Universidad de las Ciencias Informáticas

Facultad 4


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

# *Módulo de Gestión de Incidencias del Sistema de Gestión de los Servicios del Centro de Soporte de la Universidad de las Ciencias Informáticas (UCI)*

*Autores:* Carlos Alberto Nuez García

Yunior Quesada Magdaleno

*Tutores:* MSc. Yulio Seriocha García Gallardo

Ing. Neybis Lago Clara

*La Habana, junio 2014.*

*"Año 56 de la Revolución"*

# *DECLARACIÓN DE AUTORÍA*

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) que haga el uso que estime conveniente con este trabajo.

Para que así conste firmo la presente a los 24 días del mes de junio del año 2014.

Carlos Alberto Nuez García

_____

Autor

Yunior Quesada Magdaleno

_____

Autor

MSc. Yulio Seriocha García Gallardo

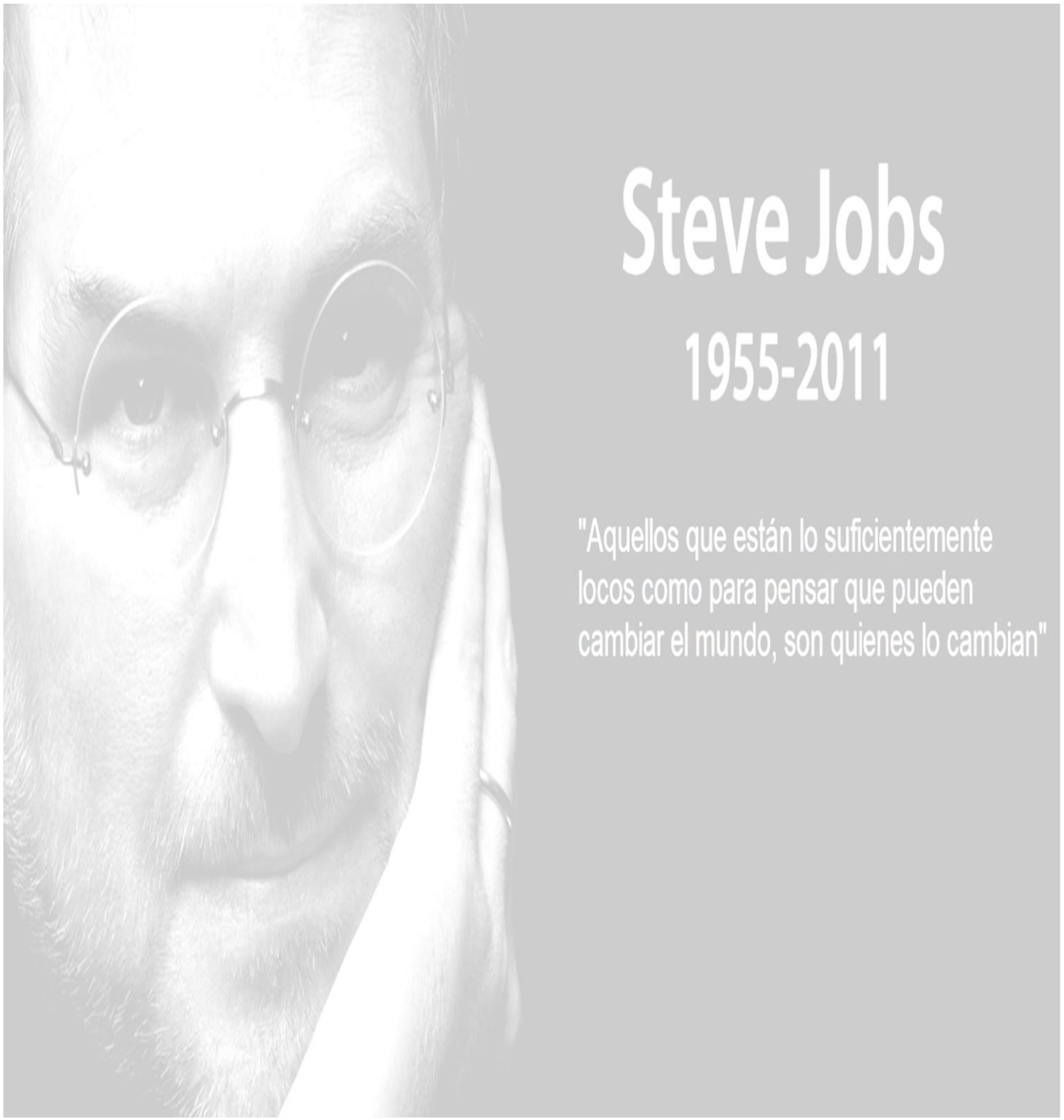
_____

Tutor

Ing. Neybis Lago Clara

_____

Tutor


Steve Jobs
1955-2011
"Aquellos que están lo suficientemente locos como para pensar que pueden cambiar el mundo, son quienes lo cambian"

# *DEDICATORIA*

## *De Junior:*

*Dedico este diploma a lo más grande que tengo en la vida, mi familia. En especial a mi abuelita que no me puede acompañar físicamente, pero en lo espiritual siempre está presente. A mi mamá y mi papá por estar siempre junto a mí en cada etapa de mi vida, brindándome su comprensión, cariño y confianza. A mis hermanos por ser cómplices de mis acciones. A mi tío por estar siempre ahí y brindarme su apoyo, a mi cuñadita, a mi sobrinita linda. A mi pareja por acompañarme en todo momento y brindarme todo su amor y comprensión.*

## *De Carlos:*

*A todo el que ha querido verme como ingeniero y que me ha apoyado para serlo.*

**De Junior:**

*Quiero agradecer a mi familia en general, a mi mamá, mi papá, mis hermanos, a mis tíos, a mi cuñadita, a mi sobrinita y a mi novia, por creer en mí. A todas las personas que aportaron su granito de arena y siempre desearon ver este sueño hecho realidad.*

**De Carlos:**

*Agradezco a mi familia especialmente a mi papá y a mi tía que siempre han estado ahí para mí.*

# *RESUMEN*

El software de línea de asistencia actualmente en uso por el Centro de Soporte de la Universidad de las Ciencias Informáticas (UCI), ManageEngine ServiceDesk Plus, proporciona la administración de incidencias, problemas, soluciones, cambios e inventarios. Debido a su carácter privativo representa un freno para el avance de este Centro, dado que sus funcionalidades no logran satisfacer totalmente las crecientes necesidades del Centro de Soporte y no se puede modificar su código fuente para adecuarlas a sus exigencias. Otras causas de la insatisfacción de los clientes son: la demora en el tiempo de servicio y que no se tienen en cuenta sus expectativas y niveles de satisfacción para con el servicio de soporte técnico. En este sentido, el presente Trabajo de Diploma aborda el desarrollo de un módulo para la gestión de incidencias basado en el estándar ITIL 2011, que permita contribuir a la calidad de los servicios que ofrece dicho Centro.

Esta aplicación *web* (como principal resultado de la investigación) facilita la gestión de incidencias; permite vincularlas, asociarlas y administrar todos sus componentes. Incorpora las últimas y mejores prácticas recomendadas por ITIL 2011 para gestionar incidencias. Para su implementación se utilizaron los lenguajes de programación Groovy, Javascript, CSS3, HTML5 y como *frameworks* Grails, Boostrap y jQuery. El proceso de desarrollo del software estuvo guiado por la metodología XP y fueron generados los artefactos correspondientes en cada una de las actividades del marco de trabajo.

Palabras claves:

Aplicación *web,* artefacto, gestión, incidencia, ITIL 2011, metodología, módulo, servicio, soporte.

# ÍNDICE

# ÍNDICE DE FIGURAS

# *ÍNDICE DE TABLAS*

## Introducción

En la actualidad el desarrollo de las Tecnologías de la Información (TI) ha propiciado un alto nivel de dependencia por parte del sector empresarial. “Las TI han sido conceptualizadas como aquellas herramientas y métodos empleados para recabar, retener, manipular o distribuir información. Las TI se encuentran generalmente asociadas con las computadoras y las tecnologías afines aplicadas a la toma de decisiones” (G. Jack Bologna 1997).

La mayoría de las organizaciones hoy día hacen uso de las TI para gestionar en gran medida los aspectos del negocio, como el manejo de registros financieros, registros de empleados y facturación. Otras de las disímiles ventajas que posibilita su uso son: lograr un mejor desempeño en cuanto a tomas de decisiones y control, el almacenamiento de datos, así como la planeación de recursos empresariales. “Un servicio de tecnologías de la información es un conjunto de actividades que buscan responder las necesidades de un cliente por medio de un cambio de condición en los bienes informáticos, potenciando el valor de estos y reduciendo el riesgo inherente del sistema” (OnetOne 2013). El uso de los servicios de TI se ha convertido en una práctica generalizada por parte de las empresas y esta dependencia ha propiciado su auge.

El auge y el constante avance en los servicios de las TI ha aumentado su complejidad, lo cual hace su manejo cada vez más necesario para asegurar la calidad de la información y el soporte del cumplimiento de los objetivos del negocio. La Gestión de Servicios de Tecnologías de la Información (GSTI*)* “es una disciplina basada en procesos, enfocada en alinear los servicios de TI proporcionados con las necesidades de las empresas, poniendo énfasis en los beneficios que puede percibir el cliente final” (Ingenius 2009). Para lograr su optimización las organizaciones, de acuerdo a sus necesidades, pueden utilizar diferentes estándares.

La GSTI está enfocada en la promoción y soporte de aplicar las buenas prácticas y estándares de aceptación internacional como lo son: Objetivos de Control para la Información y Tecnologías (COBIT, según sus siglas en inglés), la Organización Internacional para Estandarización/Comisión Electrónica Internacional (ISO/IEC, según sus siglas en inglés) 20000 y el Modelo de Madurez de la Capacidad Integrado para Servicios (CMMI-SVC, según sus siglas en inglés) y en el que se profundizará durante la investigación en curso: la Biblioteca de Infraestructura de Tecnología de Información (ITIL, según sus siglas en inglés). “ITIL puede ser definido como un conjunto de buenas prácticas destinadas a mejorar la gestión y provisión de servicios TI. Su objetivo es mejorar la calidad de los servicios TI ofrecidos, evitar los problemas asociados a los mismos y en caso de que estos ocurran ofrecer un marco de actuación para que estos sean

solucionados con el menor impacto y a la mayor brevedad posible" (OSIATIS 2006b).

La función principal de ITIL es ofrecer un marco para facilitar todas las tareas y procesos a los proveedores y receptores de servicios de TI con calidad y un coste adecuado. ITIL posibilita alinear mejor los servicios de TI a los requerimientos del negocio. También establece ciertas disciplinas, entre ellas "Soporte del Servicio", la cual trata de asegurar el acceso apropiado a los servicios prestados a los clientes.

El Soporte del Servicio, de forma general, ayuda proactivamente a los clientes a disminuir los riesgos de indisponibilidad, aumentar en capacidad y rendimiento de gestión de los sistemas. Es el encargado de garantizar la continuidad, disponibilidad y calidad del servicio prestado al usuario. La ocurrencia de alguna irregularidad de las mencionadas anteriormente se considera como incidencia, definida según el libro de Soporte del Servicio de ITIL como: "cualquier evento que no forma parte de la operación estándar de un servicio y que causa, o puede causar, una interrupción o una reducción de calidad del mismo" (OSIATIS 2006c). La gestión de dichas incidencias está enfocada en resolver de manera rápida y eficaz cualquier incidente que cause alteraciones en el servicio que se ofrece.

Cuba no está ajena al desarrollo de la ciencia y la tecnología, desde hace un tiempo se perfecciona una estrategia enfatizada en el avance de la industria del software. Tanto así que se han creado una serie de empresas e instituciones como lo son: Desarrollo Automatizado de Tecnología Informática y Seguridad (DATYS), Empresa Nacional de Software (DESOFT), Empresa Productora de Software para la Técnica Electrónica (SOFTEL), Empresa de automatización integral (CEDAI), así como la Universidad de las Ciencias Informáticas (UCI).

La UCI es una institución donde se combina el estudio con la producción y la investigación, por lo que se destaca como un centro docente-productor. Está compuesta por facultades y centros, donde se desarrollan aplicaciones y servicios informáticos. Entre sus centros se encuentra el Centro de Soporte, el cual ofrece servicios de soporte y mantenimiento a productos desarrollados en la red de producción de esta institución y resuelve las incidencias tecnológicas que se le pueden presentar al cliente.

Actualmente dicho Centro utiliza el software de línea de asistencia ManageEngine ServiceDesk Plus para gestionar las incidencias, proceso estrechamente relacionado con la gestión de problemas, niveles de servicio, soluciones, cambios e inventario. Este proceso de gestión de incidencias se lleva a cabo de manera incorrecta, lo que trae consigo resultados no deseados y atenta contra los objetivos que persigue este Centro. El ignorar los Acuerdos de Niveles de Servicios (ANS), a la hora de su realización, es uno de los

factores que aumenta sus efectos negativos. Esto ocasiona incumplimiento en los tiempos de respuesta y resolución de las incidencias, dado que no se tienen bien claro las necesidades del cliente, actor principal en los contratos de servicios con el Centro.

Otra de las causas que influye negativamente en la calidad del servicio que ofrece este Centro es: no definir correctamente el personal asignado para resolver las incidencias, lo que conlleva a que se consuma más en tiempo y recursos durante el proceso de su gestión. Los especialistas que utilizan la plataforma para dar solución a las incidencias se demoran en determinar su causa raíz y sus síntomas, dado que se ignoran los cambios realizados en las aplicaciones informáticas, lo cual causa descontento e insatisfacción por parte de los clientes. Constituye otra causa el desconocimiento de los niveles de satisfacción y las expectativas del consumidor para con el servicio de soporte técnico. Un freno para el avance del Centro lo representa el carácter privativo de la plataforma, dado que no se tiene acceso a su código fuente, lo que imposibilita adecuar sus funcionalidades a las necesidades existentes o las que puedan surgir en un futuro.

Por lo anteriormente expuesto se plantea como **problema a resolver** ¿Cómo contribuir con el proceso de gestión de incidencias de los servicios en el Centro de Soporte de la UCI?

Definiéndose como **objeto de estudio** de esta investigación el proceso de Gestión de Incidencias de la metodología ITIL 2011.

Enmarcándose como **campo de acción** la Gestión de Incidencias de la metodología ITIL 2011 en el Centro de Soporte de la UCI.

De acuerdo con la problemática planteada se propone como **objetivo general**: Desarrollar un módulo que contribuya con el proceso de Gestión de Incidencias de los servicios que presta el Centro de Soporte de la UCI, según la metodología ITIL 2011.

Del objetivo general planteado se derivan los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación y el estado del arte referente a los sistemas homólogos a la solución propuesta.
2. Realizar el análisis y diseño de la solución.
3. Implementar los componentes del diseño del módulo de gestión de incidencias.
4. Aplicar pruebas para comprobar las funcionalidades de la solución.

Para cumplir con el objetivo general enunciado se proponen las siguientes **tareas de investigación**:

1 Realización del marco teórico de la investigación.
2 Realización del análisis del estado del arte sobre los procesos de gestión de incidencias.
3 Realización del estudio de las herramientas, tecnologías y metodología a utilizar en el proceso de desarrollo.
4 Realización del análisis de los procesos y de las fases de ITIL 2011.
5 Definición de los requerimientos de la propuesta de solución.
6 Implementación de la solución propuesta.
7 Diseño de los casos de prueba de las funcionalidades implementadas.
8 Ejecución de las pruebas para la detección de errores.

La **idea a defender** para la presente investigación es: Con el desarrollo del módulo de Gestión de incidencia se contribuirá con los servicios que presta el Centro de Soporte de la UCI, según la metodología ITIL 2011.

Para resolver y dar cumplimiento a los objetivos y las tareas propuestas se emplearon los siguientes **métodos de investigación:**

**Teóricos:**

*Histórico–lógico*: Este método se utilizó para entender los antecedentes y tendencias actuales de la gestión de incidencia, así como otras temáticas estrechamente relacionadas con la misma a lo largo de la historia de la Informática.

*Analítico–sintético*: Este método se utilizó para analizar y comprender la teoría y documentación relacionada con la gestión de incidencias, lo cual permitió, extraer los elementos más importantes relacionados con el objeto de estudio.

**Empíricos:**

*Observación:* Este método permitió percibir cómo se desarrolla el proceso que constituye el objeto de estudio. Se utilizó para elaborar el diagnóstico acerca de la realización del módulo propuesto.

*Entrevista:* Este método se utilizó para entrevistar a especialistas del Centro de Soporte de la Universidad de Ciencias Informáticas, quienes aportaron elementos significativos a la investigación.

*Análisis de documentos:* Este método se utilizó con el objetivo de seleccionar la información necesaria para la investigación, a partir del estudio de documentos y diferentes bibliografías relacionadas con el tema a desarrollar.

