Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema para el diagnóstico y seguimiento de riesgo en el centro DATEC.

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

Autor:

Maiker Rodríguez Troche

Tutores:

MSc. Omar Mar Cornelio

Ing. Bárbara Bron Fonseca

La Habana 2015

La revolución no se lleva en los labios para vivir de ella, se lleva en el corazón para morir por ella.

Ernesto Che Guevara

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Maiker Rodríguez Troche

Autor

_____

Omar Mar Cornelio

Tutor

_____

Bárbara Bron Fonseca

Tutor

Datos de contacto

Autor:

Maiker Rodríguez Troche

Universidad de las Ciencias Informáticas.

e-mail: maiker@uci.cu

Tutor:

Omar Mar Cornelio (omarmar@uci.cu)

Universidad de las Ciencias Informáticas.

e-mail: omarmar@uci.cu

Función: Vicedecano Administrativo, vinculado a la docencia, en el curso regular diurno en la disciplina de Sistemas Digitales. Certificado como especialista en calidad para revisiones de Software. Certificado como instalador en Centros de Datos por empresa española EQUINSA.

Estudios realizados: Licenciado en Educación en la Especialidad de Informática, Máster en Informática Aplicada. Ha recibido Curso de postgrado de Planificación de proyectos productivos, Auditores internos para proyectos, Arquitecturas de Redes y Comunicaciones, Sistemas inteligentes, Arquitectura de Bases de datos, Ingeniería de Software y Gestión de Software, Ciencia tecnología y Sociedad, Gestión de la Información, Gestión del Conocimiento, Gestión de Proyecto, Metodología de la Investigación, CPDs en cobre y fibra, Estadística Aplicada, Diseños de experimentos, Publicación de revistas con OJS, Arquitectura de máquinas computadoras.

Experiencia como tribunal de tesis, tutor y oponte en pregrado. Actualmente realiza doctorado en Automática con la Universidad Central de Las Villas.

Resumen

En la Universidad de las Ciencias Informáticas, específicamente el Centro de Tecnologías de Gestión de Datos “DATEC”, se especializa en el desarrollo de aplicaciones informáticas entre las que se pueden contar: el sistema Elecciones, el sistema Generador dinámico de reportes y el proyecto Fiscalía, entre otros. A pesar de la experiencia en cuanto al desarrollo de sistemas y la calidad de los productos presentados por el centro, existen deficiencias en cuanto a la gestión y seguimiento de riesgos de los proyectos.

Actualmente en el centro DATEC la gestión de riesgos se realiza de forma manual, pues a pesar de que existe una herramienta para la gestión de los mismos, no está regida por las directivas establecidas en la Resolución No. 60/11. El hecho de que el proceso de gestión captura y seguimiento de riesgos se realice de forma manual, dificulta y demora el trabajo debido a que se necesita más tiempo para aplicar las acciones correctivas, pues las personas involucradas no son notificadas a tiempo y por tanto las decisiones no se toman oportunamente. Con el objetivo de dar solución a las deficiencias antes mencionadas, se propone el desarrollo de una herramienta que contribuya a erradicar las deficiencias y mejorar el proceso de gestión, diagnóstico y seguimiento de los riesgos en el centro DATEC de la Universidad de las Ciencias Informáticas.

Palabras Clave: Riesgo, Gestión de riesgos, Gestión de datos.

Índice

Introducción 1

Capítulo 1 Fundamentación teórica de la Investigación 6

1.1 Introducción 6

1.2 Conceptos asociados al dominio del problema 6

1.2.1 Proyecto 6

1.2.2 Gestión de Proyectos 6

1.2.3 Riesgo 7

1.2.4 Gestión de riesgos 9

1.3 Clasificación de los riesgos 10

1.4 Estrategias de riesgo 13

1.4.1 Estrategia de riesgo reactiva 13

1.4.2 Estrategia de riesgo proactiva 13

1.5 Procesos para la gestión de riesgos 13

1.5.1 Identificación de riesgos 14

1.5.2 Análisis de riesgos 15

1.5.3 Proceso de planificación 17

1.5.4 Seguimiento y control de riesgos 17

1.6 Análisis de soluciones existentes 17

1.6.1 Active Risk Manager (ARM) 18

1.6.2 Redmine 18

1.6.3 Enterprise Risk Management (ERM) 19

1.7 Tecnologías y tendencias actuales 19

1.7.1 Servidor web 20

1.7.2 Lenguaje de programación 21

1.7.3 Sistema gestor de base de datos 24

1.7.4 Metodologías de desarrollo de software 25

1.7.5 Lenguaje unificado de modelado (UML) 30

1.7.6 Herramienta CASE (*Computer Aided Software Engineering*) 31
1.7.7 Herramientas de desarrollo 32
1.7.8 Framework de desarrollo.....33
1.8 Conclusiones parciales.....36
Capítulo 2: Características del sistema 37
2.1 Introducción.....37
2.2 Descripción del Sistema Propuesto 37
2.3 Modelo del Dominio.....38
2.4 Descripción de los Objetos 39
2.5 Especificación de los Requisitos del Sistema 39
2.5.1 Requisitos Funcionales 40
2.5.2 Requisitos no Funcionales 41
2.6. Actores y Casos de Uso del Sistema.....43
2.6.1 Actores del Sistema 43
2.6.2 Casos de Uso del Sistema.....43
2.6.3 Patrón de Casos de Uso (CRUD).....44
2.6.4 Diagrama de Casos de Uso del Sistema.....45
2.7 Conclusiones parciales.....49
Capítulo 3: Análisis y Diseño del Sistema 50
3.1 Introducción.....50
3.2 Diagrama de Clases del Análisis.....50
3.3 Modelo del Diseño 51
3.3.1Arquitectura del sistema.....51
3.3.2Patrones arquitectónicos.....52
3.3.3 Patrones de diseño 56
3.4 Diagramas de Clases del Diseño (DCD) 59
3.5 Diagramas de interacción 62

3.6 Diseño de la Base de Datos....63
3.7 Diagrama de despliegue....64
3.8 Conclusiones Parciales....65
Capítulo 4: Implementación y Análisis de los resultados....66
4.1 Introducción....66
4.2 Modelo de implementación....66
4.2.1 Diagrama de componentes....66
4.3 Interfaces del sistema....67
4.4 Pruebas....68
4.4.1 Pruebas de seguridad....68
4.4.2 Pruebas de Caja Negra....69
4.5 Conclusiones parciales....71
Conclusiones generales....72
Recomendaciones....73
Bibliografía....74
Anexos....77

Índice de figuras

Figura 1. Fases de la metodología XP. 28
Figura 2. Capas de OpenUp. 29
Figura 3. Fases de OpenUp. 29
Figura 4: Comparación de frameworks java (Tomado de (DRIVE 2010)) 35
Figura 5: Diagrama del Modelo de Dominio creado a partir del problema planteado. 38
Figura 6: Diagrama de Casos de Uso del Sistema. 45
Figura 7: Diagrama de Clase del Análisis CU Autenticar Usuario. 51
Figura 8: Diagrama de Clase del Análisis CU Gestionar Riesgo. 51
Figura 9: Representación gráfica del flujo Cliente-Servidor. 54
Figura 10: Parte del código de la Clase Tb_riesgo. 54
Figura 11: Vista de la Clase Tb_riesgo. 55
Figura 12: Parte del código de la Clase RiesgoController. 55
Figura 13: Ciclo de vida del Patrón MVC (Tomado de (MONOGRAFIAS.COM)). 56
Figura 14: Funcionamiento de un sistema utilizando el patrón MVC (Tomado de (LIBROSWEB)). 56
Figura 15: Patrón Experto en la solución. 58
Figura 16: Aplicación del patrón Creador. 58
Figura 17: Patrón Controlador. 59
Figura 18: DCD CU: Autenticar Usuario. 61
Figura 19: DCD CU: Gestionar Riesgo. 61
Figura 20: DS: Autenticar Usuario. 62
Figura 21: DS: Gestionar Usuario. 63
Figura 22: Modelo de Datos 63
Figura 23: Diagrama de Despliegue. 64
Figura 24: Diagrama de Componentes del sistema. 67
Figura 25: Primera iteración de pruebas. 68
Figura 26: Segunda iteración de pruebas. 69
Figura 27: Tercera iteración de pruebas. 69
Figura 28: Componentes de la capa Modelo. 77
Figura 29: Componentes de la capa Vista. 77
Figura 30: Componentes de la capa Controlador. 78
Figura 31: Pantalla de autenticación del sistema. 78
Figura 32: Descripción del Plan de Contingencia: Mejorar la conectividad. 79
Figura 33: Iteraciones de pruebas de caja negra. 85

Índice de tablas

Tabla 1. Comparación entre servidores Web. 21
Tabla 2. Comparación entre metodologías. 29
Tabla 3: Descripción de las entidades que interactúan en el negocio. 39
Tabla 4: Descripción de los actores del sistema. 43
Tabla 5: Descripción de los casos de uso del sistema. 43
Tabla 6. CU Autenticar Usuario. 45
Tabla 7. CU Gestionar Usuario. 46
Tabla 8: Estereotipos a utilizar en el diagrama de clases del análisis. 50
Tabla 9: Estereotipos utilizados en el Diagrama de Clases del Diseño. 60
Tabla 10: Relaciones de clases del diseño con estereotipos Web. 60
Tabla 11: Caso de Prueba Autenticar Usuario. 79
Tabla 12: Descripción de variables. 83
Tabla 13: SC 2 Adicionar usuario. 83
Tabla 14: SC 3 Modificar usuario. 84
Tabla 15: SC 4 Eliminar usuario. 84

Introducción

En la actualidad, la informática ocupa un rol importante, no solo en la vida diaria de las personas, sino también en el desarrollo de las empresas. El significativo avance de las Tecnologías de la Información y la Comunicación (TIC) ha brindado grandes progresos en diferentes sectores de la actividad humana como: la medicina, la biología, la ingeniería, la industria, la investigación científica, la gestión empresarial, entre otros.

No es extraño que hoy día, la mayoría de las tareas de oficina, en las empresas o incluso en lugares como bancos, hospitales, centros educacionales y agencias de viajes, sean ejecutadas y controladas por sistemas de información con el objetivo de recolectar, procesar, almacenar y distribuir datos en apoyo al control y la toma de decisiones.

El desarrollo alcanzado en cuanto a la conectividad entre las computadoras, las posibilidades de transmisión de datos entre ellas, así como el uso de Internet e Intranet, han aportado numerosas ventajas contribuyendo a facilitar la gestión de la información, pero también trayendo consigo riesgos y amenazas por lo que la Seguridad Informática y sus estándares cobran una importancia vital en el uso de sistemas informáticos conectados a pequeñas o grandes redes.

Para el caso de las entidades estatales cubanas, el 1ro de agosto de 2009, fue aprobada por la Asamblea Nacional del Poder Popular (ANPP) la Ley No. 107 “De la Contraloría General de la República”, como resultado de un proceso de fortalecimiento de la Entidad Fiscalizadora Superior. La creación de la Contraloría General de la República (en lo adelante CGR) forma parte del proceso de institucionalización del país, así como el fomento de la gestión gubernamental. Gracias la creación de este Órgano del Estado una vez más se eleva el rango de las funciones de control del Estado al tiempo que se eliminan dualidades innecesarias en las funciones de control(CONTRALORÍA 2009).

En la sección segunda de la resolución antes mencionada, titulada Gestión y prevención de riesgos, el Artículo 11 plantea: “El componente Gestión y Prevención de Riesgos establece las bases para la identificación y análisis de los riesgos que enfrentan los órganos, organismos, organizaciones y demás entidades para alcanzar sus objetivos. Una vez clasificados los riesgos en internos y externos, por procesos, actividades y operaciones, y evaluadas las principales vulnerabilidades, se determinan los objetivos de control y se conforma el Plan de Prevención de Riesgos para definir el modo en que habrán de gestionarse. Existen riesgos que están regulados por disposiciones legales de los organismos rectores, los que se gestionan según los modelos de administración previstos”(CONTRALORÍA 2011).

Introducción

La Universidad de las Ciencias Informáticas (UCI) ha sido desde sus inicios la mayor protagonista de los logros informáticos del país. Debido a que en Cuba existen pocas empresas que se dediquen al desarrollo de software, la UCI ocupa un lugar importante en la producción de software a nivel nacional.

En la UCI existen diversos Centros de Desarrollo de Software, los cuales están integrados por diferentes Proyectos Productivos. Tal es el caso del Centro de Tecnologías de Gestión de Datos “DATEC”, el cual se especializa en la gestión de datos y el desarrollo de sistemas que manejan bases de datos.

En el centro se han realizado varios proyectos de gran relevancia entre los que se pueden mencionar: el sistema Elecciones y el sistema Generador dinámico de reportes entre otros. Pero a pesar de la experiencia en cuanto a desarrollo de sistemas y la calidad de los productos presentados por el centro, existen deficiencias en cuanto a la gestión y seguimiento de riesgos de los proyectos. Actualmente la gestión de riesgos en dicho centro se realiza de forma manual, pues a pesar de que existe una herramienta para la gestión de los mismos, esta no posee todas las funcionalidades requeridas, permitiendo solo realizar acciones correctivas trayendo como resultado que el proceso de gestión del diagnóstico y seguimiento de riesgos no se realice de manera eficiente. El hecho de que el proceso de gestión captura y seguimiento de riesgos se realice de forma manual, dificulta y demora el trabajo debido a que se necesita más tiempo para aplicar las acciones correctivas, pues las personas involucradas no son notificadas a tiempo y por tanto las decisiones no se toman oportunamente.

La situación descrita anteriormente acarrea otros problemas entre los que se pueden mencionar:

El aumento del margen de errores humanos debido a la mala manipulación de la información derivada del proceso de diagnóstico y seguimiento de riesgos.

Incertidumbre en la gestión de riesgos del centro afectándose el proceso de toma de decisiones.

Inexistencia de registros que posibiliten identificar deficiencias para su corrección oportuna.

Inexistencia de planes de mitigación y contingencia.

Inexistencia de acciones de mitigación y contingencia.

