Universidad de las Ciencias Informáticas

Facultad 7

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

"Subsistema de Gestión de Auditorías del Sistema Automatizado de Monitoreo y Control de Servicios del Centro de Soporte UCI"

Autores: Dairis Almaguer Pérez

Danilo Rodríguez Díaz

Tutores: Dra. C. Marely del Rosario Cruz Felipe

Msc. Yulio Seriocha García Gallardo

La Habana, 16 de mayo 2013.

"Año 55 de la Revolución"

DATOS DE CONTACTO

- Dra. Marely del Rosario Cruz Felipe, Universidad de las Ciencias Informáticas. Jefa de departamento de programación y sistemas digitales de la facultad 5, Profesora titular, máster en telemática y doctora en ciencias técnicas. Coordinadora de la maestría de informática aplicada.

- Mc. Yulio Seriocha García Gallardo, graduado de Ingeniero en Ciencias Informáticas de la UCI del año 2007, se desempeña como especialista del Centro Soporte UCI. Es máster en informática aplicada. Categoría docente: Asistente.

"El revolucionario verdadero está guiado por grandes sentimientos de amor".

"Seamos realistas, y hagamos lo imposible".

Ernesto Guevara de la Serna.

DEDICATORIA

Dedico esta tesis a mi madre por ser la persona más especial que existe en el mundo. Es la estrella que ha iluminado mi camino toda la vida y quien me dio apoyo desde el primer día. Cuando pensaba en dar un paso atrás siempre estaba ahí para impulsarme a dar dos al frente.

Dairis

Dedico este trabajo a mi abuelo, Fernando Luis Díaz Hernández. Aunque ya no está presente, me acompaña en todos mis pasos.

Danilo

Agradecimientos

A la Revolución, por habernos dado la oportunidad de desarrollarnos profesionalmente.

A la Universidad de las Ciencias Informáticas.

A los tutores Yulio y Marely, por transmitirnos su experiencia y ayudarnos en todo momento.

A Yicel, Eddy, Yonny y Alexander por ser amigos, por su apoyo incondicional, su preocupación y dedicación.

A los profesores que nos han formado como profesionales.

A nuestros amigos y a los que aportaron al desarrollo de este trabajo.

RESUMEN

La gestión de los servicios informáticos en los últimos años ha aumentado vertiginosamente producto de la habitual utilización de los recursos informáticos que presenta la humanidad. A razón de esto, los requisitos en los productos son cada vez mayores y se exige una calidad lo más óptima posible. La Universidad de las Ciencias Informáticas (UCI), conocida como la mayor institución informática de Cuba, presta un conjunto de servicios entre ellos el servicio de soporte técnico a los softwares liberados por la universidad.

En el centro de soporte de la UCI se debe llevar un control adecuado de los servicios que se prestan. Para realizar este control se hace necesario auditar estos servicios de acuerdo a un conjunto de normas definidas por el estándar COBIT.

En la presente investigación se desarrolla un subsistema de gestión de auditorías del Sistema Automatizado de Monitoreo y Control de Servicios (SAMOS) para incrementar la prestación de servicios de Tecnologías de la Información en el Centro de Soporte UCI. Para llevar a cabo la implementación de dicho subsistema fueron analizados los sistemas de gestión de auditoría de servicios de TI existentes en la actualidad como resultado de un profundo estudio del arte en el ámbito nacional e internacional, se describieron las herramientas y tecnologías utilizadas en la solución desarrollada (metodología, patrones, softwares, lenguajes de programación, etc.) que permitieron una implementación eficiente en corto tiempo del subsistema propuesto. La solución implementada, contribuirá a la mejora de la prestación de los servicios de TI en el Centro de Soporte UCI. Lo antes escrito se refleja de manera positiva en una mejor satisfacción de los clientes y por consiguiente en una mayor aceptación de los servicios brindados por la entidad.

Palabras claves:

Auditoría, servicios, tecnologías de información, servicios de TI.

ABSTRACT

In recent years, there has been an increase in the use of computing resources for managing services related to information technology. As a consequence, user requirements on this kind of product's functionalities and capabilities are ever growing in complexity and reach. Plus, there is a high level of expectancy on the user's side about the optimal quality of the software they will use.

The University of Informatics Sciences (UCI), regarded as the largest computing Cuban institution, provides a wide range of services, including technical support for software applications released by the university. In the UCI Software Support & Technology Center, it is required to attain adequate control over the services provided therein. In order to fulfill such control needs, it is necessary to audit those services according to a set of guidelines and policies defined by the COBIT standard.

Within the boundaries of this research effort, an audit management subsystem has been developed for the Automated System Monitoring and Control Services (SAMOS) solution. Such subsystem should constitute a tool aimed at elevating the quality, effectiveness and efficiency of the services provided by the aforementioned Center.

Prior to the development of this subsystem, a thorough study was conducted about state-of-the-art software systems related to IT services audit management, in both national and international arenas. Then, the tools and technologies used in the developed solution were described (methodology, patterns, software, programming languages, etc.), which enabled an efficient and fast implementation of the proposed solution. Such developed solution will contribute to improving the provision of IT services in the UCI Software Support & Technology Center. All previously stated has a positive impact in customer satisfaction and, therefore, in greater acceptance of the services provided by the entity.

Keywords:

auditing, services, information technology, IT services.

ÍNDICE

INTRODUCCIÓN

Cuando se relacionan los contenidos, la información, el equipamiento necesario, una infraestructura material adecuada y el factor humano logrando la integración y convergencia de las técnicas para el procesamiento de datos y las telecomunicaciones, se está en presencia de una de las herramientas más potentes con las que ha contado la humanidad: las tecnologías de la información(TI).

Los orígenes de las TI son recientes. Aunque su nombre se remonta a los años 70, su utilización en los negocios data de mediados del siglo XX, durante la segunda guerra mundial. Sin embargo, ha sido en los últimos 20 años donde ha alcanzado niveles de uso y aplicaciones tan variadas y ubicuas, que se ha convertido en un área de gran amplitud e impacto en todos los aspectos de la vida cotidiana (1). Demostrando así que son las principales protagonistas del desarrollo informático, en una sociedad, tanto para su desarrollo como para su aplicación. Además se reconoce como las TI constituyen el núcleo central de una transformación multidimensional que experimenta la economía y la sociedad ya que tiende a modificar no sólo sus hábitos y patrones de conducta, sino, incluso, su forma de pensar.

Es necesario establecer que las TI se entienden como “aquellas herramientas y métodos empleados para recabar, retener, manipular o distribuir información”. Las tecnologías de la información se encuentran generalmente asociadas con las computadoras y las tecnologías afines, aplicadas a la toma de decisiones (2).

Las TI están cambiando la forma tradicional de hacer las cosas, las personas que trabajan en gobierno, en empresas privadas, que dirigen personal o que trabajan como profesional en cualquier campo utilizan las TI cotidianamente mediante el uso de Internet, las tarjetas de crédito, el pago electrónico de la nómina, entre otras funciones; es por eso que la función de las TI en los procesos de la empresa como manufactura y ventas se han expandido grandemente. La primera generación de computadoras estaba destinada a guardar los registros y monitorear el desempeño operativo de la empresa, pero la información no era oportuna ya que el análisis obtenido en un día determinado en realidad describía lo que había pasado una semana antes. Los avances actuales hacen posible capturar y utilizar la información en el momento que se genera, es decir, tener procesos en línea (3).

La integración de las TI en los procesos de negocio con el objetivo de aportar valor a las compañías, se ha convertido en uno de los principales retos actuales (4). Existe en el mundo una gran competencia en la

venta de soluciones Informáticas, es por ello que los productores de software buscan mejorar sus productos cada vez más, haciendo uso de las Tecnologías de la Información.

La información y la tecnología que la soporta representan los activos más valiosos de muchas empresas, aunque con frecuencia son poco entendidos. Las empresas exitosas reconocen los beneficios de las TI y la utilizan para impulsar el valor de sus interesados (stakeholders). Estas empresas también entienden y administran los riesgos asociados, es decir, el aumento en los requerimientos regulatorios, así como también una gran dependencia de muchos de los procesos de negocio en TI. Pero todos estos elementos son clave para el gobierno de la empresa. El valor, el riesgo y el control constituyen la esencia del gobierno de TI. (5)

Con el objetivo de mejorar los servicios de TI se han desarrollado un conjunto de buenas prácticas y estándares como son: ISO/IEC *20000 (Organización Internacional de Normalización / Comisión Electrotécnica Internacional)*, CMMI-SVC (*Modelo de Madurez de la Capacidad Integrado para Servicios)*, ITIL *(Biblioteca de Infraestructura de Tecnologías de Información)* y COBIT (*Objetivos de Control para la Tecnología de la Información*) (6). COBIT aporta un factor diferencial enormemente importante: la orientación hacia el negocio. Permite a las empresas aumentar el valor de las TI y reducir los riesgos asociados a proyectos tecnológicos, esto a partir de parámetros generalmente aplicables y aceptados, para mejorar las prácticas de planeación, control y seguridad de las TI.

COBIT fue desarrollado por la Información del Sistema de Auditoría y Control de Asociación (ISACA por sus siglas en inglés) y el Instituto de Gobierno de TI (ITGI por sus siglas en inglés). Es un estándar aceptado mundialmente para el adecuado control de proyectos de tecnología, los flujos de información y los riesgos que éstas implican, y es la principal propuesta metodológica realizada a nivel internacional para abordar la auditoría de sistemas de información. El estándar COBIT se utiliza para planear, implementar, controlar y evaluar el gobierno sobre TI, incorporando objetivos de control, directivas de auditoría, medidas de rendimientos y resultados, factores críticos de éxito y modelos de madurez (6).

La filosofía de COBIT asimila los principios de reingeniería de empresas BPR (Reingeniería de Procesos) y divide las funciones que ha de realizar un sistema de información en procesos que, a su vez, están subdivididos en actividades y tareas más simples; como los sistemas de información están orientados a los procesos y por tanto su auditoría se debe adaptar a estos conceptos (6).

El enfoque del control en TI se lleva a cabo visualizando la información necesaria para dar soporte a los procesos de negocio. Considerando la información como el resultado de aplicación combinada de recursos relacionados con las TI que deben ser administradas por procesos de TI. Para alcanzar los requerimientos de negocios, la información necesita satisfacer ciertas exigencias y criterios como lo son: calidad, costo y entrega de servicio. Efectividad y eficiencia de las operaciones, fiabilidad de la información y cumplimiento de las leyes y normas. Además de garantizar la confidencialidad, integridad y disponibilidad (6).

Existen varias empresas alrededor del mundo así como grupos grandes, pequeños y medianos que se dedican al desarrollo de soluciones informáticas y productos software en general. Cuba no está ajena a los avances de la ciencia y la tecnología. Desde hace varias décadas viene desarrollando una estrategia encaminada al avance de esta industria del software. En el país se han creado empresas e instituciones como Desoft, Softel y la Universidad de las Ciencias Informáticas (UCI).

La UCI es una de las principales instituciones productoras de software de nuestro país. Cuenta con varios centros de desarrollo orientados a la aplicación de la informática en las diferentes esferas de la sociedad, en los que se realizan soluciones para las diferentes problemáticas existentes. La universidad cuenta además con un Centro de Soporte que ofrece servicios de soporte y mantenimiento a los productos desarrollados en los diferentes centros de investigación y desarrollo, de la institución, y resuelve a los clientes las incidencias tecnológicas que tengan solución inmediata al alcance del equipo técnico.

Actualmente el Centro de Soporte no cuenta con una estrategia clave para la gestión de servicios de TI. Esto conlleva a una baja productividad de los responsables incidiendo en el óptimo funcionamiento de la gestión de incidencias, la Base de Conocimientos, el Centro de Descargas, la Comunidad de Soporte y el Centro de Llamadas. Todo esto acarrea una baja eficiencia a la hora de canalizar las peticiones con rapidez hacia el servicio de soporte correspondiente, aumentando el tiempo de respuesta de las peticiones de los clientes. No se realiza una auditoría en tiempo real de los servicios prestados que permita evaluar la eficiencia de estos y el desempeño del personal implicado.

La gestión de las incidencias ocurridas a las aplicaciones informáticas; se realiza actualmente sin tener en cuenta los ANS (Acuerdos de Niveles de Servicios) establecidos con el cliente. La estructura organizacional del Centro de Soporte, no permite balancear los recursos humanos, ni gestionar la demanda del servicio de soporte técnico. Se desconocen los cambios realizados a las aplicaciones informáticas, lo que provoca demoras en el servicio a los clientes además de que existe un desconocimiento total en el nivel de

satisfacción de los mismos, tampoco se tiene conocimiento de sus expectativas para con el servicio de soporte técnico, como consecuencia el cliente en repetidas ocasiones se comunica directamente con el centro de desarrollo cuando necesita resolver una incidencia. Existe un desconocimiento de las actividades realizadas por los técnicos y especialistas de soporte durante la prestación del servicio.

Por lo antes planteado, se identifica como **problema a resolver:** ¿cómo realizar la auditoría de servicios de Tecnologías de la Información en el Centro de Soporte UCI?

Este problema se enmarca en el **objeto de estudio:** gestión de auditorías de servicios de Tecnologías de la Información.

El objeto delimita el **campo de acción:** gestión de auditorías de servicios de Tecnologías de la Información del Sistema Automatizado de Monitoreo y Control de Servicios del Centro de Soporte UCI.

Para la solución del problema se plantea como **objetivo general:** desarrollar un subsistema de gestión de auditorías del Sistema Automatizado de Monitoreo y Control de Servicios para mejorar los servicios de Tecnologías de la Información en el Centro de Soporte UCI.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de la Investigación:**

- Realización de un análisis crítico y valorativo de los sistemas que gestionan las auditorías de servicios de Tecnología de la Información, tanto a nivel nacional como internacional.
- Identificación de los requerimientos del software.
- Definición de la arquitectura y tecnologías para el desarrollo del subsistema de gestión de auditorías para el Centro de Soporte UCI.
- Implementación del subsistema de auditorías de servicios de Tecnologías de la Información del Sistema Automatizado de Monitoreo y Control de Servicios del Centro de Soporte UCI.
- Validación de resultados del sistema de auditoría.

Los **métodos científicos** utilizados en la investigación son los siguientes:

Métodos Teóricos

- *Analítico–sintético:* se utiliza para analizar y comprender la teoría y documentación relacionada con el tema de investigación. Se analizaron las generalidades acerca del tema de investigación y se definieron los elementos significativos relacionados en mayor medida con el objeto de estudio.

- *Histórico–lógico:* este método permite entender la evolución y el surgimiento de las TI, así como las temáticas relacionadas con la auditoría de servicios de las tecnologías de la información a lo largo de la historia de la Informática. El estudio de los principales acontecimientos que han marcado la evolución de las tecnologías de la información, permite una mayor comprensión del marco teórico relacionado con los temas de estudio, de la presente investigación científica.
- *Modelación:* mediante este método se modela el sistema (los elementos que lo componen y su funcionamiento). El entendimiento de los elementos unitarios que componen el sistema, permite además, la correcta comprensión del sistema como un todo.

Métodos Empíricos

- *Entrevista:* para el desarrollo de este método se han entrevistado a especialistas del Centro de Soporte UCI, quienes han aportado elementos significativos a la investigación. Los resultados han permitido detectar a partir de la situación real que no se está realizando el proceso de auditoría en el Centro de Soporte UCI.
- *Análisis de documentos:* este método ha sido de gran ayuda porque a partir de un grupo de resoluciones y documentos que exponen con mucho detalle el flujo que sigue el proceso que representa el objeto de la investigación, ha sido más fácil la comprensión del mismo. Se utilizó a la hora de la realización de los procedimientos y plantillas para que estuvieran en concordancia con los demás documentos del centro y de la universidad.

Para mostrar el desarrollo de la investigación y los resultados, el trabajo se ha estructurado en tres capítulos, además de Conclusiones, Recomendaciones, Bibliografía y Anexos.

Capítulo 1: Contiene un estudio del estado del arte de las principales herramientas que existen en el mundo para la gestión de auditorías de servicios de TI. El análisis de las diferentes metodologías y tecnologías que se emplean en la solución para llevar a cabo un proceso con calidad en el desarrollo de la investigación.

Capítulo 2: Contiene lo referente al análisis, diseño e implementación del sistema. Se realiza la especificación de los requisitos funcionales y no funcionales del sistema, detallándose a su vez las historias de usuarios para un mejor entendimiento de los mismos. También se describen las clases y el modelo de Base de Datos para llevar a cabo una implementación eficiente.

Capítulo 3: Aborda todo lo relacionado con la validación del sistema. Se recoge toda la documentación relacionada con las pruebas realizadas al subsistema, los estándares de codificación y patrones de diseño. Además se muestra y describe el método con el que se realizan las pruebas.

CAPÍTULO 1. **FUNDAMENTACIÓN TEÓRICA**

En el presente capítulo se expone el marco teórico y conceptual asociado a la problemática a resolver para lograr un mayor entendimiento de la misma. Se incluyen los resultados del análisis del estado del arte de los diferentes sistemas de auditorías de TI que existen en el mundo. Además, se exponen las principales características de las tecnologías que serán utilizadas para la implementación de la aplicación así como las herramientas y metodologías a utilizar.

1.1. Auditoría de servicios de TI

Conceptualmente toda y cualquier auditoría, es la actividad consistente en la emisión de una opinión profesional sobre si el objeto sometido a análisis posee adecuadamente la realidad que pretende reflejar y/o cumple las condiciones que le han sido prescritas (7). El objetivo de la Auditoría consiste en apoyar a los miembros de la empresa en el desempeño de sus actividades. Para ello les proporciona análisis, evaluaciones, recomendaciones, asesoría e información concerniente a las actividades revisadas.

1.1.1. Conceptos de Auditoría:

Según Arthur Holmes, la auditoría son las comprobaciones científicas y sistemáticas de los libros de cuentas, comprobantes y otros registros financieros y legales de un individuo, firma o corporación, con el objetivo de determinar la exactitud e integridad de la contabilidad; mostrar la verdadera situación financiera y la operación y certificar los estados que se brindan. Holmes continúa diciendo que la auditoría es el examen de las demostraciones y registros administrativos. El auditor observa la exactitud, integridad y autenticidad de tales demostraciones, registros y documentos (8).

Según la definición de un profesor de la Universidad de Harvard, la auditoría es el examen de todas las anotaciones contables a fin de comprobar su exactitud, así como la veracidad de los estados o situaciones que dichas anotaciones producen (9).

La auditoría es un examen comprensivo de la estructura de una empresa, en cuanto a los planes y objetivos, métodos y controles, su forma de operación y sus equipos humanos y físicos. Es una visión formal y sistemática para determinar hasta qué punto una organización está cumpliendo los objetivos establecidos por la gerencia, así como para identificar los que requieren mejorarse (10).

Tomando en cuenta los criterios expuestos se puede concluir que la auditoría es una función de dirección cuya finalidad es analizar y apreciar, el control y funcionamiento interno de las organizaciones. Su finalidad principal consiste en garantizar la integridad de su patrimonio, la veracidad de su información y el mantenimiento de la eficacia de sus sistemas de gestión.

1.1.2. Definiciones de Servicios.

Los servicios son actividades identificables, intangibles y perecederas que son el resultado de esfuerzos humanos o mecánicos que producen un hecho, un desempeño o un esfuerzo que implican generalmente la participación del cliente y que no es posible poseer físicamente, ni transportarlos o almacenarlos, pero que pueden ser ofrecidos en renta o a la venta; por tanto, pueden ser el objeto principal de una transacción ideada para satisfacer las necesidades o deseos de los clientes (11).

Tomando en cuenta la definición expuesta anteriormente y varios criterios de expertos se concluye que un servicio es un conjunto de actividades que realiza internamente una empresa para responder y satisfacer las necesidades de un cliente. Es una relación de contrato entre empresas y/o personas donde se comprometen a cumplir con un contrato o acuerdo.

1.2. Estándares para la mejora de servicios de TI.

1.2.1. Objetivos de Control para la Tecnología de la Información (COBIT)

COBIT, es un guía volcado para la Gestión de TI. Tiene una serie de recursos que pueden servir como un modelo de referencia para Gestión de TI, incluyendo un sumario ejecutivo, un "framework" con control de los objetivos, mapas de auditoría, herramientas para su implementación y principalmente, una guía con técnicas de gestión (12).

COBIT constituye una importante herramienta en la estructuración y control de los procesos de TI de forma que atiende la demanda de las diversas áreas de la empresa, de los accionistas, de los órganos reguladores y de las entidades externas, por alineación, transparencia y ecualización de los riesgos de TI.

El framework de COBIT fue creado con las características para ser enfocado en los negocios de la empresa y es totalmente orientado al proceso y las mediciones de madurez con base en controles. Provee informaciones detalladas para gestionar los procesos basados en objetivos del negocio.

COBIT fue proyectado para auxiliar tres áreas distintas:

- Gestores que necesitan evaluar el riesgo y controlar las inversiones de TI en las organizaciones.
- Usuarios que necesitan tener garantías de que los servicios de TI que dependen de que sus productos y servicios para los clientes internos y externos están siendo bien gestionados.
- Auditores que pueden apoyarse en las recomendaciones de COBIT para evaluar el nivel de la Gestión de TI y aconsejar el control interno de la organización.

El principio de este framework plantea: con el fin de facilitar informaciones que la empresa necesita para alcanzar sus objetivos, que la empresa necesita gestionar y controlar los recursos de TI usando un conjunto estructurado de procesos para entregar las informaciones que fueron requeridas.

