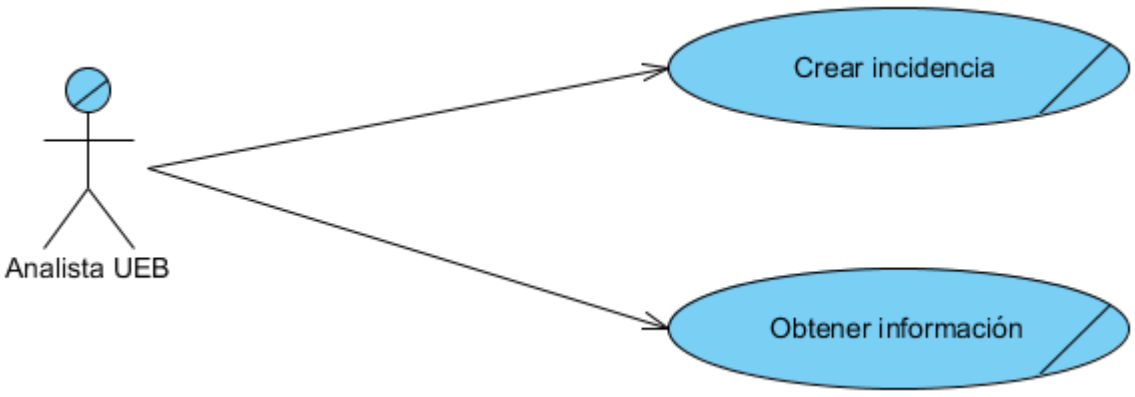


Figura 2 Diagrama de Caso de uso del negocio.

2.2.3 Descripción textual del CUN "Crear incidencia"

Tabla 4 Descripción textual del Caso de uso "Crear incidencia".

Nombre del caso de uso del negocio:	Crear incidencia
Actores del negocio:	Analista UEB
Propósito:	Crear las nuevas incidencias y darle seguimiento.
Resumen: El caso de uso inicia cuando el analista agroindustrial de una entidad presenta ante el Jefe de la sala de análisis una nueva incidencia, este la recepta y se dispone a darle seguimiento, en caso de que la incidencia cuente con la documentación adecuada y los datos necesarios la envía al especialista superior para su seguimiento, en caso contrario notifica al autor las deficiencias, las cuales deben ser corregidas por el autor quien nuevamente presentará dicha incidencia. Si finalmente la incidencia es enviada con la solución y medidas tomadas se informa de dicho resultado a la comisión de hechos extraordinarios, en caso contrario se reciben las nuevas recomendaciones por parte del especialista superior que son devueltas al autor.	
Casos de Uso asociados:	
Flujo normal de los eventos	
Acción del actor:	**Respuesta del negocio:**
1 El analista UEB de una entidad presenta la incidencia junto a la documentación necesaria.	2 El jefe de la sala de análisis de la entidad analiza la incidencia y su documentación asociada. 2.1 El jefe de la sala de análisis chequea que todos los elementos de la incidencia estén correctos. 3 El jefe de la sala de análisis envía la incidencia a la especialista superior para que sea registrada. 3.1 El especialista superior revisa la información e informa a la comisión.
5 El analista UEB recibe la notificación de que la incidencia ha sido recepcionada finalizando así el Caso de Uso.	4 El especialista superior informa al analista de la entidad el estado de la incidencia.
Prioridad:	Alta
Mejoras:	El registro de las incidencias presentadas permitirá la obtención de estadísticas beneficiando la toma de decisiones. La organización de la documentación asociada a cada incidencia evitará pérdidas y manejo de documentos intercambiados o duplicados. Facilitará la comunicación entre autores y el especialista superior.
Cursos alternos:	

2.1 El especialista superior devuelve el informe por contener elementos incorrectos o incompletos en cuyo caso el autor corrige los señalamientos y presenta nuevamente el informe de la incidencia.	
Casos de Uso Incluidos:	

2.2.4 Diagrama de actividades del CUN

Un diagrama de actividades describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio. Para la mejor comprensión del caso de uso anterior se propone el siguiente diagrama de actividades donde las actividades objeto de automatización serán las señaladas en amarillo (LARMAN 1999).

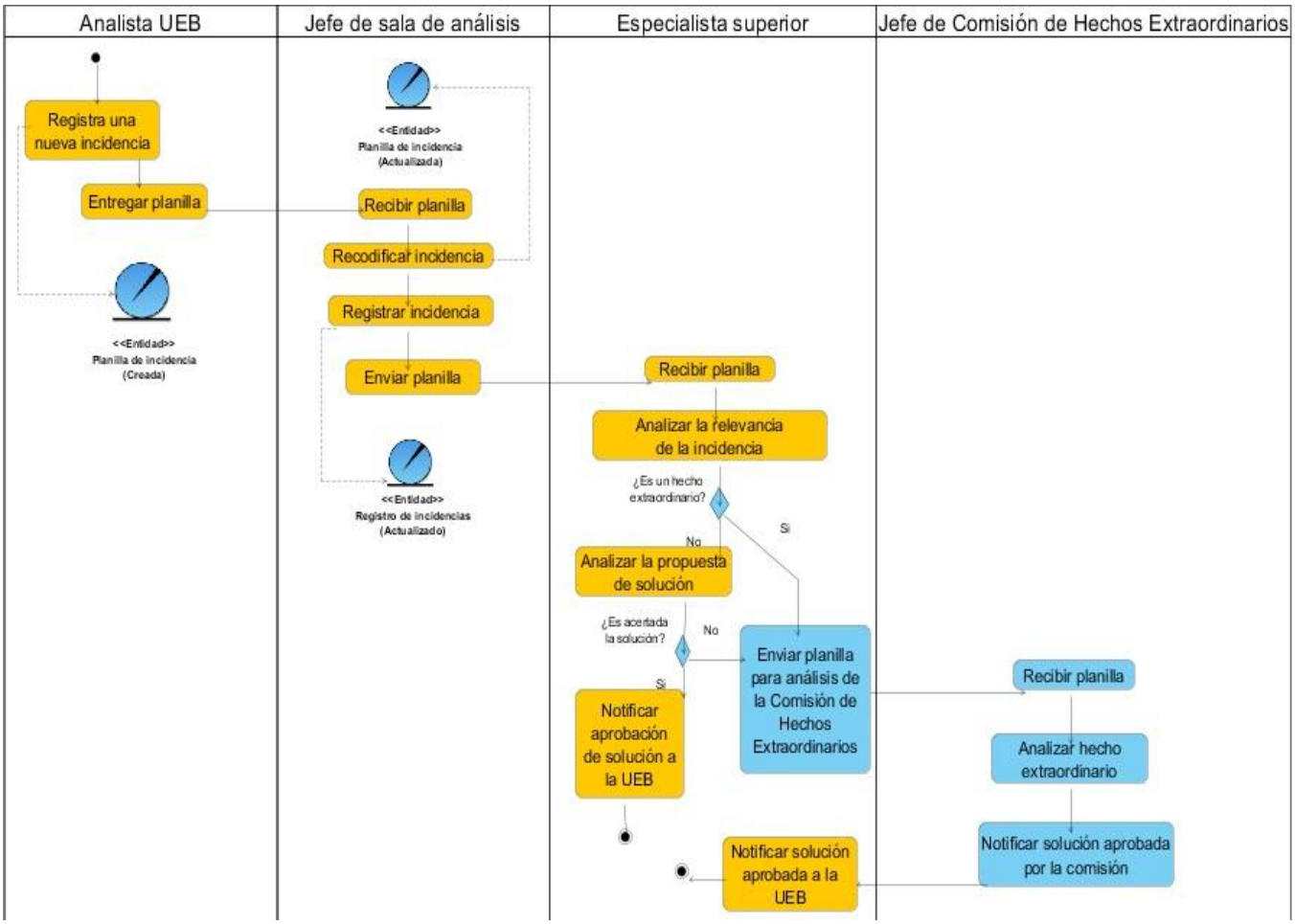


Figura 3 Diagrama de actividades del Caso de uso del negocio.

2.3 Levantamiento de requisitos

Según Sommerville, durante el levantamiento u obtención de requisitos, los desarrolladores del software trabajan directamente con los clientes y los usuarios finales del sistema, para determinar el dominio de la aplicación, los servicios que debe proporcionar el sistema, el rendimiento requerido, las restricciones de hardware, entre otros elementos, que de conjunto garantizarán la entrega de un producto final que satisfaga las necesidades planteadas por el cliente (SOMMERVILLE 2005).

Por su parte Pressman plantea que los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar (PRESSMAN, ROGER S. 2005). En dependencia de sus características, los requisitos de software se dividen en funcionales y no funcionales, lo cual se detalla a continuación.

2.3.1 Requisitos funcionales

Los requisitos funcionales(RF) son declaraciones de los servicios que debe proporcionar un sistema de manera en que este reaccione a determinadas entradas y su comportamiento sea el esperado (SOMMERVILLE 2005). A continuación, se detallan los requisitos funcionales que debe cumplir la propuesta de solución planteada.

RF1. Autenticar usuario.
RF2. Crear usuario.
RF 3. Actualizar usuario.
RF 4. Eliminar usuario.
RF 5. Buscar usuario.
RF 6. Listar usuarios.
RF 7. Crear incidencia.
RF 8. Modificar incidencia
RF 9. Buscar incidencia.
RF 10. Listar incidencias.
RF 10.1 Mostrar incidencias nuevas
RF 10.2 Mostrar incidencias resueltas
RF 10.3 Mostrar incidencias pendientes
RF 11. Crear entidad.
RF 12. Eliminar entidad.
RF 13. Modificar entidad.
RF 14. Mostrar entidad
RF 15. Crear persona.
RF 16. Agrupar persona.
RF 17. Eliminar persona.
RF 18. Modificar persona.
RF 19. Mostrar persona.
RF 20.Obtener reporte.
RF 21. Crear reporte.
RF 22. Mostrar reporte.
RF 23. Modificar reporte.
RF 24. Eliminar reporte.
RF 25. Exportar reporte a formato PDF
RF 26. Exportar reporte a Microsoft Excel

2.3.2 Requisitos no funcionales

Los requerimientos no funcionales (RNF) son aquellos que no se refieren directamente a las funciones específicas que proporciona la herramienta, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y capacidad de almacenamiento (SOMMERVILLE 2005). Dentro de los requisitos no funcionales que debe cumplir la propuesta de solución se encuentran los siguientes:

Requisitos de usabilidad

RNF 1: Aunque el sistema está orientado a usuarios con un dominio en el manejo de computadoras el mismo se caracterizará por un diseño sencillo e intuitivo.

Requisitos de fiabilidad:

RNF 2: La información manejada por el sistema estará protegida de acceso no autorizado y divulgación.