Con la herramienta actual (Gespro) el proceso es lento y engorroso para el usuario.

Los riesgos detectados solo pueden publicarlos los especialistas.

De lo anteriormente expuesto surge el **problema a resolver** ¿Cómo contribuir a una mejor gestión para el diagnóstico y seguimiento de riesgos para el centro de desarrollo DATEC?

Para dar solución al problema planteado surge el **objetivo general:** Desarrollar una herramientaque contribuya a la gestión del diagnóstico y seguimiento de riesgo en el centro DATEC.

Una vez identificado el problema, se hace necesario enfocar la investigación en los sistemas informáticos de información, lo cual constituye el **objeto de estudio** de la misma; definiéndose así como **campo de acción,** los sistemas de informáticos de información para el diagnóstico y seguimiento de riesgos en el centro DATEC.

Partiendo del objetivo general del presente trabajo, surge la siguiente **idea a defender:** Conel desarrollo de un sistema informático, se contribuirá a la gestión del diagnóstico y seguimiento de riesgos, en el centro de desarrollo DATEC.

Para la organización del trabajo se proponen los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación.
2. Diseñar un sistema que permita la gestión del diagnóstico y seguimiento de riesgos según la metodología establecida por la resolución 60 para entidades estatales.
3. Implementar el sistema para el diagnóstico y seguimiento de riesgo en centro DATEC.
4. Validar el sistema mediante pruebas de software.

Para dar cumplimiento a los objetivos trazados, se definen las siguientes **tareas de investigación:**

1. Análisis de la bibliografía científica sobre la gestión de riesgos.
2. Búsqueda de soluciones informáticas existentes para la gestión de riesgos.
3. Selección de las principales herramientas que se ajustan a las necesidades de la investigación para el desarrollo del sistema propuesto.
4. Identificación de las principales funcionalidades del sistema.
5. Diseño del sistema propuesto.
6. Creación de los artefactos para la fase implementación.
7. Implementación del sistema propuesto.
8. Validación de los resultados obtenidos mediantes pruebas para comprobar el cumplimiento de las funcionalidades del sistema.

Métodos investigativos

Para el desarrollo de la presente investigación, se utilizaron los siguientes métodos de la investigación científica:

Histórico-lógico: Se utiliza para el trabajo recopilatorio sobre el proceso de diagnóstico y seguimiento de riesgos y su evolución.

Análisis – síntesis: Se emplea con el objetivo de analizar las tecnologías y herramientas para el desarrollo de sistemas de gestión de la información, así como las metodologías existentes para la construcción de los artefactos para la fase ingenieril del sistema propuesto y extraer los elementos más importantes relacionados con el objeto de estudio.

Entrevista: Aplicada a directivos y especialista del centro DATEC para determinar el estado actual del proceso de gestión y seguimiento de riesgos.

Observación: Permitió a partir de la aplicación de guías de observación en la práctica cotidiana, constatar la existencia del problema en cuestión.

Análisis-Sintético: este método se emplea con el objetivo de analizar las tecnologías y herramientas para el desarrollo de los sistemas de información, las metodologías existentes para la construcción de los artefactos para la fase ingenieril del sistema propuesto; así como extraer los elementos más importantes relacionados con el objeto de estudio de la presente investigación.

Modelación: este método se aplica para la modelación de los componentes y sus dependencias. Se utiliza además para llevar a cabo la realización de los diagramas utilizando UML como lenguaje de modelado.

Posibles resultados:

Sistema informático que permita la gestión del proceso de diagnóstico y prevención de riesgo en el centro DATEC de la Universidad de las de las Ciencias Informáticas.

El trabajo está estructurado en cuatro capítulos, los cuales se describen a continuación:

Capítulo 1: Fundamentación Teórica de la Investigación

Se recogen los principales conceptos que serán tratados durante el desarrollo de la investigación. Se hace referencia a las tendencias actuales, así como las tecnologías, herramientas y la metodología que se aplican durante el proceso de investigación y desarrollo del sistema.

Capítulo 2: Características del Sistema

Se describe las características del sistema a desarrollar. Se define el modelo de dominio, los requisitos funcionales y no funcionales del sistema para comprender mejor su funcionamiento, así como, los actores, diagrama de casos de uso del sistema y el patrón de casos de uso utilizado.

Capítulo 3: Análisis y Diseño del Sistema

Está relacionado con el diseño del sistema; se describen los patrones de diseño empleados. Se presentan los diagramas de clases del diseño, diagramas de interacción y diagrama de despliegue del sistema con el objetivo de tener una idea de cómo quedará.

Capítulo 4: Implementación y Análisis de los Resultados

Está enfocado a la fase de implementación para dar solución a los requerimientos funcionales identificados. Se modela los diagramas de componentes; se describen los estándares, codificación y el tratamiento de errores para darle solución al sistema propuesto.

Capítulo 1 Fundamentación teórica de la Investigación

1.1 Introducción

En este capítulo se plantea la base y la estructura del marco teórico del presente trabajo de diploma. Se realiza una breve descripción sobre la investigación abarcando: los conceptos asociados al dominio del problema, se realiza el estudio del estado del arte de la gestión de riesgos, las principales estrategias usadas para controlar los riesgos, los procesos de la gestión de riesgos y los principales modelos de gestión de riesgos existentes en la actualidad.

Otra de las tareas a realizar en el presente capítulo consiste en seleccionar el modelo para la gestión de riesgosa aplicar en la solución, teniendo en cuenta las características, ventajas y desventajas de los mismos. Se definen además, las tecnologías y herramientas, así como el lenguaje de programación a utilizar en el desarrollo de la misma.

1.2 Conceptos asociados al dominio del problema

En este epígrafe se abordarán los conceptos fundamentales asociados al dominio del problema, los cuales son útiles para un mejor entendimiento del objeto de estudio.

1.2.1 Proyecto

Según varios autores entre los que se encuentran(PMI 2008);(GONZÁLEZ 2000)y (UNESCO 2008)coinciden en que un proyecto es la organización de recursos para obtener determinados objetivos. PMI plantea en la Guía de los Fundamentos de la Dirección de Proyectos (*The Guide to the Project Management Body of Knowledge*, PMBOK Guide, según sus siglas en inglés) que esta organización es temporal(PMI 2008). El sitio de La UNESCO (UNESCO 2008)amplía su definición diciendo que para conseguir esos objetivos se debe contar con un presupuesto.

Por otra parte González Lara(GONZÁLEZ 2000)coincide con lo planteado anteriormente y además agrega en su definición, que para lograr los objetivos definidos son necesarias: autonomía, actividades políticas y medidas institucionales. El proyecto se desarrolla para una región geográfica específica y para un grupo predefinido de beneficiarios, que produce bienes y presta servicios tras la retirada del apoyo externo y cuyos efectos perduran una vez que finaliza su ejecución.

Partiendo de los conceptos analizados anteriormente, para la presente investigación, un proyecto queda definido como un conjunto de inversiones con determinados recursos, un objetivo específico en un período de tiempo determinado, con un presupuesto asignado, en una región geográfica delimitada, diseñada para producir bienes o prestar servicios tras la retirada del apoyo externo.

1.2.2 Gestión de Proyectos

La gestión de proyectos puede definirse como la disciplina encargada de organizar y administrar los recursos de un proyecto de manera tal que se pueda culminar todo el trabajo requerido dentro del alcance, tiempo, y costo definidos inicialmente(PMI 2008).

Es el proceso que se centra en la planificación, implantación y control de las actividades previstas y de los recursos que se utilizan. Es además una hábil utilización de técnicas para alcanzar los resultados esperados(PRESEDO 2010).

Partiendo de los conceptos planteados anteriormente se deduce que la Gestión de Proyectos, comienza al inicio del proyecto y está dada por el conjunto de actividades (planificar, coordinar, organizar, liderar, controlar y verificar) que facilitarán la toma de decisiones durante todo el ciclo de vida del mismo. La misma consiste en un conjunto de procesos, actividades y herramientas, donde juegan un papel importante las prácticas que permiten determinar el estado del proyecto, los riesgos que puedan afectarlo, así como el impacto que estos pueden representar y las estrategias a poner en práctica para prevenir o amortiguar los posibles riesgos en caso de que impacten el proyecto.

La Gestión de Riesgos (en lo adelante GR) del Proyecto incluye los procesos relacionados con llevar a cabo la planificación de la gestión, la identificación, el análisis, la planificación de respuesta a los riesgos, así como su monitoreo y control en un proyecto(PMI 2008).

Algunos conceptos que se pueden citar acerca de la GR son los siguientes:

Enfoque sistemático para reducir la probabilidad de riesgos y/o limitar los daños causados por el riesgo mediante el uso de contramedidas adecuadas o acciones preventivas(MAP 1996).

La Gestión del Riesgo (GR) es una técnica que maneja los recursos que se pueden emplear en el proyecto para limitar la diferencia entre su Estado Final Deseado (EFD) y su Estado Final Conseguido (EFC). Si la diferencia supera un límite establecido, se materializa un riesgo de incumplimiento del objetivo. Para asegurar la pertinencia del resultado suelen requerirse decisiones de realización de nuevas acciones que permitan reducir esa diferencia. Si el EFC está muy alejado del EFD, el proyecto incumplirá el objetivo; hasta su misma consecución puede resultar imposible(J MARCELO COCHO 2003).

Sobre lo anteriormente planteado, los objetivos de la GR son aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos adversos para el proyecto(PMI 2008). Para esto será necesario alcanzar una armonía entre la exposición a los riesgos y la respuesta ante ellos según el costo de aceptarlos, evitarlos, transferirlos, mitigarlos, planear contingencias o ignorarlos.

1.2.3 Riesgo

Según el Diccionario de la Real Academia de la Lengua Española, el riesgo se define como la contingencia o proximidad de un daño y en una segunda acepción, como cada una de las contingencias que pueden ser objeto de un contrato de seguro(RAE 2014).

El riesgo es un evento incierto, indeseable, imprevisto e involuntario que, en caso de producirse, puede tener consecuencias negativas para quien lo sufre y puede generar al mismo tiempo unas necesidades cuantificables económicamente, haciendo peligrar en determinadas ocasiones la estabilidad económico-financiera de la empresa(MARTÍNEZ CARRERA 1998).

Los riesgos surgen de la incertidumbre que rodea a las decisiones y a los resultados de las organizaciones. También es posible que los resultados de una organización no hayan alcanzado las expectativas, por lo que la incertidumbre en la toma de decisiones que han derivado en este resultado también puede considerarse un elemento de riesgo (CAMPOVERDE VÉLEZ 2011).

Otra de las definiciones existentes para el riesgo es la posibilidad de que ocurra un acontecimiento que tenga un impacto en el alcance de los objetivos, por lo cual el riesgo se mide en términos de impacto y probabilidad (DURÁN A. 2007).

Enfocándose en la industria del software, aunque existe un considerable debate en torno a la definición para el riesgo, existe un acuerdo general en que el riesgo siempre involucra dos características(PRESSMAN 1998):

Incertidumbre: El riesgo puede o no ocurrir; esto quiere decir que no existen riesgos 100% probables.

Pérdida: Si el riesgo se convierte en realidad, ocurrirán consecuencias o pérdidas indeseables.

El riesgo es la posibilidad de que un evento adverso, desgracia o contratiempo pueda manifestarse produciendo una pérdida(PRESSMAN 1998).

Por otra parte, el SEI (*Software Engineering Institute*) expresa la definición de riesgo como la posibilidad de sufrir una pérdida, como una medida de la posibilidad en que una amenaza conlleve a una pérdida con un impacto asociado (SEI 2012).

Otra definición de riesgo la brinda Robert Charette en su libro sobre la gestión de riesgo y el análisis cuando plantea:

En primer lugar, el riesgo afecta a los futuros acontecimiento (...) La pregunta es, podemos por tanto, cambiando nuestras acciones actuales, crear una oportunidad para una situación diferente y, con suerte, mejor para nosotros en el futuro. Esto significa, en segundo lugar, que el riesgo implica cambio, que puede venir dado por cambios de opinión, de acciones, de lugares. En tercer lugar, el riesgo implica la elección y la incertidumbre que entraña la elección.

Por tanto, el riesgo, como la muerte, es una de las pocas cosas inevitables de la vida (CHARETTE 1989).

En la mayoría de los casos, el riesgo es considerado un evento negativo con una probabilidad de impacto en el desarrollo un proyecto. Algunos autores, consideran que su ocurrencia genera pérdidas si se hace realidad, pues influyen en el buen desarrollo del proyecto trayendo consigo retrasos en la planificación temporal del proyecto y aumento de los costos.

En los riesgos del proyecto se identifican los problemas potenciales de presupuesto, la planificación temporal, el personal (asignación y organización), los recursos, cliente y requisitos, entre otros.

En algunos casos, los riesgos son considerados como una experiencia que forma parte del aprendizaje y la madurez para gestionar los proyectos de software. Esta experiencia aporta nuevos conocimientos y estrategias para el análisis, clasificación e identificación de riesgos, previendo así mediante la aplicación de una adecuada Gestión de los Riesgos, ser afectados por el impacto de un problema surgido de improviso durante el desarrollo de un proyecto. De igual manera un riesgo puede ser provocado por múltiples causas, o solo una, y de producirse, suele tener uno o más impactos.

El SEI (*Software Engineering Institute*) establece que para que un riesgo sea entendible, debe ser expresado claramente, incluyendo (SEI 2012):

- Una descripción de las condiciones actuales que pueden conducir a la pérdida.
- Una descripción de la pérdida.

La Resolución 60/11 define el riesgo como la incertidumbre de que ocurra un acontecimiento que pudiera afectar o beneficiar el logro de los objetivos y metas de la organización. El riesgo se puede medir en términos de consecuencias favorables o no y de probabilidad de ocurrencia (CONTRALORÍA 2011).

1.2.4 Gestión de riesgos

La gestión de riesgos nace de la necesidad de organizar e interpretar datos. Se aplica cuando la información es limitada y existe incertidumbre para la toma de decisiones. Su aplicación repetida desarrolla las capacidades de las personas y perfecciona el método utilizado(URCELAY 2009).

