Universidad de las Ciencias Informáticas

Facultad 4

Título:

Desarrollo del Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del Centro FORTES

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

Autores:

Mirian del Carmen González Elias

Sandy Guerra Fernández

Tutor:

Ing. Roberto Carlos Aballe Ochoa

Co-Tutora:

MSc. Yuniet Del Carmen Toll Palma

Ciudad de La Habana, 11 Junio del 2012

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____
Mirian del Carmen González Elias
Autor

_____
Sandy Guerra Fernández
Autor

_____
Ing. Roberto Carlos Aballe Ochoa
Tutor

_____
MSc. Yuniet Del Carmen Toll Palma
Co-Tutora

AGRADECIMIENTOS

A nuestro tutor y nuestra co-tutora muchas gracias por toda su ayuda y comprensión, los que a fuerza de regaños y exigencias nos ayudaron a salir adelante en los momentos en que se veía todo oscuro.

A las profes Maria Elena y Mayi por estar ahí en los momentos difíciles dándonos el consejo adecuado y enseñándonos a hacer las cosas de la manera correcta.

Mirian:

A mis padres, Julio y María, por darme la vida, por su apoyo incondicional y aceptarme como soy. No hay palabras para expresar mi agradecimiento ni todo lo que siento por ustedes, infinitas gracias.

A mí querida hermanita, que por ella he llegado hasta el final para darle un futuro mejor y poder cuidarla.

A mis abuelitos, que me han dado fuerzas para seguir y que están muy orgullosos de verme ingeniera.

A mi tía Dalmis por sus sabios consejos y motivación a seguir estudiando, este logro se lo debo en parte a ella.

A mi tío Pablito, que sé que está orgulloso de que termine la universidad.

Al resto de mi familia que forma directa o indirecta me ayudaron a llegar hasta aquí, muchas gracias.

A mí novio Rolando por aguantarme y apoyarme en los momento difíciles, por todos tus consejos.

A mi antigua tutora Greisy, que el tiempo no le alcanzó para ver este momento y sé que hubiera estado muy orgullosa de verme ingeniera.

A todos mis amigos de la escuela y del barrio, quisiera mencionarlos a todos, pero es imposible escribir todos los nombres, muchas gracias por el simple hecho de existir y cree en mí.

A todos los profesores que me dieron clases, gracias por todo lo que me enseñaron.

A mi compañero de tesis Sandy bello por darme fuerza, comprensión y apoyo, muchas gracias.

Sandy:

Creo que no podría comenzar de otra forma que no sea agradeciendo a la persona más especial en mi vida, sin la cual no hubiese llegado hasta aquí, ni estuviese escribiendo estas líneas: mi abuela Eva, a la que nunca le podré estar lo suficientemente agradecido por todo lo que ha hecho por mí. Ha sido mi madre, mi padre y mi conciencia en los momentos más difíciles. Me ha apoyado en todas mis decisiones, siempre que han estado correctas. Ha vivido para mí, para complacer todos mis gustos. Nunca podré encontrar la forma de demostrarle cuanto la quiero y le estoy agradecido.

A mis padres, por confiar en mí en cada momento, por su inmenso cariño y apoyo en estos años, por hacerme mucho mejor la vida todo este tiempo y por todos sus consejos.

A mis amigos, pocos pero con el corazón de un pueblo. A Yonder, que más que mi amigo es mi consejero y hermano. A Mara y Lirisandra, más que mis amigas, mis hermanitas, las que han estado ahí cuando he necesitado un consejo y me han enseñado que a la vida hay que mirarla con los lentes correctos. A Yudiel, el Chiqui, Felito, Geiber y Alex, amigos de los buenos. A Laly, Yura, Lisbet y Capullo, mis compañeras del pre, de las que hoy me siento orgullo de haber conocido. A Aymee, Cuni, Deilys, otro tanto especiales durante estos cinco años de carrera. A mi gemelo Gabriel, quien se ha convertido, en menos de dos cursos, en el amigo que todos quisieran tener. A Yoan, por ser la persona con que más he discutido en un curso, y en especial, por ser mi amigo.

A Lisbethy, más que mi amiga, la que ha estado cada vez que he tenido un problema para darme ánimos y que se ha portado conmigo como pocos lo han hecho.

A mis compañeros de aula y de apartamento en la UCI, desde primer año hasta momento, todos los que he compartido.

A mis vecinos, en especial a Aleja, Leopo, Raquel, Lisy y Sandra, por ayudarme a comprender las cosas buenas de la vida y darme aliento en los momentos de flaqueza.

A mi familia por estar ahí cuando más lo he necesitado, en especial mis tíos Lito e Iván y mi tía Yudith.

Agradecimientos

A mi segundo padre, Ramón, y a mi segunda madre, Eva, que más que mi madrastra ha sabido ser toda una madre.

A mi profesora y casi madre, Hilda, que ha estado presente en los momentos más difíciles de mi vida mostrándome el camino a transitar, apoyando en mis decisiones y regañándome cuando ha hecho falta. A mi profe Maura.

A los muchachos del teatro por permitirme soñar un poco durante estos años, en especial a Ernesto, Linet y Arcadio, este último es una de las personas que más admiro, por ser uno de esos que solo con hablar te hace sentir que no hay cosas imposibles cuando sobran ganas de hacer.

A mi compañera de tesis Mirian por su apoyo y comprensión.

DEDICATORIA

Mirian:

A mis padres Julio y María.

A mi hermanita Mailen.

A mis abuelos Pablo y Miriam.

A mi tía Dalmis.

A mi novio.

Sandy:

A mi abuela Eva, por ser tan especial.

A mis amigos.

A mi familia.

RESUMEN

Los procesos de auditoría y revisión de *software* permiten detectar errores y fallos en el proceso de desarrollo. El Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del Centro de Tecnologías para la Formación (FORTES) responde a la necesidad de mejorar el control y evaluación de los proyectos de dicho centro. Se realizó un estudio del estado del arte sobre sistemas homólogos para los procesos de auditoría y revisión, se obtuvo como resultado que las herramientas para realizar auditorías a nivel internacional son privativas y no fueron diseñadas para realizar auditorías de *software,* mientras que a nivel nacional no permite gestionar los planes y cronogramas de auditoría y revisión, ni evaluar auditoría. En el proceso de desarrollo se utilizaron las siguientes herramientas y tecnologías: como Sistema de Gestión de Contenidos (CMS) Drupal, pues permitió reducir el tiempo de implementación de las Historias de Usuario; como Sistema Gestor de Base de Datos: PostgreSQL; como servidor *web*: Apache HTTP Server. La metodología de desarrollo: SXP, obteniéndose los diferentes artefactos que esta propone; como lenguajes de programación: PHP, Javascript, XHTML y CSS. El entorno de desarrollo seleccionado fue Netbeans. Se obtuvo como resultado un módulo que permite un mayor control de los documentos generados en el proceso de auditoría o revisión, así como la evaluación de los proyectos que son inspeccionados. Además, se logró estandarizar la gestión de información de ambos procesos.

Palabras Clave: Auditoría de *Software*, Drupal, Revisión de *Software*

ÍNDICE DE CONTENIDO

INTRODUCCIÓN:

Con el creciente desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), crece la industria del *software* que, con el transcurso del tiempo, ha llegado a convertirse en el pilar fundamental de la economía de muchos países. Posibilitando la interconexión entre las personas e instituciones a nivel mundial eliminando barreras espaciales y temporales. Las TIC abren nuevas posibilidades de investigar e innovar, que junto con la intrepidez, curiosidad y motivación del usuario permiten obtener nuevas creaciones, potenciando la imaginación, habilidades comunicativas y colaborativas. Permitiendo el acceso a mayor cantidad de información y a herramientas para el desarrollo integral.

Con el avance de las TIC, cada día se perfeccionan las tecnologías. El desarrollo, a su vez, posibilita un crecimiento en la producción de *software*, lo que implica que el desarrollo de la producción de recursos informáticos no se encuentra exento de introducción de posibles errores. Es de vital importancia resaltar la necesidad de revisar, controlara e inspeccionar el trabajo en las empresas e instituciones dedicadas a la producción de *software*. Para ello se realizan los procesos de auditoría y revisión bajo regulaciones y leyes, para no cometer ningún error en el proceso de control, almacenamiento de datos y revisión. De esta forma se puede evitar la pérdida de datos por mal manejo y desorganización de la información.

Las auditorías permiten identificar las deficiencias que puedan presentar los productos de *software*, así como reportar las sugerencias correspondientes a las oportunidades de mejora encontradas para evaluar el nivel del cumplimiento y el impacto de las recomendaciones hechas. Según Nubia Fernández Grajales en el 2005, "*Puede afirmarse que el éxito de las auditorías de software mejora la imagen de la empresa, genera confianza en los usuarios de sus productos, optimiza las relaciones internas y del clima de trabajo y disminuye los costos de la mala calidad"* (1).

Según IEEE1028-1997, la revisión del *software* es: "un proceso o una reunión durante los cuales un producto de *software* es [examinado cerca] personal del proyecto, encargados, usuarios, clientes, representantes del usuario, u otros partidos interesados para el comentario o la aprobación" (2).

A partir de las definiciones de auditorías y revisiones se decidió realizar un estudio sobre las principales empresas que llevan la delantera a nivel mundial en la creación de herramientas para la realización de auditorías, aunque no están diseñadas para auditorías de *software* sino financieras. Entre las empresas se

encuentran ACL *Services Ltd.*, *Thomson Reuter Accelus*, SIT (Sistemas Informático). Los principales *software* creados por estas empresa son ACL (en español: Lenguaje de Comandos de Auditoría), creado por ACL Services, AutoAudit por Thomson Reuter Accelus y Audita de SIT. Todas estas herramientas fueron creadas para realizar auditorías financieras.

A pesar que las herramientas mencionadas tienen características y funcionalidades que pueden ser de utilidad para el trabajo de los auditores, constituye una limitante para nuestro país el uso de las mismas, pues no son multiplataforma y además, se está llevando a cabo un proceso de migración hacia *software* libre, por lo que la tendencia apunta al uso de herramientas libres.

En Cuba el desarrollo de las actividades de producción de *software* ha aumentado rápidamente durante los últimos años, esto se ha logrado mediante la creación de centros dedicados al desarrollo de esta actividad. Algunos de los centros cubanos que producen *software* para el mercado nacional e internacional son la Empresa Productora para la Técnica Electrónica (SOFTEL), Unidad de Compatibilización Integración y Desarrollo de *Software* (UCID-FAR), Centro Nacional de Calidad de *Software* (CALISOFT). Estas entidades fueron establecidas para aumentar el desarrollo tecnológico del país y la escalabilidad en el mercado. Los productos creados han tenido un impacto relevante en la educación, la salud, el deporte y la economía en general, por lo que es necesario que estos productos al ser liberados tengan la calidad requerida, esto se puede lograr con la ayuda de las auditorías y revisiones de software.

En SOFTEL se realizan revisiones manuales utilizando las normas ISO 9126 e ISO 2919, pues no utilizan ninguna herramienta para apoyar este proceso y tampoco han desarrollado alguna para este fin. A partir del mes de abril del 2012 también se realizarán auditorías de *software*, para lo cual tampoco cuentan con una herramienta. El grupo de especialistas encargados de realizar esta tarea utiliza el Redmine, pero solo como apoyo a la ejecución manual que practican.

En UCID-FAR se realizan ambos procesos. Las revisiones son ejecutadas después de cada fase y las auditorías según las peticiones de los proyectos que pertenecen al centro. Los responsables se auxilian del Redmine para gestionar las no conformidades detectadas, así como para darle seguimiento.

En CALISOFT se realizan auditorías y revisiones apoyándose en GESOFT, una herramienta creada por un grupo de especialistas del centro. Se utiliza para apoyar las revisiones y también utilizan herramientas como el Redmine y el Trac (herramienta para almacenar y dar seguimiento a las no conformidades).

En la delantera de la industria de *software* se encuentra la Universidad de las Ciencias Informáticas (UCI), que ha alcanzado un nivel discreto de competitividad en el mercado de soluciones informáticas como resultado del trabajo que se realiza en los diferentes proyectos productivos. En pos de lograr mayor cohesión y organización en los procesos de desarrollo de *software*, la UCI organiza la infraestructura productiva mediante la creación de los centros de desarrollo. Como parte de este proceso surge el Centro de Tecnologías para la Formación (FORTES). Su misión principal es desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las TIC, a todo tipo de instituciones con diferentes modelos de formación y condiciones tecnológicas (3).

Dentro de las características que hereda FORTES, al igual que el resto de los centros de desarrollo, del proceso productivo en la UCI, se aprecia la organización rigurosa del proceso de desarrollo de *software* y la puesta en marcha de buenas prácticas de ingeniería de *software* y gestión de proyecto. Durante el ciclo de vida del *software* se genera una serie de documentos que serán guardados como evidencia en un expediente de proyecto, que se actualiza constantemente. Con esta política organizativa se pretende garantizar que los productos entregados tengan la calidad suficiente y que exista una mayor organización y control del estado de los proyectos durante su ciclo de desarrollo para elevar el prestigio internacional.

Este centro de desarrollo está integrado por varios proyectos. Entre ellos se encuentra el Grupo de Calidad, el cual tiene como misión: “desarrollar actividades de aseguramiento de la calidad para la orientación más acertada, monitoreo y control del proceso de desarrollo de productos en FORTES, con el ofrecimiento de servicios clasificados en: Estratégicos (revisiones y auditorías), Fundamentales (pruebas de *software*) y de Soporte (asesoramiento) tanto a proyectos productivos del centro como a cualquier interesado en las funciones del Departamento y con el apoyo de un laboratorio de pruebas que respalde dichos servicios; garantizando así, la calidad de las soluciones y la formación de los recursos humanos (estudiantes y profesores) partiendo de las investigaciones internas que combinen los elementos pedagógicos y tecnológicos más avanzados e integrando los procesos de formación, producción e investigación que requiere el modelo de formación profesional en la UCI” (4).

Actualmente en FORTES, los procesos de auditoría y revisión se realizan de forma manual, lo cual provoca lentitud a la hora de completar el servicio. Se hace complejo llevar los datos estadísticos de los proyectos, departamentos y a nivel de centro y no existe una forma de extraer y manejar esta información con vistas a confeccionar reportes, con formato estándar, de los resultados de las auditorías y revisiones. Esta situación trae como consecuencia que en la mayoría de los casos no se logre terminar con el cierre de ambos procesos en la fecha prevista.

Teniendo en cuenta lo anteriormente expuesto se plantea el siguiente **problema de investigación:** ¿Cómo mejorar el control y evaluación de los proyectos del centro FORTES?

El problema planteado se enmarca en el **objeto de estudio**: los procesos de auditoría y revisión de *software*, se tiene como **campo de acción**: los procesos de auditoría y revisión de *software* en el centro FORTES.

El **objetivo general** es desarrollar un módulo que informatice los procesos de auditoría y revisión de *software* del centro FORTES.

Teniendo en cuenta lo anterior, se propone la siguiente **idea a defender**: El Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del centro FORTES, mejorará el control y evaluación de los proyectos que pertenezcan a este centro.

Los **objetivos específicos** son:

- Analizar los principales conceptos de auditorías y revisiones y las herramientas existentes dedicadas a la realización de ambos procesos.
- Analizar las tecnologías y herramientas existentes para desarrollar aplicaciones *web*.
- Describir la propuesta del módulo.
- Implementar y validar un módulo que sirva de apoyo a los procesos de auditoría y revisión de *software*.

Posibles resultados:

- Se obtendrá un módulo que permitirá mejorar el control y evaluación de los proyectos del centro FORTES.

Tareas a cumplir por estudiantes:

- Determinación de los principales conceptos de auditorías de *software*.
- Determinación de los principales conceptos de las revisiones de *software*.
- Determinación de las herramientas existentes dedicadas a la realización del proceso de auditorías.
- Determinación de las herramientas existentes dedicadas a la realización del proceso de revisiones.
- Análisis crítico de las características principales de las diferentes metodologías de desarrollo de *software* para determinar la adecuada para la solución del problema.
- Selección de las herramientas y del lenguaje de programación a utilizar.
- Selección del Entorno Integrado de Desarrollo (IDE) a utilizar.
- Definición y descripción de las funcionalidades y características que tendrá la aplicación.
- Realización de la fase Desarrollo de la metodología SXP empleada para la implementación del módulo.
- Implementación de las funcionalidades identificadas.
- Elaboración de los Casos de Prueba de Aceptación.
- Ejecución de Pruebas de Aceptación.

Métodos investigativos

En la realización del presente trabajo se han utilizado diferentes métodos científicos para la investigación, entre ellos los métodos teóricos, los empíricos y métodos del nivel estadístico.