Seguidamente, las principales ventajas en la adopción de COBIT:

- COBIT está alineado con otros estándares y con todas las mejores prácticas.
- El framework de COBIT soporta y es soportado por las mejores prácticas, permitiendo así un ambiente de TI bien gestionado y flexible.
- COBIT se enfoca en la mejora del Gobierno Corporativo de TI de las organizaciones en la reducción de los riesgos operacionales.
- Gestiona y controla las actividades de TI.
- COBIT proporciona un ambiente de control responsable por garantizar las necesidades del negocio.
- Deja disponible herramientas para auxiliar en la gestión y en el control de las actividades de TI.
- Garantiza que las funciones corporativas ocurran de forma sistemática para el alcance de los objetivos del negocio.
- Provee la creación de un lenguaje común a todos los involucrados en el control de los procesos.

Desventajas de COBIT:

- Resulta un modelo ambicioso que requiere de profundidad en el estudio, que se enriquece constantemente y provee de guías de auditorías que por dificultades económicas y de gestión no hemos podido obtener.
- No existe en la bibliografía resultados de la experiencia práctica de los países en la implementación de este modelo que lo hagan medible, sin embargo conocemos que se emplea pero no en la generalidad de los casos.

1.2.2. Modelo de Madurez de la Capacidad Integrado para Servicios (CMMI-SVC)

CMMI-SVC, (13) es un abordaje de mejora del proceso que provee a las organizaciones los elementos esenciales para procesos eficaces que resultan en un mejor desempeño de la organización como un todo. Este estándar puede ser utilizado para orientar la mejora del proceso a través de un proyecto, una división, o una organización entera. CMMI ayuda a integrar las funciones organizacionales tradicionalmente separadas, definir objetivos y prioridades para la mejora del proceso, proveer orientación para los procesos de la calidad y proporcionar un punto de referencia a la evaluación del proceso actual. Mejora de la eficiencia en muchas disciplinas de procesos en una organización. Además tiene una visión común e integrada de mejoras para todos los elementos de una organización.

Ventajas de CMMI-SVC:

- Integrar de forma más explícita las actividades de gestión y de ingeniería con sus objetivos de negocio.
- Ampliar el alcance y visibilidad para el ciclo de vida del producto y las actividades de ingeniería para garantizar que el producto o servicio corresponda a las expectativas del cliente.
- Integrar lecciones aprendidas de otras áreas de buenas prácticas (por ejemplo, medición, gestión de riesgos y gestión de proveedores).
- Implementar prácticas de alta madurez más robustas.
- Abordar funciones organizacionales adicionales que también son críticas a sus productos y servicios.
- Conformidad más plena con las normas ISO.

Desventajas de CMMI-SVC

- Tamaño y complejidad mucho mayor que modelos vigentes.
- El proceso de evaluación es más costoso en tiempo y esfuerzo.
- La complejidad de la evaluación continua puede atentar contra la definición de objetivos concretos de madurez.

La mayoría de las empresas están buscando mejorar sus procesos de negocio, a fin de mejorar la productividad y reducir costos, buscando en CMMI ayuda para implementar sus mejores prácticas. Sin embargo, el abordaje de CMMI no dice "como" se hace para alcanzar esa mejora, lo que permite que cada empresa pueda crear su propio abordaje, basada en lo que mejor se adecua a la organización. Debido a las muchas maneras por las cuales una organización puede realizar esas actividades, cada proceso puede (y debe) ser único a cada empresa.

1.2.3. Organización Internacional de Normalización / Comisión Electrotécnica Internacional (ISO/IEC 20000)

La serie ISO/IEC 20000 normalizada y publicada por las organizaciones ISO (Organización Internacional de Normalización) e IEC (Comisión Electrotécnica Internacional) el 14 de diciembre de 2005, es el estándar reconocido internacionalmente en gestión de servicios de TI. La serie 20000 proviene de la adopción de la serie BS 15000 desarrollada por la entidad de normalización británica, la BSI (Institución Británica Estandarizada) (14).

ISO/IEC 20000 es una norma de TI que permite a las empresas demostrar excelencia y comprobar las mejores prácticas en gestión de TI. Se concentra en el compromiso de la alta dirección en construir motivación humana, con la finalidad de atender a todos los objetivos del negocio. La norma establece las mejores prácticas de trabajo, con la finalidad de aumentar la ganancia, así como la satisfacción del cliente (15).

ISO / IEC 20000:2005 está basada en la norma ISO 9001:2000, pero tiene como finalidad atender principalmente la Administración de los Servicios de TI (ITSM). La certificación ISO / IEC 20000:2005 ayuda a los proveedores, propietarios de marcas y minoristas a proteger sus organizaciones y consumidores. La norma describe un conjunto integrado de procesos de gestión para la efectiva prestación de servicios a la empresa y sus clientes (15).

Iniciada por dos organizaciones británicas, ITSMF (*El Fórum de Administración de Servicios de TI*) y BSI está modelada con base en los principios de ITIL. Sin embargo, en contraste con los libros de ITIL, ISO 20000 no ofrece consejos específicos sobre como modelar los procesos de la organización. Pero sí, un conjunto de requisitos que deben ser cumplidos con la finalidad de calificarse para la certificación ISO 20000 (15).

Las mejores prácticas de ISO 20000 facilita la realización de los siguientes objetivos de negocios de TI:

- Optimizar el uso de recursos de TI.
- Reducir costos.
- Mejorar la disponibilidad.
- Ajustar la capacidad.
- Aumentar la eficiencia.

- Mejorar la escalabilidad.
- Alinea la TI con las necesidades del negocio.
- Crea el ambiente para cambiar de la cultura basada en tecnología a la basada en servicio.
- Mejora la capacidad de absorber cambios rápidos.
- Mejora la calidad de los servicios de TI.
- TI conocida como proveedora de valor para la organización.
- Garantiza que todos hablen el mismo idioma.
- Mejora la información sobre los servicios actuales de TI.
- Proporciona una visión más clara de la capacidad actual de la TI.
- Aumenta la satisfacción del cliente.
- Mejora la motivación del personal de TI.
- Permite mayor flexibilidad a la empresa, a través, de una mejor comprensión del soporte de TI.
- Mejora la imagen de TI como un aliado del negocio.
- Permite mayor tiempo de ciclo de cambios y mayor tasa de éxito.

Desventajas de ISO/IEC 20000

- Cuando el proceso de cambio es manejado solo con recursos internos, se corre el riesgo de no poder cambiar el estado, porque implicaría marcar errores en vez de oportunidades de mejoras y responsables en vez de líderes del cambio.
- Un proyecto sin hitos intermedios puede llevar mucho tiempo y esfuerzo y requerir un cambio cultural en la organización importante. Un enfoque demasiado ambicioso puede llevar a la frustración porque los objetivos no son cumplidos.
- Si la estructura de procesos se transforma en un objetivo en sí mismo, la calidad del servicio puede verse afectada. En este caso, los procedimientos se transforman en obstáculos burocráticos que requieren ser evitados.
- Puede no haber mejoras, debido a la falta de entendimiento sobre el alcance de procesos, los indicadores de rendimiento, o cómo los procesos deberían ser controlados.

1.2.4. Biblioteca de Infraestructura de Tecnologías de Información (ITIL)

ITIL, es un conjunto de mejores prácticas que orientan la Gestión de Servicio de TI. Este estándar es propiedad del OGC (*Oficina de Comercio Gubernamental,* de Reino Unido) y consiste en una serie de

publicaciones que proveen recomendaciones para el aprovisionamiento de la calidad de los Servicios de TI, y de los procesos y recursos necesarios para soportarlos (16). Ha sido creado para que organizaciones de toda clase pudieran controlar en forma más efectiva los gastos y la eficiencia de los servicios de TI que contrata. ITIL es soportado por un sistema de cualificación abarcador, organizaciones de entrenamiento acreditadas y herramientas de implementación y evaluación.

Desde el punto de vista del negocio, la adopción de prácticas de ITIL por prestadores de servicios de TI ya sea interno o de Proveedores externos garantiza varios beneficios, incluyendo:

- Servicios de TI que se alinean mejor con las prioridades y objetivos del negocio, lo que significa que la empresa realiza más en términos de sus objetivos estratégicos.
- Costos de TI conocidos y gerenciables, garantizando mejor planificación de sus finanzas.
- Aumento de la productividad, eficiencia y eficacia, porque los servicios de TI son más confiables y funcionan mejor para los usuarios.
- Genera economía financiera debido a la mejor gestión de los recursos y reducción de trabajo.
- Gestión de cambios más efectivos, permitiendo a la empresa mejor acompañamiento de los cambios y a mejor dirigir los cambios para su provecho.
- Mejoría de la satisfacción del usuario y del cliente con la TI.
- Mejoría de la percepción de la imagen de la marca por el cliente final.

1.3. Integración de estándares de control y monitoreo de servicios de TI

Existen varios estándares y buenas prácticas para la gestión de servicios de TI, pero no todos se basan y enfocan en lo mismo. Por eso es conveniente estudiarlas y saber elegir entre ellas la más adecuada para la empresa y el trabajo que se desea desarrollar en la misma.

Tabla 1: Comparación de estándares para la mejora de servicios de TI

Comparación

COBIT	El marco de trabajo de COBIT tiene un triple enfoque: • Enfocado a la administración: puesto que provee a la administración de una base de mejores prácticas con las cuales se pueden tomar decisiones de TI e inversión. • Enfocado a los usuarios de TI: Debido a la seguridad que les brinda para el control de objetivos y procesos • Enfocado a auditores: Debido a que permite identificar problemas de control de TI dentro de la infraestructura de TI de la compañía.
ITIL	ITIL se centra en brindar servicios de alta calidad para lograr la máximo satisfacción del cliente a un costo manejable. Además determina la forma de ejecutar procesos estándar ayudados de la tecnología para lograr la satisfacción de las personas y usuarios de los servicios de TI.
CMMI	El objetivo de CMMI es establecer una guía que permita a las organizaciones mejorar sus procesos y su habilidad para organizar, desarrollar, adquirir y mantener productos y servicios informáticos.
ISO 20000	ISO 20000 se centra principalmente en atender la administración de los servicios de TI. Crea el ambiente para cambiar de la cultura basada en tecnología a la basada en servicios y mejora la capacidad de absorber cambios rápidos.

En la Tabla 1 se refleja la convivencia o integración de los diferentes estándares en las diferentes áreas de negocio. Integrarlos todos puede resultar un rompecabezas debido a que se centran en partes diferentes de la gestión de servicios. COBIT se centra principalmente en el control y monitoreo de los servicios, CMMI generalmente se aplica al desarrollo del servicio o infraestructura en TI y el estándar ISO/IEC 20000 se aplica más a la calidad de los servicios.

1.4. Sistemas automatizados para la gestión de auditorías

En la actualidad son varias las compañías que en distintos lugares del mundo han desarrollado sistemas automatizados para la gestión de auditorías de servicios de TI, los cuales son ampliamente utilizados en prestigiosas empresas. Además, la gama de productos de este tipo es una cifra en constante incremento.

1.4.1. MKinsight ™

Es un sistema integral de gestión de auditoría altamente configurable, potente y fácil de usar. A partir de los auditores individuales al estado MKinsight Auditoría ™ Instituciones es fácil de usar, sencillo de implementar y asequible sea cual sea el tamaño de su equipo de auditoría (17).

Características principales:

- Base de datos integrados a la perfección individual.
- Totalmente configurable terminología y flujo de trabajo.
- Seguridad completa de datos cifrados.
- Interfaz de usuario amigable.
- Programa de Desarrollo Robusto.
- Multi-Lenguaje.

Funciones Claves:

- Planificación de la auditoría.
- Programación de Auditoría.
- Auditoría de Gestión.
- Informar el Rendimiento.
- Gestión del riesgo.
- Información Integral.
- Tiempo de grabación.
- Recomendación / Acción de Seguimiento.
- Cuestionarios en línea.
- Bibliotecas.
- MKinsight nuevo servicio de suscripción.

1.4.2. SoftExpertITSM

SE Audit es un sistema 100% WEB, multiusuario y multi-departamental, que incorpora herramientas de: organización, clasificación y búsqueda. Características que dan al producto simplicidad, agilidad, confiabilidad y eficiencia. La inversión en SoftExpert dependerá del tamaño de la empresa y la metodología de implementación (18).

El software proporciona el trabajo en equipo, a través de un práctico mecanismo de control de pendencias, denominado *Team Workflow*, que notifica vía e-mail, el momento exacto, a los responsables por actividades pendientes, exhibe estas pendencias y autoriza el registro de las firmas electrónicas y demás informaciones aplicables a cada etapa del proceso. Este mecanismo asegura la agilidad y el compromiso con el cumplimento de los plazos en todas las etapas del proceso de auditoría (18).

SE Audit le da a la organización total flexibilidad para establecer:

- Requisitos normativos que serán auditados.
- Criterios para cualquier tipo de auditoría adoptada por la organización.
- Equipos de auditores internos y externos que quedarán responsables por la conducción del proceso de auditoría.
- Cursos y competencias (CHA - Conocimientos, Habilidades y Aptitudes) necesarias para que cada auditor, sea él interno o externo, pueda asumir una responsabilidad en la realización de la auditoría, tal como anexar registros que comprueben su cualificación.
- Etapas las cuales cada clasificación de auditoría será sometida (18).

1.4.3. Optisoft Latinoamérica

Es una solución requerida de sistemas automatizados para la gestión y administración de los controles de TI, basado en el Marco de Control COBIT. Le suministra una herramienta automatizada, altamente modular e integrada que le permite interactuar y le asiste, brindándole a la organización operar con un alto grado de control y eficiencia en la implementación de la metodología de controles COBIT, así como en la gobernabilidad de TI (19).

Gestor COBIT es la aplicación que toda organización requiere para administrar y controlar en forma efectiva y eficiente los controles de las TI. Ha sido desarrollado utilizando todas las facilidades que permite la tecnología de punta en computación colaborativa y orientada totalmente al Web. Está presente en todos los dominios y procesos de la metodología COBIT, desde la planificación y organización del área de TI, hasta el monitoreo y control de los procesos de TI (19).

Características Principales

- Gestor COBIT opera en la Red Interna de la Empresa o Institución, por medio de un Navegador de Internet.

- Corre en Redes que van desde Micros, Minicomputadores y Mainframes.
- Cuenta con niveles de seguridad de acceso jerarquizado para el manejo de la información, esto significa que, los Colaboradores tienen derechos de acceso para solicitar, aprobar, consultar, editar o modificar información de acuerdo con el nivel de jerarquía del puesto y del tipo de usuario que tienen en la organización.
- Permite el control automatizado de la revisión y actualización del estado de cada proceso y de cada control de la Gestión y Administración de los Controles de TI, basado en el Marco de Control COBIT.
- El sistema permite que en todos los procesos y las autorizaciones sean realizadas en tiempo real, a través de Internet y en forma electrónica por las personas autorizadas.
- Cada proceso por sí mismo contiene una bitácora que se genera en forma automática y contiene toda la información para seguirle la pista a eventos que han afectado dicho proceso.
- Permite llevar un seguimiento completo de las auditorías y revisiones, tanto sobre las evaluaciones de brechas de los procesos, como del grado de madurez de dichos procesos.

Beneficios:

- Administración eficiente y efectiva de los controles de los procesos de TI.
- Permite identificar cómo los recursos de TI contribuyen efectivamente al logro de los objetivos del negocio.
- Administración eficiente y efectiva del proceso de la Administración y Gestión de los Controles de TI;
- Minimización drástica de la utilización del papel.
- No dependencia de herramientas ofimáticas para llevar a cabo todos los procesos de la metodología COBIT.
- Optimización del uso del tiempo del personal en los procesos.
- Reducción significativa en los costos y tiempos de proceso y errores que se pudieran presentar, haciendo dichos procesos más eficientes y permitiendo conocer además su estado en todo momento.
- Monitoreo en tiempo real de los estados de los procesos y los cumplimientos de las actividades y tareas.
- Generación de consultas e informes en tiempo real de problemas y el seguimiento a la solución de los mismos (19).

1.4.4. AUDITWorks™

AUDITWorks ™ ayuda en la preparación y documentación de cumplimiento de seguridad, salud y medio ambiente y otros tipos de auditorías que se basan en listas de verificación. El software proporciona una guía para la realización de auditorías, un marco en el que se registran los resultados de auditoría, incluyendo las capacidades de gestión de datos y protocolos para la evaluación de su cumplimiento (20).

Listas de control dispone de diversas AUDITWorks ™ que contiene preguntas que se pueden utilizar para auditar varios programas, incluyendo OSHA Proceso de Gestión de la Seguridad y las regulaciones de la EPA de Gestión de Riesgos del Programa. También puede utilizar el software para auditar otras regulaciones gubernamentales, estándares de la industria, o las normas propias de su empresa de seguridad, salud y medio ambiente. AUDITWorks ™ proporciona los medios para facilitar todo tipo de auditorías. Ha sido diseñado para ayudarle a gastar menos tiempo a aprender el software y más tiempo a la realización de la auditoría. Con su fácil paso a paso enfoque de proceder en la hoja de trabajo de auditoría, podrás llevar a cabo la auditoría en cuestión de minutos (20).

De los sistemas existentes en el ámbito internacional para la planificación y realización de auditorías de TI descritos anteriormente, se puede concluir que no son sistemas que satisfacen las necesidades específicas de un centro de soporte. Sin embargo, fueron seleccionadas un conjunto de características importantes para la definición de funcionalidades necesarias en un sistema basado en COBIT de producción nacional como son:

- Creación de un historial de auditorías y organización de las actividades de las mismas.
- Optimización del uso del tiempo del personal en los procesos de auditoría.
- Reducción significativa en tiempos de procesos y errores que se pudieran presentar, haciendo dichos procesos más eficientes y permitiendo conocer además su estado en todo momento.
- Monitoreo en tiempo real de los estados de los procesos y los cumplimientos de las actividades y tareas.
- Generación de reportes e informes en tiempo real de problemas y el seguimiento a la solución de los mismos.
- Utilización de una estructura arbórea para facilitar la navegación dentro del sistema.

1.5. Descripción de tecnologías, patrones, lenguajes de programación y herramientas.

La construcción de un sistema de calidad exige del uso de tecnologías de última generación, que permitan desarrollar con rapidez y eficacia. Tecnologías que brinden buenas librerías en lo que a graficación se refiere, las cuales deben ser poderosas.

Es necesario para obtener una buena aceptación por parte del cliente, lograr un sistema que sea eficiente, consuma lo menos posible de los recursos de la computadora y muestre los resultados esperados en el tiempo establecido, sin largos intervalos de espera por las salidas del mismo; para lo cual las siguientes tecnologías son las indicadas.

1.5.1. Metodología de Desarrollo XP

Extreme Programming (XP por sus siglas en inglés) es la primera metodología ágil y la que le dio conciencia al movimiento actual de metodologías ágiles. Centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (21)

En la figura 1.1 se muestra la evolución de los largos ciclos de desarrollo en cascada (a) a ciclos Iterativos más cortos (b) y a la mezcla que hace XP.

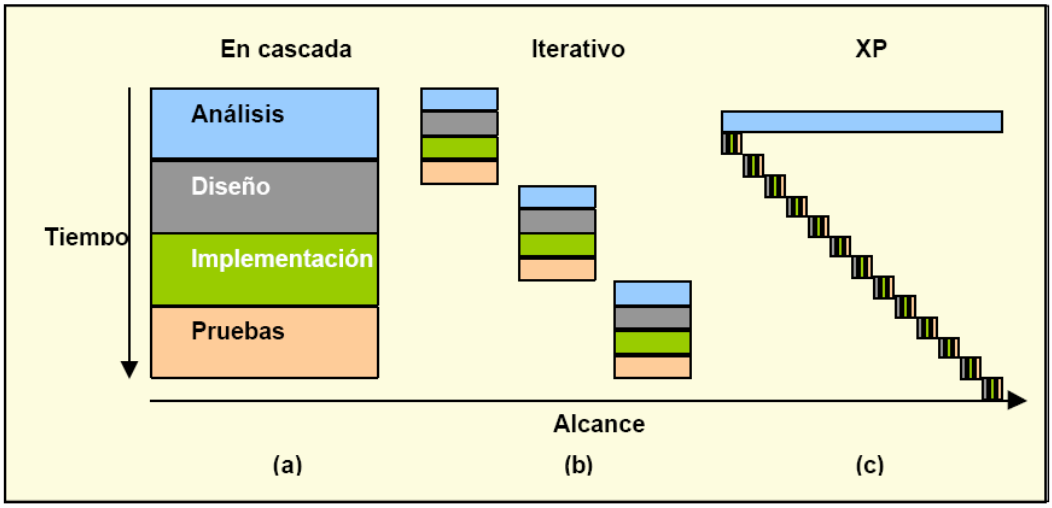


Figure 1-1 Ejemplo del funcionamiento de la metodología XP

1.5.2. IDE de Desarrollo: Intellij IDEA 11.0

Es un entorno de desarrollo Java creado por Jet Brains del que existen dos distribuciones: Community Edition (Open source) y Ultimate (comercial).La primera versión de Intellij IDEA fue lanzado en enero de 2001,y en ese momento era el único disponible JavaIDE de código avanzado y capacidades de navegación de código de refactorización integradas. Sus creadores definen este IDE como el más inteligente del mundo. La mayoría de gente que lo prueba lo define como el mejor entorno de desarrollo Java que existe. Soporta varias tecnologías como: Groovy, Android, JavaScript, Struts, Spring, Hibernate, JSF y otros. Las principales ventajas son el autocompletado de código. Integración con sistemas de control de versiones y la contención de un amplio set de plugins. Produce sensación de fiabilidad y robustez muy superior a otros entornos. Además es una herramienta de refactorización extremadamente inteligente. Este entorno tiene como dificultad que es propietario (22).

1.5.3. Framework: Grails 2.1

Grails es un framework de aplicaciones web dinámica basada en Java y Groovy, aprovechando lo mejor delas API de raza incluyendo Spring, Hibernate y Site Mesh. Grails aporta a los desarrolladores de Java y Groovy las alegrías de la convención basada en rápido desarrollo, mientras que lo que les permite aprovechar sus conocimientos y aprovechar las probadas y potente API de desarrolladores Java han estado utilizando durante años (23).