La Gestión de Riesgos es la práctica compuesta de procesos, métodos y herramientas que posibilita la gestión de los riesgos en un proyecto y que provee de un entorno disciplinado para la toma de decisiones proactiva en base a determinar constantemente qué puede ir mal, identificar los riesgos más importantes en los cuales enfocarse e implementar estrategias para

gestionarlos; se inicia en la primera etapa de un proyecto y se desarrolla a lo largo de su ciclo de vida(MANIASI 2005).

El SEI define la gestión de riesgos como “la práctica compuesta de procesos, métodos y herramientas que posibilita la gestión de los riesgos en un proyecto y que provee de un entorno disciplinado para la toma de decisiones proactiva en base a determinar constantemente que puede ir mal (riesgos), identificar cuáles son los riesgos más importantes en los cuales enfocarse e implementar estrategias para gestionarlos” (SEI 2004).

La Organización de Estándares Internacionales (ISO, por sus siglas en inglés) generaliza la gestión de riesgos como la coordinación de actividades para dirigir y controlar una empresa en relación con el riesgo. Incluye la evaluación, tratamiento, aceptación y comunicación de los riesgos (ISO 2004).

La definición dada por COSO en 2004, señala: “La gestión de los riesgos empresariales es un proceso, efectuado por la dirección de la entidad, directores y demás personal, aplicado a la estrategia y al establecimiento de objetivos y que se desarrolla a través de toda la organización, destinado a identificar los eventos potenciales que pueden afectar la entidad y manejar los riesgos dentro de su apetito de riesgo para proveer una seguridad razonable en el logro de los objetivos de dicha entidad”(COSO 2004).

Según lo establecido en la Resolución 60/11 la gestión y prevención de riesgos establece las bases para la identificación y análisis de los riesgos que enfrentan los órganos, organismos, organizaciones y demás entidades para alcanzar sus objetivos. Una vez clasificados los riesgos en internos y externos, por procesos, actividades y operaciones, y evaluadas las principales vulnerabilidades, se determinan los objetivos de control y se conforma el Plan de Prevención de Riesgos para definir el modo en que habrán de gestionarse. Existen riesgos que están regulados por disposiciones legales de los organismos rectores, los que se gestionan según los modelos de administración previstos.

Partiendo de los conceptos presentados anteriormente, la gestión de riesgos queda definida como la forma de identificar, controlar, planear, organizar y dirigir las actividades realizadas para evitar cualquier desviación en el cumplimiento de los objetivos del proyecto en cuestión y maximizar las oportunidades de éxito del mismo.

1.3 Clasificación de los riesgos

Los riesgos empresariales tienen diferentes clasificaciones las que le han otorgadas a partir de su identificación, trayendo consigo una mejor organización a la gestión de los riesgos. Sin embargo, la clasificación de los riesgos constituye una tarea de gran complejidad debido a los innumerables factores que pueden causarlos.

Según lo expresado por Yadira Rodríguez Carranza, las clasificaciones tratadas a continuación, son las más ajustadas al entorno empresarial cubano. Estos fenómenos, en su mayoría se pueden presentar en las entidades cubanas y son provocados por conductas poco responsables de los trabajadores o por hechos vinculados al objeto social de la entidad(RODRÍGUEZ CARRAZANA 2009).

El riesgo especulativo: es aquel cuyo efecto puede producir una pérdida o una ganancia, como por ejemplo las apuestas o los juegos de azar, las inversiones.

El riesgo puro: es el que se da en la empresa y existe la posibilidad de perder o no perder, pero jamás ganar. El riesgo puro en la empresa se clasifica a su vez en: riesgo inherente y riesgo incorporado.

El riesgo inherente: es propio de cada empresa en dependencia de la actividad que realice, estos son fenómenos producidos por factores objetivos que vienen de la misma naturaleza de la actividad empresarial. Estos riesgos se deben eliminar o controlar de inmediato, pues la existencia de la entidad depende de la actividad que realiza y como estos están en directa relación con la actividad de la empresa, si esta no los asume no puede existir.

El riesgo incorporado: es aquel que no es propio de la actividad de la empresa en cuestión, sino que es producto de conductas poco responsables de un trabajador, el que asume otros riesgos con objeto de conseguir algo que cree que es bueno para él y/o para la empresa, como por ejemplo ganar tiempo, terminar antes el trabajo para destacar, demostrar a sus compañeros que es mejor. Es decir, son riesgos de segundo nivel, que aparecen como resultado de errores o fallas humanas. Este tipo de riesgo se debe eliminar de inmediato.

Para el caso de la industria del software y con el objetivo de cuantificar el nivel de incertidumbre y el grado de pérdidas asociado con cada riesgo, Pressman propone considerar diferentes categorías de riesgos(PRESSMAN 1998):

- Riesgos del proyecto
- Afectan la planificación temporal, el costo y calidad del proyecto.
- Identifican problemas potenciales de presupuesto, calendario, personal, recursos, cliente, requisitos y su impacto en un proyecto de software.
- Riesgos técnicos
- Amenazan la calidad y la planificación temporal del software (producto) que hay que producir.
- Identifican posibles problemas de incertidumbre técnica, ambigüedad en la especificación, diseño, implementación, obsolescencia técnica, interfaz, verificación y mantenimiento.

- Riesgos del negocio
- Amenazan la viabilidad del software a construir.

Los principales riesgos de negocio son:

Riesgo de mercado: construir un producto o sistema excelente que no quiere nadie en realidad.

Riesgo estratégico: construir un producto que no encaja en la estrategia comercial general de la compañía.

Riesgo de ventas: construir un producto que el departamento de ventas no sabe cómo vender.

Riesgo de dirección: perder el apoyo de una gestión experta debido a cambios de enfoque o a cambios de personal.

Riesgo de presupuesto: perder presupuesto o personal asignado.

De igual forma se puede realizar otra categorización de los riesgos teniendo en cuenta su detección:

Riesgos conocidos: Son aquellos que se pueden predecir después de una evaluación del plan del proyecto, del entorno técnico y otras fuentes de información fiables.

Riesgos predecibles: Se extrapolan de la experiencia de proyectos anteriores.

Riesgos impredecibles: Pueden ocurrir, pero es extremadamente difícil identificarlos por adelantado.

Para cada categoría de riesgo existen dos tipos de riesgos diferenciados:

Riesgos Genéricos: son una amenaza potencial para todos los proyectos de software.

Riesgos Específicos: son aquello que solo los pueden identificar los que tienen una clara visión de la tecnología, el personal y el entorno específico del proyecto en cuestión.

Para identificar los riesgos específicos del producto, se examina el plan del proyecto y la declaración del ámbito del software, acto seguido se da respuesta a la siguiente interrogante: ¿Qué características especiales del producto en cuestión pueden estar amenazadas por el plan del proyecto? Es sumamente importante tener en cuenta que tanto los riesgos genéricos como los específicos se deberán identificar sistemáticamente para lograr mitigar su impacto o de ser posible eliminarlos.

Un método que puede utilizarse para identificar riesgos es crear una lista de comprobación de elementos de riesgo. La lista de comprobación se puede utilizar para identificar riesgos y se enfoca en un subconjunto de riesgos conocidos y predecibles en las siguientes sub-categorías genéricas(ASENSIO 2005):

Tamaño del producto: riesgos asociados con el tamaño general del software a construir o modificar.

Impacto en el negocio: riesgos asociados con las limitaciones impuestas por la gestión o por el mercado.

Características del cliente: riesgos asociados con la sofisticación del cliente y la habilidad del desarrollador para comunicarse con el cliente en los momentos oportunos.

Definición del proceso: riesgos asociados con el grado de definición del proceso del software y su seguimiento por la organización de desarrollo.

Tecnología a construir: riesgos asociados con la complejidad del sistema a construir y la tecnología de punta que contiene el sistema.

Entorno de desarrollo: riesgos asociados con la disponibilidad y calidad de las herramientas que se van a emplear en la construcción del producto.

Tamaño y experiencia de la plantilla: riesgos asociados con la experiencia técnica y de proyectos de los ingenieros del software que van a realizar el trabajo.

1.4 Estrategias de riesgo

En el proceso de la gestión riesgo las estrategias trazadas para controlar los mismos se clasifican en estrategias reactivas y proactivas.

1.4.1 Estrategia de riesgo reactiva

La estrategia reactiva es la que reacciona en el momento que ocurren los problemas, para a partir de ahí combatirlos. En el mejor de los casos, la estrategia reactiva supervisa el proyecto en previsión de posibles riesgos. Los recursos se ponen aparte, en caso de que pudieran convertirse en problemas reales, pero lo más frecuente es que el equipo de software no haga nada respecto a los riesgos hasta que algo esté mal. Después el equipo se agiliza para corregir el problema rápidamente. La gestión entra en crisis, encontrándose el proyecto en peligro real(GONZALEZ CEDEÑO 2008).

1.4.2 Estrategia de riesgo proactiva

La estrategia proactiva es la estrategia considerada como la más inteligente para el control del riesgo. Esta empieza mucho antes de que comiencen los trabajos técnicos. Se identifican los riesgos potenciales, se valoran su probabilidad y su impacto y se establece una prioridad según su importancia. Después el equipo de software establece un plan para controlar el riesgo. El primer objetivo es evitar el riesgo, aunque es poco común que todos puedan ser detectados. Entonces el equipo trabaja para desarrollar un plan de contingencia que le permita responder de una manera eficaz y controlada(GONZALEZ CEDEÑO 2008).

1.5 Procesos para la gestión de riesgos

Según la Resolución 60/11, la gestión de riesgos está compuesta por varios procesos entre ellos están: la identificación de riesgos, análisis cualitativo de los riesgos, análisis cuantitativo

de los riesgos, planificación de respuesta a los riesgos y seguimiento y control de los riesgos. A continuación se detallan los procesos antes mencionados (CONTRALORÍA 2011):

1.5.1 Identificación de riesgos

La Resolución 60/11 define que en la identificación de los riesgos, se tipifican todos los que pueden afectar el cumplimiento de los objetivos. La identificación de riesgos se nutre de la experiencia derivada de hechos ocurridos, así como de los que puedan preverse en el futuro y se determinan para cada proceso, actividad y operación a desarrollar(CONTRALORÍA 2011).

Los factores externos incluyen los económico - financieros, medioambientales, políticos, sociales, tecnológicos y los internos incluyen la estructura organizativa, composición de los recursos humanos, procesos productivos o de servicios y de tecnología, entre otros(CONTRALORÍA 2011).

La identificación de riesgos consiste en descubrir los factores de riesgo antes de que estos lleguen a convertirse en problemas y deriven en daños o pérdidas. Es un intento sistemático para especificar las amenazas al plan del proyecto (estimaciones, planificación temporal, carga de recursos, etc.). Identificando los riesgos conocidos y predecibles, el líder del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario. Este es, probablemente, el paso más importante entre todos aquellos que componen las actividades de Gestión de Riesgos, ya que sin la correcta determinación de los mismos, no es posible desarrollar e implementar anticipadamente respuestas apropiadas a los problemas que puedan surgir en el proyecto. El resultado de la identificación de riesgos es una lista conteniendo los riesgos que se han identificados y su categoría correspondiente(GONZALEZ CEDEÑO 2008).

Según plantea Edmundo Pelegrin en “La administración de los Riesgos, su impacto en la empresa”, el proceso de identificar los riesgos debe incluir un ambiente en el que las personas sientan la libertad de expresar puntos de vista especulativos o controversiales. Cuando los riesgos se perciben como algo negativo, los integrantes de un equipo se sienten renuentes a informar sobre ellos. En algunos proyectos, el mencionar los riesgos nuevos se toma como una queja. En ciertas situaciones, una persona que habla de los riesgos recibe el calificativo de conflictiva y las reacciones se encuentran en la persona, antes que en los riesgos. Bajo estas circunstancias, los miembros de un equipo tienen reservas para comunicar sus opiniones con libertad. Seleccionan y suavizan la información de riesgos que deciden compartir, para que no resulte demasiado negativa en relación con las expectativas de los demás integrantes(PELEGRIN PÉREZ 2006).

Como resultado final de este proceso, se obtiene un inventario lo más completo posible de los riesgos a los que está expuesta la organización por áreas, procesos, productos, proyectos.

Este inventario se debe hacer continuamente, al igual que la identificación de los nuevos riesgos, que pueden ir surgiendo o cambiando, al cambiar las condiciones tecnológicas, los requerimientos de seguridad, entre otros factores, por esta razón dicho proceso debe ser dinámico. La actualización permanente del inventario se convierte en una condición necesaria para el logro de un proceso eficaz de gestión de riesgos.

1.5.2 Análisis de riesgos

Luego de identificar, evaluar y cuantificar, siempre que sea posible, los riesgos por procesos, actividades y operaciones, la máxima dirección y demás directivos de las áreas, con la participación de los trabajadores, realizan un diagnóstico y determinan los objetivos de control, dejando evidencia documental del proceso. En la etapa de análisis de riesgos se lleva a cabo la determinación de los objetivos de control que son el resultado o propósito que se desea alcanzar con la aplicación de procedimientos de control, los que deben verificar los riesgos identificados y estar en función de la política y estrategia de la organización(CONTRALORÍA 2011).

Durante esta etapa se examina la lista de riesgos previamente identificados y se les asigna la prioridad, quedando definido el orden final de la lista. Partiendo del análisis de la lista obtenida, se determinan los riesgos más importantes y se reservan recursos para planificar y ejecutar una estrategia específica. Se pueden identificar riesgos que, por su poca prioridad, pueden ser eliminados de la lista. En la medida que el proyecto avance y las circunstancias del mismo vayan cambiando, la identificación y el análisis de riesgos se repiten y la lista de riesgos se irá modificando. Puede que surjan nuevos riesgos y puede que los riesgos más antiguos que han bajado de prioridad se eliminen(QUINTERO ACOSTA 2009).