Los **métodos teóricos** usados para dar cumplimiento a las tareas a desarrollar son:

Analítico-sintético: en todo el proceso investigativo, principalmente en la precisión de los fundamentos teóricos relacionados con los sistemas informáticos para los procesos de auditoría y revisión de *software* y el análisis e interpretación de los resultados obtenidos con la utilización del módulo para ejecutar ambos procesos.

Los **métodos empíricos** utilizados para obtener información sobre el objeto de estudio son:

Encuesta: permitió realizar un análisis de las posibles soluciones existentes, mediante preguntas escritas a un grupo de personas de los centros de desarrollo de *software* con conocimientos del tema.

Los **métodos del nivel estadístico** utilizados son:

- *Estadística descriptiva:* se utilizó para determinar la evaluación de cada auditoría, por parte del sistema, mediante la utilización de cálculos matemáticos.
- *Estadística inferencial:* para la toma de decisiones por parte del asesor de calidad sobre el rechazo o no de los datos que se obtengan de la evaluación de la auditoría según el proyecto sometido a evaluación.

El presente documento está estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica

Se construye el marco teórico y referencial de la investigación, referido a la evaluación de los procesos de auditorías y revisiones a partir del análisis de la bibliografía consultada. Los referentes teóricos de la investigación se centran en los siguientes aspectos: las definiciones de los conceptos de las auditorías y revisiones, las herramientas existentes dedicadas a la realización de ambos procesos., el análisis de las principales tecnologías y herramientas existentes para desarrollar aplicaciones *web* y la identificación de las funcionalidades que tendrá el módulo.

Capítulo 2: Análisis de la solución propuesta

Se realiza el análisis de la solución propuesta. Se presenta la Lista de Reserva del Producto, en la que se priorizan los requerimientos funcionales a incluir. Además se describen las funcionalidades que el sistema debe ejecutar, mediante las Historias de Usuarios. También se muestra la descripción de la Plantilla Plan de *Releases*, de las Plantillas Tareas de Ingeniería y Pruebas.

Capítulo 3: Implementación y prueba de la solución propuesta

Contiene el Plan de *Releases* con la planificación y duración que tendrá cada Historia de Usuario. Se expone el contenido referente a las Tareas de Ingeniería. Se explica la realización de las Pruebas de Aceptación y la confección de los Casos de Prueba. Además, se detallan los temas de tratamiento de errores, roles de usuario, estándar de código utilizado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

Con el desarrollo de las tecnologías usadas en las diferentes ramas que sustentan la sociedad se ha producido un desarrollo paralelo de las técnicas para evaluar y controlar la información que se maneja. Las tecnologías de la informática, en la producción de *software*, no han estado exentas de la aplicación de estas técnicas para asegurar el cumplimiento de normas y criterios que garanticen que el producto pueda competir en el mercado.

En el presente capítulo quedan expuestos aspectos relacionados con la auditoría y la revisión *software*. Se expone la situación actual de las herramientas para la realización de auditorías a nivel nacional e internacional, así como tecnologías informáticas utilizadas para la Planificación-Definición y posterior Desarrollo que se propone.

1.2. Definición de Auditoría

Una de las características principales de la sociedad actual es el gran caudal de información que poseen. Para que esta información sea confiable es necesario que cumpla con ciertas garantías que le ofrezcan un alto grado de confianza para la institución u organización que las utilice. Este problema es posible solucionarlo a través del uso de las auditorías.

La palabra auditoría proviene del latín *audire* (“oír”), que hace referencia a la forma en que los primeros auditores cumplían con su función (escuchaban y juzgaban la verdad o falsedad de lo que era sometido a su verificación) (5).

Varios autores han aportado su propia definición. Una de ellas es la planteada por Ivan Dimitrie Moyasevich B, una auditoría es: “Es un examen crítico que se realiza con el fin de evaluar la eficacia y eficiencia de una sección, un organismo, una entidad” (6). Por otra parte, el autor Gustavo Alonso Cepeda la define como: “La recopilación y evaluación de datos sobre información de una entidad para determinar e informar sobre el grado de correspondencia entre la información y los criterios establecidos. La auditoría debe ser realizada por una persona competente e independiente” (7).

Sin embargo, Enrique Hernández propone que: “Es un proceso formal y necesario para las empresas con

el fin de asegurar que todos sus activos sean protegidos en forma adecuada. Asimismo, la alta dirección espera que de los proyectos de auditoría surjan las recomendaciones necesarias para que se lleven a cabo de manera oportuna y satisfactoria las políticas, controles y procedimientos y definidos formalmente, con objeto de que cada individuo o función de la organización opere de modo productivo en sus actividades diarias, respetando las normas generales de honestidad y trabajo aceptadas" (8).

En la norma ISO 19011:2002 se define auditoría como el proceso sistemático, independiente y documentado para obtener evidencias y evaluarlas de manera objetiva con el fin de determinar la extensión en que se cumplen los criterios de auditoría (9).

Luego de analizar diferentes definiciones de auditoría los autores de la investigación consideran que la auditoría es el proceso de recopilar información sobre una determinada entidad, con el objetivo de realizar una evaluación, teniendo en cuenta criterios preestablecidos y la confección de un informe con los resultados de la misma, que contenga las conclusiones de este proceso evaluativo y las recomendaciones necesarias. Mediante el proceso de auditorías es posible recopilar los datos necesarios para determinar si la labor de los responsables de esta información se está realizando de manera correcta y con ello se garantiza el éxito de la empresa. Es un proceso que se encarga de asesorar a la dirección de la organización en cuanto a la gestión de la información necesaria para la toma de decisiones. Un proceso de auditoría exitoso permite elevar los niveles de validez, productividad, seguridad y rentabilidad de la empresa, a la vez que aumenta su prestigio.

Para la realización de este tipo procesos se utilizan estándares y normas como la 19011: 2002, que permiten valorar el nivel de correspondencia entre los datos a evaluar y los definidos mediante estándares (por los que se rige el auditor). Estos estándares y normas establecen de forma clara las exigencias de un proceso de auditoría exitoso; explican en detalle los pasos, las actividades y objetivos del mismo, proporcionando al auditor una guía para realizar su trabajo correctamente.

La auditoría depende de varios principios que la convierten en una herramienta efectiva en el apoyo a las políticas de gestión; proporcionando información que permiten a la organización mejorar su nivel de desempeño. Estos principios permiten a los auditores trabajar y plantear sus conclusiones haciendo uso del mismo procedimiento, facilitando el entendimiento para realizar el informe final, lo cual constituye un requisito de obligatorio cumplimiento.

Se definen una serie de etapas para realizar las auditorías, una de ellas es determinar el alcance y el objetivo de su ejecución, el estudio del entorno que se auditará, los recursos que serán necesarios para efectuar la auditoría, la elaboración del plan de trabajo y sus actividades. Además, facilita la confección del informe final y la presentación de este

Existen diversas clasificaciones de las auditorías, tales como: las auditorías financieras, las operacionales, las de cumplimiento, las de rendimiento y revisiones especiales. Dentro de las revisiones especiales se encuentran las auditorías informáticas. Estas últimas se detallan a continuación en el epígrafe 1.3 auditorías informáticas.

1.3. Auditoría informática

La auditoría informática es un examen, pues debe partir de una situación dada; éste es metódico, puesto que seguirá un plan de trabajo perfectamente sistematizado que permite llegar a conclusiones suficientemente justificadas (está es una conclusión exigible a cualquier auditoría). La misma es puntual, pues se da un corte en el calendario para llevarla a cabo y obtener la objetividad requerida, por lo que será ejecutada por personas ajenas al departamento, que no guarden relación con las funciones a auditar (10).

Según Gabriel Buades en el 2002, auditoría informática es: “Es el conjunto de técnicas, actividades y procedimientos destinados a analizar, evaluar, verificar y recomendar en asuntos relativos a la planificación control eficacia, seguridad y adecuación del servicio informático en la empresa, por lo que comprende un examen metódico, puntual y descontinuo del servicio informático, con vistas a mejorar en: rentabilidad, seguridad y eficacia” (11).

Por su parte Marisa Haderne, plantea que: “La auditoría informática es la revisión y evaluación de los controles, sistemas, procedimientos y de los equipos de cómputo, su utilización, eficiencia y seguridad, con el fin de que por medio del señalamiento de cursos alternativos se logre una utilización más eficiente y segura de la información que servirá para una adecuada toma de decisiones” (10).

Coincidimos con lo planteado por Gabriel Buades que: la auditoría informática se desarrolla a través de normas, procedimientos y técnicas definidas por institutos establecidos a nivel nacional e internacional. Permite llevar el control de la eficiencia y seguridad de la empresa que es auditada, para lograr una mejor

organización.

Los principales objetivos de la auditoría informática son: el control de la función informática; el análisis de la eficiencia de los sistemas informáticos; la verificación del cumplimiento de la normativa en este ámbito; La revisión de la eficaz gestión de los recursos informáticos.

En la rama de la informática existen diferentes tipos de auditorías que se realizan a los sistemas:

- Auditoría de la gestión.
- Auditoría legal del Reglamento de Protección de Datos.
- Auditoría de los datos.
- Auditoría de las bases de datos.
- Auditoría de la seguridad.
- Auditoría de la seguridad en producción.

1.3.1. Auditoría de Software

La Auditoría de *Software* es un término general que se refiere a la investigación y al proceso de entrevistas que determina cómo se adquiere, distribuye y usa el *software* en la organización. “El propósito de una auditoría de *software* es proporcionar una evaluación independiente a la conformidad del producto y los procesos de sus regulaciones aplicables, estándares, directrices, planes y procedimientos” (2) .

La norma lEEE Std 1028-1997 establece que: *“La auditoría es una actividad organizada formalmente, con participantes que tienen funciones específicas, tales como el auditor principal, otro auditor, un registrador o un iniciador, e incluye un representante de la organización auditada. La auditoría identificará los casos de no conformidad y producirá un informe el cual requiere que tome medidas correctivas correspondientes”* (12).

El equipo de desarrollo coincide con los autores citados, los que le permitieron determinar que: la auditoría de *software* el proceso de investigación y entrevistas que se realiza sobre un producto de *software*, para determinar el nivel de cumplimiento de las normativas preestablecidas y detectar no conformidades, con el objetivo de proporcionar una evaluación, tomando como base los resultados obtenidos.

1.4. Definición de Revisión

Existen varios tipos de revisiones como es el caso de las revisiones entre pares (*Peer Review*), la cual es un método complejo y riguroso de gran importancia que se ha usado para validar trabajos científicos escritos por un grupo de expertos. El propósito de la revisión de pares es medir la calidad, factibilidad y credibilidad de las investigaciones, con vistas a ser publicadas en libros, revistas, periódicos y sitios *web* (13).

Se entiende por revisión, una observación que se hace con cuidado y atención con el objetivo de corregir errores. Además se puede definir como: una prueba que se realiza para comprobar que algo funciona de forma correcta. La revisión constituye una etapa que comprende la comprobación de que lo que está escrito corresponde con lo establecido.

Una revisión es una forma de aprovechar la diversidad de un grupo de personas para: determinar la necesidad de mejoras en el producto de una persona o de un equipo de trabajo; reconocer las partes del producto en las que no es necesaria o no es deseable una mejora; alcanzar un trabajo de mayor calidad maximizando los criterios de Correctitud y Completitud fundamentalmente.

1.4.1. Revisión de *software*

En la Norma ISO 9000:2000 se define revisión de *software como:* "*Actividad emprendida para asegurar la conveniencia, adecuación y eficacia del tema objeto de la revisión, para alcanzar unos objetivos establecido*s" (14).

La revisión sistemática es un método de investigación desarrollado para obtener, analizar y evaluar toda la investigación relevante para una pregunta de investigación o un área de interés particular. (15)

Las Revisiones de *Software* son un "filtro" para el proceso de Ingeniería del *Software*. Las revisiones se aplican en diferentes momentos del desarrollo y sirven para detectar errores y defectos que puedan ser eliminados. Sirven también para "purificar" las actividades de Ingeniería del *Software* que suceden como resultado del análisis, el diseño y la codificación. (16)

De forma general se puede definir a las revisiones como actividades de control que se realizan de forma sistemática dentro del proceso de desarrollo de *software* con el objetivo de detectar errores en la documentación y el sistema. Las revisiones de *software* sirven para validar la calidad y el estado de un

producto, que puede ser un documento, un módulo o un prototipo. Existen distintos tipos de revisiones que se pueden realizar a lo largo del ciclo de vida del *software*, cada una especializada en un determinado producto. Las revisiones ayudan a encontrar errores, y a identificar riesgos, que serían difíciles de encontrar de otra manera.

Tipos de Revisiones

Existen varios tipos de revisiones entre los que se encuentran las revisiones formales e informales (17):

Informales

- No hay proceso definido
- No existen roles
- Usualmente no son planeadas

Formales

- Objetivos definidos
- Proceso documentado
- Roles definidos y personas entrenados en ellos
- Se utilizan listas de chequeo, reglas y métodos para encontrar defectos
- Reporte del resultado
- Recolección de datos para el control del proceso

El objetivo principal de las revisiones técnicas formales o inspección es encontrar errores durante el proceso, para evitar que se conviertan en defectos después de la entrega del *software*. Estas inspecciones permiten encontrar errores en cada iteración, para que no repercutan en la siguiente fase del proceso de desarrollo de *software*.

Las revisiones técnicas consisten en seis pasos (18):

1. Planificación: Cuando el desarrollador completa un "producto de trabajo", se forma un grupo de inspección y se designa un moderador. El moderador asegura que el "producto de trabajo" satisfaga el criterio de inspección. Se le asignan diferentes roles a las personas que integran el grupo de inspección, así como la planificación de tiempos y recursos necesarios.
2. *Overview*[1]: Si los inspectores no están familiarizados con el desarrollo del proyecto, es necesaria

[1] ***Overview*** (significado en español: visión global, perspectiva general, información general)

una vista general.

3. Preparación: Los inspectores se preparan individualmente para la evaluación en la reunión, estudiando los productos de trabajo y el material relacionado. En este caso es aconsejable la utilización de listas de chequeos para ayudar a encontrar defectos comunes. El tiempo que pueda llevar esta etapa va a depender de cuan familiarizado esté el inspector con el trabajo que debe analizar.
4. Examen: En esta etapa, los inspectores se reúnen para analizar su trabajo individual en forma conjunta. El moderador deberá asegurarse que todos los inspectores están suficientemente preparados. La persona designada como lector presenta el "producto de trabajo", interpretando o parafraseando el texto, mientras que cada participante observa en busca de defectos. Es recomendable que este examen no dure más de dos horas ya que la atención en busca de defectos va disminuyendo con el tiempo. Al terminar con la reunión, el grupo determina si el producto es aceptado o debe ser retrabajado para una posterior inspección.
5. Retrabajo: El autor corrige todos los defectos encontrados por los inspectores.
6. Seguimiento: El moderador chequea las correcciones del autor. Si el moderador está satisfecho, la inspección está formalmente completa, y el "producto de trabajo" es puesto bajo el control de configuración.

Las auditorías y revisiones juegan un papel fundamental en el desarrollo de un producto de *software*. Estos procesos aseguran el cumplimiento de los procedimientos y estándares establecidos. Permiten descubrir de forma temprana defectos que puedan tener el producto y determinar el nivel de impacto de los mismos. Facilitan la evaluación sistemática de las 4 P (Producto, Proceso, Proyecto y Persona) a la vez que pueden constituir una vía para la búsqueda de soluciones a los problemas.

Los procesos de auditorías y revisiones se pueden realizar de manera manual o automáticas. Por lo que se necesitó estudiar diversas herramientas automáticas que realicen estos procesos. Para ello se encontraron las siguientes: ACL, AutoAudit y Audita.

1.5. Análisis de herramientas existentes

A nivel mundial existen varias herramientas que ayudan a realizar el proceso de auditoría. A continuación se aborda sobre algunas de ellas.

ACL

ACL (Lenguaje de Comandos de Auditoría) es una herramienta para el análisis y extracción de datos. Es poderoso y fácil de usar, permite importar archivos de diferentes fuentes o formatos, los datos importados no son modificados asegurando la integridad e incrementando el nivel de confianza de los datos trabajados. Además de posibilitar la generación de pistas de auditoría (Quién, Cómo, Cuándo, Dónde), identificar tendencias, señalar excepciones y destacar áreas que requieren atención. (19)

La herramienta ACL no satisface las necesidades del equipo de desarrollo, a pesar de las funcionalidades que brinda, debido a que no fue creado específicamente para auditoría de *software*, por lo que no es aconsejable usarlo como solución.