RNF 3: El sistema se encargará de controlar los diferentes accesos y privilegios de los usuarios, además de identificar al usuario antes de que pueda realizar cualquier acción sobre el sistema.

RNF 4: Cada tres meses se realizarán por parte del administrador del sistema copias de seguridad a la Base de Datos para garantizar la persistencia de la información ante posibles fallos.

Requisitos de eficiencia:

RNF 5: El sistema soportará la conexión simultánea de un promedio de 50 y un máximo de 100 usuarios concurrentes.

RNF 6: El tiempo promedio de respuesta del sistema no excederá los 5 segundos para el mínimo de usuarios concurrentes y para el máximo los 15 segundos.

Requisitos de hardware

Tabla 5 Interfaces de Hardware. Características.

RNF 7 Para las computadoras cliente:	RNF 8 Para los servidores:
• 1 GB de RAM como mínimo. • Al menos 40 GB de disco duro. • Procesador 800MHz Intel Pentium III o equivalente como mínimo.	• 2 GB de RAM como mínimo. • Al menos 80 GB de disco duro. • Procesador Dual Core como mínimo.

Requisitos de software

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, el servidor del usuario final debe tener como requerimientos mínimos de software:

Tabla 6 Interfaces de Software. Características.

RNF 9 Para las computadoras cliente:	RNF 10 Para los servidores:
• Navegador como Mozilla Firefox 16.0 o superior, que cumpla con los estándares W3C[1]. • Sistema operativo: GNU/Linux (la distribución es irrelevante), Windows XP o superior o Mac OS.	• Sistema operativo GNU/Linux Ubuntu Server 11.04 o superior. • PostgreSQL 9.3 o superior como Sistema Gestor de Bases de Datos.

Portabilidad

RNF 11: La herramienta desarrollada deberá ser multiplataforma teniendo un buen funcionamiento tanto en Linux como en Windows.

2.4 Modelo de casos de uso del sistema

Jacobson plantea que el modelo de casos de uso del sistema está conformado por los actores y casos de uso que interactúan en el sistema. Este modelo describe las funcionalidades que tendrá el sistema, lo cual posibilita al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizarlo (RUMBAUGH *et al.* 2000). Por su parte Larman agrega que este modelo proporciona una explicación clara y consistente de lo que debería hacer el software, de modo que el modelo se use a lo largo del proceso de desarrollo (LARMAN 1999).

2.4.1 Casos de Uso del Sistema (CUS)

Los Casos de Uso son descripciones funcionales del sistema que permiten definir los límites del software y las relaciones entre la aplicación y el entorno; describen cómo los actores pueden usar un sistema. Especifican una secuencia de acciones que debe devolver algún resultado de valor a un actor (JACOBSON, IVARBOOCH *et al.* 2000b).

[1] El W3C es una comunidad internacional que incluye un personal de tiempo completo, expertos de la industria, y varias organizaciones miembros. Estos grupos trabajan conjuntamente para desarrollar estándares para la World Wide Web.

2.4.2 Actores del sistema

Los actores constituyen cada uno de los roles que agrupan a los diversos tipos de usuarios que interactúan con los diferentes CUS. A partir del análisis de las características que debe tener la solución propuesta se definen para la misma los siguientes actores:

Tabla 7 Actores del sistema.

Actor	Descripción
Usuario	Es el usuario básico del sistema, con acceso a la autenticación y con un mínimo nivel de privilegios.
Analista	Es el usuario del sistema, con permisos para gestionar las incidencias y los reportes relacionados con las incidencias.
Especialista superior	Es el usuario encargado de listar las incidencias para darle seguimiento en el sistema y además puede obtener reportes del comportamiento de las mismas, también puede exportar a los formatos Excel y PDF.
Administrador del sistema	Es el súper usuario del sistema, con permisos para agregar, eliminar y asignar privilegios a los usuarios.

2.4.3 Diagrama de CUS

Los diagramas de CUS describen parte del modelo de casos de uso y muestran un conjunto de casos de uso y actores con una asociación entre cada par de estos que interactúan en el sistema (JACOBSON, IVARBOOCH *et al.* 2000b). A esto se puede agregar que constituye una excelente representación del contexto del sistema, ya que sirve como herramienta de comunicación para visualizar, especificar, resumir y documentar el comportamiento de este y sus actores.

A continuación, se presenta el diagrama de CUS correspondiente a la solución propuesta.

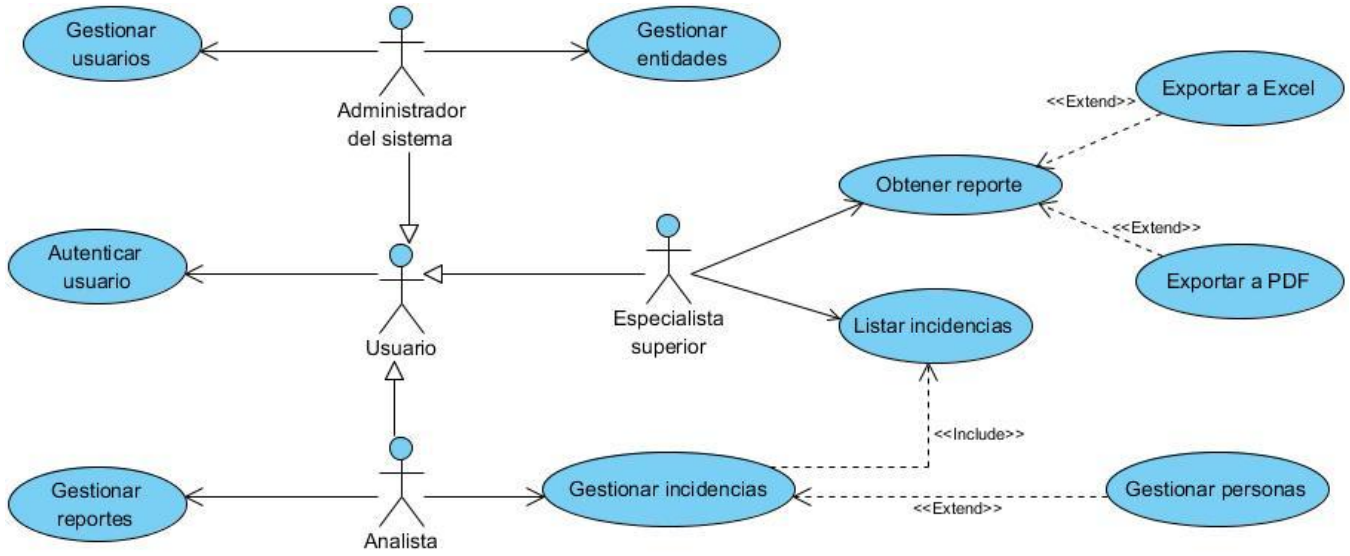


Figura 4 Diagrama de Casos de uso del sistema.

- **Descripción textual del CUS Gestionar incidencias**

La descripción textual o especificación de los CU tiene como objetivo describir en detalle el flujo normal de eventos, así como los flujos alternos, de una manera precisa y fácilmente comprensible. Además, se incluyen las precondiciones y poscondiciones y se define cómo comienza, termina e interactúa cada caso de uso con sus actores.

A continuación, se presenta la descripción textual del CUS Gestionar incidencias.

Tabla 8 Caso de uso del sistema "Gestionar incidencias".

Caso de Uso:	Gestionar incidencias.
Actores:	Analista
Propósito:	Este caso de uso se realiza con el objetivo de realizar el registro de incidencias por parte de los autores además de posibilitar su posterior edición.
Resumen:	Este caso de uso se inicia cuando el analista selecciona la opción de crear una nueva incidencia y culmina con la realización de una de las siguientes operaciones sobre una incidencia: *Crear*, *Actualizar*, *Mostrar o Eliminar*.
Precondiciones:	El actor debe estar autenticado en el sistema y debe tener permisos para realizar las operaciones. Debe seleccionarse la incidencia sobre la cual se pretende realizar una de las siguientes acciones: *Actualizar*, *Mostrar o Eliminar.* Para realizar una de estas acciones debe existir en el sistema al menos una incidencia activa.
Referencias:	***RF 7***, ***RF 8***, ***RF 9***, ***RF 10***.

Prioridad:	Crítico.
Flujo Normal de Eventos	
Acción del Actor:	**Respuesta del Sistema:**
1 El caso de uso inicia cuando el analista selecciona o despliega la opción gestionar incidencia y selecciona la acción a realizar.	**2** En caso de seleccionar: ✓ "crear incidencia" ver sección *crear incidencia*. ✓ "mostrar" ver sección *mostrar incidencia.* ✓ "actualizar" ver sección *actualizar incidencia*. ✓ "eliminar" ver sección *eliminar incidencia*.
Sección Crear incidencia	
Flujos Básicos	
Acción del Actor:	**Respuesta del Sistema:**
	1 El sistema muestra una interfaz para introducir los datos referentes al autor de la incidencia y la incidencia en sí misma.
2 El analista introduce los datos necesarios y selecciona el botón *crear*.	**3** El sistema verifica que no existan campos vacíos. **3.1** El sistema verifica que los datos introducidos sean válidos.
	4 El sistema crea una nueva incidencia, muestra el mensaje: "Incidencia creada con éxito" dando paso a la ejecución del caso de uso (Incluido)
Flujos Alternos	
Acción del Actor:	**Respuesta del Sistema:**
	3 El sistema muestra un mensaje informando que existen campos vacíos.
	3.1 El sistema muestra un mensaje informando que existe algún campo inválido.
Sección Mostrar incidencia	
Flujos Básicos	
Acción del Actor:	**Respuesta del Sistema:**
	1 El sistema muestra la interfaz que contiene todos los detalles asociados a una incidencia. La posibilidad de descargar los archivos que esta posea y en el caso de tener recomendaciones podrán ser visualizadas.
Sección Eliminar incidencia	

Flujos Básicos	
Acción del Actor:	**Respuesta del Sistema:**
	1 El sistema elimina la incidencia y muestra el mensaje: "Incidencia eliminada con éxito".
Sección Actualizar incidencia	
Flujos Básicos	
Acción del Actor:	**Respuesta del Sistema:**
	1 El sistema muestra una interfaz para actualizar los datos de la incidencia seleccionada.
2 El analista actualiza los datos necesarios y selecciona el botón *actualizar*.	**3** El sistema verifica que no existan campos vacíos. **3.1** El sistema verifica que los datos introducidos sean válidos.
	4 El sistema modifica la Incidencia y muestra el mensaje: "Incidencia actualizada con éxito".
Flujos Alternos	
Acción del Actor:	**Respuesta del Sistema:**
	3 El sistema muestra un mensaje informando que existen campos vacíos.
	3.1 El sistema muestra un mensaje informando que existe algún campo inválido.
Poscondiciones:	Queda creada, modificada o eliminada una incidencia previamente seleccionada por el usuario.
Casos de Uso Incluidos:	Listar incidencias

2.4.4 Patrones de casos de uso utilizados

- **CRUD Completo**

Este patrón consiste en un caso de uso para administrar información, nos permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja. El mismo se manifiesta en los casos de uso "Gestionar incidencias", "Gestionar personas" y "Gestionar usuario".