El presente Trabajo de Diploma está constituido por tres capítulos estructurados de la siguiente forma:

**Capítulo 1 Fundamentación teórica del módulo de gestión de incidencias:** En este capítulo se precisan elementos teóricos que sustentan la investigación y se hace referencia a los conceptos fundamentales para permitir entender cualquier definición asociada a la comprensión de la misma. Se realiza un estudio del estado del arte de soluciones existentes que puedan dar respuesta al problema científico planteado. Se detalla la metodología de desarrollo de software a utilizar, así como las herramientas para su implementación.

**Capítulo 2 Propuesta del módulo de gestión de incidencias:** En este capítulo se especifican los requerimientos funcionales y no funcionales según las necesidades del cliente y a partir de esto se diseña la solución para darle cumplimiento a la situación problemática planteada.

**Capítulo 3 Codificación y pruebas del módulo de gestión de incidencias:** En este capítulo se exponen los elementos que se tuvieron en cuenta para realizar la implementación de la aplicación. Se definen y se aplican las pruebas a la solución para verificar que cumple con los requerimientos definidos por el cliente.

# Capítulo 1: Fundamentación teórica del módulo de gestión de incidencias

En este capítulo se definen términos y conceptos de vital importancia para el logro de un mayor entendimiento de la investigación. Se manifiestan los resultados del estado del arte de los diferentes Sistemas de Gestión de Incidencias en el mundo. Además, se incluyen las principales características de las tecnologías y herramientas, así como, la metodología de desarrollo a utilizar para la implementación de la solución.

## 1.1 Conceptos fundamentales

### Servicio

El término servicio será empleado a lo largo de la presente investigación por lo que se hace necesario conocer su definición. Según Van Jan Bon un servicio es “un medio para entregar valor a los clientes, facilitando los resultados que los clientes quieren asumir sin costes o riesgos específicos. Los resultados dependen de la realización de tareas y están sujetos a diversas restricciones. Los servicios mejoran el rendimiento y reducen el efecto de las restricciones, lo que aumenta la probabilidad de conseguir los resultados” (Bon 2005).

Stanton, Etzel y Walker, consideran a los servicios "como actividades identificables e intangibles que son el objeto principal de una transacción ideada para brindar a los clientes satisfacción de deseos o necesidades" (Stanton William, Etzel Michael, 2004) (en esta propuesta, cabe señalar que según los mencionados autores esta definición excluye a los servicios complementarios que apoyan la venta de bienes u otros servicios, pero sin que esto signifique subestimar su importancia).

Para Richard L. Sandhusen, "los servicios son actividades, beneficios o satisfacciones que se ofrecen en renta o a la venta, y que son esencialmente intangibles y no dan como resultado la propiedad de algo" (Sandhusen 2002).

Lamb, Hair y McDaniel tratan el término servicio como "el resultado de la aplicación de esfuerzos humanos o mecánicos a personas u objetos. Los servicios se refieren a un hecho, un desempeño o un esfuerzo que no es posible poseer físicamente" (Lamb Charles, Hair Joseph 2002).

En términos de marketing y de economía el servicio es considerado como un conjunto de actividades que se lleva a cabo internamente en una empresa para poder responder y satisfacer las necesidades de un

cliente. Es un bien, pero se diferencia porque siempre se consume en el momento en que es prestado (Definicion ABC 2013).

Se considera como servicio en esta investigación a “un medio para entregar valor a los clientes facilitándoles un resultado deseado sin la necesidad de que estos asuman los costes y riesgos específicos asociados” (OSIATIS 2006d).

**Incidencia**

En terminologías de TI, “una incidencia es cualquier comportamiento inesperado de un servicio que afecta de manera negativa a la calidad del mismo” (Kook 2013). También puede ser definida como “los sucesos que provocan la degradación o pérdida del funcionamiento normal de un servicio” (OverTI 2011).

El término “incidencia” es utilizado para hacer referencia a distintas situaciones. Puede referirse a un hecho que acontece mientras está ocurriendo un negocio u otra situación, relacionada con ello. En un contexto informático es asociado usualmente con cualquier mal funcionamiento de los sistemas de hardware y software. Se considera para esta investigación que una incidencia 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” (OSIATIS 2006c).

## 1.2 Biblioteca de la Infraestructura de Tecnología de Información

“Desarrollada a finales de 1980” (Strite 2012), a raíz de los elevados niveles de dependencia de las organizaciones con la informática para alcanzar sus objetivos corporativos, la Biblioteca de Infraestructura de Tecnologías de la Información (ITIL, por sus siglas en inglés) se ha convertido en el estándar mundial para la Gestión de Servicios Informáticos.

Constaba inicialmente con 10 libros centrales cubriendo las dos principales áreas de Soporte del Servicio y Prestación del Servicio. Estos libros centrales fueron más tarde soportados por 30 libros complementarios que cubrían una numerosa variedad de temas, desde el cableado hasta la gestión de la continuidad del negocio (OSIATIS 2006a).

La metodología ITIL, se basa en coleccionar una serie de “mejores prácticas” en diferentes sectores de actividad a nivel mundial, que se publican con el objetivo de lograr una gestión eficiente de la Infraestructura y los Servicios de TI.

ITIL se centra en prestar servicios de alta calidad con el objetivo de lograr mayor satisfacción por parte de los clientes a un costo manejable. Este parte de un enfoque estratégico basado en el triángulo procesos-personas-tecnología como muestra la Figura 1.

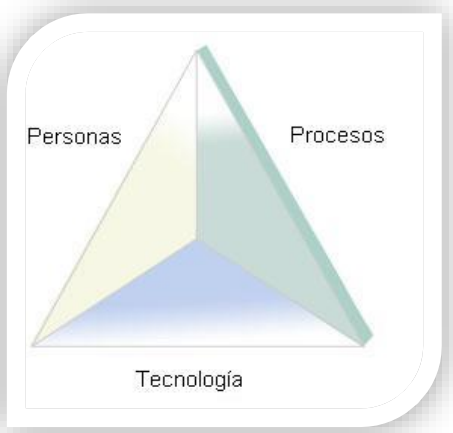



*Figura 1. Enfoque estratégico de ITIL* (García Gallardo 2012)

Es un modelo que ofrece la manera de gestionar los procesos, roles, actividades, así como sus interrelaciones, con el uso de las tecnologías para la satisfacción del cliente, usuario final del servicio de TI.

Principales beneficios que ofrece ITIL a las organizaciones (Espiral MS 2013):

- Mejorar la integración de TI con el negocio. Es fundamental contar con un enfoque de Gestión de Servicios de TI que le permita alinear estos servicios a los procesos de Negocio.
- Fortalecer líneas de comunicación entre el área de TI con las demás áreas de la organización. Enfocados a la Gestión de Servicios de TI, la comunicación es mucho más eficiente y se relaciona con los intereses de las áreas operativas.
- Reducir los costos de TI y mejora de la calidad del servicio. Los procesos de TI con mayor madurez generan mayor productividad y más calidad, lo cual reduce los costos y amplía los beneficios.
- Eliminar silos organizacionales y de conocimiento, implementando procesos integrados en toda el área de TI. La implementación exitosa de Mejores Prácticas de ITIL define un modelo de procesos

sustentado por roles y responsabilidades, generando una forma de trabajo basada en responsabilidades puntuales.

- Contar con un modelo de gobernabilidad y gestión de TI con el objetivo de tener información a través de controles y estructuras que aseguren que el área de TI está actuando como soporte a las estrategias del negocio.
- Mejorar la Gestión de Proveedores: A través de ITIL se definen los niveles de servicio que necesita cada componente de los Servicios de TI, y con esta información y con un proceso maduro de gestión de proveedores, las áreas de TI podrán realmente alinear no sólo el área de TI al negocio, sino también a sus proveedores.

ITIL está enfocada a todo el ciclo de vida del servicio y propone como ventaja la posibilidad de integración con otros estándares para reforzar áreas específicas.

### 1.2.1 **Integración de estándares**

Existen diferentes estándares que posibilitan la obtención de un mayor control de la gestión de servicios como lo son COBIT, ISO/IEC 20000 y CMMI-SVC. Es importante determinar y analizar sus peculiaridades y preferencias de usos, así como sus similitudes y diferencias respecto a la metodología ITIL 2011.

Cada estándar constituye un marco de trabajo para el gobierno de las TI, pues se debe analizar y seleccionar cual es el que más se adapta a las características de la empresa. Estos no siempre se ajustan el uno con el otro, pues fueron hechos por diferentes autores, en regiones y espacios de tiempos diferentes y con diversos propósitos. Aunque existan varios estándares que den solución a determinada problemática, se debe definir qué parte de cada modelo o estándar es el más adecuado a implementar en la empresa.

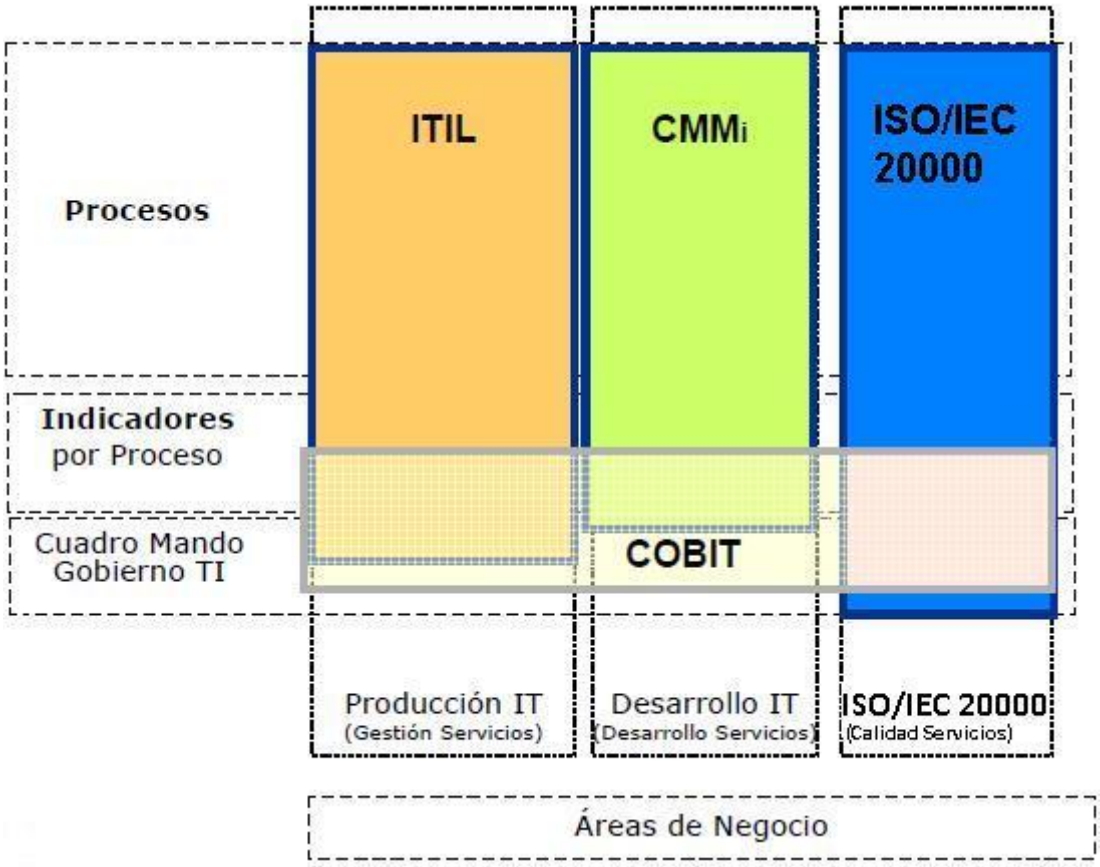



*Figura 2. Convivencia de Estándares* (García Gallardo 2012)

En la Figura 2 se muestra la convivencia de los estándares COBIT, ISO/IEC 20000, CMMI-SVC e ITIL en diferentes áreas del negocio. Integrarlos a todos resulta difícil pues están enfocados a partes diferentes de la gestión de servicios. CMMI generalmente se aplica al desarrollo del servicio o infraestructura en TI, mientras COBIT se enfoca en qué se requiere para lograr una administración y un control adecuado de TI. Por su parte el estándar ISO/IEC 20000 se orienta más a la calidad de los servicios e ITIL a la gestión de servicios de manera general.

Aunque estos estándares estén enfocados a partes diferentes de la gestión de servicio presentan áreas en común. A continuación se exponen los puntos de convergencia entre ITIL y los estándares COBIT, CMMI-SVC y la ISO/IEC 20000, lo cual evidencia la posibilidad de integración de ITIL con cada uno de estos estándares para un mejor gobierno de TI en la empresa.

**COBIT**

Objetivos de Control para la Información y Tecnologías (COBIT, por sus siglas en inglés). Es un conjunto de mejores prácticas para el gobierno de las Tecnologías de Información (TI), a través de dominios y un marco de procesos con una estructura lógica. Estas prácticas ayudan a optimizar la inversión en Tecnologías de la Información, asegurar la entrega de servicios y además proveer de un conjunto de indicadores para saber cuándo el rendimiento no se encuentra dentro de los límites deseables (Mitre 2010).

COBIT posee muchos puntos de coincidencia con ITIL. Tiene un alcance mayor, que abarca todo el espectro de actividades de TI, mientras ITIL se enfoca específicamente a la Gestión de Servicio. Estos modelos son complementarios a su vez, y pueden usarse juntos de acuerdo a las necesidades de la empresa. ITIL enfatizaría en un mejor logro de efectividad y eficiencia de los servicios TI, mientras COBIT estaría centrado en verificar la conformidad en cuanto a disponibilidad, rendimiento, eficiencia, riesgos asociados de los servicios con los objetivos y estrategias de la compañía.

**CMMI-SVC**

CMMI para servicio es un modelo de capacidad que ofrece el marco de referencia adecuado para administrar y entregar los servicios ofertados. Proporciona una guía para aplicar las mejores prácticas en una organización que provee servicios.

El modelo de CMMI-SVC cubre las actividades requeridas para establecer, entregar y administrar uno o varios servicios. Pretende mejorar la inversión realizada en los proyectos de mejora de procesos ampliando su aplicación a otras áreas de la organización; de tal manera que permita que la operación de los diferentes modelos operen de forma síncrona con otras iniciativas existentes (Consultores 2013).

El modelo ITIL se aplica al ciclo de vida de TI, enfocándose en los procesos operacionales (post-implementación de un determinado servicio o infraestructura de TI), mientras CMMI se aplica al desarrollo del servicio o infraestructura de TI (durante la implantación). Ambos modelos poseen un punto común donde se intersectan en la Gestión de Entrega. CMMI se centra en proporcionar la calidad en el desarrollo de software mientras que ITIL garantiza la explotación del producto software. Ambas metodologías son complementarias y en conjunto abarcan desde el desarrollo del software hasta la gestión del mantenimiento y servicios del mismo.

**ISO/IEC 20000**

La ISO/IEC 20000 es el primer estándar mundial creado específicamente para la Gestión de Servicios de TI, estableciendo métricas para medir los servicios de Tecnologías de Información (NYCE S.C. 2013).

Esta define los requerimientos para la entrega de Servicios de TI con la calidad aceptable para sus clientes. También introduce una cultura del servicio y proporciona la metodología para entregar servicios que satisfagan requerimientos del negocio. Se centra en los procesos para soportar la calidad de la provisión de los Servicios de TI en la actividad diaria.

ITIL está enfocado a establecer las mejores prácticas para la gestión de servicios. Proporciona lineamientos de gestión para adoptar roles, responsabilidades y actividades que realizan diferentes procesos para una entrega con calidad del servicio. La ISO/IEC 20000 se centra en indicar los requerimientos mínimos que se deben cumplir para establecer un sistema de GSTI, estos sirven como base para la realización de auditorías de certificación. ITIL ofrece un amplio espectro de recomendaciones y mejores prácticas para desarrollar e implementar un Sistema de Gestión basado en ISO/IEC 20000.

### 1.2.2 Gestión de incidencias

ITIL 2011 se basa en el ciclo de vida del servicio a la hora de mostrar los distintos procesos involucrados en la gestión de servicios. El ciclo de vida del servicio lo divide en 5 etapas:

1. Estrategia del servicio.
2. Diseño del servicio.
3. Transición del servicio.
4. Operación del servicio.
5. Mejora continua del servicio.


*Figura 3. Ciclo de vida de ITIL 2011*

En la Figura 3 se muestra el ciclo de vida de ITIL 2011 y los procesos que componen sus fases, también se muestran las metas de cada una de las fases.