La metodología de evaluación de los riesgos de una entidad consiste en una combinación de las técnicas cualitativas y cuantitativas. Se aplican técnicas cualitativas cuando los riesgos no se prestan a la cuantificación o cuando no están disponibles datos suficientes y creíbles para una evaluación cuantitativa o la obtención y análisis de ellos no resulte eficiente por su coste. Las técnicas cuantitativas típicamente aportan más precisión y se usan en actividades más complejas y sofisticadas, para complementar las técnicas cualitativas (COSO 2004).

Tanto para el análisis cualitativo como el cuantitativo se calculan mediante la fórmula:

$R = P \times C$

R es el valor del riesgo.

P es la probabilidad o frecuencia de ocurrencia del riesgo.

C es la consecuencia o impacto que puede ocasionar a la organización la materialización del riesgo.

Cuando se realiza el análisis cualitativo de los riesgos, se utilizan formas descriptivas para representar la magnitud de los impactos potenciales y la posibilidad de ocurrencia. La propia entidad diseña una escala la cual es ajustada a las circunstancias, en función de sus necesidades particulares o del riesgo evaluado(COSO 2004).

Por su parte el análisis cuantitativo contempla valores numéricos, dependiéndola calidad de este proceso de lo exactas y completas que estén las cifras utilizadas, la forma en la cual son expresadas la probabilidad y el impacto y la forma en la que pueden ser combinadas para proveer de esta forma el nivel del riesgo que puede variar de acuerdo con el tipo del riesgo.

Las técnicas cuantitativas de evaluación de riesgos pueden utilizarse cuando existe suficiente información para estimar la probabilidad o consecuencia del riesgo, empleando mediciones de intervalo o de razón. Los métodos cuantitativos incluyen técnicas probabilísticas y no probabilísticas. Una consideración importante en la evaluación cuantitativa es la disponibilidad de la información precisa, ya sea de fuentes internas o externas y uno de los retos que plantea el uso de estas técnicas es el de obtener suficientes datos válidos(COSO 2004).

Para realizar correctamente el análisis de riesgos, se deben seguir los siguientes pasos:

Evaluación de la probabilidad e impacto de los riesgos: en este paso se investiga y evalúa la probabilidad de ocurrencia de cada riesgo. La evaluación del impacto de los riesgos investiga el posible efecto sobre un objetivo del proyecto (tiempo, coste, alcance, calidad etc.), en este punto se incluyen tanto los efectos negativos por las amenazas que ellos implican, como los efectos positivos por las oportunidades que los mismos generan. La probabilidad de un riesgo debe ser mayor que cero o el mismo no representa una amenaza para el proyecto.

Categorización de riesgos: los riesgos pueden agruparse en:

Riesgos del proyecto (RP): cuando estos riesgos se hacen realidad, es probable que la planificación temporal del proyecto se retrase y los costos del mismo aumenten.

Los Riesgos técnicos (RT): estos riesgos amenazan la planificación temporal y la calidad del software. Si un riesgo técnico se convierte en realidad, la implementación puede llegar a ser difícil y en algunos casos imposible.

Riesgos del negocio (RN): los riesgos del negocio ocurren cuando el problema es más difícil de resolver de lo anticipado por el equipo. Estos amenazan la viabilidad del proyecto y a menudo ponen en peligro el producto.

Estimar la exposición al riesgo: Una vez identificados y categorizados los riesgos, se prosigue a realizar el análisis de cada riesgo para determinar su impacto. Para ello se multiplica la posibilidad de ocurrencia de un riesgo con el efecto del mismo si se hiciera realidad, de esta

operación se obtiene la exposición que tiene el proyecto al mismo. En otras palabras, se obtiene la vulnerabilidad que tiene el proyecto a ese riesgo.

Priorización del riesgo: Asignar la prioridad de cada riesgo identificado es de suma importancia brinda da una visión clara a la hora de desarrollar el plan de mitigación.

Evaluación urgente de riesgos: Los riesgos que requieren respuestas a corto plazo pueden ser considerados como más urgentes.

1.5.3 Proceso de planificación

La planificación tiene como principal objetivo desarrollar un plan para la mitigación de los riesgos más importantes del proyecto según lo definido por la estrategia de la Gestión de Riesgos. Si el equipo de trabajo adopta un enfoque proactivo frente al riesgo, evitarlo será siempre la mejor estrategia. Esto se consigue desarrollando los planes de reducción del riesgo y de contingencia(QUINTERO ACOSTA 2009).

Las actividades para este proceso son:

Revisar los parámetros de los riesgos.

Determinar los niveles y los umbrales que definen cuando un riesgo llega a ser inaceptable y cuando debe comenzar la ejecución del plan de mitigación del riesgo o de un plan de contingencia.

Identificar a personas o grupos responsables de tratar cada riesgo.

Desarrollar un plan total de mitigación de riesgos para el proyecto que permita organizar la puesta en práctica de los planes individuales de mitigación y contingencia de cada riesgo.

Supervisar el estado del riesgo.

Proponer estrategias de manipulación de riesgos.

Actualizar el registro de riesgos y el plan de gestión.

1.5.4 Seguimiento y control de riesgos

Durante el seguimiento y control de riesgos se identifican, analizan y planifican nuevos riesgos, se realiza el seguimiento de los riesgos identificados y se plantean alternativas de cómo controlar los riesgos que puedan aparecer en el entorno a medida que el proyecto avanza. El seguimiento y control de riesgos es un proceso que se realiza continuamente durante la vida del proyecto y determina(HECHAVARRIA GUIBERT 2008):

Si las asunciones del proyecto aún son válidas.

Si el riesgo, según fue evaluado, ha cambiado de su estado anterior, a través del análisis de tendencias.

Si se están siguiendo políticas y procedimientos de Gestión de Riesgos correctos.

1.6 Análisis de soluciones existentes

Con respecto a la Gestión de Riesgos, existen numerosas herramientas de software disponibles en el mercado, las cuales mayormente presentan como deficiencias que se enfocan solo en una categoría de riesgos (TRIMS – *Technical Risk Identification and Mitigation System*) o están orientadas a grandes compañías las que poseen una amplia base de datos organizacional que les permite generar información de categorías propias de riesgos (*RiskTrak* y *Welcome Risk*).

Dentro de las herramientas existentes y utilizadas a nivel internacional se encuentran:

1.6.1 Active Risk Manager (ARM)

ARM es un software de gestión del riesgo basado en la web del "proyecto/programa/portafolio /empresa ampliamente desplegado". ARM permite la identificación, análisis, manejo y monitoreo de los riesgos, las oportunidades y los problemas/incidentes, tanto cuantitativa como cualitativamente. ARM ofrece una visibilidad, con seguridad, a toda la información relacionada con el riesgo a través del programa/cartera/organización para mejorar en última instancia, los resultados del programa, optimizar los recursos, y asegura un proceso de gestión de riesgo común, estandarizada e integrada y la cultura que se adopten con rapidez y eficacia. ARM es fácil de usar, con múltiples interfaces de usuario basadas en web, promueve la adopción y ayuda a infundir una cultura de gestión de riesgos de gestión transformadora impulsada por el riesgo de una capacidad en una ventaja competitiva (STG 2010).

La interfaz rediseñada y fácil de usar de ARM, se puede configurar para adaptarlas a las necesidades de distintas comunidades de usuarios que proveen una imagen apropiada para cada nivel de la organización. Esto ofrece a los clientes de ARM y sus socios la posibilidad de medir los estados y los problemas de cumplimiento de una manera sencilla y coherente(STG 2010).

1.6.2 Redmine

Redmine es una aplicación web flexible de gestión de proyectos. Escrito utilizando el framework *Ruby on Rails*, es multiplataforma y *cross-database*.*Redmine* es de código abierto y liberado bajo los términos de la v2 Licencia Pública General de GNU (GPL) (LANG 2014)(PELEGRIN PÉREZ 2006).

Dentro de las principales características de Redmine se encuentran:

Apoyo de múltiples proyectos

Permite gestionar todos los proyectos con una instancia Redmine.

Cada usuario puede tener un papel diferente en cada proyecto.

Cada proyecto puede ser declarado como público (visible por cualquier persona) o privado (visible por los miembros del proyecto).

Módulos (por ejemplo wiki, repositorio, seguimiento de problemas, etc.) se pueden activar o desactivar a nivel de proyecto.

Apoyo múltiple a subproyectos.

Permite definir roles de forma personalizada y establecer los permisos de manera sencilla.

Soporte Múltiple de autenticación LDAP

Redmine permite autenticar usuarios contra LDAP múltiple.

Las cuentas pueden ser creadas sobre la marcha cuando un usuario se encuentra en el directorio (opcional).

Soporte de múltiples plataformas de base de datos

Redmine soporta MySQL, PostgreSQL o SQLite.

1.6.3 Enterprise Risk Management (ERM)

ERM tiene entre sus funciones Identificar, definir y mejorar las actividades clave del negocio, entendiendo sus riesgos y sus impactos en la generación de valor, en los procesos y en el desempeño organizacional. *SoftExpert* ERM *Suite* (*Enterprise Risk Management*) brinda un apoyo para identificar los indicadores clave de riesgo (KRI) y alinear los eventos de riesgo con su impacto potencial, para facilitar y apoyar la toma de decisiones en relación a la gestión de riesgo, independiente del tamaño de la empresa o segmento de actuación.

Al proporcionar conformidad con la norma ISO 31000, COSO ERM y muchas otras metodologías de gestión de riesgo, *SoftExpert* ERM Suite apoya a las empresas en mantener una mirada atenta sobre sus riesgos organizacionales en todos los niveles y a obtener respuestas sobre cómo los riesgos pueden afectar el valor de negocio y la reputación. Con la integración con la solución de *Business Process Management* (BPM), es posible embeber prácticas operacionales de gestión de riesgo y auditoría de procesos en la cultura corporativa, haciendo que los procedimientos sean más eficaces y eficientes y transformando la gestión de riesgo y conformidad en una herramienta de gestión estratégica(SOFTEXPERT 2013).

Valoración del autor

Las descritas anteriormente aunque posibilitan la gestión de los riesgos de un proyecto, están patentizadas por empresas privadas, además se enfocan en una sola categoría de riesgos, de esta forma, el proceso de la gestión de riesgos resulta incompleto pues existen varias categorías de riesgo y además siempre pueden surgir otras nuevas y cada una de ellas se enfrenta de manera diferente. Por las razones antes expuestas quedan descartadas todas las herramientas analizadas como posible solución a la problemática tratada en la investigación.

1.7 Tecnologías y tendencias actuales

En este epígrafe se realizará la selección de las tecnologías y herramientas a utilizar en la implementación de la solución.

1.7.1 Servidor web

Existen numerosos servidores web dentro de los que se encuentran: Microsoft IIS, Java System Web Server, APACHE, etc.

Microsoft IIS

El de servidor web (IIS) proporciona una plataforma segura, fácil de administrar, modular y extensible para el hospedaje seguro de sitios web, servicios y aplicaciones. IIS permite compartir información con usuarios en Internet, una intranet o una extranet. Dicho servidor constituye además una plataforma web unificada que integra IIS, ASP.NET, servicios de FTP, PHP y *Windows Communication Foundation* (WCF)(TECHNET 2012).

Dentro de las principales aplicaciones prácticas de IIS se encuentran:

Mediante el Administrador de IIS se pueden configurar las características del y administrar sitios web.

El protocolo de transferencia de archivos (FTP) permite a los propietarios de sitios web cargar y descargar archivos.

Permite el aislamiento de sitios web para evitar que un sitio web interfiera con otros sitios instalados en el servidor.

Permite configurar aplicaciones web escritas en distintas tecnologías, como ASP clásico, ASP.NET y PHP.

Mediante el *Windows Power Shell* se puede automatizar la administración de la mayoría de las tareas administrativas del servidor web.

Permite configurar varios servidores web en una granja de servidores, que puede administrar mediante IIS.

Apache

Apache server es software libre y el servidor web más popular. Algunas investigaciones realizadas han arrojado que más del 70% de los sitios web en internet están manejados por Apache, haciéndolo más extensamente usado que todos los otros servidores web juntos(OPENSUSE 2014).

Entre las características más relevantes de Apache se encuentran:

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos HTTP.

Multiplataforma.

Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.

Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

Entre las ventajas que presenta el servidor Apache se encuentran:

La arquitectura modular de Apache es personalizable.

Permite construir un servidor hecho a la medida.

Permite la implementación de los últimos y más nuevos protocolos.

A continuación se muestra una tabla comparativa entre los servidores web estudiados.

Comparación entre servidores				
Servidor Web.	Aspectos			
	Multiplataforma	Open source	Modular	Gratuito
Apache	X	X	X	X
IIS	-	-	X	-

Tabla 1. Comparación entre servidores Web.

Según el estudio realizado se escoge Apache en su versión 2.0 como servidor web para el desarrollo del presente trabajo, pues además de las ventajas de ser *OpenSource,* gratis y de fácil adquisición, Apache es el servidor web más empleado en los centro de desarrollo de la UCI por los que los trabajadores están familiarizados con su uso.

1.7.2 Lenguaje de programación

Entre los varios lenguajes de programación existentes de código abierto para desarrollar aplicaciones orientadas a la web se encuentran dos grupos fundamentales de acuerdo con la arquitectura Cliente/Servidor: está la programación del lado del servidor y la programación del lado del cliente.

La programación del lado del cliente incluye aquellos lenguajes que son únicamente interpretados por una aplicación cliente como el navegador Web, entre estos lenguajes se encuentran HTML, Java Script, Visual Basic y Java. Los lenguajes de programación del lado servidor son los que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Entre ellos se encuentra: PHP, ASP, PERL etc.

HTML (*HyperText Markup Language*)

Es un lenguaje sencillo desarrollado por el *World Wide Web Consortium*. Es el lenguaje estático fácil de aprender que permite preparar documentos web insertando en el texto de los mismos,

marcas que controlan los distintos aspectos de la presentación y el comportamiento de sus elementos. Es un lenguaje común para la construcción de una página Web donde el texto es presentando de forma estructurada y agradable. Se caracteriza por poseer archivos pequeños y despliegue rápido además de que es admitido por todos los exploradores. El lenguaje HTML es extensible, se le pueden añadir características, etiquetas y funciones adicionales para el diseño de páginas web, generando un producto vistoso, rápido y sencillo.