AutoAudit

AutoAudit es una herramienta de auditoría para la gestión de papeles de trabajo, administración de recursos, tiempos, costos, evaluaciones y generación de reportes integrados con Microsoft Office o Lotus Notes y con alternativa de acceso mediante la *web*. Provee evaluaciones y procesos de niveles de riesgo, programación de equipos de auditoría, planificación de recursos, manejo de observaciones, reporte de gastos, reporte de tiempos, control de calidad, métricas de departamento y la flexibilidad de un módulo de reportes (20).

Ha sido diseñado para automatizar todos los procesos primarios que pueden ocurrir en una auditoría. Contiene una biblioteca estándar para almacenar programas de auditoría, papeles de trabajo, memos y observaciones estándar. Los documentos pueden ser generados incluyendo formatos de Word, Excel, PowerPoint, Visio u otro documento de cualquier aplicación que pueda ser soportada en ambientes Microsoft (20).

AutoAudit posee características de interés para el equipo de trabajo, pero presenta la limitación de que su funcionamiento lo realiza sobre sistemas operativos Windows, lo cual no apoya la política que sigue el país hacia la migración al uso de herramientas libre, por lo que no representa una solución al problema existente.

Audita

Es una herramienta para la administración de auditorías con orientación a riesgos. Creado especialmente para cubrir integralmente la informatización de todas las actividades que el área de auditoría debe realizar

para cumplir con su misión. Se complementa con **Audita2** creado para que los sectores auditados interactúen con el área de auditoría y el comité de auditoría. Fue diseñado para cubrir las regulaciones vigentes para la Entidades Financieras Argentinas tanto del Banco Central de la República Argentina como internacionales. Está realizado para operar en sistemas operativos Windows 2000 en adelante y Windows Vista. Las funcionalidades de Audita son: la Planificación, Gestión de Riesgos, Administración de recursos, Papeles de Trabajo, Informe y Observaciones, Seguimiento de Auditoría y Comité, Reportes (21).

Audita constituye un herramienta sólido para la realización de auditorías, pero ha sido diseñado para auditar entidades financieras y no productos de *software*, lo cual constituye la principal limitación para ser usado como solución del problema, a pesar de que brinda funcionalidades necesarias por el cliente.

En Cuba existe una herramienta que es de gran ayuda para realizar auditorías y revisiones de *software*, esta fue creada por varios especialistas del grupo de calidad de CALISOFT para suplir una necesidad del propio centro, ya que no existía anteriormente ninguna aplicación que ofreciera ayuda a los auditores. La herramienta permite que las auditorías y revisiones de *software* se realicen de acuerdo a la planificación que es elaborada por los diferentes centros de la universidad, los que atendiendo a sus necesidades hacen la solicitud al grupo de calidad de la dirección de CALISOFT. Tiene como objetivo principal verificar que los proyectos cumplan con todas las normas y estándares que son necesarios para que satisfagan todos los requisitos que requiere el cliente.

GESOFT

La herramienta GESOFT permite el almacenamiento de las no conformidades que se encuentran en los proyectos a los que se les aplica la auditoría y la revisión de *software*, de las que se guardan datos tales como: el código para identificar la no conformidad, la fecha, el nombre del proyecto, el tipo de revisiones, el nombre del revisor. También permite darle seguimiento a las mismas por parte del o los revisores que las realizaron. El Grupo de Auditorías y Revisiones (GAR) de CALISOFT utiliza esta herramienta apoyada por otras como: Trac y Redmine que le permiten realizar el trabajo satisfactoriamente. En la actualidad los especialistas de la dirección de calidad del propio centro manifiestan que están satisfechos con la herramienta, pero que en futuras versiones puede ser beneficioso agregarle funcionalidades que permitan almacenar la documentación que está siendo objeto de evaluación.

La herramienta exige a los usuarios la autenticación, definiendo varios roles, brindando integridad y confidencialidad a la información referente a los procesos de auditorías y revisiones que se están ejecutando.

GESOFT constituye una herramienta potente para realizar auditorías y revisiones de *software*. Permite mantener un control sobre las no conformidades encontradas, dándole seguimiento a cada una. Esta herramienta aunque tiene varias funcionalidades que pueden ser de ayudar en la solución del problema planteado no lo satisface, debido a que necesita del apoyo de otras herramientas para el manejo de toda la información necesaria para realizar el proceso; no permite gestionar plan de auditorías y revisiones, ni gestionar cronogramas de auditorías y revisiones, tampoco admite gestionar los expedientes de auditorías y revisiones, además de no generar reportes, ni evaluar auditoría.

Las herramientas informáticas que apoyan el proceso de auditorías y, aunque no todos siguen las mismas políticas y estrategias, de manera general han conformado un conjunto de herramientas muy útiles para las empresas que los utilizan. Estos sistemas presentan como desventaja que no permiten ser utilizados como solución al problema planteado, pues no tienen como objetivo automatizar el proceso de auditorías de *software.*

1.6. Herramientas y tecnologías de desarrollo

Las herramientas y tecnologías no son más que un conjunto de técnicas y recursos de *software* que son utilizados para el desarrollo de aplicaciones, posibilitando el trabajo del equipo. A continuación se brindan características de algunas herramientas y tecnologías de desarrollo que existen, a fin de definir cuál será utilizada para la obtención del producto que se requiere.

1.6.1. Metodología de desarrollo de *software*

Un proceso de *software* detallado y completo según Ivan Jacobson suele denominarse “**Metodología”**. Este término lo define como: Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Las metodologías se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de *software*, que cada vez son más complejos. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de *software*. No existe una metodología de *software* universal (22), las características de cada proyecto (equipo de desarrollo, recursos técnicos y humanos,

tiempo de desarrollo, tipo de sistema) exigen que el proceso sea configurable para lograr un éxito en el proceso de desarrollo.

La comparación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. A grandes rasgos, si se toma como criterio su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, se nombran Metodologías Tradicionales o Pesadas. Aquellas que están centradas en la interacción, comunicación, y en la reducción de la creación de artefactos intermedios son denominadas Metodologías Ágiles.

Metodologías tradicionales

Estas metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo de *software*, con el fin de conseguir un producto más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar, y una vez que está todo detallado, comienza el ciclo de desarrollo del producto *software*. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar (23). Algunas de las metodologías tradicionales más conocidas son: RUP, *Microsoft Solution Franmework* (MSF), *Win-Win Spiral Model e Iconix*.

Metodologías ágiles

Las metodologías ágiles pueden variar en su ejecución y en su énfasis, tiene como característica el desarrollo iterativo. Desarrollar mediante iteraciones permite al equipo de desarrollo adaptarse a los cambios de los requisitos. Trabajar con un alto grado de comunicación permite que el equipo pueda tomar decisiones y aplicarlas inmediatamente. La reducción de artefactos intermedios que no dan valor añadido al producto final, nos permite asignar más recursos al producto en sí y podrá ser terminado antes (24).

Son basadas en heurísticas provenientes de prácticas de producción de código, especialmente preparados para cambios durante el proyecto, impuestas internamente (por el equipo), proceso menos controlado, con pocos principios, No existe contrato tradicional o al menos es bastante flexible, el cliente es parte del equipo de desarrollo, grupos pequeños (menos de 10 integrantes) y trabajando en el mismo

sitio, pocos artefactos y roles (25).

Los principios del manifiesto ágil son (26)**:**

- Se valora al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Las personas son el principal factor de éxito de un proyecto de *software*.
- Desarrollar un *software* que funcione es más importantes que conseguir una buena documentación. Los documentos deben ser cortos y centrarse en lo fundamental.
- La colaboración con el cliente es esencial, más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo.
- Responder a los cambios, más que seguir estrictamente un plan. La planificación no debe ser estricta, sino flexible y abierta.

Algunas de las metodologías ágiles que existen son: *Adaptative Software Development, Agile Modeling*, *Crystal Methods,* SCRUM, *Extreme Programming, entre otras*.

Para la realización de este trabajo de diploma, es necesario obtener resultados a corto plazo, permitiendo que el cliente evalúe el progreso, lo cual permite adaptar la aplicación a los cambios que surgen a lo largo del ciclo de vida. Después de realizar un análisis profundo de las metodologías existentes, el equipo de desarrollo decidió utilizar metodologías ágiles tales como: SCRUM, XP y SXP para el desarrollo del sistema, teniendo en cuenta el corto tiempo de que se dispone, que el cliente es parte del equipo de desarrollo, lo que permite la creación de un sistema completo y con calidad, y se centrará la atención en la programación más que en la documentación.

SCRUM

SCRUM es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos (27).

La metodología SCRUM se divide en tres fases las cuales son: Planificación, Desarrollo y Entrega, en las cuales se efectúan varias reuniones, donde se elabora la lista de todos los cambios requeridos sobre un producto, se determina que tareas deben desempeñarse para cumplir el objetivo de cada sprint (iteración),

entre otros artefactos que se genera en cada sprint. En la figura 1 se muestra el Proceso de Desarrollo de *Software de* SCRUM.

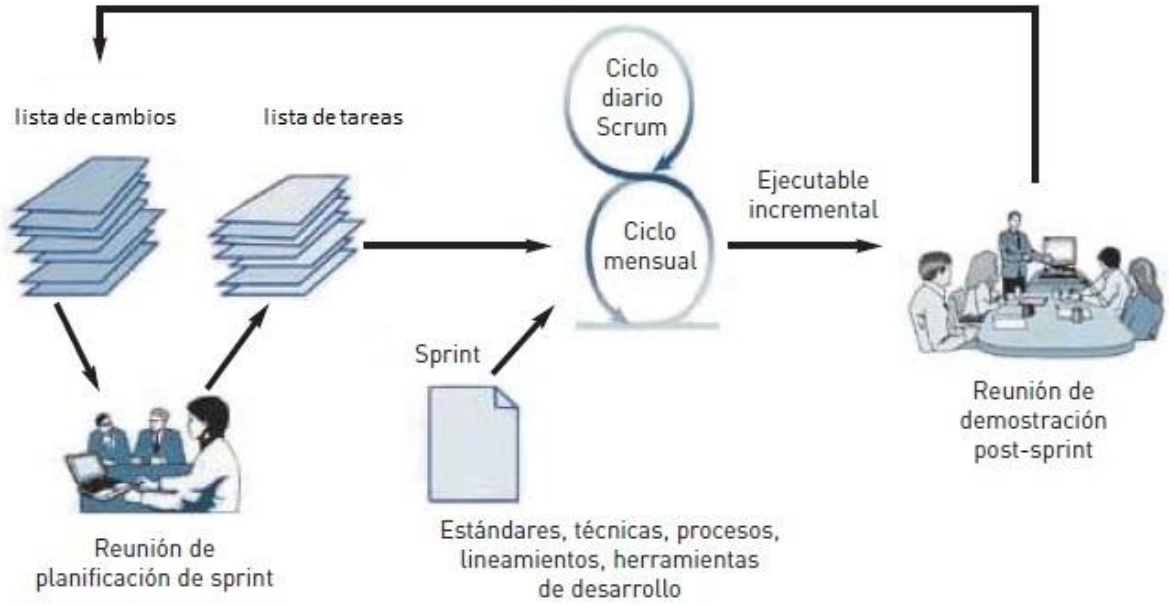


Figura 1: Proceso de Desarrollo de *Software* de SCRUM.

En SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, SCRUM está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales (27).

SCRUM permite a las organizaciones eliminar los impedimentos en el desarrollo de los proyectos, aumentando la satisfacción de los clientes mediante las entregas de resultados tangibles e integrándolos activamente en el ciclo de desarrollo, potencia la formación de equipos de trabajos autosuficientes y multidisciplinares, proporciona a los miembros del equipo un entorno amigable y productivo para desarrollar sus habilidades al máximo (28). Además permite combinarse con otras metodologías y marcos de gestión de proyectos.

XP (*eXtreme Programming*)

XP es una metodología ligera de desarrollo de *software* que se basa en la simplicidad, la comunicación y

la realimentación o reutilización del código desarrollado (29).

Esta metodología tiene tres fases las cuales son: Definición, Desarrollo y Mantenimiento. En cada fase se genera una serie de artefactos, que permiten producir rápidamente versiones del sistema que sean operativas, para que cada entrega sea lo más corta posible y reduzca el riesgo de errores. En la figura 2 se muestra el modelo de XP.

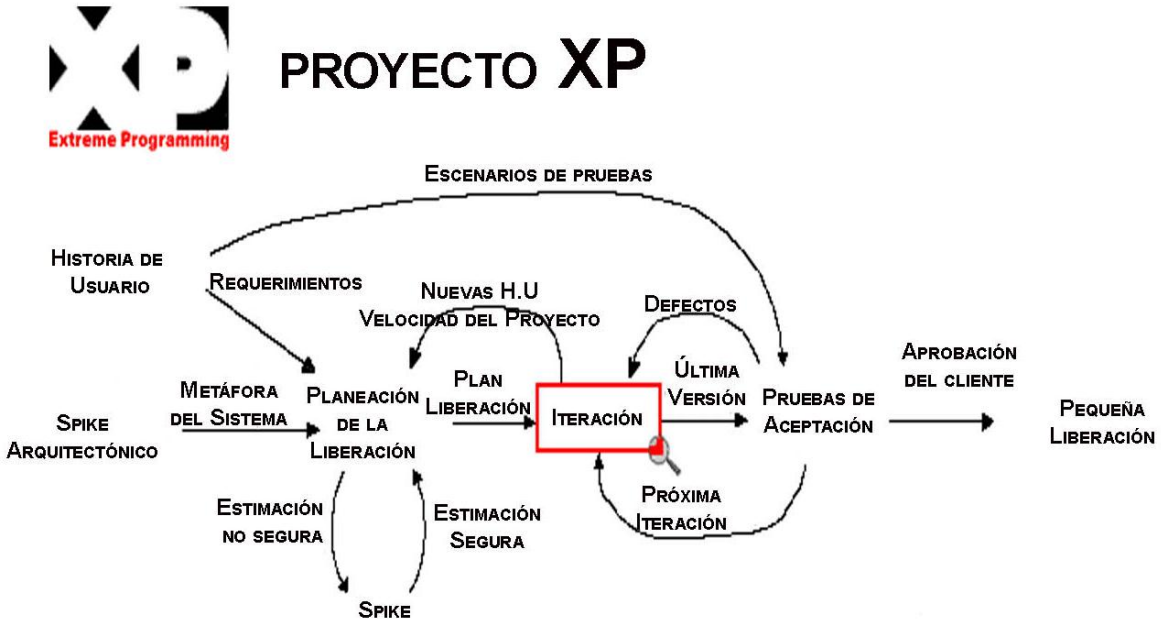


Figura 2: Modelo XP (Programación Extrema).

Las características más notales de la metodología XP son (29)**:**

1. Desarrollo iterativo e incremental.
2. Establece mejoras en cada una de las iteraciones del proyecto de forma incremental.
3. Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
4. Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
5. Frecuente interacción del equipo de programación con el cliente o usuario.
6. Corrección de todos los errores antes de añadir una nueva funcionalidad y realización de entregas frecuentes.

7. Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.
8. Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.
9. Simplicidad en el código.

SXP (SCRUM y eXtreme Programming)

SXP es un híbrido cubano de metodologías ágiles que tiene como base las metodologías SCRUM y XP, las que permiten actualizar los procesos de desarrollo de *software* para el mejoramiento de su producción. Esta metodología ayuda a fortalecer el trabajo en equipo, enfocados en una misma dirección, permitiendo además seguir de forma clara el avance de las tareas a realizar, a partir de la inserción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la producción, aumentando el nivel de interés del equipo (30). En la figura 3 se describe el Esquema de la metodología SXP.

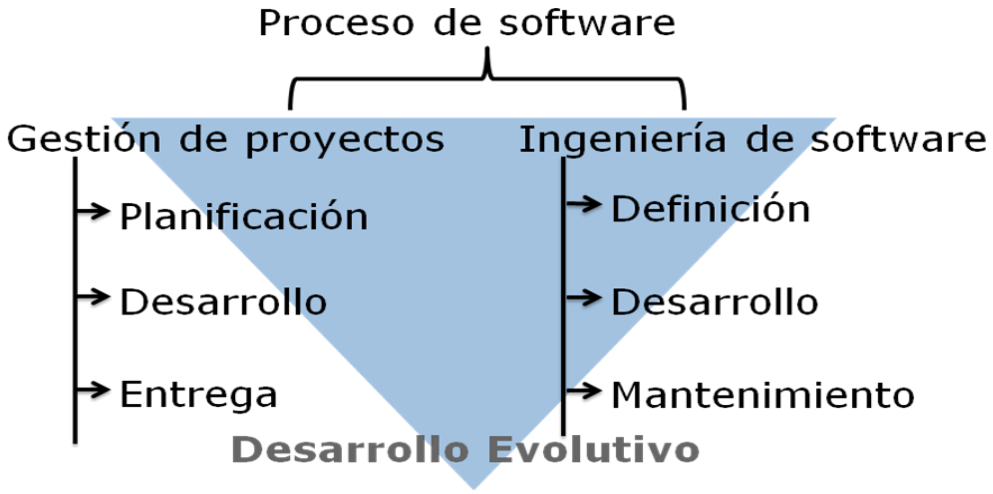


Figura 3: Esquema de la metodología SXP.

