Universidad de las Ciencias Informáticas

Facultad 3

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

Módulo de Administración del Sistema Automatizado de Monitoreo y Control de los Servicios (SAMOS) del centro de soporte UCI.

Autor(es): Ariel Ramírez Leyva

Iván Horta Wong

Tutor (es): Msc. Yulio Seriocha García Gallardo

Ing. Neybis Lago Clara

La Habana, Cuba
"Año 55 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos por este medio que somos los únicos autores del presente trabajo de diploma y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría a los _____ días del mes de _____ del año _____.

Ariel Ramírez Leyva

_____

Autor

Iván Horta Wong

_____

Autor

Ing. Neybis Lago Clara

_____

Tutor

MSc. Yulio Seriocha García Gallardo

_____

Tutor

DATOS DE CONTACTO

Nombre y apellidos del tutor: *Yulio Seriocha García Gallardo.*

Institución: *Universidad de las Ciencias Informáticas.*

Título: *Máster en Informática Aplicada.*

Correo electrónico: *ygarciag@uci.cu*

Graduado de Ingeniero en Ciencias Informáticas de la UCI del año 2007, se desempeña como especialista del centro de soporte UCI. Es Máster en Informática Aplicada. Categoría docente: Asistente.

Nombre y apellidos del tutor: *Neybis Lago Clara.*

Institución: *Universidad de las Ciencias Informáticas.*

Título: *Ingeniero en Ciencias Informáticas.*

Correo electrónico: *nlago@uci.cu*

Graduada de Ingeniera Informática de la UCI del año 2011, se desempeña como especialista del centro de soporte UCI. Categoría docente: Instructor.

DEDICATORIA

Dedico esta investigación a mi abuelita Aurora, mi tata querida, que aunque ya no esté siempre la llevaré en un lugar muy especial de mi corazón. A mis padres, estas dos personas maravillosas que me dieron el don de la vida. A mis tías Roxana y Niriam por ser mis segundas madres y por darme de su amor como si fuera su verdadero hijo. A mi prima Ayme por brindarme esos grandes sentimientos de amor y ternura. A mi hermano y sobrinos por complementar mi felicidad con su existencia.

Ariel Ramírez Leya

A mi familia entera, específicamente a mi padre Ivan, a mi madre Normaris, a mi hermano Ian y a mi abuela Norma, incluyo tambien a mis amigos del alma Marcos y Damian y a todas las personas que no dudaron de mi para alcanzar mis metas.

Ivan Horta Wong

AGRADECIMIENTOS

Quiero ante todo agradecer a la familia tan linda que tengo por ayudarme siempre y enseñarme a transitar en el camino tan difícil de la vida, dándome la mano siempre que la he necesitado y brindándome su cariño y apoyo. A mis cuatro estrellas guías, las mujeres más lindas de este mundo que llenan de luz mi alma y mi existencia. Mi Abuela Aurora, mi tata; por darme la oportunidad de ser su nieto malcriado. Gracias Abuela por todos los momentos que pase contigo, mi mayor deseo era que me acompañaras en este día tan importante para mi vida, pero dios dispuso que fueras un ángel y me vieras desde el cielo, solo quiero que sepas que todo lo que hecho para llegar hasta aquí ha sido gracias a ti. A mi mamá Adis, mi pequeña, por profesarme tanto amor y dedicación, por ser ejemplo de empeño y perseverancia, por estar siempre dispuesta cuando la he necesitado sin importar el sacrificio, pero por sobre todas las cosas por ser la madre que es. A mis tías Rosi y Niriam, a las cuales considero mis otras madres, por su preocupación y su amor entregado. Gracias a las cuatro por siempre tenerme en el centro de sus vidas.

A mi papá, por su apoyo y cariño, porque de él aprendí lo que es ser trabajador y comprometido, porque sin saberlo me ha enseñado muchas cosas que han ayudado a formar el hombre que soy hoy. A mis primas Ayme y Lissett que la distancia no le ha impedido para regalarme tan lindo amor y comprensión como las hermanas que me hubiera gustado mucho tener. A mi hermano Yuri, que me ha dado su mano cuando más lo necesitaba y a mis adorables sobrinitos Cristian y Estefani por entregarme ese montón de cariño tan tierno que alegra mi corazón. A mis tíos Yordanis y Alberto, por ser ellos mi apoyo en muchos momentos difíciles en estos 5 años, no tengo palabras para decirles lo agradecido que estoy por todo lo que han hecho por mí y la familia. Al pequeño Paulo por ser el tesoro de la familia con tan linda sonrisa.

A mi hermano Daldi, por apoyarme siempre, por estar siempre dispuesto, por no dudar y confiar en mí, por ser mí escudo cuando lo he necesitado. A mi eterno piquete del IPVCE compuesto por Zulema, Eiler, Eliecer, Renides y Fernando por brindarme su amistad y apoyo durante tantos años. A mi familia de la FEU por los grandes momentos que me regalaron, aportandome a mi formación cualidades importantes que de seguro me servirán para enfrentar momentos que me prepara la vida. A mis entrenadores y compañeros del equipo de Taekwondo por estos 5 años compartiendo como hermanos. A mis compañeros de aula y amigos, los de aquí y los de allá, por darme la oportunidad de conocerlos, por su apoyo incondicional, por ser tan especiales y estar ahí siempre que los necesité en los buenos y en los malos momentos, en especial a Macias, Dayana, Yerandi, Luis Manuel, Yadier, Josúe, Raniel, Jorge Luis, Yordan, Carmen, Lisandra, Miguel, Juán, Jeandi, Yanet, Eiler, Anel, Zoemi, Zoima, Yisel, Reiner y el más gordito e inquieto de todos Jorgito.

A los profesores que durante toda la etapa de la universidad me prepararon para desempeñarme como una profesional.

A todas las personas que me han ayudado en la realización de la tesis, en especial a las profesoras Yanay y Danaysa, a mis tutores Neybis y Yulio por guiarnos y ayudarnos en este largo camino.

A mi compañero de tesis por ser más que un compañero un amigo. Gracias mi hermano.

Ariel Ramírez Leyva

A mi padre por apoyarme en cada decisión que tomé, a mi madre por siempre estar ahí en cada momento crucial de mi vida, a mi hermano por no separarse de mi ni un segundo y siempre alegrarme el dia con sus ocurrencias, a mi abuela por siempre darme su opinión incondicional, a mis amigos por siempre preocuparse por mi sin importar la distancia ni las adversidades.

En la universidad le agradezco a una persona que fue especial para mi,Yaimari, que soportó 4 años mis caprichos, a todos con los que he compartido aula en especial a : Alianna, Yander, El Chen, Fidel, Yosi,Luis Manuel, Jorge Jesus, Yerandi,Wilfredo Palma, Wilfredo Barrios, Yordan, Medinilla, Jesus, Eddy, El Rafa, Carlos Javier, Javier, Carlos Miguel, Angel Luis, Julio Cesar, Jeandy, Juan Ignacio, Camilo, Dargel, Carmen, Eiler y Anel.

También quiero agradecer a la pequeña familia de la cual he formado desde que llegué aqui: Juan Darien, Ariel, Yadier, Yanet, Valido, Dayana y Leanet.

A mi tutora Neybis por siempre estar disponible y a la que es como una tutora para mi, Danaysa, por ayudarme en toda mi carrera.

Tambien quiero agradecer a un grupo de profesores que influyeron en mi comportamiento a lo largo de estos 5 años: Daniel, Boris, Maurice, Herson, Cedeño, Sergio, Odette y Orlando.

Ivan Horta Wong

RESUMEN

Con el desarrollo de las Tecnologías de la Información y el aumento de su impacto social, cada vez son más las instituciones que deciden incorporar sistemas de administración a su flujo de trabajo. Puesto que por medio de procesos administrativos se obtienen características asociadas al monitoreo y control de los servicios, dando soporte al cumplimiento de los objetivos de los respectivos negocios. Actualmente el centro de soporte de la Universidad de las Ciencias Informáticas no cuenta con un sistema de administración que realice el monitoreo y control los servicios de soporte técnico.

Debido a esto, en el presente trabajo se propone un módulo de administración para gestionar los servicios de soporte técnico del Sistema Automatizado de Monitoreo y Control de los Servicios. Para ello se realizó un estudio del estado del arte, donde se analizaron los procesos de administración que se realizan en diferentes herramientas de Gestión de Servicios de Tecnologías de la Información.

Fueron descritas las herramientas y tecnologías utilizadas en la solución desarrollada. Para llevar a cabo su implementación se utilizó la metodología de desarrollo Programación Extrema, generándose los artefactos pertinentes en cada fase, culminando con las pruebas para validar el sistema y comprobar que cumple con todo lo definido por el centro de soporte UCI.

Palabras Claves: administración, control, gestión, información, monitoreo, servicios, tecnología.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de la Información (TI) han logrado cambiar la forma tradicional que las organizaciones utilizaban para la toma de decisiones y la satisfacción de las necesidades de sus clientes. Por otra parte, las TI se han convertido en un área de gran amplitud e impacto en todos los aspectos de la vida cotidiana, incluyendo la gerencia de cualquier empresa, en la cual hoy en día es casi indispensable. La integración de las TI en los procesos de negocio, representa un objetivo significativo para aportar valor a las empresas, convirtiéndolo en un reto a nivel mundial.

Es importante establecer que las TI se definen como aquellas herramientas y métodos empleados para recabar, retener, manipular o distribuir información. Se encuentran generalmente asociadas con las computadoras y las tecnologías afines aplicadas a la toma de decisiones (Bologna y Walsh, 1997). Con la utilización de las TI en la gestión de información, se brinda una mejor administración de los recursos y a la vez menos costos en los procesos realizados en una empresa o institución.

El uso de las TI generalmente redunda en un procesamiento más rápido y confiable de los datos de una institución. Puesto que la información resultante tiene mayor movilidad, accesibilidad, y cuenta con mayor integridad, que cuando se procesa de forma manual (Meltom Technologies, 2013). Las TI ayudan a obtener la aptitud necesaria para actualizarse y perfeccionarse dinámicamente con las necesidades del mercado o con los cambios en la estrategia para llevar a cabo los procesos de negocio.

Si los procesos y servicios de TI son implementados, gestionados y respaldados en la manera apropiada, el negocio será más exitoso, sufrirá menos interrupciones y pérdida de horas productivas, reducirá costos, incrementará ingresos, mejorará relaciones públicas y alcanzará los objetivos del negocio (Customer Care Associates, 2011). La calidad de los resultados de una empresa o institución dependen de una buena Gestión de Servicios de las Tecnologías de la Información (GSTI) por ser la clave de la recolección, análisis, producción y distribución de la información dentro de una organización.

La GSTI es una disciplina basada en procesos, enfocada en alinear los servicios de TI proporcionados con las necesidades de los clientes, poniendo énfasis en los beneficios que puede percibir el usuario final (ALM Solutions, 2013). Entre las principales funciones de las GSTI se encuentra apoyar las actividades del mantenimiento preventivo, el cual hace referencia a revisiones, comprobaciones y cambios donde se pueden mencionar elementos importantes, como aquellos relacionados con el monitoreo y control de servicios.

El monitoreo y control dentro de la GSTI se debe destacar como un factor de suma importancia en la evaluación de los procesos en una institución. Ofrece métodos de gobernabilidad y gestión, logra tomar el control de los servicios implementados, cuenta con la flexibilidad necesaria en relación con la implementación de servicios y con las interacciones para poder cumplir con las necesidades del negocio. Es el encargado de garantizar la seguridad y fiabilidad de cualquier tipo de sistema, además de lograr los principales objetivos de la GSTI.

Entre los objetivos de la GSTI se encuentra proporcionar una adecuada gestión de la calidad, aumentar la eficiencia, alinear los procesos de negocio y la infraestructura TI, reducir los riesgos asociados a los servicios TI y generar ganancia. Para su cumplimiento se utiliza el soporte de aplicación de las mejores prácticas, marcos referenciales y estándares de aceptación internacional, tales como ISO/IEC[1] 20000, ISO 9001, CMMI-SVC[2], ITIL[3] y COBIT[4].

Todas estas herramientas, estándares y procesos surgieron a medida que la tecnología se ha estado desarrollando al cabo de los tiempos. Cuba a pesar de ser un país subdesarrollado, no está ausente del avance de la ciencia y la tecnología. Ha fundado en varios puntos de la isla, un gran grupo de instituciones para la formación de profesionales y el desarrollo tecnológico. Se pueden mencionar ejemplos como las empresas: Softel, Desoft y la creación de la Universidad de las Ciencias Informáticas (UCI) en el año 2002 por el Comandante en Jefe Fidel Castro Ruz.

La UCI es una institución de estudios donde se aplica el método de universidad productiva, la cual persigue el objetivo de elevar el desarrollo económico y formación de jóvenes con aptitudes suficientes para alcanzar conocimientos que aporten al avance de esferas dentro de la ciencia y la tecnología. Cuenta con 24 centros de desarrollo donde se gestionan productos informáticos, entre ellos se puede mencionar al centro de soporte UCI, el cual, tiene como misión brindar el servicio de soporte técnico a las aplicaciones informáticas y productos desarrollados por la red de centros de desarrollo.

Actualmente el centro de soporte UCI no cuenta con una estrategia clave para que la gestión de servicios de soporte técnico sea de la manera requerida. La información relacionada con este proceso se gestiona de forma manual en formato duro y a través de documentos Excel. Por otro lado los especialistas no pueden acceder a la información

[1] Organización Internacional de Normalización / Comisión Electrotécnica Internacional, significado de sus siglas en español.
[2] Modelo de Madurez de la Capacidad Integrado, significado de sus siglas en español.
[3] Biblioteca de Infraestructura de Tecnologías de Información, significado de sus siglas en español.
[4] Objetivos de Control para Información y Tecnologías Relacionadas, significado de sus siglas en español.

simultáneamente, y no existe una vía para analizar y resumir de un modo detallado el gran cúmulo de datos relacionados con los servicios de soporte técnico. Además del desconocimiento de técnicas para recopilar información del cliente dirigidas a comprender sus necesidades.

Todo esto conlleva a que se puedan cometer errores en el control de toda la información y atrasos en la entrega de informes. También trae como consecuencia que el almacenamiento de los datos no cuente con la seguridad requerida, lo que propicia que estos sean vulnerables ante posibles cambios, que no van en correspondencia con la información que se maneja. En los marcos de las observaciones anteriores, se puede plantear que los problemas provocados por la situación actual del centro de soporte UCI, afectan en gran medida el tiempo de trabajo de los especialistas a la hora de llevar a cabo sus tareas productivas.

Luego de realizar un estudio de las necesidades que presenta la dirección del centro de soporte UCI y establecer un seguimiento de los servicios que se brindan, se plantea como **problema a resolver:** El modo en que se monitorea y controla la información en el centro de soporte UCI afecta el tiempo de trabajo de los especialistas en la gestión de los servicios de soporte técnico.

Para el problema expuesto se define el **objeto de estudio**: El monitoreo y control de los servicios de soporte técnico.

Para darle solución a la problemática existente se establece como **objetivo general:** Desarrollar un Módulo de Administración que contribuya a disminuir el tiempo de trabajo de los especialistas en la gestión de los servicios de soporte técnico.

Enmarcado en el **campo de acción**: Monitoreo y control de los servicios de soporte técnico en el centro de soporte UCI.

Para dar cumplimiento al objetivo general se proponen los siguientes **objetivos específicos**:

1. Definir el marco teórico de la investigación y el estado del arte sobre los sistemas de monitoreo y control de los servicios existentes y el marco de gobierno de COBIT para sentar las bases de la investigación.
2. Realizar el análisis y diseño del módulo de administración del SAMOS para un mejor entendimiento del negocio.

3. Implementar el módulo de administración del SAMOS para llevar a cabo la producción de las funcionalidades especificadas por el cliente.
4. Realizar métricas dirigidas al tamaño operacional de las clases y a las relaciones de clases para validar el diseño del sistema.
5. Aplicar pruebas de aceptación, de unidad y estrés para validar funcionalmente el sistema.

Con el propósito de guiar y perfilar el trabajo hacia el alcance de los objetivos trazados, se definieron las siguientes **tareas investigativas**:

1. Realización del diseño teórico de la investigación.
2. Estudio de conceptos fundamentales de la investigación.
3. Estudio del marco de trabajo COBIT.
4. Estudio de las herramientas y tecnología a utilizar en el proceso de desarrollo.
5. Realización del análisis y diseño de la solución.
6. Realización de las métricas de diseño.
7. Realización del modelo de implementación.
8. Implementación de los componentes del modelo.
9. Diseño de los casos de prueba de la funcionalidades implementadas.
10. Ejecución de las pruebas de funcionalidad.

Como **idea a defender** se plantea**:** Con el desarrollo de un Módulo de Administración se disminuirá el tiempo de trabajo de los especialistas en la gestión de los servicios de soporte técnico.

Definición y operacionalización de las variables.

Variable dependiente: tiempo de trabajo de los especialistas en la gestión de los servicios de soporte técnico. **Variable independiente:** Módulo de Administración.

Tabla 1. Operacionalización de la variable dependiente.

Nombre de la Variable	U/M
tiempo de trabajo de los especialistas en la gestión de los servicios de soporte técnico	Minutos, Segundos y Milisegundos

Muestreo

Población: los 20 especialistas del centro de soporte UCI.

Unidad de Estudio: el monitoreo y control de los servicios de soporte técnico.

Selección de la muestra y criterio de selección

Muestra: se selecciona de modo intencional, está compuesta por 4 especialistas del centro de soporte UCI.

Diseño del experimento

Tipo de experimento: Para la validación de la propuesta se realizará un diseño preexperimental.

Instrumentos: Para medir la variable operacional se utiliza la comparación entre dos maneras de realizar el monitoreo y control de los servicios de soporte técnico.

Los **métodos científicos utilizados** en la presente investigación fueron:

- Métodos Teóricos:
  - ✓ Histórico – Lógico**:** Para determinar las tendencias actuales de los sistemas de monitoreo y control.
  - ✓ Analítico – Sintético**:** Para el procesamiento de la información del negocio de la organización y arribar a las conclusiones de la investigación, así como para precisar las características del trabajo a realizar.
  - ✓ Modelación: Para modelar la realidad futura que será la investigación, los principales elementos que lo componen y su funcionamiento, además de la utilización del Lenguaje Unificado de Modelado (UML) para la modelación teórica del resultado generándose así varios modelos, ejemplo: Modelo conceptual y Diagrama de despliegue.
- Métodos Empíricos:
  - ✓ Consulta de las fuentes de información**:** Para seleccionar la información necesaria para construir el marco teórico.
  - ✓ Observación**:** Para reunir información visual sobre lo que el objeto de estudio hace y cómo se comporta.
  - ✓ Pruebas**:** Para valorar el desempeño del sistema elaborado.

- ✓ Experimentación: utilizado para determinar una forma confiable de monitorear y controlar los servicios de soporte técnico de manera que contribuya a disminuir el tiempo de trabajo de los especialistas.

El trabajo de diploma está estructurado por:

Capítulo I: Fundamentación teórica del Módulo de Administración del Sistema Automatizado de Monitoreo y Control de los Servicios (SAMOS) del centro de soporte UCI: En este capítulo se realiza el estudio del estado del arte sobre la gestión de servicios de monitoreo y control. Se establece la fundamentación teórica de la investigación, a partir de hacer un análisis crítico de las disposiciones establecidas por COBIT y de los principales sistemas de gestión de servicios administrativos, además de la caracterización de las metodologías y herramientas de desarrollo de software que se utilizaron.