ASP.NET

Desde hace algún tiempo, Microsoft está llevando adelante una estrategia para construir una nueva tecnología tendiente a crear aplicaciones web distribuidas y que aprovechen al máximo las posibilidades que ofrece Internet. Esta tecnología, lleva el nombre de .NET, y que incluye un nuevo lenguaje denominado C#, una nueva versión de Visual Basic, con el nombre de Visual Basic.Net y otra serie de tecnologías, entre las que se encuentra: ASP.NET, que viene a reemplazar a las *Active Server Pages* (ASP), logrando el desarrollo de aplicaciones web más dinámicas, con un código más claro y limpio, por ende reusable, multiplataforma y definitivamente más simple, ya que el entorno ASP.NET permite la creación automática de alguna de las tarea más comunes para un creador web, cómo los formularios o la validación de los datos(DANISOFT 2014).

PHP (*HyperText Preprocessor*)

PHP es un lenguaje de programación gratuito, interpretado, usado normalmente para la creación de páginas web dinámicas. Es un lenguaje rápido, con una gran librería de funciones y una amplia documentación, se encuentra incluido dentro de los lenguajes del lado del servidor.

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML ("*HyperText Markup Language*"). Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje, su escasez de consumo de recursos y que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales (PHP 2014).

Entre las numerosas ventajas que tiene PHP se encuentran que es un lenguaje multiplataforma, con capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, capacidad de expandir su potencial al utilizar una enorme cantidad de módulos (llamados ext's o extensiones), posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos, permite las técnicas de Programación Orientada a

Objetos, biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables, tiene manejo de excepciones, además, está basado en el lenguaje C++ y la sintaxis usada es muy similar a C/C++ el cual es considerado un buen lenguaje de programación por muchos programadores(PROGRAMACIÓNWEB.NET 2014).

Groovy

Groovy es un lenguaje de programación con una sintaxis análoga a Java que compila para Java *Bytecode* y corre en la JVM[1]. Se integra completamente con Java, le permite mezclar y acoplar código Groovy y Java con un esfuerzo mínimo (ABDUL-JAWAD 2009).

Groovy es un lenguaje dinámico, es decir, todo ocurre en tiempo de ejecución (*Runtime*), incluida la lógica de gestión de llamadas a métodos y acceso a propiedades. Este tampoco es interpretado o compilado pues es diseñado específicamente para la plataforma Java. Ha sido influenciado por lenguajes como Ruby, *Python*, Perl, y *Smalltalk*, así como también Java. A diferencia de otros lenguajes que se adaptaron para la JVM, Groovy fue diseñado para la JVM, así es que no existe incompatibilidad(CHRISTOPHER M. JUDD 2008).

La sintaxis de *Groovy* es más flexible y poderosa que la de Java. Las docenas de líneas de código en Java pueden acortarse a pocas líneas de código en *Groovy* ganando en legibilidad, mantenibilidad y eficiencia. Algunas personas se refieren a *Groovy* como un lenguaje *scripting*, aunque es mucho más que eso. Es un lenguaje orientado a objeto(ABDUL-JAWAD 2009).

Sus creadores lo conciben como: "...una súper versión de Java. Puede explotar las capacidades empresariales Java pero además junto a su *framework*[2]*Grails* presenta características frescas y modernas para el desarrollo eficiente de aplicaciones Web, como los closures[3]...". También ha resultado una gran plataforma para conceptos como lenguajes de meta programación[4] y DSL5[5] (*Domain Specific Languages*), sintaxis breve adaptada y específica al dominio del problema (ABDUL-JAWAD 2009).

Para el desarrollo del presente trabajo se selecciona Groovy en su versión 2.0.7 como lenguaje de programación, pues al utilizar Groovy como lenguaje se aprovechan todas las ventajas de la tecnología Java y se agregan algunas frescas como: el tipado dinámico y estático, el soporte nativo a listas, mapas, expresiones regulares. Además, Groovy es un lenguaje dinámico, que

[1] JVM (Java Virtual Machine) Máquina Virtual de Java.

[2] Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

[3] Trozo de código empaquetado como un objeto y definido entre llaves. Actúa como un método, al cual se le pueden pasar parámetros y pueden devolver valores.

[4] Programas que modifican otros programas o a sí mismos (modificación del código desde el código).

[5] "Mini-lenguaje" que permite modelar el conocimiento en un ámbito concreto de forma mucho más sencilla que con un lenguaje de propósito general.

disminuye cantidad de líneas de código respecto al lenguaje Java e incluye muchas nuevas funcionalidades en su GDK (Análogo al JDK de Java).

1.7.3 Sistema gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD) o DBMA (*Data Base Management System*) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Está compuesto por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, además de garantizar la seguridad e integridad de los mismos(CAVSI 2014).

MySQL

MySQL es un sistema multiplataforma de manejo, creación y administración de base de datos O*penSource* (*Database Management System, DBMS*) para bases de datos relacionales. Cuenta con un completo sistema multihilo, que ofrece un soporte completo para diferentes formas de manera eficiente y veloz, permitiendo acceder a todos los campos que resguardan los datos de trabajo(MYSQL 2014).

Características de MySQL:

Cuenta con la capacidad de realizar tareas multiprocesador, debido a que posee la opción de trabajo multihilo.

Puede ingresar una enorme cantidad de datos por columna de trabajo.

Cuenta con varias API disponibles para los principales lenguajes de programación que existen.

Aplicación con una portabilidad sobresaliente.

Capacidad de soportar hasta 32 índices de tablas diferentes.

Un nivel de seguridad que permite gestionar varios usuarios mediante contraseñas individuales.

Sistema de sub-consultas un poco arcaico en relación a opciones más modernas, lo que implica que el desarrollador tenga que buscar opciones más complicadas para solventar esta situación.

PostgreSQL

PostgreSQL es un SGBD relacional, orientado a objetos y libre, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales.

Dentro de sus principales ventajas se encuentran:

Soporta distintos tipos de datos.

Posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que atienden un gran número de solicitudes.

Puede ser instalado un número ilimitado de veces sin temor de sobrepasar la licencia.

Posee estabilidad y confiabilidad legendaria.

Es extensible a través del código fuente, disponible sin costos adicionales.

Soporte nativo para los lenguajes más populares del medio: PHP, C++, Perl, Python, entre otros.

Extensiones para alta disponibilidad, nuevos tipos de índices, datos especiales, minería de datos, entre otros.

Es multiplataforma, disponible en Linux, Unix, MacOsX y Windows, entre otros(SCRIBID 2014).

Se seleccionó PostgreSQL v8.3, debido a que es confiable, estable, con gran escalabilidad, control de concurrencia y funcionalidades que lo destacan como uno de los SGBD más potentes en la actualidad. Las ventajas y prestaciones brindadas por este SGBD son amplias y ajustables al objetivo propuesto. PostgreSQL es además un sistema multiplataforma, lo que posibilita el constante perfeccionamiento de sus funcionalidades y está publicado al amparo de una licencia BSD la cual pertenece a las licencias de software libre.

1.7.4 Metodologías de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software(CARRILLO P. 2014).

Según Jacobson y Booch, la metodología de desarrollo hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos que rigen en una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Alternativamente puede definirse la metodología como el estudio o elección de un método pertinente para un determinado objetivo(BOOCH 2005).

RUP *(Rational Unified Process)*

RUP es una metodología guiada por casos de uso, es centrada en la arquitectura, iterativa e incremental. Dicha metodología plantea que el primer paso hacia la división del proceso de desarrollo de software, consiste en separar las partes en cuatro fases atendiendo al momento en que se realizan: inicio, elaboración, construcción y transición(BOOCH 2005).

Las características del ciclo de vida de RUP son:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, estos son captados cuando se modela el negocio y representados a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los

modelos que obtenidos como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

Iterativo e Incremental: RUP propone que cada fase sea desarrollada en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

El proceso de desarrollo de **RUP** está dividido en ciclos, teniendo un producto funcional al final de cada ciclo. Cada ciclo se divide en fases que finalizan con un hito donde se toma una decisión importante, las cuatro fases que incluye RUP son:

Inicio: el objetivo en esta etapa es determinar la visión del proyecto.

Elaboración: en esta etapa el objetivo es determinar la arquitectura óptima.

Construcción: en esta etapa el objetivo es obtener la capacidad operacional inicial.

Transición: el objetivo es llegar a obtener el release[6] del proyecto.

XP *(Extreme Programing)*

La metodología *Extreme Programing* (XP) o Programación Extrema, es una de las variantes de las metodologías ágiles más destacadas y con más aceptación en la comunidad internacional de desarrollo. Es una metodología 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. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico(PENADÉS 2014).

La metodología se basa en:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se podrá hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

[6] Liberación del proyecto.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

XP se basa en cuatro valores imprescindibles para el desarrollo del software:

Simplicidad: enfocado en un diseño sencillo de código generado.

Comunicación: potenciada por el desarrollo en pares, la presencia del cliente y la simplicidad en cuanto al código.

Retroalimentación: propiciada por el protagonismo del cliente que participa activamente y por el trabajo en ciclos cortos.

Coraje: enfrentando decisiones, en ocasiones complejas, que pudieran afectar el tiempo de desarrollo y la calidad del producto.

El ciclo de vida ideal consta de 6 fases:

Exploración: los clientes plantean a grandes rasgos las Historias de usuario que son de interés para la primera entrega del producto.

Planificación de Entregas: se establece la prioridad de cada Historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas.

Iteraciones: incluye varias iteraciones sobre el sistema antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas.

Producción: requiere de pruebas adicionales y revisiones del rendimiento antes de que el sistema sea trasladado al entorno del cliente.

Mantenimiento: mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones.

Muerte del proyecto: un proyecto entrara en esta fase cuando el cliente no tenga más historias de usuario para ser incluidas en el sistema(PENADÉS 2014).

Figura 1. Fases de la metodología XP.

OpenUp

Open Up es un proceso modelo, extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños o de bajos recursos, aplicable a un conjunto amplio de plataformas de desarrollo. Basada en la metodología de Proceso Unificado de Rational (*Rational Unified Process*) RUP. Es el subconjunto de esta última que contiene el conjunto mínimo de prácticas que ayudan a un equipo de desarrollo de software a realizar un producto de alta calidad. Utiliza un punto de vista pragmático, una filosofía ágil que se centraliza en la naturaleza colaborativa del proceso de desarrollo del software. Una de sus principales características es su alto grado de adaptabilidad a las necesidades de un proyecto en particular.

OpenUp es un proceso de desarrollo iterativo del software que es mínimo, completo y extensible. Es mínimo en que solamente el contenido fundamental es incluido; es completo en que puede ser manifestado como todo el proceso para construir un sistema; extensible en que puede ser utilizado como fundamento sobre el cual el contenido de proceso se pueda agregar o adaptar según lo necesitado(EPF 2011).

Como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Es el proceso unificado que aplica acercamientos iterativos e incrementales dentro de un ciclo vital estructurado (EPF 2011). En la Figura 2 se muestran las capas de *OpenUp*.

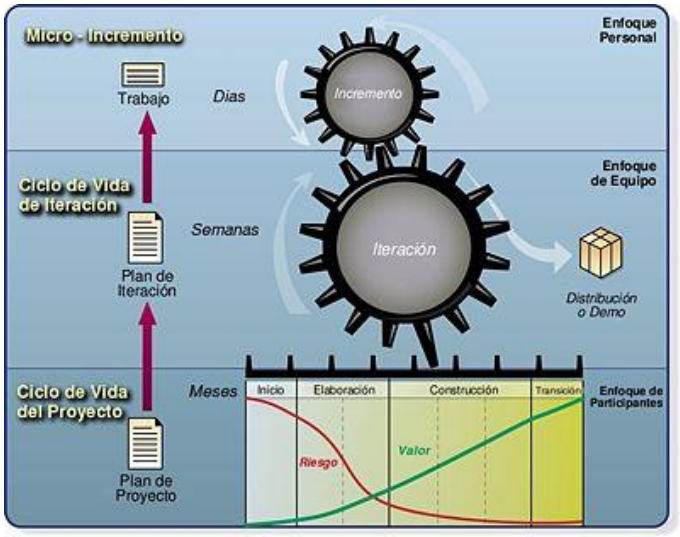


Figura 2. Capas de OpenUp.

El OpenUp estructura el ciclo de vida de un proyecto en cuatro fases: inicio, elaboración, construcción y transición. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos. A continuación en la Figura 3 se muestran las fases de OpenUp.

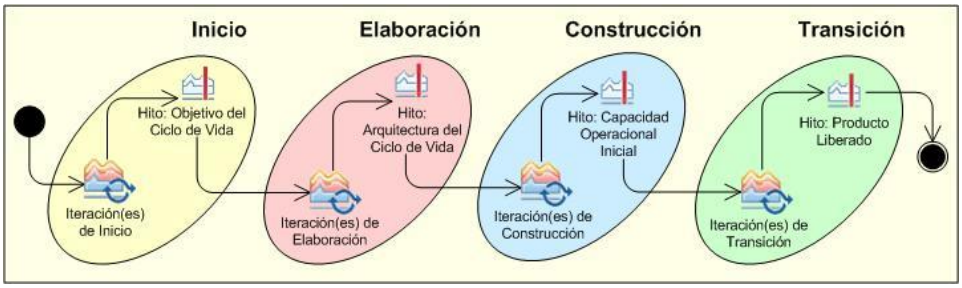


Figura 3. Fases de OpenUp.

Tabla 2. Comparación entre metodologías.