- **Extensión Concreta o Inclusión**

Este patrón se divide en concreta extensión o concreta inclusión. Poniéndose en evidencia el empleo de la concreta inclusión al establecerse una relación de inclusión por particionamiento entre el caso de

uso del sistema “Gestionar incidencias” y el caso de uso del sistema “Listar incidencias”. También se evidencia la concreta extensión entre el caso de uso “Obtener reporte” y los casos de uso “Exportar a Excel” y “Exportar a PDF”.

- **Múltiples actores**

Varios actores pueden jugar el mismo rol en un caso de uso particular, como se pone de manifiesto en el diagrama de casos de uso del sistema con la generalización/especialización establecida entre los actores del sistema “Administrador del sistema”, “Especialista superior” y “Analista” al asumir el rol del actor “Usuario”.

2.5 Elementos fundamentales del diseño y arquitectura del sistema.

El diseño del sistema tiene como propósitos fundamentales adquirir una profunda comprensión de los aspectos relacionados con los requisitos funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos y tecnologías utilizadas. Permite crear un punto de partida para la actividades de la implementación, descomponiéndolas en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo (JACOBSON, IVAR *et al.* 2000a).

2.5.1 Estilo Arquitectónico

Buschmann, Henney y Schimdt definen estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural que expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y las reglas para su construcción (BUSCHMANN *et al.* 2007). Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software.

En el caso de la solución propuesta se determina el empleo del estilo Llamada y Retorno ya que permite obtener una estructura de programa fácil de modificar persiguiendo la escalabilidad del sistema. Emplear este estilo posibilitará además la comunicación, la coordinación y cooperación entre los componentes y las restricciones que definen como se integran para conformar el sistema, además de los modelos semánticos que facilitan al diseñador el entendimiento de todas las partes del sistema, evitando que las variaciones realizadas a funcionalidades o componentes específicos afecten el funcionamiento general (PRESSMAN, ROGER S. 2005).

2.5.2 Arquitectura y Patrones de Diseño

Según Sommerville, el diseño arquitectónico o arquitectura del software es la primera etapa en el proceso de diseño y representa un enlace crítico entre los procesos de ingeniería de diseño y de requerimientos.

Está relacionado con el establecimiento de un marco estructural básico que identifica los principales componentes de un sistema y las comunicaciones entre estos componentes (SOMMERVILLE 2005).

Patrón arquitectónico Modelo Vista Controlador (MVC)

Un patrón arquitectónico brinda la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución (BUSCHMANN *et al.* 2007).

Para el desarrollo de la propuesta de solución se define el patrón arquitectónico MVC, el cual se basa en las ideas de reutilización de código y la separación de conceptos, buscando facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. Su principal característica radica en que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, el modelo, la vista y el controlador, los cuales se detallan a continuación:

El modelo: es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. En la solución propuesta esta capa agrupa los paquetes “Entity” y “Model”. El primero incluye dos paquetes: “Entity”, que recoge las entidades del sistema que se crean a partir de las tablas de la base de datos, y “Repository” que contiene entidades que se encargan del manejo y selección de datos. Por su parte, en el paquete “Model” se incluyen las clases del dominio y la lógica de datos.

Las vistas: se encargan de presentar la información del sistema al usuario y generar los eventos de la interacción con éste. Capturan eventos del usuario y se los envía al sistema a través del controlador. Posteriormente reciben una respuesta del controlador y muestra información al usuario. En la propuesta de solución esta capa contiene los paquetes “View”, “Angular” y “CSS”. El primero recoge las páginas y plantillas que conforman al sistema. El paquete “Angular” contiene los archivos “Javascript” y el “CSS” las hojas de estilo.

El controlador: contiene el código necesario para responder a las acciones que se solicitan en la aplicación, sirve de enlace entre las vistas y el modelo, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. En la solución propuesta esta capa contiene los controladores del sistema, quienes se encargan de crear y devolver una respuesta, una vez que han recibido una petición. Para esto se utilizan clases que son destinadas a la lógica de control y otras correspondientes a los formularios, las cuales se agrupa el paquete “Controller”.

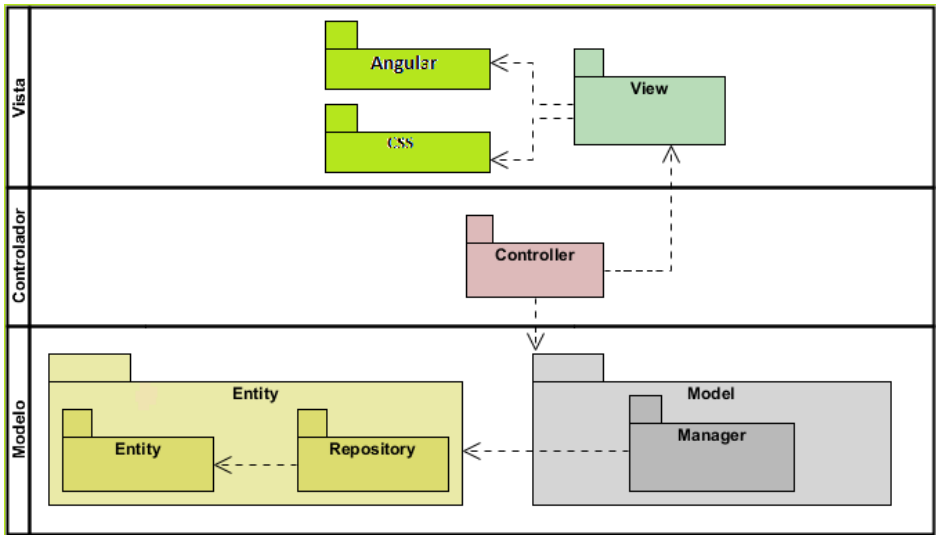


Figura 5 Vista global de paquetes de diseño del Caso de Uso: Gestionar Incidencia.

Patrones de diseño

Los patrones de diseño constituyen una descripción de un problema y la solución, a la que se da un nombre y que se puede aplicar a nuevos contextos; idealmente, proporcionan consejos sobre el modo de aplicarlos en varias circunstancias, y consideran los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (LARMAN 2003).

Patrones de Principios Generales para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. A continuación se explican los patrones de este tipo que se utilizan en la solución propuesta.

- **Experto:** propone asignar las responsabilidades a las clases de acuerdo a la información que contienen las mismas cumpliendo así un principio básico e intuitivo de la programación orientada a objetos. Este patrón se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos (LARMAN 2003). En la solución propuesta este patrón se pone de manifiesto ya que Sails utiliza el objeto relacional de mapeo Waterline, para la capa del modelo. Waterline se encarga de crear una clase experta por cada tabla de la base de datos del modelo, como se puede apreciar en la figura en la clase

"Solicitudes", esto permite que se pueda manejar su información como un objeto de tipo la entidad mapeada.

- **Controlador:** sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, manejando los eventos de entrada de dicha interfaz. Es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema, sino que define el método para la operación del mismo (LARMAN 2003). Este patrón está presente en la propuesta de solución pues se crearon varios controladores que se encargan de manejar eventos entre la vista y el modelo, como por ejemplo en la figura el controlador "SolicitudController".
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento (LARMAN 2003). Este patrón está presente en la propuesta de solución ya que existen algunas clases en la capa del modelo que se encargan de crear instancias de las clases que proveen la información necesaria para su propio manejo, como por ejemplo la clase "SolicitudController" en la figura.
- **Bajo acoplamiento:** propone la asignación de responsabilidades de manera tal que la dependencia entre una clase y otra sea la menor posible, de tal forma que se potencie la reutilización y se mitiguen los efectos que puedan producir en una, la realización de cambios en la otra. Para esto soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio (LARMAN 2003). Se pone de manifiesto en la solución propuesta pues entre las clases del controlador y la vista o el controlador y el modelo, existe una baja interdependencia, lo que se traduce en la posibilidad de efectuar cambios en estas sin que ocurran grandes afectaciones al resto del sistema.
- **Alta cohesión:** se basa en que los elementos de un componente colaboran para producir algún comportamiento bien definido, como una clase que tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas (LARMAN 2003). Se manifiesta el uso de este patrón en la propuesta de solución pues, por ejemplo, como se puede apreciar en la figura, las clases "Solicitud" y "SolicitudController" muestran responsabilidades relacionadas coherentemente, que se complementan entre sí, lo cual garantiza que exista además un bajo acoplamiento que favorece el equilibrio y un diseño en el cual los objetos sean capaces de interactuar.

2.5.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema (JACOBSON, IVAR *et al.* 2000a).

2.5.4 Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación y permiten modelar la vista de diseño del sistema (JACOBSON, IVARBOOCH *et al.* 2000b). A continuación, se presenta el diagrama de clases del diseño correspondiente al Caso de Uso: Gestionar Incidencia. El mismo contiene las clases fundamentales implementadas.

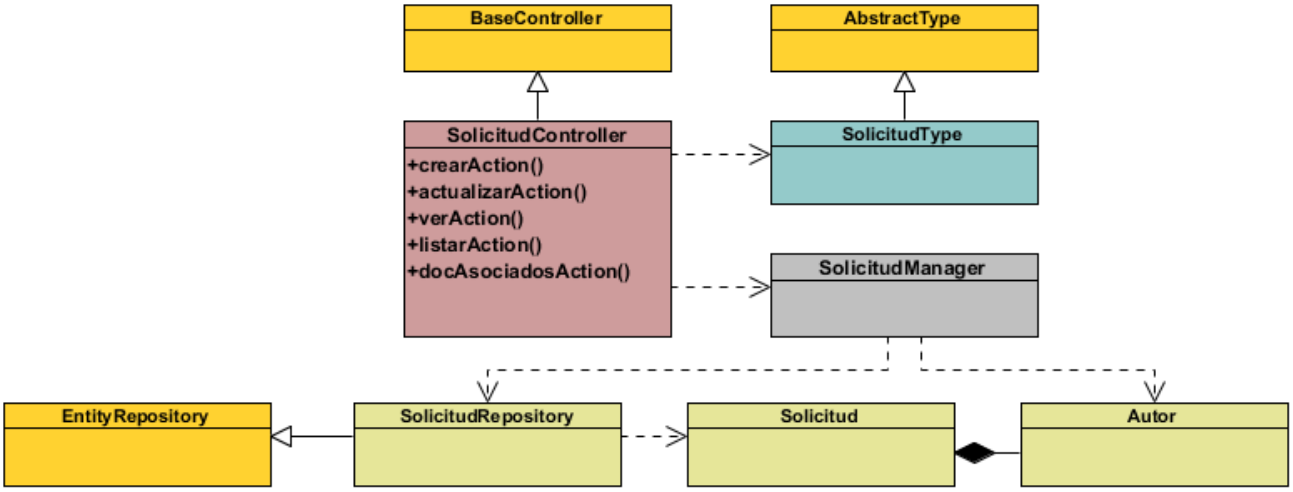


Figura 6 Diagrama de clases del diseño. Caso de Uso: Gestionar Solicitud.