Esta metodología se divide en cuatro fases: Planificación-Definición, Desarrollo, Entrega y Mantenimiento, cada una de estas fases está compuesta por una serie de actividades que son las que generan los artefactos que quedan incluidos en el nuevo expediente de proyecto. Para la creación del módulo se utilizaran algunas de las actividades y artefactos que genera SXP.

Los flujos de trabajo y actividades que se generan por cada una de las fases son (31)**:**

Planificación ↔ Definición

- Entrevista con el cliente.
- Captura de requisitos:
  - Creación de la LRP.

    Plantilla LRP (Lista de Reserva del Producto).
  - Priorización de la LRP.
  - Definir las historias de usuario.

    Plantilla Historia de usuario.
  - Asignar las historias de usuario.

    Plantilla Historia de usuario.
- Valoración del esfuerzo.

  Plantilla Historia de usuario.
- Reunión de revisión del diseño.

Desarrollo:

- Junta de planificación.

  Plantilla de Glosario de términos.
  - Definir las historias de usuario a implementar.
  - Definir las tareas para lograr la implementación.

    Plantilla de Tareas de Ingeniería.

    Plantilla Cronograma de producción.

    Plantilla de Plan de Releases.
- Implementación.
  - Estándar de código.

    Estilo de código.
- Pruebas

  Plan de Pruebas

  Plantilla Caso de Prueba de aceptación.

Entrega

- Entrega de la documentación.

Mantenimiento

- Soporte.

  Plantilla de Gestión de cambios.

Esta metodología funciona bajo los principios del manifiesto ágil, lo cual posibilita una mayor respuesta ante los cambios que pueden ocurrir a lo largo del ciclo de vida del sistema. Además facilita la planificación y organización del proyecto centrada en los principales aspectos del sistema.

Selección de la metodología a utilizar

Los resultados de la utilización de la metodología SXP en aplicaciones que se encuentran explotación han sido satisfactorios. Se considera que SXP es una metodología viable para el desarrollo de *software*, pues permite minimizar el tiempo de desarrollo y se centra más en la implementación, que en la documentación. Además, permite mejorar el trabajo en equipo y darle seguimiento a las tareas a partir de los procedimientos ágiles. También se establece una estrecha relación entre las personas implicadas en el desarrollo del producto, eliminando errores por mala definición de los requisitos, por lo que en las primeras iteraciones se pueden obtener resultados precisos SXP toma las mejores prácticas de gestión de proyectos de la metodología SCRUM, mientras que tiene en cuenta las mejores prácticas de la metodología XP para realizar el proceso de desarrollo del proyecto, con el objetivo de que el proceso sea efectivo y eficiente. Por todo lo anteriormente expuesto se escoge como metodología de desarrollo SXP.

1.6.2. Sistema Gestor de Contenidos (CMS)

Un CMS (*Content Management System*, en español: Sistema Gestor de Contenidos) es un sistema que permite gestionar contenidos, posibilitando administrar contenidos en un medio digital. Es una herramienta que permite a un editor crear, clasificar y publicar cualquier tipo de información en una página *web*. Generalmente los CMS trabajan con una base de datos, de modo que el editor simplemente actualiza una base de datos. Un CMS puede adaptarse a las preferencias o necesidades de cada usuario. También puede proporcionar compatibilidad con los diferentes navegadores disponibles en distintas plataformas, ejemplo de ello son Windows, Linux, Mac y Palm.

A partir de solicitudes realizadas por la asesora de calidad del centro FORTES, se decidió realizar la implementación de varios módulos que permitieran gestionar la información relacionada con los servicios que brinda el Grupo de Calidad del centro, con el objetivo de asegurar el éxito de las soluciones que se

desarrollan en el mismo. Para eliminar las limitaciones que se presentaban en el manejo de la información relacionada con los procesos de pruebas se realizó la implementación del Módulo de Prueba durante el curso 2010-2011. Este constituyó el primer módulo del Sistema de Gestión de la Calidad, requerido por la asesora de calidad, al cual se propuso integrar otros módulos que permitieran gestionar la información de los distintos servicios que el grupo ofrece. El equipo de desarrollo del Módulo de Prueba propuso utilizar Drupal como CMS, en su versión 6.9, para la implementación de la solución, propuesta que fue aprobada por la asesora de calidad del centro. Tomando en cuenta lo anteriormente planteado se impone la utilización de Drupal, elemento necesario para la integración al módulo desarrollado con anterioridad.

Drupal es un CMS que se utiliza para crear sitios *web* dinámicos y con gran variedad de funcionalidades. Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios y desarrolladores que colaboran conjuntamente en su mejora y ampliación. Se trata de un sistema modular con una arquitectura muy consistente, que permite que los módulos creados por cualquier desarrollador puedan interactuar con el núcleo del sistema y con los creados por otros miembros de la comunidad. Con Drupal es posible implementar una gran variedad de sitios *web*. Permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Se destaca por la calidad de su código, el respeto de los estándares de la *web*, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

Posee algunas características como las que se relacionan a continuación (32):

- Ayuda on-line, para ello, posee un robusto sistema de ayuda online y páginas de ayuda para los módulos del “núcleo”, tanto para usuarios como para administradores.
- Es de código abierto, está libremente disponible bajo los términos de la licencia GNU/GPL. Además, es posible adaptarlo según las necesidades.
- Posee disímiles módulos, la comunidad de Drupal ha contribuido con muchos módulos que proporcionan diversas funcionalidades. Permite las URLs amigables, usa el *mod_rewrite* de Apache para crear URLs que son manejables por los usuarios y los motores de búsqueda.
- Es multiplataforma; puede funcionar con Apache o Microsoft IIS como servidor *web* y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Al estar implementado en PHP, es totalmente portable.
- Además, soporta múltiples idiomas.

La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún *software* adicional. Toda la actividad y los sucesos del sistema son capturados en un "registro de eventos", que puede ser visualizado por un administrador. Posee un sistema de cache, el cual elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

Drupal presenta una gran flexibilidad de diseño y una gran escalabilidad, a la vez que posee amplia documentación. Presenta un buen sistema de taxonomía, tiene un alto rendimiento. Las características de este CMS permiten reafirmar que constituye una solución idónea para la implementación de la solución propuesta.

1.6.3. Sistema Gestor de Base de Datos (SGBD)

Un SGBD (en inglés: *Database Management System*) es una colección de programas cuyo objetivo es servir de interfaz entre las bases de datos, los usuarios y las aplicaciones. Está compuesto de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad e integridad de los mismos. Un SGBD es una aplicación que permite a los usuarios realizar los procesos de gestión de una BD (33).

Los SGBD permiten (33):

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

A partir de los argumentos dados para la selección de CMS, fue necesario utilizar PostgreSQL, en su versión 8.4, como SGBD, el cual fue seleccionado por el equipo de desarrollo del Módulo de Prueba para implementar la solución que se proporcionó.

PostgreSQL es un potente motor de bases de datos objeto-relacional de código abierto, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Distribuido bajo la licencia *Berkeley Software* Distribución (BSD), que permite su libre uso, modificación y redistribución con la única condición de mantener el derecho de autor del *software* original a sus autores, por lo que los costos en este sentido son nulos. Posee una amplia comunidad de desarrolladores y

especialistas, de los que se puede obtener beneficios y contribuir. PostgreSQL ha sido diseñado y creado en función de que requiera un mantenimiento y ajuste mucho menor que otros a productos, conservando todas las características, estabilidad y rendimiento (34).

Es multiplataforma, por lo que facilita su amplio uso en soluciones informáticas. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). También soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeo. Cuenta además con una amplia documentación.

Cuenta con un rico conjunto de tipos de datos y provee un *framework* que permite a los desarrolladores definir y crear tipos de datos personalizados, conjuntamente con las funciones que estos soportarán y los operadores que definirán su comportamiento (34).

Este SGBD además tiene una implementación multiproceso, posibilitando una mejor disponibilidad de la información ante cualquier fallo de algún proceso, evitando que el servidor deje de brindar servicios. Presenta una alta concurrencia, permitiendo que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin que ocurra un bloqueo. Provee soporte para varios tipos nativos, como: números de precisión arbitraria, texto de largo ilimitado, direcciones IP (IPv4 e IPv6), direcciones MAC y arreglos.

1.6.4. Servidor *web*

Un servidor *web* es un programa que se ejecuta de forma continua en una computadora, se encarga de atender las demandas de información y responder a peticiones de los clientes o navegantes, entregando como resultado una página *web* (código HTML[2]) o información de todo tipo de acuerdo a los comandos solicitados, implementando el protocolo HTTP (35).

Apache HTTP Server

Apache HTTP Server es un servidor *web* desarrollado por la Fundación de *Software* Apache (Apache *Software Foundation*). Es “un esfuerzo colaborativo de desarrollo de *software* que apunta a crear una implementación del código fuente de un servidor (*web*) HTTP que sea robusto, comercial, de libre disponibilidad y con muchas funcionalidades”. El servidor HTTP Apache ofrece un mejor uso de las especificaciones de HTTP existentes, es libre y de distribución gratuita. Además ofrece instalaciones

[2] **HTML** es la sigla de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto, en español).

sencillas para sitios pequeños (36).

Apache posee características que han permitido su inserción y utilización en ámbitos empresariales, tecnológicos y educativos de forma exitosa:

- Se ejecuta sobre una multitud de plataformas.
- Es de tecnología libre y código abierto.
- Es un servidor *web* configurable y de diseño modular, capaz de extender su funcionalidad y la calidad de sus servicios.
- Trabaja en conjunto con gran cantidad de lenguajes de programación interpretados como PHP, Perl, Java, JSP (*Java Server Pages*) y otros lenguajes de *script*, que son el complemento ideal para los sitios *web* dinámicos.
- Es posible encontrar gran cantidad de documentos, ejemplos y ayuda en internet en todos los idiomas, lo que permite a los usuarios utilizar el servidor adecuadamente y evacuar cualquier duda.

Tener un servidor usando Apache es una solución eficaz y rápida para tener los sitios *web* funcionando eficientemente de forma gratuita. Otra ventaja de este servidor es que resulta sencillo conseguir ayuda o soporte.

Internet Information Services (IIS)

Internet Information Services (IIS) es un servidor *web* y un conjunto de servicios para el sistema operativo Microsoft Windows. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Es propietario y su uso está restringido por una licencia (37).

Selección del servidor *web*

Para el desarrollo de la aplicación será utilizado Apache HTTP Server como servidor *web*, porque su uso no está restringido por una licencia, es de código abierto, es configurable, multiplataforma y fácil de usar. Posee abundante documentación, ejemplos y ayuda en internet en distintos idiomas, permitiendo a cualquier usuario implementar módulos para una función específica. Para la instalación de Drupal es necesario tener instalado un servidor *web*, puede ser usado IIS, pero se recomienda la utilización de Apache para obtener un mejor rendimiento y soporte de la comunidad.

1.6.5. Lenguajes de desarrollo

Un lenguaje de desarrollo es un mecanismo mediante el cual es posible crear programas a través de un conjunto de instrucciones, operadores y reglas de sintaxis. Este permite establecer la comunicación entre el programador y los dispositivos (de hardware y *software*) existentes. Los lenguajes de desarrollo se pueden clasificar en dos grupos, del lado del servidor y del lado del cliente, pues permiten llevar a cabo el desarrollo de las aplicaciones.

Lenguajes del lado del servidor

Existen varios lenguajes de lado del servidor tales como: Java, Perl, PHP, entre otros. La investigación se enmarcará en este último, tomando en cuenta que el CMS, definido para el desarrollo del módulo, necesita para su funcionamiento tener instalado PHP, debido a que está escrito en este lenguaje.

PHP 5.2

PHP (acrónimo de *Hipertext Preprocesor*) es un lenguaje multiplataforma, orientado al desarrollo de aplicaciones *web* dinámicas que, puedan necesitar o no, acceso a información almacenada en una base de datos. El código fuente escrito en PHP es invisible al navegador *web* y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado al navegador. Esto hace que la programación en PHP sea segura y confiable. Presenta capacidad de conexión con la mayoría de los Sistemas Gestores de Base de Datos que se utilizan en la actualidad. Destaca su conectividad con MySQL y PostgreSQL. Posee una amplia documentación, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Permite aplicar técnicas de programación orientada a objetos (38).

PHP es un lenguaje que no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes. Permite la conexión a diferentes tipos de Sistemas Gestores de Base de Datos, tales como *MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite* (38).

Lenguaje del lado del cliente

Para la implementación de aplicaciones *web* existen distintos lenguajes de desarrollo del lado del cliente. En el presente módulo se decidió hacer uso de XHTML, CSS y Javascript, los que permiten construir un sitio *web* en Drupal con una mayor facilidad y mejor estética.

XHTML

XHTML (Lenguaje de Marcado de Hipertexto Extensible) es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos (39).

XHTML surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella. Al estar orientado al uso de un etiquetado correcto, exige una serie de requisitos básicos a cumplir en lo que a código se refiere. Entre estos requisitos básicos se puede mencionar una estructuración coherente dentro del documento donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, etc. (39).

CSS

Las Hojas de Estilo en Cascada (*Cascading Style Sheets* o CSS, por sus siglas en inglés) es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser mostrada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

El CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores *web* controlar el estilo y el formato de múltiples páginas *web* al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. (40).

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne. El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto (40).

Las tres formas más conocidas de dar estilo a un documento son las siguientes (40):

- Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento <link>, el cual debe ir situado en la sección <head>.
- Utilizando el elemento <style>, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección <head>. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.
- Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo <style> dentro de <body>. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

Javascript

Javascript es un lenguaje de programación interpretado, es decir, que no requiere compilación, con una sintaxis semejante a la de los lenguajes Java y C. Es utilizado para crear pequeños programas que luego son insertados en una página *web* y en programas, orientados a objetos, que posean una mayor complejidad. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador *web* permitiendo mejoras en la interfaz de usuario y páginas *web* dinámicas.

Este lenguaje posee varias características, entre ellas se puede mencionar que es un lenguaje basado en acciones. Además, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del *mouse*, aperturas, utilización de teclas y cargas de páginas. Es soportado por la mayoría de los navegadores, destacándose Internet Explorer, Netscape, Opera y Mozilla Firefox, al estar entre los más conocidos, brindando la posibilidad al usuario de elegir la opción de Activar/Desactivar el Javascript en los mismos.

1.6.6. Entorno Integrado de Desarrollo (IDE)

Un IDE, es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. También puede ser una aplicación por sí sola o puede ser parte de aplicaciones existentes. Ofrece un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi y Visual Basic. Además en algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C (41).

Los IDE presentan características como: son multiplataforma; poseen soporte para diversos lenguajes de programación, reconocimiento de sintaxis, extensiones y componentes para el IDE; permiten la integración con *framework* populares, así como importar y exportar proyectos. Además poseen un manual de usuarios y ayuda.

Un IDE suele tener los siguientes componentes: editor de texto, compilador, intérprete, herramientas de automatización, depurador, posibilidad de ofrecer un sistema de control de versiones y factibilidad para ayudar en la construcción de interfaces gráficas de usuarios. Existen diversos IDEs de programación, ejemplo de ellos son: Eclipse, NetBeans, Zend Studio y CodeRun (41).

NetBeans

NetBeans es un entorno integrado de desarrollo modular, escrito en el lenguaje de programación Java. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (*framework*) para compilar cualquier tipo de aplicación. Ofrece un excelente entorno para programar en PHP y tiene un excelente balance entre una interfaz con múltiples opciones. NetBeans es multilenguaje, utilizado tanto por programadores con poca experiencia como por expertos. Fue creado por la propia compañía de Java, Sun MicroSystems. Presenta características como: permite automatizar la compilación de proyectos complejos y realizarla, ofrece la posibilidad de refactorizar código, ordenarlo, modifica nombres de variables, incorpora el completamiento de código, permite trabajar sobre el mismo código a más de un programador. Además, cuenta con excelente soporte para aplicaciones *web*.