Entre los procesos que posee ITIL 2011 se encuentra la Gestión de Incidencias dentro de la fase de Operación de Servicio, la cual cuenta con tres funciones definidas por el Centro de Servicio de ITIL. Esta tiene como objetivo procesar y establecer el buen comportamiento de un servicio con rapidez y eficacia, que disminuya el impacto negativo en el negocio.

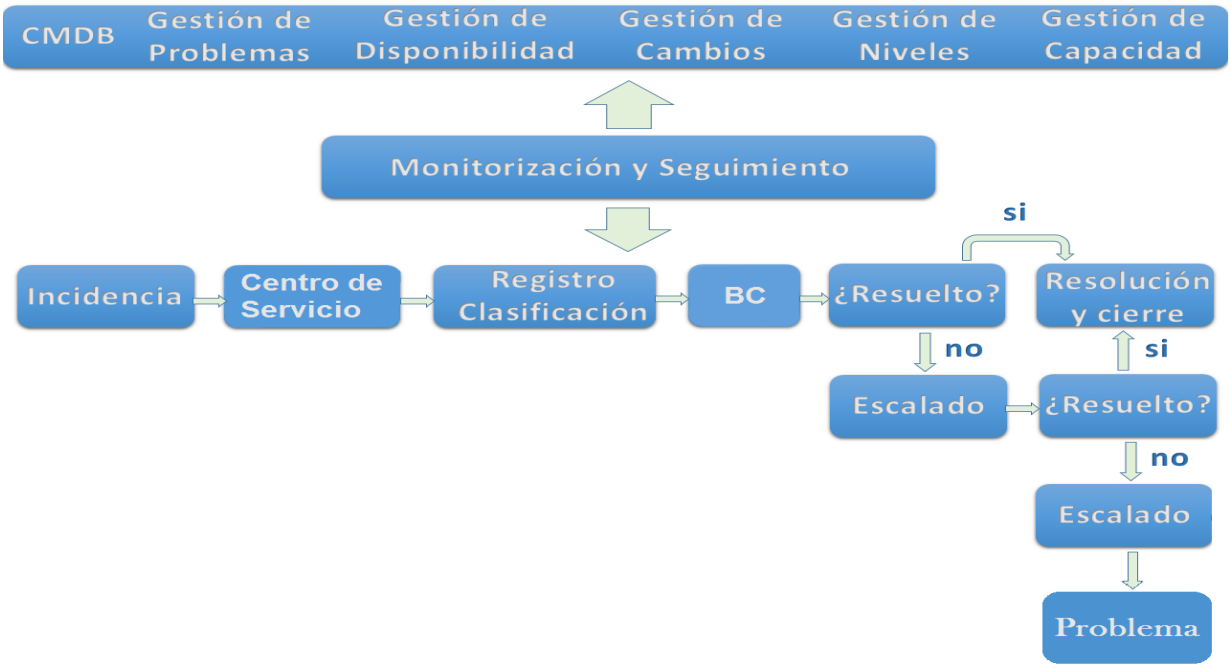



*Figura 4. Flujo de trabajo del proceso Gestión de Incidencias*

En la Figura 4 se expone el flujo de una incidencia de acuerdo al proceso de Gestión de Incidencias de ITIL 2011, así como la interrelación de este con otros procesos de TI como la Gestión de Cambios, la Gestión de Capacidad, la Gestión de Problemas, la Gestión de Disponibilidad, la Gestión de Niveles de Servicios y la Gestión de Configuración, este último se evidencia como la Base de Datos de la Gestión de Configuraciones (CMDB, según sus siglas en inglés).

El flujo de trabajo de la Gestión de Incidencias comienza desde que surge la incidencia y es introducida por el usuario en el Centro de Servicio, responsable directo de su gestión. El proceso Registrar es el primer

paso para su buen desempeño. Se debe tener en cuenta que las incidencias provienen de diferentes fuentes, como puede ser a través de un sitio *web*, una llamada telefónica o un correo electrónico al personal técnico.

Clasificación es el proceso siguiente que tiene como objetivo principal la recopilación de información necesaria para la resolución de la incidencia. Después le siguen Análisis, Resolución y Cierre, donde primeramente se verifica si el incidente tiene una respuesta semejante ya registrada en la base de conocimiento (BC), si no se encuentra una solución similar entonces se realiza un proceso de escala a un nivel superior para su investigación por expertos asignados.

Si los expertos asignados a resolver las incidencias no son capaces de encontrar su solución entonces se Escala a un nivel superior y pasa a ser objetivo del proceso de Gestión de Problemas. Si estos resuelven la incidencia se cambia su estado a Resuelto y si el cliente está satisfecho con la solución entonces se Cierra. La vida de la incidencia debe actualizarse durante todo su ciclo en las base de datos para disponer de información sobre su estado. Si es necesario después de Cerrada la incidencia se puede emitir una Petición de Cambio que se enviaría a la Gestión de Cambios.

Si no se encuentra una solución definitiva para la incidencia, se informa a la Gestión de Problemas para su estudio detallado. También se toma información registrada sobre la duración, el impacto y el desarrollo temporal de los incidentes para realizar informes sobre la disponibilidad real del sistema. Los ANS son accedidos por la Gestión de Incidencia en todo momento para determinar el curso de las acciones a realizar.

## 1.3 Sistemas de Gestión de Incidencias internacionales y nacionales

En este epígrafe se realiza un estudio de los diferentes Sistemas de Gestión de Incidencias internacionales y nacionales. Para su realización se utiliza una guía centrada en las necesidades del cliente como lo son:

1. El flujo de trabajo de la aplicación debe estar orientado al proceso de Gestión de Incidencias que propone ITIL 2011.
2. El carácter del software no debe ser propietario.
3. Tome en cuenta los ANS cuando se realiza la gestión de incidencias.
4. Tome en cuenta las expectativas del cliente para con el servicio.

**KMKey Help Desk:**

Knowledge Management Key (KMKey) Help Desk es un software de gestión de incidencias indicado para servicios de mantenimiento, ayuda al usuario y resolución de problemas en cualquier sector. Permite definir flujos de trabajo para abordar problemáticas derivadas de anomalías en servicios y maquinaria. La incidencia puede recibirse de forma automática (e-mail, entrada a través de una web, desde un dispositivo móvil) o bien ser abierta por el servicio de atención. Una vez en marcha seguirá el flujo diseñado por el cliente para su resolución (Earcon 2006).

Este sistema no se ajusta a las necesidades del cliente pues abarca otras áreas, además de la gestión de incidencias, que no son de interés. Está enfocado en minimizar los tiempos de respuesta y brindar servicios de mantenimiento, lo cual sacrifica el correcto seguimiento de la incidencia procesada, uno de los requerimientos principales exigidos por el cliente.

**NetSupport ServiceDesk:**

NetSupport ServiceDesk controla, organiza, administra y responde con facilidad a los problemas más difíciles de soporte. Está caracterizado por la funcionalidad central de administración de incidentes, problemas y solicitudes de cambio. “Está basado en web y completamente conforme a las normas de prácticas recomendadas de ITIL. Este proporciona a su helpdesk todas las herramientas necesarias para administrar las expectativas del cliente y minimiza el tiempo de inactividad del sistema” (NetSupport 2013).

Ofrece una interfaz intuitiva y un procesamiento del flujo de trabajo racionalizado para garantizar la utilización óptima del tiempo por parte de su equipo de soporte y no verse así cargados con excesivos deberes.

Este sistema tiene como principal desventaja que su licencia es privativa, lo cual incumple, con una de las principales exigencias del cliente, dado que no se puede acceder a su código fuente para adecuar sus funcionalidades a sus exigencias. No cuenta con un detallado registro histórico de incidencias lo cual afectaría a posibles soluciones homólogas en el futuro.

**JIRA Service Desk:**

JIRA Service Desk combina una experiencia de usuario intuitiva para los clientes con una potente gestión de ANS, colas personalizables, gestión de solicitudes automatizadas e informes en tiempo real (Atlassian 2013).

Posee una interfaz intuitiva y clara, que facilita una agradable experiencia al usuario. Los formularios son fáciles de crear, usan lenguaje natural y tienen llamadas a acciones simples y claras. Con este, puedes automatizar asignaciones de componentes y tipos de tareas, aplicar los ANS apropiados y poner todos los asuntos en el orden correcto en la cola de tareas del equipo.

Este sistema no se ajusta a las necesidades del Centro pues incumple dos de sus principales indicadores, posee un carácter privativo, que imposibilita las modificaciones en su código fuente para adecuar sus funcionalidades a las exigencias crecientes del cliente. No soporta el flujo adecuado que brinda ITIL 2011 para la gestión de incidencias.

**AvilaQuid:**

AvilaQuid es una aplicación *web* (desarrollada en la provincia de Ciego de Ávila por la empresa DESOFT), destinada a la automatización de la gestión de incidencias. Satisface atractiva y rápidamente los objetivos para los cuales fue destinada, lo cual facilita el seguimiento de todas las incidencias generadas por los clientes a través de toda su entidad o grupo de entidades interconectadas en una Intranet, basándose principalmente en la agrupación de las incidencias en función de las Unidades Organizativas que intervienen en su gestión. En estos momentos se encuentra en uso en varias empresas e instituciones de nuestro país, por ejemplo, ETECSA, Correos de Cuba, Inmunología Molecular, DESOFT y Asambleas del Poder Popular.

Este sistema no se adecua a las exigencias del cliente, no soporta los procesos de ITIL 2011, uno de los indicadores de más interés dentro de las expectativas del cliente.

**CuCERT:**

CuCERT es el sitio de la Oficina de Seguridad para las Redes Informáticas (OSRI), entidad nacional adscripta al Ministerio de la Informática y las Comunicaciones que lleva a cabo la prevención, evaluación, aviso, investigación y respuesta a las acciones, tanto internas como externas, que afecten el normal funcionamiento de las TI de Cuba. Su misión es la de prevenir y responder a los incidentes computacionales en el país. Dentro de sus funciones se encuentran recibir los reportes y responder los incidentes computacionales que se presenten en el país. También proporciona informes acerca de vulnerabilidades y amenazas sobre sistemas cubanos. Educar a la comunidad en general sobre temas de seguridad informática así como la detección con rapidez de las incidencias computacionales son otras de sus metas.

Este sistema se desajusta a los requerimientos definidos por el cliente pues no gestiona incidencias técnicas y no soporta los procesos de ITIL 2011.

**ManageEngine ServiceDesk Plus:**

ServiceDesk Plus integra las solicitudes y activos de TI ayudando a manejar estos de manera efectiva. Implementa las mejores prácticas de ITIL. ServiceDesk Plus es un software altamente personalizable de soporte técnico fácil de implementar (Ingeniería Dric S.A. de C.V. 2013).

Entre sus características destacan (Ingeniería Dric 2014):

- Gestión de incidencias y solicitudes.
- Gestión de configuraciones.
- Gestión de problemas.
- Gestión de cambios.
- Catálogo de servicios.
- CMDB.
- Inventario automático de *hardware* y *software*.
- Gestión de contratos, mantenimientos y garantías.
- Portal de Autoservicio para usuarios finales.
- Base de conocimientos.
- Sistema de ANS, reglas de negocio y flujos de trabajo configurables.
- Interfaz y reportes personalizables.
- Avisos, alertas y notificaciones automáticas.

Este sistema es actualmente utilizado por el Centro de Soporte de la UCI para gestionar incidencias, problemas, soluciones, cambios e inventarios. Está desarrollado en una plataforma de *software* propietario, por lo que representa una solución no factible dado que no se puede modificar su código fuente para adecuarlo a las necesidades crecientes de dicho Centro. No tiene en cuenta los ANS establecidos con el cliente y no resulta sencillo determinar la causa raíz y los síntomas de un problema determinado. Además, no permite tomar en cuenta el grado de satisfacción del cliente o sus expectativas para con el servicio de soporte.

## 1.4 Análisis de los sistemas homólogos

Después de realizar un análisis de los sistemas homólogos internacionales y nacionales que gestionan incidencias, se decide implementar un nuevo sistema, pues ninguno de estos cumple con todos los requerimientos deseados por el cliente. Algunos sistemas por poseer un carácter privativo no resultan soluciones factibles dado que no se puede modificar su código fuente para adecuarlo a las necesidades crecientes del Centro de Soporte. Otros no soportan los procesos de ITIL 2011, o están enfocados para satisfacer necesidades de las organizaciones para las cuales fueron desarrollados. En la Tabla 1 se observa de manera general el análisis realizado a los sistemas según las necesidades del cliente, la X denota cuando las aplicaciones cumplen con estas.

*Tabla 1 Estudio de homólogos*

| Sistemas de Gestión de Incidencias | Flujo basado en ITIL 2011 | *Software* libre | ANS | Expectativas del cliente |
|---|---|---|---|---|
| KMKey Help Desk | | X | X | |
| NetSupport ServiceDesk | X | | X | X |
| JIRA Service Desk | | | X | |
| AvilaQuid | | X | | X |
| CuCERT | | X | | |
| ManageEngine ServiceDesk Plus | X | | | |

## 1.5 Metodología, herramientas y tecnologías

Las herramientas y tecnologías, así como la metodología de desarrollo que se exponen a continuación son las definidas por el cliente de acuerdo a las necesidades de su empresa. El equipo de desarrollo es pequeño y el cliente forma parte de este. Los desarrolladores no tienen mucha experiencia en el uso de estas herramientas, por lo que es necesario realizar un profundo estudio de sus características y empleo para lograr su correcta utilización en la implementación del módulo.

### 1.5.1 **Metodología de desarrollo XP**

La Programación Extrema (XP, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (Sevilla 2010).

Esta metodología está pensada para proyectos con pocos desarrolladores, donde la comunicación, entre programadores, líderes de proyecto y clientes, es un punto de vital importancia. Los clientes forman parte del equipo de desarrollo. “XP utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido” (Pressman 2005). Contiene un conjunto de reglas y prácticas que se desarrollan en el contexto de cuatro actividades fundamentales del marco de trabajo: planeación, diseño, codificación y prueba.

Según Pressman (Pressman 2005) el proceso de XP se describe como se ilustra en la siguiente figura.

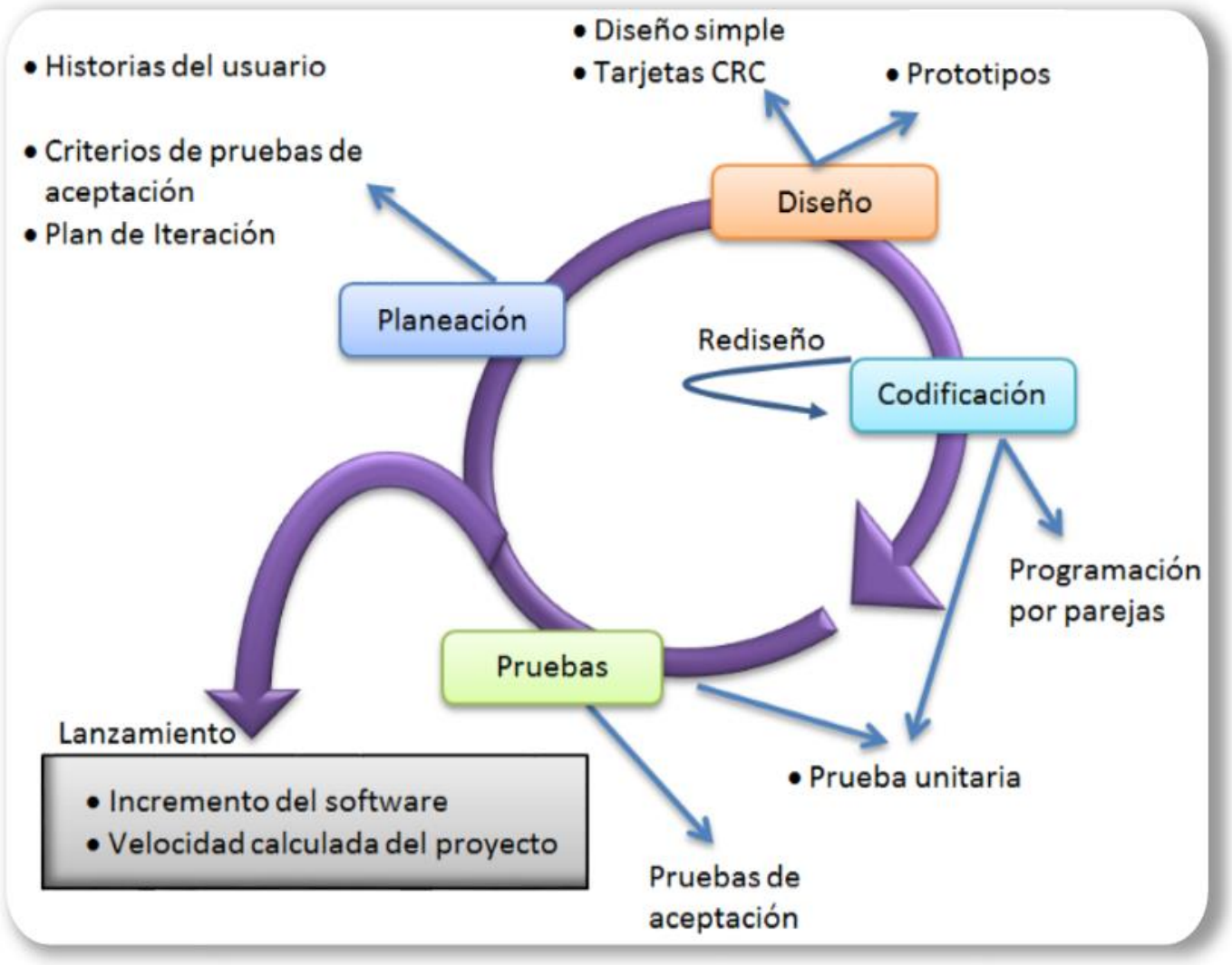



*Figura 5. Proceso de XP*

En la Figura 5 se muestra el proceso de XP y se observan algunas de las ideas y tareas claves asociadas con cada actividad del marco de trabajo: Planeación, Diseño, Codificación, Pruebas.