1.5.4. Lenguaje de Programación: Groovy 1.8

Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. La especificación JSR 241 se encarga de su estandarización para una futura inclusión como componente oficial de la plataforma Java. Usa una sintaxis muy parecida a Java, 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 byte code generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente en cualquier aplicación Java. Groovy puede usarse de manera dinámica como un lenguaje de scripting (24).

1.5.5. Herramienta CASE: Visual Paradigm

Visual Paradigm for UML (VP-UML) es una herramienta case para el desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos (25).

VP-UML brinda apoyo a UML 2, SysML y Business Process Modeling Notation (BPMN) del Object Management Group (OMG). Además del apoyo de modelado, proporciona capacidades de generación de informes y códigos de ingeniería. Se puede revertir diagramas de ingeniería de código, y proveer de ida y vuelta la ingeniería para varios lenguajes de programación. Apoya la gestión de requisitos, incluyendo los casos de uso, diagramas SysML requisitos y análisis textual. VP-UML soporta tanto Diagramas Entidad Relación (ERD) y Diagramas de asignación relacional de objetos (ORMD). ERD se utiliza para modelarla base de datos relacional (25).

1.5.6. Java Development Kit (JDK)

Java Development Kit (JDK por sus siglas en inglés) es un entorno de ejecución de Java completa, generalmente llamado un tiempo de ejecución privada, debido al hecho de que está separado de la "regular" JRE y tiene contenidos extras. Se compone de una máquina virtual Java y todas las librerías de clases presentes en el entorno de producción, así como bibliotecas adicionales sólo útiles para los desarrolladores, tales como las bibliotecas de internacionalización y las bibliotecas de DL. Puede instalarse en una computadora local o en una unidad de red. En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación (26).

El Java Development Kit (JDK) es una implementación de cualquiera de Java SE, Java EE Java o plataformas ME publicados por Oracle Corporation en la forma de un producto binario dirigido a los desarrolladores de Java en Solaris, Linux, Mac OSX o Windows. Desde la introducción de la plataforma Java, ha sido, con mucho, el más utilizado Software Development Kit (SDK). El 17 de noviembre de 2006, Sun anunció que iba a ser publicado bajo la licencia GNU General Public License (GPL), por lo que es software libre. Esto sucedió en gran parte, el 8 de mayo de 2007, cuando Sun contribuyó con el código fuente para el Open JDK (26).

1.5.7. Patrón Arquitectónico: MVC

Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa la representación de la información a partir de la interacción del usuario con él. El modelo consta de datos de aplicaciones y reglas de negocio, y la entrada de controlador de media, convirtiéndolo a los comandos para el modelo o la vista. Una vista puede ser cualquier representación de salida de datos, como un gráfico o un diagrama. Múltiples vistas de los mismos datos son posibles, como un gráfico circular para la gestión y una vista tabular para los contadores. Las ideas centrales detrás de MVC son reutilización del código y la separación de preocupaciones (27).

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Muchos de los sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en líneas generales del MVC corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí (27).

1.5.8. Sistema de Control de Versiones: SVN

SVN es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto en forma más o menos ordenada. Tiene una arquitectura cliente servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo y funciona más o menos así. En algún servidor se monta un repositorio SVN. En este lugar se van a registrar los cambios (revisiones) y los logs que se vayan generando. El cliente de SVN se baja una copia local de alguna revisión

(generalmente la última), el desarrollador hace los cambios y los sube al servidor para que estén disponibles para los otros desarrolladores (además de generar un log con un comentario de que cosa modificó) (28).

1.5.9. Sistema de Gestión de Bases de Datos: PostgreSQL 9.1

PostgreSQL es un sistema de base de datos objeto-relacional (ORDBMS) disponible para muchas plataformas, incluyendo Linux, Free BSD, Solaris, Microsoft Windows y MacOSX. Se distribuye bajo la Licencia PostgreSQL, que es un MIT licencia tipo, por lo que es software libre y de código abierto. Es desarrollado por el Grupo de Desarrollo Global de PostgreSQL, que consiste en un puñado de voluntarios empleados y supervisados por empresas como Red Hat y Enterprise DB. Implementa la mayoría del SQL, es compatible con ACID (Atomicidad, coherencia, aislamiento y durabilidad), es completamente transaccional (incluyendo todas las declaraciones DDL), tiene tipos extensibles de datos, operadores, métodos de índice, funciones, agregados, lenguajes de procedimiento, y tiene un gran número de extensiones escritas por terceros (29).

PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL (29).

1.5.10. Generador de Reportes Dinámico (GDR 1.7)

El Generador Dinámico de Reportes (GDR) ofrece una solución de reporte que le brinda a los negocios una herramienta reditual e inteligente, destinada a mejorar la rapidez y calidad de la adopción de decisiones en todos los niveles de una organización. Las instituciones requieren una manera centralizada de crear, administrar y entregar reportes en tiempo real. El sistema permite a las organizaciones encargarse de este proceso y les proporciona a los desarrolladores herramientas para confeccionar reportes e implementar soluciones personalizadas de informes para usuarios individuales en toda una organización. Constituye una de las herramientas que posibilitan la inteligencia de negocios ya que genera vistas agregadas de datos para mantener a la gerencia informada sobre el estado de su negocio (30).

Es una solución abierta y extensible para los informes administrados. Su arquitectura flexible permite a los programadores de software y a las empresas integrar el conjunto de herramientas con sus sistemas

heredados, portales empresariales o aplicaciones personalizadas. Combina las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales o de escritorio. Se evidencia en las características y potencialidades del diseñador de reportes, que se extiende al máximo las posibilidades de diseño del usuario. Este diseñador permite personalizar al detalle la salida de los reportes y la forma en que la información será visualizada (30).

1.6. Conclusiones del capítulo

Debido al estudio realizado sobre el estado del arte de las principales herramientas que existen en el mundo para la gestión de auditoría de servicios de TI, se llega a la conclusión de que:

- La mayoría son herramientas potentes, pero costosas al mismo tiempo. Además, se concluye que ninguno de ellos satisface las necesidades específicas de un Centro de Soporte.
- A partir de una comparación entre los diferentes estándares y buenas prácticas que existen en el mundo para mejorar la calidad de los servicios de TI que se brindan a la sociedad, se encontró que COBIT es el más indicado para realizar la auditoría debido a que este estándar está enfocado en gran medida a los auditores y permite identificar problemas de control de TI dentro de la infraestructura de una empresa.
- Se determinó la utilización de la metodología XP debido a que es una metodología ágil que plantea un nuevo paradigma a través del cual un proyecto de software se realiza de una forma diferente al sugerido por metodologías pesadas. Señalan la posibilidad de centrar sus esfuerzos en la implementación para lograr resultados rápidos sin sacrificar la calidad de los mismos.
- Se definieron las tecnologías y herramientas que se emplearán en la realización de este trabajo las cuales permitirán una implementación eficiente en corto tiempo. Estas fueron elegidas debido a que la solución será integrada a un sistema en construcción, el cual se realiza actualmente con las tecnologías anteriormente descritas.

CAPÍTULO 2. **PROPUESTA DE SOLUCIÓN**

En el presente capítulo se procede a describir como se realizará la auditoría de servicios de TI en el Centro de Soporte UCI. En el mismo se muestra un modelo conceptual que explica brevemente como sería el proceso de auditoría, así como los roles y responsabilidades del sistema. Se mencionan además los requisitos funcionales y no funcionales del sistema, describiéndose a su vez las historias de usuario, y finalmente se describen los patrones de diseño, estándares de codificación, así como el modelo la base de datos y las tarjetas CRC.

2.1. Modelo conceptual.

Debido a que los procesos del negocio no se encuentran bien definidos, se propone construir un modelo conceptual que proporciona una vista estructural del sistema donde se describen las entidades implicadas en el proceso de auditoría de servicios de TI.

El sistema debe permitir crear una o más auditorías, donde el equipo auditor asignado a la misma, proveniente de una entidad certificadora, podrá elegir el tipo de auditoría relacionada con los servicios que se prestan en el Centro de Soporte UCI, que se realizará en dicho Centro. A esta auditoría se le puede asignar uno o más documentos asociados a las pautas y políticas por la que se regirá el equipo para evaluar a dicha institución. El sistema permitirá mostrar el progreso de la auditoría de acuerdo con el tiempo de inicio y fin de la misma, y luego de terminado dicho tiempo la auditoría pasará automáticamente a un historial para su posterior consulta. Cada auditoría contará de varios parámetros para su evaluación, y el auditor podrá emitir criterios y reportes tanto gráficos como en formato PDF. Durante la ejecución de la auditoría el auditor tendrá la posibilidad de mandar notificaciones, agregar riesgos, no conformidades y sugerencias a dicha auditoría.

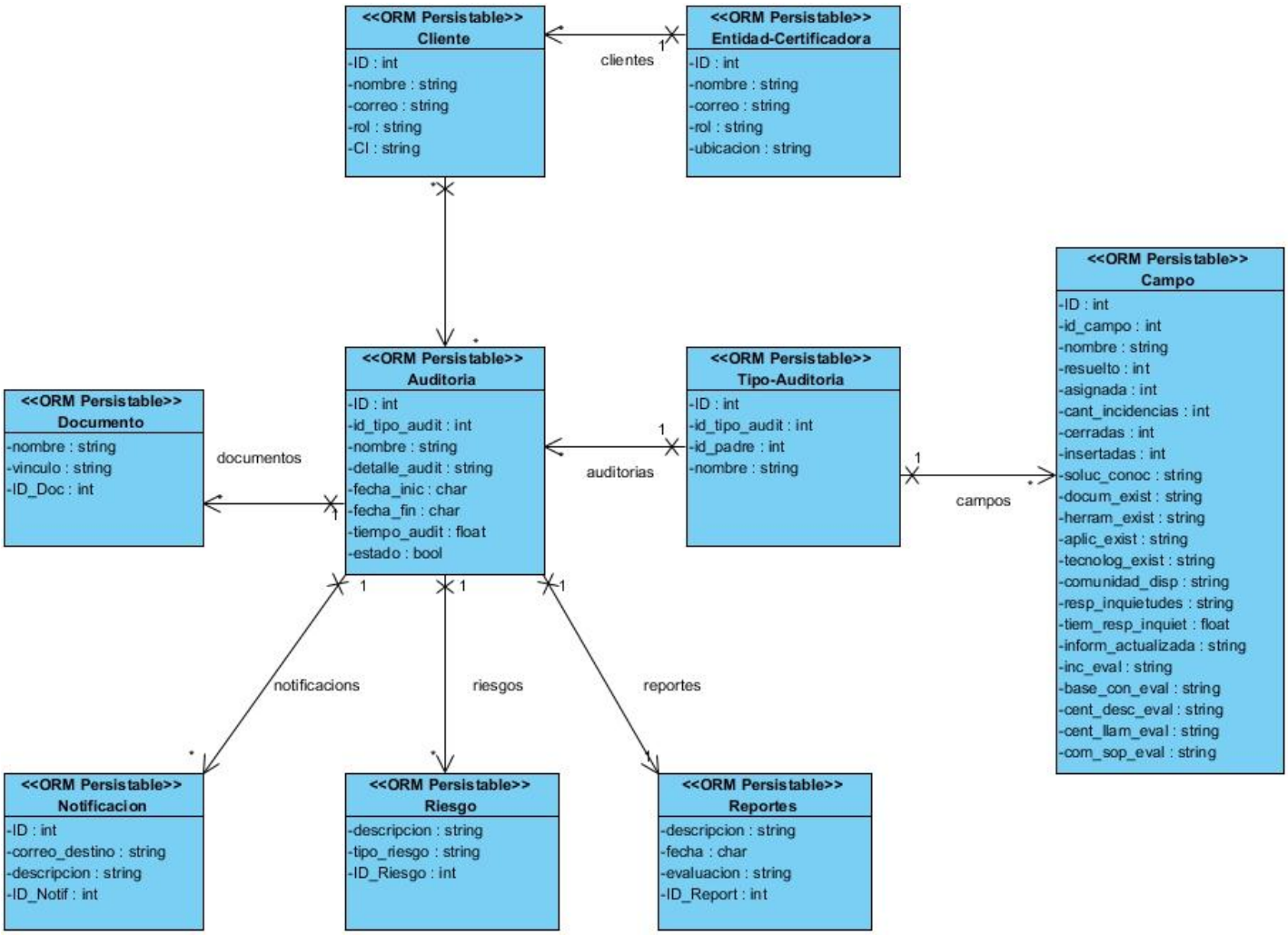


Figure 2-1: Modelo conceptual

2.2. Roles y responsabilidades

Tabla 2: Roles y responsabilidades

Roles	Responsabilidades
Director	Acceso a todos los datos y opciones del sistema. Permisos para finalizar una auditoría.

	▪ Permisos para eliminar una auditoría. ▪ Permisos para administrar en el sistema usuarios y roles con sus niveles de acceso.
Jefe de Grupo	▪ Permisos para modificar un reporte de auditoría. ▪ Permisos para declarar un documento final. ▪ Permisos como auditor.
Auditor	▪ Permisos para ver historial de auditorías en que no esté trabajando. ▪ Permisos para eliminar un documento en una auditoría en que forme parte del equipo de trabajo. ▪ Permisos para generar reportes y notificaciones de auditorías.

En la Tabla 2 se describen los roles que arroja la metodología XP con el objetivo de organizar quienes se encargaran de realizar cada una de las actividades que deben realizarse durante el transcurso del proyecto.

2.3. Requisitos Funcionales

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente (31).

Tabla 3: Requisitos funcionales

Requisitos Funcionales	
RF 1 Gestionar la planificación de la auditoría.	RF 7. Mostrar resultados de auditorías anteriores.

RF 1.1 Crear auditoría.	RF9. Permitir anexar cualquier tipo de documento.
RF 1.2 Modificar auditoría.	RF 10 Gestionar reportes.
RF 1.3 Eliminar auditoría.	RF 10.1 Crear reportes.
RF 1.4 Buscar auditoría.	RF 10.2 Modificar reportes.
RF 2. Gestionar equipo auditor.	RF 10.3 Eliminar reportes.
RF 2.1 Crear equipo auditor.	RF 10.4 Buscar reportes.
RF 2.2 Modificar equipo auditor.	RF 11. Proponer buenas prácticas, observaciones y no conformidades.
RF 2.4 Eliminar equipo auditor.	RF 12. Gestionar clientes
RF 2.5 Buscar equipo auditor.	RF 12.1 Crear clientes
RF 3. Gestionar documentos asociados.	RF 12.2 Modificar clientes
RF 3.1 Crear documentos asociados.	RF 12.3 Eliminar clientes
RF 3.2 Modificar documentos asociados.	RF 12.4 Buscar clientes
RF 3.3 Eliminar documentos asociados.	RF 13. Gestionar organismos certificadores.
RF 3. 4 Buscar documentos asociados.	RF 13.1 Crear organismos certificadores.
RF 4. Mostrar el tiempo y los pasos en que se encuentra la auditoría.	RF 13.2 Modificar organismos certificadores.
RF 5. Mostrar registro de evidencias y certificados en la realización de auditorías.	RF 13.3 Eliminar organismos certificadores.
RF 6. Mostrar registros de criterios de auditorías.	RF 13.4 Buscar organismos certificadores.

En la Tabla 3 se identifican los requisitos funcionales definidos por el cliente que serán desarrollados en el sistema.

2.4. Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente a penas se aplican a características o servicios individuales del sistema. (32)

Usabilidad

RNU 1 El usuario administrador del sistema debe tener al menos conocimientos básicos de la informática.

RNU 2 El sistema debe presentar un acceso fácil y rápido, para facilitar el uso del mismo por usuarios con pocos conocimientos en el campo de la informática.

RNU 3 Para que usuarios no experimentados adquieran los conocimientos básicos para el trabajo en el sistema se les deberá impartir una capacitación de al menos 7 días.

Soporte

RNSO 1 Luego de culminado el proceso de desarrollo se realizarán los procesos de despliegue, capacitación y mantenimiento del sistema.

RNSO 2 El personal encargado del despliegue del sistema debe contar con el nivel técnico requerido, obtenido mediante adiestramiento de servicio en no menos de 25 días.

RNSO 3 Las deficiencias detectadas en la aplicación deberán notificarse al equipo de mantenimiento, dando este una respuesta en no más de 15 días.

Diseño e implementación

RNDI 1 Se utilizará PostgreSQL en su versión 8.4 o superior como sistema gestor de bases de datos, dado que el cliente así lo exige.

RNDI 2 Se utilizará Groovy como lenguaje de programación y Grail como framework integrador, dado que el cliente así lo exige.

RNDI 3 Se utilizará como arquitectura del sistema Modelo- Vista- Controlador, dado que el cliente así lo exige.

RNDI 4 Se utilizará Intellij Idea como IDE para el desarrollo del sistema, dado que el cliente así lo exige.

RNDI 5 Se utilizará Visual Paradigm como herramienta de modelado.

RNDI 6 Para el diseño de las páginas se utilizarán las hojas de estilo en cascadas (CSS por sus siglas en inglés), lenguaje de marcas (o etiquetas) hipertexto (HTML por sus siglas en inglés), Lenguaje de marcas hipertexto extensible (XHTML por sus siglas en inglés), JavaScript, JQuery y JQuery UI.

Funcionamiento

Software:

RNFO 1 Para el funcionamiento de los servidores se puede usar cualquier sistema operativo, previa instalación de la máquina virtual de Java, Windows 95/98, Windows NT (service pack 5 o superior), Windows 2000, Windows XP o superior, Macintosh, Linux o compatible UNIX.

RNFO 2 Para el funcionamiento en la PC cliente será necesario un navegador Web que soporte HTML 5, CSS 3, JavaScript y AJAX.

RNFO 3 Máquina Virtual de Java (MVJ) 1.4 o superior.

Hardware:

RNFO 3 El servidor para la aplicación deberá tener las siguientes características mínimas:

- Tarjeta de red y capacidad de conectividad para atender alrededor de 300 peticiones por segundo.
- 1 GB de memoria RAM.
- Capacidad de 5 GB de disco duro.
- Procesador Pentium 300MHz o superior.

RNFO 4 El servidor para la base de datos deberá tener las siguientes características mínimas:

- Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior;
- Tarjeta de red o capacidad de conectividad;
- 1 GB de memoria RAM;
- Capacidad de 50 GB de disco duro.

RNFO 5 Las estaciones de trabajo deberán cumplir los siguientes requisitos mínimos:

- Procesador Intel Celeron E3200 a 2.40 Ghz, equivalente o superior;
- 256 MB de memoria RAM.

Seguridad

- Las peticiones de modificación de información realizadas por los clientes solo se podrán hacer mediante el método POST.

Integridad:

RNS 5 Se harán validaciones de la información en el servidor y en el cliente mediante expresiones regulares y restricciones contra ataques de inyección HTML y SQL.

RNS 6 Se protegerá la información entrada contra ataques CSRF (Cross Site Request Forgery).

Disponibilidad:

RNS 7 El sistema deberá estar disponible las 24 horas del día durante los 365 días del año, salvo sea necesario su mantenimiento o actualización.

Fiabilidad

RNF 1 La respuesta del sistema ante una búsqueda bajo criterios y con 270 usuarios conectados no debe exceder los 500 ms.

RNF 2 Ante una inserción de datos, vista de detalles, modificación o eliminación el sistema y con 200 usuarios conectados debe responder en no más de 300 ms.

RNF 3 El sistema debe responder en un promedio de 800 ms la petición de un reporte con 50 usuarios conectados.

RNF 4 El sistema deberá soportar transacciones de cerca de 300 usuarios conectados simultáneamente.

RNF 5 El sistema podrá ser usado de forma extendida permitiendo un tiempo de mantenimiento cada 6 meses.

RNF 6 Ninguna información ingresada en el sistema será eliminada físicamente de la base de datos, independientemente de su inexistencia para el sistema. Permitirá la recuperación de la información de la base de datos, a partir de respaldos, o salvas realizadas.

RNF 7 Se realizarán salvas periódicas de la base de datos para prevenir la pérdida de información.

Eficiencia

RNE 1 El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria.

RNE 2 Se respetarán buenas prácticas de programación para incrementar el rendimiento en operaciones costosas.

Interfaz de usuario

RNIU 1 Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la interpretación correcta de la información.

RNIU 2 La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error.

RNIU 3 El diseño de la interfaz del sistema responderá a la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.

Interconexión

RNI 1 La PC cliente se comunicará mediante el Protocolo de Transferencia de Hipertexto (HTTP por sus siglas en inglés) y Protocolo de Transferencia de Hipertexto Seguro (HTTPS por sus siglas en inglés) con el servidor de aplicaciones, éste se comunicará mediante el Protocolo de Control de Transmisión/Protocolo de Internet (TCP\IP por sus siglas en inglés) con el servidor de bases de datos.

RNI 2 Los datos de usuarios y demás datos necesarios para el sistema podrán ser extraídos de los servicios web brindados por el UDDI (Directorio de Servicios web de la UCI).

2.5. Historias de usuarios

Las historias de usuario tienen el mismo propósito que los casos de uso, pero no son lo mismo. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Por tanto, serán descripciones cortas

y escritas en el lenguaje del usuario, sin terminología técnica. Su principal diferencia está en el nivel de detalle pues las historias de usuarios solo brindan la información necesaria para que los programadores estimen un tiempo ideal de desarrollo y a la hora de implementar la historia de usuario, deberán reunirse con el cliente y ampliar los detalles de la misma (33).

Las Historias de usuarios cuentan con la estructura que se explica a continuación:

Número: A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Nombre: Nombre descriptivo de la HU.

Prioridad en Negocio: Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. 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 HU luego de analizarla. (Alto, Medio o Bajo).

Puntos Estimados: Unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU. 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 HU. 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 HU.

Descripción: Descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

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

A continuación se muestran algunas de las historias de usuario más relevantes y las restantes se pueden visualizar en el Anexo 1.