Capítulo II: Propuesta de solución del Módulo de Administración del Sistema Automatizado de Monitoreo y Control de los Servicios (SAMOS) del centro de soporte UCI: Dentro de este capítulo se definen los requisitos del sistema. Se realiza una descripción de las funcionalidades utilizando historias de usuarios, se muestra una planificación de cómo se trabajaron las iteraciones y el diseño de la aplicación. Se desarrollan las descripciones de la arquitectura de software y los patrones de diseño utilizados. Además se muestra un modelo de datos, para una mejor comprensión de la información que será manipulada y se lleva a cabo la descripción de las tareas de ingeniería generadas por cada historia de usuario.

Capítulo III: Implementación y Pruebas de la solución del Módulo de Administración del Sistema Automatizado de Monitoreo y Control de los Servicios (SAMOS) del centro de soporte UCI: En este capítulo se muestra el diagrama de despliegue. Se describen los estándares de codificación empleados para una mayor legibilidad del código. Por otro lado contiene todo lo relacionado con la validación de la solución propuesta. Se realizan las validaciones de las funcionalidades del sistema mediante pruebas de aceptación para comprobar que este se ajusta a las necesidades del cliente. Se realizan pruebas unitarias al código, evaluaciones relacionadas con el comportamiento mediante las pruebas de carga y estrés. Además de métricas para la validación del diseño.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA DEL MÓDULO DE ADMINISTRACIÓN DEL SISTEMA AUTOMATIZADO DE MONITOREO Y CONTROL DE LOS SERVICIOS (SAMOS) DEL CENTRO DE SOPORTE UCI.

Introducción

En el presente capítulo se exponen conceptos fundamentales sobre el monitoreo y control de servicios de las TI. Se efectúa un estudio del estado del arte relacionado con soluciones informáticas existentes para la GSTI. Además de una valoración de las tecnologías actuales utilizadas en la propuesta de solución. Se aborda distintos aspectos teóricos para la investigación, donde se encuentra el análisis de los estándares, herramientas y metodología de desarrollo para la construcción de la aplicación así como sus características fundamentales.

1.1. Monitoreo y control de servicios de las TI

En un organismo o institución es necesario un conjunto de actividades que permitan verificar si el proyecto va marchando según lo planificado. Con la evolución de la ciencia y las tecnologías el desarrollo de la informática ha ido incrementando considerablemente, como resultado de este avance los proyectos de desarrollo de software se han ido acrecentando a gran escala. Para lograr el éxito de estos proyectos, con la calidad requerida por los clientes, es necesario vigilar el correcto desarrollo de las tareas establecidas en el proyecto, mediante el monitoreo y control de los recursos que se disponen en su desarrollo.

Los objetivos del monitoreo y control de servicios de las TI son llevar el registro de los principales procesos, reunir evidencias concretas a cerca de los logros y permitir medidas correctivas. El beneficio que se obtiene de este proceso, es que se puede averiguar la situación inicial de cada proceso, se realiza de forma permanente e incluye datos cuantitativos y valorativos. Debido a que la definición de monitoreo y control se abordará en todo el camino de la investigación y por su gran importancia para la solución del problema científico planteado, a continuación se ilustran varios conceptos de este término de acuerdo a las posiciones de aquellos que la estudian y desarrollan.

1.1.1. Conceptos asociados al monitoreo y control de los servicios de las TI

Monitoreo:

- ✓ Monitoreo es una evaluación continua de una acción en desarrollo. Es un proceso interno coordinado por los responsables de la acción. El sistema de monitoreo debe ser integrado en el trabajo cotidiano (Tierra del Futuro, las Golondrinas – América Latina y UBV, 2013).
- ✓ Monitoreo es el proceso sistemático de recolectar, analizar y utilizar información para hacer seguimiento al progreso de un programa en pos de la consecución de sus objetivos, y para guiar

las decisiones de gestión. El monitoreo generalmente se dirige a los procesos en lo que respecta a cómo, cuándo y dónde tienen lugar las actividades, quién las ejecuta y a cuántas personas o entidades beneficia (Luccaco, 2012).

Se puede definir a Monitoreo como un conjunto de tareas para la observación y obtención de datos de un proyecto, acción o proceso con el objetivo de asegurar un correcto funcionamiento.

Definiciones del Control más importantes por Autor:

- ✓ Según la definición dada por Henry Fayol el control consiste en verificar si todo ocurre de conformidad con el plan adoptado, con las instrucciones emitidas y con los principios establecidos. Tiene como fin señalar las debilidades y errores a fin de rectificarlos e impedir que se produzcan nuevamente (Fayol, 1916).
- ✓ Robert *C*. Appleby define a Control como la medición y corrección de las realizaciones de los subordinados con el fin de asegurar que tanto los objetivos de la empresa como los planes para alcanzarlos se cumplan económica y eficazmente (Robert, 1994).
- ✓ Chiavenatto por su parte establece que el control es una función administrativa: es la fase del proceso administrativo que mide y evalúa el desempeño y toma la acción correctiva cuando se necesita. De este modo, el control es un proceso esencialmente regulador (Chiavenatto, 2000).

De las definiciones antes planteadas se ha seleccionado lo planteado por Chiavenatto para tratar al control como un factor importante en la gestión, donde se puede describir como un grupo de inspecciones o revisiones de un sistema con el fin de obtener información que permita identificar errores para luego corregirlos y permitir verificar el cumplimiento de sus principios. La función administrativa que hace parte del proceso administrativo junto con la planeación, organización y liderazgo.

Monitoreo y control:

- ✓ El monitoreo y control de proyectos permite seguir el desempeño del proyecto en cada paso de su ejecución, de forma que se pueda identificar los posibles problemas oportunamente y adoptar las acciones correctivas que permitan mantener al proyecto dentro de los límites establecidos en las líneas base: Alcance, Costo, Tiempo y Calidad (IST, 2009).
- ✓ Las actividades de monitoreo y control son referidas en el Grupo del Proceso de Seguimiento y Control, compuesto por aquellos procesos requeridos para dar seguimiento, analizar y regular el progreso y el desempeño del proyecto, para identificar áreas en las que el plan requiera cambios y para iniciar los cambios correspondiente (PMI, 2008).

Tomando en cuenta las definiciones expuesta anteriormente se puede concluir que el monitoreo y control es un conjunto de procesos que controlan el avance de los proyectos en ejecución, mide los resultados reales, en función de lo planeado y analiza el comportamiento de los indicadores de desempeño.

1.2. Principales estándares para la mejora de servicios TI.

Un correcto monitoreo y control de los procesos de una empresa representa un grupo de problemas relacionados con la gestión de las TI, principalmente en el sentido de cómo lograr que conlleven a una ventaja para la organización, de manera que sean una inversión con retorno y no solamente un gasto necesario. Es por ello que se han creado en la industria diversos marcos de trabajo y mejores prácticas que buscan eliminar estas problemáticas, donde podemos mencionar al ISO/IEC[5] 20000, ISO 9001, CMMI-SVC[6], ITIL[7] y COBIT[8].

Cabe resaltar entre los estándares antes mencionados a ITIL, como el estándar que describe los procesos necesarios para administrar el área de las TI eficazmente, con el fin de optimizar beneficios y garantizar la integración de los servicios. Mencionar al estándar CMMI como una forma de medir el grado de madurez de las organizaciones respecto a la aplicación de las mejores prácticas de desarrollo y gestión del software. La norma británica ISO / IEC 20000 por promover la adopción de un modelo de procesos integrados destinado a mejorar la eficacia en la prestación de los servicios tecnológicos y establecer las directrices para una GSTI de calidad, por otra parte se encuentra COBIT por ser el estándar generalmente aceptado que brinda buenas prácticas para gestión y control de las TI, además de ser ideal como marco de trabajo de negocios para el gobierno y la administración dentro de las empresas. En la presente investigación el marco de trabajo COBIT ha sido definido por el centro de soporte UCI para la construcción del sistema que dará solución a la problemática planteada.

1.2.1. Objetivos de Control para Tecnología de Información y Tecnologías relacionadas.

COBIT se creó con las características principales de ser orientado a negocios, orientado a procesos, basado en control e impulsado por mediciones. Se encuentra dirigido a proporcionar la información que la empresa requiere para lograr sus objetivos. COBIT permite que la empresa administre y controle los recursos de las TI, usando un conjunto estructurado de procesos que ofrezcan los servicios requeridos de información y ofrece herramientas para garantizar la alineación con los requerimientos del negocio. (ver Anexo 1).

[5] Organización Internacional de Normalización / Comisión Electrotécnica Internacional, significado de sus siglas en español.
[6] Modelo de Madurez de la Capacidad Integrado, significado de sus siglas en español.
[7] Biblioteca de Infraestructura de Tecnologías de Información, significado de sus siglas en español.
[8] Objetivos de Control para Tecnología de Información y Tecnologías relacionadas.

COBIT define las actividades de las TI en un modelo de 34 procesos genéricos agrupados en 4 dominios:

Planear y Organizar: Identifica las estrategias y tácticas para que las TI puedan contribuir al logro de los objetivos del negocio.

Adquirir e implementar: Se basa en la identificación de soluciones, desarrollo o adquisición, cambios y/o mantenimientos de sistemas existentes.

Entregar y Dar Soporte: Cubre la entrega de los servicios requeridos. Incluye la prestación del servicio, la administración de la seguridad y de la continuidad, el soporte del servicio de los usuarios, la administración de los datos y de las instalaciones operacionales.

Monitorear y Evaluar: Todos los procesos de las TI deben evaluarse de forma regular en el tiempo en cuanto a su calidad y cumplimiento de los requerimientos de control. Este dominio abarca la administración del desempeño, el monitoreo de control interno, el cumplimiento regulatorio y la aplicación del gobierno.

La estructura del modelo de COBIT propone un marco de acción donde se evalúan los criterios de información y finalmente se realiza una evaluación sobre los procesos involucrados en la organización. Por otra parte COBIT se encuentra basado en una revisión crítica y analítica de las actividades en las TI, además de estar alineado con estándares de control y auditoría.

La misión principal de COBIT es investigar, desarrollar, hacer público y promover un marco de control de gobierno TI autorizado, actualizado, aceptado internacionalmente para la adopción por parte de las empresas y el uso diario por parte de gerentes de negocio, profesionales de las TI y profesionales de aseguramiento (OverTI, 2011). El valor, el riesgo y el control constituyen la esencia del Gobierno TI por esta razón es que las buenas prácticas de COBIT están enfocadas fuertemente en el control y menos en la ejecución. Estas prácticas asegurarán la entrega del servicio y brindarán un patrón de medición con el cual se podrá calificar cuando las operaciones de cualquier institución se desarrollen de forma incorrecta.

Una necesidad básica de toda empresa es entender el estado de sus propios sistemas de las TI. La obtención de una visión objetiva del nivel de desempeño propio de una empresa no es sencilla, por ello COBIT atiende estos temas a través de (OverTI, 2011):

- ✓ Modelos de madurez que facilitan la evaluación y la identificación de las mejoras necesarias en la capacidad
- ✓ Metas y mediciones de desempeño para los procesos de las TI, que demuestran cómo los procesos satisfacen las necesidades del negocio y de las TI

- ✓ Metas de actividades para facilitar el desempeño efectivo de los procesos

La última versión del estándar COBIT, versión 5, aparece en el año 2012, esta versión incorpora las últimas ideas en la gobernanza empresarial y técnicas de gestión, proporciona principios globalmente aceptados, prácticas, herramientas analíticas y modelos para contribuir a incrementar el valor de los sistemas de información, se basa en cinco principios claves (ISACA, 2013):

- ✓ **Principio 1:** Satisfacer las necesidades de las Partes Interesadas.
- ✓ **Principio 2:** Cubrir la organización de principio a fin. Integrando el Gobierno corporativo con el Gobierno de las TI. Orientación al negocio.
- ✓ **Principio 3:** La aplicación de un único marco de trabajo integrado.
- ✓ **Principio 4:** Habilitación de un enfoque holístico. Para conseguir una Gestión y Gobierno de las TI con eficiencia y eficacia.
- ✓ **Principio 5:** La separación de la Gestión de Gobierno.

El marco de trabajo de control COBIT brinda distintos beneficios que ayudan a que el negocio logre grandes resultados donde podemos encontrar los siguientes elementos (AEC, 2013):

- ✓ Establece un vínculo con los requerimientos del negocio.
- ✓ Organiza las actividades de las TI en un modelo de procesos.
- ✓ Identifica los principales recursos de las TI.
- ✓ Optimiza el costo de los servicios de las TI y las tecnologías.
- ✓ Ofrece información más oportuna y de mayor calidad.
- ✓ Define los objetivos de control gerenciales.
- ✓ La creación de un lenguaje común a todos los involucrados en el control de los procesos.
- ✓ Las auditorías serán más eficientes y exitosas.

Luego de realizar un análisis de las principales características de COBIT, se puede mencionar que con su uso en la solución desarrollada permite eliminar un grupo de anomalías que cuenta el centro de soporte UCI, entre ellas la falta de conocimiento del estado de los servicios de soporte técnico. Para esto fue necesario usar varios de los procesos que forman parte de los dominios de COBIT.

Dentro del dominio Entregar y Dar Soporte se utilizó el proceso:

Garantizar la seguridad de los sistemas: su principal objetivo es mantener la integridad de la información y de la infraestructura de procesamiento y minimizar el impacto de las vulnerabilidades e incidentes de seguridad.

Por otro lado en el dominio Monitorear y Evaluar se centró en los procesos:

Reportes al consejo directivo y a ejecutivos: se encuentra dirigido a proporcionar reportes administrativos para ser revisados por la alta dirección sobre el avance alcanzado, específicamente en términos del desempeño.

Revisiones de Auditoría: permite monitorear y reportar la efectividad de los servicios por medio de revisiones de auditoría.

Después de las consideraciones anteriores se puede definir a COBIT como un elemento clave que brinda una guía dirigida a los directivos y la administración en general del centro de soporte UCI. Convirtiendo a la solución desarrollada en un sistema de gran ayuda para la toma de decisiones y el seguimiento de las actividades del negocio.

1.3. Sistemas informáticos para el monitoreo y control.

Como parte del desarrollo tecnológico que vive el mundo en los últimos años, los productos informáticos evolucionan dinámicamente para poder adaptarse a las necesidades tecnológicas actuales, los sistemas de monitoreo y control no han quedado exentos ante esta tendencia. Dando lugar a que los viejos y rigurosos mecanismos de trabajo, así como los arduos procesos de gestión existentes durante décadas en las empresas, sean sustituidos por eficientes sistemas informáticos capaces de gestionar esos procesos.

Son varios los países que cuentan con sistemas informáticos con características relacionadas con el monitoreo y control de los servicios. Estos sistemas tienen como objetivo analizar y regular el progreso, así como el desempeño de los negocios, para identificar áreas en las que el plan requiera cambios y para iniciar los cambios correspondientes, a continuación se realiza la descripción de varios sistemas de este tipo.

1.3.1. ORCA GRC SUITE

ORCA GRC Suite, es una Plataforma de Software altamente configurable diseñada para optimizar la operación de los procesos de Administración de Riesgos, Cumplimiento Normativo, Seguridad y Gobierno de las TI, con el fin de mejorar la eficiencia operativa del negocio y su postura ante el riesgo (ISEC, 2013). La diversidad de soluciones en prevención de riesgos que ofrece ORCA y el respaldo de expertos multidisciplinarios para la entrega de servicios consultivos le permite diferenciarse de la mayoría de sus competidores dando como resultado la optimización operativa, la reducción de costos y la simplificación de las operaciones, así como el mejoramiento de la visibilidad y la toma de decisiones para propiciar un Gobierno Corporativo más eficiente (GCP Global, 2013).

ORCA puede adaptarse prácticamente a cualquier tipo de medio relacionado con verificaciones de seguridad TI o análisis de riesgo muy específicos. Diseñada especialmente para optimizar la operación de los procesos de Administración del Negocio. Permite el aumento en la eficiencia de sus funciones,

optimizando grandemente su posición con relación al riesgo operativo, legal, financiero y de reputación (ver Anexo 2).

1.3.2. REMEDY IT SERVICE MANAGEMENT DE BMC

Remedy IT Service Management de BMC es un software para establecer, operar, medir y optimizar procesos de servicio de las TI de manera repetible y eficiente para garantizar un mejor servicio a los usuarios. Incluye aplicaciones para la automatización de procesos basados en las mejores prácticas de ITIL. Remedy IT Service Management de BMC unifica los procesos, manejo de incidentes, manejo de problemas, administración de cambios, manejo de activos y administración de acuerdos de servicio (SLAs), así como el manejo de una CMDB (base de datos para administración de configuraciones) en un único modelo de datos y con una única interfaz de usuario (Arión, 2013).

La suite de aplicaciones BMC Remedy IT Service Management incluye cuatro aplicaciones de vanguardia: BMC® Remedy® Service Desk, BMC® Remedy® Asset Management, BMC® Remedy® Change Management y BMC® Service Level Management. Las cuatro comparten la base de datos de configuración CMDB BMC® Atrium™, incluida para coordinar los procesos en torno a una sola concepción del modo en que los componentes tecnológicos sustentan los servicios de la empresa. Todas se basan en BMC® Remedy® Action Request System® (AR System®), la plataforma de gestión de procesos líder del mercado.

Remdy IT Service Management de BMC proporciona un medio común para automatizar e integrar los procesos de soporte a servicios entre áreas funcionales, grupos regionales, recursos de terceros y de otras partes de la organización. Mejora la disponibilidad del servicio, la calidad, el costo y la efectividad de los ambientes complejos de las TI en grandes organizaciones. BMC Remedy Service Desk también proporciona un flujo de trabajo común y motor de base de datos para proporcionar visibilidad de los principales parámetros de servicios de las TI en toda la organización en un solo paquete integrado. Actúa como un único punto de contacto para todos los usuarios, permite acelerar el restablecimiento de la normalidad de servicio y ayuda a prevenir futuros eventos de negocio que impacten negativamente en los servicios, al tiempo que se contribuye a mejorar la eficacia del personal de las TI (ver Anexo 3).

1.3.3. NAGIOS XI

Nagios XI es un Sistema de Monitoreo Enterprise-class que permite a las organizaciones tener una visión y control de la infraestructura de TI, antes que los problemas afecten a los procesos críticos del negocio. Proporciona el monitoreo de todos los componentes de la infraestructura de misión crítica incluyendo aplicaciones, servicios, sistemas operativos, protocolos de red, métricas de sistemas e infraestructura de red. Interfaz gráfica que permite a cada usuario una personalización adaptable a sus necesidades. Contiene alertas que se programan y se envían a las personas claves, a través de correo

electrónico o mensajes de texto móviles, proporcionándoles los detalles para que puedan comenzar a resolver los problemas inmediatamente (S.R.L., 2013) (ver Anexo 4).

1.3.4. GESPRO

Es una herramienta para el desarrollo y la innovación en Gestión de Proyectos. Es desarrollado por el Laboratorio de Investigaciones en Gestión de Proyectos de la UCI de Cuba. Se presenta como un modelo de negocios relacionado con servicios donde se combina el uso de una solución informática para la gestión de proyectos y un sistema de formación especializada en gestión de proyectos. Esta combinación posibilita no sólo la informatización de la gestión de proyectos en las organizaciones, sino también la mejora continua de sus procesos de planificación, seguimiento y control.