2.5.5 Modelo de datos

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos (MARQUÉS 2011).

Un modelo de datos es un lenguaje orientado a hablar de una base de datos. Típicamente un modelo de datos permite describir:

- Las estructuras de datos de la base: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada.
- Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base (RUMBAUGH *et al.* 2000).

A continuación se presenta el modelo de datos de la presente investigación.

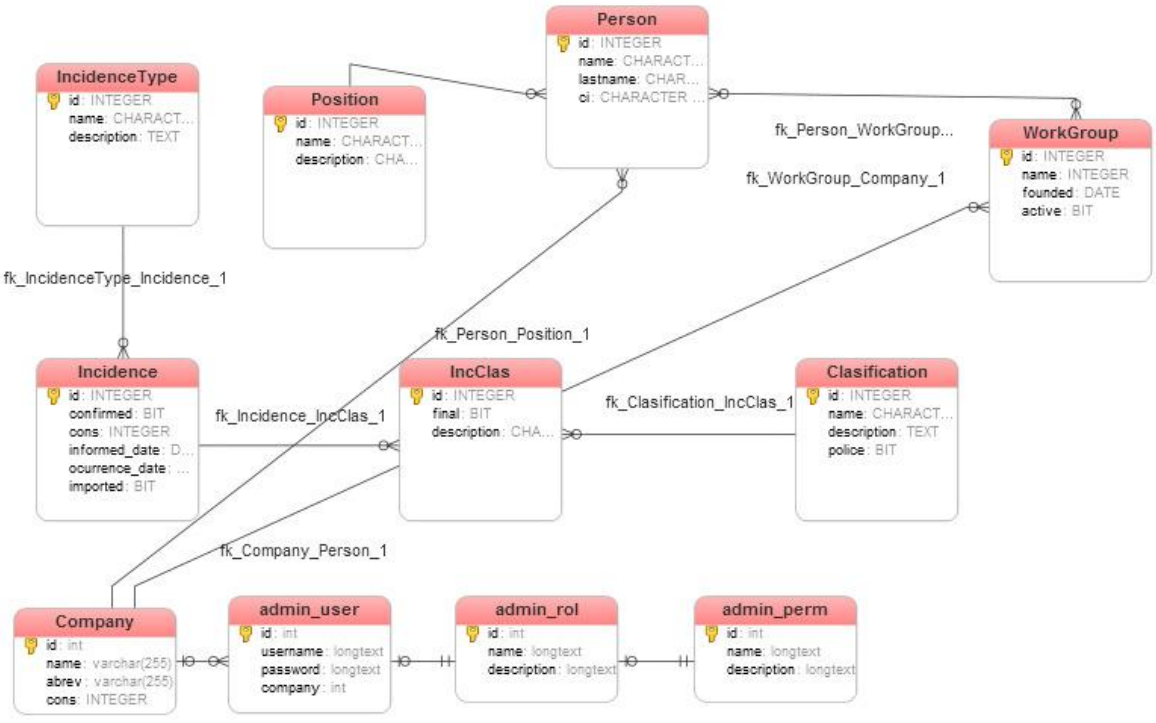


Figura 7 Modelo de datos.

2.6 Conclusiones parciales

Como resultado del presente capítulo se obtuvo el modelo del negocio, 26 requisitos funcionales y 11 no funcionales, el modelo del sistema con 4 actores y 10 casos de uso, que propiciaron el entendimiento entre el equipo de desarrollo y el cliente de la aplicación en función de lo que el sistema debe realizar y las características que debe poseer. Además se obtuvieron los artefactos correspondientes al modelo de diseño y los patrones utilizados, los cuales sirven como material de referencia para futuras modificaciones del sistema y como guía certera para comenzar su implementación.

Capítulo 3: Implementación y pruebas del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA

En el presente capítulo se abordan los principales aspectos referentes a la fase de implementación y pruebas. Se describe la distribución física del sistema propuesto a través del modelo de despliegue y se representa el sistema dividido en componentes y dependencias entre estos mediante el diagrama de componentes. Se especifican las pruebas realizadas a la solución y sus resultados, con el objetivo de determinar no conformidades, erradicarlas y probar que el sistema satisface sus requisitos funcionales y no funcionales.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes, como ficheros de código fuente, ejecutables, entre otros. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. El modelo brinda una mejor visualización del camino a seguir por los desarrolladores del sistema en el momento de enfrentar la implementación del mismo (ABRAN *et al.* 2004).

3.1.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos y sus relaciones en el entorno. Los componentes representan los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, y las relaciones de dependencia se utilizan para indicar que un componente utiliza los servicios ofrecidos por otro componente. Además agrupan varias partes de un sistema modular, desplegables y reemplazables, que encapsulan la implementación y expone un conjunto de interfaces, como por ejemplo, código fuente, binario o ejecutable (LARMAN 2003).

Un componente de software es una parte física de un sistema, pudiendo considerarse la personificación en software de una clase. La representación de dependencias entre componentes de software puede considerarse como representación tácita del modelo de implementación, en ella se pueden incluir componentes de código fuente, componentes del código binario, y componentes ejecutables (JACOBSON, IVAR *et al.* 2007). A continuación se presenta una vista de paquetes, los cuales constituyen la agrupación de una serie de componentes del sistema correspondientes al Caso de Uso: Gestionar incidencias, dicha vista recoje las principales relaciones de dependencia entre los mencionados paquetes.

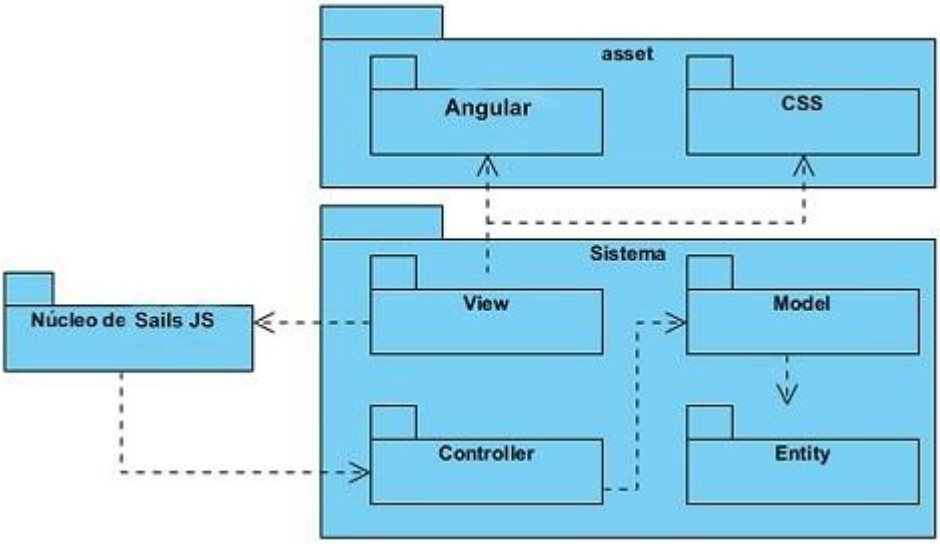


Figura 8 Vista de paquetes. Caso de Uso: Gestionar incidencias. Componentes.

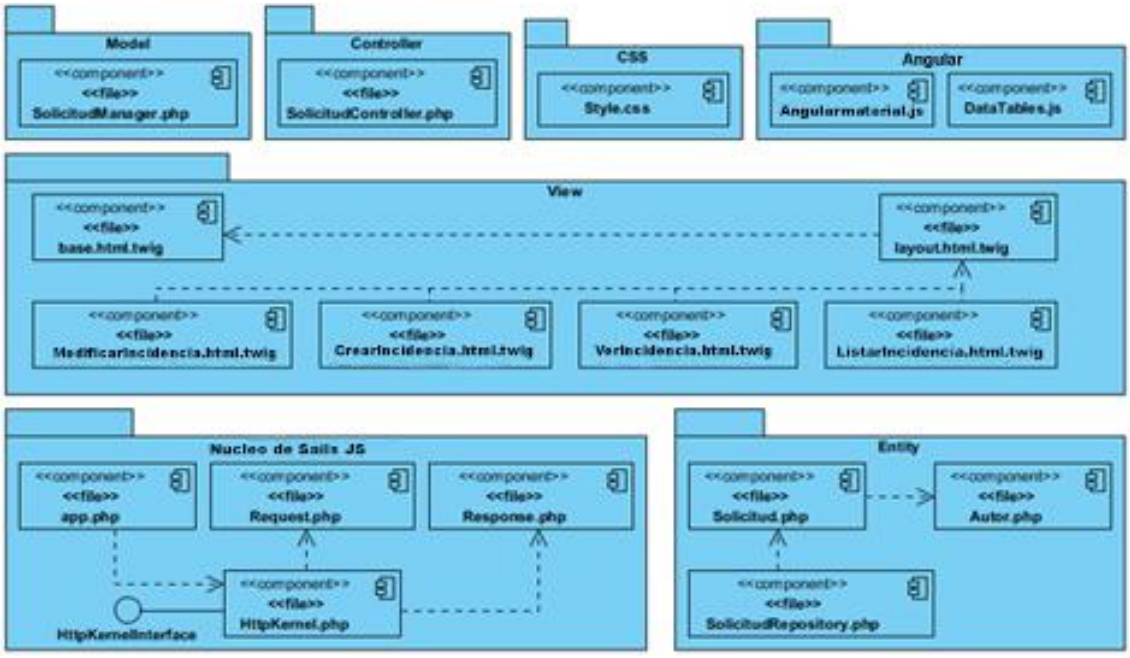


Figura 9 Distribución de componentes a partir de la vista de paquetes. Caso de Uso: Gestionar Incidencia.

La Figura. 9 ilustra la distribución de los principales componentes de software a partir de la distribución de paquetes propuesta en la Figura. 8, en la misma se recogen en esencia componentes de código fuente. El paquete denominado “Núcleo de Sails” representa solo los principales elementos propios del framework de desarrollo, que contribuyen al funcionamiento del sistema.

3.2 Modelo de despliegue

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos que son representados (PRESSMAN, ROGER S 2002b). En la Fig. se presenta el diagrama de despliegue propuesto para el sistema y en la Tabla la descripción de sus nodos.

Figura 10 Diagrama de despliegue del sistema.

Tabla 9 Descripción de los nodos correspondientes al diagrama de despliegue del sistema.