Tabla 4: Historia de Usuario #1

Historia de Usuario

Número: 1	**Nombre de la Historia de Usuario:** Gestionar la planificación de la auditoría.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 y ½ semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semana

Descripción:

1. El sistema muestra la interfaz correspondiente a las auditorías.
2. El auditor pulsa en el botón “Adicionar”.
3. El sistema muestra un conjunto de parámetros correspondientes a la auditoría.
4. El auditor completa los datos y la auditoría se la misma se adiciona.
5. El sistema muestra un cuadro de diálogo, informándole al auditor que la auditoría ha sido adicionada correctamente.
6. El auditor tiene la posibilidad de modificar la auditoría añadida, dando doble clic encima de la misma. Además de eliminarla pulsando un botón existente a la derecha de la auditoría añadida.

Observaciones: Si los datos de creación de la auditoría no son válidos la aplicación mostrará mensajes con los errores correspondientes.

Prototipo de interfaces:

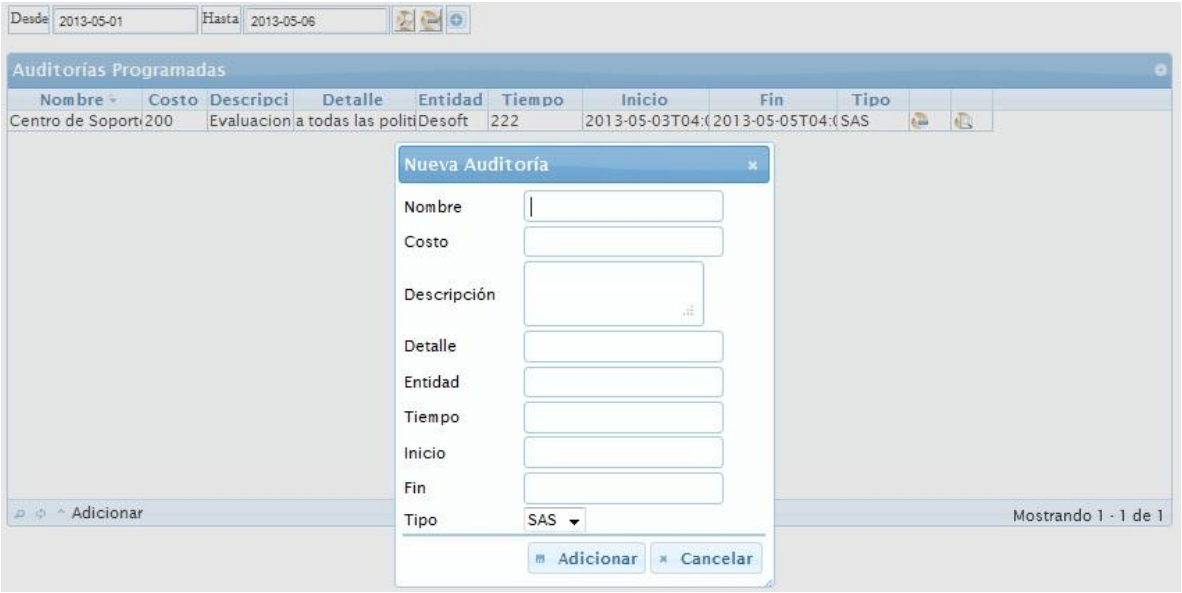


Tabla 5 Historia de usuario #2

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Gestionar equipo auditor.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** ½ semana
Riesgo en desarrollo: Alto	**Puntos reales:** ½ semana
Descripción: 1. El sistema muestra la interfaz correspondiente a la gestión del equipo auditor. 2. El auditor pulsa en el botón “Adicionar”. 3. El sistema muestra una interfaz, que le permitirá al auditor seleccionar de los clientes adicionados en el	

sistema, los que pertenecerán al equipo auditor.

4. El auditor selecciona los clientes y estos se adicionan a la auditoría.
5. El sistema muestra como resultado una tabla con los clientes que han sido añadidos, donde el usuario puede asegurarse de que la operación se ha realizado exitosamente.

Observaciones: El auditor solo podrá adicionar clientes al equipo auditor que estén insertados en el sistema.

Prototipo de interfaces:

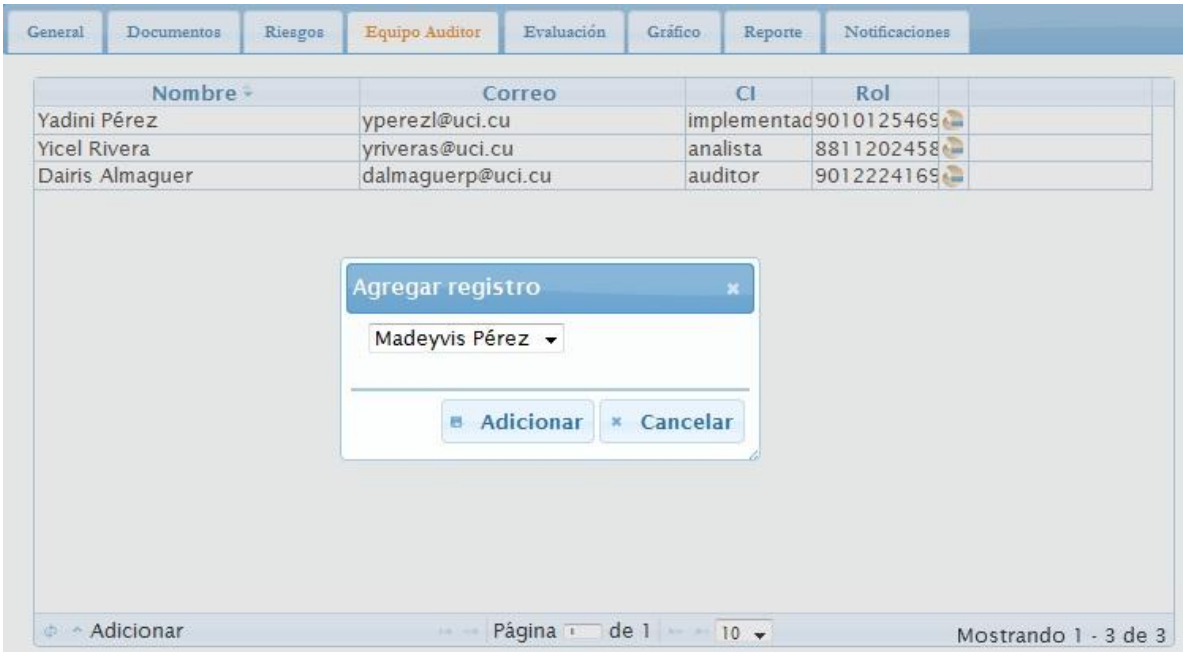


Tabla 6 Historia de usuario #3

Historia de Usuario	
Número: 3	**Nombre de la Historia de Usuario:** Gestionar documentos asociados.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 2

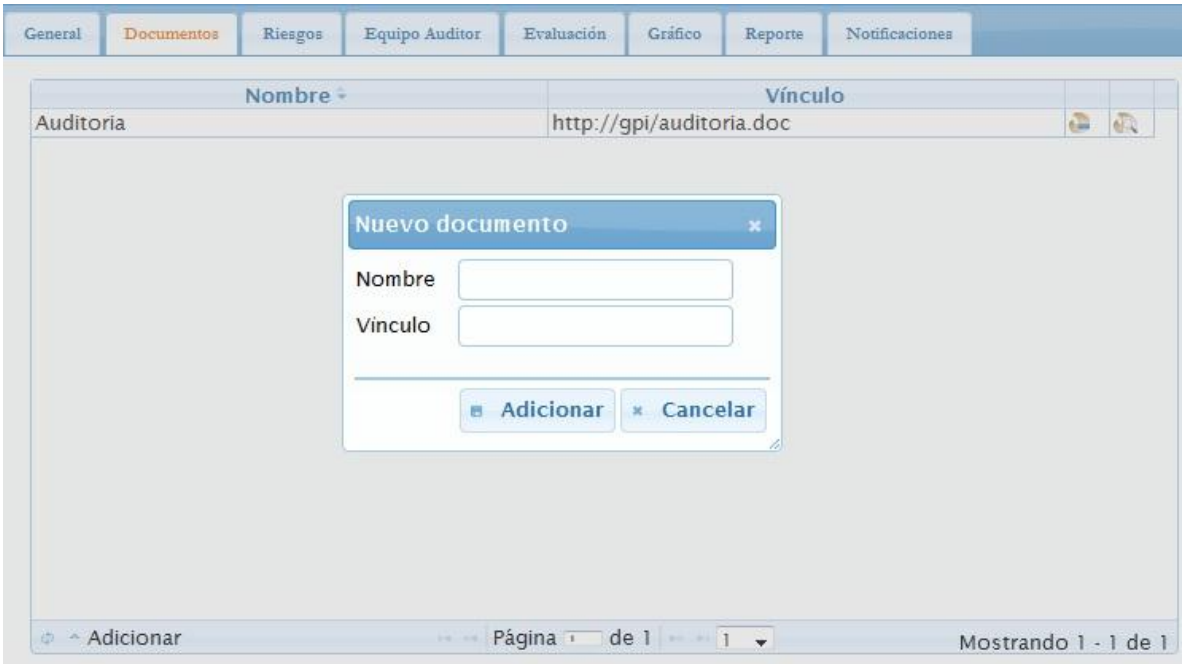
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 1 semana

Descripción:

1. El sistema muestra la interfaz correspondiente a la gestión de documentos.
2. El auditor pulsa el botón “Adicionar”.
3. El sistema muestra un cuadro de selección, para que el auditor añada el/los documento(s) asociados a la auditoría.
4. El auditor inserta el link en el que se encuentra el documento y este se añade a la auditoría.
5. El sistema muestra como resultado una tabla con los documentos que han sido añadidos, donde el usuario puede asegurarse de que la operación se ha realizado exitosamente.

Observaciones: Si los datos de creación del documento no son válidos la aplicación mostrará mensajes con los errores correspondientes.

Prototipo de interfaces:



2.6. Plan de iteraciones

En la metodología XP, la creación del sistema se divide en etapas para facilitar su realización. Por lo general, los proyectos constan de 3 iteraciones. Para cada iteración se define un módulo o conjunto de historias que se van a implementar. Al final de cada iteración se obtiene como resultado la entrega del módulo correspondiente, el cual debe haber superado las pruebas de aceptación que establece el cliente para verificar el cumplimiento de los requisitos. Las tareas que no se realicen en una iteración se tienen en cuenta para la próxima iteración, donde se define, junto al cliente, si se deben realizar o si deben ser removidas de la planeación del sistema. (34)

Tabla 7: Plan de iteraciones

Historias de Usuarios	Duración Estimada	Iteración	Duración Iteración
Gestionar equipo auditor.	7 semanas	1	5 semanas
Gestionar documentos asociados.			
Gestionar reportes.			
Proponer buenas prácticas, observaciones y no conformidades.			
Gestionar clientes			
Gestionar organismos certificadores	9 semanas	2	6 semanas
Mostrar resultados de auditorías anteriores.			
Permitir anexar cualquier tipo de documento.			

Gestionar Riesgos			
Mostrar registro de evidencias y certificados en la realización de auditorías.			
Gestionar notificaciones	5 semanas	3	6 semanas
Mostrar el tiempo y los pasos en que se encuentra la auditoría.			
Mostrar registros de criterios de auditorías.			
Gestionar equipo auditor.			

En la Tabla 7 se describe el plan de iteraciones del proyecto. Estas deben variar entre 1 y 3 semanas en las cuales debe haber una entrega de los avances del producto, los cuales deberán ser completamente funcionales.

2.7. Plan de entrega de versiones

Es un artefacto generado como parte de la planificación que permite realizar una organización del tiempo de las entregas, marcando el final de cada una de las iteraciones permitiéndoles así a los programadores una guía para el desarrollo en tiempo de la solución.

Tabla 8: Plan de entrega de versiones

Iteración	Iteración 1	Iteración 2	Iteración 3
Cantidad de Historias de Usuario	5	5	4
Fecha Entrega	Del 5 al 10 de Febrero	Del 15 al 18 de Marzo	Del 19 al 22 de Abril

En la Tabla 8 se muestra el plan de entrega de versiones el cual lo arroja también la metodología con el objetivo que organizar y definir las fechas en las que se hará entrega de las funcionalidades del sistema según las iteraciones

2.8. Tarjetas CRC "Clase, responsabilidad y colaboración"

Las tarjetas CRC significan las relaciones que existen entre Clases, Responsabilidades y Colaboraciones, permitiendo al equipo de desarrollo entender la posición de cada objeto dentro del espectro del software, con quien interactúan y a cuales clases afectan directa o indirectamente. Esto permite que exista un esclarecimiento continuo de la realización de cada clase a medida que esta se implementa ayudando al programador a centrarse en su trabajo.

Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño. (35)

A continuación se muestran algunas de las Tarjetas CRC más relevantes y las restantes se pueden visualizar en el Anexo 2.

Tabla 9: Tarjeta CRC # 1

Tarjeta CRC	
Clase: AuditoriaServicios	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere una auditoria de elemento de la configuración: - String nombre - Stringdescripcion - Stringdetalle_auditoria - Date fecha_inicio	

Date fecha_fin Stringentidad_a_auditar doubletiempo_auditar double costo boolean estado TipoAuditoria	

Tabla 10: Tarjeta CRC # 2

Tarjeta CRC	
Clase: AuditoriaServiciosController	
Responsabilidades	**Colaboraciones**
• Es la encargada de: createChart(CategoryDatasetcategorydataset) createDataset(ev) Grafico(intauthID, int category) listJSON(string sidx, string sord, int rows, int page) TiposAuditoria() Porcent(Date a,Date b) Details(intid) Evaluate(EvalSASobj) DocumentForAS(string sidx, string sord, int rows, int page, int id) AddToAS(Documentodoc, intauthID) DeleteRelAsDoc(intautID, int id) RiesgoForAS(string sidx, string sord, int rows, int page, int id) AddRiesgoToAS(intauthID, Riesgos r)	AuditoriaServicios Riesgos Usuarios Documentos TipoAuditorias EvalSAS

- DeleteRelRiesgoAsDoc(intautID, int id) - UsuariosForAS(string sidx, string sord, int rows, int page, int id) - AddUsuarioToAS(intuserID, intauthID) - DeleteRelUsuarioAsDoc(intautID, int id) - Usuarios() • Además de las funcionalidades que genera de forma automática el framework: - list(Integermax) - create() - save() - show(Long id) - edit(Long id)) - update(Long id, Long version) - delete(Long id)	

Tabla 11: Tarjeta CRC # 3

Tarjeta CRC	
Clase: AuditoriaServicios	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere un organismo certificador de elemento de la configuración: - String nombre - Stringdireccion - String correo - String rol	

2.9. Patrones de diseño

Un patrón de diseño es:

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.
- La forma de un diagrama de objeto o de un modelo de objeto.

Motivación de los Patrones

- Capturan la experiencia y la hacen accesible a los no expertos.
- El conjunto de sus nombres forma un vocabulario que ayuda a que los desarrolladores se comuniquen mejor.

Lenguajes de patrones

- Ayudan a la gente a comprender un sistema más rápidamente cuando está documentado con los patrones que usa.
- Los patrones pueden ser la base de un manual de ingeniería de software.
- Si el software se convierte en una ingeniería, las prácticas exitosas deben ser documentadas sistemáticamente y ampliamente difundidas.
- Facilitan la reestructuración de un sistema tanto si fue o no concebido con patrones en mente.

Reutilización:

- Los patrones de diseño soportan la reutilización de arquitecturas software.
- Los armazones soportan la reutilización del diseño y del código.

El software cambia:

- Para anticiparse a los cambios en los requisitos hay que diseñar pensando en qué aspectos pueden cambiar.
- Los patrones de diseño están orientados al cambio.

2.9.1. Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa *(Patrones Generales de Software para Asignar Responsabilidades)*. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (36).

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, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad (36).

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 (36).

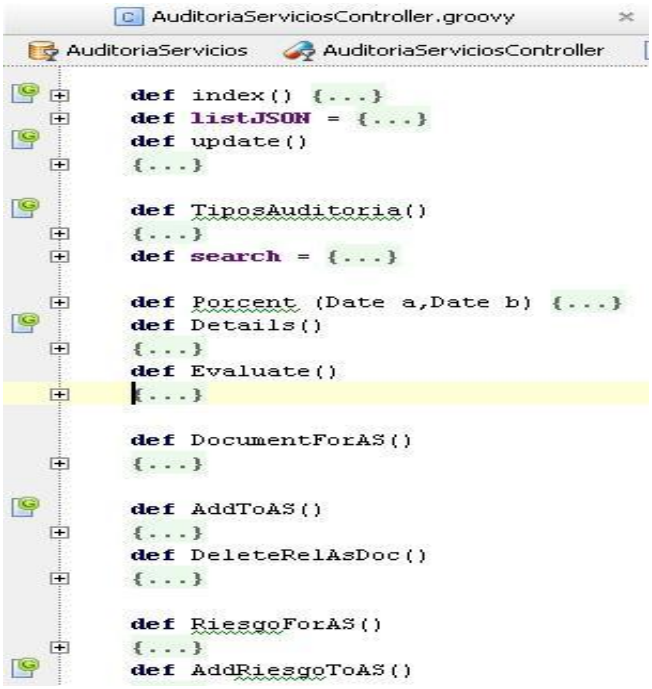


Ilustración 1: Ejemplo del uso del patrón creador en el sistema

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 la que los envía a las distintas clases según el método llamado; sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (36).

Alta cohesión:

La información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión, compartirá la responsabilidad de una operación, con otras clases.

Bajo acoplamiento:

Es la idea de tener las clases lo menos ligadas 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. Los beneficios de este patrón es que no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar (36).

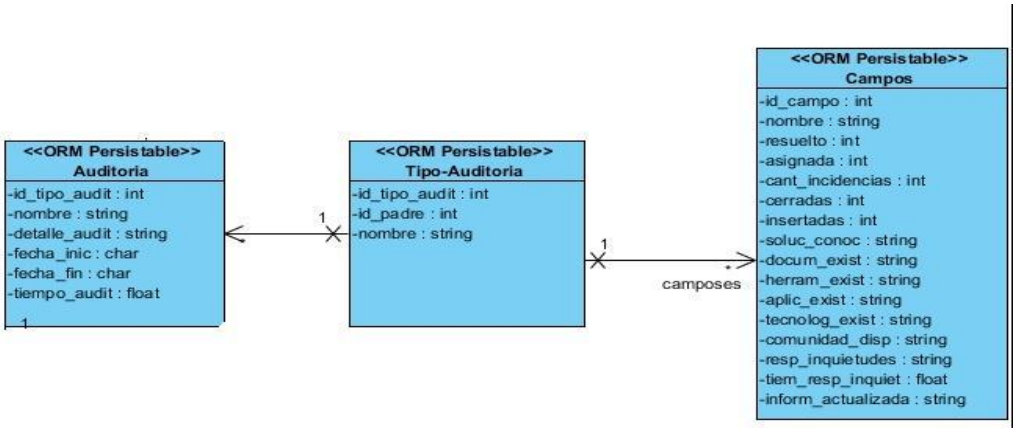


Ilustración 2: Ejemplo del uso del patrón de bajo acoplamiento en el sistema

2.9.2. Patrón de arquitectura

Patrón de arquitectura MVC:

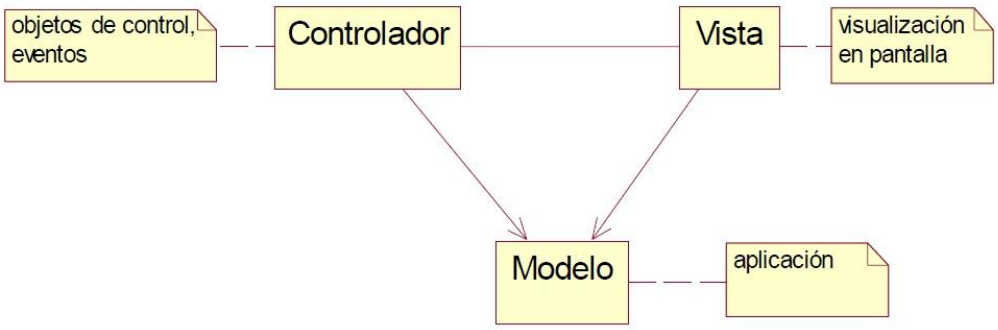


Ilustración 3: Estructura del patrón Modelo-Vista-Controlador

Modelo:

AuditoriaServicios.groovy

AuditoriaServicios | AuditoriaServiciosController | AuditoriaServicios Views | Tests

```
package my

class AuditoriaServicios {
    String nombre
    String descripcion
    String detalle_auditoria
    Date fecha_inicio
    Date fecha_fin
    String entidad_a_auditar
    double tiempo_auditar
    double costo
    boolean estado
    TipoAuditoria tipo_auditoria

    static constraints = {...}
    static mapping = {...}

    static hasMany = [usuarios: Usuario, documentos: Documento, riesgos: Riesgo]
    static belongsTo = [ Usuario]

}
```

Ilustración 4 : Ejemplo del componente modelo en el sistema

Vista:

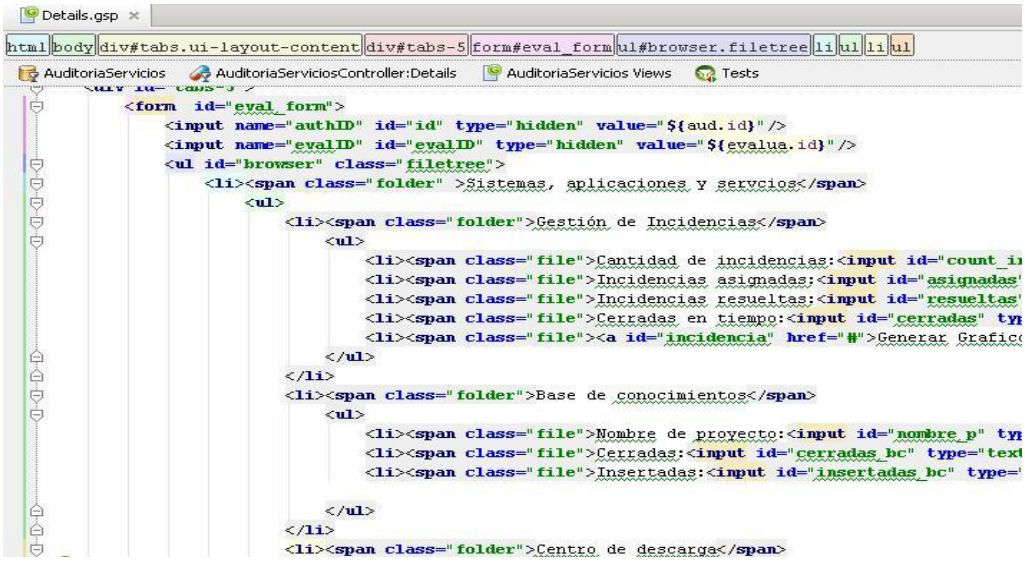


Ilustración 5: Ejemplo del componente vista en el sistema

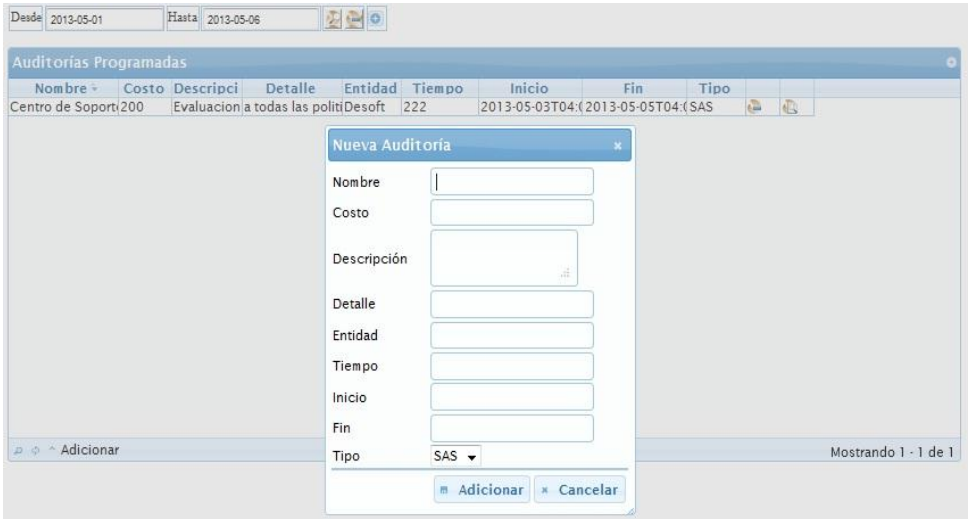


Ilustración 6: Ejemplo del componente vista en el sistema, desde la web

Controlador:

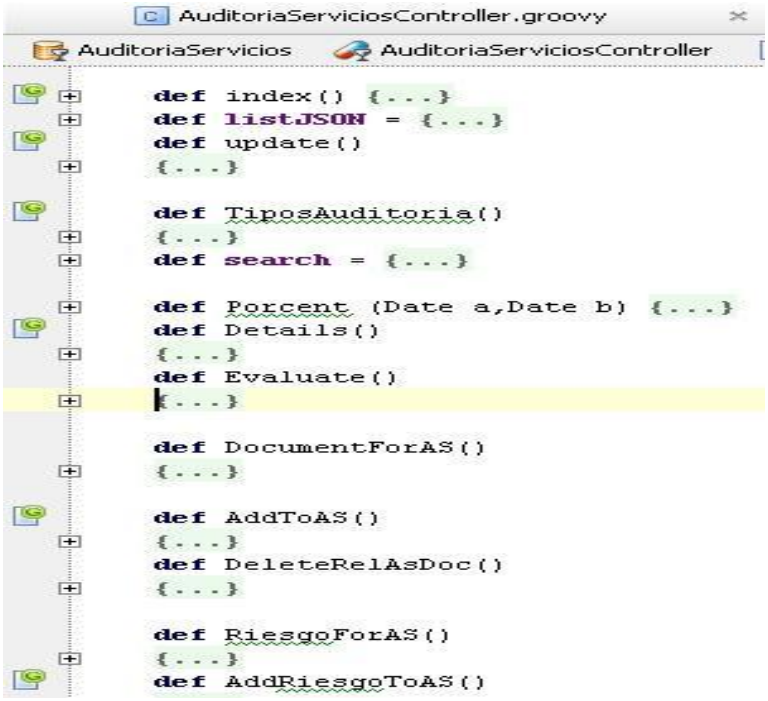


Ilustración 7: Ejemplo del componente controlador en el sistema

2.10. Modelo de base de datos

A continuación se presenta el modelo de Base de Datos, que de una manera lógica y estructurada muestra cómo se almacenan, organizan y manipulan los datos del sistema. En un modelo de base de datos, todos los datos se almacenan y se accede a ellos por medio de relaciones.

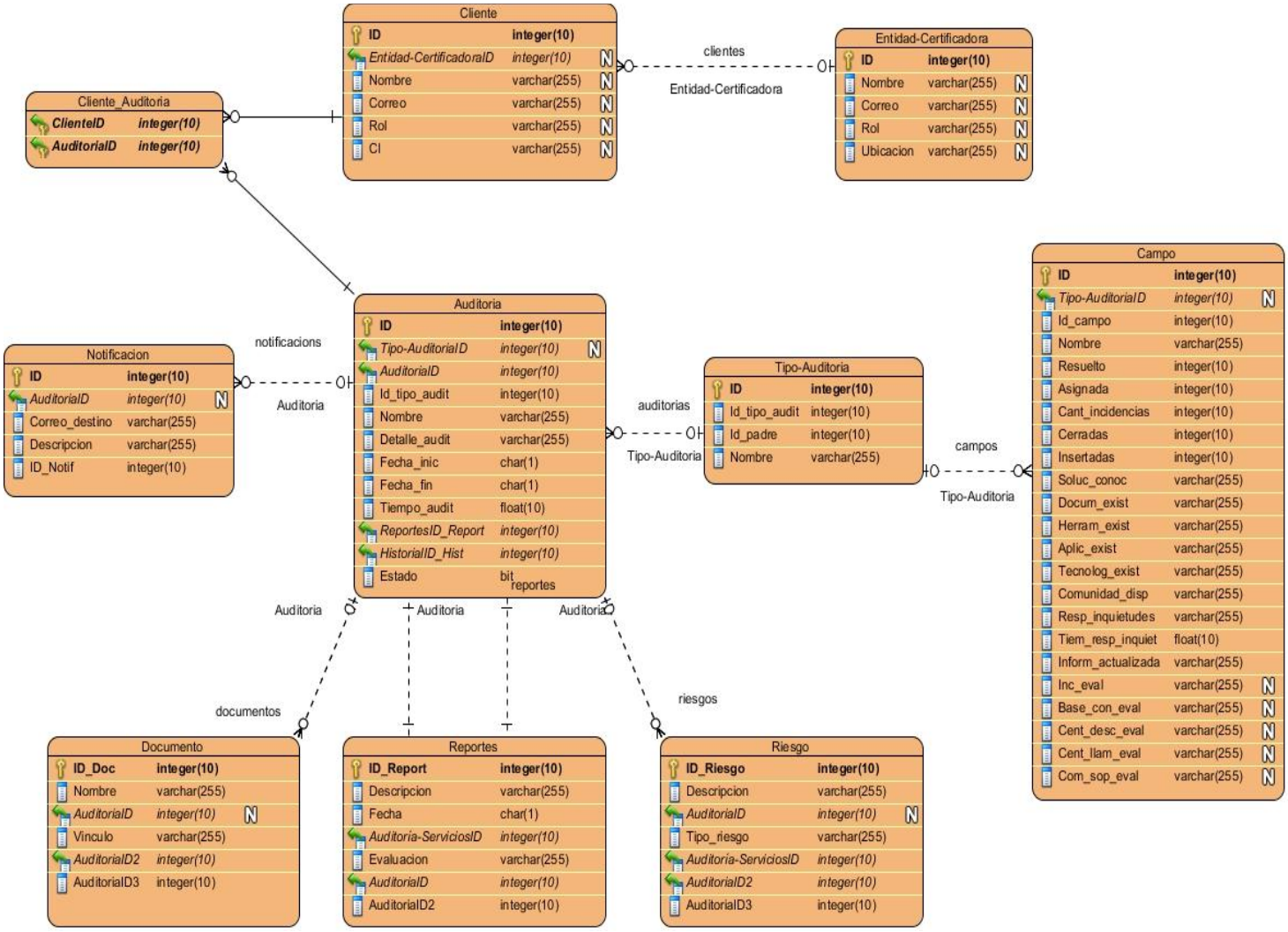


Figure 2-2 Modelo Relacional

2.11. Estándares de codificación

Las convenciones o estándares de codificación son reglas para escribir el código fuente de manera clara y legible, para facilitar la comprensión y apariencia del mismo. Un código fuente completo debe reflejar un estilo armonios, como si lo hubiese escrito un solo programador una única ves. Extreme programing destaca la comunicación de los programadores a través del código, por lo que es muy importante que se sigan uno de los estándares de programación existentes. A continuación se muestran algunos estándares de codificación, así como imágenes delos estándares utilizados en la implementación de este sistema. (37)

2.11.1. Notación Pascal Casing:

Pascal Casing: es un procedimiento de programación común en el lenguaje Java y .Net. La nomenclatura está compuesta por tantas palabras como sean necesarias. La primera letra de cada una de las palabras irá siempre en mayúsculas, obviando el uso de artículos. Es uno de los más utilizados por los programadores, incluso sin saber que se está utilizando un estándar. (37)

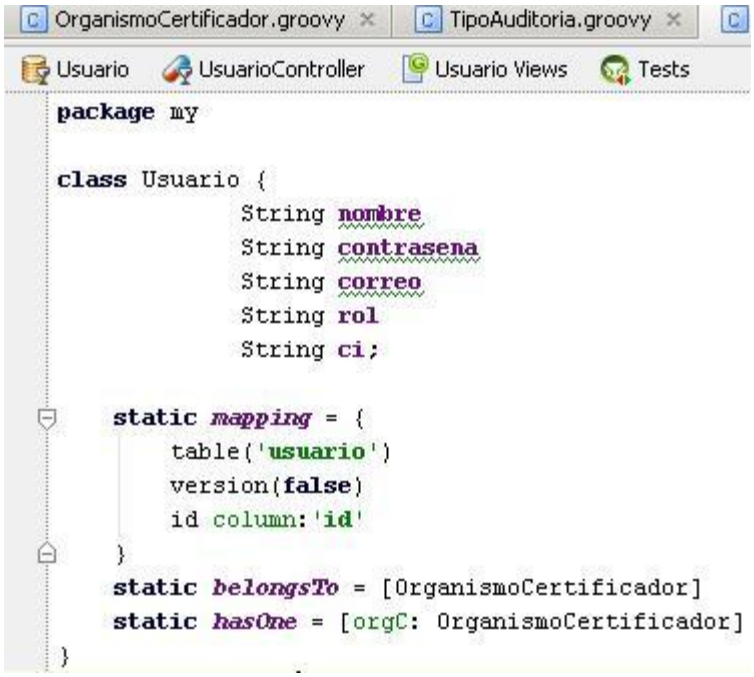


Ilustración 8: Ejemplo de Pascal Casing en el sistema

2.11.2. Notación Camel Casing:

Camel Casing: es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en Camel Case se asemejan a las jorobas de un camello. El nombre Camel Case se podría traducir como Mayúsculas/Minúsculas Camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas. (37)

Existen dos tipos de Camel Case:

- **Upper Camel Case**, cuando la primera letra de cada una de las palabras es mayúscula.

  Ejemplo: EjemploDeUpperCamelCase.

- **Lower Camel Case**, igual que la anterior con la excepción de que la primera letra es minúscula.

  Ejemplo: ejemploDeLowerCamelCase.

2.12. Conclusiones del capítulo

- La metodología XP permitió realizar un plan de iteraciones y un plan de entrega de versiones, que le puso metas tanto al diseñador como al implementador, lo cual garantizó una organización en el desarrollo del sistema.
- Al detallarse cada historia de usuario, las cuales dan respuesta a uno o más requisitos, se logró simplificar el trabajo del implementador, permitiendo que el sistema fuera concluido en menor tiempo de lo previsto en el plan de entrega de versiones.
- Además en el presente capítulo se describieron los patrones de diseño y estándares de codificación que se utilizan durante la implementación del sistema, los cuales permiten una legibilidad del código fuente y una mayor comprensión por parte del usuario.
- Se desarrolló un subsistema de gestión de auditorías para el sistema SAMOS el cual evalúa los servicios que se prestan en el centro de soporte.

CAPÍTULO 3. VALIDACIÓN DEL SISTEMA PROPUESTO

En el presente capítulo se procede a diseñar las pruebas correspondientes a la metodología XP, las cuales se dividen en dos partes, unitarias y de aceptación. Se describe de forma detallada todo el proceso de dichas pruebas y se muestra un resultado final, que explica y muestra al usuario si el sistema esta implementado correctamente y si cuenta con las funcionalidades descritas en el capítulo anterior.

3.1. Pruebas

Las pruebas son una de las prácticas fundamentales en las cuales se basa XP. Esta actividad se realiza en forma continua a lo largo del proyecto. Existen dos tipos de pruebas, las unitarias y las de aceptación *(38)*.

Las pruebas unitarias son definidas por los programadores antes de comenzar a escribir código. Estas deben contemplar cada módulo del sistema que pueda generar fallas. Para poder integrar el código realizado al ya existente, el mismo debe aprobar satisfactoriamente todos los casos de prueba definidos *(38)*.

El cliente con ayuda del probador, define las pruebas de aceptación para cada historia de usuario a principio de cada iteración. Las pruebas de aceptación se utilizan para validar que cada requerimiento implementado funciona como se había especificado *(38)*.

3.1.1. Pruebas unitarias

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. El concepto de “módulo” varía de acuerdo al lenguaje de programación que estemos utilizando; por ejemplo, en Java sería una clase. Las pruebas unitarias nos aseguran que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema (39).

Las pruebas unitarias brindan una inmediata retroalimentación en la realización de su trabajo, permiten al programador saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento actual del mismo. También permiten la aplicación de otras prácticas como refactoring y diseño simple al estar respaldados por efectivos casos de prueba. (39)

La retroalimentación que se genera por la realización de las pruebas, deriva en un constante aprendizaje sobre el sistema a realizar, lo cual permite que los programadores desarrollen de forma más rápida y eficiente.

Las principales ventajas de la utilización de pruebas unitarias en el desarrollo son:

- **Fomentan el cambio**: las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- **Simplifica la integración**: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
- **Documenta el código**: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
- **Los errores están más acotados y son más fáciles de localizar**: dado que tenemos pruebas unitarias que pueden desenmascararlos.

Camino Básico: El método del camino básico permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

Para realizar estas pruebas se utiliza la complejidad ciclomática de McCabe, que consiste en la ejecución de un conjunto de caminos independientes proporcionando una medición cuantitativa de la complejidad lógica de un programa, así como determinar el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

La complejidad ciclomática consta de tres formas para calcularse:

V (G) = R donde R: nº de regiones del grafo

V (G) = A - N + 2 donde A: nº de arcos del grafo, N: nº de nodos

V (G) = P +1 donde P: nº de nodos predicado

A continuación se detalla una de las pruebas unitarias desarrolladas al sistema, en este caso la del camino básico realizadas a diversas funcionalidades correspondientes a diferentes tarjetas CRC.

Método UsuariosForAS()

En la Ilustración 9 se muestran los posibles caminos a tomar al ocurrir la acción del método en cuestión.

```
def UsuariosForAS() {
    def sortIndex = params.sidx ?: 'name'
    def sortOrder = params.sord ?: 'asc'
    def maxRows = Integer.valueOf(params.rows)
    def currentPage = Integer.valueOf(params.page) ?: 1
    def rowOffset = currentPage == 1 ? 0 : (currentPage - 1) * maxRows
    def totalRows = AuditoriaServicios.get(params.id).usuarios.size()
    def contacts = (totalRows >= rowOffset + maxRows) ?
        AuditoriaServicios.get(params.id).usuarios.toList().subList(rowOffset,rowOffset + maxRows) :
        AuditoriaServicios.get(params.id).usuarios.toList().subList(rowOffset,totalRows )

    def numberOfPages = Math.ceil(totalRows / maxRows)

    def results = contacts?.collect { [ cell: [it.id, it.nombre, it.correo, it.rol, it.ci], id: it.id

    def jsonData = [rows: results, page: currentPage, records: totalRows, total: numberOfPages]
    render jsonData as JSON
}
```

Ilustración 9: Método UsuariosForAS()

1. Usando el código como base, se dibuja el correspondiente grafo de flujo, como se muestra en la Ilustración 10.

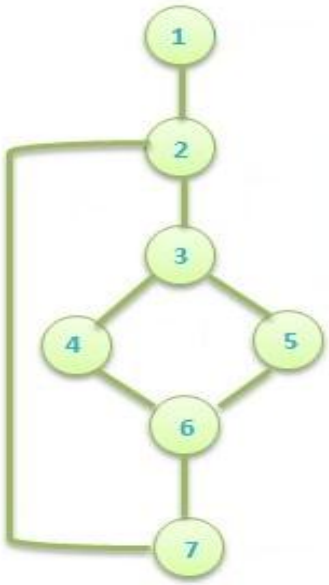


Ilustración 10: Grafo correspondiente al método UsuariosForAS()

2. Se determina la complejidad ciclomática del grafo de flujo resultante, V (G). La Ilustración 11 muestra las regiones identificadas en el grafo correspondiente.

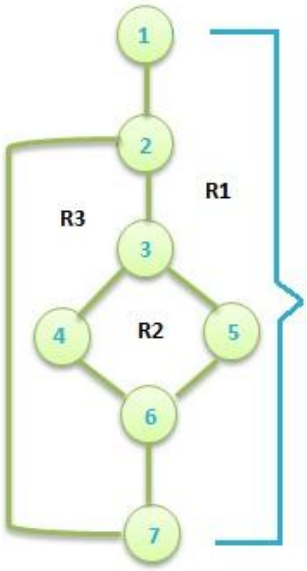


Ilustración 11: Regiones del grafo correspondiente al método UsuariosForAS()

- La complejidad ciclomática del grafo coincide con el número de regiones V (G)=3
- V (G) =A-N+2=. 8-7+2 = 3
- V (G) =P+1=. 2+1= 3

1- Se determina un conjunto básico caminos linealmente independientes.

Luego de determinada la complejidad ciclomática se tiene que 3 es el número máximo de caminos a tener en cuenta para realizar las pruebas.

Caminos independientes determinados:

Camino 1: 1-2-7

Camino 2: 1-2-3-6-7

Camino 3: 1-2-4-6-7

Método Delete()

En la Ilustración 12 se muestran los posibles caminos a tomar al ocurrir la acción del método en cuestión.

```
def Delete() {
    def auth = AuditoriaServicios.get(params.id)
    EvalSAS.findByAudit(auth).delete()
    auth.delete()
    render "La auditoria ha sido eliminada"
}
```

Ilustración 12: Método Delete()

3. Usando el código como base, se dibuja el correspondiente grafo de flujo, como se muestra en la Ilustración 13.

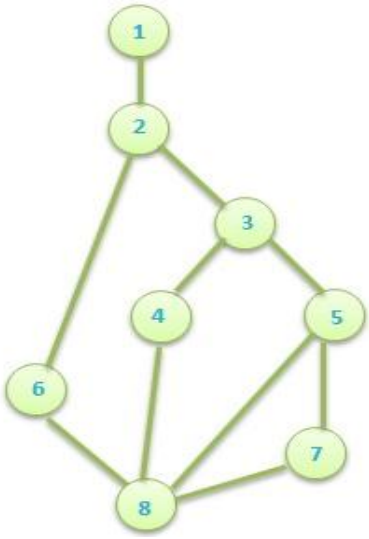


Ilustración 13: Grafo correspondiente al método Delete()

4. Se determina la complejidad ciclomática del grafo de flujo resultante, V (G). La Ilustración 14 muestra las regiones identificadas en el grafo correspondiente.

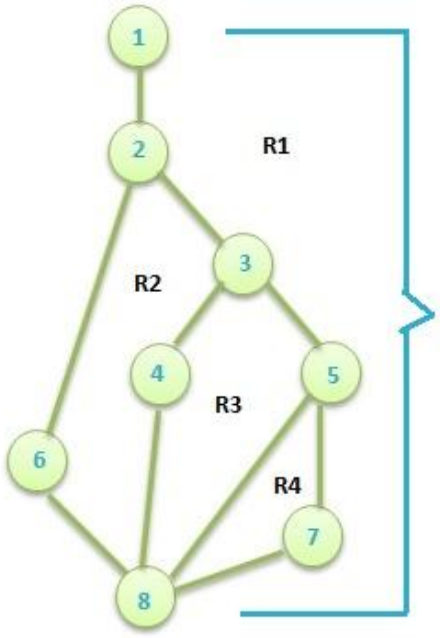


Ilustración 14: Grafo correspondiente al método Delete()

- La complejidad ciclomática del grafo coincide con el número de regiones V (G)=4
- V (G) =A-N+2=. 10-8+2 = 4
- V (G) =P+1=. 3+1= 4

2- Se determina un conjunto básico caminos linealmente independientes.

Luego de determinada la complejidad ciclomática se tiene que 4 es el número máximo de caminos a tener en cuenta para realizar las pruebas.