Comparación entre Metodologías de desarrollo			
Metodologías	**Características**	**Tipo de Método**	**Cantidad de personal**
RUP	Forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién	Pesado	Requiere un equipo de programadores grande.

	hace qué, cómo y cuándo).		
XP	Centrado en simplificar el desarrollo del software y lograr reducir el costo del proyecto.	Ligero	Requiere un equipo de programadores pequeño entre 2 – 15.
OpenUP	Es mínimo en que solamente el contenido fundamental es incluido; es completo en que puede ser manifestado como todo el proceso para construir un sistema; extensible en que puede ser utilizado como fundamento sobre el cual el contenido de proceso se pueda agregar o adaptar según lo necesitado.	Ligero	Es una metodología apropiada para proyectos pequeños o de bajos recursos

Luego de realizar el análisis de las metodologías anteriormente mencionadas se decidió escoger para el proceso de desarrollo OpenUp, debido a que es iterativo e incremental. Es apropiado para proyectos pequeños y de bajos recursos. Permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito. Permite detectar errores tempranos a través de un ciclo iterativo. Evita la elaboración de documentación, diagramas e iteraciones innecesarias. Por ser una metodología ágil tiene un enfoque centrado al cliente con iteraciones cortas. Es ligero y proporciona una comprensión detallada del proyecto, beneficia a clientes y desarrolladores sobre el producto a entregar y su formalidad. Es extensible ya que el proceso se puede agregar o adaptar según lo vayan requiriendo los sistemas.

1.7.5 Lenguaje unificado de modelado (UML)

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y modos de disponer dichos símbolos para modelar un diseño de software orientado a objetos. Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores(GARCÍA 2014).

UML es un lenguaje muy conocido y utilizado en el modelado de sistemas de software, permite visualizar, especificar, construir y documentar el sistema a desarrollar. Ofrece un estándar para detallar un plano del sistema (modelo), el cual incluye características conceptuales como procesos de negocio, funciones del sistema, aspectos concretos como expresiones de lenguaje de programación, esquemas de bases de datos, entre otros(RUMBAUGH 2000).

Debido a las características anteriormente mencionadas, se escoge UML 2.0 como lenguaje de modelado pues presenta un conjunto de herramientas que permiten modelar (analizar y diseñar) sistemas orientados a objetos, con la modelación de los de artefactos durante las primeras fases del ciclo de vida del software, los implementadores, en fases posteriores tendrán mayor dominio comprensión sobre qué es lo que se debe implementar, permitiendo tanto al cliente como a los desarrolladores tener una representación real del alcance y factibilidad que puede llegar a tener el producto.

1.7.6 Herramienta CASE (*Computer Aided Software Engineering*)

Las herramientas CASE brindan soporte para desarrollar y mantener software. Son herramientas individuales que ayudan al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software(EUI-FI 2014).

Para el desarrollo del presente trabajo se selecciona la herramienta CASE *Visual Paradigm*.

Visual Paradigm

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML(FREEDOWNLOADMANAGER 2014).

Características:

Soporte de UML versión 2.1.

Diagramas de Procesos de Negocio.

Ingeniería inversa.

Modelo a código, diagrama a código.

Diagramas de flujo de datos.

Soporte ORM.

Generación de bases de datos.

Transformación de diagramas de Entidad-Relación en tablas de base de datos.

Distribución automática de diagramas.

1.7.7 Herramientas de desarrollo

Las herramientas de desarrollo de software permiten a los desarrolladores la creación de aplicaciones para sistemas concretos. Las más comunes incluyen técnicas de soporte para la detección de errores de programación, la generación automática de código, entre otras características que facilitan y agilizan el desarrollo de soluciones de software. Frecuentemente incluyen también códigos de ejemplo, notas técnicas y documentación de soporte(HERNÁNDEZ SÁNCHEZ 2014).

NetBeans

NetBeans es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero se utiliza para desarrollar en varios lenguajes de programación(NETBEANS 2014).

Características:

NetBeans es un Entorno de Desarrollo Integrado (IDE) de código abierto y gratuito.

Es multiplataforma, haciendo posible su uso en diversos sistemas operativos.

Brinda las herramientas necesarias para el desarrollo en lenguajes como Java, XML, HTML, PHP y Java Script.

Presenta aplicaciones para la detección y tratamiento de errores.

Tiene una amplia comunidad de desarrollo y soporte.

IntelliJ IDEA

IntelliJ IDEA es un entorno de desarrollo Java creado por *Jet Brains* del que existen dos distribuciones: *Comunity Edition* (*open source*) y *Ultimate* (comercial). Sus creadores definen este IDE como el más inteligente del mundo. La mayoría de gente que lo prueba lo define como el mejor entorno de desarrollo Java que existe. IntelliJ IDEA es considerado como un sobresaliente marco de trabajo específico de asistencia de codificación y características que aumentan la productividad de Java EE, Spring, GWT, Grails, Play y otros marcos, junto con herramientas de implementación para la mayoría de los servidores de aplicaciones (INTELLIJIDEA 2014).

Dentro de las ventajas aportadas por dicho IDE se encuentran:

Posee autocompletado de código.

Integración con sistemas de control de versiones.

Posee un amplio set de plugins listos para usar desde el momento de la instalación.

Posee una herramienta de refactorización extremadamente inteligente.

Para llevar a cabo la solución propuesta, se emplea como IDE de desarrollo IntelliJ IDEA, debido a que el mismo posee gran compatibilidad con el framework de desarrollo y el lenguaje de programación seleccionados.

1.7.8 Framework de desarrollo

Un *framework* o marco de trabajo en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. No es más que una base de programación que atiende a sus descendientes (manejado de una forma estructural y/o en cascada) posibilitando cualquier respuesta ante las necesidades de sus miembros, o secciones de una aplicación web(MILANÉS LÓPEZ 2009).

Spock

Spock es un framework de pruebas y especificación para aplicaciones Java y Groovy. Una de sus principales características es el alto nivel de expresión a través de la especificación. Spock es compatible con la mayoría de los IDEs, herramientas de construcción y servidores de integración continua. Está inspirado en: JUnit, jMock, RSpec, Groovy, Scala, entre otras tecnologías(ALCOLEA 2012).

Dentro de sus principales características se encuentran:

Fácil de aprender: Si utilizas Java, Groovy y JUnit; la curva de aprendizaje es pequeña.

Groovy: Permite sacar provecho de un lenguaje como Groovy para hacer más cosas en menos código.

Flexibilidad: Se adapta a las necesidades: pruebas de unidad, pruebas de integración, test-driven, behavior-driven.

Especificación: Permite expresar ideas a través de un lenguaje de especificación altamente expresivo.

Lecciones del pasado: Toma lo mejor de JUnit, jMock y RSpec.

Groovy Trapestry

Groovestry es un framework diseñado para integrar el lenguaje de script Groovy y el framework de aplicaciones web de Tapestry. Las características principales de esta integración son las siguientes(MAVEN 2004):

- Soporta el lenguaje de scripts de Groovy.
- Soporta el uso de componentes y páginas según las especificaciones Tapestry.

- Los scripts groovy se encuentran localizados en relación con las especificaciones de Trapestry que referencia dichos scripts.
- Compila ScriptsGroovy a medida que son referenciados.
- Detecta cambios en los scripts Groovy y los recompila según sea necesario.
- Totalmente integrado con Línea Precisa de Informe de Errores.
- Los scripts Groovy son almacenados en lugares distintos de los archivos de sistema.
- Soporta scripts Groovy desplegados en la producción como archivos class.

Wicket

Wicket es un framework java orientado a componentes, desarrollado por Jonathan Locke, el cual posee las siguientes características(LOCKE 2014):

- Posee componentes POJO (del inglés *Plane Old Java Object*[7]).
- Los componentes escritos en Wicket son completamente reutilizables.
- Es compatible con cualquier editor HTML.
- Soporta todas las características básicas de HTML.
- Simplificación entre enlace HTML/Java.

Grails

Grails nació en el verano de 2005, es un framework de desarrollo web dinámico para la plataforma Java. Utiliza la flexibilidad de Groovy para proporcionar un dominio específico del lenguaje (DSL) para el desarrollo web. El objetivo es desarrollar aplicaciones web con el mínimo esfuerzo, sin tener que repetirse. Grails proporciona un entorno coherente y fiable entre todos sus proyectos (DICKINSON 2009). Las aplicaciones desarrolladas con este framework utilizan el patrón MVC (Modelo-Vista-Controlador).

Este no es un framework más para Java, es un framework que toma lo mejor de esta tecnología. Fue construido con los siguientes puntos (ABDUL-JAWAD 2009).

Convención sobre configuración: Al usar el principio de convención, no se necesita realizar labores de configuración sobre algún archivo, pues Grails hace las configuraciones por convención, ahorrando tiempo al desarrollador (ABDUL-JAWAD 2009).

No reinventar lo bueno, sino mejorarlo: Se trata de tomar lo mejor de las tecnologías y adaptarlas con mejoras. Por ejemplo, *Hibernate* es un buen *ORM* (*object - relationalmapping*), con poderosas y avanzadas herramientas. Grails crea un simple DSL, que simplifica el trabajo

[7]POJO: acrónimo de *Plain Old Java Object* utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial

con *Hibernate*, dejando fuera las complicadas configuraciones de mapeo en archivos XML (ABDUL-JAWAD 2009).

A continuación se muestra una comparación entre varios frameworks java:

Criteria	Wicket	Tapestry	Grails
Developer Productivity	0.50	1.00	1.00
Developer Perception	1.00	0.50	1.00
Learning Curve	0.50	0.50	1.00
Project Health	1.00	1.00	1.00
Developer Availability	0.50	1.00	0.50
Job Trends	0.50	0.50	0.50
Templating	1.00	1.00	1.00
Components	1.00	1.00	0.50
Ajax	0.50	0.50	0.50
Plugins or Add-Ons	1.00	0.50	1.00
Scalability	0.50	0.50	0.50
Testing	0.50	1.00	1.00
i18n and l10n	1.00	1.00	1.00
Validation	1.00	1.00	1.00
Multi-language Support (Groovy / Scala)	1.00	1.00	1.00
Quality of Documentation/Tutorials	0.50	0.50	1.00
Books Published	0.50	0.50	1.00
REST Support (client and server)	0.50	0.50	1.00
Mobile / iPhone Support	1.00	1.00	1.00
Degree of Risk	1.00	1.00	1.00
Totals	15	15.5	17.5

Figura 4: Comparación de frameworks java (Tomado de (DRIVE 2010))

Luego de realizado el estudio de frameworks existentes, para el desarrollo de la solución propuesta en la presente investigación, se decide emplear como framework de desarrollo Grails. Mediante la integración de dicho framework con el lenguaje de programación seleccionado anteriormente (Groovy), el cual está construido para la madura y robusta tecnología Java, permite a los desarrolladores Web la construcción de sistemas robustos, desarrollados en un entorno coherente y fiable, con un mínimo de esfuerzo. Además posibilita el empleo de la reutilización de código, la meta programación, etc. Obteniéndose sistemas modulares desarrollados bajo el patrón Modelo Vista Controlador (MVC). Entre otras características se tiene además que soporta tecnología Ajax, es multiplataforma y cuenta con una gran comunidad de desarrolladores que lo mantiene en constante avance.

1.8 Conclusiones parciales

Luego de realizar un estudio del marco teórico y conceptual y abordar los conceptos fundamentales relacionados con la gestión de riesgos se identifica el modelo para la gestión de riesgo propuesto por la contraloría como la solución óptima para implementar en el centro DATEC.

El análisis y valoración de las soluciones existentes, posibilitó determinar que las mismas no realizan el proceso completo de gestión de riesgos, reafirmando de esta forma la necesidad de implementar una herramienta para diagnosticar y dar seguimiento a los riesgos en el centro DATEC de la UCI.

Del análisis sobre las tendencias de aplicaciones para el desarrollo de software se seleccionaron las tecnologías y herramientas a utilizar bajo el principio de ser software libre quedando definido como servidor web Apache 2.0, Groovy 2.0.7 como lenguaje de programación, como sistema gestor de base de datos PostgreSQL 8.3, como metodología de desarrollo OpenUP, lenguaje de modelado UML 2.0, Visual Paradigm como herramienta CASE, como IDE de desarrollo IntellijIDEA y como *framework* Grails.

Capítulo 2: Características del sistema

2.1 Introducción

El presente capítulo está dedicado a las características del sistema. Se abordan temas relacionados con la propuesta de solución para responder a la situación problemática en cuestión, además quedarán definidos los requisitos no funcionales así como las principales funcionalidades del sistema. Las funcionalidades se describen mediante los casos de uso del sistema. Se lleva a cabo la realización del plan de entrega, donde se indican las historias de usuario que se crearán para cada versión de la aplicación propuesta, así como las fechas en las que se publicarán dichas versiones.

2.2 Descripción del Sistema Propuesto

El presente trabajo tiene como finalidad, implementar de manera ordenada y sistemática los procesos que dan solución a todo tipo de riesgo detectado, asociado a los diferentes proyectos que existen en el centro DATEC. La herramienta propuesta consiste en un sistema de gestión orientado a la web, el cual permite el diagnóstico y seguimiento de los riesgos. El sistema cuenta con varios módulos, los cuales desempeñan diferentes funciones.

El sistema propuesto emplea el principio de seguridad informática control de acceso basado en roles (RBAC), además cuenta con un registro de los diferentes riesgos detectados en cada proyecto del centro, así como las diferentes acciones para la mitigación de los mismos. Mediante la realización de los planes de mitigación y contingencia, los integrantes de cada proyecto, cuentan con las herramientas necesarias para actuar de forma inmediata ante la detección de riesgos que puedan afectar al proyecto en curso. Por otra parte y con el objetivo de prevenir la ocurrencia de riesgos se lleva a cabo el monitoreo de los riesgos lo cual consiste en la revisión del correcto funcionamiento de los componentes que integran la administración de riesgos del centro mediante actividades de supervisión permanentes. A continuación se listan las principales funcionalidades brindadas por el sistema las cuales se detallan posteriormente en el presente documento.

Principales funcionalidades del sistema:

- Autenticar usuario.
- Gestionar usuario.
- Gestionar proyecto.
- Gestionar riesgo.
- Determinar exposición al riesgo.
- Gestionar plan de mitigación.

- Gestionar plan de contingencia
- Gestionar acciones de mitigación.
- Gestionar acciones de contingencia.