GESPRO es un sistema elaborado con estupendas características de planificación, gestión de recursos humanos y sus competencias, enfocado solamente para los proyectos productivos, es una solución informática desarrollada bajo un ecosistema de *software* basado en tecnologías libres, que permite:

- ✓ La planificación, control y seguimiento de los proyectos y sus recursos asociados, alineados con la proyección estratégica de las organizaciones.
- ✓ La planificación del alcance y el tiempo, la gestión de recursos humanos y sus competencias, la gestión de riesgos, así como la financiera de los proyectos. Gestión logística y gestión de recursos compartidos.
- ✓ El control y seguimiento de proyectos a través de la combinación de un cuadro de mando integral y un sistema para el diseño dinámico de reportes, que permiten el acceso a la información del estado de los proyectos con diferentes niveles de detalles de la información.
- ✓ Gestión documental con facilidades para la gestión del expediente de los proyectos.
- ✓ Gestión de contratos y de interesados en los proyectos, que permite tanto la gestión de acuerdos con clientes como con los proveedores y garantiza integrar el control y el seguimiento de los compromisos alineados completamente con la información del estado de los proyectos.

1.3.5. Definición de la herramienta de monitoreo y control para el centro de soporte UCI.

A partir de la información obtenida sobre el proceso de monitoreo y control de servicios del centro de soporte UCI, se realizó un análisis con los sistemas anteriormente mencionados, en aras de conocer cómo estos gestionan sus recursos y determinar si era factible su utilización. El análisis arrojó como resultado que ninguno es viable para utilizarlo en el centro de soporte UCI, debido a una serie de inconvenientes encontrados que se exponen a continuación.

Tabla 2. Criterios comparativos a evaluar.

Sistema-Criterio	ORCA GRC SUITE	REMEDY	NAGIOS XI	GESPRO
Gestión de Riesgos.	✓	✓	✓	✓
Gestión de Auditorías.	✓	✓	✓	✓
Gestión de Roles.	✓	✓	✓	✓
La utilización de gráficas.	✓	✓	✓	✓
La generación de reportes.	✓	✓	✓	✓
Tecnologías libres.	-	-	-	✓
Envío de notificaciones.	-	-	✓	✓
Altos aspectos organizacionales.	✓	-	✓	✓
Guía para directivos.	✓	✓	✓	✓
Formas de recopilar información del cliente.	-	-	-	✓
Enfocado en servicios de soporte técnico.	-	-	-	-

- ✓ Los sistemas **ORCA GRC SUITE**, **REMEDY IT SERVICE MANAGEMENT DE BMC**, **NAGIOS IX** y **GESPRO**, poseen módulos a través del cual se puede gestionar riesgos, gestionar auditorías y realizar la gestión de roles, pero no se encuentran enfocados a los servicios de soporte técnico.

- ✓ Los sistemas **ORCA GRC SUITE**, **REMEDY IT SERVICE MANAGEMENT DE BMC**, **NAGIOS IX** y **GESPRO** brindan una guía para los directivos y la administración en general de las instituciones para la toma de decisiones y el seguimiento de las actividades que intervienen en sus procesos, pero el sistema **REMEDY IT SERVICE MANAGEMENT DE BMC** responde a niveles bajos de organización.
- ✓ Los sistemas **ORCA GRC SUITE, REMEDY IT SERVICE MANAGEMENT DE BMC** y **NAGIOS IX** se ejecutan sobre tecnologías privativas, lo cual impide su utilización ya que el centro de soporte UCI tiene como política el uso de tecnologías libres, además que se requiere permisos expresos de los titulares para el uso, redistribución o modificación.

Sin embargo su análisis permitió identificar varias funcionalidades importantes para el monitoreo y control de servicios de soporte técnico:

- ✓ Los sistemas **ORCA GRC SUITE, REMEDY IT SERVICE MANAGEMENT DE BMC**, **NAGIOS IX** y **GESPRO**, brindan gráficas que ayudan a la toma de decisiones de los directivos.
- ✓ Los sistemas **ORCA GRC SUITE, REMEDY IT SERVICE MANAGEMENT DE BMC, NAGIOS IX** y **GESPRO** generan detallados reportes, los cuales le permiten al cliente saber que puede o que está yendo mal en su organización.
- ✓ El sistema **GESPRO** permite la aplicación de encuestas, tanto a los recursos humanos como a los interesados, para obtener opiniones de los usuarios a cerca del funcionamiento del sistema para mejorar el negocio.

Por todas estas razones existe la necesidad de implementar un sistema que permita el monitoreo y control de servicios de soporte técnico. Además de que no utilicen y se ejecuten sobre tecnologías privativas. Razones que impulsaron al centro de soporte al desarrollo del sistema, empleando el modelo de desarrollo y tecnologías definidas así como el marco de trabajo Grails sobre el cual será desarrollado el sistema. De esta forma se estaría desarrollando un entorno propio con características específicas para un producto que responda satisfactoriamente a las necesidades del cliente.

1.4. Herramientas, tecnologías y metodología para el desarrollo del software.

Para el desarrollo del sistema se representan las principales características de varias herramientas y tecnologías que posibilitan su creación. A continuación se presenta una breve descripción:

1.4.1. Metodología de desarrollo de software.

Se entiende por metodología de desarrollo a una colección de métodos, coherentes entre si y que siguen una filosofía o enfoque de desarrollo de software. Entre otros factores aplicados a lo largo del ciclo de vida de desarrollo de software, que deben ser seleccionadas y adaptados de acuerdo al

tamaño, la experiencia del personal y el tiempo estimado, además indica cómo hay que obtener los distintos productos parciales y finales.

Existen diferentes metodologías de desarrollo las cuales se pueden clasificar en robustas y ágiles, que se pueden aplicar a diferentes proyectos, teniendo en cuenta el flujo de información con que se trabaja. Dentro de las robustas se encuentran: Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), entre otras y en el grupo de las ágiles: Extreme Programming (XP), SCRUM, Cristal Methodologies, entre otras. En la presente investigación se define como metodología de desarrollo a XP, por ser una exigencia del centro de soporte UCI para la implementación del sistema SAMOS.

Metodología XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Define como fases fundamentales: exploración, planificación, diseño, implementación y pruebas, se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas. Es una Metodología que se basa en reutilización del código desarrollado, especialmente adecuado para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Don Wells, 2013).

Esta metodología está concebida para proyectos con poco tiempo de desarrollo, equipos pequeños y con pocos roles. Propone que el diseño debe ser sencillo, que funcione con todas las pruebas y con el menor número de clases y métodos posible. Todas estas características contribuyeron a que la solución desarrollada mantuviera una programación organizada y con una taza pequeña de errores.

1.4.2. Herramientas para el modelado.

Visual Paradigm

Visual Paradigm es una herramienta UML[9] profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar, actualizar y compatible entre ediciones.

En la solución desarrollada se utilizó el Visual Paradigm en su versión 8.0, facilitando el modelado del diagrama conceptual y diagrama de despliegue, así como la representación de patrones de diseño,

[9] Unified Modeling Language.

mediante el soporte de UML que proporciona esta herramienta, lo que aportó un mejor entendimiento de conceptos claves para el negocio en cuestión. Por las características antes mencionadas se decidió utilizar Visual Paradigm, además que por políticas del centro de soporte es la herramienta que se utiliza para el modelado del sistema SAMOS.

1.4.3. Marco de Trabajo.

Grails

Para la solución desarrollada se utilizó Grails debido, entre otras razones, que posibilita la realización de aplicaciones complejas en cuestión de días, cuando se puede tardar meses. Un cambio de requisitos no obliga a reescribir toda la aplicación, porque la mayoría del código de infraestructura se genera de forma dinámica. Además que por política del centro de soporte, es el marco de trabajo de desarrollo que se utiliza para la implementación del sistema SAMOS.

Otras de las características que se aprovecharon en el desarrollo de la solución, son las siguientes: es un marco de trabajo de aplicaciones web, en el que se usa principalmente el lenguaje de Java y Groovy, incluso se puede mezclar el código. Es un marco de trabajo que funciona bajo un modelo conocido como MVC o “modelo vista controlador”, en el que principalmente lo que se hace es crear “Controladores” que son como servicios que manipulan la aplicación web, y todo el código de estos controladores es ejecutado en el servidor web. Está construido sobre sólidos proyectos de código abierto, como Spring, SiteMesh, GORM/Hibernate, todo esto corriendo sobre la Máquina Virtual de Java.

Apunta a hacer el desarrollo tan simple como sea posible. No centra su desarrollo en reinventar lo bueno, sino mejorarlo, trata de tomar lo mejor de las tecnologías y adaptarlas mucho más eficientes. Cuenta con Hibernate que es un buen ORM[10], con poderosas y avanzadas herramientas. Grails crea un simple DSL[11], que simplifica el trabajo con Hibernate, dejando fuera las complicadas configuraciones de mapeo en archivos XML. (Beginning, 2009).

1.4.4. Lenguaje de programación.

Groovy

Para el desarrollo del Módulo de Administración del SAMOS se emplea groovy ya que es el utilizado en el marco de trabajo Grails. Además de contar con características que facilitaron el trabajo, entre ellas

[10] Mapeo Objeto-Relacional (más conocido por su nombre en inglés, Object-Relational mapping).
[11] tipo de lenguaje de programación o lenguaje de especificación en el desarrollo de software y la ingeniería de dominio dedicado a un dominio particular problema, una técnica de representación problema particular, y / o una técnica de solución particular.

mencionar que es un lenguaje dinámico para la JVM[12]. También puede acceder directamente a todas las API[13] existentes en Java.

Groovy es un lenguaje de programación que puede ser utilizado para combinar módulos de Java, ampliar las aplicaciones existentes de Java o escribir nuevas aplicaciones. Groovy tiene una sintaxis similar a Java y funciona a la perfección con el bytecode[14] de Java. Muchas de las características del lenguaje se parecen a las de Perl, Python, Ruby y Smalltalk. Por otro lado, comparte el mismo modelo de objetos, de hilos y de seguridad del lenguaje Java. (Christopher M. Judd, 2008).

1.4.5. Entorno Integrado de Desarrollo.

Intellij IDEA

Es un entorno inteligente para desarrollar aplicaciones Java, cliente y servidor, también permite desarrollar aplicaciones para móviles (J2ME). Posee un avanzado editor de código, compatible con multitud de tecnologías (AJAX, JSP, EJB) y dentro de un mismo entorno. En la solución desarrollada se utilizó para el análisis del código, compilación/ejecución/debugging, para control de versiones, detección de duplicaciones, análisis de dependencias y como soporte para plugins. Se decidió utilizar este IDE[15] debido a que es el más optimizado para programar en Grails, además que por políticas del centro de soporte es el entorno de desarrollo que se utiliza para la implementación del sistema SAMOS.

1.4.6. Sistema Gestor de Base de Datos.

PostgreSQL

Como Sistema Gestor de Base de Datos (SGBD) se usó la versión 9.1 de PostgreSQL en la solución desarrollada, ya que por política del Centro Soporte, es el SGBD que se utiliza para el desarrollo del sistema SAMOS. Se usó en la ejecución de consultas complejas y uniones de gran tamaño. Por otro lado permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo.

Es una herramienta orientada a objetos, de bajo coste y multiplataforma. Potente gestor de base de datos relacional libre (liberado bajo licencia BSD[16]). Cuenta con más de 15 años de desarrollo activo y arquitectura probada que ha ganado mucha reputación dada su confidencialidad e integridad en los datos (The PostgreSQL Global Development Group, 2013).

[12] Java virtual machine significado de sus Siglas en español es Máquina Virtual de Java.
[13] Interfaz de programación de aplicaciones (IPA) o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.
[14] Archivo binario que contiene un programa ejecutable similar a un módulo objeto.
[15] Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.
[16] Berkeley Software Distribution, licencia de software libre que permite el uso del código fuente en el software privativo.

1.4.7. Servidor web.

Apache Tomcat

Es un servidor web que se desarrolla en un entorno abierto y participativo publicado bajo la licencia Apache versión 2. Está integrado en la implementación de referencia Java 2 Enterprise Edition (J2EE)[17] de Sun Microsystems. Actualmente es el servidor Web más utilizado a la hora de trabajar con Java en entornos web.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad (Genos Cloud Services S.L, 2013).

Apache Tomcat es uno de los más utilizados actualmente para el despliegue de las aplicaciones desarrolladas por la universidad, primero porque es gratis y además fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java, es muy fiable por la solidez que le brindan los miles de desarrolladores que contribuyen a su estabilidad.

En cada una de las aplicaciones que se despliegan surge la necesidad de dominar este tipo de tecnologías, específicamente para implementar clústeres de servidores de aplicaciones Java, en este caso de Apache Tomcat para lograr una alta disponibilidad y rendimiento de los sistemas. Por las características antes mencionadas se decidió utilizar Apache Tomcat como servidor web.

1.4.8. Biblioteca de componentes.

Jquery UI

Jquery UI es una biblioteca de componentes para el marco de trabajo Jquery que le añaden un conjunto de plugins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de Jquery (WEB, 2013). Esta biblioteca permitió implementar componentes diversos en la solución desarrollada, para generar interfaces de usuario, además de otras funcionalidades básicas, como su propio nombre indica, está basado en el popular marco de trabajo JavaScript.

[17] Plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

1.4.9. Componentes de diseño

Bootstrap

Bootstrap brinda una base pre-codificada de HTML[18] y CSS[19] para armar el diseño de una página o una aplicación web, y al ofrecerse como un recurso de código abierto es fácil de personalizar y adaptar a múltiples propósitos. Permite crear interfaces que se adapten a los diferentes navegadores, tanto escritorio como tablets y móviles a distintas escalas y resoluciones. Se integra perfectamente con las principales librerías JavaScript, por ejemplo jQuery. Es un marco de trabajo ligero que se integra de forma limpia. Funciona con todos los navegadores, incluido Internet Explorer (Bootstrap, 2012).

Es un conjunto de herramientas de interfaz que permitieron realizar la solución desarrollada, que simplificó el proceso de creación de diseño combinando CSS y JavaScript. Permitió crear interfaces que se adaptan a los distintos navegadores. Además de utilizar formularios, botones, tablas, cuadrículas, menús y muchas otras herramientas para acelerar el trabajo.

Highcharts

Highcharts permite añadir gráficas animadas a un sitio web en una gran variedad de formatos (spline, área, barra, columna, pastel, entre otras). Está implementado en JavaScript por lo que es sencillo de incorporar a un sitio web. Es compatible con cualquier navegador que soporte JavaScript, por lo que funciona en Chrome, Firefox, Safari, Internet Explorer ya sea con Linux, Windows, OSX, Android, entre otros (Leyva, 2012).

Se usó en la solución desarrollada para añadir gráficos interactivos para un mejor análisis de los datos del negocio. Pues una de las características clave de Highcharts es que en cualquiera de las licencias libres o no, se le permite descargar el código fuente y hacer nuevas ediciones, permitiendo así modificaciones personales y una gran flexibilidad.

1.5. Conclusiones parciales.

Mediante la elaboración del marco teórico y el estudio del estado del arte se permitió sentar las bases para el desarrollo de la investigación identificando conceptos fundamentales asociados al monitoreo y control. Así como herramientas y metodologías acordes a las necesidades de la problemática planteada. De esta forma se demostró la necesidad de desarrollar una serie de funcionalidades de apoyo al monitoreo y control de servicios de soporte técnico en el centro de soporte UCI.

[18]Lenguaje de marcado hipertextual para la elaboración de páginas web.
[19]Hojas de estilo en Cascada

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se hace una propuesta general del sistema a desarrollar, después de haber realizado un análisis crítico de los procesos de negocio que tienen lugar en el centro de soporte de la Universidad. Se desarrolla la identificación de los requisitos que son capacidades y condiciones con las cuales debe ser conforme el sistema. Estos comprenden necesidades de información y control, funcionalidad del producto y comportamiento, rendimiento general del producto, diseño, restricciones de la interfaz y otras necesidades especiales.

Se realiza una descripción detallada de la solución propuesta utilizando XP como metodología de desarrollo donde se definen tres de sus fases fundamentales: exploración, planificación y diseño. Se describen las Historias de Usuario (HU) en la fase de Exploración, las cuales muestran los requisitos funcionales del sistema definidos por el cliente y es el primer artefacto generado por la metodología a utilizar. Luego se desarrolla la fase de Planificación, la cual lleva a cabo la estimación del esfuerzo para cada una de las HU y el plan de duración de cada iteración, también en esta fase se define el plan de entregas con las propuestas de liberación de las versiones de la aplicación. Para el diseño del sistema se describe la arquitectura del módulo, los patrones de diseño que se utilizaron y la descripción en tarjetas Clase-Responsabilidad-Colaboración (CRC).

2.1. Propuesta del sistema.

La aplicación desarrollada tiene como objetivo facilitar el monitoreo y control de servicios que se brindan en el centro de soporte UCI. Cuenta con una serie de funcionalidades entre las que se encuentran la gestión de servicios, la cual será una funcionalidad para posibilitar a los especialistas crear servicios a sus clientes, la cual tendrá asociada otras funcionalidades necesarias para el desarrollo del trabajo tales como la gestión de auditorías y gestión de riesgos. Las dos últimas funcionalidades mencionadas fueron integradas a la solución desarrollada, ya que antes se encontraban construidas en dos subsistemas que no resolvían los problemas que presenta el centro de soporte UCI, siendo necesarias para el flujo de trabajo del centro para poder llevar la gestión de los servicios de soporte técnico. Con la solución desarrollada se pueden realizar auditorías detectando riesgos, que serán mitigados por actividades asociadas a la funcionalidad de gestión de riesgos.

Cuenta con otras funcionalidades que ayudarán a la toma de decisiones y una mejor calidad de los servicios prestados como es la generación de reportes, gráficos personalizados por la organización y

anexo de cualquier tipo de documento (textos, plantillas, presentaciones, gráficos e imágenes). Además de permitir la realización de encuestas de satisfacción y evaluar sus resultados.

Para garantizar la seguridad de la información del sistema proporciona funcionalidades que garantizan la disponibilidad, el acceso rápido, confidencial y simultáneo a la información por parte de los involucrados según el nivel de privilegios. Se autenticará usuarios a través del LDAP[20] y Vía Local, además de permitir la personalización de los perfiles de usuarios y el control de las operaciones que se realicen en el sistema.

2.2. Modelo conceptual.

El modelo conceptual es una técnica usada para la representación gráfica del conocimiento. Se puede describir como una red de conceptos, donde los nodos representan los conceptos, y los enlaces representan las relaciones entre los conceptos significativos para el área que se analice. El objetivo del modelo conceptual es describir el contenido de la información de la base de datos, y no las estructuras de almacenamiento que se necesitarán para manejar esta información. Permite además identificar, organizar y realizar razonamientos sobre los componentes y comportamientos del sistema, siendo la guía para el proceso de diseño del sistema informático, pudiéndose usar además como referencia para evaluar un diseño particular y razonar sobre la solución realizada.

La propuesta solución diseñada en el modelo conceptual de la figura 1 se muestra la vista estructural del sistema, donde se describen las entidades implicadas en el proceso de monitoreo y control de servicios de TI, la cual representa un sistema que debe permitir gestionar servicios, auditorías y riesgos. Por otro lado los usuarios contarán con el acceso a cada una de las funcionalidades mediante la gestión de roles. Se brindará servicios a un cliente y las auditorías se aplicarán a un servicio. El sistema contará con varios grupos auditores, donde cada uno estará compuesto por uno o varios usuarios certificados por una entidad certificadora y un grupo auditor será asignado a una o varias auditorías.