Caminos independientes determinados:

Camino 1: 1-2-6-8

Camino 2: 1-2-3-4-8

Camino 3: 1-2-3-5-8

Camino 4: 1-2-3-5-7-8

Método listJSON()

En la Ilustración 15 se muestran los posibles caminos a tomar al ocurrir la acción del método en cuestión.

```
def listJSON = {
    def sortIndex = params.sidx ?: 'name'
    def sortOrder = params.sord ?: 'asc'
    def maxRows = Integer.valueOf(params.rows)
    def currentPage = Integer.valueOf(params.page) ?: 1
    def rowOffset = currentPage == 1 ? 0 : (currentPage - 1) * maxRows
    def contacts = AuditoriaServicios.createCriteria().list(max: maxRows, offset: rowOffset) {
          if (params.start_date != '' && params.end_date != '')
          {
              and {
                  def start_date = new Date().parse("yy-MM-dd", params.start_date).clearTime()
                  def end_date = new Date().parse("yy-MM-dd", params.end_date).clearTime()
                  between('fecha_inicio',start_date,end_date)  &&
                          between('fecha_fin',start_date,end_date)
              }
          }
        order(sortIndex, sortOrder)
    }
    def totalRows = AuditoriaServicios.count
    def numberOfPages = Math.ceil(totalRows / maxRows)
    def results = contacts?.collect { [ cell: [it.id, it.nombre, it.costo, it.descripcion,
            it.detalle_auditoria, it.entidad_a_auditar,it.tiempo_auditar,
            it.fecha_inicio, it.fecha_fin,it.tipo_auditoria.nombre], id: it.id ] }

    def jsonData = [rows: results, page: currentPage, records: totalRows, total: numberOfPages]
    render jsonData as JSON
}
```

Ilustración 15: Método listJSON()

Usando el código como base, se dibuja el correspondiente grafo de flujo, como se muestra en la Ilustración 16.

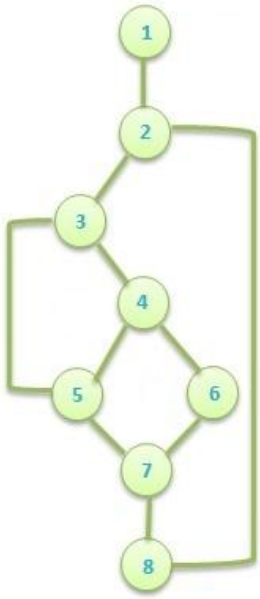


Ilustración 16: Grafo correspondiente al método listJSON()

5. Se determina la complejidad ciclomática del grafo de flujo resultante, V (G). La Ilustración 17 muestra las regiones identificadas en el grafo correspondiente.

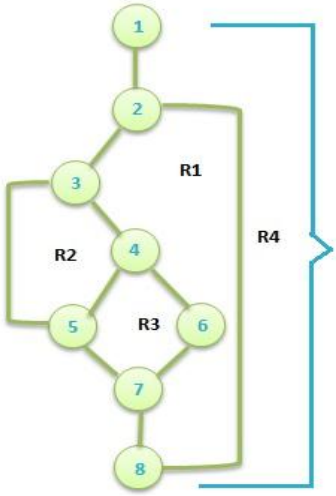


Ilustración 17: Regiones del grafo correspondiente al método listJSON()

La complejidad ciclomática del grafo coincide con el número de regiones V (G)=4

- V (G) =A-N+2=. 10-8+2 = 4
- V (G) =P+1=. 3+1= 4

3- Se determina un conjunto básico caminos linealmente independientes.

Luego de determinada la complejidad ciclomática se tiene que 4 es el número máximo de caminos a tener en cuenta para realizar las pruebas.

Caminos independientes determinados:

Camino 1: 1-2-8

Camino 2: 1-2-3-5-7-8

Camino 3: 1-2-3-4-5-7-8

Camino 4: 1-2-3-6-7-8

3.1.2. Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención (39).

Las pruebas de aceptación tienen una importancia crítica para el éxito de una iteración. Por lo tanto el tester debe tenerlas prontas lo antes posible a partir del comienzo de la iteración, y lograr que el cliente las apruebe para poder presentárselas cuanto antes al equipo de desarrollo.

Como resultado de las pruebas de aceptación se obtendrán artefactos descritos en tablas, estas contarán con los siguientes campos:

- **Código:** servirá como identificador de la prueba realizada, a su vez será sugerente al nombre de la prueba a la que hace referencia.
- **HU:** tendrá el nombre de la HU a la que hace referencia la prueba a realizar.
- **Nombre:** nombre que se le da a la prueba a realizar.
- **Descripción:** se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución:** mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- **Entradas/Pasos de Ejecución:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, 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:** se hará una breve descripción del resultado que se espera obtener con la prueba realizada.
- **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los tres resultados que a continuación se describen:

1) **Bien:** cuando el resultado de la prueba es exactamente el esperado por el usuario.
2) **Parcialmente bien:** cuando el resultado no es completamente el esperado por el cliente o usuario de la aplicación y muestra resultados erróneos o fuera de contexto.
3) **Mal:** cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida también la HU.

A continuación se muestran algunas de las pruebas de aceptación más relevantes y las restantes se pueden visualizar en el Anexo 3.

Tabla 12: Prueba de aceptación # 1

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Gestionar la planificación de la auditoría.	

Descripción: Prueba para la funcionalidad de gestionar la planificación de la auditoría, ya sea crearlas, modificar sus parámetros y eliminarlas.
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorias existentes.
Entradas/Pasos de Ejecución: Del listado de las auditorías existentes, se puede seleccionar la opción adicionar, luego se presiona el botón guardar, se validan las entradas y se retorna a la interfaz inicial. Se puede además seleccionar una auditoría para su posterior modificación, así como eliminarla. El sistema tras toda acción del usuario valida que las entradas sean correctas.
Resultado Esperado: Se gestiona correctamente la auditoría.
Evaluación de la Prueba: Bien.

Tabla 13: Prueba de aceptación # 2

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de Usuario:** 2
Nombre: Gestionar equipo auditor.	
Descripción: Prueba para la funcionalidad de gestionar el equipo que pertenecerá a una auditoría, ya sea crearlos, modificar sus parámetros y eliminarlos.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de los clientes pertenecientes al equipo auditor de la auditoría.	
Entradas/Pasos de Ejecución: Del listado de las auditorías existentes, se selecciona una auditoría, luego el sistema muestra el listado de los clientes pertenecientes al equipo auditor de dicha auditoría. Del listado de los clientes existentes, se puede seleccionar la opción adicionar, luego se presiona el botón guardar, se validan las entradas y se retorna a la interfaz inicial. Se puede además seleccionar un cliente para su posterior modificación, así como eliminarlo. El sistema tras toda acción del usuario valida que las entradas sean correctas.	
Resultado Esperado: Se gestiona correctamente el equipo auditor.	
Evaluación de la Prueba: Bien.	

Tabla 14: Prueba de aceptación # 3

Caso de Prueba de Aceptación	
Código: HU3_P3	**Historia de Usuario:** 3
Nombre: Gestionar documentos asociados.	
Descripción: Prueba para la funcionalidad de gestionar un documento que pertenecerá a una auditoría, ya sea crearlos, modificar sus parámetros y eliminarlos.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de los documentos pertenecientes a la auditoría.	
Entradas/Pasos de Ejecución: Del listado de las auditorías existentes, se selecciona una auditoría, luego el sistema muestra el listado de los documentos pertenecientes dicha auditoría. Del listado de los documentos existentes, se puede seleccionar la opción adicionar, luego se presiona el botón guardar, se validan las entradas y se retorna a la interfaz inicial. Se puede además seleccionar un documento para su posterior modificación, así como eliminarlo. El sistema tras toda acción del usuario valida que las entradas sean correctas.	
Resultado Esperado: Se gestionan correctamente los documentos.	
Evaluación de la Prueba: Bien.	

3.1.3. Análisis de resultados

Se realizaron 13 pruebas funcionales para validar que el sistema funcionara correctamente, mostrando las salidas correspondientes a cada escenario. De estas pruebas realizadas en una primera iteración fueron clasificadas de bien 9 de ellas, representando un 69.23% del total y 4mal, para un 30.76% del total. En una segunda iteración de un total de13pruebas realizadas fue clasificadas de bien12de ellas, representando un 92.30%del total y 1mal, representando un 7.69% del total. En una tercera y última iteración de un total de 13 pruebas realizadas, resultaron todas satisfactorias, constituyendo un 100% de pruebas funcionales exitosas.

3.2. Conclusiones parciales

- Las pruebas unitarias realizadas a las funcionalidades claves de la aplicación, arrojaron como resultado que el sistema se encuentra listo, ya que cumple con los requisitos propuestos por el cliente.

- Se realizaron pruebas de aceptación que permitieron concluir que cuenta con la calidad requerida, siendo el cliente el que más participación tuvo en la evaluación realizada, con el objetivo de garantizar la completa satisfacción de sus exigencias.
- Se obtuvo un documento firmado por el jefe de proyecto del Centro de Soporte UCI que avala que la aplicación obtenida cumple con los requisitos establecidos y contiene la calidad requerida, el cual es mostrado en el Anexo 5.

CONCLUSIONES

A partir de la presente investigación científica, se desarrolló un sistema de gestión de auditorías del Sistema Automatizado de Monitoreo y Control de Servicios (SAMOS) para incrementar la prestación de servicios de Tecnologías de la Información en el Centro de Soporte UCI. La solución implementada, contribuirá en la toma de decisiones del jefe de centro con el personal implicado en la prestación de los servicios de TI. Lo antes escrito se refleja de manera positiva en una mejor satisfacción de los clientes y por consiguiente en una mayor aceptación de los servicios brindados por la entidad.

Fueron descritas las herramientas y tecnologías utilizadas en la solución desarrollada (metodología, patrones, softwares, lenguajes de programación, etc.) que permitieron una implementación eficiente en corto tiempo. Finalmente se documentó de forma detallada todo el proceso de pruebas, mostrando un resultado general de todas las validaciones realizadas al sistema desarrollado.

Las pruebas unitarias, realizadas a las funcionalidades claves de la aplicación, arrojaron como resultado que el sistema se encuentra listo, pues cumple con los requisitos funcionales propuestos por el cliente; las pruebas de aceptación permitieron concluir que cuenta con la calidad requerida, siendo el cliente el que más participación tuvo en la evaluación realizada, con el objetivo de garantizarla completa satisfacción de sus exigencias.

RECOMENDACIONES

Al concluir esta investigación, se recomienda para el desarrollo de estudios futuros:

- Implementar en el sistema la evaluación para otros tipos de auditoría como son:
  - Centro de datos
  - Desarrollo de sistemas
  - Gestión de TI y Arquitectura Empresarial
  - Cliente, Servidor, Telecomunicaciones, Intranet y Extranet
- Implementar la compatibilidad del generador de reportes con formatos exel, doc y xml.
- Revisar el estándar COBIT e integrarlo en su totalidad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Carnota Lauzán, Orlando.** Gerencia y Negocios en Hispano América. [Online] [Cited: Noviembre 20, 2012.] http://www.degerencia.com/tema/tecnologia_de_informacion.

2. **Bologna, Whas.** *Tecnologías de la Información.* 1997.

3. **Alter.** *¿Qué es la Tecnología de la Información?* 1999.

4. **Coello, Helkyn.** [Online] [Cited: Noviembre 20, 2012.] http://helkyncoello.wordpress.com/2008/12/08/itil-COBIT -cmmi-pmbok-como-integrar-y-adoptar-los-estandares-para-un-buen-gobierno-de-ti/.

5. **Institute, IT Governance.** *Resumen Ejecutivo.* 2011.

6. **Company, Bit.** COBIT: Un marco de referencia para la información y la tecnología. [Online] Bit Company, abril 9, 2012. [Cited: enero 6, 2013.] http://www.bitcompany.biz/que-es- COBIT /.

7. *Auditoría Informática - Un Enfoque Práctico.*

8. **Holmes, Arthur.** *Auditoría: Principios y procedimientos.* México : Editorial Hispano América, 1952.

9. American Business Academy . [Online] 2012. [Cited: marzo 5, 2013.] http://www.aba.ac.cr/tecnicoauditoria.htm.

10. **Mendoza Arce, Nelson.** *Auditorías.* 2010.

11. **Thompson, Ivan.** Definición de Servicios. [Online] agosto 2006. [Cited: enero 10, 2013.] http://www.promonegocios.net/mercadotecnia-servicios/definicion-servicios.html.

12. Isaca. [Online] http://www.isaca.org/Knowledge-Center/cobit/Pages/COBIT-Online.aspx?utm_source=multiple&utm_medium=multiple&utm_content=friendly&utm_campaign=cobitonline .

13. **Globales.** *CMMI - Capability Maturity Model Integration.*

14. **Soto, Lorena.** slideshare. [Online] octubre 18, 2012. [Cited: enero 9, 2013.] http://www.slideshare.net/rochocho/tema-viii-14785122.

15. ISO. [Online] 2011. [Cited: febrero 2, 2013.] http://www.iso.org/iso/catalogue_detail?csnumber=51986.

16. ITIL. [Online] 2007. [Cited: febrero 4, 2013.] http://www.itil-officialsite.com/.

17. Mkinsight. [Online] [Cited: enero 25, 2013.] http://www.mkinsight.com/.

18. Softexpert. [Online] [Cited: enero 20, 2013.] http://www.softexpert.es/.

19. Optisoft Latinoamerica. [Online] [Cited: enero 20, 2013.] http://www.optisoftla.com/index_cobit.html.

20. AUDITWorks. [Online] [Cited: febrero 1, 2013.] http://www.primatech.com/software/auditworks?gclid=CIjYvsOe_bQCFQ45nAodnGYAHw.

21. **Amaro Calderón, Sarah Dámaris.** *Metodologías Ágiles.* Trujillo – Perú : s.n., 2007.

22. **D'Arcy, Hamlet.** JetBrains.ORG/IntelliJ. [Online] [Cited: enero 30, 2013.] http://www.jetbrains.org/display/IJOS/Home;jsessionid=689081AE72E3750157BF04901E9916A1.

23. Grails. [Online] 2012. [Cited: enero 15, 2013.] http://grails.bandcamp.com/.

24. Grovy A dynamic language. [Online] [Cited: enero 30, 2013.] http://groovy.codehaus.org/.

25. Visual Paradigm. [Online] [Cited: enero 26, 2013.] http://www.visual-paradigm.com/.

26. **Corporation, Oracle.** Open JDK. [Online] Diciembre 4, 2012. [Cited: enero 26, 2013.] http://openjdk.java.net/projects/jdk8/.

27. Microsoft ASP.NET. [Online] [Cited: enero 30, 2013.] http://www.asp.net/mvc.

28. SVN. [Online] [Cited: enero 30, 2013.] http://lihuen.linti.unlp.edu.ar/index.php?title=C%C3%B3mo_usar_SVN.

29. PostgreSQL. [Online] [Cited: enero 30, 2013.] http://www.postgresql.org/.

30. **Abreu Medina, Aldis Joan.** *Generador dinámico de reportes.* La Habana : s.n., 2012.

31. **Campderrich Falgueras, Benet.** *Ingeniería de Software.* Barcelona : UOC, 2013. 84-8318-997-6.

32. **Sommerville, Ian.** *Ingeniería del software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

33. **Garzás, Javier.** JavierGarzas.com. [Online] diciembre 22, 2011. [Cited: febrero 27, 2013.] http://www.javiergarzas.com/2011/12/historia-de-usuario-diferente-de-requisito.html.

34. **Echeverry Tobón, Luis Miguel .** *Caso práctico de la metodología ágil XP al desarrollo de software.* Pereira : s.n., 2007.

35. **Mesa Reyes, Yunior and Vázquez Ortí, Yudisney .** *Espacio de comunicación e intercambio para la comunidad técnica cubana de postgreSQL.* Habana, Cuba : s.n.

36. **Visconti, Marcello.** *Fundamentos de la ingeniería de software.* Federico, Santa María : s.n.

37. **Miranda, Marco.** *Documento de Estándares de Programación.* 2010.

38. **Malfará, Dayvis.** *Testing en eXtreme Programming.* 2006. 3860676-3.

39. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** *Pruebas del sistema en programación.* University of Sevilla : s.n.

40. **Echeverry Tobón, Luis Miguel.** *Caso práctico de la metodología ágil XP al desarrollo de software.* Pereira : s.n., 2007.

BIBLIOGRAFÍA

1. **Abreu Medina Aldis Joan** Generador dinámico de reportes [Report]. - La Habana : [s.n.], 2012.
2. **Alter** ¿Qué es la Tecnología de la Información? [Report]. - 1999.
3. **Amaro Calderón Sarah Dámaris** Metodologías Ágiles [Book]. - Trujillo – Perú : [s.n.], 2007.
4. American Business Academy [Online]. - 2012. - marzo 5, 2013. - http://www.aba.ac.cr/tecnicoauditoria.htm.
5. Auditoría Informática - Un Enfoque Práctico [Book].
6. AUDITWorks [Online]. - febrero 1, 2013. - http://www.primatech.com/software/auditworks?gclid=CIjYvsOe_bQCFQ45nAodnGYAHw.
7. **Araiza Zapata, Alma**. Auditoría de Tecnología de la Información.
8. **Bologna Whas** Tecnologías de la Información [Report]. - 1997.
9. **Campderrich Falgueras Benet** Ingeniería de Software [Book]. - Barcelona : UOC, 2013. - 84-8318-997-6.
10. **Carnota Lauzán Orlando** Gerencia y Negocios en Hispano América [Online]. - Noviembre 20, 2012. - http://www.degerencia.com/tema/tecnologia_de_informacion.
11. **Coello Helkyn** [Online]. - Noviembre 20, 2012. - http://helkyncoello.wordpress.com/2008/12/08/itil-cobit-cmmi-pmbok-como-integrar-y-adoptar-los-estandares-para-un-buen-gobierno-de-ti/.
12. **Company Bit** COBIT: Un marco de referencia para la información y la tecnología [Online]. - Bit Company, abril 9, 2012. - enero 6, 2013. - http://www.bitcompany.biz/que-es-cobit/.
13. **Corporation Oracle** Open JDK [Online]. - Diciembre 4, 2012. - enero 26, 2013. - http://openjdk.java.net/projects/jdk8/.
14. **D'Arcy Hamlet** JetBrains.ORG/IntelliJ [Online]. - enero 30, 2013. - http://www.jetbrains.org/display/IJOS/Home;jsessionid=689081AE72E3750157BF04901E9916A1.
15. **Echeverry Tobón Luis Miguel** Caso práctico de la metodología ágil XP al desarrollo de software [Report]. - Pereira : [s.n.], 2007.

16. **Garzás Javier** JavierGarzas.com [Online]. - diciembre 22, 2011. - febrero 27, 2013. - http://www.javiergarzas.com/2011/12/historia-de-usuario-diferente-de-requisito.html.
17. **Globales** CMMI - Capability Maturity Model Integration [Report].
18. Grails [Online]. - 2012. - enero 15, 2013. - http://grails.bandcamp.com/.
19. Grovy A dynamic language [Online]. - enero 30, 2013. - http://groovy.codehaus.org/.
20. **Holmes Arthur** Auditoría: Principios y procedimientos [Book]. - México : Editorial Hispano América, 1952.
21. **Institute IT Governance** Resumen Ejecutivo [Report]. - 2011.
22. Isaca [Online]. - http://www.isaca.org/Knowledge-Center/cobit/Pages/COBIT-Online.aspx?utm_source=multiple&utm_medium=multiple&utm_content=friendly&utm_campaign=cobitonline.
23. ISO [Online]. - 2011. - febrero 2, 2013. - http://www.iso.org/iso/catalogue_detail?csnumber=51986.
24. ITIL [Online]. - 2007. - febrero 4, 2013. - http://www.itil-officialsite.com/.
25. **J. J. Gutiérrez M. J. Escalona, M. Mejías, J. Torres** Pruebas del sistema en programación [Report]. - University of Sevilla : [s.n.].
26. **López, Patricia.** Herramienta Case Visual Paradigm.
27. **López Gavira, Rosario.** Consecuencias de la prestación de servicios adicionales.
28. **Malfará Dayvis** Testing en eXtreme Programming [Report]. - 2006. - 3860676-3.
29. **Mendoza Arce Nelson** Auditorías [Report]. - 2010.
30. Microsoft ASP.NET [Online]. - enero 30, 2013. - http://www.asp.net/mvc.
31. **Mesa Reyes Yunior, Vázquez Ortíz Yudisney** Espacio de comunicación e intercambio para la comunidad técnica cubana de postgreSQL [Report]. - Habana, Cuba : [s.n.].
32. **Miranda Marco** Documento de Estándares de Programación [Report]. - 2010.
33. Mkinsight [Online]. - enero 25, 2013. - http://www.mkinsight.com/.
34. Optisoft Latinoamerica [Online]. - enero 20, 2013. - http://www.optisoftla.com/index_cobit.html.

35. PostgreSQL [Online]. - enero 30, 2013. - http://www.postgresql.org/.

36. Softexpert [Online]. - enero 20, 2013. - http://www.softexpert.es/.

37. **Sommerville Ian** Ingeniería del software [Book]. - Madrid : Pearson Educación S.A., 2005. - 84-7829-074-5.

38. **Soto Lorena** slideshare [Online]. - Rochin Piolin, octubre 18, 2012. - enero 9, 2013. - http://www.slideshare.net/rochocho/tema-viii-14785122.

39. SVN [Online]. - enero 30, 2013. - http://lihuen.linti.unlp.edu.ar/index.php?title=C%C3%B3mo_usar_SVN.

40. **Thompson Ivan** Definición de Servicios [Online]. - agosto 2006. - enero 10, 2013. - http://www.promonegocios.net/mercadotecnia-servicios/definicion-servicios.html.

41. **Visconti Marcello** Fundamentos de la ingeniería de software [Report]. - Federico, Santa María : [s.n.].

42. Visual Paradigm [Online]. - enero 26, 2013. - http://www.visual-paradigm.com/.

ANEXOS

Anexo 1. Entrevista

(realizada a especialistas Del Centro de Soporte)

Objetivo: Obtener las pautas para evaluar los servicios de Tecnología de la Información que se prestan en el Centro de Soporte UCI.