Eclipse

Eclipse es otro tipo de IDE de código abierto y multiplataforma. La compilación que realiza Eclipse es en tiempo real. Este IDE posee una gran comunidad de desarrollo y una amplia documentación publicada en su sitio oficial. Ofrece el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Su misión consiste en evitar tareas repetitivas, facilitar la escritura correcta del código y disminuir el tiempo de depuración. Eclipse provee algunos *plugins* de soporte para aplicaciones *web*, pero no igualan a otros IDEs como Netbeans.

Zend Studio

Zend Studio es un programa de la casa Zend, orientado a desarrollar aplicaciones *web* en lenguaje PHP. Este programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más (42). Es un *software* propietario, por lo que su uso está restringido por una licencia comercial.

Selección del IDE

Luego de estudiar varios entornos integrados de desarrollo se decidió escoger Netbeans para el desarrollo del sistema, teniendo en cuenta varios aspectos: es de código abierto, ofrece un excelente entorno para programar en PHP y posee características que facilitan la implementación del sistema. Es multiplataforma y tiene un potente motor para lo conexión a bases de datos. Se determinó no utilizar Zend Studio porque es un s*oftware* propietario, no siendo así NetBeans, pues es libre y gratuito sin restricciones de uso. No será utilizado Eclipse, a pesar de las potencialidades que brinda, debido a que el equipo de desarrollo tiene mayor experiencia en el uso de Netbeans, lo cual permite reducir el tiempo de desarrollo, al evitar la contante búsqueda de información sobre un determinado funcionamiento del sistema. La versión de Netbeans que será utilizada será 7.1.

1.7. Conclusiones parciales

- Se determinó entre la gama de clasificaciones de auditorías, analizar las auditorías de *software,* debido a que el centro FORTES no posee una herramienta que sirva de apoyo al proceso de auditoría que se le realiza a los proyectos que pertenecen a dicho centro.

- Se determinó entre los tipos de revisiones, analizar las de revisiones de *software*, porque el centro FORTES tampoco tiene una herramienta que sirva de ayuda al proceso de revisión que se le aplica a los distintos proyectos pertenecientes al centro.
- Las aplicaciones existentes a nivel internacional son privativas y no satisfacen por completo, las necesidades de la problemática planteada; mientras que la existente en Cuba no se ajusta completamente a los requerimientos del cliente.
- Para el desarrollo del sistema se empleó la metodología de desarrollo de *software* SXP; como CMS, Drupal versión 6.9; como Sistema Gestor de Base de Datos, PostgreSQL en su versión 8.4. Se seleccionó como servidor *web,* Apache HTTP Server versión 2.2. Los lenguajes de programación a utilizar fueron: PHP 5.2, para el lado del servidor; y Javascript, XHTML y CSS del lado del cliente. Además, como IDE se determinó el uso de NetBeans en su versión 7.1.

CAPÍTULO 2: ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1. Introducción

La creación de sistemas informáticos es un proceso en el que se desarrollan artefactos que, luego de ser integrados, posibilitan responder a las necesidades del cliente. En este proceso se identifican varias etapas, que van desde la declaración del problema y los requerimientos del sistema, hasta las pruebas y la liberación del mismo.

En el presente capítulo se describe la solución propuesta del Módulo Auditoría y Revisión (MAR) del Sistema de Gestión de la Calidad del centro FORTES, con el objetivo de conseguir una comprensión más precisa de los requisitos y una descripción de los mismos, que sea fácil de mantener y que ayude a estructurar el sistema. Se describirá las fases de la metodología SXP, con sus artefactos generado como son: la Lista de Reserva del Producto, las Historias de Usuarios, la Plantilla Plan de *Releases*, las Plantillas Tareas de Ingeniería y Pruebas de Aceptación. El presente módulo solo llegará hasta la fase de Entrega, pues el Sistema de Gestión de la Calidad, pues aún no se encuentra en explotación por parte del cliente.

2.2. Fase 1: Planificación-Definición

En esta fase se generan todos los documentos que se encuentran relacionados con la concepción inicial del sistema, así como la definición del mismo. Se incluyen algunos que están vinculados a la primera parte de los procesos de Ingeniería de *Software*, tales como los relacionados con el negocio y los requisitos. Además se incluyen aquellos documentos que están relacionados con la estimación inicial de esfuerzos, y la valoración de los riesgos (43).

2.3. Breve descripción del sistema

El centro de desarrollo de *software* FORTES requiere informatizar los procesos de auditoría y revisión que se realizan actualmente de forma manual. En respuesta a estas necesidades se desarrolla el Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del centro FORTES. El sistema es una aplicación *web*, elemento que aporta la posibilidad de conexión remota por varios usuarios. Además, se integrará al Sistema de Gestión de la Calidad del centro FORTES.

El sistema permite gestionar la información de las auditorías y las revisiones que se realizan a los proyectos del centro FORTES. La aplicación brindará la opción de gestionar plan de auditoría y revisión, cronogramas de auditoría y revisión, entrevistas de auditoría y revisión, listas de chequeo de auditoría y revisión, no conformidades, expediente de auditoría y revisión. También se pueden generar reportes de auditoría y revisión y evaluar auditoría; además de permitir el acceso a las otras opciones que ofrecerá el Sistema de Gestión de la Calidad del centro FORTES relacionadas con las actividades que realiza.

La aplicación es fácil de usar, tiene una interfaz amigable, con colores suaves y pocos elementos visuales, lo que facilita que el usuario centre su atención en los elementos propios del trabajo con el módulo y no distraiga su atención.

2.4. Lista de Reserva del Producto

La gestión de proyecto en la metodología ágil SXP define artefactos para describir las características del producto, integrando al cliente como parte importante del equipo de desarrollo, debido a las grandes posibilidades de que existan cambios en los requerimientos. Es dentro de la gestión de proyecto que se genera el artefacto Lista de Reserva del Producto (LRP).

La LRP es una lista priorizada que define el trabajo a realizar en el proyecto. En ella se incluyen los requerimientos sobre el producto, por lo que puede modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. El completamiento de la lista (acompañado de los cambios en el entorno y el producto) asegura que el producto sea lo más correcto, útil y competitivo posible (43).

En la LRP se incluyen los requerimientos que demanda el cliente del sistema, separados por su prioridad (alta, media o baja), la que indica la necesidad que tiene el sistema de dicha funcionalidad, siendo las muy altas, funcionalidades indispensables para cubrir las necesidades del cliente; mientras que las bajas son aquellas que se desean incluir, pero sin las que el sistema podría, todavía, ser funcional y útil. Además, se enuncian los requerimientos no funcionales especificados por el cliente como necesarios.

La tabla de la LRP incluye los siguientes campos: **Prioridad**, que contiene los niveles de prioridad de las funcionalidades incluidas (divididas en 3 grupos: Alta, Media y Baja); ***Ítem***[3], que contiene la numeración secuencial de las funcionalidades que están contenidas en cada grupo de prioridad; y **Descripción**, que

[3] ***Ítem***: (para este caso) Numeración secuencial de las funcionalidades que están contenidas en cada grupo de prioridad .

contiene la descripción de la funcionalidad. Además, se incluyen los **Requerimientos No Funcionales** (que no llevan estimación) (43).

En el presente trabajo los niveles de prioridades de funcionalidades que se tendrán en cuenta serán divididas en 3 grupos: Alta, Media y Baja. En la LRP del Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del centro FORTES, las funcionalidades de prioridad alta serán las consideradas como arquitectónicamente significativas. Es válido aclarar que esta LRP puede sufrir cambios en el transcurso del proyecto, adicionando, modificando y eliminando funcionalidades. Se muestra la LRP en el Anexo 2.

2.5. Historias de Usuario

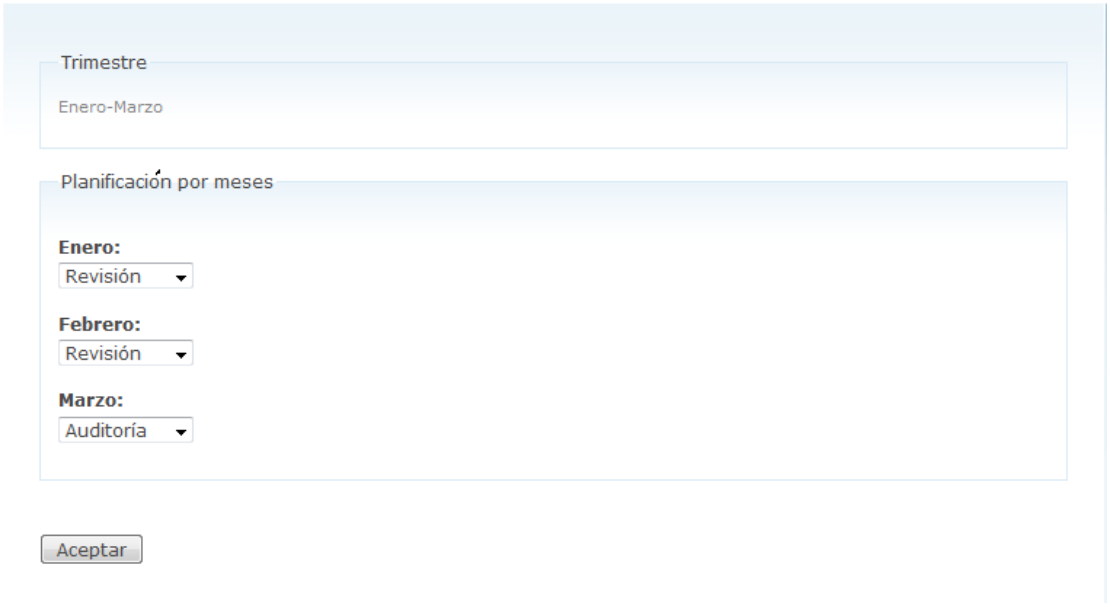
En la metodología SXP el único documento de requisitos generado es Historias de Usuarios (HU). En el Proceso Unificado de *Software* (RUP), los rectores del desarrollo de los artefactos en el análisis, diseño, implementación y prueba, son los casos de uso.

Las HU son escritas por los clientes, quienes especifican y describen los requisitos del *software* en forma de tareas que el sistema debe hacer, utilizando un lenguaje natural, sin formato predeterminado y en pocas líneas de texto. Guían la construcción de otros artefactos incluidos en la metodología (Tareas de Ingeniería, Plan de *Raleases* y Pruebas de Aceptación), siendo utilizadas para estimar tiempos de desarrollo (44).

En las tablas se incluyen los siguientes campos: **Número**, que contiene el identificador de la HU; **Nombre Historia de Usuario**, que contiene el nombre que identifica a la HU; **Usuario**, nombre del programador encargado de implementar la HU; **Iteración Asignada**, número de la iteración en la que se desarrollará la HU; **Prioridad en Negocio**, tipo de prioridad de la HU (Alta, Media o Baja); **Puntos Estimados**, valor que describe la cantidad de semanas estimadas para completar la HU; **Riesgo en Desarrollo**, al desarrollar la HU (Alto, Medio o Bajo); **Descripción**, breve descripción del proceso que define la HU; **Observaciones**, comentarios aclaratorios relacionados con la HU y **Prototipo de Interfaz**, que contiene la imagen de una de las interfaces de usuario relacionadas con la HU.

En la Tabla 1, se muestra la HU Gestionar Plan de Auditoría y Revisión, priorizada como Alta, es decir, arquitectónicamente significativa. El resto de las HU se incluyen en el Anexo 3.

Tabla 1: HU Gestionar Plan de Auditoría y Revisión

Historia de Usuario	
Número: HU01	**Nombre Historia de Usuario:** Gestionar Plan de Auditorías y Revisiones
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Sandy Guerra Fernández.	**Iteración Asignada:** Primera
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1 Semana
Descripción: La Historia de Usuario consiste en crear, modificar, ver y eliminar el plan de auditorías y revisiones.	
Observaciones:	
Prototipo de Interfaz: 	

2.6. Fase 2: Desarrollo

La fase de Desarrollo, es la segunda que define la metodología SXP. En esta fase se generan todos los documentos relacionados con la planificación de las iteraciones, y además se recogen las principales

definiciones que se manejan en la metodología y otros términos de difícil entendimiento para los clientes, así como de las tareas a realizar durante la implementación. Además se genera el código fuente en la etapa de implementación y los documentos relacionados con las pruebas (43).

Plan de *Releases*

La metodología SXP basa su funcionamiento en iteraciones, en las que se busca "transformar un subconjunto de la Reserva del producto en un incremento en la funcionalidad del producto que sea potencialmente entregable a los usuarios. Para ello se determina en qué funcionalidad del producto trabajará el equipo durante la próxima iteración" (51). Como resultado de determinar dichas funcionalidades, aparece la plantilla Plan de *Releases*, en la que "se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas" (47).

En la Tabla 2: Plan de *Releases*, se incluyen los siguientes campos: ***Release***, que contiene el identificador de la iteración que se va a desarrollar; **Descripción de la iteración**, breve descripción del objetivo de la iteración; **Orden de la HU a implementar**, contiene los identificadores de las HU a implementar en la iteración, en el mismo orden en que se deben realizar; y **Duración total**, cantidad de semanas que durará realizar la iteración, la que depende del tiempo estimado de las HU propuestas. A continuación la tabla del plan *realeses*.

Tabla 2: Plantilla Plan de *Releases*

Release	**Descripción de la iteración**	**Orden de la HU a implementar**	**Duración total**
[Número de la iteración que se va a desarrollar para la realización del producto]	*[Hacer una breve descripción del objetivo de la iteración]*	*[Número de las historias de usuario que se ven a implementar en cada una de las iteraciones por orden de prioridad]*	*[La duración total va a ser el tiempo estimado según las HU propuestas en que demorará su implementación]*

Tareas de Ingeniería

En la metodología SXP se genera el documento de Tareas de Ingeniería, donde se describe las tareas que componen las HU, las cuales marcarán el proceso para cumplir con los objetivos de cada HU. Para alcanzar los objetivos de una iteración es necesario completar cada una de ellas.

En la Tabla 3: Tarea de Ingeniería se incluyen los siguientes campos: **Número Tarea**, que contiene un número consecutivo en base a la historia de usuario correspondiente; **Número Historia de Usuario**, que contiene el identificador de la HU a la que pertenece esta tarea; **Nombre Tarea**, contiene un nombre que identifica a la tarea; **Tipo de Tarea**, contiene el tipo de tarea, que puede ser de desarrollo, estudio o la especificación de otra; **Puntos Estimados**, estimación en semanas de la duración de la tarea; **Fecha Inicio**, contiene la fecha de inicio de la tarea; **Fecha Fin**, contiene la fecha de finalización de la tarea; **Programador Responsable,** nombre del programador responsable de desarrollar la tarea; y **Descripción** que contiene la descripción de la tarea.

Tabla 3: Tarea de Ingeniería

Tarea de Ingeniería	
Número Tarea: *[Los números deben de ser consecutivos.]*	**Número Historia de Usuario:** *[Número de la historia de usuario a la que pertenece la tarea.]*
Nombre Tarea: *[Nombre que identifica a la tarea.]*	
Tipo de Tarea: *[Las tareas pueden ser de: Desarrollo, Corrección, Mejora, Otra (Especificar).]*	**Puntos Estimados:** *[Tiempo en semanas que se le asignará.(Estimado)]*
Fecha Inicio: *[Fecha de inicio de tarea.]*	**Fecha Fin:** *[Fecha de fin para cumplir la tarea.]*
Programador Responsable: *[Nombre y apellidos del programador.]*	
Descripción: *[Breve descripción de la tarea.]*	

Pruebas

Al realizar la planificación de la iteración es necesario definir también el sistema de pruebas que se aplicarán para verificar que el sistema cumpla con las funcionalidades que precisa el cliente. Con este objetivo se construye y entrega el artefacto de la metodología SXP: Pruebas de Aceptación, en las que "el desarrollador escribe las pruebas realizadas según la historia de usuario seleccionada para realizar la

comprobación y validar las funcionalidades del sistema, y de esta forma saber si está apto para ser liberado" (47).

En la Tabla 4: Caso de Prueba, se incluyen los siguientes campos: **Código Caso de Prueba**, que contiene el identificador de caso de prueba (en el caso de las presentes, se utiliza el identificador de la HU, al que se le adiciona 'P' y un número consecutivo); **Nombre Historia de Usuario**, que contiene el nombre de la HU correspondiente a este caso de prueba; **Nombre de la persona que realiza la prueba**, contiene el nombre del responsable que realiza la prueba; **Descripción de la Prueba**, que contiene una breve descripción de la prueba realizada; **Condiciones de Ejecución**, se incluyen las condiciones necesarias para que se pueda realizar la prueba; **Entrada/Pasos de ejecución**, contiene una serie de pasos enumerados para lograr realizar la prueba de esta HU; **Resultado Esperado**, contiene la descripción de lo que se espera luego de realizar la prueba (cumplimiento de las restricciones del producto); y **Evaluación de la Prueba**, muestra si la prueba fue satisfactoria o insatisfactoria.