Nodos	Descripción
PC_Cliente	Nodo que representa aquellas PC que pueden ser utilizadas por los usuarios para acceder a la aplicación.
Servidor Web	Nodo que representa el servidor donde está alojada la aplicación destinada a la gestión de incidencias del Grupo Azucarero AZCUBA.
Servidor de Base de Datos	Nodo que representa el servidor donde se encuentra la Base de Datos que contiene los datos persistentes de la aplicación.

3.3 Modelo de pruebas

El modelo de pruebas describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema, así como otros aspectos específicos del sistema (JACOBSON, IVAR *et al.* 2000a). Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (IEEE 1995).

Según Pressman los objetivos fundamentales del proceso de prueba están dados por las siguientes afirmaciones:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces (PRESSMAN, ROGER S. 2008).

3.3.1 Tipos de pruebas

Existen varios tipos de pruebas, algunas recomendadas específicamente para aplicaciones web, como las que se utilizan en esta investigación y que a continuación se explican:

Pruebas de función o a nivel de componentes, que ejercitan el contenido y las unidades funcionales dentro de la aplicación y se enfocan sobre un conjunto de pruebas que intentan descubrir errores en la misma.

Pruebas de desempeño o de rendimiento, que se aplican para descubrir problemas debido a la falta de recursos en el lado del servidor, ancho de banda de red inapropiado, capacidades inadecuadas de bases de datos, defectuosas o débiles capacidades del sistema operativo, funcionalidades mal diseñadas y otros conflictos de hardware o software que pueden conducir a un pobre desempeño cliente - servidor (PRESSMAN, ROGER S. 2008).

Pruebas de aceptación, que representan aquella fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y el área usuaria de un sistema de información tienen que garantizar que el sistema desarrollado se corresponde con los requerimientos definidos (GONZÁLEZ, JESUS PONCE et al. 2014). En la presente investigación se utilizaron estas pruebas específicamente las de tipo Alfa, que son las que realiza el usuario final, una vez recibido el producto terminado y su documentación, de conjunto con los desarrolladores del sistema. Para ello se le entregó al especialista superior el software terminado, junto a una guía para el desarrollo de estas pruebas. Este proceso se realizó en la Dirección de Informática, Comunicaciones y Análisis del Grupo Azucarero AZCUBA, en presencia de los desarrolladores y los especialistas del área, que se encargaron de comprobar todas las funcionalidades y de informar de las inconsistencias y errores que detectaron.

3.3.2 Métodos y técnicas de prueba

Para desarrollar las pruebas de función o a nivel de componentes se aplica el método de prueba de **Caja Negra**, también denominado Pruebas de Comportamiento. Este método permite obtener un conjunto de condiciones de entrada que ejerciten por completo los requisitos funcionales de un programa, ignorando la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Las pruebas de Caja Negra según Pressman buscan encontrar errores en cinco categorías:

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y terminación

Dentro del método de Caja Negra se decide el uso de la técnica de Partición de equivalencia, que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. Además, esta técnica se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de pruebas a desarrollar. Para su implementación se divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software a partir de las cuales pueden derivarse casos de prueba (PRESSMAN, ROGER S. 2008).

Para desarrollar las pruebas de desempeño o de rendimiento se pueden aplicar pruebas de carga y de tensión. En el caso de la presente investigación debido a que la propuesta de solución durante su despliegue no estará expuesta a grandes niveles de tensión ni concurrencia significante de usuarios, se define realizar solamente **pruebas de carga**, que tienen como objetivo determinar cómo la aplicación en su ambiente del lado del servidor responderá ante varias condiciones de carga, a partir de condiciones de pruebas definidas por las permutaciones entre variables que dependen del número de usuarios concurrentes, el número de transacciones en línea por usuario por unidad de tiempo y la carga de datos procesada por el servidor por transacción (PRESSMAN, ROGER S. 2008).

3.3.3 Diseño de Casos de Prueba (DCP)

Según (PEÑA 2006) “el conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito” es a lo que podríamos llamar o determinar cómo caso de prueba.

DCP del caso de uso: Gestionar incidencias

- **Descripción general**

El presente Caso de Uso se inicia cuando el usuario despliega la opción Gestionar incidencia, selecciona las operaciones de adicionar una nueva incidencia o listar las existentes y culmina con la realización de una de las acciones *Adicionar*, *Editar*, *Mostrar o Eliminar* alguna incidencia.

- **Condiciones de ejecución**

El usuario debe estar autenticado en el sistema, así como contar con los permisos para realizar alguna de las operaciones antes mencionadas. En el caso de que se pretenda *Actualizar*, *Mostrar o Eliminar* alguna incidencia, esta debe haber sido adicionada anteriormente al sistema. En el caso de que se pretenda actualizar una incidencia, el usuario solo podrá realizar la acción si la misma no ha sido analizada, en otro caso solo podrá eliminarla y consultar sus detalles.

- **Secciones a probar en el Caso de Uso: Gestionar incidencia**

Tabla 10 Secciones a probar. Caso de uso ‘‘Gestionar incidencia’’.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: “Crear Incidencia”.	**EC 1.1:** El usuario inserta y selecciona datos correctos en los campos de entrada.	El sistema almacena una nueva Incidencia. El sistema muestra un mensaje indicando que se ha creado la Incidencia por tanto debe cargar sus documentos asociados.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Crear Incidencia”. ▪ Inserta datos en los campos. ▪ Clic en el botón “Crear”.
	EC 1.2: El usuario deja campos vacíos.	El sistema muestra un mensaje indicando que se deben especificar todos los campos para crear una incidencia.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Crear Incidencia”. ▪ Inserta datos en los campos. ▪ Clic en el botón “Crear”.
	EC 1.3: El usuario introduce datos incorrectos.	El sistema muestra un mensaje indicando que existen datos incorrectos.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Crear Solicitud”. ▪ Inserta datos en los campos. ▪ Clic en el botón “Crear”.
SC 2: “Actualizar Incidencia”.	**EC 2.1:** El usuario modifica correctamente los datos en los campos.	El sistema modifica los datos persistentes de la incidencia y muestra un mensaje confirmando el éxito de la operación.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Listar Incidencias”. ▪ Clic en el icono actualizar de la incidencia dada. ▪ Insertar datos a actualizar. ▪ Clic en el botón “Actualizar”.
	EC 2.2: El usuario deja campos vacíos.	El sistema muestra un mensaje indicando que todos los campos son requeridos.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Listar Incidencias”. ▪ Clic en el icono actualizar de la incidencia dada.

			▪ Insertar datos a actualizar. ▪ Clic en el botón “Actualizar”.
	EC 2.3: El usuario introduce datos incorrectos.	El sistema muestra un mensaje indicando que existen datos incorrectos.	▪ Clic en “Gestionar Solicitud”. ▪ Clic en “Listar Incidencias”. ▪ Clic en el icono actualizar de la incidencia dada. ▪ Insertar datos a actualizar. ▪ Clic en el botón “Actualizar”.
SC 3: “Ver Detalles Incidencia”.	**EC 3.1:** Ver detalles de una incidencia.	El sistema muestra los datos de una incidencia dada así como la posibilidad de descargar sus documentos asociados y ver sus recomendaciones, si estas existen.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Listar Incidencias”. ▪ Clic en el icono ver detalles de la incidencia dada.
SC 4: “Eliminar Incidencia”.	**EC 4.1:** Eliminar una incidencia.	El sistema elimina la incidencia dada, sus documentos asociados y recomendaciones. Muestra un mensaje de confirmación.	▪ Clic “Gestionar Incidencia”. ▪ Clic en “Listar Incidencias”. ▪ Clic en el icono eliminar de la incidencia dada.

3.3.4 Resultados de la aplicación de las pruebas.

Como parte de la ejecución de las pruebas de Caja Negra, con el objetivo esencial de identificar en qué medida satisface la aplicación las funcionalidades implementadas, se realizó una primera iteración de pruebas, en la que fueron aplicados los diseños de casos de pruebas realizados a partir de las descripciones textuales de los Casos de Uso del Sistema. En esta primera iteración se identificaron un total de 23 No Conformidades, distribuidas conforme se muestra en la figura11. Una vez corregidas las No Conformidades anteriores, se ejecutan nuevamente los diseños elaborados en una segunda iteración obteniéndose un total de ocho No Conformidades. Luego de corregidas estas últimas No Conformidades se decide la realización de una tercera iteración de los diseños anteriormente aplicados obteniéndose resultados satisfactorios por lo que se determina la no realización de una nueva iteración.