Una de las funciones de las auditorías será la identificación de riesgos, donde los usuarios que integran el grupo auditor tendrán la opción de enviar notificaciones con la información necesaria para la gestión de los riesgos identificados. Las notificaciones se enviarán al subsistema de gestión de riesgos, específicamente a usuarios con privilegios necesarios para realizar la gestión de riesgos. Los riesgos gestionados se le asignarán actividades de respuesta las cuales mitigarán, aceptarán o evitarán a uno o varios riesgos de acuerdo a sus características. Además, el sistema permitirá que los usuarios realicen encuestas de satisfacción con el objetivo de ayudar a la toma de decisiones.

[20] Protocolo Ligero de Acceso a Directorios.

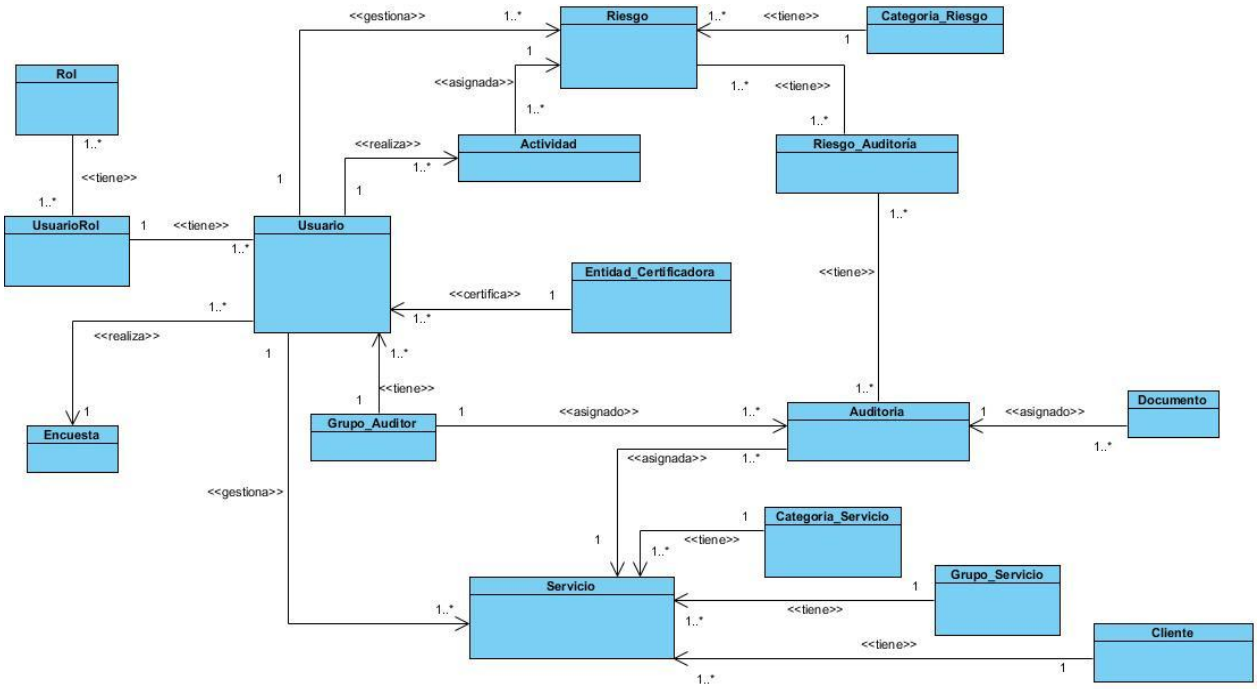

Figura 1. Modelo Conceptual.

2.3. Roles y Responsabilidades.

Para desarrollar un proyecto, sus ejecutores suelen tener diferentes responsabilidades y derechos a fin de facilitar la organización de las actividades encaminadas al cumplimiento de los objetivos. De igual manera, para el proceso de software las actividades son asignadas con base a la ingeniería del software, de acuerdo a las funciones requeridas en el modelo implementado, estos son los llamados roles en la ingeniería de software. A continuación se describen algunos de los roles que propone la metodología de desarrollo XP en correspondencia a las necesidades del equipo de desarrollo del Módulo de Administración del SAMOS.

Analista y diseñador: Es la persona que se encarga del análisis y el diseño previo que se debe realizar para el desarrollo del software. Este rol es el responsable de establecer el contacto con el cliente para conocer las particularidades del proceso de negocio. Es el encargado junto al cliente de identificar los requisitos funcionales y no funcionales del sistema. Guía a los desarrolladores en la construcción del software, además, de ser la persona que genera el respaldo documental definido por la metodología de desarrollo seleccionada.

Programador: El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Encargado de pruebas (*Tester*): El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Entrenador (*Coach*): Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Encargado de seguimiento (*Tracker*)

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Cliente: En la metodología XP, el cliente junto al analista establece los requisitos del sistema. También escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. Acompaña a cada rol dentro del equipo de desarrollo en todas las fases del software, apoyando el trabajo que se realiza y dando su criterio acerca de los resultados que se obtiene en cada etapa.

Tabla 3. Roles y Responsabilidades.

Rol	Responsabilidad	Nombre
Analista y diseñador	-Realizar el proceso de negocio. -Definir requisitos funcionales y no funcionales. -Generar apoyo documental (Historias de Usuarios, Tareas de Ingeniería). -Diseñar el modelo conceptual y el diagrama de paquetes.	Ariel Ramírez Leyva Iván Horta Wong
Programador	-Conocer las tecnologías a utilizar. -Implementar los requisitos funcionales. -Implementar las funcionales secundarias para automatizar el proceso de negocio establecido.	Ariel Ramírez Leyva Iván Horta Wong
Probador	-Diseñar casos de pruebas. -Ejecutar casos de pruebas.	Ariel Ramírez Leyva Iván Horta Wong
Entrenador	-Representar al equipo de desarrollo. -Guiar y organizar el proceso de desarrollo del software.	Yulio Seriocha García Gallardo Neybis Lago Clara

	-Conocer el proceso de negocio establecido. -Conocer las tecnologías a utilizar.	
Encargado de seguimiento	-Verificar el grado de acierto entre las estimaciones realizadas y el tiempo real del desarrollo del sistema. -Realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables. -Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.	Ariel Ramírez Leyva Iván Horta Wong
Cliente	-Establecer el proceso de negocio. -Definir los requisitos funcionales del sistema. -Apoyar y participar en cada fase del desarrollo del software. -Participar en el proceso de pruebas del software.	centro de soporte UCI

2.4. Técnicas de recopilación de requisitos.

En el proceso de desarrollo del sistema de gestión docente metodológica el equipo de desarrollo se enfrenta a la identificación de requisitos, de manera que resulta importante entender lo que el cliente quiere, analizar sus necesidades, especificar una solución sin ambigüedades y administrar los requisitos conforme a su transformación en un sistema operacional (Pressman, 2010). Para ello existen diversas técnicas de captura de información que facilitan la comunicación y la extracción de los elementos más esenciales del negocio. Las técnicas seleccionadas para realizar la captura de requisitos fueron:

Tormentas de ideas: realizada mediante encuentros del equipo de trabajo, de modo que tanto desarrolladores como clientes aportaron ideas enriqueciendo el conocimiento de los implicados, facilitando la obtención desde diferentes puntos de vista y razonamientos de la información inherente a la investigación.

Mapa conceptual: empleada para favorecer la representación visual de todos los conceptos que intervienen en el negocio, de manera que esta primera entrega contribuya con la creación de la base de datos.

Entrevistas: desarrollada a través de encuentros con el cliente para poder establecer los objetivos del sistema, qué es lo que debe lograr y de qué forma satisfacer las necesidades del negocio.

2.5. Requisitos del sistema.

Los requisitos del software son la descripción de los servicios y restricciones de un sistema, lo que el software debe hacer y bajo qué circunstancias debe hacerlo. La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software.

Por otra parte se encuentra el análisis de requisitos, el cual cumple un papel primordial en el proceso de desarrollo de software, se basa en la comunicación entre los desarrolladores, clientes y usuarios, para definir el nuevo sistema que brinde una solución al problema. Permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software

Requisitos funcionales del sistema.

Los requisitos funcionales del sistema son la condición o capacidad que el usuario necesita para resolver un problema o resolver un objetivo. Es la condición o la capacidad que tiene que ser alcanzada o poseída por un sistema o componente de sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. A continuación se exponen requisitos funcionales del sistema:

Tabla 4.Requisitos funcionales del sistema.

Ítem*	Descripción
RF1	Integración de los subsistemas Gestión de Auditorías y Gestión de Riesgos.
RF2	Gestionar usuario.
RF3	Mostrar usuario.
RF4	Autenticar usuarios a través del LDAP y Vía Local.
RF5	Gestionar Rol.
RF6	Mostrar rol.
RF7	Asignar Roles a los usuarios.
RF8	El sistema debe permitir la personalización de los perfiles de usuarios.
RF9	Gestionar categoría.
RF10	Mostrar Categoría.
RF11	Gestionar grupo.
RF12	Mostrar grupo.
RF13	Gestionar cliente.
RF14	Mostrar cliente.
RF15	Gestionar servicio.
RF16	Mostrar servicio.
RF17	Realizar notificaciones a través de correo a los usuarios.
RF18	Mostrar un gráfico con los modelos de datos de los sistemas que están incluidos en el SAMOS.
RF19	Controlar el histórico de todas las acciones del sistema.
RF20	Realizar búsquedas por parámetros.

RF21	Un panel de control que contenga un resumen de datos de manera gráfica para ayuda a la toma de decisión de los directivos.
RF22	Generación de reportes.
RF23	Permitir anexo de cualquier tipo de documento (textos, plantillas, presentaciones, gráficos e imágenes).
RF24	El sistema debe de permitir la realización de encuestas de satisfacción y evaluar los resultados de las encuestas.

Requisitos no funcionales del sistema.

Los requerimientos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente, están vinculados a requerimientos funcionales, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. En la siguiente tabla se exponen los requisitos no funcionales del sistema:

Tabla 5. Requisitos no funcionales del sistema.

Requisitos No Funcionales		
Usabilidad		
	RNF1	La interfaz debe ser sencilla y amigable de manera que potencie la comodidad del usuario para su trabajo, además de que las opciones más usadas presentarán vías rápidas y cómodas de invocarse.
Diseño e Implementación		
	RNF2	Se utilizará PostgreSQL en su versión 8.4 o superior como Sistema Gestor de Bases de Datos, dado que el cliente así lo exige.
	RNF3	Se utilizará Groovy como Lenguaje de Programación y Grails como marco de trabajo integrador, dado que el cliente así lo exige.
	RNF4	Se utilizará como arquitectura del sistema Modelo- Vista- Controlador, dado que el cliente así lo exige.
	RNF5	Se utilizará Intellij Idea como IDE para el desarrollo del sistema, dado que el cliente así lo exige.
	RNF6	Se utilizará Windows 7 o superior como Sistema Operativo para el desarrollo del sistema.
	RNF7	Se utilizará Visual Paradigm como herramienta de modelado.
	RNF8	Para el diseño de las páginas se utilizarán las hojas de estilo en cascadas (CSS por sus siglas en inglés), lenguaje de marcas (o etiquetas) hipertexto (HTML por sus siglas en inglés), Lenguaje de marcas hipertexto extensible (XHTML por sus siglas en inglés), JavaScript, JQuery.
Funcionamiento		
Software	RNF9	Para el funcionamiento de los servidores es necesario Windows 7 o superior.
	RNF10	Para el funcionamiento en la PC cliente será necesario Firefox 28 o superior, Google Chrome 16 o superior. Sistema operativo Windows o Mac capaces de soportar los navegadores mencionados anteriormente.
Hardware	RNF11	El servidor para aplicaciones web deberá tener las siguientes características mínimas: ✓ Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior;

		✓ Tarjeta de red o capacidad de conectividad que soporte 1 Gb; ✓ 1 GB de memoria RAM; ✓ Capacidad de 100 GB de disco duro.
	RNF12	El servidor para la base de datos deberá tener las siguientes características mínimas: ✓ Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior; ✓ Tarjeta de red o capacidad de conectividad; ✓ 1 GB de memoria RAM; ✓ Capacidad de 80 GB de disco duro.
	RNF13	Las estaciones de trabajo deberán cumplir los siguientes requisitos mínimos: ✓ Procesador Intel Celeron E3200 a 2.40 Ghz, equivalente o superior; ✓ 256 MB de memoria RAM.
Seguridad		
Confidencialidad	RNF14	El acceso al sistema así como la información se encontrarán protegidos contra accesos no autorizados utilizando mecanismos de autenticación propios del sistema.
	RNF15	La autenticación del sistema será la primera acción del usuario, el cual deberá proporcionar un usuario único y una contraseña, los cuales serán de uso exclusivo del propio usuario.
	RNF16	Las diferentes áreas del sistema se encontrarán protegidas contra acceso no autorizado utilizando roles y grupos de usuarios.
	RNF17	El administrador del sistema como política de seguridad podrá restringir el acceso a las diferentes áreas del sistema a los usuarios o grupos de usuarios.
Integridad	RNF18	La información podrá ser modificada solo por el personal autorizado.
	RNF19	Se harán validaciones de la información en el servidor contra ataques de inyección HTML y SQL.
Fiabilidad		
	RNF20	La respuesta del sistema ante una búsqueda no debe exceder los 5 segundos.
	RNF21	Ante una inserción de datos, vista de detalles, modificación o eliminación el sistema debe responder en no más de 3 segundos.
	RNF22	El sistema debe responder en un promedio de 6 segundos la petición de un reporte.
	RNF23	El sistema deberá soportar transacciones de cerca de 300 usuarios conectados simultáneamente.
Eficiencia		
	RNF24	El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria.
	RNF25	Se respetarán buenas prácticas de programación para incrementar el rendimiento en operaciones costosas.
Interfaz de Usuario		
	RNF26	Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la interpretación correcta de la información.
	RNF27	La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error y mostrando como título el detalle del error.
	RNF28	Todos los textos y mensajes en pantalla serán mostrados según el idioma seleccionado para el sistema.
	RNF29	El diseño de la interfaz del sistema responderá a la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.
Interconexión		
	RNF30	La PC cliente se comunicará mediante el Protocolo de transferencia de hipertexto (HTTP por sus siglas en inglés) y Protocolo de transferencia de hipertexto seguro

		(HTTPS por sus siglas en inglés) con el servidor de aplicaciones, éste se comunicará mediante el Protocolo de control de transmisión/Protocolo de Internet (TCP\IP por sus siglas en inglés) con el servidor de bases de datos.
	RNF31	Los datos de usuarios y demás datos necesarios para el sistema podrán ser extraídos de los servicios web brindados por el UDDI (Directorio de Servicios web de la UCI).

2.6. Fase de Exploración.

La metodología de desarrollo XP inicia con su fase de exploración, en esta fase se realiza el proceso de identificación de las HU. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Dentro de la fase de exploración, se describen las HU, en correspondencia con cada uno de los requisitos funcionales identificados. También se hace una descripción de todos los artefactos generados durante la fase de Exploración, la que toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.6.1. Actores del sistema.

Usuario: Define pruebas funcionales y puede ocupar cualquiera de los siguientes roles.

Roles que puede ocupar el usuario:

Administrador General: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento del sistema informático.

Administrador de Auditorías: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento de los procesos correspondientes a las auditorías.

Administrador de Riesgos: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento de los procesos correspondientes a los Riesgos.

Gestor de Riesgos: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento de los procesos correspondientes a los Riesgos pero sin acceso a la gestión de usuarios de riesgo.

Técnico: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento del servicio que se le ha sido asignado.

Auditor: Es aquel profesional que tiene la responsabilidad de ejecutar, mantener, operar y asegurar el correcto funcionamiento de la auditoría que se le ha sido asignada.

Usuario Auditorías: Es aquel usuario que solo tiene permisos para observar información correspondiente a las auditorías.

Usuario Riesgos: Es aquel usuario que solo tiene permisos para observar información correspondiente a los Riesgos.

Usuario normal: Es aquel usuario que solo tiene permisos para observar información correspondiente a los servicios.

2.6.2. Historias de Usuario.

Las HU son la forma en que se especifican en XP los requisitos del sistema, es uno de los artefactos más significativos que genera la metodología, las cuales son escritas por el propio cliente tal y como ve las necesidades del sistema, aunque el programador puede brindar su ayuda en su identificación. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en corto tiempo. Su tratamiento es muy flexible y dinámico debido a que en cualquier momento pueden ser modificadas o reemplazadas por otras más específicas.

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

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

Nombre: Nombre descriptivo de la HU.

Prioridad en Negocio: Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que puede tomar son (Alta, Media o Baja).

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

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

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

Iteración Asignada: Número de la iteración en la cual será implementada la HU.

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

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

A continuación como resultado de la fase de exploración aparece descrita una de las HU, la cual representa parte del flujo de actividades que permite el monitoreo y control de servicios; en total se definieron 24 HU. Las restantes pueden ser consultadas en el documento "Historias de Usuario" en la carpeta de anexos:

Tabla 6. HU: Adicionar servicio.

Historia de Usuario	
Número: 15.1	**Nombre de la Historia de Usuario:** Adicionar servicio.
Cantidad de modificaciones a la Historia de Usuario: **Ninguna**	
Usuario**: Administrador General.**	**Iteración asignada:** 2
Prioridad en negocio: **Muy Alta**	**Puntos estimados:** 0,2 semanas
Riesgo en desarrollo: **Alto**	**Puntos reales:** 0,1 semanas
Descripción: El sistema debe permitir al usuario adicionar servicios, especificando los datos requeridos para la aplicación: ▪ *Nombre: ▪ *Correo: ▪ *Categoría: ▪ *Nombre del Cliente: ▪ *Estado: ▪ *Prioridad: ▪ *Sitio: ▪ Grupo: ▪ *Asunto: ▪ *Técnico:	
Observaciones: ▪ Si los datos de creación del servicio no son válidos la aplicación mostrará mensajes con los errores correspondientes Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación debe mostrar un mensaje correspondiente a cada error. ▪ Los campos con (*) son obligatorios.	

2.6.3. Tareas de Ingeniería.

Cada una de las historias de usuario se dividirá en tareas a desarrollar, las cuales serán creadas para mejorar la organización de la implementación. Estas tareas serán para el uso estricto de los programadores y pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente. A continuación se muestra una de las tareas de ingeniería, en total se definieron 150 de ellas. Las restantes pueden ser consultadas en el documento "Tareas de Ingeniería" en la carpeta de anexos:

Tabla 7. Tarea de Ingeniería: Diseño e implementación de la clase del dominio "Servicio".

Tarea de Ingeniería	
Número Tarea: 112	**Número Historia de Usuario:** 15
Nombre Tarea: Diseño e implementación de la clase del dominio "Servicio".	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:** 3/8 semanas
Fecha Inicio: 14/03/2014	**Fecha Fin:** 14/03/2014
Programador Responsable: Ariel Ramírez Leyva, Iván Horta Wong	
Descripción: Definición y desarrollo de la clase del dominio "Servicio", así como de sus atributos, métodos y relaciones con el resto de las clases. Esta clase se utiliza como comunicación entre el sistema y los datos persistentes que existen en la tabla "servicio" en la Base de Datos.	