### 1.5.2 **Framework Grails 2.1.1**

Grails es un *framework* de desarrollo para aplicaciones *web* dentro de la plataforma Java, utiliza el lenguaje de programación Groovy. El objetivo de Grails es brindar al usuario un entorno de alta productividad, extensible y fácil de utilizar, ofreciendo el balance adecuado entre consistencia y funcionalidades potentes

(Leandro Bertolami 2011). Está basado en proyectos de correcto funcionamiento, tales como: Spring, Hibernate y SiteMesh (Smith, 2009). Una aplicación desarrollada con Grails sigue la arquitectura Modelo-Vista-Controlador (MVC) (Koenig 2007). En general, este *framework* proporciona un marco de trabajo estable, robusto, sencillo de usar y de fácil mantenimiento.

### 1.5.3 **Lenguaje de programación Groovy 2.1**

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk[1]. La especificación JSR 241[2] se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java. Usa una sintaxis muy parecida a Java, y comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. El *bytecode*[3] generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM[4]), por tanto puede usarse directamente en cualquier aplicación Java. Groovy puede usarse de manera dinámica como un lenguaje de scripting (Koenig 2007).

### 1.5.4 **jQuery UI**

jQuery UI es una biblioteca de componentes para el *framework* jQuery que le añaden un conjunto de *plugins*, *widgets* y efectos visuales para la creación de aplicaciones *web*. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery (JQuery 2011). Esta biblioteca permite implementar componentes diversos, para generar interfaces de usuario en páginas web, además de otras funcionalidades básicas, para crear aplicaciones *web* enriquecidas (Alvarez 2010).

Es compatible con los navegadores y sus versiones posteriores Internet Explorer 6.0, Mozilla Firefox 3, Safari 3.1, Ópera 9.6 y Google Chrome (JQuery 2011). jQuery es un *framework* de JavaScript muy intuitivo que facilita el diseño web y presenta como ventaja la facilidad de uso respecto a sus competidores.

---

[1] Python, Ruby, Perl y Smalltalk son lenguajes de programación.
[2] JSR 241: La solicitud de especificación java 241: Lenguaje de programación groovy.
[3] Bytecode: Código intermedio entre el código fuente y el código máquina. Suele tratarse como un fichero binario que contiene un programa ejecutable similar a un módulo objeto.
[4] Una máquina virtual Java (en inglés Java Virtual Machine, JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

### 1.5.5 **Bootstrap 3.0**

Bootstrap es una colección de herramientas de *software* libre para la creación de sitios y aplicaciones *web*. Contiene plantillas de diseño basadas en HTML y CSS con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript. Fue desarrollado por Mark Otto y Jacbod Thornton de Twitter, como un *framework* para fomentar la consistencia a través de herramientas internas. Es de código abierto, además de ser compatible con la mayoría de los navegadores *web* (Cochran 2012).

Bootstrap es un *framework* diseñado para simplificar el proceso de creación de diseños *web*. Este guiará en la aplicación del conjunto necesario de buenas prácticas. Ofrece una serie de plantillas CSS y de ficheros JavaScript, con los cuales se consigue interfaces elegantes y que puedan ser visualizadas desde cualquier dispositivo a diferentes escalas y resoluciones. Se integra además con la biblioteca jQuery y está basado en herramientas actuales y potentes como CSS3 y HTML5.

### 1.5.6 **Apache Tomcat 7.0**

Apache Tomcat es un servidor *web* http y funciona como un contenedor de *servlets*. Es usado como servidor *web* autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Es el más utilizado a la hora de trabajar con Java en entornos *web*. Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java (Jcalderon 2008).

Este servidor *web* esta implementado a partir de las especificaciones de Servlet 3.0 y JavaServer Pages (JSP) 2.2, como ventajas ofrece mejoras para detectar y prevenir "fugas de memoria" en las aplicaciones *web*. Brinda además soporte para la inclusión de contenidos externos directamente en una aplicación *web*. Este servidor viene incluido en el IDE IntelliJ IDEA.

### 1.5.7 **IntelliJ IDEA 12.1.3**

IntelliJ IDEA - Inteligente *IDE* Java, se centra en la productividad de los desarrolladores proporcionando una sólida combinación de herramientas avanzadas. Un editor que reconoce Java, HTML / XHTML, XML / XSL, CSS, Ruby y JavaScript. No define un idioma y posee un avanzado código de seguridad (Software B.d.l. 2012).

IntelliJ IDEA soporta múltiples compiladores Java: javac, jikes, eclipse. Permite compilar proyectos usando el compilador favorito, o sea, denota como ventaja que no es necesario instalar otro *IDE* para java (Software

B.d.l. 2012).

En este trabajo se utilizará la versión 12.1.3, debido al aumento de la velocidad de ejecución y proyectos de diseño, y la reducción de uso de memoria con respecto a versiones anteriores.

### 1.5.8 **PostgreSQL 9.2.4**

PostgreSQL es un servidor de base de datos avanzado con un largo historial de desarrollo. Está disponible para una amplia variedad de plataformas (2ndQuadrant Ltd 2014). En el desarrollo del trabajo se utilizará en su versión 9.2.4. Este se ha ganado una excelente reputación debido a sus características innovadoras, integridad, seguridad y fiabilidad.

PostgreSQL tiene las siguientes características principales (2ndQuadrant Ltd 2014):

- Excelente cumplimiento del estándar SQL, siguiendo el último SQL: 2011.
- Arquitectura cliente-servidor con un amplio rango de drivers y clientes.
- Diseño de alta concurrencia donde los lectores y escritores no se bloquean.
- Altamente configurable y extensible para muchos tipos de aplicación.
- Excelente escalabilidad y rendimiento con características de ajustes extensas.
- Optimizador de consultas sofisticado, adecuado para inteligencia de negocios.
- Soporta totalmente el acceso y procedimientos de base de datos en Java, Python, Perl, PHP y otros lenguajes.
- Altamente confiable con características extensivas para durabilidad y alta disponibilidad.
- Tipos de datos avanzados como Información Geográfica, búsqueda completa de texto y más.
- PostgreSQL es altamente escalable tanto en la enorme cantidad de datos que puede manejar como en el número de usuarios concurrentes que puede acomodar.

### 1.5.9 **Visual Paradigm 8.0**

Visual Paradigm es una herramienta CASE para desarrollo de aplicaciones. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman 2002).

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un

menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Dentro de sus características se encuentran (Visual Paradigm International Ltd. 2013):

1. Soporte de UML versión 2.1.
2. Diagramas de Procesos de Negocio: Proceso, Decisión, Actor de negocio, Documento.
3. Modelado colaborativo con CVS y Subversion.
4. Ingeniería de ida y vuelta.
5. Ingeniería inversa: Código a modelo, código a diagrama.
6. Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
7. Generación de código: Modelo a código, diagrama a código.
8. Editor de Detalles de Casos de Uso: Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
9. Diagramas de flujo de datos.
10. Soporte ORM - Generación de objetos Java desde la base de datos.
11. Generación de bases de datos: Transformación de diagramas de Entidad-Relación en tablas de base de datos.
12. Ingeniería inversa de bases de datos: Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
13. Generador de informes para generación de documentación.
14. Distribución automática de diagramas: Reorganización de las figuras y conectores de los diagramas UML.
15. Importación y exportación de ficheros XMI.

Visual Paradigm posee más ventajas con respecto a otras herramientas similares, es multiplataforma, posee *plugins* como el SDE para IntelliJ IDEA, que permite la integración entre estas dos herramientas.

**Conclusiones parciales**

Las definiciones proporcionadas por ITIL 2011 de los conceptos incidencia y servicio proporcionaron un mejor entendimiento de la investigación. Se tuvieron en cuenta aspectos del estudio realizado a las soluciones similares que proporcionaron una base para el posterior diseño e implementación de la solución. El estudio del proceso Gestión de Incidencias que brinda ITIL 2011 permitió una mejor comprensión del flujo de trabajo de la aplicación. El análisis de la metodología de desarrollo XP permitió profundizar los conocimientos sobre su uso para guiar el proceso de desarrollo del *software*. El estudio de las herramientas y tecnologías definidas por el cliente para el desarrollo de la solución, el lenguaje de programación Groovy, los *frameworks* Grails, jQuery y Bootstrap, el servidor web Apache Tomcat, el entorno de desarrollo IntelliJ IDEA, la herramienta CASE Visual Paradigm y el servidor de base de datos PosgreSQL, posibilitó profundizar sobre el correcto empleo de las mismas para la posterior implementación.

## Capítulo 2: Propuesta del módulo de gestión de incidencias

En este capítulo se describe la propuesta de solución del módulo de gestión de incidencias, a través de la utilización de la metodología de desarrollo XP. Se especifican los requisitos funcionales y los no funcionales de la propuesta de solución a implementar, se tienen en cuenta sus restricciones y características. Se abarcaran dos de las cuatro actividades del marco de trabajo de XP: planeación y diseño.

### 2.1 Solución propuesta

La propuesta de solución a implementar incorpora las últimas y mejores prácticas recomendadas por ITIL 2011 para gestionar incidencias. Posibilitará una correcta gestión de incidencias y aportará beneficios como: mejorar la productividad de los usuarios, el cumplimiento de los ANS, mayor control y monitorización del proceso de gestión de incidencias y principalmente la satisfacción general de clientes y usuarios.

Entre sus funcionalidades más destacadas se encuentran:

- La utilización de una base de datos, donde se almacenarán todos los datos e información obtenida en gestiones de incidencias ya ocurridas para su posible reutilización futura.
- La disposición de un cuadro de mando (*dashboard,* por sus siglas en inglés) con informes, métricas e indicadores claves de desempeño adecuado y siempre actualizado. Lo cual permite el acceso al seguimiento del número de incidencias en cada estado en cualquier instante de tiempo.
- El envío automático de emails durante el ciclo de vida de las incidencias. Lo cual posibilitará los flujos automáticos de comunicación, que faciliten la coordinación entre todos los involucrados.
- La confección de reportes en formato *pdf*, donde se evidenciarán estadísticas de incidencias durante un intervalo de tiempo dado.
- La sincronización con los datos proporcionados en el directorio activo de la UCI. Esto permitirá una posible sincronización y consolidación con todos los datos de usuarios que se encuentren en este directorio activo.
- La monitorización de las actividades llevadas a cabo en el sistema por los usuarios en un instante de tiempo dado.

- Además de gestionar las incidencias la aplicación será capaz de administrar cada uno de sus componentes: usuario, estado, impacto, urgencia, nivel, prioridad, categoría, subcategoría, grupo, lugar asociado, modo de entrada, categoría de servicio y también adjuntará archivos para la mejor comprensión del incidente.
- Proporcionará la realización de un vínculo o asociación entre las incidencias, diferenciándose esta acción por su estado y obteniéndose como resultado la misma solución para cada una.
- Posibilitará el conocimiento de la satisfacción y expectativas de los usuarios para con el servicio técnico. Con esto podrá conocer la satisfacción real de los usuarios, lo cual originará oportunidades de mejora de una manera rápida y sencilla. Esto ayudará a ubicar la institución en un camino hacia la mejora continua y todo ello en un entorno amigable y sencillo.
- Garantizará la integración con la Gestión de Configuración, Gestión de Cambios y Entregas y Gestión de Problemas.

## 2.2 **Lista de Reservas del Producto (LRP)**

Según define la metodología XP, la LRP está compuesta por los requerimientos funcionales con los que debe contar la aplicación para satisfacer las necesidades requeridas por el cliente. La correcta definición de requerimientos constituye un paso significativo en el desarrollo de software.

### 2.2.1 **Requerimientos funcionales de la solución**

Sommerville define los requerimientos funcionales como: “declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares” (Sommerville 2005). Estos especifican la capacidad y condiciones que debe cumplir el sistema a implementar. A continuación se reflejan los requerimientos funcionales del software, los cuales serán descritos detalladamente en el epígrafe 2.3.2, mediante Historias de Usuarios, como propone la metodología XP en su primera fase.

Los requerimientos funcionales del software se muestran en la Tabla 2, estructurada de la siguiente manera:

**Ítem:** Identificador numérico secuencial que indica el orden del requerimiento.

**Descripción:** Nombre del requerimiento, con orden según la prioridad de implementación: alta,

media o baja.

**Estimación:** Estimación, de acuerdo a la experiencia del grupo de desarrollo, de cada uno de los requerimientos para su implementación por semanas. Una unidad de tiempo corresponde a una semana de trabajo de 40 horas, un día sería 0.2 unidades equivalente a 8 horas laborales.

*Tabla 2. Requerimientos funcionales*

| Ítem | Descripción | Estimación |
|---|---|---|
| **Prioridad: Alta** | | |
| 1 | Autenticar usuarios | 0.6 semanas |
| 2 | Manejar usuarios | 0.4 semanas |
| 3 | Visualizar los estados de las incidencias | 0.4 semanas |
| 4 | Gestionar el impacto de las incidencias | 0.4 semanas |
| 5 | Gestionar la urgencia de las incidencias | 0.4 semanas |
| 6 | Crear reportes de incidencias | 0.4 semanas |
| 7 | Crear incidencias | 0.2 semanas |
| 8 | Visualizar incidencias | 0.2 semanas |
| 9 | Modificar incidencias | 0.2 semanas |
| 10 | Eliminar incidencias | 0.2 semanas |
| 11 | Cerrar incidencias | 0.2 semanas |
| 12 | Filtrar por parámetros las incidencias | 0.4 semanas |
| 13 | Mostrar historial | 0.4 semanas |
| 14 | Gestionar la categoría de servicio | 0.4 semanas |
| **Prioridad: Media** | | |
| 15 | Permitir adjuntar archivos a las incidencias | 0.2 semanas |
| 16 | Gestionar grupo de usuarios | 0.4 semanas |
| 17 | Visualizar el nivel de las incidencias | 0.4 semanas |

| | | |
|---|---|---|
| 18 | Gestionar la categoría de las incidencias | 0.4 semanas |
| 19 | Gestionar la subcategoría de las incidencias | 0.4 semanas |
| 20 | Vincular las incidencias | 0.4 semanas |
| 21 | Gestionar el lugar asociado a las incidencias | 0.4 semanas |
| 22 | Notificar por correo electrónico a los técnicos | 0.2 semanas |
| 23 | Notificar por correo electrónico a los administradores | 0.2 semanas |
| 24 | Notificar por correo electrónico a los usuarios | 0.2 semanas |
| 25 | Escalar las incidencias a otro nivel | 0.2 semanas |
| 26 | Asociar las incidencias | 0.2 semanas |
| 27 | Mostrar estadísticas | 0.4 semanas |
| **Prioridad: Baja** | | |
| 28 | Gestionar el modo de entrada de las incidencias | 0.4 semanas |

### 2.2.2 **Aspectos no funcionales de la solución**

Según Sommerville los requerimientos no funcionales “Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente a penas se aplican a características o servicios individuales del sistema” (Sommerville 2005). Los requerimientos no funcionales son aquellos que no reflejan funcionalidades específicas que el sistema proporciona, sino propiedades emergentes como la fiabilidad, tiempo de respuestas, capacidad. A continuación se muestran los requerimientos no funcionales que debe cumplir la propuesta de solución:

**Usabilidad**

1. La propuesta de solución debe brindar un acceso fácil y rápido.
2. La interfaz debe ser sencilla y amigable de manera que potencie la comodidad del usuario para su trabajo, además de que las opciones más usadas presentarán vías rápidas y cómodas de invocarse.

**Diseño e implementación**

3. Se utilizará PostgreSQL en su versión 9.2.4 o superior como Sistema Gestor de Bases de Datos.
4. Se utilizará Groovy como Lenguaje de Programación y Grails como Framework Integrador.
5. Se utilizará IntelliJ Idea como IDE para el desarrollo de la propuesta de solución.
6. Se utilizará Visual Paradigm en su version 8.0 como herramienta de modelado.