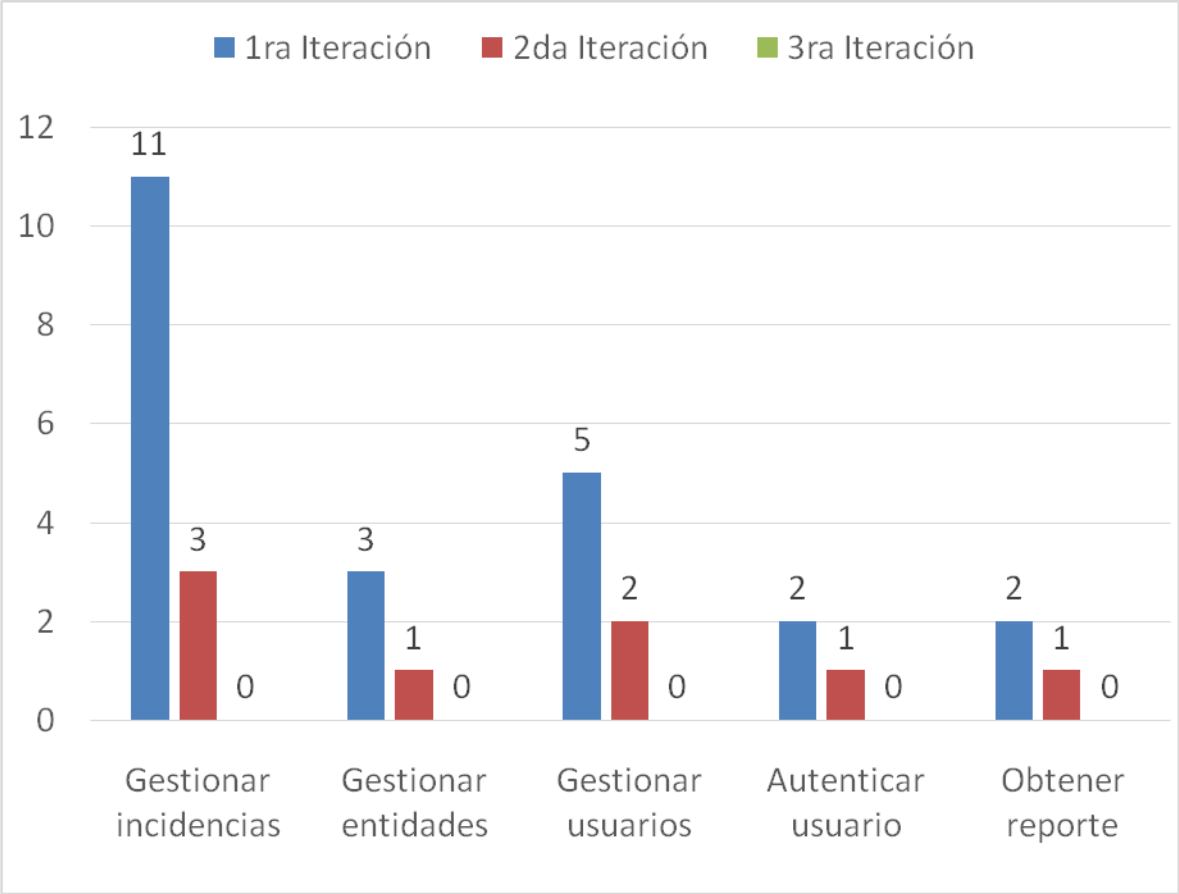
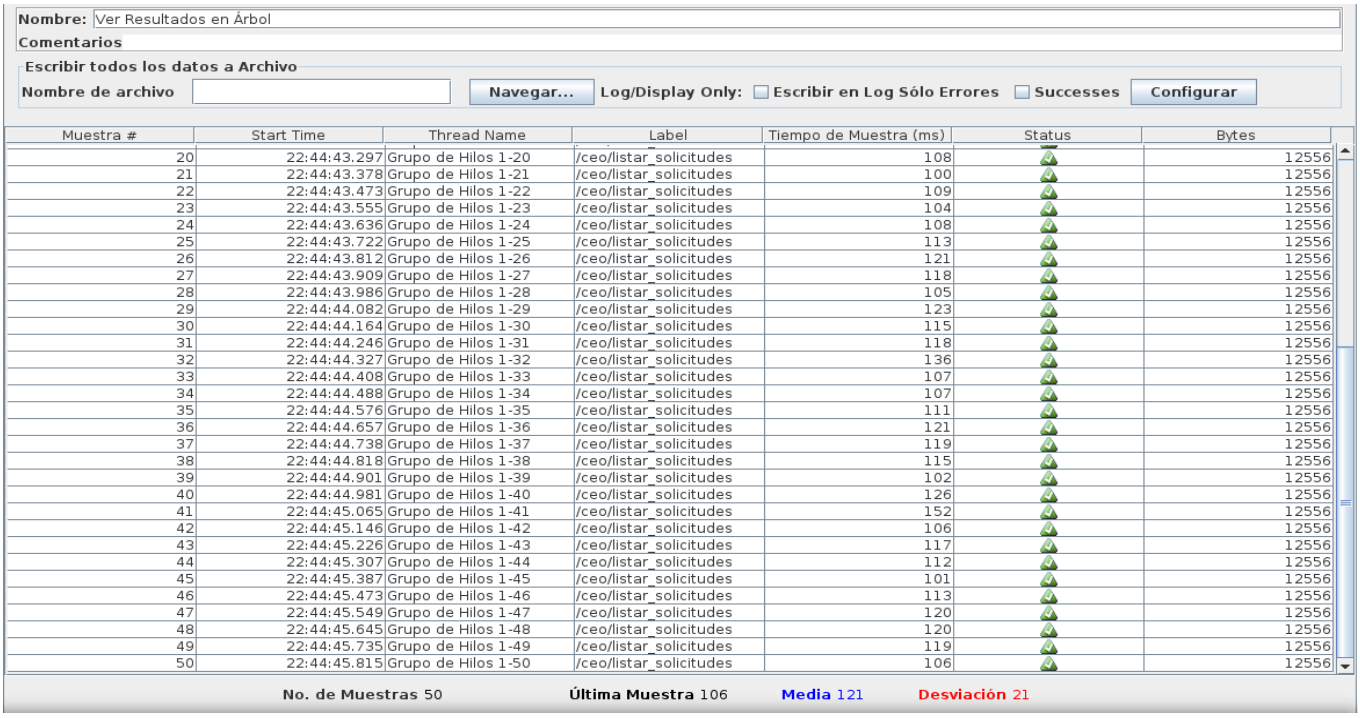


Figura 11 Resultado de la aplicación de las pruebas.

Al realizarse la ejecución de las pruebas de carga, se obtuvieron los resultados obtenidos a partir del uso de la herramienta Apache JMeter 2.3.4.

Los resultados obtenidos como parte de la aplicación de la prueba de carga, realizada con una simulación de 50 usuarios concurrentes en el sistema. En la figura. 12 se reflejan datos como el tiempo de respuesta en milisegundos y los errores mediante la columna Status para cada petición y la media del tiempo de respuesta, que no sobrepasa los 5 segundos conforme a lo definido en el Requisito No Funcional de eficiencia.



Muestra #	Start Time	Thread Name	Label	Tiempo de Muestra (ms)	Status	Bytes
20	22:44:43.297	Grupo de Hilos 1-20	/ceo/listar_solicitudes	108		12556
21	22:44:43.378	Grupo de Hilos 1-21	/ceo/listar_solicitudes	100		12556
22	22:44:43.473	Grupo de Hilos 1-22	/ceo/listar_solicitudes	109		12556
23	22:44:43.555	Grupo de Hilos 1-23	/ceo/listar_solicitudes	104		12556
24	22:44:43.636	Grupo de Hilos 1-24	/ceo/listar_solicitudes	108		12556
25	22:44:43.722	Grupo de Hilos 1-25	/ceo/listar_solicitudes	113		12556
26	22:44:43.812	Grupo de Hilos 1-26	/ceo/listar_solicitudes	121		12556
27	22:44:43.909	Grupo de Hilos 1-27	/ceo/listar_solicitudes	118		12556
28	22:44:43.986	Grupo de Hilos 1-28	/ceo/listar_solicitudes	105		12556
29	22:44:44.082	Grupo de Hilos 1-29	/ceo/listar_solicitudes	123		12556
30	22:44:44.164	Grupo de Hilos 1-30	/ceo/listar_solicitudes	115		12556
31	22:44:44.246	Grupo de Hilos 1-31	/ceo/listar_solicitudes	118		12556
32	22:44:44.327	Grupo de Hilos 1-32	/ceo/listar_solicitudes	136		12556
33	22:44:44.408	Grupo de Hilos 1-33	/ceo/listar_solicitudes	107		12556
34	22:44:44.488	Grupo de Hilos 1-34	/ceo/listar_solicitudes	107		12556
35	22:44:44.576	Grupo de Hilos 1-35	/ceo/listar_solicitudes	111		12556
36	22:44:44.657	Grupo de Hilos 1-36	/ceo/listar_solicitudes	121		12556
37	22:44:44.738	Grupo de Hilos 1-37	/ceo/listar_solicitudes	119		12556
38	22:44:44.818	Grupo de Hilos 1-38	/ceo/listar_solicitudes	115		12556
39	22:44:44.901	Grupo de Hilos 1-39	/ceo/listar_solicitudes	102		12556
40	22:44:44.981	Grupo de Hilos 1-40	/ceo/listar_solicitudes	126		12556
41	22:44:45.065	Grupo de Hilos 1-41	/ceo/listar_solicitudes	152		12556
42	22:44:45.146	Grupo de Hilos 1-42	/ceo/listar_solicitudes	106		12556
43	22:44:45.226	Grupo de Hilos 1-43	/ceo/listar_solicitudes	117		12556
44	22:44:45.307	Grupo de Hilos 1-44	/ceo/listar_solicitudes	112		12556
45	22:44:45.387	Grupo de Hilos 1-45	/ceo/listar_solicitudes	101		12556
46	22:44:45.473	Grupo de Hilos 1-46	/ceo/listar_solicitudes	113		12556
47	22:44:45.549	Grupo de Hilos 1-47	/ceo/listar_solicitudes	120		12556
48	22:44:45.645	Grupo de Hilos 1-48	/ceo/listar_solicitudes	120		12556
49	22:44:45.735	Grupo de Hilos 1-49	/ceo/listar_solicitudes	119		12556
50	22:44:45.815	Grupo de Hilos 1-50	/ceo/listar_solicitudes	106		12556

Figura 12 Interfaz de salida de JMeter.

Luego de la aplicación de una prueba de tensión o estrés sobre la aplicación, haciendo uso de la herramienta Apache JMeter 2.3.4, simulando el uso del sistema con más del doble de los usuarios respecto al requisito de eficiencia definido, esta fue realizada con una simulación de 150 usuarios realizando peticiones concurrentes al sistema. Se aprecian en la figura. 13 datos referentes al tiempo de respuesta en milisegundos, los errores en la columna Status para cada petición y la media del tiempo de respuesta que no excede los 15 segundos, acorde a una medida aceptable de acuerdo a la respuesta que debe brindar la aplicación ante una situación de estrés.

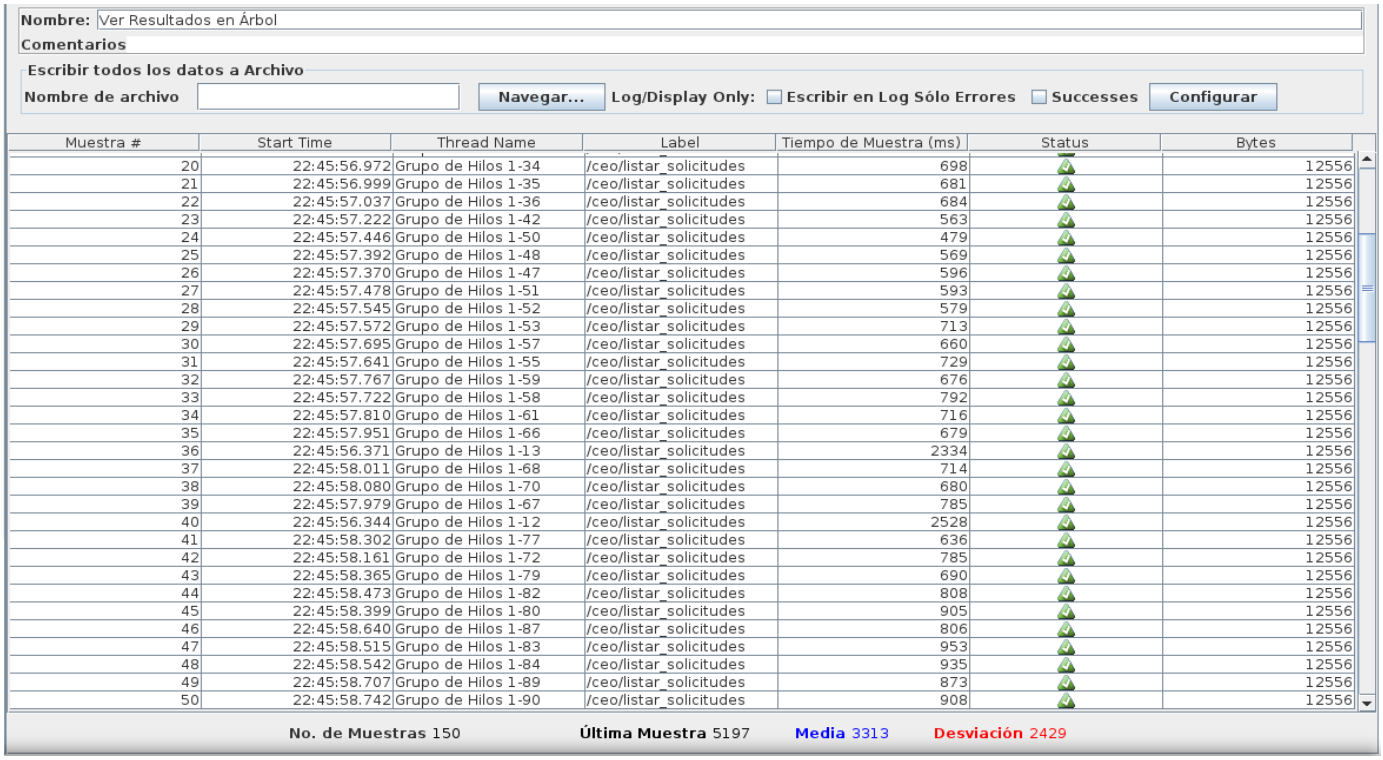


Figura 13 Interfaz de salida de JMeter.