2.7. Fase de Planificación.

Luego de concluida la fase de Exploración, se inicia la fase de Planificación del proyecto, donde el cliente define la prioridad de cada historia de usuario, y en correspondencia con los programadores realizan una estimación del esfuerzo necesario para cada una de las HU definidas por el cliente. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

2.7.1. Estimaciones de esfuerzo por Historias de Usuario.

A continuación se muestran la estimación de las HU por parte del desarrollador, para satisfacer las necesidades del cliente. La estimación incluye todo el esfuerzo asociado a la implementación de cada una de las HU.

Tabla 8. Estimación de esfuerzo por historias de usuario.

Número	HU	Estimación(semanas)
1	Integración de los subsistemas de Gestión de Riesgos y Gestión de Auditorias.	3

2	Gestionar usuario.	1
3	Mostrar usuario	0,2
4	Autenticar usuarios a través del LDAP y Vía Local.	2,2
5	Gestionar rol.	1
6	Mostrar rol.	0,2
7	Asignar Roles a los usuarios.	1,8
8	El sistema debe permitir la personalización de los perfiles de usuarios.	1,2
9	Gestionar categoría.	0,8
10	Mostrar categoría.	0,4
11	Gestionar grupo.	0,8
12	Mostrar grupo.	0,2
13	Gestionar cliente.	0,8
14	Mostrar cliente.	0,2
15	Gestionar servicio.	0,8
16	Mostrar servicio.	0,4
17	Realizar notificaciones a través de correo a los usuarios.	1,2
18	Mostrar un gráfico con los modelos de datos de los sistemas que están incluidos en el SAMOS.	0,4
19	Controlar el histórico de todas las acciones del sistema.	0,6
20	Realizar búsquedas por parámetros.	0,4
21	Un panel de control que contenga un resumen de datos de manera gráfica para ayuda a la toma de decisión de los directivos.	0,6
22	Generación de reportes.	0,4
23	Permitir anexo de cualquier tipo de documento (textos, plantillas, presentaciones, gráficos e imágenes).	0,4
24	El sistema debe de permitir la realización de encuestas de satisfacción y evaluar los resultados de las encuestas.	0,4

2.7.2. Plan de duración de iteraciones.

Después de haber sido estimado el esfuerzo dedicado a la realización de cada una de las HU identificadas, se procede a la creación del plan de iteraciones, el cual tiene como objetivo mostrar la duración y el orden en que serán implementadas.

Tabla 9. Plan de iteraciones.

Id de la HU	Historias de Usuarios	Duración Estimada	Iteración	Duración Iteración
HU_1	Integración de los subsistemas Gestión de Auditorías y Gestión de Riesgos.	11,8 semanas	1	8,8 semanas
HU_2	Gestionar usuario.			
HU_3	Mostrar usuario.			
HU_4	Gestionar Rol.			
HU_5	Insertar Rol.			
HU_6	Autenticar usuarios a través del LDAP y Vía Local.			
HU_7	Asignar Roles a los usuarios.			
HU_8	El sistema debe permitir la personalización de los perfiles de usuarios.			
HU_9	Gestionar categoría.			
HU_10	Mostrar categoría.			
HU_11	Gestionar grupo.	4,4 semanas	2	3 semanas
HU_12	Mostrar grupo.			
HU_13	Gestionar cliente.			
HU_14	Mostrar cliente.			
HU_15	Gestionar servicio.			
HU_16	Mostrar servicio.			
HU_17	Realizar notificaciones a través de correo a los usuarios.			
HU_18	Mostrar un gráfico con los modelos de datos de los sistemas que están incluidos en el SAMOS.	3,2 semanas	3	2,2 semanas
HU_19	Controlar el histórico de todas las acciones del sistema.			
HU_20	Realizar búsquedas por parámetros.			
HU_21	Un panel de control que contenga un resumen de datos de manera gráfica para ayuda a la toma de decisión de los directivos.			
HU_22	Generación de reportes.			
HU_23	Permitir anexo de cualquier tipo de documento (textos, plantillas, presentaciones, gráficos e imágenes).			
HU_24	El sistema debe de permitir la realización de encuestas de satisfacción y evaluar los resultados de las encuestas.			

2.7.3. Plan de entrega.

El plan de entrega es el compromiso del equipo de desarrollo con el cliente, pues en él se define cuando será entregado el producto. A continuación se muestra el plan de entrega para cada iteración:

Tabla 10. Plan de entrega de versiones.

Iteración	Iteración 1	Iteración 2	Iteración 3
Cantidad de HU	10	7	7
Fecha Entrega	3 de marzo del 2014	2 de abril del 2014	24 de abril del 2014

Como se muestra en la tabla anterior, se realizarán 3 entregas del sistema, cada una de ellas implementará las funcionalidades especificadas por la iteración a la que fue asignada, comenzando la implementación el 9 de diciembre del 2013 y se hace la primera entrega el 3 de marzo del 2014 perteneciente a la primera iteración, una segunda entrega el 2 de abril del 2014 perteneciente a la segunda iteración y la tercera entrega perteneciente a la última iteración se realizará el 24 de abril del 2014.

2.8. Diseño del sistema.

Una vez planificada la realización del sistema se procede al diseño, la metodología XP no obliga a la realización de diagramas UML para representar las clases que contendrán la lógica del negocio, en su lugar propone el uso de tarjetas CRC como técnica de modelado para ayudar a los desarrolladores de software a crear diseños de clases orientados a responsabilidades. Esto no implica que no se utilicen los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando su complejidad no sea alta y defina información importante.

2.8.1. Tarjetas Clase-Responsabilidad-Colaboración del sistema.

Las tarjetas CRC permiten ver las clases no como un depósito de datos, sino para conocer el comportamiento de cada una en un alto nivel, son utilizadas para representar las responsabilidades de las clases y sus interacciones. La metodología XP estipula su uso como un artefacto obligatorio durante el desarrollo de un proyecto, debido a los beneficios que aportan a los desarrolladores.

A continuación se muestra una tarjeta CRC, la cual representa el diseño de la clase “ServicioController” orientada a sus responsabilidades; en total se definieron 46 tarjetas CRC. Las restantes pueden ser consultadas en el documento “Tarjetas CRC” en la carpeta de anexos:

Tabla 11. Tarjeta CRC "ServicioController".

Tarjeta CRC	
Módulo de Administración del Sistema Automatizados de Monitoreo y Control de servicios.	
Datos de la clase	
Nombre de la clase: ServicioController	
Responsabilidades	**Colaboradores**

- Es la encargada de:

  - **index()**
  - **list(Integer max)**
  - **create()**
  - **save()**
  - **show(Long id)**
  - **edit(Long id)**
  - **update(Long id, Long version)**
  - **delete(Long id)**

  - Grupo
  - Categoria
  - Cliente
  - Usuario
  - TrazaController
  - AuditoríaController

2.8.2. Patrones de diseño.

El uso de patrones proporciona una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta, los patrones GRASP[21] describen los principios fundamentales de la asignación de responsabilidades a objetos, son una serie de buenas prácticas de aplicación recomendable en el diseño de software. Entre ellos se encuentran el experto, creador, alta cohesión, bajo acoplamiento y el controlador (Larman, 1999).

Experto: Indica que la responsabilidad de creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento) (GAMMA, 2013).

Es el patrón más utilizado para la asignación de responsabilidades. Se encarga de asignar una responsabilidad al experto en información (la clase que cuenta con la información necesaria para cumplir la responsabilidad). Ejemplo del uso del patrón experto en el sistema se encuentra en la clase “Riesgo”, la cual permite conocer la criticidad de un riesgo, siendo la única clase que contiene los datos necesario para calcularla.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias

[21] General Responsability Asignment Software Patterns(patrones generales de software para asignar responsabilidades)

creadas del objeto. En el sistema se evidencia mediante la clase “ServicioController”, ya que tiene la responsabilidad de crear una instancia de la clase “Servicio”, puesto que “ServicioController” es un creador de los objetos de “Servicio”, como se muestra a través de la siguiente imagen:

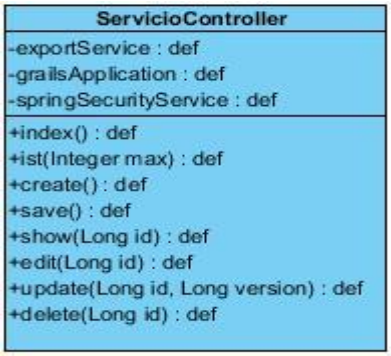

Figura 2. Ejemplo de patrón Creador.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

En el sistema se visualiza el uso del patrón controlador mediante todas las clases controladoras, las tarjetas CRC anteriormente descritas reflejan las principales responsabilidades que tienen, cada clase de este tipo se identifica fácilmente en la implementación por llevar la palabra Controller en su nombre. A continuación se presenta un ejemplo del uso de este patrón en el sistema, donde la clase “ServicioController” recibe los datos del usuario y luego interactúa con el modelo de datos según el método llamado.

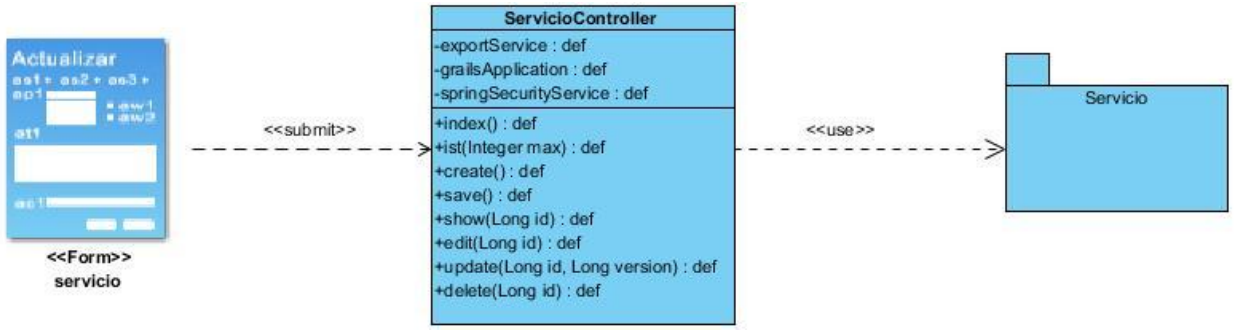

Figura 3. Ejemplo de patrón controlador.

Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño, soporta el diseño de clases más independiente que reduce el impacto de los cambios y también más reutilizable, que aumenta la oportunidad de una mayor productividad. Este patrón no se afecta por cambios que pueden ocurrir de otros componentes, es fácil de reutilizar.

En el sistema se encuentra evidenciado este patrón en la figura 4, la cual ilustra los procesos que ocurren al ser eliminado un servicio de la base de datos, donde la clase "ServicioController" aplica la acción eliminar a las auditorías asociadas, luego la clase "AuditoriaServiciosController" aplica la misma acción a los riesgos identificados por esas auditorías. Por lo tanto se reduce la dependencia de la clase "ServicioController" con el resto de las clases, ya que no tiene asociaciones directas con la clase "Riesgo".

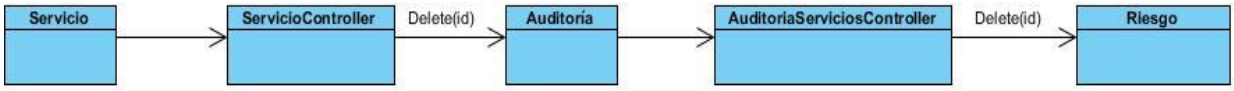

Figura 4. Ejemplo del uso del patrón Bajo Acoplamiento.

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

El diseño de la figura 4, se ilustra como la clase "ServicioController" delega la responsabilidad de eliminar los riesgos identificados por las auditorías de un servicio específico a la clase "AuditoríaServiciosController", evidenciando el uso del patrón Alta Cohesión en el sistema. Este diseño es conveniente ya que da soporte a una alta cohesión y a un bajo acoplamiento.

2.8.3. Patrones GoF.

Durante la fase de diseño orientado a objetos se utilizan los patrones GoF[22], los cuales se clasifican en patrones de creación, estructurales y de comportamiento.

- ✓ Patrones de creación: muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resueltas dinámicamente decidiendo qué clases instanciar o sobre qué objeto otro objeto delegará responsabilidades.
- ✓ Patrones estructurales: describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- ✓ Patrones de comportamiento: se utilizan para organizar, manejar y combinar comportamientos.

A continuación se describe el patrón de diseño utilizado:

Singleton (Solitario):

Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

El uso del patrón singleton se representa en la utilización de los servicios, pues estos por defecto lo utilizan, pues existe una sola instancia de la clase que se usará en todos los artefactos que declaren la variable asociada a esta. Tiene como inconveniente que no se puede guardar de forma privada la información de una petición realizada por un usuario, pues otro usuario que estuviera trabajando en la misma sección del software vería instantáneamente cualquier modificación. Este inconveniente puede eliminarse haciendo uso de la variable “scope”, asignándole el valor “session”, creando de esta forma una instancia diferente del servicio para cada sesión.

2.8.4. Patrón de arquitectura.

Los patrones de arquitectura expresan un esquema organizativo estructural fundamental para sistemas de software. Para el desarrollo del este trabajo se utilizará Grails, este marco de trabajo está basado en el patrón MVC[23]. El patrón de arquitectura MVC permite realizar la programación multicapa, separando en tres componentes distintos:

[22] “Gang of Four” o Pandilla de los cuatro
[23] Modelo Vista Controlador

Modelo: representa la información con la que trabaja la aplicación, la lógica de negocio. Maneja los datos y controla todas sus transformaciones, no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo. El modelo está representado por sus clases de dominio.

Vista: Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

Controlador: proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, se activa ya sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

En el estilo arquitectónico MVC, el procesamiento se lleva a cabo entre sus tres componentes. El controlador recibe una petición y decide quién la lleva a cabo en la capa del modelo. Una vez que el modelo termina las operaciones pertinentes, devuelve el control de ejecución al controlador, y éste envía los resultados a la capa de la vista para que muestre los resultados al usuario (ver Figura 5).

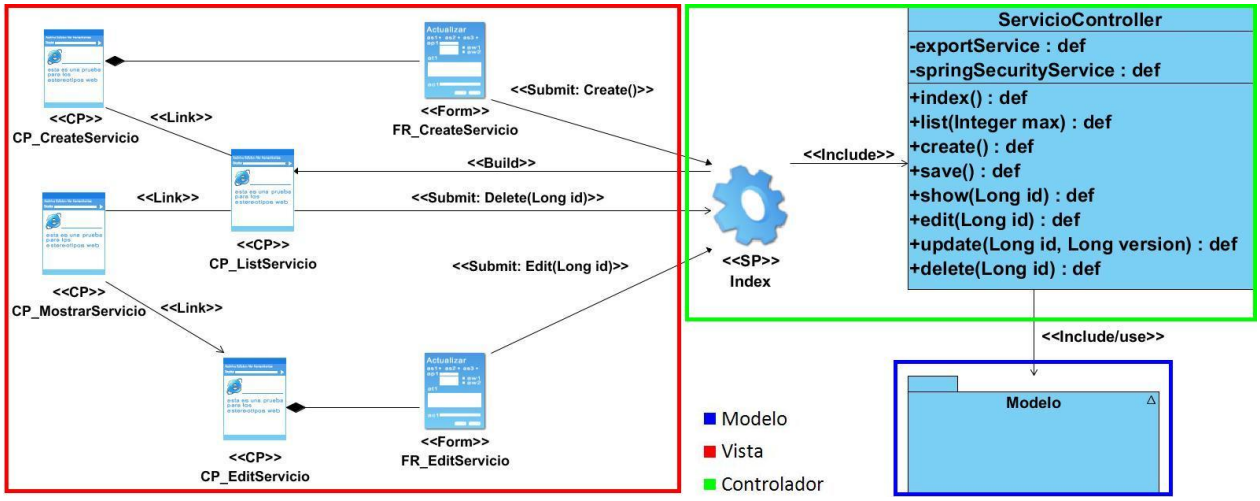

Figura 5. Patrón Modelo Vista Controlador en Diagrama de Clases del Diseño del requisito: Gestionar Servicio.

2.8.5. Diseño de la Base de Datos.

Luego de realizarse el diseño del sistema se pasa a desarrollar el modelo de datos, el cual se puede describir como un conjunto de herramientas conceptuales, para describir la representación de la información en términos de datos. La siguiente figura muestra una un fragmento del modelo de datos

elaborado para el Módulo de Administración del SAMOS, donde se observan las principales entidades y sus relaciones, ver Anexo 5 para visualizar al modelo de datos de la solución desarrollada en su totalidad.

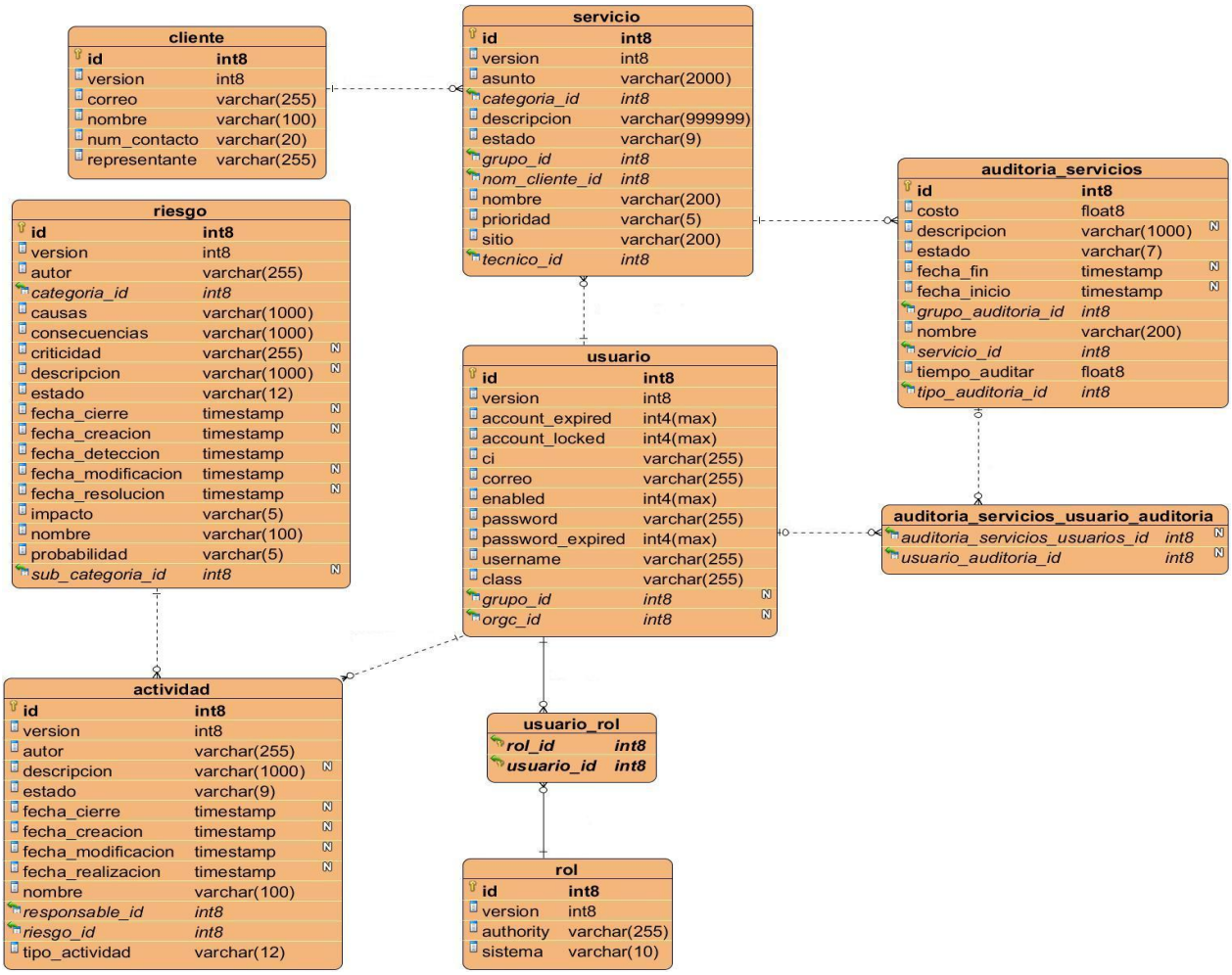

Figura 6. Modelo-Entidad-Relación.