**Funcionamiento**

***Software:***

Para el funcionamiento en la PC cliente será necesario Firefox 6 o superior, Google Chrome 16 o superior.

**Seguridad**

***Confidencialidad:***

7. El acceso al módulo propuesto, así como a su información, se encontrarán protegidos contra accesos no autorizados, con el uso de mecanismos de autenticación propios de la aplicación.
8. La autenticación de la propuesta de solución será la primera acción del usuario, quien deberá proporcionar su identificador de usuario y una contraseña, los cuales serán de uso exclusivo del propio usuario.
9. Las diferentes áreas de la propuesta de solución se encontrarán protegidas contra el acceso no autorizado, dado el uso de roles y grupos de usuarios.

***Integridad:***

10. La información podrá ser modificada solo por el personal autorizado.
11. Se harán validaciones de la información en el servidor contra ataques de inyección SQL.
12. Se protegerá la información entrada contra ataques CSRF (*Cross-site request forgery* o falsificación de petición en sitios cruzados).

**Interfaz de usuario**

13. Las interfaces de la propuesta de solución contendrán los datos de forma estructurada, permitiendo la interpretación correcta de la información.
14. La entrada incorrecta de datos será mostrada al usuario claramente, resaltando el campo

donde se encuentra el error y mostrando como título el detalle del mismo.

15. El diseño de la interfaz de la propuesta de solución responderá a la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.

## 2.3 **Planeación**

La actividad planeación comienza con la elaboración de historias de usuarios que describen las características y funcionalidades requeridas para el software a implementar. Estas historias son creadas por el cliente quien le asigna cierta prioridad de acuerdo al grado de importancia que representa para el negocio. Los miembros del equipo le asignan un costo medido en semanas de desarrollo, sosteniendo que cada historia debe estar dentro de un costo de tres semanas.

En esta actividad el cliente y el equipo de desarrollo acuerdan el compromiso básico (historias de usuarios a implementar para el próximo incremento, fecha de entrega). Después de un primer incremento (lanzamiento) del software se determina la velocidad del proyecto, para posibilitar una mejor estimación sobre posteriores fechas de entregas. Esto permite también comprobar si el compromiso es excesivo y poder realizar el cambio adecuado.

### 2.3.1 **Planificación del proyecto por roles**

En la Tabla 3 se muestran las personas que intervienen de forma directa en el proceso de gestión de incidencias y una breve descripción del papel que desempeñan en la propuesta de solución.

*Tabla 3. Descripción de los roles*

| Actores | Descripción |
|---|---|
| Administrador | El administrador es el tipo de usuario que posee los privilegios para configurar todo el módulo de gestión de incidencias, puede participar de manera directa en el proceso de resolución de las incidencias y monitorizar el desempeño de estas. |
| Técnico | El usuario con rol técnico es aquel que participa de manera directa en el proceso de gestión de las incidencias. Este usuario no puede realizar modificaciones en las configuraciones del sistema. |

| | |
|---|---|
| Usuario final | Este usuario posee los privilegios mínimos en la aplicación solo puede añadir incidencias, nutrirse de su proceso de resolución y evaluar sus soluciones. |

### 2.3.2 **Historias de usuario**

Las historias de usuario se utilizan para especificar los requisitos del software. Son realizadas por el cliente con el uso de un lenguaje no técnico. Se emplean para hacer las estimaciones de tiempo y para el plan de incremento y presiden las pruebas de aceptación. Su tratamiento es dinámico y flexible, en cualquier momento pueden ser reemplazadas, modificas, eliminadas o añadidas. Cada historia de usuario es lo bastante comprensible y delimitada para que pueda ser resuelta en pocas semanas (menos de tres).

Existen varias plantillas sugeridas con respecto a la información contenida en las historias de usuario, en esta investigación la estructura queda de la siguiente forma:

**Número:** Identificador de la historia de usuario por parte del equipo de desarrollo.

**Nombre:** Nombre descriptivo de la historia de usuario.

**Prioridad en Negocio:** Grado de prioridad que le asigna el cliente a la historia de usuario de acuerdo al nivel de importancia que representa para el negocio. Los valores que puede tomar son: alta, media o baja.

**Riesgo en Desarrollo:** Grado de complejidad que le asigna el equipo de desarrollo a la historia de usuario luego de analizarla. (Alto, Medio o Bajo).

**Puntos Estimados:** Unidades de tiempo estimadas de acuerdo a la experiencia del equipo de desarrollo para darle cumplimiento a la historia de usuario. Una unidad de tiempo equivale a una semana de trabajo de 40 horas, por lo que un día de trabajo se representaría por 0.2 unidades que sería el equivalente a 8 horas laborales.

**Puntos Reales:** Unidades de tiempo reales que el equipo de desarrollo necesitó para darle cumplimiento a la historia de usuario. Una unidad de tiempo equivale a una semana de trabajo de 40 horas, por lo que un día de trabajo se representaría por 0.2 unidades que sería el equivalente a 8 horas laborales.

**Iteración Asignada:** Número de la iteración en la cual será implementada la historia de usuario.

**Descripción:** Descripción sobre la funcionalidad que representa la historia de usuario.

**Observaciones:** Condiciones que deben tenerse en cuenta para el desarrollo de la funcionalidad.

A continuación se muestran cuatro historias de usuario que representan un alto nivel de importancia para el cliente, las demás pueden ser visualizadas en el Anexo 1.

*Tabla 4. Historia de usuario 1*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_1 | **Nombre Historia de Usuario:** Autenticar usuarios |
| **Modificación de Historia de Usuario:** Ninguna | **Iteración Asignada:** 1 |
| **Prioridad en Negocio:** Alta | **Puntos Estimados:** 0.6 semanas |
| **Riesgo en Desarrollo:** Alta | **Puntos Reales:** 0.6 semanas |
| **Descripción:** Para acceder a la aplicación y realizar cualquier operación, primeramente el usuario debe autenticarse. Cada usuario, en correspondencia con su rol, ejerce una función diferente dentro de la aplicación. | |
| **Observaciones:** Para acceder a la solución, todos los usuarios deben estar registrados en el directorio activo de la UCI, o ser usuarios locales. | |

*Tabla 5. Historia de usuario 2*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_2 | **Nombre Historia de Usuario:** Manejar usuarios |
| **Modificación de Historia de Usuario:** Ninguna | **Iteración Asignada:** 1 |
| **Prioridad en Negocio:** Alta | **Puntos Estimados:** 1 semana |
| **Riesgo en Desarrollo:** Alta | **Puntos Reales:** 1 semana |
| **Descripción:** El sistema debe permitir que el usuario autenticado con rol administrador pueda listar todos los usuarios y modificar sus datos. | |
| **Observaciones:** | |

*Tabla 6. Historia de usuario 3*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_20 | **Nombre Historia de Usuario:** Crear incidencias |
| **Modificación de Historia de Usuario:** Ninguna | **Iteración Asignada:** 2 |
| **Prioridad en Negocio:** Alta | **Puntos Estimados:** 0.2 semanas |
| **Riesgo en Desarrollo:** Media | **Puntos Reales:** 0.2 semanas |
| **Descripción:** Todos los usuarios pueden insertar incidencias en el sistema. | |
| **Observaciones:** | |

*Tabla 7. Historia de usuario 4*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_22 | **Nombre Historia de Usuario:** Visualizar incidencias |
| **Modificación de Historia de Usuario:** Ninguna | **Iteración Asignada:** 2 |
| **Prioridad en Negocio:** Alta | **Puntos Estimados:** 0.2 semanas |
| **Riesgo en Desarrollo:** Media | **Puntos Reales:** 0.2 semanas |
| **Descripción:** El sistema debe permitir a todos los usuarios visualizar incidencias. | |
| **Observaciones:** | |

### 2.3.3 **Tareas de ingeniería**

Después de obtenerse las historias de usuarios se genera el artefacto de la metodología XP denominado tareas de ingeniería. Una historia de usuario puede ser dividida en una o más tareas de ingeniería. Una tarea de ingeniería posee un lenguaje técnico, desglosa la funcionalidad que describe la historia de usuario y es asignada a los programadores para su posterior implementación. A continuación se muestran cuatro ejemplos de tareas de ingeniería, el resto pueden ser visualizadas en el Anexo 2.

La estructura que poseen las tareas de ingeniería es la siguiente:

**Número Tarea:** Es el identificador de la tarea de ingeniería.

**Número Historia de Usuario:** Es el identificador de la historia de usuario para la cual está dirigida.

**Nombre Tarea:** Es el nombre de la tarea de ingeniería.

**Tipo de Tarea:** Es el tipo de tarea de ingeniería, puede ser de investigación o de desarrollo, este último refiriéndose a una tarea de implementación.

**Puntos Estimados:** Unidades de tiempo estimadas dada la experiencia del equipo de desarrollo para darle cumplimiento a la tarea de ingeniería. Una unidad de tiempo equivale a una semana de trabajo de 40 horas, por lo que un día de trabajo se representaría por 0.2 unidades que sería el equivalente a 8 horas laborales.

**Fecha inicio:** Día que comenzó el desarrollo de la tarea. Tiene un formato DD-MM-AAAA, o sea día, mes, año.

**Fecha fin:** Día que finalizó el desarrollo de la tarea. Tiene un formato DD-MM-AAAA, o sea día, mes, año.

**Programador Responsable:** Nombre de los programadores responsables para la ejecución de la tarea de ingeniería.

**Descripción:** Descripción de la tarea de ingeniería.

*Tabla 8. Tarea de Ingeniería 1*

| **Tarea de Ingeniería** | |
|---|---|
| **Número Tarea:** 1.1 | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Investigar como autenticar usuarios en un dominio | |
| **Tipo de Tarea:** Investigación | **Puntos Estimados:** 0.2 semanas |
| **Fecha Inicio:** 17-02-2014 | **Fecha fin:** 17-02-2014 |
| **Programador Responsable:** Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Investigar como autenticar usuarios en el sistema que pertenezcan al directorio activo de la UCI. | |

*Tabla 9. Tarea de Ingeniería 2*

| **Tarea de Ingeniería** | |
|---|---|
| **Número Tarea:** 1.2 | **Número Historia de Usuario:** HU_1 |
| **Nombre Tarea:** Implementar la funcionalidad autenticar usuarios en un dominio | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 0.4 semanas |
| **Fecha Inicio:** 18-02-2014 | **Fecha fin:** 19-02-2014 |
| **Programador Responsable:** Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Implementar la funcionalidad autenticar usuarios en el sistema que pertenezcan al directorio activo de la UCI, o sea, que estén bajo contrato. Una vez que un usuario genere incidencias, pasa a ser usuario del sistema hasta que finalice su contrato con el Centro de Soporte. | |

*Tabla 10. Tarea de Ingeniería 3*

| **Tarea de Ingeniería** | |
|---|---|
| **Número Tarea:** 2.1 | **Número Historia de Usuario:** HU_2 |
| **Nombre Tarea:** Investigar cómo manejar usuarios | |
| **Tipo de Tarea:** Investigación | **Puntos Estimados:** 0.4 semanas |
| **Fecha Inicio:** 20-02-2014 | **Fecha fin:** 21-02-2014 |
| **Programador Responsable:** Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Investigar cómo se debe desarrollar el visualizar y modificar los usuarios en la propuesta de solución. | |

*Tabla 11. Tarea de Ingeniería 4*

| **Tarea de Ingeniería** | |
|---|---|
| **Número Tarea:** 2.2 | **Número Historia de Usuario:** HU_2 |
| **Nombre Tarea:** Implementar la funcionalidad modificar usuarios | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 0.2 semanas |

| | |
|---|---|
| **Fecha Inicio:** 24-02-2014 | **Fecha fin:** 24-02-2014 |
| **Programador Responsable:** Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Implementar la funcionalidad modificar usuario de la solución. | |

### 2.3.4 **Plan de duración de iteraciones**

Las historias de usuario que se seleccionan para cada entrega, son desarrolladas y probadas, se debe tener en cuenta el ciclo de iteración en que se encuentren y su orden. Este plan define qué historias de usuario serán implementadas para cada iteración y las fechas en que serán liberadas. A continuación se muestran las iteraciones y el número de las historias de usuario que se encuentran dentro de cada iteración planificada para este proyecto.

*Tabla 12. Plan de duración de iteraciones*

| Iteraciones | Historias de usuario | Duración total |
|---|---|---|
| 1 | ❖ Autenticar usuarios<br>❖ Manejar usuarios<br>❖ Gestionar grupo de usuarios | 1 semana y 2 días |
| 2 | ❖ Permitir adjuntar archivos a las incidencias<br>❖ Gestionar el estado de las incidencias<br>❖ Gestionar el modo de entrada de las incidencias<br>❖ Gestionar el nivel de las incidencia<br>❖ Gestionar el impacto de las incidencias<br>❖ Gestionar la urgencia de las incidencias<br>❖ Gestionar la prioridad de las incidencias<br>❖ Gestionar categoría de las incidencias<br>❖ Gestionar categoría de servicio<br>❖ Gestionar subcategoría de las incidencias<br>❖ Gestionar lugar asociado a las incidencias<br>❖ Escalar las incidencias a otro nivel<br>❖ Asociar incidencias<br>❖ Vincular incidencias<br>❖ Crear incidencias | 5 semanas y 3 días |

| | | |
|---|---|---|
| | ❖ Visualizar incidencias<br>❖ Modificar incidencias<br>❖ Cerrar incidencias<br>❖ Eliminar incidencias | |
| 3 | ❖ Notificar por correo electrónico a los técnicos<br>❖ Notificar por correo electrónico a los administradores<br>❖ Notificar por correo electrónico a los usuarios<br>❖ Filtrar por parámetros las incidencias<br>❖ Mostrar estadísticas<br>❖ Mostrar historial<br>❖ Crear reportes de incidencias | 2 semanas |
| **Duración total** | | 9 semanas |

### 2.3.5 **Plan de entregas**

Este artefacto permite organizar el tiempo de las entregas, marca el final de cada una de las iteraciones y facilita a los programadores una guía para el desarrollo en tiempo de la solución. En la Tabla 13 se muestran las entregas acordadas con el cliente. Son tres que se dividen en versiones, en cada una de ellas se implementarán las funcionalidades definidas en las iteraciones acordadas.

*Tabla 13. Plan de entregas de las iteraciones*

| Artefacto | Iteración | Entrega |
|---|---|---|
| SGI 1.0 | Final de la primera iteración | Del 3-4 marzo |
| SGI 1.1 | Final de la segunda iteración | Del 15-16 abril |
| SGI 1.2 | Final de la tercera iteración | Del 1-2 mayo |

## 2.4 Diseño de la propuesta de solución

Después de planificada la realización de la propuesta de solución se procede al diseño de la misma, se definen los patrones a utilizar. En XP la actividad del diseño sigue, de una manera rigurosa, el principio de mantenerse simple. Se prefiere un diseño simple respecto a una representación más compleja. Este hace uso de las tarjetas CRC (clase-responsabilidad-colaborador), definidas en el epígrafe 2.4.2, para pensar en el software en un contexto orientado a objetos, donde se identifican y organizan las clases orientadas a objetos más relevantes para el incremento de la propuesta de solución.

### 2.4.1 Modelo conceptual

El modelo conceptual permitirá identificar y explicar los conceptos significativos en el dominio del problema y evidenciará las relaciones existentes entre ellos. En la Figura 6 se expone el modelo conceptual del software a implementar.

En este diagrama se muestran los diferentes tipos de usuarios de la propuesta de solución de acuerdo a sus responsabilidades (administrador, técnico y usuario) y las diferentes acciones que les son permitidas ejecutar en el entorno. El administrador puede gestionar las incidencias y sus componentes. Además de generar y visualizar incidencias, como un usuario final de la aplicación, puede resolverlas, asociarlas, vincularlas y cerrarlas. Dentro de las funcionalidades que puede realizar un técnico se encuentran: generar, visualizar, modificar, vincular, asociar y resolver las incidencias. Todos los usuarios al autenticarse en la aplicación generan una sesión y a su vez dejan las trazas de sus acciones, esto se ilustra en la relación existente entre usuario-sesión y usuario-histórico. La imagen evidencia todos los componentes que conforman una incidencia como lo son: prioridad, nivel, categoría, subcategoría, urgencia, modo, impacto, estado, categoría de servicio y grupo.