Tabla 4: Caso de Prueba

Caso de Prueba	
Código Caso de Prueba: *[Número que identifica el caso de prueba]*	**Nombre Historia de Usuario:** *[Nombre de la HU a realizar prueba.]*
Nombre de la persona que realiza la prueba: *[Nombre y apellidos.]*	
Descripción de la Prueba: *[Descripción de la prueba realizada.]*	
Condiciones de Ejecución: *[Condiciones necesarias para poder realizar las pruebas]*	
Entrada / Pasos de ejecución: *[Serie de pasos necesarios para lograr la realización de la HU, y así realizar la prueba.]*	
Resultado Esperado: *[Que cumpla con las restricciones del producto.]*	
Evaluación de la Prueba: *[Satisfactoria o no satisfactoria.]*	

2.7. Fase 3: Entrega

En ella se realiza la entrega del producto y su documentación, generándose aquellos documentos que son imprescindibles para el entrenamiento y entendimiento del producto.

Manual de usuario

El manual de usuario es un documento, que sirve de apoyo para la actividad de capacitación. En este se recogen todos los aspectos significativos, que contribuyan a que los usuarios que interactúen con el sistema tengan una mejor comprensión del mismo. Para su confección hay que tener en cuenta a quién va dirigido, es decir el manual puede ser manejado por diferentes usuarios, desde los más simples hasta los más complejos. Por consiguiente, debe redactarse de forma clara y sencilla para que lo entienda cualquier tipo de usuario.

El manual expone los procesos que el usuario puede realizar con el sistema implantado. Para lograr esto, es necesario detallar las características que tiene el módulo y la forma de acceder e introducir información. Además reúne la información necesaria para que el usuario conozca y utilice adecuadamente la aplicación desarrollada. De manera general, el manual de usuario proporciona ventajas tales como: posibilita que el usuario conozca como interactuar con el sistema, se utiliza como manual de aprendizaje y sirve como manual de referencia.

2.8. Fase 4: Mantenimiento

En la fase de Mantenimiento se realizan las actividades relacionadas con el soporte del *software* y se generan los documentos relacionados con los cambios que puedan ocurrir en el mismo. Es válido reiterar que, aunque se menciona esta fase, el equipo de desarrollo no la tendrá en cuenta. Además, se definió durante la fase Planificación – Definición que el trabajo de los miembros del equipo concluye con la entrega del presente módulo.

2.9. Conclusiones parciales

- Se explicó la solución que se provee con el Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del centro FORTES.
- Se explicaron y presentaron los artefactos de la metodología ágil SXP utilizados: Lista de Reserva del Producto e Historias de Usuario, Plantilla Plan de *Releases* y Plantillas Tareas de Ingeniería.
- La correcta gestión de proyecto que propone SXP posibilitó un mayor control de las tareas.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

La concepción de la propuesta del sistema y los artefactos generados y presentados en el capítulo anterior, ayudan al programador a entender las funcionalidades que necesita el cliente y, por tanto, a ser capaz de implementar el sistema. Igual de importante para un sistema es el proceso de pruebas, que tiene como objetivo determinar si el producto obtenido se ajusta a los requisitos o necesidades del cliente.

La metodología SXP gestiona los procesos de implementación y prueba con pocos artefactos, de forma tal que la comunicación entre todos los miembros del equipo y el cliente sea la base para la obtención del producto que realmente desea este último. En el presente capítulo se incluyen los artefactos: Plan de *Releases* y Tareas de Ingeniería, que constituyen los rectores del proceso de implementación y las Pruebas de Aceptación, base del proceso de prueba en la metodología SXP. Además, el resultado de las pruebas, el estándar de código utilizado, la distribución física necesaria para el despliegue del módulo, la seguridad y el tratamiento de errores.

3.2. Tablas Aportadas a la Base de Datos

Durante la fase Planificación-Definición, aunque la metodología no propone las tablas aportadas a la BD, se decidió realizar un diseño de las nuevas tablas que debía contener la base de datos. Estas son utilizadas durante de la implementación, y posteriormente durante el funcionamiento del presente módulo. Este diseño permitió una mayor organización durante el desarrollo del sistema. Además, proporciona un mayor entendimiento a los miembros del grupo de trabajo e interesados. Para realizar el diseño fue necesario tener en cuenta que Drupal durante su instalación crea una serie de tablas en la base de datos, las cuales permiten almacenar y manejar los datos del sistema. En la figura 4 se muestran las tablas definidas el equipo de desarrollo.

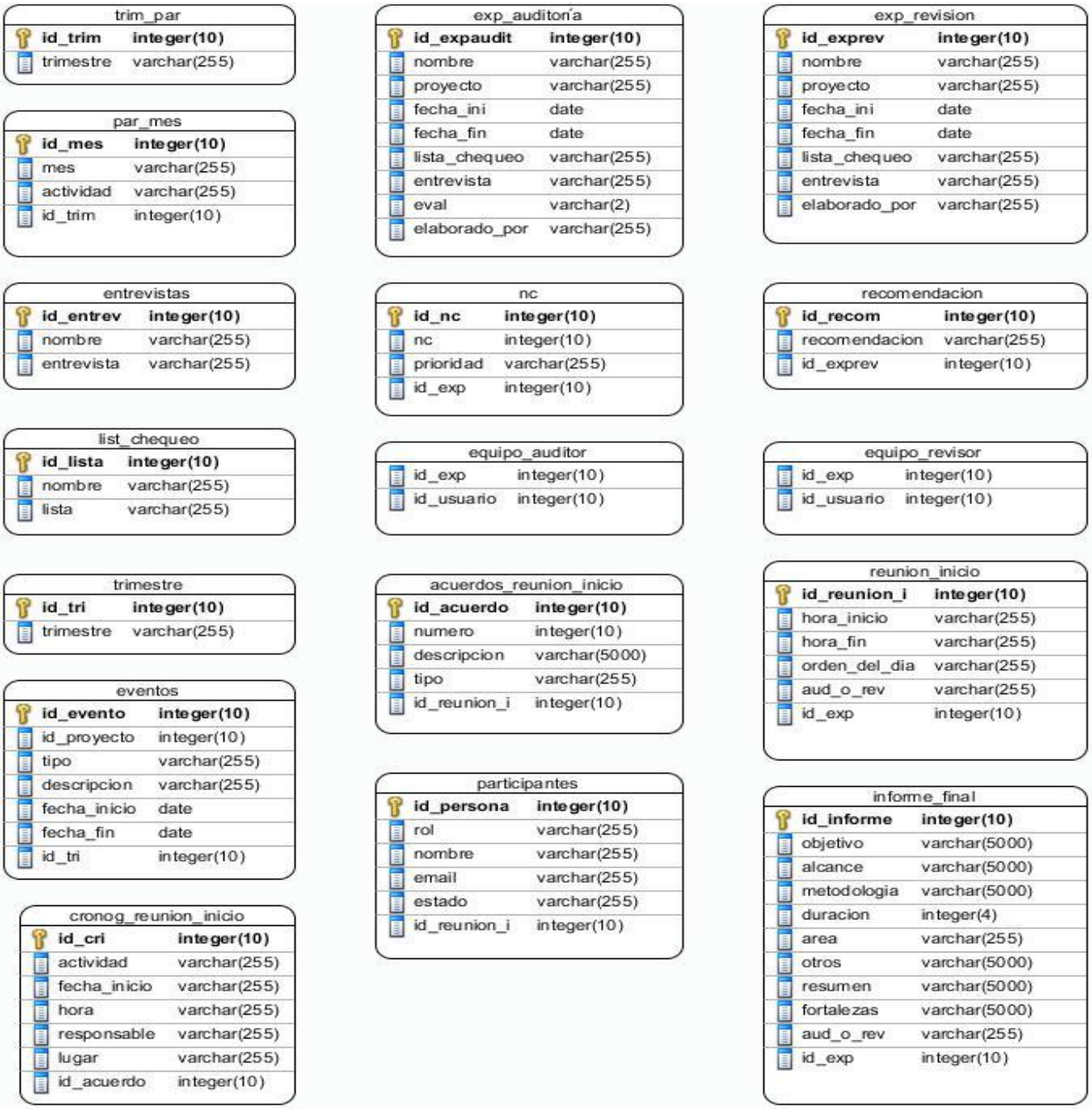


Figura 4: Representación gráfica de las tablas adicionales.

3.3. Funcionalidades significativas.

Durante concepción inicial del sistema, proceso que se realizó en reunión con el cliente, fueron identificadas las funcionalidades más significativas, la implementación de las cuales fue priorizada, algunas de las cuales se describen a continuación.

1. **Gestionar Plan de Auditorías y Revisiones**

La funcionalidad le permite al asesor de calidad realizar una planificación trimestral de las auditorías y revisiones a realizar por el Grupo de Calidad del centro FORTES sobre los proyectos que pertenecen a este centro productivo. Además permite conocer a los implicados, tanto miembros del grupo de calidad como líderes de proyectos cuando será realizado cada proceso. Permite gestionar todos los datos referentes a cada plan de auditoría y revisión, verificando que solo pueda existir una planificación para cada trimestre con el objetivo de brindar una confiablidad y un nivel de información elevado. A esta funcionalidad solo tendrá acceso el asesor de calidad del centro, el cual está calificado para la toma de decisiones de este tipo.

2. **Gestionar Lista de Chequeo de Auditoría y Revisión**

El cliente requiere manejar las plantillas de listas de chequeo que puedan ser utilizadas por los auditores y revisores del centro para la realización de su trabajo y la obtención de un resultado confiable. Estas plantillas permiten realizar una serie de recomendaciones a los proyectos que han sido objeto de evaluación, así como proporcionar una evaluación a cada auditoría. La creación, modificación y eliminación de estos artefactos de apoyo a los especialistas solo la puede realizar el asesor de calidad del centro. Esta estrategia permite evitar la pérdida de plantillas que puedan ser de utilidad para el arribo de conclusiones de forma rápida.

3. **Gestionar Expediente de Auditoría**

Durante cada el proceso de auditorías realizado se genera un cúmulo de información que debe ser resguardada en por los especialistas implicados. Esta información permite a los implicados arribar a conclusiones de forma certera, además permite tener un control del trabajo que se realiza por parte del

equipo auditor e implicados. Toda esta información se gestiona de forma manual, lo cual trae consigo que puedan aparecer riesgos como la perdida de información y la desorganización del proceso. Esta funcionalidad provee al Sistema de Gestión de la Calidad del centro FORTES de una forma de reducir los riesgos antes mencionados. Además permite a todos los implicados tener información actualizada sobre una determinada auditoría. El permiso de modificación de los datos que se generan está restringido, solo podrán realizarla los especialistas encargados de realizar esa auditoría.

4. **Generar Reporte de Auditoría y Revisión**

Esta funcionalidad, aunque no fue identificada como significativa durante la fase Planificación - Definición, se considera que debe ser descrita por el valor que aporta al módulo. Provee al asesor de calidad de un instrumento de control del trabajo de los distintos proyectos en materia de calidad, así como del grupo encargado de esta actividad en el centro productivo, además de facilitarle la toma de decisiones en el área. La generación de reportes permite conocer de forma rápida el comportamiento de un determinado equipo de proyecto en materia de calidad, a través del análisis de los resultados de la ejecución de un proceso de auditoría o revisión. A esta funcionalidad solo tendrá acceso el asesor de calidad, debido a que es quien requiere conocer los resultados de todos los procesos de calidad realizados o de alguno en específico, que sea de interés para la dirección de calidad.

3.4. Plan de *Releases*

Con el objetivo de obtener productos de calidad, la metodología SXP basa su funcionamiento en iteraciones, que le permiten organizar el trabajo del equipo de desarrollo, las cuales se recogen en la plantilla Plan de *Releases*. Para la realización de este artefacto fueron definidas tres iteraciones, con duración total de seis, tres y una semanas respectivamente. En cada iteración se desarrollan las Historias de Usuario, según el grado de prioridad definido por el cliente, resultando la lista como se muestra en la Tabla 5.

Tabla 5: Plan de *Releases*

Release	**Descripción de la iteración**	**Orden de la HU a implementar**	**Duración total**
Iteración 1	En esta iteración se desarrollan las funcionalidades priorizadas como altas.	HU01 HU03 HU04 HU05 HU06	6
Iteración 2	En esta iteración se desarrollan las funcionalidades priorizadas como medias.	HU02 HU07 HU09	3
Iteración 3	En esta iteración se desarrollan las funcionalidades priorizadas como baja.	HU08 HU10	1

3.5. Tareas de Ingeniería

Para obtener los resultados esperados, resulta de gran utilidad realizar una descripción de las tareas de cada Historia de Usuario a través de las Tareas de Ingeniería. A continuación se incluyen las Tareas de Ingeniería correspondientes a la HU Gestionar Plan de Auditoría y Revisión. El resto de las Tareas de Ingeniería se pueden encontrar en el Anexo 4.

En la Tabla 6 se describe la Tarea de Ingeniería “Estudiar el CMS Drupal”, la cual es de gran utilidad para la implementación haciendo uso de este. Esta tarea facilita la comprensión del funcionamiento de este CMS y las potencialidades que brinda. Esta tarea es significativa, por lo cual se le asignó prioridad Alta.

Tabla 6: Tarea de Ingeniería 1 de HU01

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HU01
Nombre Tarea: Estudiar el CMS Drupal.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 25/11/2011	**Fecha Fin:** 14/12/2011
Programador Responsable: Sandy Guerra Fernández Mirian del Carmen González Elias	
Descripción: Estudiar el funcionamiento del Drupal, en especial el trabajo con los módulos, los formularios y el modelo.	

En la Tabla 7 se describe la Tarea de Ingeniería "Estudiar el entorno de desarrollo integrado NetBeans", la cual es de gran utilidad para la implementación haciendo uso de este. Esta tarea facilita ampliar los conocimientos que tenían los miembros del equipo de desarrollo acerca de este IDE. Esta tarea es significativa, por lo cual se le asignó prioridad Alta.

Tabla 7: Tarea de Ingeniería 2 de HU01

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HU01
Nombre Tarea: Estudiar el entorno de desarrollo integrado NetBeans.	
Tipo de Tarea : Estudio	**Puntos Estimados:** 1
Fecha Inicio: 10/01/2012	**Fecha Fin:** 31/01/2012
Programador Responsable: Sandy Guerra Fernández	
Descripción: Estudiar el funcionamiento del NetBeans y su integración con Drupal.	

Por otra parte la Tabla 8 describe la Tarea de Ingeniería "Desarrollar CRUD del Plan de Auditoría y Revisión", la que permite satisfacer una necesidad del cliente. Se determinó que esta funcionalidad era

arquitectónicamente significativa para el sistema, por lo cual se le asignó prioridad Alta y se incluyó en la primera iteración.

Tabla 8: Tarea de Ingeniería 3 de HU01

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** HU01
Nombre Tarea: Desarrollar CRUD del Plan de Auditoría y Revisión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 01/02/2012	**Fecha Fin:** 07/02/2012
Programador Responsable: Sandy Guerra Fernández	
Descripción: Implementar las funcionalidades de crear, modificar, ver y eliminar los datos de un Plan de Auditoría y Revisión.	

3.6. Módulos y Librería de Apoyo

Para el desarrollo de una aplicación haciendo uso de un CMS resulta beneficiosa la utilización de diferentes módulos ya implementados, que facilitan el trabajo del equipo de desarrollo permitiendo reducir el tiempo de implementación, estos constituyen módulos de apoyo. Para la desarrollo de una solución a la problemática planteada se hizo necesario hacer uso de varios de estos módulos, los que fueron desarrollados con anterioridad por un grupo de desarrolladores que con su contribución permiten una constante actualización y mejoramiento de este CMS. Los módulos utilizados son:

- **DATE:** Este módulo se utiliza para facilitar la inserción de fechas con formatos predeterminados en la aplicación, permitiendo validarlas y reducir el número de errores cometidos por el usuario durante la inserción de datos.

- **LDAP:** Este módulo sirve de soporte al protocolo que lleva el mismo nombre, el cual se utiliza para garantizar la autenticación de los usuarios en el sistema, permitiendo obtener datos de forma confiable, que son necesarios para el correcto funcionamiento del módulo.

- **SMTP:** Este módulo permite dar soporte al protocolo que lleva el mismo nombre, el cual se utiliza para el envío de notificaciones por correo electrónico.