Compañero o compañera.

Se necesita su valiosa colaboración para la realización de esta investigación, de sus respuestas dependen los resultados de la misma. **MUCHAS GRACIAS**.

Datos generales.

- Nombre y Apellidos:
  - ✓ Suset Fernández Rojas
  - ✓ Yandry Alberto Terry

Aspecto a encuestar

1. ¿Cuáles son los servicios de TI que se brindan en el Centro?

2. De los servicios que se prestan en el Centro ¿Cuáles son auditables?

3. ¿Cuáles son las pautas por las que debe regirse un auditor para evaluar los servicios de TI que se prestan en el Centro?

4. ¿Consideran que una auditoría es necesaria para contribuir en la mejora de la prestación de los servicios? ¿Por qué?

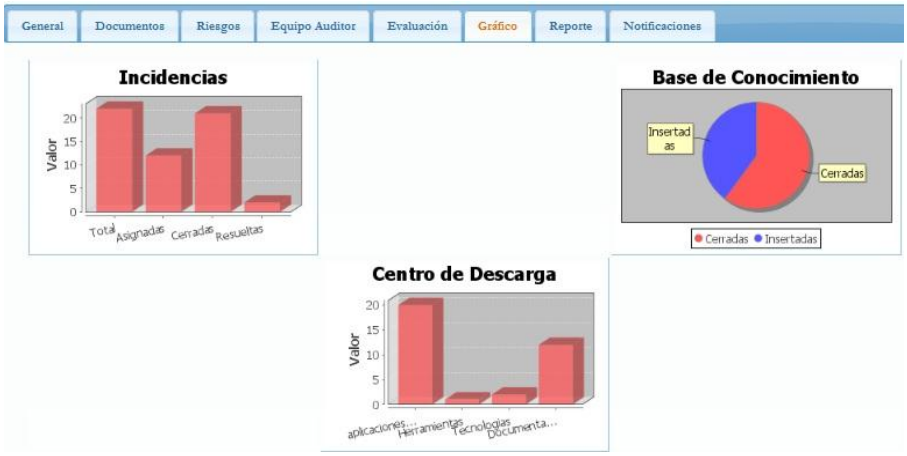
Anexo 2. Historias de Usuarios.

Historia de Usuario # 4.

	Historia de Usuario
Número: 4	**Nombre de la Historia de Usuario:** Gestionar reportes.

Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 3 semanas
Riesgo en desarrollo: Alto	**Puntos reales:** 3 y ½ semanas
Descripción: 1. El sistema muestra la interfaz correspondiente a la evaluación. 2. El auditor llena los datos correspondientes a las pautas de evaluación definidas y pulsa el botón "Evaluar". 3. El sistema muestra dos imágenes correspondientes a los reportes gráficos y PDF. 4. El auditor pulsa de manera independiente cada uno de los botones. 5. El sistema muestra en secciones independientes ambos reportes, que le permiten al auditor dar una evaluación a la institución.	
Observaciones: Para generar un reporte la auditoría debe haber sido previamente evaluada.	

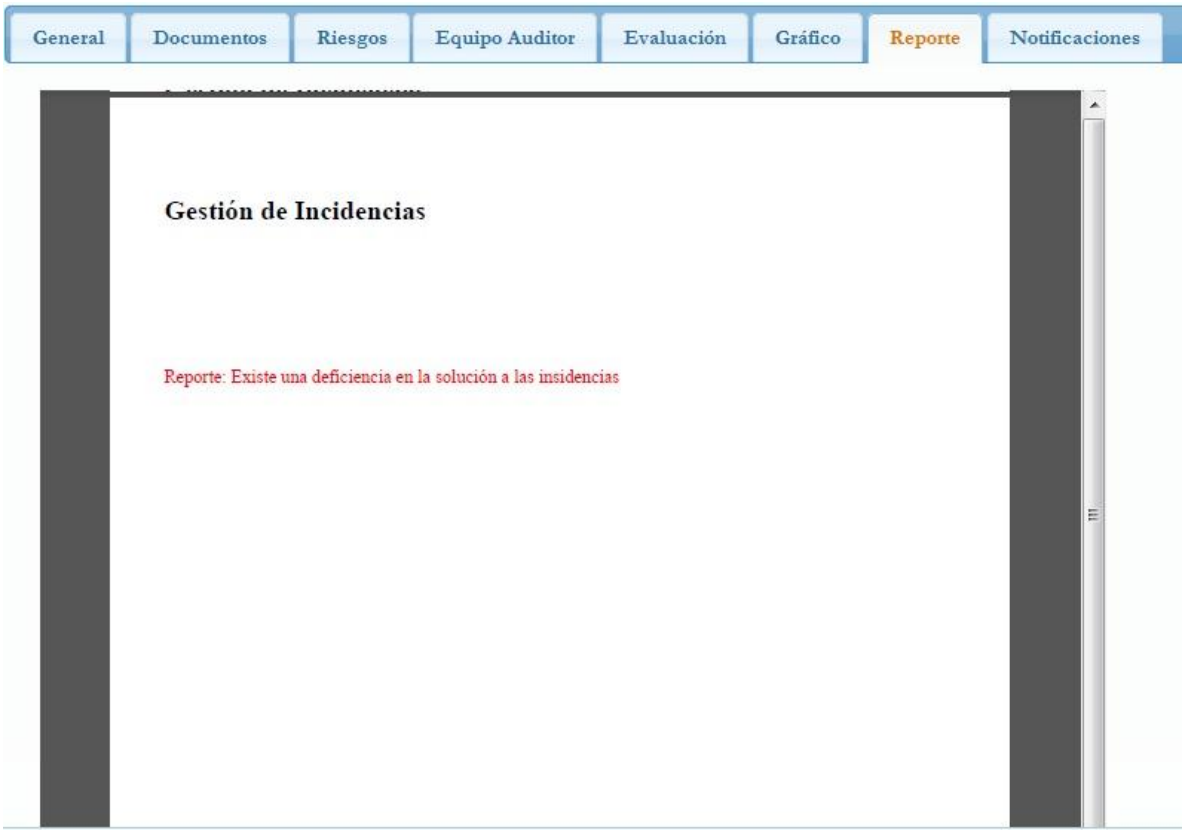
Prototipo de interfaces:



Historia de Usuario # 5.

	Historia de Usuario
Número: 5	**Nombre de la Historia de Usuario:** Proponer buenas prácticas, observaciones y no conformidades.

Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** ½ semanas
Riesgo en desarrollo: Alto	**Puntos reales:** ½ semanas
Descripción: 1. El sistema muestra la interfaz correspondiente a la evaluación. 2. El auditor llena los datos correspondientes a las pautas de evaluación definidas y en el campo llamado descripción tiene la posibilidad de redactar las observaciones, no conformidades así como buenas prácticas para la mejora de la institución. Luego pulsa en el botón "Evaluar" 3. El sistema evalúa, guarda los resultados y muestra dos imágenes correspondientes a los reportes gráficos y PDF. 4. El auditor pulsa el botón correspondiente al reporte PDF. 5. El sistema muestra en una sección el reporte PDF con lo escrito anteriormente por el auditor.	
Observaciones: Para generar un reporte la auditoría debe haber sido previamente evaluada.	
Prototipo de interfaces:	



Historia de Usuario # 6.

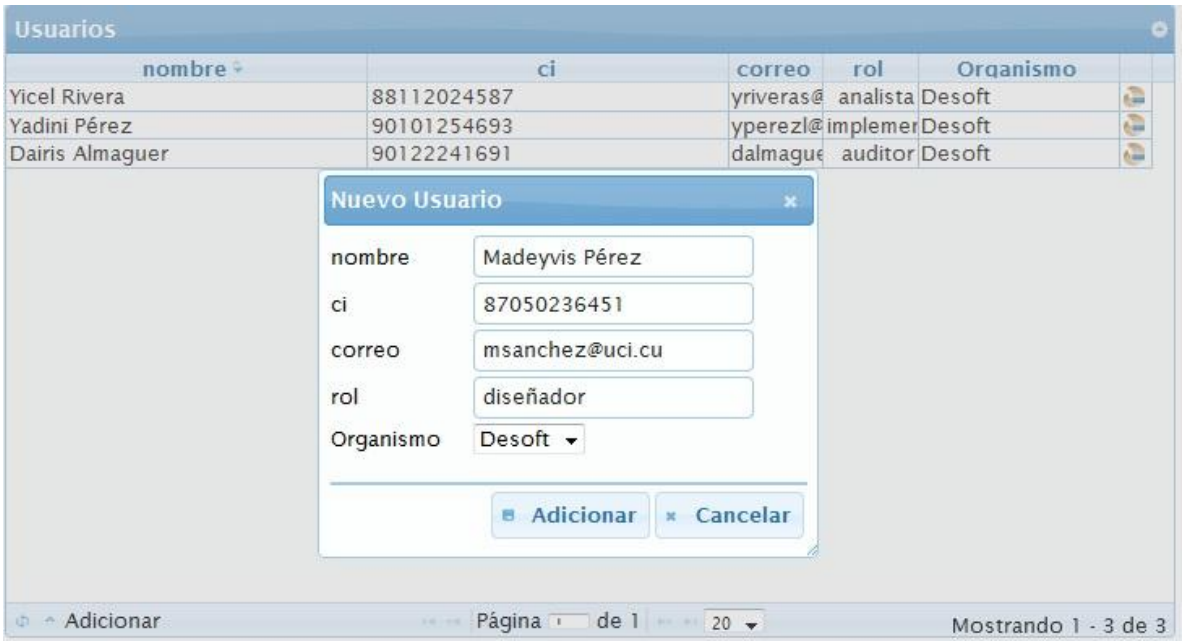
	Historia de Usuario
Número: 6	**Nombre de la Historia de Usuario:** Gestionar clientes
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 2 semanas

Descripción:

1. El sistema muestra la interfaz correspondiente a la gestión de clientes.
2. El usuario pulsa en el botón “Adicionar”.
3. El sistema muestra un conjunto de parámetros correspondientes con los datos de un cliente.
4. El usuario completa los datos y el cliente se adiciona.
5. El sistema muestra un cuadro de diálogo, informándole al usuario que el cliente ha sido adicionado correctamente.

Observaciones: Si los datos del cliente no son correctos, la aplicación mostrará mensajes con los errores correspondientes. Además para insertar un cliente debe asociársele un organismo certificador.

Prototipo de interfaces:



Historia de Usuario # 7.

Historia de Usuario

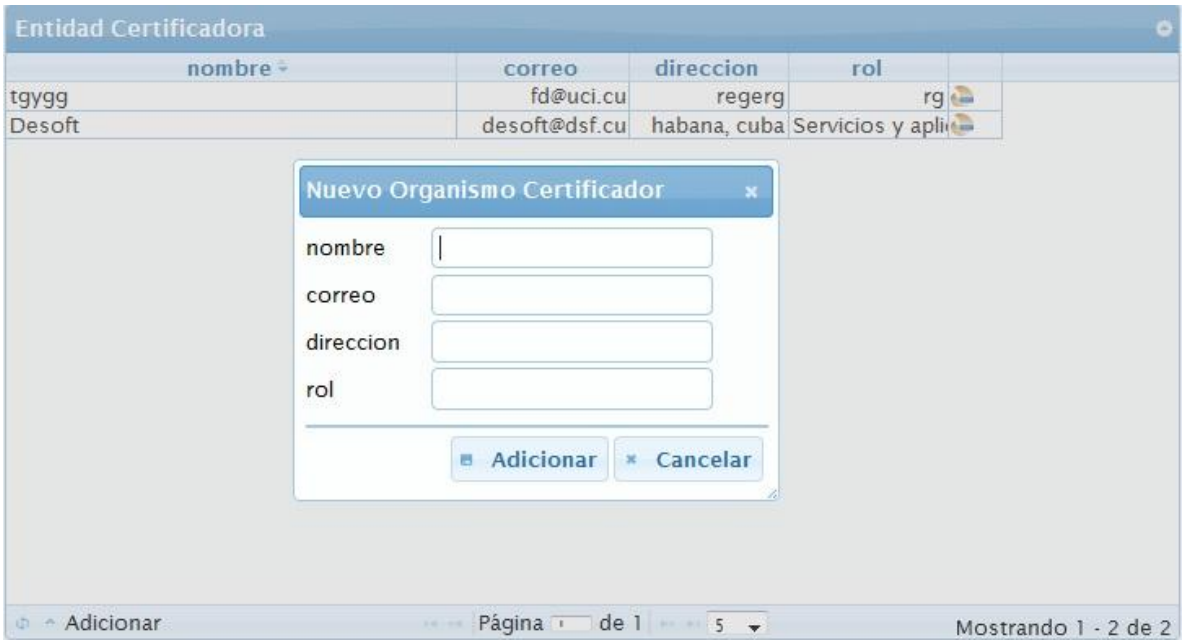
Número: 7	**Nombre de la Historia de Usuario:** Gestionar organismos certificadores
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 2 semanas

Descripción:

1. El sistema muestra la interfaz correspondiente a la gestión de organismos certificadores.
2. El usuario pulsa en el botón “Adicionar”.
3. El sistema muestra un conjunto de parámetros correspondientes con los datos de un organismo certificador.
4. El usuario completa los datos y el organismo certificador se adiciona.
5. El sistema muestra un cuadro de diálogo, informándole al usuario que el organismo certificador ha sido adicionado correctamente. Además el sistema le permite al usuario modificar, eliminar o buscar dicho organismo certificador.

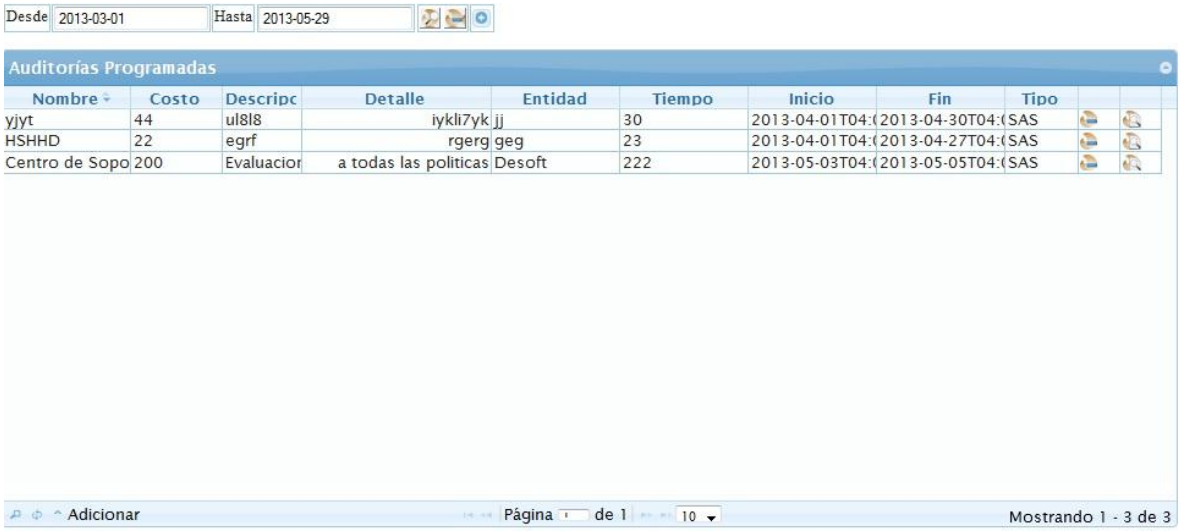
Observaciones: Si los datos del organismo certificador no son correctos, la aplicación mostrará mensajes con los errores correspondientes.

Prototipo de interfaces:



Historia de Usuario # 8.

Historia de Usuario	
Número: 8	**Nombre de la Historia de Usuario:** Mostrar resultados de auditorías anteriores.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 3 y ½ semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 3 y ½ semanas
Descripción: 1. El sistema muestra la interfaz correspondiente a las auditorías y en la parte superior existe un buscador con los campos “Fecha inicio” y “Fecha fin”.	

2. El auditor pulsa ambos campos y selecciona el rango de fecha en el que se encuentra la auditoría que busca. 3. El sistema muestra todas las auditorías que se realizaron en ese rango de fechas. 4. El auditor tiene la posibilidad de ver esa auditoría y todos los resultados de la misma.
Observaciones:
Prototipo de interfaces: 

Historia de Usuario # 9.

	Historia de Usuario
Número: 9	**Nombre de la Historia de Usuario:** Gestionar Riesgos
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 1 y ½ semanas

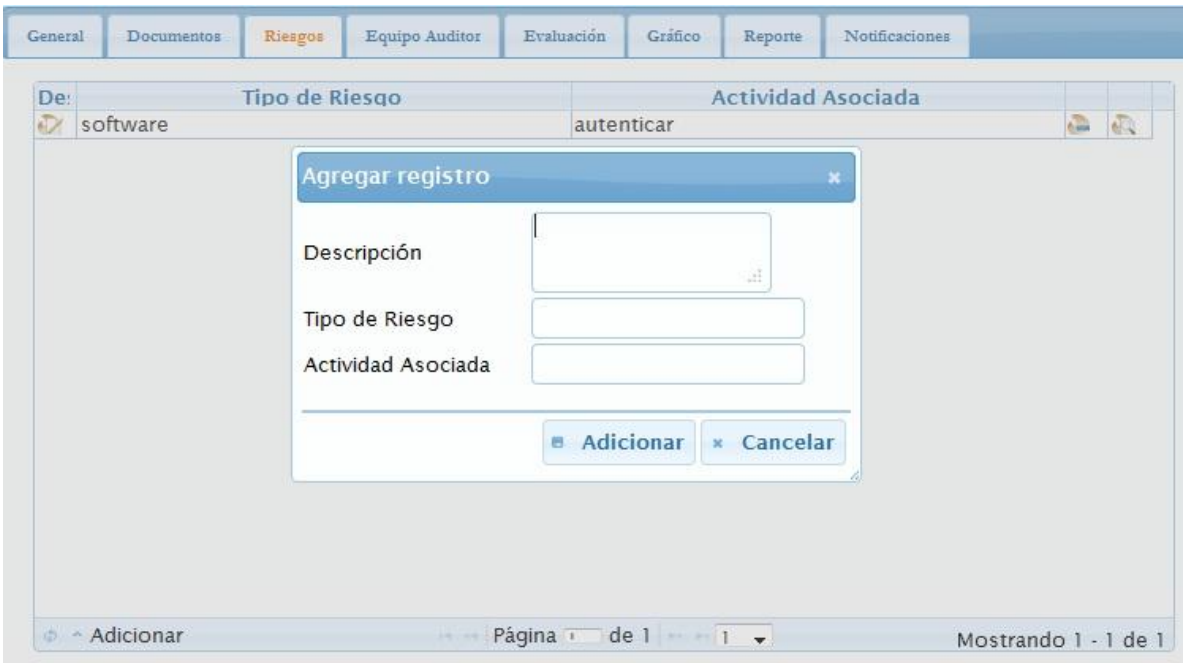
Riesgo en desarrollo: Medio	**Puntos reales:** 1 semana

Descripción:

1. El sistema muestra en su interfaz principal la opción de gestionar riesgos en cualquier paso en que se encuentre la auditoría.
2. El auditor pulsa en el botón “Adicionar”.
3. El sistema muestra los parámetros del riesgo.
4. El auditor introduce los parámetros del riesgo y lo añade.
5. El sistema muestra el riesgo añadido para que el auditor esté seguro de que el riesgo se adicionó correctamente. Además el sistema le muestra la opción al usuario de eliminar dicho riesgo y volver a adicionar tantos riesgos como sea necesario.

Observaciones: Si los datos de los riesgos no son correctos, la aplicación mostrará mensajes con los errores correspondientes.

Prototipo de interfaces:



Historia de Usuario # 10.

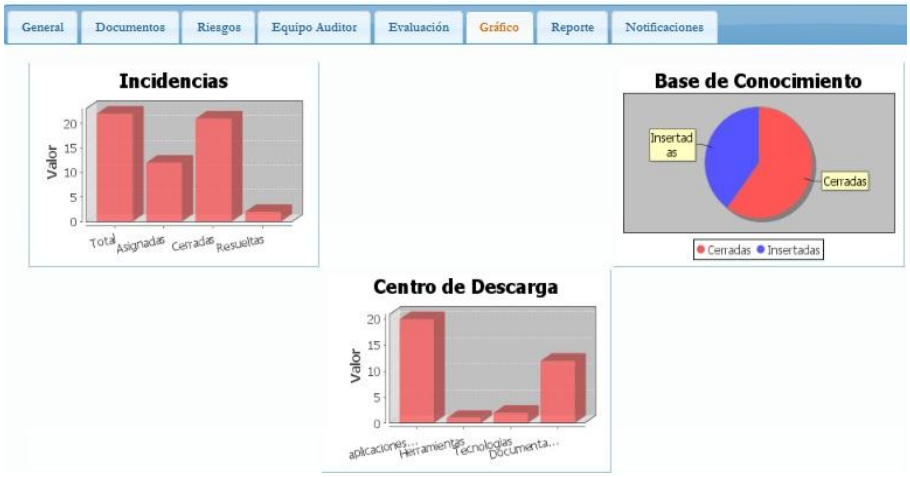
Historia de Usuario	
Número: 10	**Nombre de la Historia de Usuario:** Mostrar registro de evidencias y certificados en la realización de auditorías.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 3 y ½ semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 3 y ½ semanas

Descripción:

1. El sistema muestra una interfaz con las auditorías existentes permitiendo al auditor seleccionar una para su evaluación.
2. El auditor selecciona la auditoría que va a evaluar.
3. El sistema muestra toda la información correspondiente a la auditoría seleccionada y los campos inherentes a la evaluación.
4. El usuario llena dichos campos y pulsa el botón “Evaluar”.
5. El sistema muestra dos imágenes correspondientes a los reportes tanto gráficos como en formato PDF obtiene como resultado un reporte gráfico y en formato PDF.
6. El auditor pulsa de manera independiente cada botón.
7. El sistema muestra en secciones diferentes los reportes gráficos y PDF con lo escrito anteriormente por el auditor.