2.9. Conclusiones parciales.

Con la realización de las fases exploración, planificación y diseño se permitió obtener los artefactos propuestos por la metodología seleccionada para el desarrollo de la solución. Obteniéndose entre ellos a las HU para la especificación funcional, facilitando la comunicación del programador con el cliente a la hora de implementarlas. Se elaboraron tareas de ingeniería para una mejor organización de la implementación. Por otro lado se realizaron los patrones de diseño y la descripción de los roles y actores que interactuarán con el sistema. Con todos estos elementos se logró una mayor comprensión del negocio y quedaron definidos los requisitos exigidos por el centro de soporte UCI.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA PROPUESTO.

Introducción

En el presente capítulo se procede a la implementación de las HU en su correspondiente iteración, obteniéndose en cada una de ellas una versión funcional del producto. Se muestra el diagrama de despliegue. Se analiza la propuesta de solución desde el punto de vista de la calidad, para evaluar cómo se le da cumplimiento al problema de la presente investigación y verificar si fueron implementadas de forma correcta cada una de las HU definidas por el cliente. Se procede a diseñar un conjunto de pruebas correspondientes a la metodología XP, las cuales se dividen en tres partes: carga y estrés, unitarias y de aceptación. A través de métricas se evalúa el diseño propuesto, en aras de comprobar su correcta elaboración. Por otro lado, se describe de forma detallada todo el proceso de pruebas y se muestra un resultado final, que explica y muestra al usuario que el sistema está implementado correctamente y que cuenta con las funcionalidades descritas en el capítulo anterior.

3.1. Métricas para la validación del diseño.

Las métricas, dentro del contexto de la ingeniería del software son un grupo de medidas efectuadas sobre programas, documentación, procesos de desarrollo o al mantenimiento, que permite una previa comparación con valores que proporcionan una indicación cuantitativa de extensión, cantidad, dimensión, capacidad y tamaño de algunos atributos de un proceso o producto. Las métricas permiten descubrir y corregir problemas potenciales, antes de convertirse en defectos catastróficos. Se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo.

Métricas propuestas por Lorenz y Kidd.

Lorenz y Kidd concentran las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Estas métricas están enfocadas a las características internas del diseño orientado a objeto, de esta manera contribuyen a asegurar la mantenibilidad de los productos de software.

Las métricas dirigidas al tamaño para una clase, provienen de la normalización de las medidas de calidad y/o productividad considerando el tamaño del sistema que se haya producido, se centran en cálculos de atributos y de operaciones para una clase individual, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y temas

relacionados con el código así como las métricas orientadas a valores externos, que examinan el acoplamiento y la reutilización (Lorenz, y otros, 1994). Del conjunto de métricas planteadas, se aplicó al diseño propuesto:

Tamaño operacional de clase.

La métrica Tamaño Operacional de Clase (TOC) se encarga de medir la calidad de acuerdo a los siguientes atributos:

- ✓ **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

El tamaño general de una clase se puede determinar empleando las medidas siguientes (Pressman, 1998):

- ✓ El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
- ✓ El número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales para esta métrica basándose en el promedio de operaciones por clases, el cual es comparado con la cantidad de operaciones existentes en cada una de las clases, el umbral que se aplica es en correspondencia con el atributo de calidad que se encuentre en evaluación, los umbrales utilizados en el desarrollo de la métrica, son reflejados en la siguiente tabla.

Tabla 12. Umbrales para el TOC.

Atributo	Clasificación	Valores de los umbrales
Responsabilidad	Baja	<=Promedio de operaciones(PO)
	Media	>PO y <=2*PO
	Alta	>2*PO
Reutilización	Baja	>2*PO
	Media	>PO y <=2*PO
	Alta	<=Promedio de operaciones(PO)

Complejidad	Baja	<=Promedio de operaciones(PO)
	Media	>PO y <=2*PO
	Alta	>2*PO

A continuación se muestran los resultados de la evaluación de la métrica TOC:

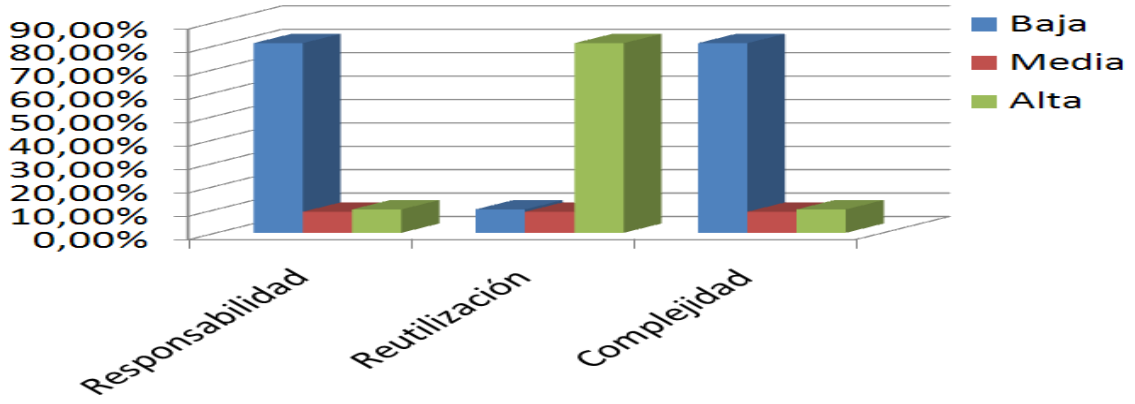

Figura 7. Resultados de la evaluación del diseño con TOC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la métrica TOC, se demuestra que el diseño presenta una calidad aceptable, puesto que las evaluaciones son positivas para los atributos de calidad que la componen. Contando con un 81% de evaluaciones que son clasificadas de baja para los atributos de Responsabilidad y Complejidad, por otro lado se encuentra la Reutilización, con evaluaciones de alta para un 81% de las clases evaluadas.

Métrica de Relaciones entre Clases (RC).

Las relaciones entre clases están dadas por el número de relaciones de uso de una clase con otras y evalúa los siguientes atributos de calidad:

- ✓ **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.
- ✓ **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- ✓ **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- ✓ **Cantidad de pruebas:** Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Los umbrales para esta métrica se estarán basando en el promedio de relaciones por clases, comparando la cantidad de relaciones existentes en cada una de las clases con el promedio antes mencionado, el umbral que se aplica es en correspondencia con el atributo de calidad que se encuentre en evaluación, los umbrales utilizados para el desarrollo de la métrica en el sistema, son reflejados en la siguiente tabla.

Tabla 13. Rango de valores para los criterios de evaluación de la métrica RC.

Atributo	Clasificación	Valores de los umbrales
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de Mantenimiento	Baja	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Alta	>2*Promedio
Reutilización	Alta	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Baja	>2*Promedio
Cantidad de Pruebas	Baja	<=Promedio
	Media	>Promedio y <= 2*Promedio
	Alta	>2*Promedio

A continuación se muestran los resultados de la evaluación de la métrica RC:

Figura 8.Gráfica que representa el porciento por cantidad de dependencias.

Figura 9.Gráfica que representa el porciento de clases por categorías del atributo Acoplamiento obtenidos en la aplicación de la métrica RC.

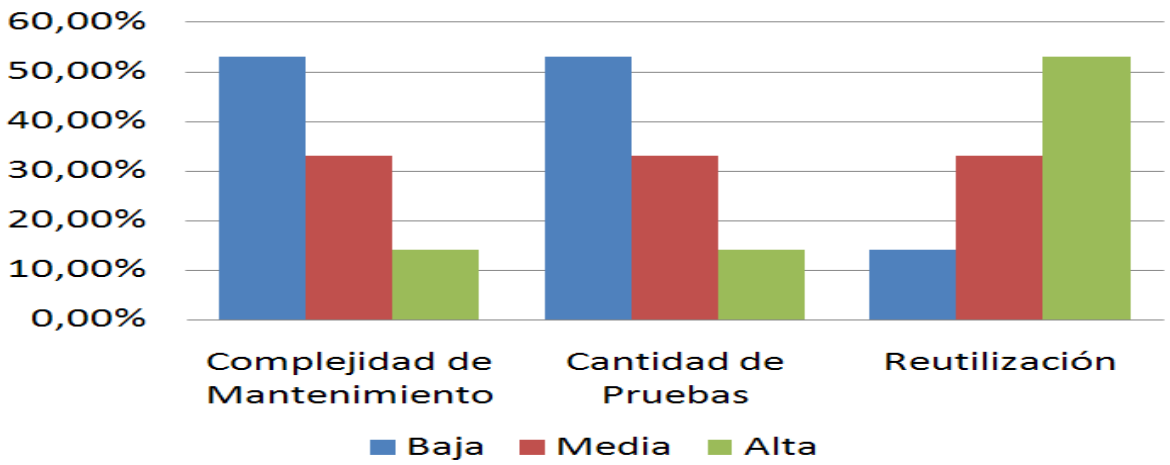

Figura 10. Gráfica que representa el porciento de clases por categorías de los atributos: Reutilización, Cantidad de Pruebas y Complejidad de Mantenimiento obtenidos en la aplicación de la métrica RC.

Haciendo un análisis de los resultados obtenidos en la evaluación de la métrica RC, se puede concluir que el diseño del sistema tiene una calidad aceptable teniendo en cuenta que el 80% de las clases tienen 2 o menos dependencias de otras clases, el 53% de las clases posee índices aceptables en cuanto a Reutilización. Además de contar con evaluaciones satisfactorias para los atributos de Complejidad de Mantenimiento, Cantidad de pruebas y Acoplamiento.

Los resultados obtenidos en la aplicación de las métricas TOC y RC demuestran que la solución desarrollada cuenta con un diseño aceptable, donde todas las clases contienen las acciones necesarias para satisfacer las necesidades del cliente, quedando probado el buen diseño de la solución desarrollada.

3.2. Implementación de la solución propuesta.

En la disciplina de implementación se lleva a cabo la producción de la solución propuesta. El equipo de desarrollo se encarga de implementar las funcionalidades especificadas por el cliente en las historias de usuario utilizando el lenguaje de programación elegido. También se deben garantizar los estándares de codificación para el logro de una buena implementación del sistema.

3.2.1 Diagrama de despliegue.

El Modelo de Despliegue o diagrama de despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En muchos casos el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema (Cauzilla Rojas, 2013).

El diagrama de despliegue que se presenta está compuesto por los cuatro nodos siguientes:

Cliente: accede a la aplicación a través de un computador, donde es ejecutada mediante el navegador Mozilla Firefox 28.0 o superior y sobre el sistema operativo Windows.

Servidor Web: el servidor de aplicaciones empleado donde radica la lógica de negocio de la aplicación, compuesto por el Servidor Web Apache Tomcat 2.1.1 sobre la Máquina Virtual de Java en su versión 1.6.

Servidor de Base de datos: utilizando PostgreSQL en su versión 9.1.1 como SGBD encargado de almacenar los valores que trabajará la aplicación, interconectado con la PC cliente por medio del protocolo HTTP.

Servidor LDAP: el servidor web le realizará consultas a través del protocolo TCP/IP.

A continuación se presenta el modelo de despliegue elaborado para el Módulo de Administración del SAMOS:

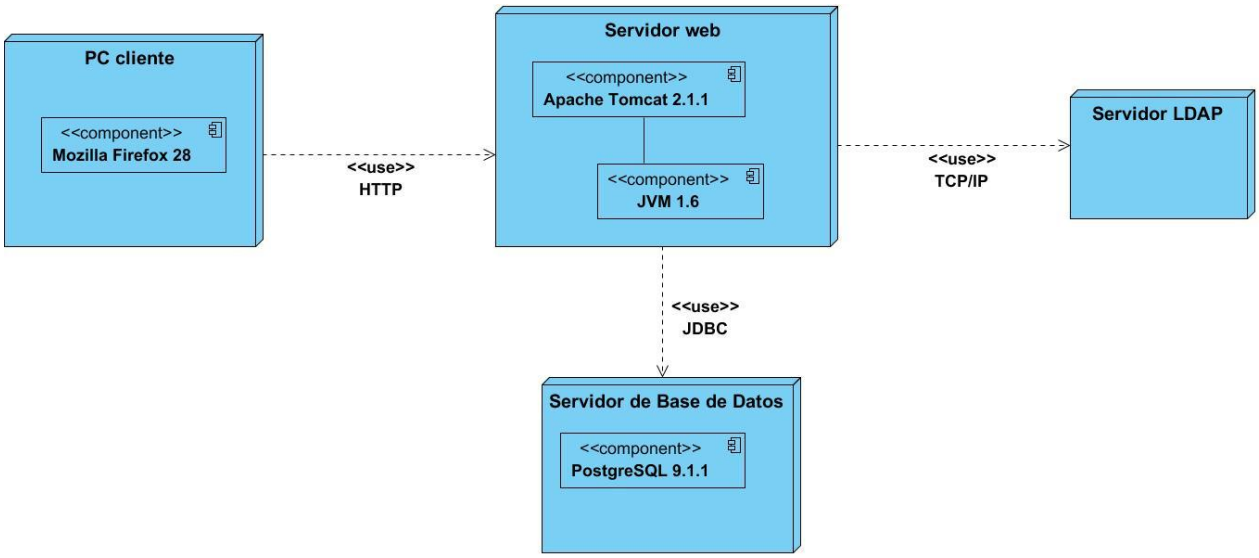

Figura 11. Diagrama de despliegue.

3.2.2 Aspectos principales de la implementación.

Para comenzar con la implementación de la solución desarrollada con la utilización del marco de trabajo Grails, se necesitó crear una carpeta mediante consola con los comandos *grails create-app*. El contenido de esta carpeta se encuentra estructurado por el patrón arquitectónico MVC, compuesto por los paquetes: lib, scripts, target, test, web-app y grails-app. Esta última cuenta con los paquetes que contienen los principales componentes que permiten la implementación del flujo de acciones del negocio, entre ellos:

controllers: Paquete que contiene los controladores de la aplicación Grails. En los controladores es donde se definen acciones, se re-direcciona o se personaliza el scaffolding[24] de la aplicación.

domain: Paquete que contiene las clases del dominio, que no son más que las clases persistentes de la aplicación que se convierten en tablas de la base de datos.

views: Paquete que contiene todas las vistas correspondientes con las acciones definidas en los diferentes controladores organizadas en subdirectorios con el mismo nombre.

El principal aspecto de la solución desarrollada es la utilización de acciones relacionadas con la gestión de servicios, con la existencia de un flujo capaz de listar, crear, salvar, mostrar, editar, actualizar y eliminar mediante la implementación de sus respectivas clases. Además del uso del plugin Spring Security, el cual permite que los usuarios ocupen permisos de acceso a cada una de las acciones según el rol que

[24] Técnica que contiene algunos *frameworks* basados en el patrón arquitectónico MVC, genera controladores y vistas.

desempeñen en el sistema. Un ejemplo de cómo se comunican los diferentes componentes que permiten estos aspectos, específicamente dentro de la acción eliminar, es la implementación en la clase "ServicioController" que se muestra en la siguiente declaración:

```
  @Secured("hasAnyRole('Administrador General')")
def delete(Long id) {
    def servicioInstance = Servicio.get(id)
    if (!servicioInstance) {
            TrazaContrllerInstance.adicionar("Fall\u00f3 al eliminar un servicio del Subsistema de Servicios")
    flash.message = message(code: 'default.not.found.message', args: [message(code: 'servicio.label', default: 'Servicio'), id])
    redirect(action: "list")
    return
    }

    try {
                    def auditoriaController= new AuditoriaServiciosController();
                    auditoria.AuditoriaServicios.findAllByServicio(servicioInstance).each{
                    auditoriaController.delete(it.id)
                    }
        def n = servicioInstance.nombre
        servicioInstance.delete(flush: true)
                    TrazaContrllerInstance.adicionar("Elimina al servicio ${n} del Subsistema de Servicios")
        flash.message = message(code: 'default.deleted.message', args: [message(code: 'servicio.label', default: 'Servicio'), id]

    }
    catch (DataIntegrityViolationException e) {
    flash.message = message(code: 'default.not.deleted.message', args: [message(code: 'servicio.label', default: 'Servicio'), id]
        redirect(action: "show", id: id)
    }
}
```

Figura 12. Implementación de la acción eliminar de la clase "ServicioController".

En este caso se muestra y explica el flujo de eliminar un servicio del sistema para el cual se le pasa el identificador desde la vista. Primero el sistema se asegura de que el usuario tenga los permisos necesarios mediante la anotación de Spring Security **@Secured("hasAnyRole('Administrador General')")**, que solo le dará permisos a los administradores generales a realizar la acción de eliminar servicio . Dentro de la funcionalidad lo primero que se hace es buscar el servicio que se quiere eliminar en el modelo de datos, mediante el método dinámico **get()** del gestor de persistencia GORM[25]. Se comprueba la existencia de esa entidad y de no ser así se crea una nueva traza reportando lo sucedido y se redirecciona a la vista de la lista de servicios mostrando un mensaje de error.

De encontrarse el servicio, el sistema crea una instancia de la controladora auditoriaController, luego llama a la función **findAllByServicio()** que crea una lista de las auditorías que están asociadas al servicio seleccionado recorriéndola mediante el método **each()**, seguidamente la controladora va eliminando cada una de las auditorías asociadas. Al terminar esta acción guarda el nombre del servicio en la variable n, elimina el servicio y emite una traza con la acción sucedida. Si al eliminar el servicio incurre en alguna

[25] Mapeo Objeto-Relacional de Grails.

excepción por violar la integridad de los datos, el sistema muestra el servicio e indica mediante un mensaje que no puede eliminarse.

3.2.3. Estándares de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse que todos los programadores del proyecto trabajen de forma coordinada.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente, es en la técnica de codificación. Para la solución del problema tratado en este trabajo se ha establecido un estándar de codificación, que es presentado a continuación:

Notación Pascal Casing: es un procedimiento de programación común en el lenguaje Java y .Net. La nomenclatura está compuesta por tantas palabras como sean necesarias. La primera letra de cada una de las palabras irá siempre en mayúsculas, obviando el uso de artículos. En el sistema se hace uso de Pascal Casing para nombrar las clases que conforman el dominio, en la figura 9 se muestra el ejemplo de este tipo de procedimiento con la estructura de la clase Servicio.

```
package gestionServicio

class Servicio {
    String nombre

 static constraints = {
    nombre(blank: false, size: 1..200,shared:"nombreEntidad")
  }
 public String toString(){
    nombre
  }
}
```

Figura 13. Ejemplo del uso de PascalCasing en el sistema.

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

Existen dos tipos de CamelCase:

UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula. Ejemplo: EjemploDeUpperCamelCase.

lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeLowerCamelCase.