- **CKEDITOR:** Es un editor de texto HTML que permite realizar acciones en la *web* similares a las que realizan editores como Microsoft Word, sin la necesidad de instalar ningún componente en la computadora del cliente. Permite dar formato de fuente, cortar, copiar, pegar, inserción de imágenes y creación de tablas. En el presente modulo se utiliza para brindar la posibilidad al usuario que de formato a las cadenas de texto entradas en a través de textareas, como son las descripciones de los distintos eventos y los resultados de los procesos de auditorías y revisión de *software*. Se muestra un ejemplo de un campo "textarea", cuando se usa el módulo CKEditor.

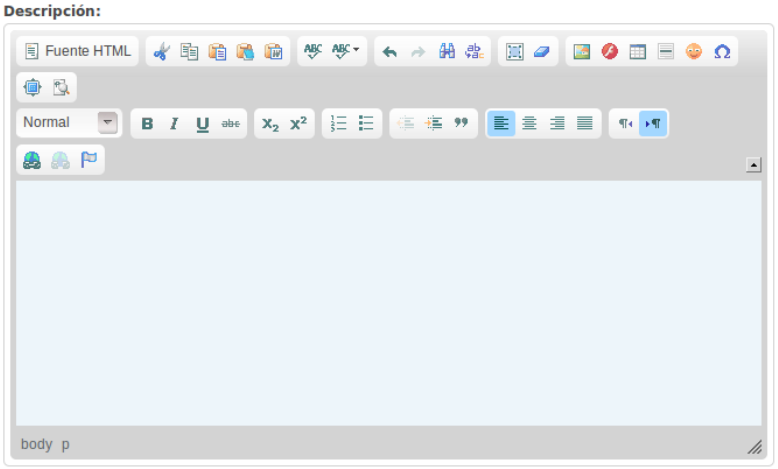


Figura 5: Ejemplo de textarea usando el módulo CKEditor

- **CCK (*Content Construction Kit,* en español: Kit de Construcción de Contenido):** Es utilizado para la creación de tipos de contenido de manera fácil a través de un panel de control. Inicialmente Drupal define como tipos de contenidos Artículo y Página, este módulo le permite al usuario crear un nuevo tipo de contenido, si así lo desea. Este módulo es utilizado por el módulo DATE, que lo requiere para su funcionamiento. El paquete de instalación trae incluido algunos submódulos como son:

- Generales:
  - ***Content:*** El núcleo.
  - ***Content Copy:*** Permite importar y exportar definiciones de campos.
  - ***Content Permissions:*** Permite establecer permisos a nivel de campo y no sólo de nodo.
  - ***Fieldgroup:*** Agrupar los campos para la presentación.
- Específicos de tipos de campo:
  - ***Node Reference:*** Campo para referenciar a otro nodo.
  - ***Number:*** Campos numéricos.
  - ***Option Widgets:*** Para definir campos de selección como *checkbox*, *select list* o *radio buttons*.
  - ***Text:*** Campos para texto genérico.
  - ***User Reference:*** Campo para referenciar a un usuario desde un nodo.

Para la implementación del presente módulo fue utilizada la librería **FPDF.** Esta es una clase desarrollada en PHP para poder realizar documentos en formato pdf. Permite modificar la unidad de medida, el formato de la página, los márgenes, las cabeceras y los pies de página, los saltos de línea, las imágenes, colores y enlaces. Además permite establecer el orden que de la información a mostrar, así como el formato que tendrá el documento. Es una librería gratuita para cualquier uso, tanto comercial como personal. Se hizo uso del mismo para la creación de reportes que tuvieran información referente a los procesos de auditoría y revisión de *software* realizados en el centro FORTES como son: los reportes por proyectos, tipos de procesos, responsable del proceso, rango de fechas, general y el informe final de una auditaría o revisión específica, que en sí mismo constituye un reporte de la actividad realizada por el equipo de trabajo responsabilizado con el proceso. Se muestra en la figura 6 un ejemplo de su utilización:

```php
<?php
require('/fpdf/fpdf.php');

$pdf=new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'¡Mi primera página pdf con FPDF!');
$pdf->Output();
?>
```

Figura 6: Estructura general para crear un documento con formato PDF usando la librería FPDF.

3.7. Roles de usuario

El Módulo Auditoría y Revisión permite el acceso a distintos usuarios, agrupándolos por roles, para garantizar la integridad de la información que se gestiona, a través de los niveles de acceso definidos para cada uno. Un rol no es más que una clasificación mediante la cual se definen distintos privilegios para los usuarios de un sistema. Drupal permite gestionar los usuarios y roles con sus permisos a través de la interfaz administrativa en la dirección administrar/administración_de_usuario, mediante la cual el asesor de calidad del centro podrá crear usuarios y roles en el sistema. Se definieron los siguientes roles para ser usados en este módulo específicamente:

Tabla 9: Roles de Usuario

ROL	NIVEL DE ACCESO
Administrador del sistema	✓ Acceso a todos los datos y opciones del sistema. ✓ Permisos para administrar en el sistema usuarios y roles con sus niveles de acceso. ✓ Permisos para cambiar la configuración del sitio.
Asesor de Calidad	✓ Acceso a todos los datos y opciones del módulo. ✓ Permisos para administrar en el sistema usuarios y roles con sus niveles de acceso.
Auditor Líder	✓ Permisos para gestionar un expediente de auditoría.

	✓ Permisos para actualizar un expediente de auditoría. ✓ Permisos para ver los planes de auditoría y revisión trimestrales. ✓ Permisos para ver cronogramas de auditoría y revisión trimestrales. ✓ Permisos para ver las plantillas de entrevistas almacenadas. ✓ Permisos para ver las plantillas de listas de chequeo almacenadas.
Auditor	✓ Permisos para actualizar un expediente de auditoría. ✓ Permisos para ver los planes de auditoría y revisión trimestrales. ✓ Permisos para ver cronogramas de auditoría y revisión trimestrales. ✓ Permisos para ver las plantillas de entrevistas almacenadas. ✓ Permisos para ver las plantillas de listas de chequeo almacenadas.
Revisor Líder	✓ Permisos para gestionar un expediente de revisión. ✓ Permisos para actualizar un expediente de revisión. ✓ Permisos para ver los planes de auditoría y revisión trimestrales. ✓ Permisos para ver cronogramas de auditoría y revisión trimestrales. ✓ Permisos para ver las plantillas de entrevistas almacenadas. ✓ Permisos para ver las plantillas de listas de chequeo almacenadas.
Revisor	✓ Permisos para actualizar un expediente de revisión. ✓ Permisos para ver los planes de auditoría y revisión trimestrales. ✓ Permisos para ver cronogramas de auditoría y revisión trimestrales. ✓ Permisos para ver las plantillas de entrevistas almacenadas.

	✓ Permisos para ver las plantillas de listas de chequeo almacenadas.
Jefe de proyecto	✓ Permisos para ver los planes de auditoría y revisión trimestrales. ✓ Permisos para ver cronogramas de auditoría y revisión trimestrales. ✓ Permisos para ver las plantillas de entrevistas almacenadas. ✓ Permisos para ver las expedientes de auditorías. ✓ Permisos para ver las expedientes de revisión.

3.8. Estándar de código

Para la implementación de un sistema informático, es de suma utilidad que el código programado sea legible y brinde la posibilidad de adicionarle nuevas sentencias, con el objetivo de ser entendido con facilidad y mejorado (de ser necesario). Para poder conseguir un código con estas dos características es necesario establecer un conjunto de reglas a la hora de escribir el código del programa, para ello la metodología SXP propone la utilización de estándares de programación.

Un estándar de programación, también conocido como estándar de programación o estilo de código, define la escritura y organización del código fuente de un programa, de forma tal que al trabajar en un proyecto, cualquiera de las personas involucradas en el mismo tenga acceso y comprenda el código. Seguir un estándar de programación le facilita al programador la modificación de su propio código fuente aunque no esté trabajando en un equipo.
Para la implementación del presente sistema, se decidió utilizar el siguiente estándar de código:

1. Los bloques de código deben estar siempre confinados por llaves. Las llaves siempre estarán al mismo nivel de la sentencia de código de la que proceden. Ejemplo:

```
function eliminar_entrevistaform()
{
        if(isset($_GET['id']))
        {
          $id = $_GET['id'];

          $form['id_e']=array('#type'=>'value','#value'=>$id);
               return confirm_form($form,
               t('¿Está seguro de que quiere eliminar la entrevista?'),
               'audit/entrev/ver_entrevistas',
               t('Esta acción no se puede deshacer.'),
               t('Eliminar'), t('Cancelar'));
        }
}
```

2. Los nombres de las variables y funciones comenzarán siempre con letra minúscula. Ejemplo: $nc, reporte_general().
3. Las declaraciones de clases abren llaves en la línea inferior de la declaración, y cierran llaves con el mismo nivel de margen de la declaración, una línea después de la última sentencia. Ejemplo:

```
class PDF extends FPDF
{
   ...
}
```

4. Antes de cada bloque de funciones de un mismo tipo, se plasma el nombre de la función como un comentario, para facilitar si localización. Ejemplo:

```
/*****************************Eliminar Entrevistas*****************************/
function eliminar_entrevistaform()
{
        if(isset($_GET['id']))
        {
          $id = $_GET['id'];

          $form['id_e']=array('#type'=>'value','#value'=>$id);
               return confirm_form($form,
               t('¿Está seguro de que quiere eliminar la entrevista?'),
               'audit/entrev/ver_entrevistas',
               t('Esta acción no se puede deshacer.'),
               t('Eliminar'), t('Cancelar'));
        }
}
```

5. La terminología usada para escribir código CSS es:

```
selector
{
    propiedad: valor;
}
```

6. En el código CSS los selectores deben estar seguidos con una llave abierta en la próxima línea. Luego de las propiedades debe terminar con una llave cerrada con la misma sangría de la llave abierta. Ejemplo:

```
.derecha p
{
    margin-left: 20px;
    text-align: justify;
}
```

7. En el código CSS, cada propiedad debe estar en su propia línea, identada con dos espacios y debe terminar con un punto y coma. Ejemplo:

```
.derecha p
{
    margin-left: 20px;
    text-align: justify;
}
```

El estándar de código anteriormente descrito es el utilizado al adicionar nuevas líneas de código a los ficheros generados por el equipo de desarrollo durante la implementación; el resto de las clases, funciones u otros elementos que componen los distintos módulos que requiere el CMS Drupal o los creados por diferentes programadores, pueden diferir de dicho estándar.

3.9. Seguridad

La seguridad es una característica de cualquier sistema (informático o no) que nos indica que ese sistema está libre de todo peligro, daño o riesgo, y que es, en cierta manera, infalible. A grandes rasgos, se entiende que mantener un sistema seguro (o fiable) consiste, básicamente, en garantizar tres aspectos: confidencialidad, integridad y disponibilidad(45).

Garantizando la confidencialidad los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ello. La integridad permite que los objetos sólo puedan ser modificados por

elementos autorizados, y de una manera controlada. Al garantizar la disponibilidad los objetos del sistema tienen que permanecer accesibles a elementos autorizados.

Para garantizar un funcionamiento adecuado y seguro se establecen como políticas de seguridad: el acceso a las distintas funcionalidades que posee el módulo de los usuarios previamente autorizados por el administrador del sistema (el asesor de calidad, para el caso de este módulo), así como la administración de la base de datos por parte del administrador del sistema. Solo tendrán acceso a la base de datos algunos usuarios con privilegios mínimos, los suficientes para realizar solamente las operaciones básicas de acceso a datos.

Drupal posee un potente sistema de seguridad basado en roles, lo cual permite controlar el acceso a los diferentes módulos según los permisos que el administrador le haya definido a cada rol. En este caso, el administrador no tiene que establecer los permisos para cada usuario, sino se asignan los permisos a un determinado rol y se agrupan los usuarios por roles. Esta asignación por roles permite controlar a qué funcionalidades el usuario tiene acceso, de manera que la información accedida esté limitada y protegida en dependencia de los distintos niveles de usuarios, obteniendo como resultado un sistema más seguro.

En el módulo se decidió establecer, adicionalmente, como política de seguridad el establecimiento de permisos de modificación de los expedientes de auditoría y revisión a los miembros del equipo responsable de cada proceso. Además, solo podrá eliminar un expediente el jefe del equipo que ejecuta el proceso lo genera.

3.10. Distribución física de la solución para su despliegue

Aunque la metodología seleccionada no propone la generación del artefacto “Modelo de Despliegue”, el equipo de desarrollo considera necesario ilustrar la forma en que quedara distribuido el módulo para su despliegue y realizar algunas aclaraciones pertinentes sobre la misma, con el objetivo de mejorar su entendimiento. En la figura 7 se muestra la disposición física que tendrá el módulo una vez que se ponga en explotación.

Figura 7: Distribución física de la solución para su despliegue.

Descripción de la capacidad que el dispositivo provee al sistema

El sistema utiliza una impresora como dispositivo externo para su completo funcionamiento, que se conecta a la PC Cliente. Este aporta la funcionalidad de impresión de los informes asociados al proceso de auditoría y revisión que se realicen y el reporte de las mismas. No se necesita de una impresora en específico, solo requiere estar activa y lista para imprimir cuando el usuario esté trabajando con el software.

Descripción de la funcionalidad y capacidad del nodo

Estación de trabajo: ordenador cliente que se conecta a través de un navegador *web* al servidor central donde reside la aplicación. No se necesita tener instalado el producto localmente.

Existirá un servidor central, el cual tendrá instalado un servidor *web* (Apache 2.2), uno de base de datos (PostgreSQL 8.4), el intérprete de PHP 5.2 y hospeda todos los componentes necesarios para el funcionamiento del producto. En la imagen anterior se muestran ambos servidores conectados a través del protocolo TCP/IP. La memoria RAM debe ser de al menos 512 Megabyte (MB).

Descripción de elementos e interfaces de comunicación

<<HTTP>>: representa la conexión que se va a establecer entre una Estación de trabajo que se va a conectar con el servidor central donde reside el servidor *web*, el de base de datos y la aplicación; en otras palabras, significa la conexión entre el navegador de usuario y el servidor del sistema.

<<USB[4]>>: conexión que existe entre la impresora y la Estación de trabajo del usuario. Se modela con USB, pero puede existir otro tipo de conexión, todo depende del tipo de impresora que esté usando el usuario y el tipo de conexión que utilice esta.

3.11. Pruebas

Durante la definición de cada iteración se estableció la planificación de las pruebas que debían ser aplicadas para determinar si el sistema satisface las funcionalidades requeridas por el cliente. Para ello se hizo necesaria la confección de Casos de Prueba. A continuación se incluyen los casos de prueba de la HU Gestionar Plan de Auditoría y Revisión. El resto de los casos de prueba se encuentran en el Anexo 5.

En la Tabla 10 se describe el caso de prueba perteneciente a la HU Gestionar Plan de Auditoría y Revisión, específicamente al escenario “Insertar Plan de Auditoría y Revisión”. En él se describe como se debe realizar la prueba para verificar que el sistema permite insertar el plan de auditoría y revisión de forma correcta. Además se incluye el responsable y la evaluación otorgada después de ejecutarse la prueba.

[4] **USB** es la sigla de Universal Serial Bus (Bus Universal en Serie, en castellano).

Tabla 10: Caso de Prueba del escenario "Insertar Plan de Auditoría y Revisión".

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU01-P1	**Nombre Historia de Usuario:** Gestionar Plan de Auditoría y Revisión
Nombre de la persona que realiza la prueba: Mirian del Carmen González Elias	
Descripción de la Prueba: Se insertan los datos del plan de auditoría y revisión, inicialmente se insertarán incorrectamente para verificar las validaciones del sistema, luego de forma correcta para comprobar que los datos sean almacenados.	
Condiciones de Ejecución: El usuario debe tener los permisos suficientes para realizar esta operación.	
Entrada / Pasos de ejecución: 1. No seleccionar un trimestre. 2. No seleccionar una actividad. 3. Insertar un trimestre que ya esté almacenado en la base de datos. 4. Insertar dos veces la actividad de auditoría en un trimestre. 5. Insertar los datos correctos. 6. Verificar que aparece el nuevo plan de auditoría y revisión.	
Resultado Esperado: El sistema debe alertar al usuario cuando no se seleccionen datos en los campos obligatorios (Trimestre, Actividad); además debe alertar cuando ya exista un elemento con el mismo nombre, almacenado en la base de datos. Cuando se inserten los datos correctamente, el sistema debe almacenarlos en la base de datos y mostrar el plan de auditoría y revisión.	
Evaluación de la Prueba: Satisfactoria	

En la Tabla 11 se describe el caso de prueba perteneciente a la HU Gestionar Plan de Auditoría y Revisión, específicamente al escenario "Ver Plan de Auditoría y Revisión". En él se describe como se debe realizar la prueba para verificar que el sistema muestra los planes de auditoría y revisión

almacenados en la base de datos y la información específica de cada uno de ellos. Además se incluye el responsable y la evaluación otorgada luego de ejecutarse la prueba.