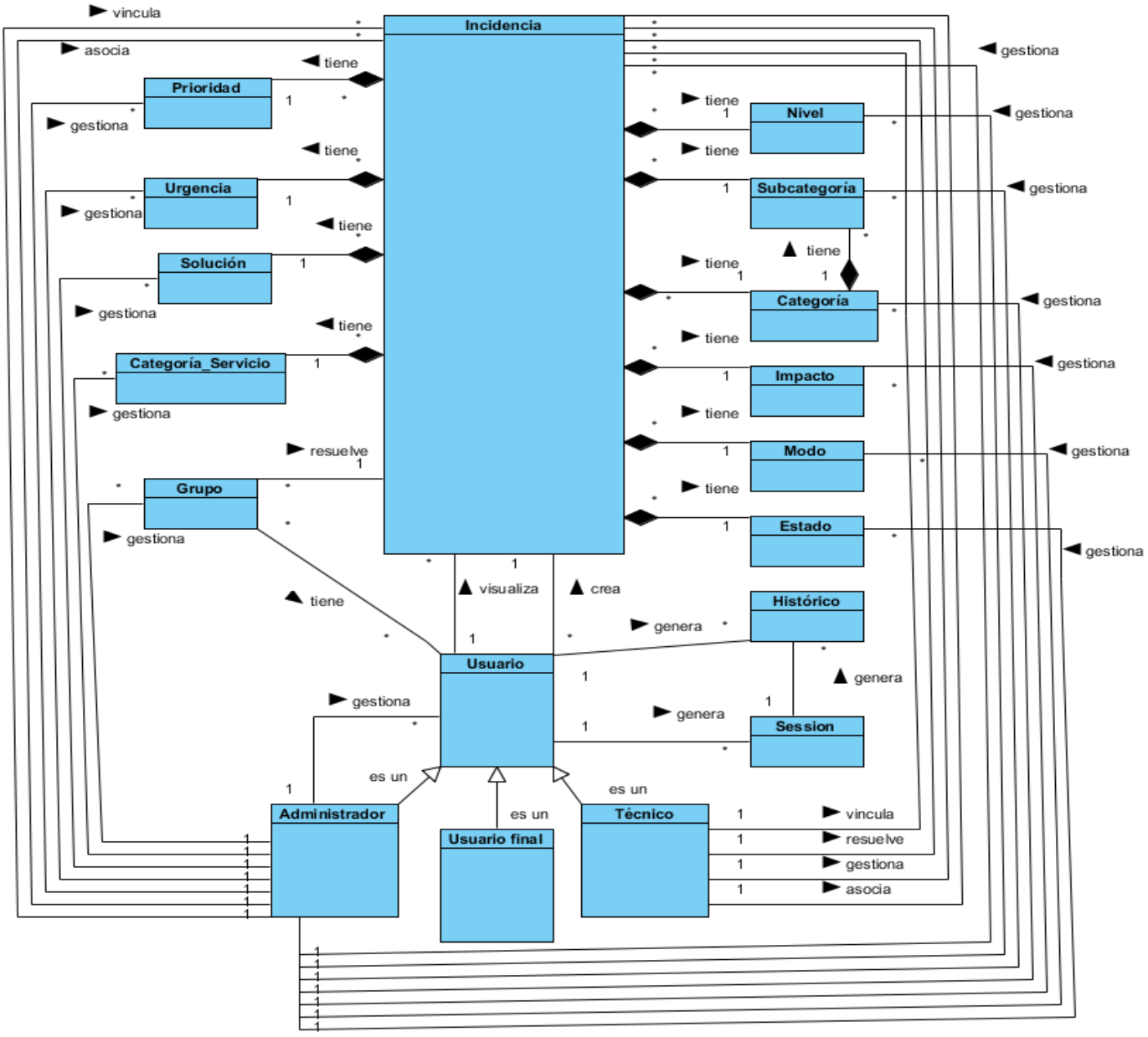



*Figura 6. Modelo conceptual de la propuesta de solución*

### 2.4.2 **Tarjetas CRC de la propuesta de solución**

El uso de las tarjetas CRC permite al programador centrarse y apreciar el desarrollo orientado a objetos. Estas representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte superior de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

Una clase puede ser cualquier persona, evento, concepto o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. A continuación se muestran cuatro tarjetas CRC que representan funcionalidades de prioridad en la propuesta de solución, el resto puede visualizarse en el Anexo 3.

*Tabla 14. Tarjeta CRC 1*

| **Tarjeta CRC** | |
|---|---|
| **Clase:** Usuario | |
| **Responsabilidades** | **Colaboraciones** |
| • Es la encargada de almacenar los atributos que requiere un usuario.<br>❖ String username<br>❖ String password<br>❖ String correo<br>❖ String nombre<br>❖ String apellido<br>❖ boolean enabled<br>❖ boolean accountExpired<br>❖ boolean accountLocked<br>❖ boolean passwordExpired | |

*Tabla 15. Tarjeta CRC 2*

| **Tarjeta CRC** | |
|---|---|
| **Clase:** UsuarioController | |
| **Responsabilidades** | **Colaboraciones** |
| Es la encargada de :<br>❖ anadirUsuario() | UsuarioService<br>SpringSecurityService |

| | |
|---|---|
| ❖ modificarUsuario(Usuario usuario)<br>❖ eliminarUsuario(Usuario usuario)<br>❖ mostrarUsuario() | Usuario |

*Tabla 16. Tarjeta CRC 3*

| **Tarjeta CRC** | |
|---|---|
| **Clase:** Incidencia | |
| **Responsabilidades** | **Colaboraciones** |
| • Es la encargada de almacenar los atributos que requiere una incidencia.<br>❖ int id_incidencia<br>❖ String asunto<br>❖ int id_autor<br>❖ int id_impacto<br>❖ int id_modo<br>❖ String descripción<br>❖ String detalles_impacto<br>❖ int id_fecha_creacion<br>❖ String archivo_adjunto<br>❖ int id_area<br>❖ int id_tecnico_asociado<br>❖ int id_categoria<br>❖ int id_subcategoria<br>❖ int id_categoria_servicio<br>❖ int id_grupo<br>❖ int id_estado<br>❖ int id_fecha_cierre<br>❖ int id_fecha_resuelta<br>❖ int id_nivel | |

| | |
|---|---|
| ❖ int id_urgencia<br>❖ String solucion | |

*Tabla 17. Tarjeta CRC 4*

| **Tarjeta CRC** | |
|---|---|
| **Clase:** IncidenciaController | |
| **Responsabilidades** | **Colaboraciones** |
| Es la encargada de :<br>❖ anadirIncidencia()<br>❖ modificarIncidencia(Incidencia incidencia)<br>❖ mostrarIncidencia()<br>❖ mostrarHistorico()<br>❖ notificarCorreo(Usuario usuario)<br>❖ escalarIncidencia()<br>❖ combinarIncidencia()<br>❖ cerrarIncidencia()<br>❖ filtrarIncidencia()<br>❖ crearReporte() | Incidencia<br>UsuarioService<br>SpringSecurityService<br>Historico<br>Usuario |

### 2.4.3 **Patrones de diseño**

En el año 1979 el arquitecto Christopher Alexander escribió un libro titulado “The Timeless Way of Building” donde establecía el siguiente concepto de patrones de diseño y es el más adecuado para la investigación: “Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software” (Alexander 1979). En otras palabras, proporcionan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. De los patrones se deben tener en cuenta su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos

y beneficios).

**Patrones GRASP**

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés) son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. “Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones” (Larman 1999).

En el diseño de la propuesta de solución se utilizarán los patrones: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador.

**Patrón Experto:** Es el principio básico de asignación de responsabilidades en diseño Orientado a Objetos. Se encarga de “asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad” (Larman 1999).

El patrón Experto se evidencia en la propuesta de solución, por ejemplo en la clase Usuario, encargada de almacenar todos los atributos que puede contener un usuario, como se visualiza en la Figura 7.

Usuario | UsuarioController | Usuario Views | Tests

```
class Usuario {
    static hasMany = [incidencias: Incidencia, grupos: Grupo]
    transient springSecurityService

    String username
    String password
    String correo
    String nombre
    String apellido
    boolean enabled
    boolean accountExpired
    boolean accountLocked
    boolean passwordExpired

    static transients = ['springSecurityService']
```

*Figura 7. Patrón Experto*

**Patrón Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Es el responsable de asignarle

a la clase B la responsabilidad de crear una instancia de clase A, o sea que B es un creador de los objetos A (Larman 1999).

Este patrón se evidencia en la propuesta de solución, por ejemplo la clase controladora IncidenciaController, que se encarga realizar funcionalidades como crear, modificar, visualizar instancias de la clase del dominio Incidencia, como se ilustra en la Figura 8.

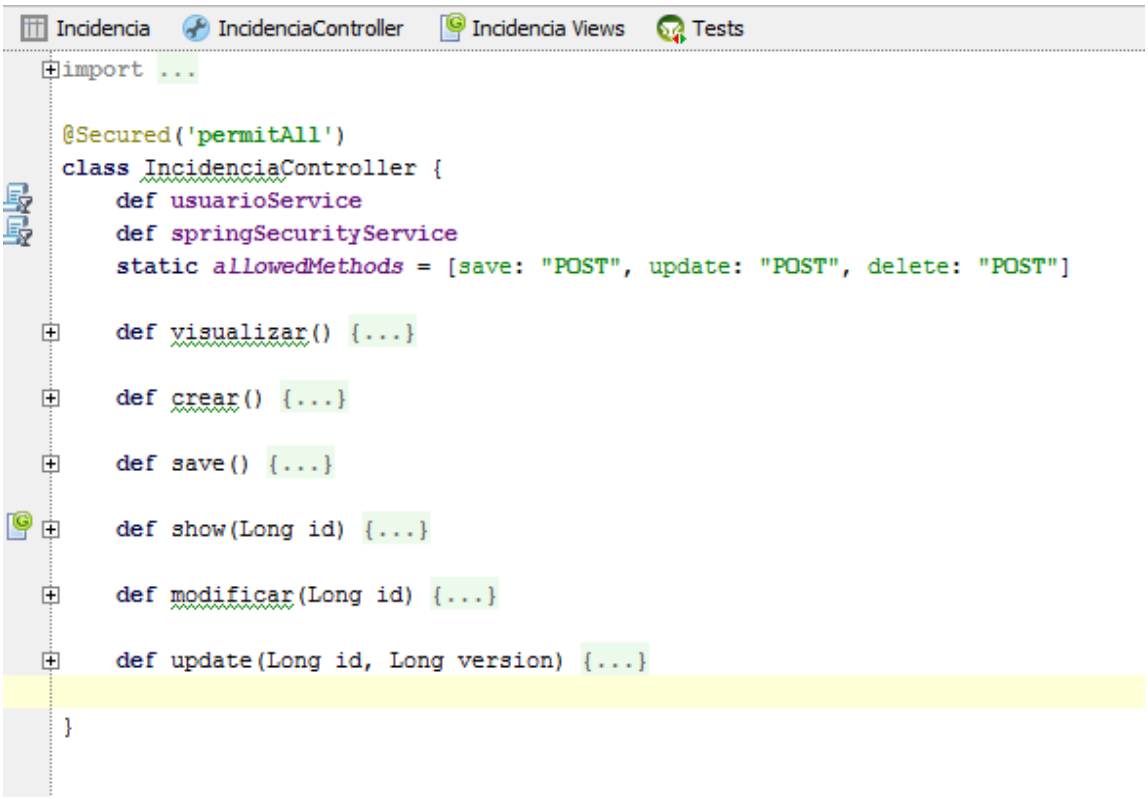



*Figura 8. Patrón Creador*

**Patrón Bajo Acoplamiento:** Es la idea de tener las clases lo menos fusionadas entre sí. "De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases" (Grosso 2011).

Este patrón también se evidencia en la propuesta de solución, por ejemplo la relación que existe entre las clases public.grupo y grupo_usuario, public.usuario y grupo_usuario, como se ilustra en la Figura 9.

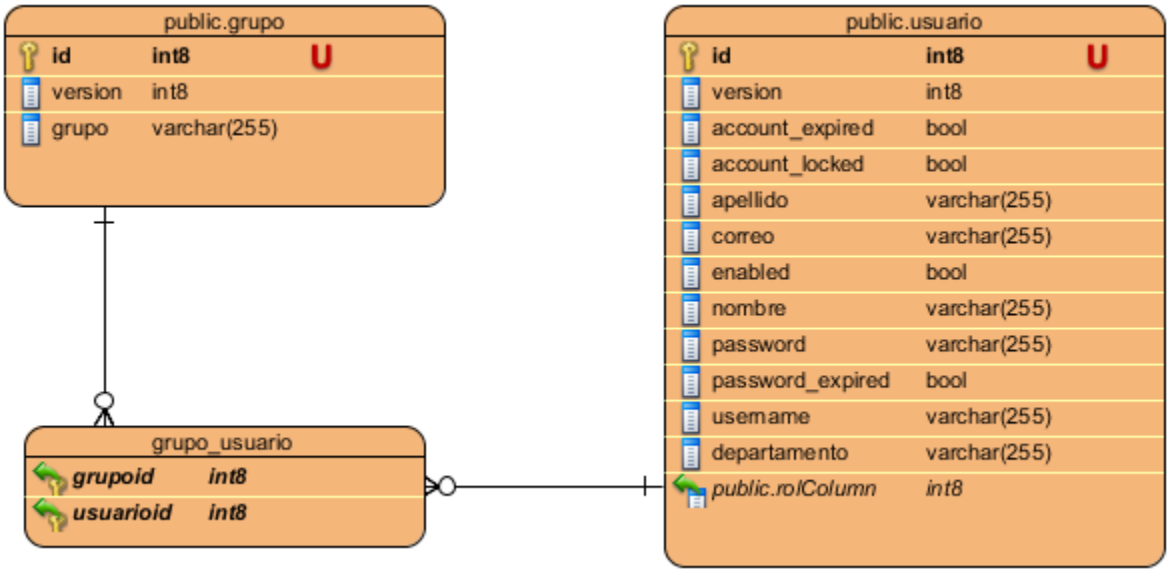



*Figura 9. Patrón Bajo Acoplamiento*

**Patrón Alta Cohesión:** "En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están, las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme" (Larman 1999).

Un ejemplo claro de este patrón en la propuesta de solución se muestra en la Figura 9, donde public.usuario y public.grupo (clases de la aplicación) colaboran para crear grupos de usuarios, o sea, colaboran con grupo_usuario.

**Patrón Controlador:** Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y los envía a las distintas clases según el método llamado. Sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control (Hernández 2010).

Este patrón se evidencia en la propuesta de solución con cada clase controladora, o sea, las que poseen en su nombre la terminación controller.

### 2.4.4 **Patrón arquitectónico**

"Los patrones arquitectónicos sobre aspectos fundamentales de la estructura de un software, especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes" (Buschmann 1996).

**Patrón Modelo-Vista-Controlador** (MVC, por sus siglas en inglés)**:** Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador realiza la gestión de eventos, usualmente acciones del usuario. En la Figura 10 se ilustran sus niveles, este patrón es el utilizado por el *framework* Grails.

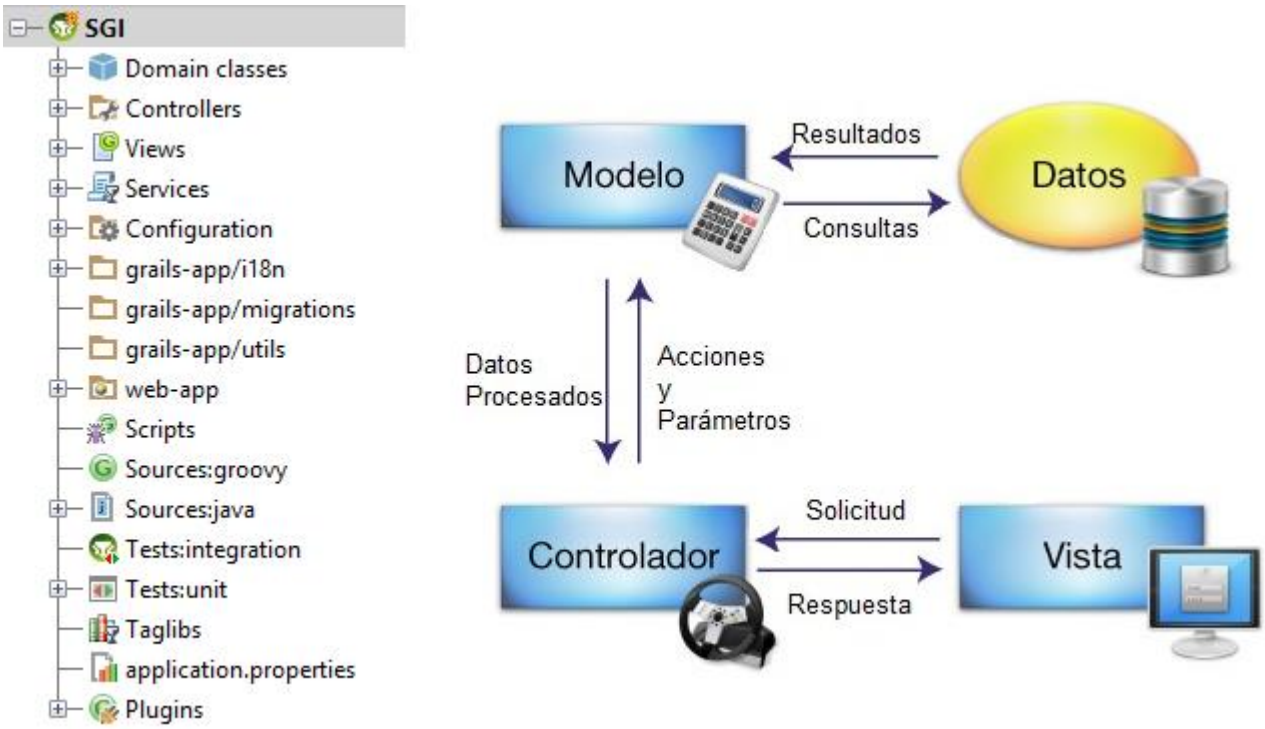



*Figura 10. Estructura del MVC*

En la Figura 10 se muestran las relaciones existentes entre los componentes que conforman el patrón de arquitectura de software MVC, los cuales son:

**Modelo:** Esta es la representación específica de la Información con la cual la solución opera, o sea, la lógica del negocio.