3.4 Conclusiones parciales

En el desarrollo de este capítulo se obtuvo el diagrama de componente y el diagrama de despliegue. Se aplicaron las pruebas funcionales, de rendimiento y de aceptación que permitieron identificar y resolver los errores en la implementación aumentando la calidad del producto final obtenido. Además se pudo comprobar que el mismo satisface las funcionalidades requeridas.

Conclusiones Generales

Con la investigación realizada se logró cumplir el objetivo general, desarrollando un sistema capaz de mejorar el control de las incidencias en el Grupo Azucarero AZCUBA. Teniendo en cuenta lo anterior se puede plantear que:

1. El análisis de los principales conceptos asociados a la gestión de incidencias y de las principales soluciones existentes, logró que se identificaran las tendencias y las características de estas que pudieran aportar al diseño de la solución.
2. La realización del análisis y diseño utilizando la metodología Open UP contribuyó a obtener los artefactos para el desarrollo del sistema permitiendo su implementación.
3. La implementación del Sistema de Gestión de Incidencias para el Grupo Azucarero AZCUBA permitió que se obtuviera una aplicación que mejorara el control de los procesos que se desarrollan en la entidad.
4. La validación de la solución mediante la aplicación de las pruebas funcionales, de rendimiento y aceptación permitió obtener un sistema funcional que respondiera a las necesidades del cliente.

Recomendaciones

1. Elaborar un manual de usuario que sirva de guía para la explotación óptima del sistema.
2. Desarrollar funcionalidades para la gestión de plantillas personalizadas.

Referencias Bibliográficas

ABRAN, A.; J. W. MOORE*, et al. Guide to the software engineering body of knowledge: 2004 version SWEBOK*. IEEE Computer Society, 2004. p. 0769523307

ARTOLOGIK. *Portal Web Artologik.com*, 2017. [2016]. Disponible en: http://www.artologik.com/es/HelpDesk/Sobre-HelpDesk.aspx

ATLASSIAN. *Jira Software: La herramienta de desarrollo de software líder de los equipos ágiles*, 2017. [2017]. Disponible en: https://es.atlassian.com/software/jira

AZCUBA, G. A. *Portal Web de la insdustria azucarera cubana*, 2014. [Disponible en: http://www.azcuba.cu/

---. *Reglamento del Grupo Azucarero AZCUBA*. 2012. 359 p.

BALDUINO, R. *Introduction to openup*, Disponible en Internet: http://www. eclipse. org/epf/general/OpenUP. pdf, consultado el, 2007.

BASALO, A. and M. A. ÁLVAREZ. *Qué es AngularJS*, 2014. [2016]. Disponible en: https://desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html

BLASCO, D. *Ventajas y desventajas de programar en AngularJS*, 2016. [2016]. Disponible en: https://www.atraura.com/angularjs/

BUSCHMANN, F.; K. HENNEY*, et al. Pattern-oriented Software Architecture: on patterns and pattern language*. John wiley & sons, 2007. p. 8126512830

DAVIS, G. B. and M. H. OLSON. *Management information systems: conceptual foundations, structure, and development*. McGraw-Hill, Inc., 1984. p. 0070158282

DELGADO MAGDALENA, J. *Tutorial de Sails.js: Instalación y primer proyecto*, 2015. [2016]. Disponible en: https://openwebinars.net/blog/tutorial-sailsjs-instalacion/

ECLIPSE. *Ciclo de vida de un proyecto según OpenUP*, 2011. [2016]. Disponible en: http://epf.eclipse.org/wikis/opneup

EGUÍLUZ PÉREZ, J. *Introducción a javascript*, 2012.

ESTUPIÑÁN DÍAZ, S. and I. VARGAS VARGAS Mainpack 10.0. Software para la gestión de la actividad de mantenimiento en la industria azucarera *ICIDCA. Sobre los Derivados de la Caña de Azúcar*, 2015, 49(2).

FIGUEROA, R. G.; C. J. SOLÍS*, et al.* Metodologías tradicionales vs. Metodologías ágiles *Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.(En línea), Disponible en: http://adonisnet. files. wordpress. com/2008/06/articulo-metodologia-de-sw-formato. doc*, 2008.

FLANAGAN, D. *JavaScript: The Definitive Guide*, 2002a.

---. *Javascript: the definitive guide. OLReilly Media*, Inc, 2002b.

FOWLER, M. *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002. p. 0321127420

GONZÁLEZ, A. H. Autora: Dra. Anaisa Hernández González Facultad de Ingeniería Industrial Instituto Superior Politécnico José Antonio Echeverría, 2004.

GONZÁLEZ, J. P.; F. J. DOMINGUEZ*, et al. Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental*. España, Universidad de Sevilla, 2014.

GROUP, T. P. G. D. *PostgreSQL*, 2017. [2016]. Disponible en: https://www.postgresql.org/

IEEE IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools, 1995.

JACOBSON, I.; G. BOOCCH*, et al. El proceso unificado de desarrollo de software*. Madrid, España, Pearson Educacion S.A., 2000a. 464 p.

JACOBSON, I.; G. BOOCH*, et al. Rational Unified Proces*, 2007.

JACOBSON, I.; G. RUMBAUGH*, et al. El proceso unificado de desarrollo de software/The unified software development process*. Pearson Educación, 2000b. p. 8478290362

LARMAN, C. *UML y Patrones*. Pearson, 1999. p. 8420534382

---. *UML y Patrones*. 2da. Prentice Hall, 2003. p.

MANTISBT. *Mantis Bug Tracker*, 2017. [2017]. Disponible en: http://www.mantisbt.org/

MARQUÉS, M. *Bases de datos*. Castelló de la Plana: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions, 2011., 2011. p. 8469301462

MARTINEZ GUERRERO, R. *PostgresSQL-es*, 2010. [2016]. Disponible en: http://www.postgresql.org.es/

NETBEANSIDE. *Portal Web NetBeans IDE*, 2017. [2017]. Disponible en: https://netbeans.org/

NODE.JS. *Node.js*®, 2017. [2017]. Disponible en: https://nodejs.org/es/

PARRA, E. *Instalar Netbeans en español en Ubuntu* 2011. [2016]. Disponible en: http://www.eduardoparra.es/2011/01/instalar-netbeans-en-espanol-en-ubuntu.html

PEÑA, J. M. F. *Pruebas de Software*, 2006. [Disponible en: htpp://www.uv.mx/personal

PGADMIN. *pgAdmin PostgreSQL Tools*, 2017. [2017]. Disponible en: https://www.pgadmin.org/

PONJUÁN DANTE, G. *Gestión de información en las organizaciones: principios, conceptos y aplicaciones*. Santiago, CL: Universidad de Chile, Centro de Información en Capacitación, 1998. p. 9567782008

PRESSMAN, R. S. *Ingeniería de Software Un Enfoque Práctico*. 6ta. 2005. 958 p. 0-07-285318-2

---. *Ingeniería de Software. Un enfoque práctico*. 5ta. 2002a. p. 8448132149

PRESSMAN, R. S. *Ingeniería del Software*. 5ta. La Habana, 2002b. 579 p.

PRESSMAN, R. S. *Ingeniería del Software*. 2008. p.

RUMBAUGH, J.; G. JACOBSON*, et al. El lenguaje unificado de modelado: manual de referencia*. Addison Wesley, 2000. p. 8478290370

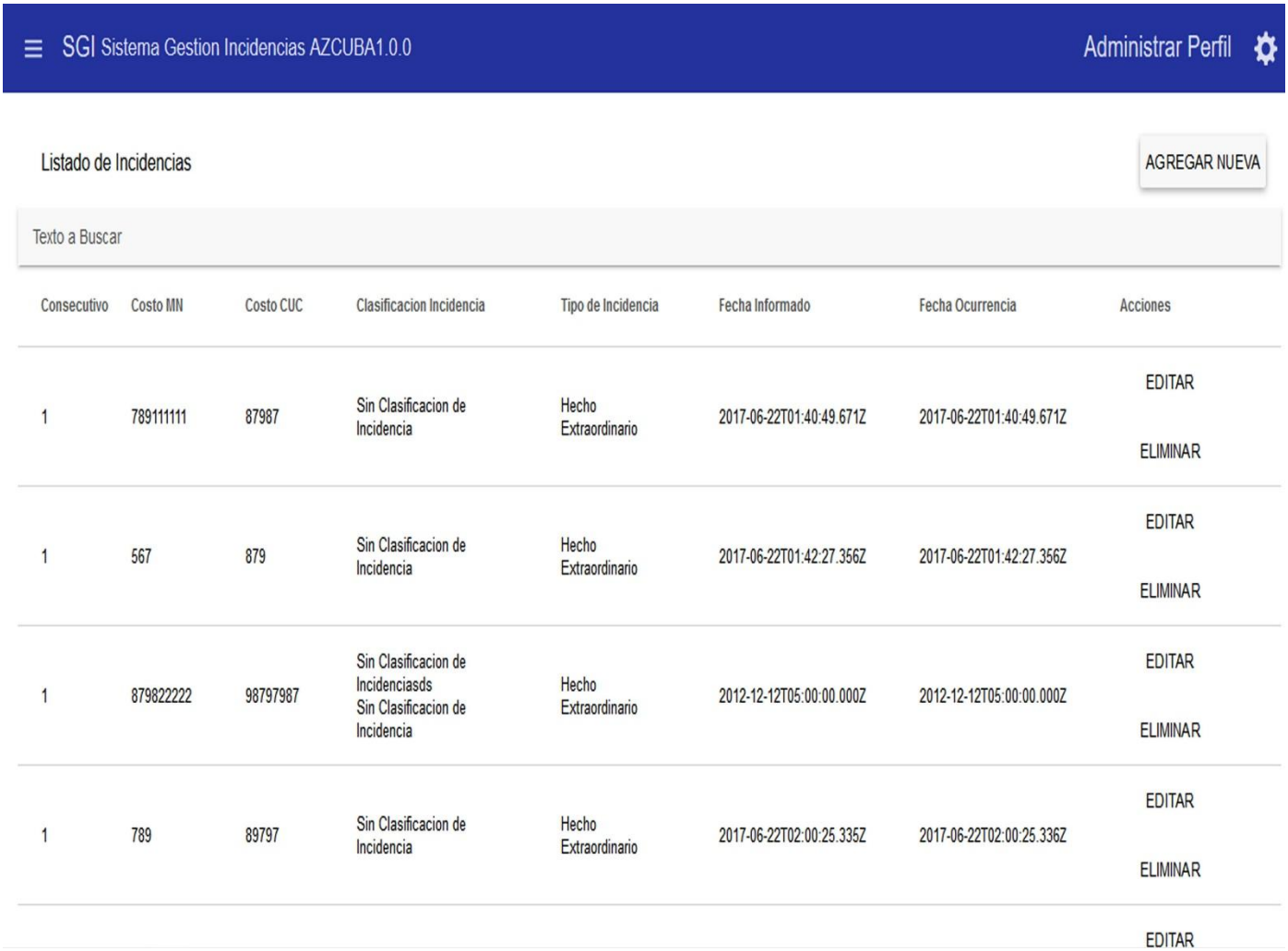
SAAVEDRA LÓPEZ, D.; Y. ARMENTERO MORENO*, et al.* Aplicación web para la realización de estudios farmacocinéticos, versión 2.0 *Revista Cubana de Informática Médica*, 2013, 5(2): 118-131.