En el sistema se hace uso del lowerCamelCase para nombrar los atributos y métodos de cada una de las clases existentes, en la figura 10 se muestra el ejemplo de este tipo de procedimiento con la estructura de la clase AuditoriaServiciosController.

```
package auditoria

class AuditoriaServicios implements Serializable {

 static hasMany = [usuarios: Seguridad.UsuarioAuditoria, documentos: Documento]
 static hasOne = [grupo_auditoria : GrupoA]
   String nombre
   Date fechaInicio

   static constraints = {
       nombre(unique:true,blank: false, size: 1..200, shared:"nombreEntidad")
       fechaInicio display: false, nullable: true
   }

def beforeInsert()
{
   if(estado=='Abierto')
    fechaInicio= new Date()
 }

}
```

Figura 14. Ejemplo del uso de CamelCasing en el sistema.

Comentarios:

Los comentarios del código se deben localizar en sus líneas correspondientes y en el siguiente formato.

// space first, with no full stop needed

Con el desarrollo del estilo de código planteado anteriormente hace que el programa fuente sea más legible, lo cual ayuda en la comunicación entre desarrolladores, y permite una temprana detección de errores.

3.3. Pruebas.

Uno de los pilares de la metodología XP es el uso de pruebas con el fin de asegurar en todo momento la realización de lo planteado en el diseño. Según esta metodología se debe ser estricto en la fase de pruebas, debido a que de esta manera se reduce el número de errores no detectados, para así evitar que la calidad del producto desarrollado se vea afectada.

3.3.1. Pruebas Unitarias.

Las pruebas unitarias son procedimientos para validar que una porción del código del sistema funciona apropiadamente de manera aislada. Cabe agregar, que los programadores siempre prueban el código durante el desarrollo, por lo que las pruebas unitarias son realizadas solamente después de que el programador considera que el componente está libre de errores.

Encontrar errores es una tarea desafiante para cualquier desarrollador, pero existen herramientas que proporcionan una manera sencilla, rápida y elegante para escribir pruebas y validarlas automáticamente. JUnit es precisamente, una de estas herramientas, convertida en el estándar universal para realizar pruebas de unidad en Java. Para la realización de las pruebas unitarias, en la presente investigación se utilizó la biblioteca JUnit, la cual está integrada en el marco de trabajo de desarrollo Grails.

JUnit.

Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. JUnit permite ejecutar pruebas automatizadas fácilmente. JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación. Contiene mecanismos para recolectar resultados de manera estructurada, produce varios tipos de reportes a partir de los resultados obtenidos en la ejecución de las pruebas, permite que existan relaciones entre las pruebas y que unas reutilicen código de otras. Además, soporta la jerarquía entre las pruebas pudiendo establecerse la prioridad y el orden de unas con otras.

Es importante destacar que aunque la implementación supere todas las pruebas, no necesariamente implica que todo este correcto; sólo infiere que funciona conforme a los casos de prueba que se han diseñado o para las pruebas realizadas. En otras palabras, el hecho de que JUnit visualice que la prueba realizada está correcta, no sería confiable darlo por hecho. Los desarrolladores de la capa de persistencia al ser los encargados de manejar directamente todos los componentes para el acceso a datos, deberían cerciorarse accediendo al gestor de base de datos y verificar visualmente si la operación se realizó satisfactoriamente. A continuación se presenta como ejemplo, la realización de las pruebas unitarias la clase "ServicioController", las restantes pueden ser consultadas dentro de la carpeta de anexos.

Figura 15. Pruebas Unitarias con JUnit: Clase "ServicioController".

Resultados de las Pruebas Unitarias.

Las pruebas dieron resultados satisfactorios y se accedió al gestor de base de datos verificándose visualmente que las operaciones se realizaron satisfactoriamente, demostrando así un correcto funcionamiento del sistema. A continuación se plantean varios elementos aportados por las pruebas unitarias desarrolladas:

- ✓ Aseguran la calidad del código entregado. Es la mejor forma de detectar errores tempranamente en el desarrollo.
- ✓ Ayudan a definir los requisitos y responsabilidades de cada método en cada clase probada.

- ✓ Facilitan que el programador cambie el código para mejorar su estructura (lo que se conoce como refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- ✓ Aseguran que una corrección no repercuta en otros módulos, permitiendo al programador centrarse en la corrección del error y no en la repercusión que puede tener esa corrección.
- ✓ Las pruebas funcionales se hacen más sencillas: pues la mayoría de los aspectos individuales de cada unidad ya están probados a través de las pruebas unitarias. De este modo, las pruebas funcionales deben centrarse solo en verificar la correcta cooperación de las distintas unidades y en los funcionamientos generales del programa.

3.3.2. Pruebas de aceptación.

Las pruebas de aceptación, también consideradas como "*pruebas de caja negra*" o "*pruebas funcionales*", describen un escenario de ejecución o uso del sistema desde la perspectiva del cliente, los cuales son responsables de verificar que los resultados de estas pruebas son correctos y de esta forma comprobar que las HU han sido correctamente implementadas. Una prueba de este tipo puede tener infinitas ejecuciones con valores concretos y cubrir desde escenarios típicos y frecuentes hasta los más excepcionales. Además que una HU no se puede considerar terminada hasta que pase todas las pruebas de aceptación.

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

- ✓ **Código:** servirá como identificador de la prueba realizada, a su vez será sugerente al nombre de la prueba a la que hace referencia.
- ✓ **HU:** tendrá el nombre de la HU a la que hace referencia la prueba a realizar.
- ✓ **Nombre:** nombre que se le da a la prueba a realizar.
- ✓ **Descripción:** se describe la funcionalidad que se desea probar.
- ✓ **Condiciones de Ejecución:** mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.

- ✓ **Entradas/Pasos de Ejecución:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Resultado esperado:** se hará una breve descripción del resultado que se espera obtener con la prueba realizada.
- ✓ **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación que tendrá uno de los tres resultados que a continuación se describen:

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

Para asegurarse que la aplicación desarrollada cumple con los requisitos exigidos por el cliente, a continuación se muestran algunos de los casos de prueba de aceptación realizados a las HU identificadas en el capítulo anterior; los restantes casos se encuentran en la plantilla “Casos de pruebas de aceptación” de la carpeta “Anexos”.

Tabla 14. Caso de Prueba de Aceptación: Adicionar servicio.

Caso de Prueba de Aceptación	
Código: **HU30_P31**	**Historia de Usuario:** 30
Nombre: Adicionar servicio	
Descripción: Prueba para la funcionalidad que permite adicionar un servicio.	
Condiciones de Ejecución: El usuario debe estar autenticado con el rol de Administrador General.	
Entradas/Pasos de Ejecución: El usuario podrá seleccionar la opción que le permite acceder a la lista de servicios al desplegar el subsistema de Gestión de servicios en el menú de la interfaz inicial, seguidamente se selecciona el botón que permite adicionar un servicio. Luego se muestra el formulario con todos los campos a llenar y procede a insertar los datos requeridos. Finalmente el usuario ejecuta la opción “Crear” y el sistema muestra un mensaje confirmando que la acción fue realizada correctamente.	
Resultado Esperado: Los datos son guardados satisfactoriamente	
Evaluación de la Prueba: Bien.	

Resultados de las pruebas de aceptación.

Se realizaron 47 pruebas funcionales para validar que el sistema funcionara correctamente, mostrando las salidas correspondientes a cada escenario, en tres iteraciones de pruebas donde se detectaron varias no conformidades (NC) que fueron mitigadas durante el transcurso de la presente etapa. En la primera iteración de pruebas se detectaron un total de 12 NC que representa un 25.53% de las pruebas, donde la mayoría eran por problemas de validación. Posteriormente en la segunda iteración se validó la corrección de los errores encontrados en la primera, aunque se detectaron 4 nuevas NC representando el 8,51% de las pruebas y por último en la tercera iteración no se encontraron NC, constituyendo esto un 100% de pruebas de aceptación exitosas, todas estás iteraciones se realizaron de manera conjunta con el cliente.

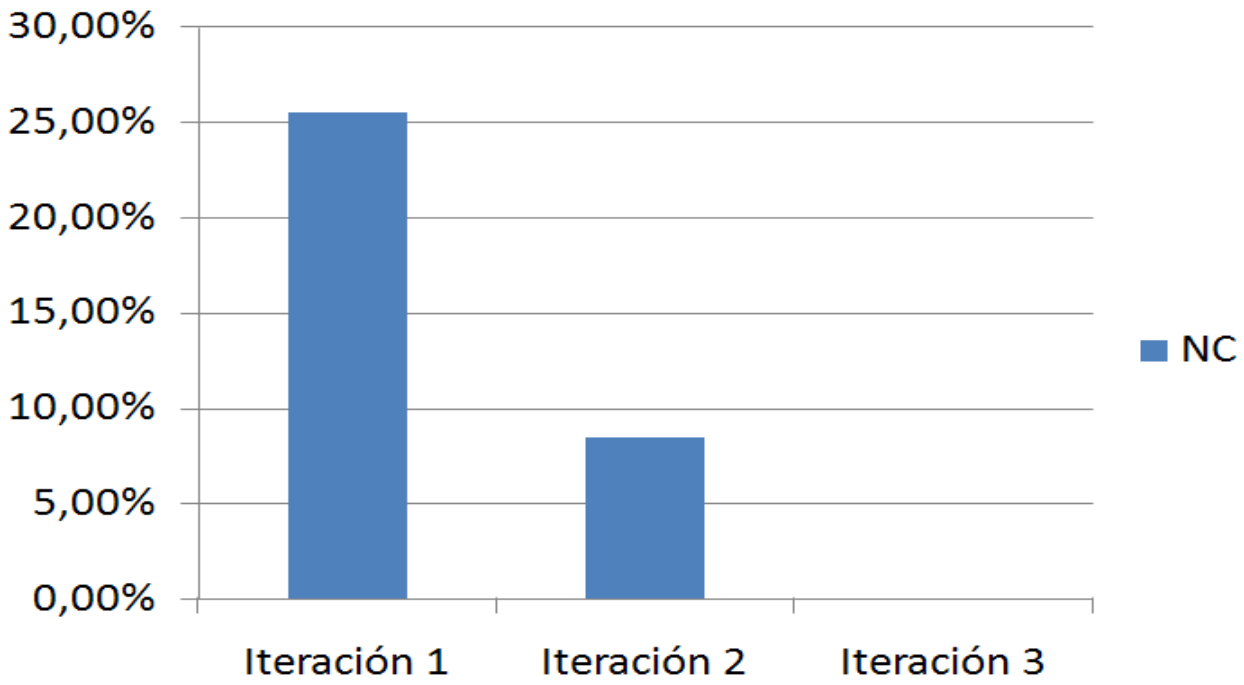

Figura 16. Resultados de las NC.

3.3.3. Pruebas de Carga y Estrés.

Con el fin de realizar evaluaciones relacionadas con el comportamiento, criterios de rendimiento y la capacidad de carga máxima que puede tener la aplicación en el servidor, se realizaron las pruebas de Carga y Estrés. Estas se centran en determinar la velocidad con la que el sistema realiza una tarea en las condiciones particulares del escenario de pruebas.

Por otro lado existen variadas herramientas que permiten desarrollar pruebas de software con el objetivo de verificar y revelar la calidad de un producto informático, algunos ejemplos de estas son JUnit, JWebUnit, LoadRunner y el JMeter. Esta última es la usada para realizar las pruebas de carga y estrés al sistema desarrollado, debido a que se destaca por su versatilidad, estabilidad y por ser de uso gratuito. Además de la herramienta antes mencionada se cuenta con la ayuda de los diseños de casos de pruebas y los requisitos establecidos del sistema desarrollado, donde se obtendrá una serie de datos necesarios para identificar, interpretar e informar el rendimiento de la aplicación.

Aplicación del procedimiento para las pruebas de rendimiento de carga y estrés.

Este procedimiento se le aplica al Módulo de Administración del SAMOS, de manera que los usuarios puedan realizar las acciones correspondientes a los subsistemas que componen el módulo.

Objetivo del procedimiento.

Detectar errores en el rendimiento de la aplicación con respecto a las expectativas del centro de soporte UCI, teniendo en cuenta las características del centro y los recursos que se usen en la ejecución de las pruebas de carga y estrés.

Alcance del procedimiento.

Se realizan las pruebas de rendimiento de carga y estrés a la solución desarrollada. Se observa y se identifican fallos en el rendimiento de la aplicación con la ayuda de los principales escenarios de los casos de pruebas que se definan como críticos (se lleva al sistema a una situación extrema donde se prueba realmente sus capacidades y donde se comienza a ver que puede presentar algunos errores en las peticiones de los diferentes usuarios, al llevar a cabo ciertas tareas) y la ejecución de la herramienta automatizada JMeter.

Fase para aplicar el procedimiento.

Para aplicar el procedimiento a la solución desarrollada se basa en la fase de planificación de las pruebas, ejecución de las pruebas y cierre de las pruebas, donde se describen todos los elementos necesarios de las actividades que se definieron en el capítulo anterior para lograr ejecutar las pruebas de carga y estrés de manera satisfactoria.

Planificación de las pruebas.

Actividad 1. Identificar el entorno de las pruebas.

El Módulo de Administración del SAMOS contiene los documentos necesarios para ejecutar las pruebas de carga y estrés como: casos de pruebas y documentos de especificación de requisitos**.** Primeramente se capturan las funciones de la aplicación para identificar criterios de aceptación de rendimiento, que se hace sobre la base de toda la documentación del módulo desarrollado.

Segundo obtener una visión rápida de lo que hace el módulo que se va a probar: Este módulo tiene como objetivo facilitar el monitoreo y control de servicios que se brindan en el centro de soporte UCI, para esto cuenta con una serie de funcionalidades, entre ellas se encuentra la gestión de servicios por parte de los especialistas, la cual posibilitará la creación de servicios de soporte técnico y tendrá asociada otras funcionalidades necesarias para el desarrollo del trabajo tales como la gestión de auditorías y gestión de riesgos, así como la satisfacción de que los usuarios interactúen con las distintas acciones del sistema sin ningún tipo de problema.

Luego se escoge las principales actividades que el usuario realiza en el sistema, debido a que es imposible simular todas las tareas que realiza el usuario en la aplicación, ya que es recomendable hacer las pruebas sin mucha navegación del sistema, ya que se hace tedioso analizar las peticiones que se le hacen al servidor, por lo que se escogen los siguientes casos de pruebas críticos para el sistema: Caso de prueba: Adicionar servicio, Caso de prueba: Modificar Riesgo, Caso de prueba: Listar Auditoría.

Para la aplicación del procedimiento de pruebas de rendimiento diseñado, se identifica un entorno lo más similar posible al laboratorio de producción del centro de soporte UCI.

Servidor para aplicaciones web con las siguientes características:

- ✓ Procesador Intel Pentium Dual Core a 2.0 Ghz.
- ✓ Tarjeta de red o capacidad de conectividad de 1 Gb.
- ✓ 1 GB de memoria RAM.
- ✓ Capacidad de 100 GB de disco duro.

Servidor para la base de datos con las siguientes características:

- ✓ Procesador Intel Pentium Dual Core a 2.0 Ghz.
- ✓ Tarjeta de red o capacidad de conectividad.
- ✓ 1 GB de memoria RAM;
- ✓ Capacidad de 80 GB de disco duro.

Software con las siguientes características:

- ✓ Windows 7 como Sistema Operativo.
- ✓ PostgreSQL en su versión 9.1 como Gestor de Base de Datos.
- ✓ Kaspersky Workstation 6.0 como Antivirus.
- ✓ Virtual Machine 1.6 como Máquina Virtual Java.
- ✓ JMeter Versión 2.3.1 como Software de Prueba.

Actividad 2. Requisitos de aceptación de rendimiento.

Los requisitos de rendimiento suelen identificarse por las características del proyecto y las necesidades del usuario. Para esta aplicación se definen los siguientes requisitos:

La máxima carga de usuarios que debe soportar la aplicación en el centro de soporte UCI es de 300 usuarios conectados. Para realizar esta prueba se toma como muestra 300 usuarios conectados concurrentemente. Con 300 usuarios conectados, la aplicación no debe distorsionar imágenes, ni gráficos. Otros aspectos importantes a la hora de medir el rendimiento del sistema son:

Tiempos de respuesta: En situaciones de carga de una transacción sería entre 1 y 3 segundo.

Utilización de recursos: La cantidad de recursos que la aplicación está consumiendo en términos de procesador, memoria, disco de entrada y salida y la red. Ejemplo el rendimiento de un servidor se degrada, si el uso del procesador regularmente supera el 80 %.

Actividad 3. Diseño de las pruebas.

Para calcular el total de los resultados esperados de cada transacción que se hace en los escenarios de los casos de prueba seleccionados en la actividad de identificar los casos de pruebas a simular, se descarga el plugins firebug 1.6.1 que representa una extensión del Mozilla Firefox, que analiza el código fuente de las aplicaciones que se ejecuten en este navegador y muestra el tiempo de respuesta de determinada petición que se haga al servidor.

Para las siguientes pruebas se analiza el parámetro de rendimiento que ofrece la herramienta JMeter. Y se tendrá en cuenta la capacidad del procesador de la máquina donde se realiza la prueba.

Tabla 15.Carga de trabajo y resultados esperados de los escenarios.

Identificador de Escenarios	Escenarios	Carga de Trabajo	Descripción	Resultados esperados	Resultados obtenidos de las pruebas

EC1	Seleccionar la opción de Crear servicio. http://localhost:8080/admin/servicio/create/	300	Permite al usuario acceder a la opción de crear servicios.	La aplicación debe responder en un tiempo menor de 3 seg.	La aplicación responde en un Tiempo (parámetro de rendimiento de JMeter) de 2.6 seg.
EC2	Seleccionar la opción de editar riesgo. http://localhost:8080/admin/riesgo/edit/	300	Permite al usuario editar riesgos, especificando los datos requeridos para la aplicación.	La aplicación debe responder en un tiempo menor a 3 seg.	La aplicación responde en un tiempo (parámetro de rendimiento de JMeter) de 2.6 seg.
EC3	Seleccionar la opción de Listar auditorías. http://localhost:8080/admin/auditoriaServicios/list/	300	Permite al usuario listar las auditorías.	La aplicación debe responder en un tiempo menor a 3 seg.	La aplicación responde en un tiempo (parámetro de rendimiento de JMeter) de 2.8 seg.

Ejecución de las pruebas.

Una vez definido los casos de prueba con los escenarios y rutas de navegación principales se procede a ejecutar la herramienta JMeter la cual genera los script que permiten mostrar toda la información referente al rendimiento de los casos de pruebas.

Análisis e interpretación de los resultados.

El rendimiento de la aplicación para los casos de pruebas evaluados se evidencia en las siguientes tablas que contienen los resultados generados por el informe agregado que se escoge como plan de prueba de la herramienta JMeter para mostrar los valores de las pruebas.

Tabla 16. Resultados EC1: Selecciona la opción de crear un nuevo servicio.

Peticiones	#Muestra	Media	Mediana	Línea de 90%	Mix	Max	%Error	Rendimiento	KB/S
Servicios	300	6.3	0	25	0	43	0.0%	2,6 SEC	101,7
Total	2800	1844	18	1145	0	14532	0.0%	39,6	18514.9

Tabla 17. Resultados EC2: Selecciona la opción de editar un nuevo riesgo.