Observaciones: Para mostrar el registro de evidencias y certificados de la auditoría, ésta debió ser evaluada previamente.

Prototipo de interfaces:



Historia de Usuario # 11.

Historia de Usuario

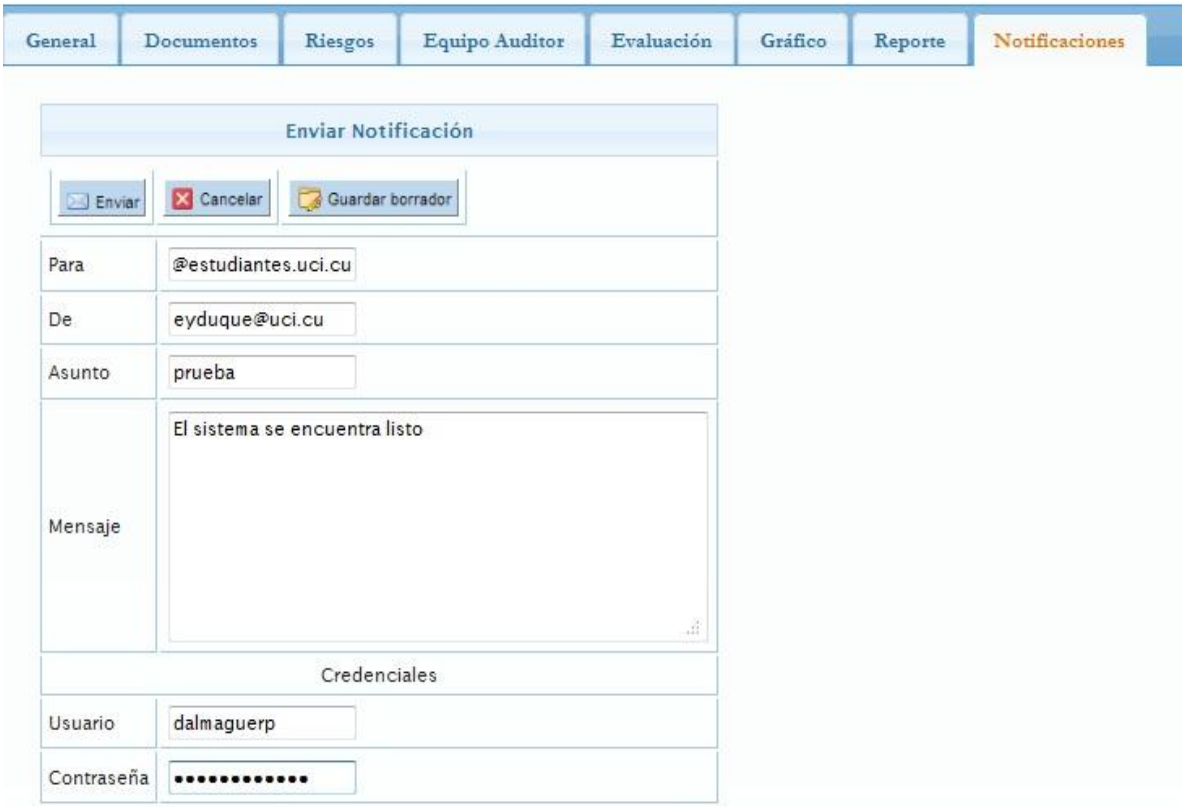
Número: 11	**Nombre de la Historia de Usuario:** Gestionar notificaciones
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 2 y ½ semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 2 y ½ semanas

Descripción:

1. El sistema muestra en su interfaz la opción de gestionar notificaciones en cualquier paso en que se encuentre la auditoría.
2. El auditor pincha en el botón “Enviar notificación”.
3. El sistema muestra los parámetros correspondientes a la notificación.
4. El auditor introduce los parámetros de la notificación y presiona el botón “Enviar”.
5. El sistema muestra un cuadro de dialogo informándole al usuario que la notificación ha sido enviada satisfactoriamente. Además el sistema le permite al auditor guardar como borrador dicho mensaje para su posterior envío o eliminarlo.
6. El auditor podrá enviar tantas notificaciones como sean necesarias.

Observaciones: El sistema valida las credenciales del usuario que enviará la notificación, verificando su identidad y privilegios necesarios.

Prototipo de interfaces:

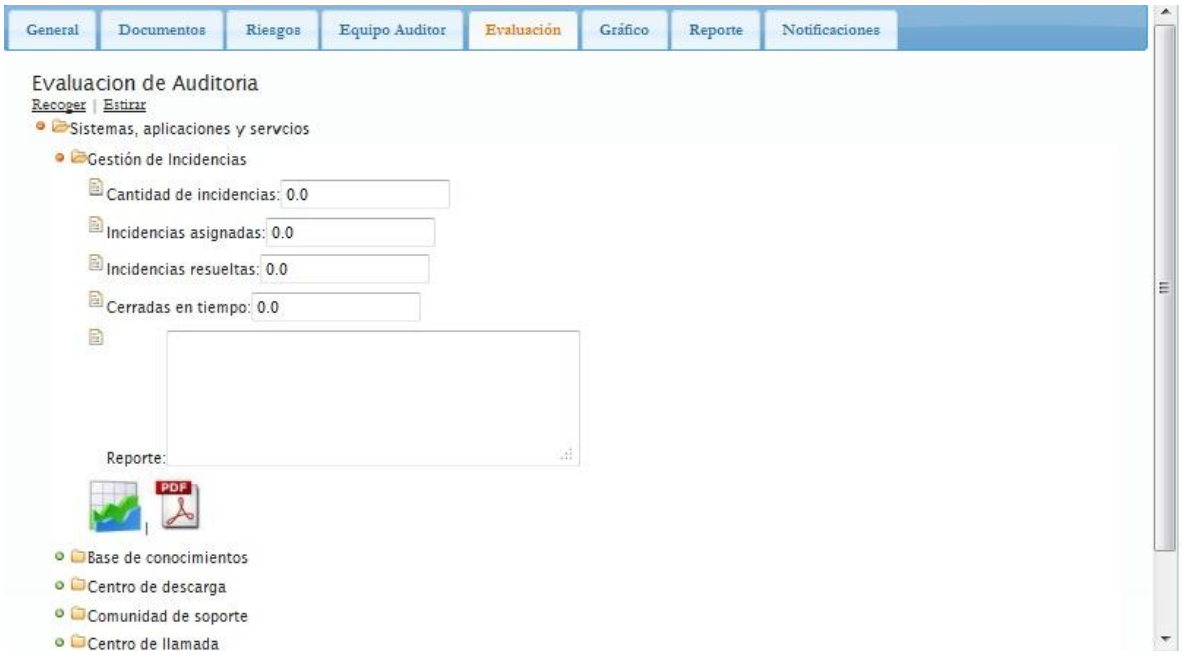


Historia de Usuario # 12.

<table>
<tr><td colspan="2">Historia de Usuario</td></tr>
<tr><td>Número: 12</td><td>Nombre de la Historia de Usuario: Mostrar el tiempo y los pasos en que se encuentra la auditoría.</td></tr>
<tr><td colspan="2">Cantidad de modificaciones a la Historia de Usuario: Ninguna</td></tr>
<tr><td>Usuario: Auditor</td><td>Iteración asignada: 3</td></tr>
<tr><td>Prioridad en negocio: Media</td><td>Puntos estimados: 2 semanas</td></tr>
<tr><td>Riesgo en desarrollo: Medio</td><td>Puntos reales: 2 semanas</td></tr>
<tr><td colspan="2">Descripción:
1. El sistema muestra en su interfaz todas las auditorías.
2. El auditor selecciona la auditoría para verificar el progreso de la misma.
3. El sistema muestra una barra de progreso en cuanto al tiempo de duración de la misma.</td></tr>
<tr><td colspan="2">Observaciones:</td></tr>
<tr><td colspan="2">Prototipo de interfaces:
 
</td></tr>
</table>

Historia de Usuario # 13.

	Historia de Usuario
Número: 13	**Nombre de la Historia de Usuario:** Mostrar registros de criterios de auditorías.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Auditor	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 2 y ½ semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 2 y ½ semanas
Descripción: 1. El sistema muestra en su interfaz todas las auditorías. 2. El usuario selecciona la auditoría 3. El sistema muestra a continuación los criterios de evaluación de la auditoría mediante una estructura arbórea.	
Observaciones: Si la auditoría ha sido evaluada los campo aparecerán completados, en caso contrario aparecerán vacíos.	
Prototipo de interfaces:	



Anexo 3. Tarjetas CRC.

Tarjeta CRC # 4.

Tarjeta CRC	
Clase: OrganismoCertificadorController	
Responsabilidades	**Colaboraciones**

• Es la encargada de:
  ↳ listJSON(string sidx, string sord, int rows, int page)
• Además de las funcionalidades que genera de forma automática el framework:
  ↳ list(Integermax)
  ↳ create()
	↳ Usuarios

save() show(Long id) edit(Long id)) update(Long id, Long version) delete(Long id)	

Tarjeta CRC # 5.

Tarjeta CRC	
Clase: Usuario	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere un cliente de elemento de la configuración: String nombre String correo String rol String ci	

Tarjeta CRC # 6.

Tarjeta CRC	
Clase: UsuarioController	
Responsabilidades	**Colaboraciones**
• Es la encargada de: listJSON(string sidx, string sord, int rows, int page)	OrganismoCertificador

▪ EntidadesCertificadoras() • Además de las funcionalidades que genera de forma automática el framework: ▪ list(Integermax) ▪ create() ▪ save() ▪ show(Long id) ▪ edit(Long id)) ▪ update(Long id, Long version) ▪ delete(Long id)	

Tarjeta CRC # 7.

Tarjeta CRC	
Clase: Documento	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere un documento de elemento de la configuración: ▪ String nombre ▪ String vinculo	

Tarjeta CRC # 8.

Tarjeta CRC	
Clase: DocumentoController	
Responsabilidades	**Colaboraciones**

• Es la encargada de:
  listJSON(string sidx, string sord, int rows, int page)
• Además de las funcionalidades que genera de forma automática el framework:
  list(Integermax)
  create()
  save()
  show(Long id)
  edit(Long id))
  update(Long id, Long version)
  delete(Long id)
	AuditoriaServicios

Tarjeta CRC # 9.

Tarjeta CRC	
Clase: Riesgo	
Responsabilidades	**Colaboraciones**

• Es la encargada de almacenar los atributos que requiere un riesgo de elemento de la configuración:
  Stringdescripcion
  Stringtipo_riesgo
  Stringactividad_asociada	

Tarjeta CRC # 10.

Tarjeta CRC

Clase: TipoAuditoria	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere un tipo de auditoria de elementos de la configuración: - String nombre - Stringdescripcion	

Tarjeta CRC # 11.

Tarjeta CRC	
Clase: EvalSAS	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere una evaluación de elemento de la configuración: - doublecount_inc - double asignadas - double resueltas - double cerradas - Stringnombre_p - doublecerradas_bc - doubleinsertadas_bc - doubleaplicaciones_informaticas - doubledocumentacion - double herramientas - doubletecnologías	

- double inquietudes - double respuestas - doubletiempo_cs - doublellamadas_p_dias - doublellamadas_p_semanas - AuditoriaServiciosaudit	

Tarjeta CRC # 12.

Tarjeta CRC	
Clase: Notificaciones	
Responsabilidades	**Colaboraciones**
• Es la encargada de almacenar los atributos que requiere una notificación de elemento de la configuración: - String remitente - String destinatario - String asunto - String mensaje	

Anexo 4. Pruebas de Aceptación.

Prueba de aceptación # 4.

Caso de Prueba de Aceptación	
Código: HU4,5,10_P4	**Historia de Usuario:** 4,5,10
Nombre: Mostrar registro de evidencias y certificados en la realización de auditorías.	

Descripción: Prueba para la funcionalidad gestionar las evidencias y certificados en la realización de auditorías.
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorías y seleccionar una de ellas.
Entradas/Pasos de Ejecución: En el listado de las auditorías existentes, se selecciona una de ellas. El sistema muestra los campos necesarios en orden de evaluar dicha auditoría, permitiéndole al usuario llenar dicha información. Luego el sistema valida las entradas del usuario y evalúa la auditoría, mostrando en secciones diferentes los reportes tanto gráficos como PDF.
Resultado Esperado: Se evalúa correctamente la auditoría y se obtienen los reportes asociados a la misma.
Evaluación de la Prueba: Bien.

Prueba de aceptación # 5.

Caso de Prueba de Aceptación	
Código: HU6_P5	**Historia de Usuario:** 6
Nombre: Gestionar clientes	
Descripción: Prueba para la funcionalidad de gestionar clientes, ya sea crearlos, modificar sus parámetros y eliminarlos.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de los clientes.	
Entradas/Pasos de Ejecución: Del listado de los clientes existentes, se puede seleccionar la opción adicionar, luego se presiona el botón guardar, se validan las entradas y se retorna a la interfaz inicial. Se puede además seleccionar un cliente para su posterior modificación, así como eliminarlo. El sistema tras toda acción del usuario valida que las entradas sean correctas	
Resultado Esperado: Se gestiona correctamente el cliente.	
Evaluación de la Prueba: Bien.	

Prueba de aceptación # 6.

Caso de Prueba de Aceptación	
Código: HU7_P6	**Historia de Usuario:** 7
Nombre: Gestionar organismos certificadores	
Descripción: Prueba para la funcionalidad de gestionar organismos certificadores, ya sea crearlos, modificar sus parámetros y eliminarlos.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de los organismos certificadores.	
Entradas/Pasos de Ejecución: Del listado de los organismos certificadores existentes, se puede seleccionar la opción adicionar, luego se presiona el botón guardar, se validan las entradas y se retorna a la interfaz inicial. Se puede además seleccionar un organismo certificador para su posterior modificación, así como eliminarlo. El sistema tras toda acción del usuario valida que las entradas sean correctas	
Resultado Esperado: Se gestiona correctamente el organismo certificador.	
Evaluación de la Prueba: Bien.	

Prueba de aceptación # 7.

Caso de Prueba de Aceptación	
Código: HU8_P7	**Historia de Usuario:** 8
Nombre: Mostrar resultados de auditorías anteriores.	
Descripción: Prueba para la funcionalidad de mostrar las auditorías anteriores.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorías.	
Entradas/Pasos de Ejecución: El listado de las auditorías existentes, se puede filtrar por período de tiempo, luego se presiona el botón buscar se validan las entradas y se retorna a la interfaz inicial con los resultados de las auditorías coincidentes.	
Resultado Esperado: Se muestra correctamente el resultado de la búsqueda.	
Evaluación de la Prueba: Bien.	

Prueba de aceptación # 8.

Caso de Prueba de Aceptación	
Código: HU9_P8	**Historia de Usuario:** 9
Nombre: Gestionar riesgos	
Descripción: Prueba para la funcionalidad gestionar riesgos.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorías y seleccionar una de ellas.	
Entradas/Pasos de Ejecución: En el listado de las auditorías existentes, se selecciona una de ellas. El sistema muestra los riesgos pertenecientes a dicha auditoría, permitiéndole al usuario eliminar, modificar y adicionar riesgos. Luego el sistema valida las entradas del usuario y retorna a la interfaz principal, mostrando todos los riesgos pertenecientes a la auditoría.	
Resultado Esperado: Se gestionan correctamente los riesgos.	
Evaluación de la Prueba: Bien.	

Prueba de aceptación # 9.

Caso de Prueba de Aceptación	
Código: HU11_P7	**Historia de Usuario:** 11
Nombre: Gestionar notificaciones	
Descripción: Prueba para la funcionalidad gestionar notificaciones.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorías y seleccionar una de ellas.	
Entradas/Pasos de Ejecución: En el listado de las auditorías existentes, se selecciona una de ellas. El sistema muestra las notificaciones pertenecientes a dicha auditoría que están guardadas como borrador, permitiéndole al usuario enviarla, modificarla y eliminarla. Luego el sistema valida las entradas del usuario y retorna a la interfaz principal, mostrando todos las notificaciones pertenecientes a la auditoría. Además el usuario podrá crear y enviar tantas notificaciones como sea necesario.	
Resultado Esperado: Se gestionan correctamente las notificaciones.	
Evaluación de la Prueba: Bien.	

Prueba de aceptación # 10.

Caso de Prueba de Aceptación	
Código: HU12_P8	**Historia de Usuario:** 12
Nombre: Mostrar el tiempo y los pasos en que se encuentra la auditoría.	
Descripción: Prueba para la funcionalidad de desarrollo de la auditoría.	
Condiciones de Ejecución: Se debe acceder a la interfaz que muestra el listado de las auditorías y seleccionar una de ellas.	
Entradas/Pasos de Ejecución: En el listado de las auditorías existentes, se selecciona una de ellas. El sistema muestra toda la información que describe a la auditoría: general, documentos, riesgos, notificaciones asociadas y el progreso en cuanto a tiempo.	
Resultado Esperado: Se visualiza correctamente la información del desarrollo de la auditoría.	
Evaluación de la Prueba: Bien.	

Anexo 5. Aval del Centro de Soporte

UCI
Universidad de las Ciencias Informáticas

CENTRO DE SOPORTE

Aval del Centro de Soporte

Por este medio avalo que la Tesis de grado "Subsistema de Gestión de Auditorías del Sistema Automatizado de Monitoreo y Control de Servicios del Centro de Soporte UCI" de los Tesistas Dairis Almaguer Pérez y Danilo Rodríguez Díaz es de gran importancia para el centro de soporte y la aplicación generada formará parte del nuevo Sistema Automatizado de Monitoreo y control de los Servicios . El sistema cuenta con los requerimientos especificados por las metodologías de gestión de servicios a nivel internacional además de un buen diseño.

Sobre el Trabajo de Diploma realizado considero que favorece la calidad de la Gestión de servicios , sirviendo de soporte y ayuda para el equipo de desarrollo de este proyecto productivo que dará continuación a este Trabajo de Diploma presentado, sentando las bases para una futura construcción del sistema concebido.

Los diplomantes han puesto en evidencia los conocimientos adquiridos siendo capaces de ponerlos en práctica para solucionar la problemática planteada. Han mantenido una actitud muy responsable y comprometida con la investigación, logrando realizar un excelente trabajo individual y en equipo.

Y para que así conste, se da por aceptada en fecha 20/5/2013 por .

Director del Centro de Soporte
Firma

GLOSARIO DE TÉRMINOS

Acuerdo: Es un documento que describe un entendimiento formal entre dos o más partes. Un acuerdo no es jurídicamente obligatorio, a menos que forme parte de un contrato.

Acuerdo de niveles de servicio (SLA): Es un acuerdo entre el proveedor de servicios de TI y un cliente. Un acuerdo de niveles de servicio describe los servicios de TI, documenta los objetivos de nivel deservicio, y especifica las responsabilidades del proveedor de servicios de TI y el cliente. Un acuerdo único puede cubrir múltiples servicios de TI o varios clientes.

Auditoría: Es una función de dirección cuya finalidad es analizar y apreciar, el control y funcionamiento interno de las organizaciones. Su finalidad principal consiste en garantizar la integridad de su patrimonio, la veracidad de su información y el mantenimiento de la eficacia de sus sistemas de gestión.

Cliente: Alguien que compra bienes o servicios. El cliente de un proveedor de servicios de TI es la persona o grupo que define y acuerda los objetivos de nivel de servicio. A veces, el término también se utiliza de manera informal para referirse al usuario.

Gestión de servicios de TI: Es la implementación y gestión de la calidad de los servicios de TI que cumplan las necesidades del negocio. La gestión de servicios de TI se lleva a cabo por los proveedores de servicios de TI a través de una combinación adecuada de personas, procesos y tecnología de información.

Procedimiento: Es un documento que contiene pasos que especifican cómo llevar a cabo una actividad. Los procedimientos se definen como parte de los procesos.

Proceso: Es un conjunto estructurado de actividades diseñadas para lograr un objetivo específico. Un proceso tiene una o más entradas definidas y las transforma en salidas definidas. Puede valerse de cualquier rol, responsabilidad, herramientas y controles de gestión que sean necesarios para entregar de forma confiable los resultados. Un proceso puede definir, si son necesarios, políticas, normas, directrices, actividades e instrucción de trabajo.

Servicios: Es un medio de entregar valor a los clientes, al facilitar los resultados que los clientes quieren lograr sin apropiarse de los costos y riesgos específicos. A veces se utiliza el término “Servicio” como sinónimo de servicio base, servicio de TI o paquete de servicios.

Servicio de TI: Es un servicio proporcionado por un proveedor de servicios de TI. Un servicio de TI se compone de una combinación de tecnología de información, personas y procesos. Los servicios de TI de cara-al-cliente dan soporte directo a los procesos del negocio de uno o más clientes y sus objetivos de niveles de servicio deben definirse en un acuerdo de nivel de servicio. Otros servicios de TI, llamados servicios de soporte, no son utilizados directamente por el negocio, pero el proveedor de servicios los requiere para entregar los servicios de cara-al-cliente.

ACRÓNIMOS

BPR: Reingeniería de Procesos.

CMMI-SVC: Modelo de Madurez y Capacidad Integrada para Servicios.

COBIT: Objetivos de Control para las Tecnologías de la Información.

HTTPS: Protocolo de Transferencia de Hipertexto Seguro.

IEC: Comisión Electrotécnica Internacional.

IP: Protocolo de Internet.

ISO: Organización Internacional de Normalización.

ITIL: Biblioteca de Infraestructura de Tecnologías de la Información.

SLA: Acuerdo de Nivel de Servicio.

TCP: Protocolo de Control de Transmisión.

TI: Tecnologías de la Información.

UDDI: Directorio de Servicios web de la UCI.

XP: Programación Extrema.