SOMMERVILLE, I. *Ingeniería del software*. Pearson Educación, 2005. p. 8478290745

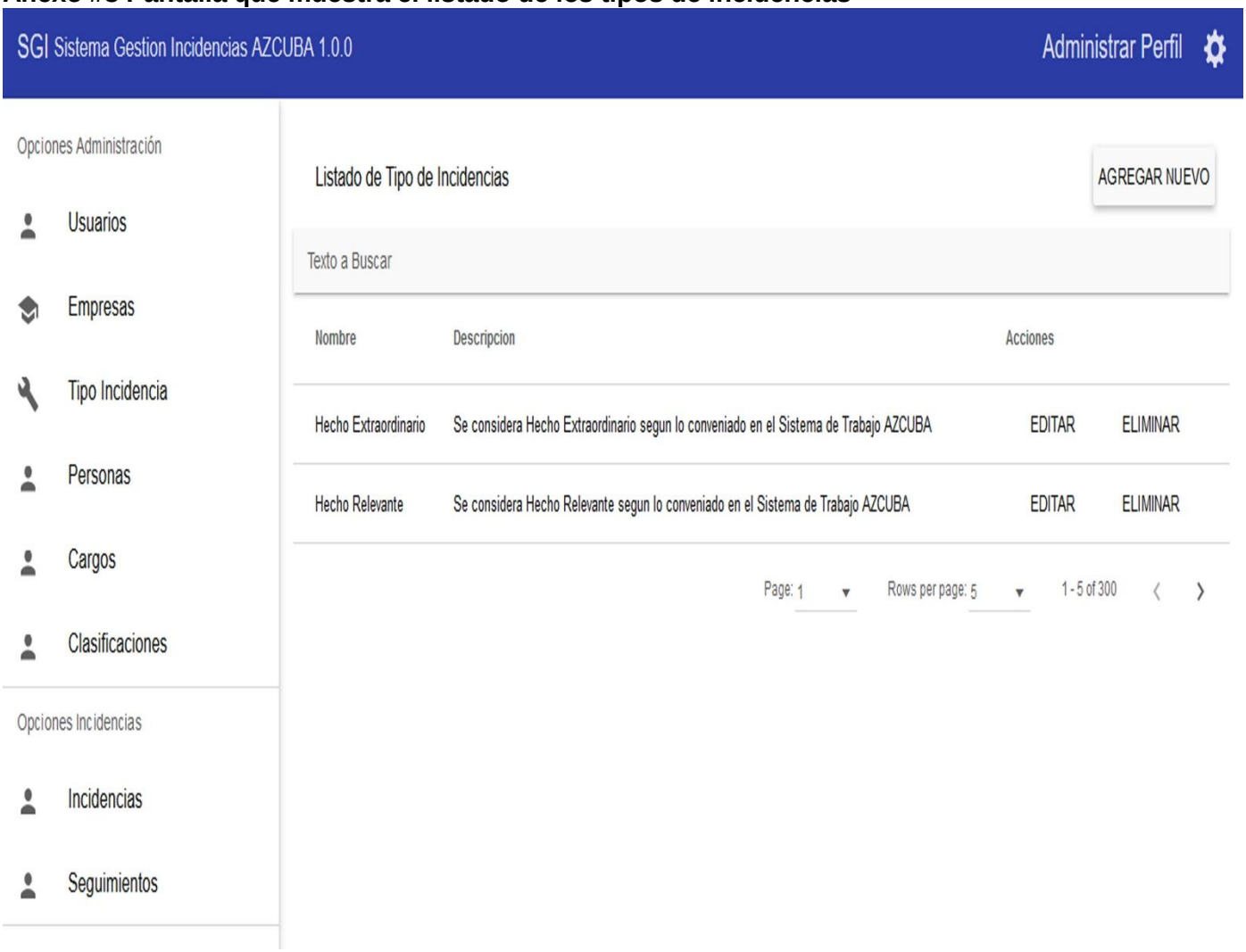
Anexos

Anexo #1 Pantalla principal

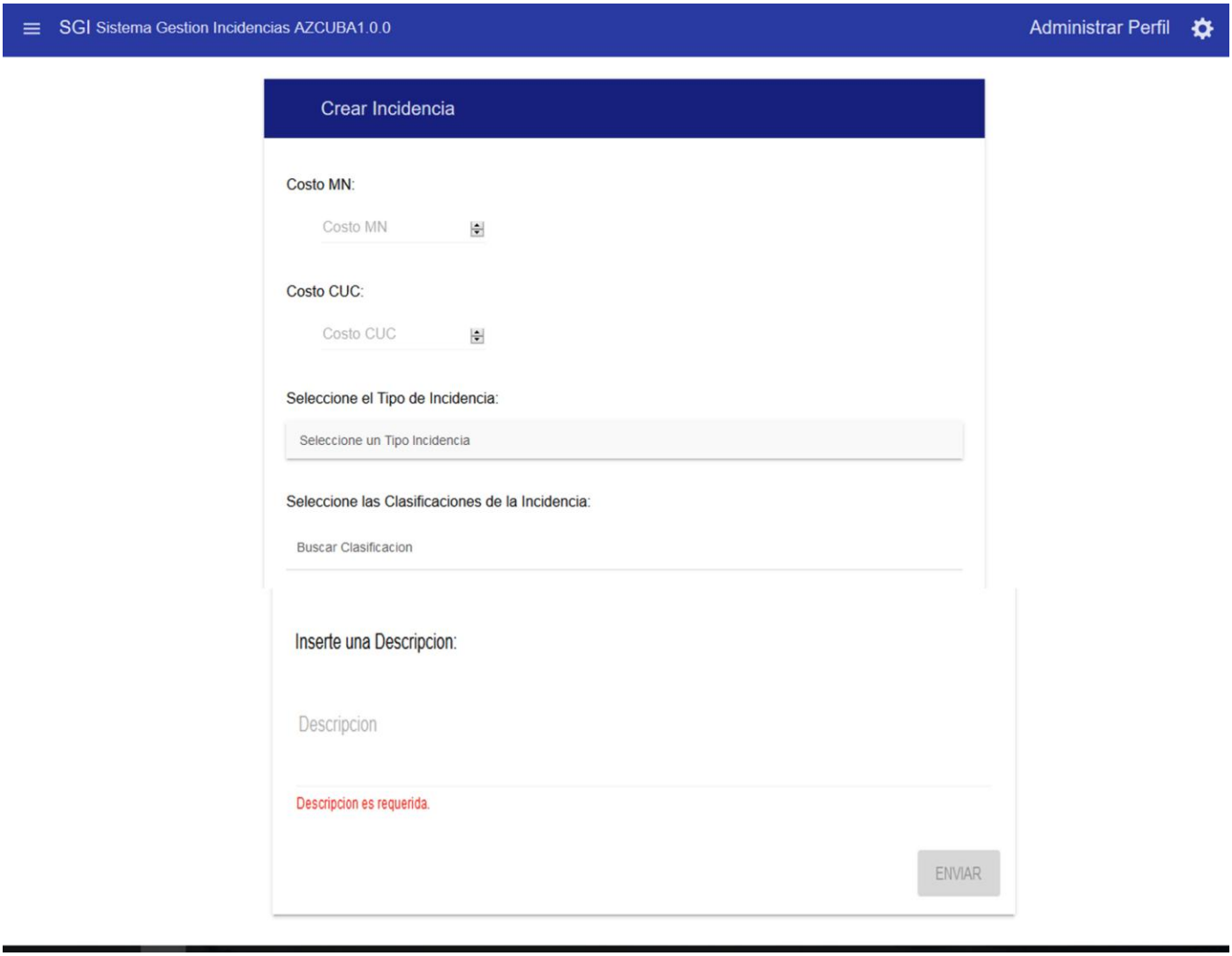
Anexo #2 Listado de incidencias



Anexo #3 Pantalla que muestra el listado de los tipos de incidencias



Anexo #4 Pantalla que muestra la funcionalidad Crear las incidencias



Anexo #5 Acta de aceptación del Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA

Dirección de Informática, Comunicaciones y Análisis

La Habana, 23 de Junio de 2017
"Año 59 de la Revolución"

ACTA DE ACEPTACIÓN

De una parte, el director de la Dirección de Informática, Comunicaciones y Análisis del Grupo Azucarero AZCUBA, representado en este acto por: Msc. Dionis Pérez Pérez, y de **otra parte** la estudiante: Tania Elena Castro Aguilera

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

POR TANTO: Las Partes acuerdan formalizar mediante la presente Acta, la aceptación del producto:

Sistema de Gestión de Incidencias del Grupo Azucarero AZCUBA

Y para que así conste, se extiende la presenta **Acta** en dos (2) ejemplares, rubricados por **Las Partes**.

Entrega
Tania Elena Castro Aguilera

Recibe
Msc. Dionis Pérez Pérez