Peticiones	#Muestra	Media	Mediana	Línea de 90%	Mix	Max	%Error	Rendimiento	KB/S
Riesgos	300	5.5	0	25	0	43	0.0%	2,6 SEC	101.7
Total	2800	1844	18	1145	0	14532	0.0%	39,6	18514.9

Tabla 18. Resultados EC2: Selecciona la opción de listar las auditorías.

Peticiones	#Muestra	Media	Mediana	Línea de 90%	Mix	Max	%Error	Rendimiento	KB/S
Auditorías	300	6.06	0	40	0	43	0.0%	2,8 SEC	101.7
Total	2800	1844	18	1145	0	14532	0.0%	39,6	18514.9

Significado de los elementos del informe agregado:

- ✓ # Muestras: Cantidad de páginas (hilos) que simulan la cantidad de usuarios, que están interactuando con el sistema desde la misma URL.
- ✓ Media: Media de páginas que se cargaron satisfactoriamente.
- ✓ Mediana: Tiempo promedio que ha tardado en cargarse las páginas.
- ✓ Línea de 90 %: El 90 % de las páginas que se cargaron de manera satisfactoria.
- ✓ Mix: Tiempo mínimo en que ha demorado en cargarse una página.
- ✓ Max: Tiempo máximo en que ha tardado en cargarse una página.
- ✓ % Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- ✓ Rendimiento: Tiempo total que demoró en cargarse la cantidad de peticiones de la prueba.
- ✓ KB/S: Tiempo total que demoró en cargarse la cantidad de peticiones de la prueba en Kbyte por segundo.

Como se puede ver el rendimiento o el tiempo en que responde la aplicación ante las transacciones de 300 usuarios conectados concurrentemente es menor que los resultados esperados para los tres escenarios de la prueba. Con la ejecución de la herramienta automatizada JMeter, se obtuvo un total de 2800 páginas,

donde la aplicación generó un total de 18514,9 KB/s de transferencia de datos, lo que incurrió en un rendimiento de 39.6 segundos y una tasa de errores de 0.0 %, demostrando que la aplicación es estable.

Realizar la prueba de Estrés.

Con el objetivo de someter la aplicación a una carga elevada como valores numéricos complejos, elevado número de entradas, elevado número de peticiones, que comprueban el límite de carga que la aplicación puede soportar. Las pruebas de estrés intentan romper el sistema desbordando sus recursos o reduciendo la cantidad de estos. Los objetivos de la prueba son encontrar los límites del sistema y asegurar que tras un fallo en el sistema, se recupera sin causar graves problemas.

Por los planteamientos antes mencionados se puede analizar el sistema desarrollado de la siguiente manera: La utilización de los recursos en el EC3, lleva al procesador a una capacidad entre 40 % y 50% para 300 usuarios simulados por la herramienta y al desarrollar la prueba con una muestra de 1500 usuarios, se obtiene un informe agregado con un 80 % de capacidad por lo que el sistema empieza a degradarse, la aplicación se torna más lenta siendo esto, una de las principales causa de abandono de páginas web por parte de los usuarios. Esta aparece debido a que durante la grabación de la carga, se observa que se graba en la herramienta JMeter todas las peticiones HTTP que hace constantemente el Jabber interno de la aplicación al servidor, por lo que interfiere en los resultados de la prueba.

Cierre de la prueba.

La aplicación de las pruebas de carga y estrés con la herramienta JMeter se realiza de manera satisfactoria, pues le permite conocer el punto donde el sistema comienza a tener problema con determinado número de usuarios, lo cual resulta necesario, para tomar ciertas precauciones como: revisar el código de los programas y los tiempos de respuestas de los servidores.

Con el desarrollo de las pruebas de Carga y Estrés al Módulo de Administración del SAMOS se identificaron varios elementos de importancia para determinar el rendimiento del sistema, los cuales son:

- ✓ Se pudo determinar un estimado sobre cómo responde la aplicación ante una carga de trabajo en específica.
- ✓ Técnicamente el resultado de estas pruebas muestra que los requisitos no funcionales que se recogen cumplen con las expectativas del centro de soporte UCI.
- ✓ Se estima que la aplicación funcione correctamente en el centro de soporte UCI para 300 usuarios no siendo así para 1500 usuarios.

- ✓ Indica cómo va a reaccionar el sistema cuando excede el límite de su funcionamiento, en este caso el exceso es de sobre el número de usuarios conectados concurrentemente a la aplicación web.

3.3.4. Seguridad.

Spring Security es un marco de trabajo usado en la solución desarrollada, está pensado para que pueda ser utilizado en cualquier entorno y puede dar soporte a todas las capas de una aplicación. En el cual se filtran todas las llamadas a los controladores de la aplicación, verificando en cada una que el usuario esté autenticado y que posea el rol que le permite hacer uso de las funcionalidades del sistema sobre las cuales tiene permiso de acceso. Se representa en la arquitectura de manera vertical debido a que la seguridad es gestionada en todo el sistema.

Sobre la base de las consideraciones anteriores se demuestra que el uso del marco de trabajo Spring Security en la solución desarrollada asegura la protección de los datos y un funcionamiento de acuerdo a los propósitos para los que fue diseñado. Además, se verifica que la aplicación y la infraestructura que la soporta no aporten vulnerabilidades que puedan ser aprovechadas por terceros para uso no deseado. Con la utilización del Spring Security se permite el cumplimiento de los siguientes aspectos de la información:

- ✓ **Confidencialidad de la información:** evitar que usuarios o sistemas no autorizados accedan a la información.
- ✓ **Integridad de la información:** asegurar la exactitud y la completitud de la información, así como los métodos que se utilizan para su procesamiento.
- ✓ **Disponibilidad de la información:** asegurar que la información esté disponible para usuarios y sistemas autorizados en el momento que lo requieran.

3.4. Validación de la solución.

Con el objetivo de demostrar que el tiempo disminuye en la gestión de los servicios de soporte técnico con la utilización de la solución desarrollada, se realizó un experimento con 4 especialistas del centro de soporte UCI.

Indicador a medir:

- ✓ **Tiempo:** magnitud física que permite determinar la duración o demora de realización de una acción correspondiente a la gestión de servicios de soporte técnico ocupado por un especialista, en caso de éxito.

El experimento consistió en el llenado de los documentos Auditoría de Servicios y Reporte de Auditorías.

Primero de forma manual y luego a través de interfaces mediante formularios para medir el tiempo de demora de un especialista (mm:ss, ms) en cada uno de los casos. Para cada escenario se utilizó el mismo juego de datos teniendo en cuenta el documento sobre el que se aplica.

Resultados del experimento:

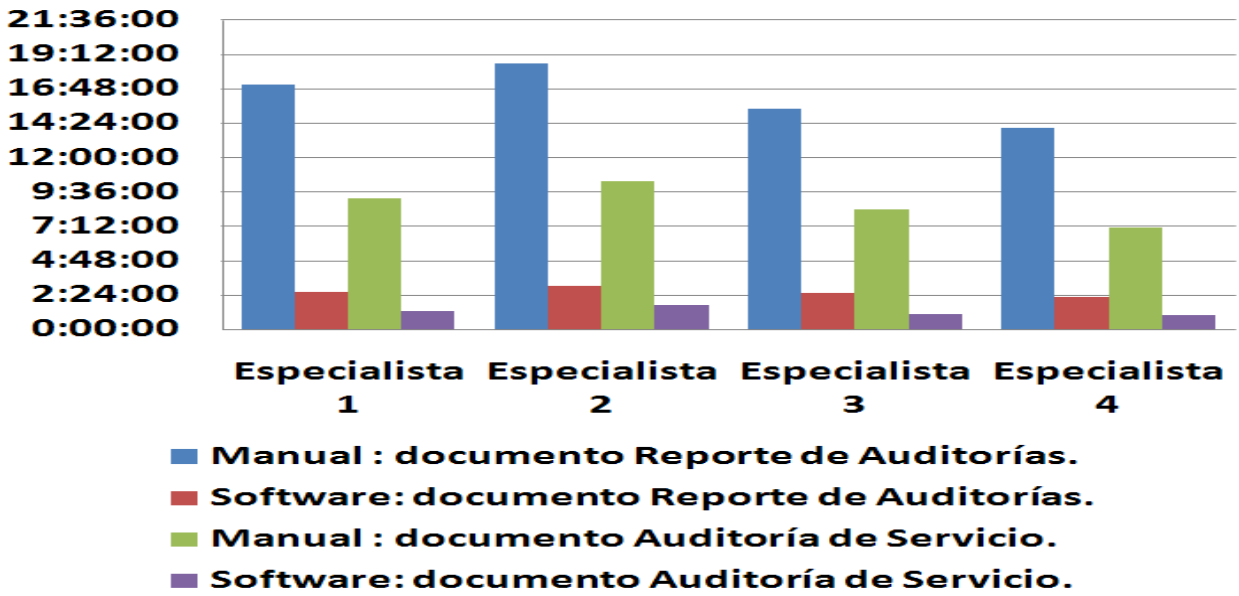

Figura 17. Resultados del experimento.

Al ser aplicado el llenado de los documentos mediante la forma manual se observa que los especialistas ocupan un promedio de 12 minutos para completar sus tareas, luego se realizó el mismo llenado pero con el uso de la solución desarrollada ocupando un promedio de 2 minutos. Todo esto indica que con la solución desarrollada se realizó el llenado de los documentos en un 16,66% del tiempo que se consume realizándolo de forma manual.

Los resultados arrojados por el experimento son muy satisfactorios para la solución desarrollada, puesto que con su uso se disminuyen los tiempos de cada especialista al realizar el llenado de los documentos.

Se obtuvo una solución funcional, donde los resultados obtenidos de las pruebas realizadas al módulo fueron satisfactorios. Una vez concluida estas y entregada la solución, el sistema fue validado por el compañero Luis Guzmán Hernández, quien ocupa el cargo de director del centro de soporte UCI actualmente, quedando de esta forma avalado, para visualizar el documento que lo demuestra, ver el Anexo 6.

3.5. Conclusiones parciales.

Se presentaron las métricas que fueron empleadas para realizar la validación del diseño, las que ofrecieron como resultado que el diseño está realizado de forma simple y con una calidad aceptable. Se realizó un estándar de codificación y se modeló el diagrama de despliegue que sustentó el proceso de implementación de la solución desarrollada. Se ejecutaron las pruebas demostrando que la solución desarrollada se encuentra lista y cumple con las exigencias del cliente. Con todo esto se demostró el correcto funcionamiento del sistema y el cumplimiento de las expectativas del centro de soporte UCI.

CONCLUSIONES GENERALES.

Tras haber culminado los tres capítulos con los que cuenta el presente trabajo de diploma se arriba a las siguientes conclusiones:

- ✓ Mediante la elaboración del marco teórico y el estudio del estado del arte se permitió sentar las bases para el desarrollo de la investigación identificando conceptos fundamentales, así como herramientas y metodologías, demostrando la necesidad de desarrollar una serie de funcionalidades de apoyo al monitoreo y control de servicios TI en el centro de soporte UCI.
- ✓ La realización del análisis y diseño permitió obtener los artefactos propuestos por la metodología seleccionada para el desarrollo de la solución, obteniéndose entre ellos el diagrama Entidad-Relación y las HU para la especificación funcional, así como tareas de ingeniería para una mejor organización de la implementación, con lo cual se logró una mayor comprensión del negocio y quedaron definidos los requisitos exigidos por el cliente.
- ✓ El uso de la metodología ágil de desarrollo XP, permitió desarrollar en el período de tiempo determinado, un producto con las funcionalidades especificadas por el cliente, conformando un Módulo de Administración para el SAMOS que cumple con las normas establecidas por el marco de trabajo de COBIT.
- ✓ Se elaboró un estándar de codificación completo, que sustentó el proceso de implementación del sistema. Se probaron las funcionalidades de la solución propuesta mediante pruebas de carga y estrés, unitarias y de aceptación, las cuales fueron vencidas satisfactoriamente, así como las métricas que fueron empleadas para realizar la validación del diseño de clases, demostrando así, el correcto funcionamiento del sistema y el cumplimiento de las expectativas del cliente.

RECOMENDACIONES.

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

- ✓ Implementar el sistema para la gestión de otros tipos de servicios, dirigidos a otras áreas de la ciencia y la economía.
- ✓ Implementar la compatibilidad del generador de reportes con formatos exel, doc y xml.
- ✓ Socializar la solución propuesta a varias entidades del país que realicen las tareas de soporte y gestión de servicios.

FUENTES BIBLIOGRÁFICAS

AEC. 2013. *AEC. [Online] 2013. [Cited: Diciembre 1, 2013.] http://www.aec.es/web/guest/centro-conocimiento/cobit.*

ALM Solutions. 2013. *Tu Socio Estratégico en Tecnología de la Información. [Online] 2013. [Cited: Noviembre 25, 2013.] http://www.almsolutions.com.pe/index.php/gestiondeservicios.*

Arión. 2013. *Grupo Arión. [Online] 2013. [Cited: Diciembre 2, 2013.] http://www.grupoarion.com.mx/SolucionesRemedyITSM/RemedyITSMSuite.aspx.*

Beginning, Judd y Christopher, M. 2009.*Groovy and Grails. New York : s.n., 2009.*

BITCompany. 2012. *[Online] 2012. [Cited: diciembre 1, 2013.] http://www.bitcompany.biz/que-es-cobit/.*

Bologna, G. Jack and Walsh, Anthony M. 1997. *Accountant's Handbook of Information Technology. [book auth.] G. Jack Bologna and Anthony M. Walsh. Accountant's Handbook of Information Technology. New York, NY, USA : s.n., 1997.*

Bootstrap. 2012. *framework de twitter. [Online] 2012. [Cited: Diciembre 7, 2013.] http://www.genbetadev.com/frameworks/bootstrap.*

Booch, G, Rumbaugh, J y Jacobson, I. 1999. *El Lenguaje Unificado de Modelado. s.l. : Addison-Wesley, 1999. ISBN: 8478290362 ISBN-13: 9788478290369.*

Christopher M. Judd, J.F.N., James Shingler. 2008.*Beginning Groovy and Grails. From Novice to Professionai. New York, NY : Apress, 2008.*

Copyright GCP Global. 2013. *Software para Gestión de Riesgo, Cumplimiento Normativo y Gobierno Corporativo. [Online] 2013. [Cited: Diciembre 2, 2013.] http://www.gcpglobal.com/english/.*

Cordoba, Rochin, Laguna, Valdivia and Vazquez, Isaac. 2013.*Instituto Politécnico Nacional UPIICSA. 2013.*

Customer Care Associates. 2011.*Sitio de Customer Care Associates. [Online] 2011. [Cited: noviembre 25, 2013.] http://www.customercareassoc.com/index.php?option=com_content&view=article&id=174&Itemid=78.*

Cauzilla Rojas, Leiza. 2013. *Módulo para la evaluación de la composición de equipos de proyectos informáticos en la plataforma GESPRO.* facultad 3. Trabajo Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Don Wells. 2013. *Extreme Programing. [Online] 2013. [Cited: febrero 3, 2014.] http://www.extremeprogramming.org/.*

Fayol, H. 1916.*Administration industrielle et générale. Paris : H. Dunod et E, 1916.*

ISACA. 2013.*COBIT 5 - A Business Framework for the Governance and Management of Enterprise IT | ISACA. [Online] 2013. [Cited: Noviembre 25, 2013.] http://www.isaca.org/COBIT/Pages/default.aspx.*

ISEC. 2013. *Infosecurity News. [Online] 2013. [Cited: Diciembre 2, 2013.] http://www.infosecurityvip.com/newsletter/productos_oct11.html.*

IST. 2009.*Ingienería y Servicios Tecnológicos S.A.C. [Online] 2009. [Cited: Noviembre 30, 2013.] http://www.ist-sac.com/conf1.html.*

Genos Cloud Services S.L. 2013. *Genos. [Online] 2013. [Cited: Diciembre 6, 2013.] http://www.genos.es/soporte/tomcat-soporte.*

GAMMA, Erich. 2013. *Design Patterns: Elements of Reusable ObjectOriented Software. . 2013.*

GCP Global. 2013. *Software para Gestión de Riesgos, Cumplimiento Normativo y Gobierno Corporativo. [Online] 2013. [Cited: Diciembre 2, 2013.] http://www.gcpglobal.com/orca-soluciones.php.*

Chiavenatto, Idalberto. 2000.*Administración: Proceso Administrativo. Tercera Edición. Colombia : McGraw-Hill, 2000.*

Leyva, Francisco I. 2012. *Agrega gráficas interactivas a tu sitio web con Highcharts [JavaScript]. [Online] Enero 31, 2012. [Cited: Diciembre 8, 2013.] http://www.panchosoft.com/blog/2012/01/31/agrega-graficas-interactivas-a-tu-sitio-web-con-highcharts-JavaScript/.*

Luccaco. 2012. *Centro Virtual de Conocimiento para poner fin a la violencia contra las Mujeres y Niñas. [Online] 2012. [Cited: Noviembre 28, 2013.] http://www.endvawnow.org/es/articles/330-cual-es-el-monitoreo-y-la-evaluacion.html.*

Larman, Craig. 1999. *UML Y PATRONES. Introducción al analisis y diseño orientado a objetos. Mexico : s.n, 1999.*

Lorenz, Mark and Kidd, Jeff. 1994. *Object-Oriented Software Metrics.* 1994.

Meltom Technologies. 2013.*Portal de Gerencia y Negocio en HispanoAmerica. [Online] 2013. [Cited: Noviembre 25, 2013.] http://www.degerencia.com/tema/tecnologia_de_informacion.*

Nagios Enterprises, LLC. 2013. *Nagios. [Online] 2013. [Cited: Diciembre 2, 2013.] http://labs.nagios.com/2011/11/15/nagios-xi-benchmarking-experiments/.*

Pressman, Roger S.1998. *Ingeniería de software un enfoque práctico. [En línea].* 1998.

Pressman, Roger S.2010. *Ingeniería del software.* s.l. : McGraw-Hill, 2010. 7.

PMI. 2008.*Guía de los Fundamentos de la Dirección de Proyectos (PMBoK) cuarta. 2008.*

OverTI. 2011. *Consultoría y Soluciones para la Gestión de Servicios TI. [Online] 2011. [Cited: marzo 4, 20114.] http://www.overti.es/procesos-itsm/cobit.aspx.*

R. JOHNSON, J. HOELLER. 2004. *Expert One-on-One™ J2EE™ Development without EJB™.* 2004.

Robert, C. 1994.*Modern Business Administration. Sexta Edición. 1994.*

S.R.L., SCD Servicios Informáticos. 2013. *SCD Servicios Informáticos. [Online] 2013. [Cited: Diciembre 2, 2013.] http://www.scd.com.ar/servicios/monitoreo_preventivo.php.*

Telerik Inc. 2013. *kendo UI. [Online] 2013. [Cited: Diciembre 8, 2013.] http://www.kendoui.com/.*

The PostgreSQL Global Development Group. 2013. *PostgreSQL. [Online] 2013. [Cited: Diciembre 6, 2013.] http://www.postgresql.org/about.*

techwork data gmbh | Erdbergstrasse. 2010. *techwork. [Online] 2010. [Cited: Diciembre 2, 2013.] http://www.techwork.at/products-remedy.html.*

Tierra del Futuro, las Golondrinas – América Latina y UBV. 2013. *La guía metodológica. [Online] 2013. [Cited: Noviembre 28, 2013.] http://www.metoder.nu/cgi-bin/met.cgi?d=s&w=2052&l=es&s=mt.*