2.3 Modelo del Dominio

En un modelo de dominio se capturan los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, trayendo consigo la ventaja de ayudar a usuarios, clientes y desarrolladores a utilizar un vocabulario común para entender el contexto en que se desarrollará el sistema.

La gestión de riesgos se encarga de definir los procesos que se manipulan en un negocio. Es necesario tener en cuenta que al no tener un cliente definido, no es posible establecer un patrón o similitud en cuanto a las características que se desarrollan en el entorno de los proyectos productivos del centro, para de esta forma manipular y controlar los riesgos asociados a cada uno de estos proyectos.

Con el objetivo de definir el flujo de eventos que se realiza, teniendo como punto de partida la detección de un riesgo en cada proyecto del centro DATEC y llegando hasta la elaboración de los planes de contingencia y mitigación asociados a dicho riesgo, se realiza la concepción del diagrama de dominio. Este modelo permite identificar las funcionalidades necesarias para que el sistema propuesto realice la gestión de riesgos definida por los proyectos de dicho centro. El modelo de dominio permitirá identificar los principales conceptos asociados al problema en cuestión y la relación existente entre ellos.

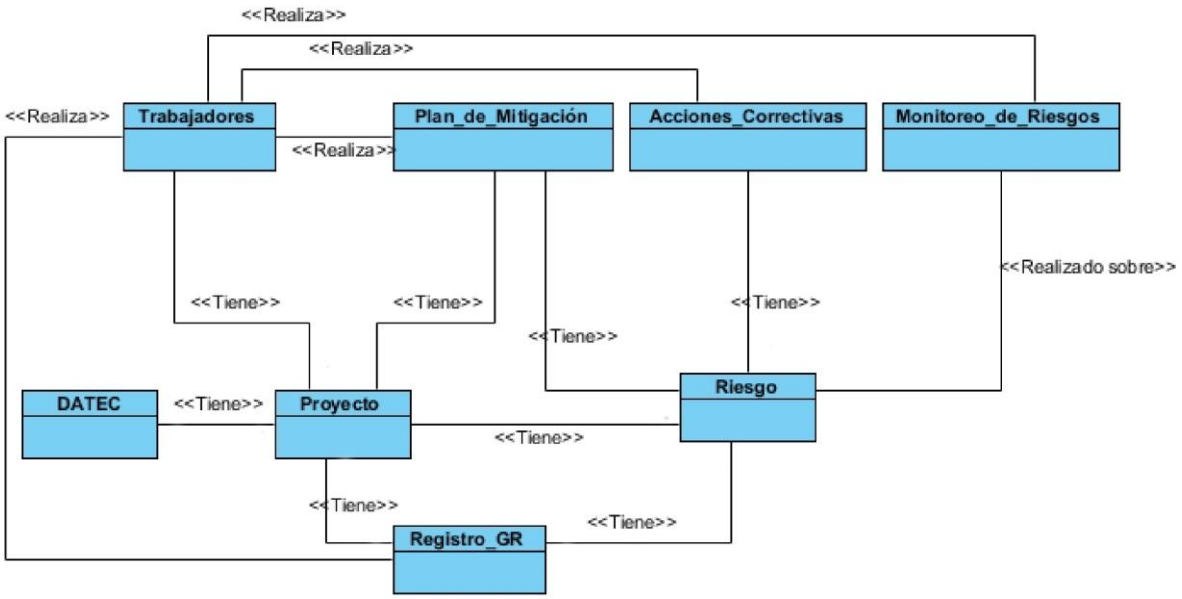


Figura 5: Diagrama del Modelo de Dominio creado a partir del problema planteado.

2.4 Descripción de los Objetos

A continuación se muestra la Tabla 2.1 con la definición de las entidades que interactúan en el negocio y los principales conceptos que se tratan en el problema que se analiza.

Tabla 3: Descripción de las entidades que interactúan en el negocio.

Entidad	Descripción
DATEC	Es el centro de desarrollo al cual se asocian diferentes proyectos productivos.
Proyecto	Representa a cada proyecto del centro DATEC. A cada proyecto se asocian uno o varios riesgos y a su vez representa el entorno en el que se va a aplicar la gestión de los riesgos.
Trabajadores	Representa la generalización de los trabajadores del centro DATEC, asociados a los diferentes proyectos.
Riesgo	Representa la probabilidad de que una amenaza que tendrá un impacto en los objetivos del proyecto se materialice.
Registro GR	Representa el registro de gestión de riesgos llevado en cada proyecto, consiste en un documento en el cual son registradas todas las características de cada uno de los riesgos detectados.
Acciones correctivas	Representa cada acción realizada encaminada a corregir los problemas causados por un riesgo.
Plan de mitigación	Representa el conjunto de acciones a realizar encaminadas a reducir la ocurrencia de un riesgo.
Monitoreo de Riesgos	Representa la revisión del funcionamiento de los componentes que integran la administración de riesgos del centro mediante actividades de supervisión permanentes, las cuales permiten calcular la exposición a los riesgos detectados, así como la calidad real de su control interno.

2.5 Especificación de los Requisitos del Sistema

Para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es

tan ardua como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas.

La calidad con que se realice la captura de los requisitos va a influenciar en todo el proceso de desarrollo del software repercutiendo en el resto de las fases del mismo. Una definición eficiente de los requisitos permite mostrar un nivel de disciplina en el proceso de desarrollo, dar un mejor soporte a la Gestión de Cambios y ganar una mayor eficiencia en las pruebas reduciendo el riesgo, mejorando la calidad y permitiendo la automatización. Además contribuye a tomar mejores decisiones de diseño y de arquitectura. También le permite al equipo de desarrollo reducir los problemas de mantenimiento.

2.5.1 Requisitos Funcionales

Los requisitos funcionales enuncian la definición de los servicios que un sistema debe proveer o su comportamiento ante las diferentes entradas y situaciones que le sean presentadas(PRESSMAN 1998). A continuación se muestran los requisitos funcionales del sistema.

Requisitos Funcionales

1. Autenticar usuario.

- **1.1**. Iniciar sesión.
- **1.2**. Cerrar sesión.

2. Gestionar usuario.

- **2.1**. Insertar usuario.
- **2.2**. Asignar permisos.
- **2.3**. Actualizar usuario
- **2.4**. Eliminar usuario.

3. Gestionar Departamento

- **3.1**. Insertar departamento.
- **3.2**. Modificar departamento.
- **3.3**. Obtener lista de departamentos.
- **3.4**. Eliminar departamentos.

4. Gestionar proyecto.

- **4.1**. Insertar proyecto.
- **4.2**. Modificar un proyecto.
- **4.3**. Obtener lista de proyectos.
- **4.4**. Eliminar proyecto.

5. Gestionar riesgo.

5.1. Insertar riesgo.

5.2. Actualizar riesgo.

5.3. Obtener lista de riesgos.

5.4. Eliminar riesgo.

6. Determinar exposición al riesgo.

7 Gestionar plan de mitigación de riesgos.

7.1. Insertar plan de mitigación de riesgos.

7.2. Modificar plan de mitigación de riesgos.

7.3. Actualizar plan de mitigación de riesgos.

7.4. Eliminar plan de mitigación de riesgos.

8. Gestionar plan de contingencia.

8.1. Insertar plan de mitigación de riesgos.

8.2. Modificar plan de mitigación de riesgos.

8.3. Actualizar plan de mitigación de riesgos.

8.4. Eliminar plan de mitigación de riesgos.

9. Gestionar acciones de mitigación.

9.1. Insertar acción de mitigación.

9.2. Actualizar acción de mitigación.

9.3. Obtener listado de acciones de mitigación.

9.4. Eliminar acción de mitigación.

10. Gestionar acciones de contingencia.

10.1. Insertar acción de contingencia.

10.2. Actualizar acción de contingencia.

10.3. Obtener listado de acciones de contingencia.

10.4. Eliminar acción de contingencia.

2.5.2 Requisitos no Funcionales

Los RNF especifican propiedades del sistema que el sistema debe cumplir, como estricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad. Tienen que ver con las características del sistema, que incluye también interfaces de usuario(PÉREZ QUINTERO 2008).

Para un correcto funcionamiento el sistema debe tener como características principales para su funcionamiento los siguientes requisitos no funcionales:

Usabilidad: El sistema puede ser utilizado por cualquier persona que tenga al menos conocimientos básicos en el manejo de la computadora, navegación y exploración de los sitios Web en sentido general, las operaciones se realizan con bajo nivel de complejidad.

Disponibilidad: El sistema garantiza sus servicios los 5 días hábiles de la semana, mientras la infraestructura (servicio eléctrico, de redes, etc.) del centro lo permita.

Seguridad: Se establecen niveles de acceso, de esta forma se garantiza que la aplicación será utilizada correctamente por cada usuario según sus respectivos niveles de acceso.

Integridad: Se debe tratar el manejo de la información de forma tal, que la información no sea modificada por personal ajeno. Esto evitará alteraciones en los resultados planteados en la documentación.

Interfaz: El diseño de la interfaz visual debe ser amigable e intuitiva para los usuarios que interactúan con la aplicación, de forma tal que permita el fácil entendimiento de las funcionalidades que brinda, además de poseer colores amigables y refrescantes para una mejor interacción entre el usuario y la aplicación.

El diseño de la interfaz permitirá mostrar mensajes para la guía de usuarios en caso de errores en entradas inválidas de los datos o de confirmación de realización de actividades.

Cliente:

Hardware:

Se necesita como requerimiento mínimo una Computadora Personal (*Personal Computer*) que posea una tarjeta de red de 10 MB o superior.

Software:

Necesita un sistema operativo y un navegador Web.

Servidor:

Hardware:

Se necesita como requerimiento mínimo una Computadora Personal (*Personal Computer*)

Memoria RAM de 1GB o superior.

HDD de 80 GB o superior.

Tarjeta de red de 100 MB o superior.

Software:

Necesita un Sistema Operativo.

Servidor Web Apache 2.2.4 o superior.

Servidor de Bases de Datos PostgreSQL 9.1 o superior.

PHP 5.2.3.

Rendimiento: La disponibilidad de trabajo en red contra el servidor es constante. Se garantiza que la respuesta a solicitudes de los usuarios, sea en un período de tiempo breve para evitar la acumulación de trabajo por parte de los responsables. El sistema deberá ser lo más estable y confiable posible.

Restricciones en el diseño y la implementación: Es un sistema Web desarrollada con la tecnología Groovyy base de datos en *PostgreSQL*.

2.6. Actores y Casos de Uso del Sistema

Un actor del sistema es una entidad externa, que representa el rol de una o varias personas, un equipo o un procedimiento automatizado que interactúa con el sistema, por lo que no forma parte del mismo. Puede intercambiar información o ser un recipiente pasivo de información.

Un caso de uso del sistema es una técnica para la captura de requisitos potenciales y a su vez una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

2.6.1 Actores del Sistema

En la Tabla 3 se describen los actores que interactúan en el sistema:

Tabla 4: Descripción de los actores del sistema.

Actores	Descripción
Administrador	Es el encargado de la gestión de usuarios, de activos fijos, áreas, centros de costo y organizaciones.
Trabajadores	Tienen permisos especiales en el sistema, poseen el rol de *"usuario avanzado".*
Usuario	Solamente puede visualizar los datos publicados en el sistema.

2.6.2 Casos de Uso del Sistema

A continuación en la Tabla 4 se describen los casos de usos del sistema:

Tabla 5: Descripción de los casos de uso del sistema.

Orden	Nombre del Caso de Uso	Prioridad	Descripción
2	Gestionar Usuario	ALTO	Puede ser accedido por el administrador del sistema del mismo. Permitiéndole crear, eliminar y modificar los usuarios que accederán al sistema, así como definir para cada uno los roles o permisos que tendrán dentro del mismo.

1	Autenticar Usuario	ALTO	Constituye un elemento de seguridad dentro del sistema, admite que solo los usuarios registrados puedan acceder y navegar dentro del sistema, de acuerdo con el permiso (rol) asignado.
4	Gestionar Riesgo	ALTO	Funcionalidad dentro del sistema considerada de vital importancia para la existencia del mismo. Se encarga crear, modificar y eliminar los riesgos de cada proyecto productivo.
5	Determinar exposición al riesgo	ALTO	Se encarga de determinar el nivel de exposición que existe dentro de un proyecto a un riesgo determinado.
8	Gestionar acción de mitigación.	ALTO	Esta funcionalidad es la encargada de crear, modificar y eliminar las acciones de mitigación de riesgo dentro del proyecto.
9	Gestionar acción de contingencia	ALTO	Esta funcionalidad es la encargada de crear, modificar y eliminar las acciones de contingencia dentro del proyecto.
3	Gestionar proyecto	MEDIO	Solo puede ser ejecutado por el Administrador del sistema, constituye la funcionalidad encargada de crear y eliminar un proyecto en el centro DATEC.
10	Gestionar departamento	MEDIO	Permite al administrador del sistema, crear y eliminar departamentos del centro.
6	Gestionar plan de mitigación.	MEDIO	Funcionalidad que permite realizar los planes de mitigación de los diferentes proyectos del centro.
7	Gestionar plan de contingencia.	MEDIO	Funcionalidad que permite realizar los planes de contingencia de los diferentes proyectos del centro.

2.6.3 Patrón de Casos de Uso (CRUD)

Los patrones de casos de uso son comportamientos que deben existir en el sistema, ayuda a describir qué es lo que el sistema debe hacer, es decir, describe el uso del sistema y cómo este interactúa con los usuarios. Estos son utilizados generalmente como plantillas que especifican

como deberían ser estructurados y organizados los casos de uso y capturan mejores prácticas para modelar casos de uso.

Los patrones de casos de uso brindan los siguientes beneficios:

- Aumentar la productividad.
- Reutilizar elementos existentes.
- Evitar el re trabajo por errores.
- No invertir tiempo en resolver problemas ya resueltos.
- Aplicarla teoría al trabajo práctico.
- Habilitar las herramientas de soporte para modelar el desarrollo.
- Durante el diseño de los casos de uso del sistema se utilizó el patrón CRUD (acrónimo de Crear, Obtener, Actualizar y Borrar del original en inglés: *Create, Read, Update and Delete*).