**Vista:** Este presenta el modelo en un formato adecuado para interactuar, es un elemento de interfaz de usuario.

**Controlador:** Interpreta las acciones que el usuario realiza en la aplicación e informa al modelo y/o a la vista para que cambien según resulte apropiado.

### 2.4.5 **Modelo de base de datos**

Después de realizado el diseño de la propuesta de solución se modela la base de datos. El modelo de datos es una abstracción que permite visualizar las entidades y relaciones de una base de datos. En la Figura 11 se muestra el diagrama relacional donde se ilustran las entidades con sus características y relaciones persistentes del modelo de datos. Se evidencian los componentes de una incidencia: prioridad, nivel, categoría, subcategoría, urgencia, modo, impacto, estado, categoría por servicio y grupo, y su estrecho vínculo. Se visualizan además las relaciones de usuario con las entidades grupo, sesión, histórico y rol.

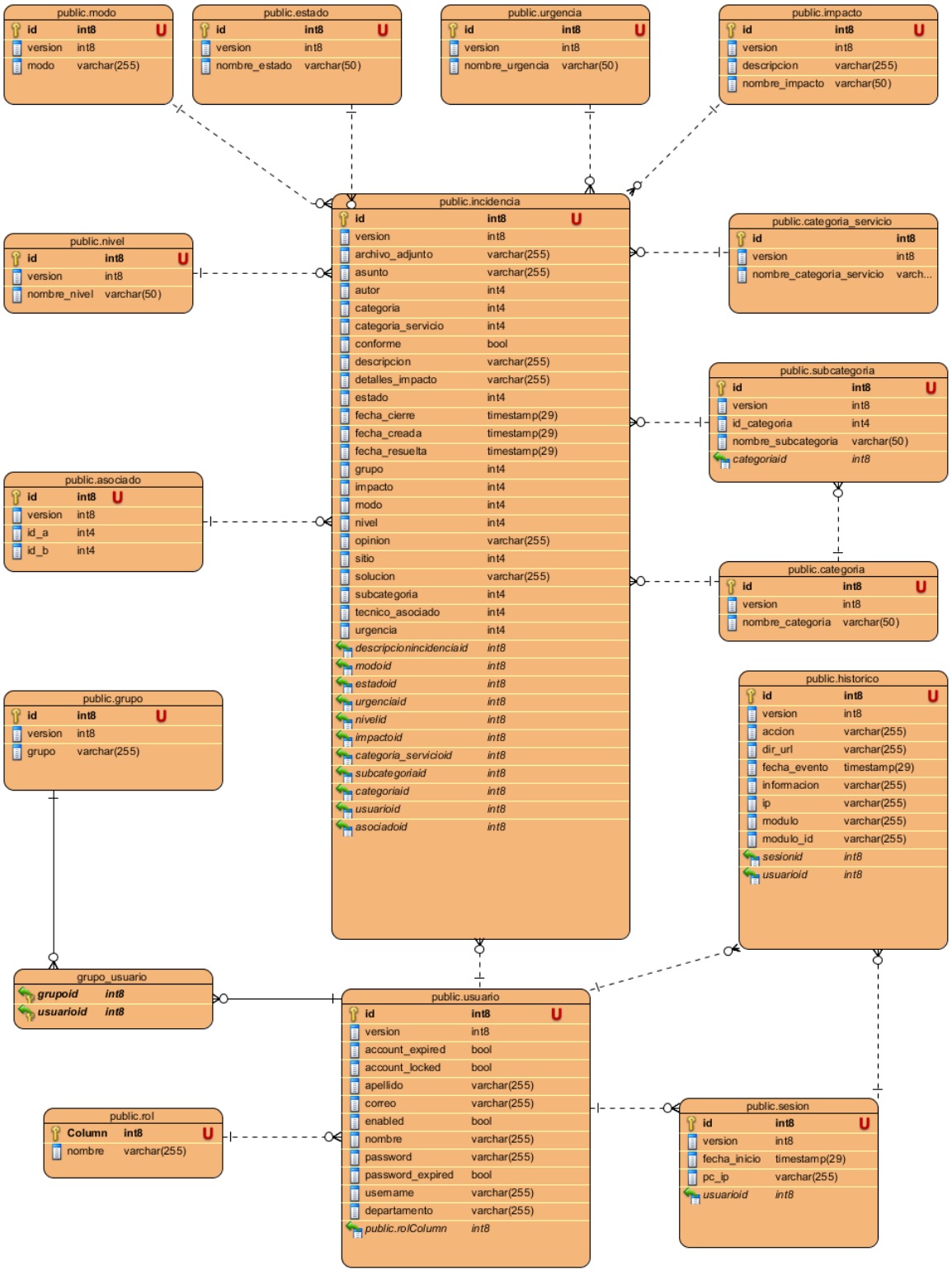



*Figura 11. Modelo de Base de Datos*

**Conclusiones parciales**

La metodología de desarrollo XP sirvió de guía para el proceso de desarrollo del software. La identificación de los roles y actores que intervienen en la propuesta de solución proporcionó la restricción del acceso a las funcionalidades del sistema. La elaboración de las historias de usuarios posibilitó la descripción de los requerimientos del software y sus niveles de prioridad. Se concibió el plan de iteraciones y el orden de implementación de las historias de usuarios, dado sus niveles de prioridad, lo que permitió establecer un compromiso básico con el cliente.

Las tareas de ingeniería posibilitaron al programador una mejor comprensión de las funcionalidades a implementar. Las clases del software fueron diseñadas a través de las tarjetas CRC, lo cual ayudó a centrarse en un desarrollo orientado a objeto e identificar las principales clases a desarrollar. Los patrones de diseño utilizados simplificaron el desarrollo de la propuesta de solución. El modelo conceptual de la propuesta de solución favoreció una mejor comprensión del software a implementar, y el modelo relacional, ilustró la estructura de su base de datos.

## Capítulo 3: Codificación y pruebas del módulo de gestión de incidencias

En el presente capítulo se abarcan dos de las cuatro actividades del marco de trabajo de la metodología XP: codificación y pruebas. Se definen los estándares de implementación a seguir y se explica en que consiste la filosofía de la programación en parejas que recomienda XP. Además, se sientan las bases de la integración de la solución con el Sistema de Gestión de Servicios (SGS). Se describen las pruebas de la solución: las unitarias y las de aceptación con el fin de comprobar su correcto funcionamiento y que cumple con los requerimientos establecidos por el cliente.

### 3.1 Implementación de la propuesta de solución

Una vez concluidas las actividades de planeación y diseño se procede a desarrollar la solución, en la tercera actividad denominada codificación. En esta fase se definen los estándares de implementación a seguir, y como recomienda la metodología, se mantiene la filosofía de la programación en parejas.

#### 3.1.1 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Estos se enfocan a la estructura y apariencia física del código, lo cual posibilita su mejor lectura, comprensión y mantenimiento. Un código fuente completo debe reflejar un estilo armonioso, como si un único desarrollador hubiera escrito todo el código de una sola vez. Para lograr una mejor comunicación, XP enfatiza en la comunicación entre los programadores mediante el código, pues el establecer un estándar de codificación asegura una forma coordinada de trabajo.

Algunos de los estándares utilizados en la implementación fueron:

**CamelCase (caso del camello):** Este tipo de escritura se caracteriza por las palabras unidas entre sí, con las primeras letras de cada término en mayúscula, para hacer más legible el conjunto. Se pueden apreciar dos tipos: el UperCamelCase, en el que las palabras inician con letra mayúscula, y el lowerCamelCase, en el que las palabras inician con letra minúscula.

El tipo UperCamelCase se utilizó para nombrar las clases del dominio y el tipo lowerCamelCase para las nomenclaturas de los métodos de las clases y de variables. Esto se ilustra en la Figura 12.

Login LoginController Login Views Tests

```
/*...*/
package tuba

import ...

@Secured('permitAll')
class LoginController {
    /**
     * Dependency injection for the authenticationTrustResolver.
     */
    def authenticationTrustResolver
    def usuarioService
    /**
     * Dependency injection for the springSecurityService.
     */
    def springSecurityService

    def autenticarUsuario(LoginCommand cmd) {

        if (usuarioService.autenticarUsuario(cmd)) {
            redirect(action: "index")
        } else {
            flash.message = message(code: "Usuario o contrase&ntildea incorrecto")
            redirect(action: "auth")
        }

    }
```

*Figura 12. Estándares de Codificación*

**Identación:** Significa mover un bloque de texto a la derecha para evidenciar los espacios en blanco. Se utiliza para mejorar la legibilidad del código fuente. Se utilizaron cuatro espacios en blanco como unidad de identación. En la Figura 12 se muestra un ejemplo de identación en la propuesta de solución, centrar la atención en el método autenticarUsuario.

### 3.1.2 **Programación en parejas**

La implementación la realizarán dos personas como recomienda XP, en una misma estación de trabajo. Esto proporciona una técnica para la resolución de problemas en tiempo real y garantiza la seguridad. Mientras un programador se encarga de la calidad de la funcionalidad a implementar, el otro analiza si es la forma correcta y si cumple con el estándar de codificación adecuado. Este estilo de dúo alienta más a los programadores a centrarse en el problema a resolver, ambos deben estar en un mismo nivel pues sus roles pueden ser intercambiables durante el desarrollo del software.

## 3.2 Bases para la integración de la solución al Sistema de Gestión de Servicios

El módulo de gestión de incidencias, principal resultado de esta investigación, debe integrarse con los restantes módulos y así conformar el SGS, el cual no existe actualmente. Para posibilitar la integración se destaca la utilización de las mismas herramientas de desarrollo, entre ellas el *framework* de desarrollo Grails. Este proporciona un entorno de desarrollo estandarizado y permite establecer una arquitectura MVC, "lo suficientemente flexible para permitir integraciones con otras aplicaciones y/o interfaces de usuario" (Leaftech 2014).

Si tanto la plataforma como sus componentes están desarrollados con el mismo *framework*, entonces toma ventaja de sus características, como por ejemplo, diseñar aplicaciones de forma modular. Grails, a través de un sistema de *plugins,* permite ampliar las funcionalidades de forma sencilla. De esta manera se integran los módulos a la plataforma. La solución propuesta, en forma de *plugin,* añade nuevos artefactos al SGS como: Controladores, Servicios, Vistas y Plantillas. El SGS garantiza el flujo de datos entre los plugins a través de servicios, los cuales permiten que un módulo determinado consuma información de una tabla de la base de datos generada por otro módulo.

## 3.3 Pruebas

El proceso de pruebas es uno de los cimientos de XP, esta metodología incita a que se realice una programación basada en pruebas. Esto reduce el número de errores no detectados, minimiza el tiempo entre su aparición y detección. Contribuye a elevar la calidad de los productos y la seguridad de los programadores, en caso de existir efectos colaterales, a la hora de realizarse una modificación o refactorización.

XP propone dos tipos de pruebas: unitarias y de aceptación. Las pruebas unitarias son desarrolladas por los programadores y permiten verificar el código de manera automática. Las pruebas de aceptación son realizadas por el cliente y permiten comprobar que las funcionalidades establecidas en las historias de usuario sean las esperadas.

### 3.3.1 Pruebas unitarias

Las pruebas unitarias comprueban el buen funcionamiento de una parte de la propuesta de solución, con el fin de asegurar el correcto funcionamiento de todos sus componentes por separado y evitar así, errores

futuros en el momento de la integración de todas sus partes. Permite acotar los errores, para facilitar su localización y no necesitan intervención manual (se realizan automáticamente).

Las pruebas unitarias aplicadas a la propuesta de solución se pueden dividir en dos grupos: las que hacen referencia a las clases de dominio y las que se aplican a los controladores. La prueba automatizada es una parte clave de Grails por lo que proporciona gran facilidad a la hora de ejecutar las pruebas a la propuesta de solución. Todos las pruebas unitarias se encuentran dentro de la carpeta test/unit del proyecto. En el Anexo 6 se muestran las imágenes de los resultados obtenidos tras aplicar dichas pruebas a la solución.

**Pruebas de clases de dominio**

Una de las características que define a las pruebas unitarias es ser atómicas (no tener dependencias de otras partes del software), esto puede resultar complicado porque hasta la pieza más pequeña de la propuesta de solución puede tener dependencias. Como técnica, se utilizaron los mockings (falsos objetos, imitación), para lograr la simulación de la dependencia. En Grails se proporcionan mocks para figurar las clases de dominio, por ejemplo mockDomain. Para comprobar las restricciones dentro de las clases de dominio se utiliza MockForConstraintsTests y para simular el comportamiento de los controladores mockController.

Para todas las clases del dominio de la propuesta de solución se utilizó la prueba llamada testConstraints, la cual verifica que las instancias de las clases se generan de forma adecuada según sus restricciones. Se centra en comprobar los tipos de valores que puede tomar una instancia de la clase dominio o la cantidad de caracteres que permiten sus campos.

**Pruebas de controladores**

Para los controladores se generaron las pruebas de forma automática en el Grails, originándose los siguientes métodos de examen para cada controlador:

- **testIndex():** Comprueba que se accede correctamente al índice.
- **testList():** Comprueba que las instancias se listan correctamente
- **testCreate():** Comprueba la correcta creación de instancias.
- **testSave():** Comprueba el correcto almacenamiento de las instancias.
- **testShow():** Comprueba que se muestran las instancias correctamente.
- **testEdit():** Comprueba que se editan las instancias correctamente.

- **testUpdate():** Comprueba que se actualizan las instancias correctamente.
- **testDelete():** Comprueba que se eliminan correctamente las instancias.

### 3.3.2 Pruebas de aceptación

Las pruebas de aceptación son originadas a partir de las historias de usuario. Cada historia de usuario puede tener uno o varios casos de pruebas de aceptación, con el fin de garantizar su correcto funcionamiento. Cada una de estas pruebas representan una salida esperada de la propuesta de solución, su realización debe ser rápida para efectuar los cambios necesarios en caso de que la aplicación no pase el intento satisfactoriamente.

Las pruebas de aceptación tienen más peso que las pruebas unitarias dado que constituyen un indicador de satisfacción del cliente con la solución y delimitan el final de una iteración. Se recomienda la participación activa del cliente en la realización de estas pruebas. Los casos de pruebas diseñados para su realización están formados de la siguiente manera:

**Código:** Es el identificador de la prueba de aceptación.

**HU:** Es el identificador de la historia de usuario a la que hace referencia la prueba de aceptación.

**Nombre:** Es el nombre de quien aplicó la prueba.

**Descripción:** Es la descripción de la funcionalidad de la propuesta de solución.

**Condiciones de Ejecución:** Son las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba de aceptación, estas condiciones deben ser satisfechas antes de su ejecución para que se puedan obtener los resultados esperados.

**Entradas / Pasos de Ejecución:** Es la descripción de cada uno de los pasos a seguir durante el desarrollo de la prueba de aceptación, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.

**Resultado esperado:** Es la breve descripción del resultado que se espera obtener después de aplicada la prueba de aceptación.

**Evaluación:** Evaluación acorde al resultado de la prueba de aceptación. Bien si todo funciona como se espera y mal si es adversa la respuesta tras haber aplicado las pruebas.

A continuación se ejemplifican tres casos de pruebas de aceptación de las historias de usuario crear

incidencia, visualizar incidencia y modificar incidencia, los demás se evidencian en el Anexo 4.