Tabla 11: Caso de Prueba del escenario "Ver Plan de Auditoría y Revisión".

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU01-P2	**Nombre Historia de Usuario:** Gestionar Plan de Auditoría y Revisión
Nombre de la persona que realiza la prueba: Mirian del Carmen González Elias	
Descripción de la Prueba: Se muestran los planes de auditoría y revisión que están almacenados en la base de datos. Se puede visualizar la información específica de un Plan de Auditoría y Revisión.	
Condiciones de Ejecución: Deben existir al menos un plan de auditoría y revisión almacenado en la base de datos. Debe estar abierto algún cliente de base de datos de PostgreSQL en el que esté cargada la base de datos del sistema.	
Entrada / Pasos de ejecución: 1. Mostrar los planes de auditoría y revisión almacenados en la base de datos. 2. Verificar que se visualizan los datos de un plan de auditoría y revisión específico.	
Resultado Esperado: El sistema debe mostrar los planes de auditoría y revisión almacenados en la base de datos y debe permitir visualizar toda la información relacionada con un plan de auditoría y revisión específico.	
Evaluación de la Prueba: Satisfactoria	

En la Tabla 12 se describe el caso de prueba perteneciente a la HU Gestionar Plan de Auditoría y Revisión, específicamente al escenario "Modificar Plan de Auditoría y Revisión". En él se describe como se debe realizar la prueba para verificar que el sistema modifica la información de un determinado plan de auditoría y revisión almacenado en la base de datos. Se incluye además el responsable y la evaluación otorgada luego de ejecutarse la prueba.

Tabla 12: Caso de Prueba del escenario "Modificar Plan de Auditoría y Revisión".

Caso de Prueba	
Código Caso de Prueba: HU01-P3	**Nombre Historia de Usuario:** Gestionar Plan de Auditoría y Revisión
Nombre de la persona que realiza la prueba: Mirian del Carmen González Elias	
Descripción de la Prueba: Se modifican los datos de un plan de auditoría y revisión, inicialmente se modificarán incorrectamente para verificar las validaciones del sistema, luego de forma correcta para comprobar que los datos sean almacenados y cargados.	
Condiciones de Ejecución: El usuario debe tener los permisos suficientes para realizar esta operación.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción Modificar en un plan de auditoría y revisión específico. 2. Modificar los datos en los campos requeridos (Trimestre y Actividad), dejar los campos sin seleccionar. 3. Modificar el trimestre dejándolo sin seleccionar. 4. Modificar las actividades dejándola sin seleccionar. 5. Modificar de forma correcta los datos. 6. Verificar que el Plan de Auditoría y Revisión aparece en el listado con los nuevos datos.	
Resultado Esperado: El sistema debe alertar al usuario cuando no se seleccionan datos en los campos obligatorios (Actividad). Cuando se modifiquen los datos correctamente, el sistema debe almacenarlos en la base de datos y mostrarlos.	
Evaluación de la Prueba: Satisfactoria	

En la Tabla 13 se describe el caso de prueba perteneciente a la HU Gestionar Plan de Auditoría y Revisión, específicamente al escenario "Eliminar Plan de Auditoría y Revisión". En él se describe como se debe realizar la prueba para verificar que el sistema elimina un determinado plan de auditoría y revisión

almacenado en la base de datos. Se incluye además el responsable y la evaluación otorgada luego de ejecutarse la prueba.

Tabla 13: Caso de Prueba del escenario "Eliminar Plan de Auditoría y Revisión".

Caso de Prueba	
Código Caso de Prueba: HU01-P4	**Nombre Historia de Usuario:** Gestionar Plan de Auditoría y Revisión
Nombre de la persona que realiza la prueba: Mirian del Carmen González Elias	
Descripción de la Prueba: Se elimina un plan de auditoría y revisión.	
Condiciones de Ejecución: El usuario debe tener los permisos suficientes para realizar esta operación.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción Eliminar en un plan de auditoría y revisión específico. 2. Verificar que el plan de auditoría y revisión eliminado no aparece.	
Resultado Esperado: El sistema debe mostrar una página de aviso al usuario preguntando si realmente desea eliminar. Cuando se acepte, debe eliminar el Plan de Auditoría y Revisión. Después de eliminado, el elemento no debe aparecer en la base de datos.	
Evaluación de la Prueba: Satisfactoria	

3.11.1. Resultado de las pruebas

Se realizaron dos iteraciones de las pruebas de aceptación con el cliente, para conocer el grado de satisfacción hacia el producto. Durante la primera iteración se encontraron seis no conformidades (NC) de prioridad Baja, relacionadas con la redacción en la aplicación, las cuales fueron resueltas por el equipo de desarrollo. Se realizó una segunda iteración, en al cual no se encontraron NC, evaluándose de esta manera todos los Casos de Prueba como Satisfactorio.

Luego, a petición del cliente y del equipo de desarrollo se propuso realizar pruebas de funcionalidad, aplicando la técnica de caja negra (las cuales se realizaron por parte del Grupo de Calidad del centro

FORTES). Durante la primera iteración del proceso se encontraron 10 no conformidades, dos de prioridad Alta, tres de prioridad Media y cinco de prioridad Baja. En la segunda iteración de las pruebas fueron detectadas 9 no conformidades, de prioridad baja, relacionadas con errores de correspondencia entre el Caso de Prueba y la aplicación, y en la validación de la entrada de datos, las cuales fueron resueltas por el equipo de desarrollo. Se realizó la tercera iteración de estas pruebas donde no fueron detectadas no conformidades.

3.12. Tratamiento de errores

La informática no es perfecta, y lo demuestran la cantidad de mensajes de error que aparecen mientras se trabaja con los distintos productos de *hardware* y *software*. Para darle solución a estos problemas es conveniente definir el término tratamiento de errores, que permite identificar, localizar, analizar y eliminar los errores en la implementación de un *software*. Las aplicaciones deben estar capacitadas para enfrentar las entradas erróneas suministradas por los usuarios y los errores que puedan ser generados por el comportamiento incorrecto de componentes internos; por ejemplo, errores en tiempo de ejecución o en consultas a la base de datos.

Mediante el tratamiento de errores se pueden evitar escrituras incorrectas, tipos de datos erróneos, direcciones de correos incorrectos o fechas no válidas. A través de ventanas de diálogo, mensajes de alerta o mensajes intuitivos, se le señala al usuario en que campo se encuentra el error.

En el Módulo Auditoría y Revisión del Sistema de Gestión de la Calidad del centro FORTES, los datos que introduce el usuario son validados con Javascript y PHP, mostrando mensajes de error de forma intuitiva que indican al usuario los datos que están incorrectos e imposibilitan el envío de la solicitud al servidor mientras los formularios contengan valores incorrectos. Además, mediante estos mensajes se orienta al usuario acerca de las causas del error y la estructura que deben tener los campos para evitarlos.

3.13. Conclusiones parciales

- Se incluyeron y explicaron los artefactos generados por la metodología SXP para la implementación, las pruebas y su aplicación en el Módulo Auditoría y Revisión.
- En el artefacto Plan de *Releases* se definieron tres iteraciones. En cada iteración se incluyeron las HU de una misma prioridad, quedando reservada la primera para las HU significativas. Este plan conjuntamente con el artefacto Tareas de Ingeniería, marcó el proceso para cumplir con los objetivos de cada HU, de cada iteración y del sistema en general.
- Se establecieron las políticas de seguridad presentes en el sistema y se explicaron las que brinda el CMS Drupal.
- Se describió el tratamiento de errores en el sistema como parte de la validación de formularios.

Conclusiones Generales

La investigación desarrollada y los resultados obtenidos permiten a los autores plantear las siguientes conclusiones:

- El análisis de las características de los procesos de auditorías y revisiones de *software*, teniendo en cuenta el procedimiento descrito para el centro FORTES, posibilitó el diseño de una solución que se adecua a las necesidades del cliente.
- La búsqueda y análisis de sistemas semejantes demostró, que las herramientas para realizar auditorías a nivel internacional son privativas y no fueron diseñadas para realizar auditorías de *software,* mientras que la existente a nivel nacional no permite gestionar los planes y cronogramas de auditoría y revisión, ni evaluar auditoría, evidenciándose la necesidad de crear un nuevo producto de este tipo para en centro FORTES.
- Mediante la aplicación de la metodología ágil SXP se logró efectividad y rapidez en el desarrollo de *software* y la gestión de proyecto.
- Se obtuvo un producto funcional probado y validado durante su proceso de desarrollo utilizando pruebas de aceptación, desarrollado con tecnologías libres. Donde en la segunda iteración se alcanzó el 100% en las pruebas realizadas de “Satisfactoria” por parte del cliente.

Recomendaciones

Basados en la investigación realizada y los resultados obtenidos durante la realización del presente trabajo de diploma, se recomienda:

1. A la asesora de calidad enriquecer las funcionalidades del Módulo Auditoría y Revisión para el Sistema de Gestión de Calidad del centro FORTES, con el objetivo de permitir modificar las Listas de Chequeo y las Entrevistas.
2. A los asesores de calidad, utilizar el Módulo Auditoría y Revisión del Sistema de Gestión de Calidad del centro FORTES en el resto de los centros productivos de la UCI.
3. A la asesora de calidad proponer la migración de todos los módulos hacia versiones actuales de Drupal.

Referencias Bibliográficas

1. **Nubia Fernández Grajales.** ENTER@TE. [En línea] Octubre de 2005. http://www.enterate.unam.mx/Articulos/2005/octubre/auditoria.htm.

2. **IEEE.** Norma IEEE1028 - 97. *Norma IEEE1028 - 97.* [En línea] [Citado el: 30 de enero de 2012.]

3. **UCI.** CALISOFT - Centro de Calidad para Soluciones Informáticas. [En línea] 2009. [Citado el: 5 de octubre de 2011.] http://calisoft.uci.cu/index.php/centro.

4. FORTES: Centro de Tecnologías para la Formación. [En línea] Enero de 2010. [Citado el: 5 de Octubre de 2011.] http://portal.fortes.prod.uci.cu/.

5. Definicion. [En línea] [Citado el: 5 de octubre de 2011.] http://definicion.de/auditoria/.

6. **B, Ivan Dimitrie Moyasevich.** Temas de Ingeniería Industrial. [En línea] [Citado el: 6 de octubre de 2011.] http://perso.wanadoo.es/idmb/a_ing/temas/auditoria_informatica.htm.

7. **Cepeda, Gustavo Alonso.** Auditoria y Control Interno. Colombia : Editorial McGraw-Hill, 1997.

8. **Hernández, Enrique Hernández.** Auditoría Informática: Un Enfoque Metodológico. México : s.n., 2000.

9. Norma internaciona ISO 19011:2002. [En línea] [Citado el: 2 de octubre de 2011.] http://webstore.ansi.org/RecordDetail.aspx?sku=ISO+19011:2002[S].

10. **Haderne, Lic. Mrisa.** *Introducción a la Auditoría Informática.*

11. **Buades, Gabriel.** [En línea] 2002. [Citado el: 18 de enero de 2012.] http://dmi.uib.es/~bbuades/auditoria/sld001.htm.

12. **IEEE Standard for Software Reviews.** IEEE Std 1028-1997. 1997.

13. *Revisión por pares: ¿Qué es y para qué sirve?* **Ladrón de Guevara Cervera, Michele, y otros, y otros.** 2, colombia : Salud Uninorte, 2008, Vol. 24. pp. 258-272.

14. Norma internacional ISO 9000:2000. [En línea] [Citado el: 3 de noviembre de 2011.] http://www.navactiva.com/es/descargas/pdf/acal/LegNormas.pdf.

15. **Revista Española de Innovacion, Calidad e Ingenieria del software.** Una revisión sistemática de la adaptación del proceso software. [En línea] 2 de octubre de 2007. [Citado el: 4 de octubre de 2011.] http://www.ati.es/IMG/pdf/PedreiraVol3Num2.pdf.

16. **PRESSMAN, R. S.** *Ingeniería del Software.Un enfoque práctico.* 2002. p. 8448132149.

17. **Chavez, R. S.** *Verificación y Validación del Software. Módulo 2: Revisiones de SW.* Monterrey : s.n., 2005.

18. **Fagan, Michael E.** "Advances in Software Inspections".IEEE Transactions on Software Engineering. [En línea] 1986. [Citado el: 6 de octubre de 2011.] www.mfagan.com/pdfs/aisi1986.pdf. Vol. 12, Number 7, July 1986, pp. 744-751..

19. Eniac. [En línea] [Citado el: 4 de diciembre de 2011.] www.eniac.com/productos/acl.htm.

20. Interop. [En línea] [Citado el: 4 de diciembre de 2011.] www.interop-la.com/docs/auto_audit.pdf.

21. **Audita.** Audita. [En línea] [Citado el: 4 de diciembre de 2011.] http://www.sitsoft.com.ar/Audita.asp.

22. **Jacobson, Ivan, Booch, Grady y Rumbaugh, James.** *El Proceso de Unificado de Desarrollo de Software.* Madrid, España : Addison Wesley, 2004.

23. EcuRed. [En línea] [Citado el: 3 de noviembre de 2011.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_Tradicionales.

24. **Riola, Jose Carlos Carvajal.** Herramientas y Modelo de Desarrollo para Aplicaciones JAVA EE como Metodología Empresarial. *Tesis Final de Máster.* 2008.

25. **Universidad de Alicante, Grupo ISSI (Ingeniería del Software y Sistemas de Información).** Metodologías Ágiles en el Desarrollo de Software. [En línea] 12 de nobiembre de 2003. [Citado el: 2010 de octubre de 6.]

26. W4. [En línea] [Citado el: 5 de noviembre de 2011.] http://w4-bpm.es/principios-manifiesto-agil.htm.

27. Proyectos Ágiles. [En línea] [Citado el: 9 de noviembre de 2011.] http://www.proyectosagiles.org/que-es-scrum.

28. **Proyectalis.** Proyectalis. Gestion de Proyecto. [En línea] [Citado el: 4 de noviembre de 2011.] http://www.proyectalis.com/servicios/formacion/scrum.

29. **Escribano, Gerardo Fernández.** Introducción a Extreme Programming. [En línea] 2002. [Citado el: 6 de nobiembre de 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

30. **FORTES, Universidad de las Ciencias Informáticas.** Documento Visión del centro FORTES. Ciudad de La Habana : s.n., 2010.

31. **Romero, Gladys Marsi Peñalver.** "Trabajo de diploma: Metodología ágil para proyectos de software libre". *"Trabajo de diploma: Metodología ágil para proyectos de software libre".* Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2008.

32. Comunidad de usuarios de Drupal. [En línea] [Citado el: 6 de diciembre de 2011.] http://drupal.org.es/.

33. Computer Audio Video Systems Integrator. [En línea] [Citado el: 3 de diciembre de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd.

34. **Humberto Espinoza.** PosgreSQL. Una Alternativa de DBMS Open Source. [En línea] 2005. [Citado el: 3 de nobiembre de 2011.] http:\\www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

35. Cibernetia. [En línea] [Citado el: 5 de noviembre de 2011.] http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php.

36. Apache Http Server Proyect.Information on the Apache HTTP Server Projec. [En línea] [Citado el: 4 de noviembre de 2011.]

37. IIS. [En línea] [Citado el: 5 de noviembre de 2011.] http://www.iis.net.

38. Centro de Especialización Profesional y Extensión Universitaria: CEPEU. [En línea] [Citado el: 4 de diciembre de 2011.] http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_4/manual-completo-php-5.pdf.

39. W3C. [En línea] [Citado el: 5 de diciembre de 2011.] http://www.w3c.es/divulgacion/guiasbreves/XHTML.

40. W3C. [En línea] [Citado el: 5 de dsciembre de 2011.] http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.

41. EcuRed. [En línea] [Citado el: 6 de diciembre de 2011.] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

42. DesarrolloWeb. [En línea] [Citado el: 6 de diciembre de 2011.] http://www.desarrolloweb.com/articulos/1178.php.

43. **Romero, Gladys Marsi Peñalver.** Expediente para metodología SXP. *UCI. Expediente para metodología SXP.* 2008.

44. **Peñalver, G. Meneses, A. García, S.** SXP, Metodología Agíl para el Desarrollo de Software. Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

45. Facultad de Ciencias Exactas y Naturales y Agrimensura - Universidad Nacional del Nordeste . [En línea] 10 de julio de 2002. [Citado el: 15 de marzo de 2012.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SEGUNIX012.htm.