*Tabla 18. Caso de prueba de aceptación 1*

| **Caso de Prueba Aceptación** | |
|---|---|
| **Código:** HU1_P1 | **Historia de Usuario:** HU1 |
| **Nombre:** Neybis Lago Clara, Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Prueba para la funcionalidad autenticar usuarios en el sistema. | |
| **Condiciones de ejecución**: El usuario debe acceder a la aplicación. | |
| **Entrada/Pasos de ejecución:** Una vez que el usuario acceda a la aplicación debe ingresar los datos necesarios para su correcta autenticación. Si pertenece al directorio activo de la UCI o es un usuario local del sistema y sus datos son correctos entonces se autenticará. | |
| **Resultado esperado:** Autentificación satisfactoria. | |
| **Evaluación de la prueba:** | |

*Tabla 19. Caso de prueba de aceptación 2*

| **Caso de Prueba Aceptación** | |
|---|---|
| **Código:** HU2_P2 | **Historia de Usuario:** HU2 |
| **Nombre:** Neybis Lago Clara, Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Prueba para la funcionalidad modificar un usuario determinado. | |
| **Condiciones de ejecución**: El usuario debe estar registrado como administrador en la aplicación. | |
| **Entrada/Pasos de ejecución:** El usuario, después de haberse autenticado, selecciona la opción Usuario. Luego se visualiza en una tabla los nombres y apellidos | |

| |
|---|
| de los usuarios que existen. Si lo que quiere es modificar uno ya existente, selecciona el usuario que requiere el cambio en sus datos, elige la opción Modificar y procede a realizar la modificación en el campo deseado. |
| **Resultado esperado:** Se actualizarán los datos del usuario modificado en el sistema. |
| **Evaluación de la prueba:** |

*Tabla 20. Caso de prueba de aceptación 3*

| **Caso de Prueba Aceptación** | |
|---|---|
| **Código:** HU2_P3 | **Historia de Usuario:** HU2 |
| **Nombre:** Neybis Lago Clara, Yunior Quesada Magdaleno y Carlos A. Nuez García | |
| **Descripción:** Prueba para la funcionalidad visualizar usuarios. | |
| **Condiciones de ejecución**: El usuario debe estar registrado como administrador o técnico en la aplicación. | |
| **Entrada/Pasos de ejecución:** El usuario, después de haberse autenticado como administrador o técnico, selecciona la opción Nueva incidencia y se le muestra un formulario con todos los campos que conforman una incidencia. Dentro de estos se encuentra la opción Cliente, que tras su elección se visualizan los datos de todos los clientes para una posterior selección. En la opción Técnicos se visualizan todos los usuarios de tipo técnico a los cuales se les asignan las incidencias a resolver. | |
| **Resultado esperado:** Visualizar usuarios. | |
| **Evaluación de la prueba:** | |

Se realizaron 41 pruebas funcionales para verificar que la propuesta de solución funcionara correctamente, o sea, para cada entrada se mostraron las salidas correspondientes. Por cada incremento (entrega) del software se realizaron dos iteraciones.

De las 5 pruebas realizadas en una primera entrega fueron clasificadas de bien 4 de ellas, que representa

un 80% del total. Se encontró una no conformidad en la funcionalidad modificar usuario, de ahí 1 prueba clasificada de mal, que indica un 20% del total. En una segunda iteración, después de resolver la no conformidad, de 5 pruebas aplicadas se obtuvieron 5 clasificadas de bien, que representa el 100% del total de pruebas exitosas.

En una segunda entrega de un total de 29 pruebas realizadas fueron clasificadas de bien 26 de ellas, que representa un 90% del total. Se encontraron tres no conformidades correspondientes a las funcionalidades modificar incidencia, eliminar subcategoría y modificar categoría, de ahí 3 pruebas clasificadas de mal, que indica un 10% del total. En una segunda iteración de 29 pruebas presentadas se clasificaron de bien 29, que representa el 100% del total de pruebas funcionales exitosas.

En una tercera y última entrega de un total de 7 pruebas realizadas fueron clasificadas de bien 5 de ellas, que representa un 71% del total. Se encontraron dos no conformidades correspondientes a las funcionalidades enviar correos a los usuarios y filtrar por parámetros las incidencias, de ahí 2 pruebas clasificadas de mal, que indica un 29% del total. En una segunda iteración de 7 pruebas aplicadas se clasificaron de bien 7 que representa el 100% del total de pruebas funcionales exitosas.

En la Figura 13 se evidencian los resultados de las pruebas de aceptación realizadas.

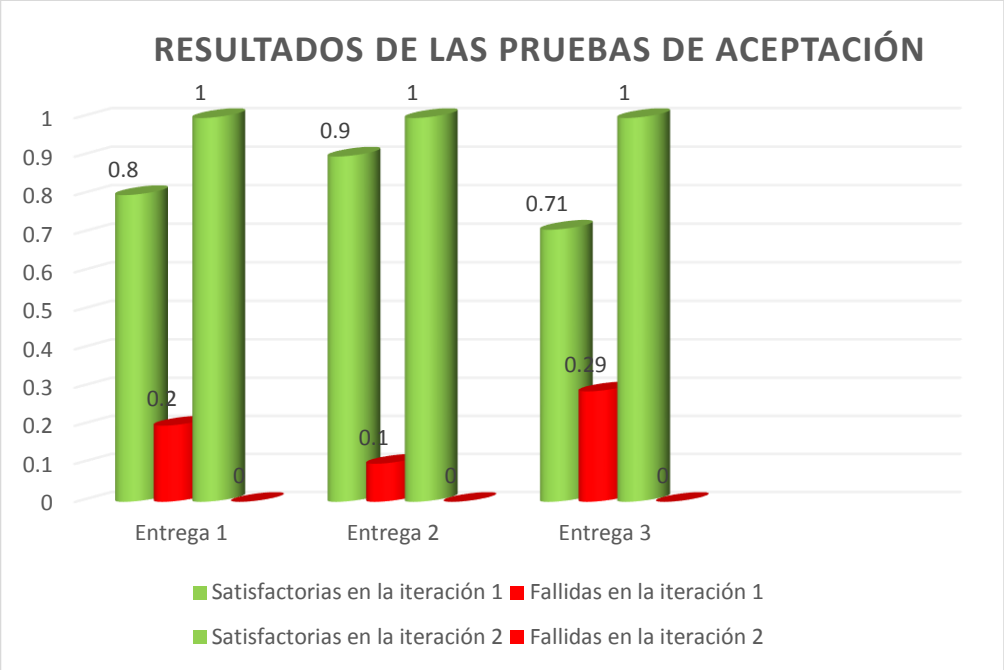



*Figura 13. Resultados de las pruebas de aceptación*

**Conclusiones parciales**

Los estándares de codificación utilizados en la implementación del software permitieron obtener un código fuente legible y de fácil comprensión para los programadores. La programación en parejas recomendada por XP permitió una notable disminución de errores. Se comprobó el correcto funcionamiento de la solución a través de la realización de las pruebas unitarias. La ejecución de las pruebas de aceptación demostró el cumplimiento satisfactorio de las exigencias del cliente. La realización de estas pruebas dio lugar a mejoras sustanciales en la implementación de las capas: lógica del negocio y presentación, como por ejemplo el desarrollo de una apariencia más profesional.

## Conclusiones generales

- La realización del estudio del arte posibilitó el análisis de conceptos fundamentales, como lo son: incidencia y servicio, resultando adecuados para el desarrollo de la investigación las definiciones proporcionadas por ITIL 2011. La realización del análisis de propuestas similares sirvió de base para la implementación de la aplicación. El estudio profundo de las herramientas y tecnologías: Groovy, Grails, jQuery UI, Bootstrap, Apache Tomcat, IntelliJ IDEA, Visual Paradigm y PosgreSQL, permitió comprender su funcionamiento para su posterior utilización en el desarrollo de la aplicación. La metodología XP sirvió de guía para el proceso de desarrollo del software proporcionando su optimización en tiempo y calidad.
- La generación de los artefactos propuestos por la metodología XP para el desarrollo de la solución, como son: las historias de usuario, las tareas de ingeniería y las tarjetas CRC, permitió la definición de los requerimientos exigidos por el cliente y el diseño del software.
- Después de la implementación de las funcionalidades definidas en las historias de usuario se obtuvo una aplicación *web* que permite la gestión de incidencias, además brinda la posibilidad de vincularlas, asociarlas y administrar todos sus componentes.
- Las pruebas unitarias y de aceptación realizadas a la aplicación demostraron el correcto funcionamiento de la solución y el cumplimiento de las exigencias del cliente. Proporcionaron mejoras en la implementación de las capas: lógica del negocio y presentación, como por ejemplo el desarrollo de una apariencia más profesional.

# *RECOMENDACIONES*

## Recomendaciones

Luego de concluida la solución propuesta se recomienda:

- Realizar la configuración personalizada de los campos. Con esto posibilitará la configuración de los campos de los formularios desde la propia interfaz web, para lograr así un mejor ajuste a las necesidades crecientes de la institución.
- Integrar con los módulos de Gestión de Configuración, Gestión de Cambios y Entregas y Gestión de Problemas.

## Bibliografía

2NDQUADRANT LTD, 2014. *2ndquadrant Professional PostgreSQL* [en línea]. 2014. S.l.: s.n. Disponible en: http://www.2ndquadrant.com/es/postgresql/.

ALEXANDER, C. 1979. *The Timeless Way of Building*. S.l.: s.n. ISBN 0195024028.

ALVAREZ, M.A. 2010. Primeros pasos con jQuery UI. S.l.:

ATLASSIAN 2013. Gestion de Proyectos e Incidencias. [en línea]. [Consulta: 12 December 2013]. Disponible en: https://www.atlassian.com/es/software/jira/service-desk.

BON, V.J. 2005. *IT Service Management: An Introduction*. 2da. S.l.: Van Haren Publishing. ISBN 978-9080671348.

BUSCHMANN, F. et. al 1996. *Pattern-Oriented Software Architecture*. S.l.: s.n. ISBN 0471958697.

COCHRAN, D. 2012. *No Title*. 1era. S.l.: s.n.

CONSULTORES, A. 2013. ¿Qué es CMMI para servicios (CMMI-SVC)? [en línea]. [Consulta: 12 December 2013]. Disponible en: http://www.avantare.com/lineas-de-negocio/materiales-de-referencia/que-es-cmmi-para-servicios-cmmi-svc?es.

DEFINICION ABC 2013. Definicionabc. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://www.definicionabc.com/economia/servicio.php.

EARCON, S.. 2006. KMKey Help Desk. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://www.kmkey.com/kmkey_com/es/productos/software_help_desk.

ESPIRAL MS 2013. Principales Beneficios de ITIL. [en línea]. [Consulta: 20 November 2013]. Disponible en: http://www.proactivanet.com/principales-beneficios-de-itil.

# *BIBLIOGRAFÍA*


G. JACK BOLOGNA, A.M.W. 1997. *Manual de Tecnología de la Información del Contador*. 1era. S.l.: s.n. ISBN 978-0471304739.

GARCÍA GALLARDO, Y.S. 2012. *Sistema de Gestión de Servicios de Soporte* [en línea]. S.l.: s.n. Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/7962/1/TM_06265_12.pdf.

GROSSO, A. 2011. Prácticas de Software. Experiencias sobre la Ingeniería y Management del Software. [en línea]. [Consulta: 15 May 2014]. Disponible en: http://www.practicasdesoftware.com.ar/.

HERNÁNDEZ, P.V., 2010. *Uso de patrones de arquitectura*. 2010. S.l.: s.n.

INGENIERÍA DRIC 2014. Ingeniería Dric. [en línea]. [Consulta: 15 May 2014]. Disponible en: http://dric.com.mx/ManageEngine/servicedesk-plus.html.

INGENIERÍA DRIC S.A. DE C.V. 2013. Manage Engine. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://manageengine.mx/service-desk/index.html.

INGENIUS 2009. Ingenius. [en línea]. Disponible en: http://ingenius.informatica.us.es/index.php/descargas/gestiondeserviciosti.

JCALDERON 2008. My Space to Share. S.l.:

JQUERY 2011. jQuery UI. .

KOENIG, D. 2007. *Groovy in Action*. 2nd. S.l.: Manning Publications Co. ISBN 1-932394-84-2.

KOOK, W.E. 2013. PROYECTO PRODUCTION BOOK. [en línea]. S.l.: Disponible en: contratos.ecopetrol.com.co/Anexos de Procesos/50020960/Anexo No. 22 Glosario (Adendo No. 7).pdf.

LAMB CHARLES, HAIR JOSEPH, M.C. 2002. *Marketing*. 6ta. S.l.: South-Western College Pub. ISBN 978-0324068610.

# *BIBLIOGRAFÍA*


LARMAN, C. 1999. *UML Y PATRONES INTRODUCCION Al ANALISIS Y DISEÑO ORIENTADO A OBJETOS* [en línea]. 1era. Mexico: s.n. ISBN 970-17-0261-1. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/UML_y_Patrones.

LEAFTECH 2014. Leaftech Making Friendly IT. [en línea]. [Consulta: 10 June 2014]. Disponible en: http://leaftech.com.mx/IntApps.php.

LEANDRO BERTOLAMI 2011. Grails: El Santo Grial de Java. [en línea]. Disponible en: http://www.1024.com.uy/revista/index.php/galileo-galilei/110-grails-el-santo-grial-de-java.

MITRE, H.A.H. 2010. Alineando COBIT® 4.1, ITIL® V3 e ISO/IEC 27002 en beneficio de la empresa. [en línea]. S.l.: Disponible en: http://e-archivo.uc3m.es/bitstream/handle/10016/9879/TesisDoctoral-HugoMitre.pdf?sequence=2.

NETSUPPORT 2013. NetSupport ServiceDesk. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://www.netsupportservicedesk.com/es/.

NYCE S.C., 2013. *ISO/IEC 20000*. 2013. S.l.: s.n.

ONETONE 2013. OnetOne. [en línea]. [Consulta: 15 May 2014]. Disponible en: http://www.onetone.cl/.

OSIATIS 2006a. Fundamentos de la Gestion TI. [en línea]. [Consulta: 28 November 2013]. Disponible en: http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/fundamentos_de_la_gestion_TI/que_es_ITIL/que_es_ITIL.php.

OSIATIS 2006b. ITIL® v3, Gestion de Servicios TI. [en línea]. Disponible en: http://itilv3.osiatis.es/itil.php.

OSIATIS 2006c. ITIL® v3, Gestion de Servicios TI. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://itilv3.osiatis.es/operacion_servicios_TI/gestion_incidencias/introduccion_objetivos.php.

OSIATIS 2006d. ITIL® v3, Gestion de Servicios TI. [en línea]. [Consulta: 13 December 2013]. Disponible en: http://itilv3.osiatis.es/gestion_servicios_ti.php.

# *BIBLIOGRAFÍA*


OVERTI 2011. Gestion de Incidencias ISO 20000. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://www.overti.es/procesos-itsm/gestion-incidencias-iso-20000.aspx.

PRESSMAN, R.S. 2002. *Ingeniería de Software, un enfoque práctico*. Quinta edi. S.l.: s.n. ISBN 8448132149.

PRESSMAN, R.S. 2005. *Ingeniería del Software: Un Enfoque Práctico, Roger Pressman*. Sexta Edic. S.l.: s.n. ISBN 9701054733.

SANDHUSEN, R. 2002. *Mercadotecnia*. 1ra. S.l.: CECSA. ISBN 978-9702402329.

SEVILLA, O.C.D.F.H. 2010. Programación Extrema. [en línea]. [Consulta: 12 December 2013]. Disponible en: http://programacionextrema.tripod.com/index.htm.

SOFTWARE B.D.L. 2012. JetBrains IntelliJ IDEA 11.1.4. Ultimate. S.l.:

SOMMERVILLE, I. 2005. *Ingenieria del Software, Ian Sommerville (Prentice Hall)*. 7ma Edició. S.l.: s.n. ISBN 8478290745.

STANTON WILLIAM, ETZEL MICHAEL, W.B. 2004. *Marketing*. 13va. S.l.: s.n. ISBN 978-0072526509.

STRITE, D. 2012. ITSM Maturity Assessment. [en línea]. S.l.: Disponible en: http://net.educause.edu/ir/library/pdf/1204ITSITSMMatAssessment.pdf.

VISUAL PARADIGM INTERNATIONAL LTD., 2013. *Visual Paradigm for UML* [en línea]. 2013. S.l.: s.n. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(MCD)_14720_p/.

## Glosario de Términos

**HTML:** (Lenguaje de marcado hipertextual), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web. Se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

**HTTP:** Protocolo de transferencia de hipertexto.

**JavaServer Pages:** Es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML entre otros tipos de documentos.

**JavaServlet:** Los servlets son objetos que corren dentro y fuera del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad.

**XML:** (Lenguaje de Marcas Extensible) es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C).

**CSS**: Hojas de Estilo en Cascada (CSS, por sus siglas en inglés)