2.6.4 Diagrama de Casos de Uso del Sistema

Gracias a las facilidades que ofrece el UML, se procede a capturar los requisitos funcionales del sistema y a representar a los mismos mediante un diagrama de casos de uso (DCU).

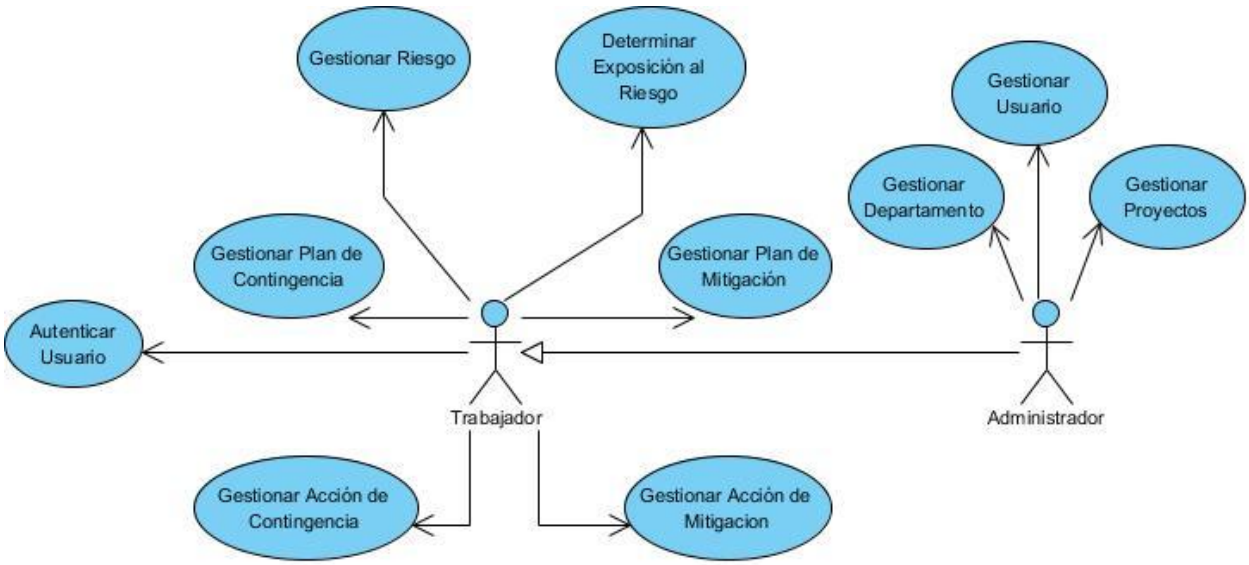


Figura 6: Diagrama de Casos de Uso del Sistema.

2.6.5Descripciones de los Casos de Uso del Sistema

A continuación se muestra la descripción de los casos de uso del sistema (CUS) Autenticar Usuario y Gestionar Usuario.

Tabla 6. CU Autenticar Usuario.

Caso de Uso	Autenticar Usuario
Actores	Usuario

Resumen	Se inicia cuando el usuario ingresa sus credenciales para acceder al sistema. Los datos serán verificados a través del servicio Web de autenticación de la UCI; de ser correctos, el acceso se habilita al sistema en dependencia de los permisos que tenga habilitado su rol. El caso de uso termina cuando el usuario accede al sistema.
Precondiciones	
Referencias	RF 1
Prioridad	Alto

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando los usuarios acceden al sistema. 3. El usuario introduce sus credenciales.	2. El sistema brinda la posibilidad de introducir usuario y contraseña. 4. El sistema verifica que los campos del formulario no estén vacíos. 5. Se verifica que el usuario exista en la BD y que la contraseña especificada sea correcta. 6. Si los datos son correctos, la aplicación muestra la interfaz principal, donde aparecen las funcionalidades según el rol que tenga.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1. De existir un campo vacío el sistema muestra un mensaje de error y regresa a la actividad 1 del flujo normal de los eventos. 5.1. Si los datos no son correctos, la aplicación muestra un mensaje de error, y le permite al usuario volver a introducir las credenciales.
Pos condiciones	El usuario queda autenticado.

Tabla 7. CU Gestionar Usuario.

Caso de Uso	Gestionar Usuario
Actores	Administrador
Resumen	El caso de uso inicia cuando el administrador selecciona en el menú la

<table>
<tr><td></td><td colspan="2">opción Gestionar Usuario. El sistema muestra las opciones de agregar, buscar, modificar y eliminar usuario. El administrador realiza las operaciones que desee y de esta forma se gestionan los usuarios. El caso de uso termina.</td></tr>
<tr><td>Precondiciones</td><td colspan="2">El administrador tiene que estar autenticado.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF 2</td></tr>
<tr><td>Prioridad</td><td colspan="2">Alto</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El administrador selecciona la opción gestionar usuario</td><td colspan="2">2. El sistema muestra el listado de usuarios existentes brinda la posibilidad de Agregar, Buscar, Modificar o Eliminar usuario.</td></tr>
<tr><td colspan="3">Sección “Agregar Usuario”</td></tr>
<tr><td>3. El administrador Selecciona la opción agregar usuario.

5. El administrador introduce los datos y selecciona la opción guardar.</td><td colspan="2">4. El sistema muestra los datos correspondientes:
• nombre de usuario
• rol
• proyecto productivo al que pertenece.
Y brinda la opción:“guardar usuario “para guardar los datos del nuevo usuario o Cancelar: ver Alternativa 1: “Cancelar Operación ”
6. Valida los datos, si hay campos incompletos: ver Alternativa 2: “Hay campos vacíos”, si hay campos incorrectos: ver Alternativa 3: “Hay campos incorrectos”.
7. El caso de uso termina.</td></tr>
<tr><td colspan="3">Flujos Alternos “Cancelar Operación”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Selecciona la opción de Cancelar operación.</td><td colspan="2">2. Regresa a la vista anterior.
3. El caso de uso termina.</td></tr>
<tr><td colspan="3">Flujos Alternos “Hay campos vacíos”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td></td><td colspan="2">1. Muestra un indicador al lado de los campos vacíos.</td></tr>
<tr><td colspan="3">Flujos Alternos “Hay campos incorrectos”</td></tr>
</table>

Acción del Actor	Respuesta del Sistema
	1. Muestra el mensaje de error "Existen campos incorrectos" 2. Muestra un indicador sobre los campos incorrectos.
Sección "Buscar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción Buscar Usuario. 3. El administrador introduce los criterios de búsquedas.	2. El sistema muestra los datos correspondientes: • nombre de usuario • rol • proyecto productivo al que pertenece. 4. El sistema muestra el resultado de acuerdo con los criterios de búsquedas. 5. El caso de uso termina.
Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona el campo que desea modificar haciendo doble clic sobre él.	2. Muestra un cuadro de texto para introducir el nuevo valor que tendrá el usuario en ese campo. 3. El sistema envía el nuevo valor para ser procesado. 4. El caso de uso termina.
Sección "Eliminar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona un usuario de la lista y luego la opción Eliminar Usuario. 3. En el mensaje Presiona el botón Sí.	2. Muestra un mensaje de confirmación de eliminación. 4. Envía la información del usuario y se elimina. 5. El caso de uso termina.
Flujos Alternos: "Cancelar operación"	
Acción del Actor	**Respuesta del Sistema**
1. En el mensaje Presiona el botón No.	2. Regresa a la vista anterior. 3. El caso de uso termina.
Pos condiciones	Se gestionaron los datos de los usuarios de manera correcta.

2.7Conclusiones parciales

- Como resultado del estudio realizado, queda definida *OpenUp* como la metodología a utilizar, se modelaron los artefactos que genera la misma entre los que se encuentran: modelo de dominio, actores y casos de uso del sistema, descripciones de los mismos, así como el levantamiento de los requisitos funcionales y no funcionales.
- Se identificaron los casos de uso del sistema en correspondencia con las funcionalidades recogidas en los requisitos funcionales.
- Se utilizó el patrón CRUD para estructurar y organizar los casos de uso, lo que definió el diagrama de casos de uso del sistema y la descripción detallada de cada uno.
- Luego de quedar definido el diseño del sistema y las relaciones entre sus elementos, es posible dar comienzo el proceso de implementación del sistema propuesto en la presente investigación.

Capítulo 3: Análisis y Diseño del Sistema

3.1 Introducción

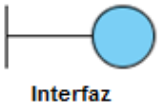
En el presente capítulo se abordan los temas relacionados al análisis y el diseño de la aplicación, luego de realizado el análisis para llevar a cabo la implementación del sistema, las tareas de esta fase dan vida a lo que posteriormente constituirá la arquitectura base del software. Por esta razón que se proponen los patrones de arquitectura, patrones de diseño y se hace una propuesta de la arquitectura base que brindará soporte a la aplicación.

El análisis consiste en obtener una visión detallada del sistema, de modo que solo se interesa por los requisitos funcionales, para refinarlos y estructurarlos, además es el principal eslabón para el comienzo de las actividades de diseño e implementación. El objetivo del análisis, se centra en comprender perfectamente los requisitos del software y no en precisar cómo se implementa el producto.

3.2 Diagrama de Clases del Análisis

El diagrama de clase del análisis se realiza para cada caso de uso del sistema. Muestra las clases participantes en dichos casos de uso, así como la relación entre ellas. En dichos diagramas identifican tres tipos de clases: Interfaz, Controladora y Entidad. En la siguiente tabla se representan los estereotipos a utilizar en el diagrama de clases del análisis.

Tabla 8: Estereotipos a utilizar en el diagrama de clases del análisis.

Clases	***Descripción***	***Representación***
Interfaz	Las clases *interfaz* se utilizan para modelar la interacción entre el sistema y sus actores.	
Entidad	Las clases *entidad* se utilizan para modelar información que poseen larga vida y que es a menudo persistente.	
Controladora	Las clases *controladora* representan coordinación, secuencia, transacciones y control de objetos y son utilizadas para encapsular el control de un caso de uso.	

Diagramas de clases del Análisis

A continuación se muestran los diagramas de clases del análisis de los casos de uso Autenticar Usuario y Gestionar Riesgo, el resto de los diagramas se encuentran en los anexos del presente documento.

Figura 7: Diagrama de Clase del Análisis CU Autenticar Usuario.

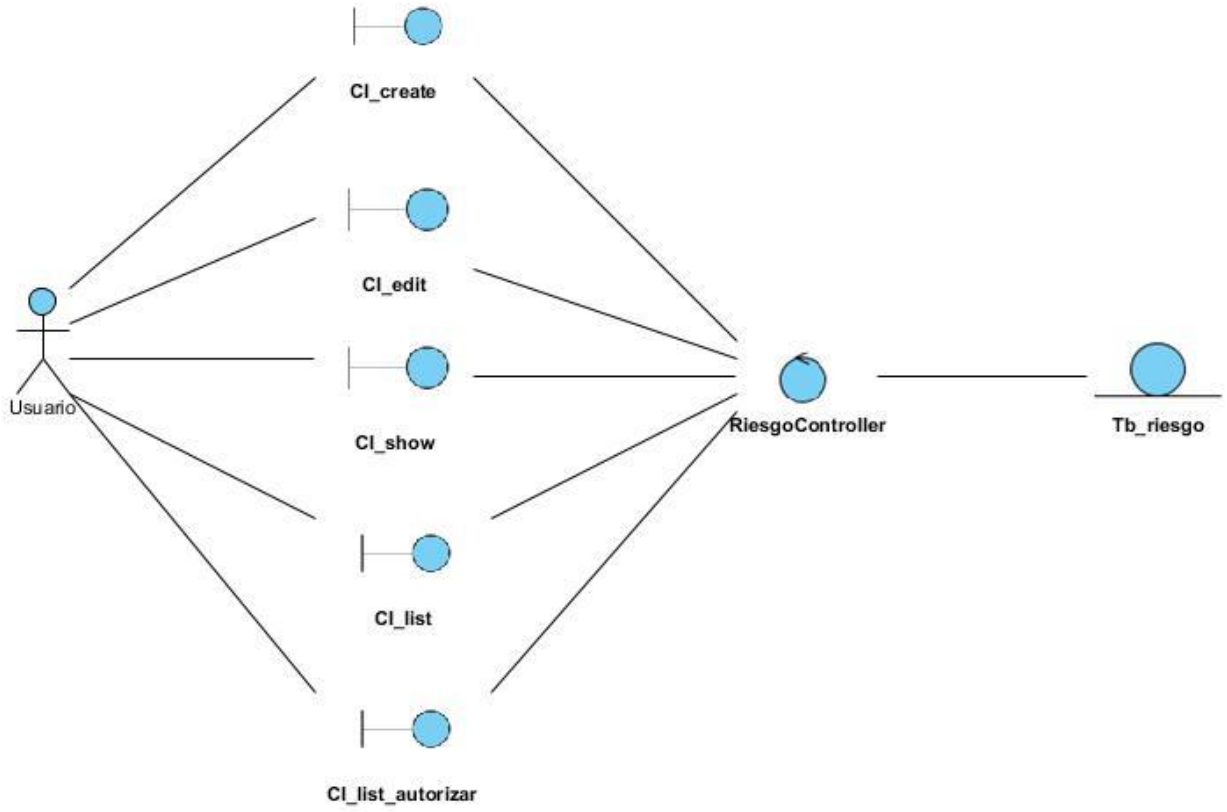


Figura 8: Diagrama de Clase del Análisis CU Gestionar Riesgo.

3.3 Modelo del Diseño

El propósito fundamental del diseño es modelar el sistema y encontrar la forma para que todos los requisitos definidos sean soportados. Se planean todos los aspectos relacionados con las restricciones y características del sistema como por ejemplo el lenguaje de programación a utilizar, el sistema operativo donde se ejecutará la aplicación y las tecnologías de interfaz de usuario, entre otras. Durante el diseño tienen en cuenta los requisitos funcionales y no funcionales previamente identificados para obtener una visión detallada de la implementación futura logrando de esta forma que el sistema sea realizado sin errores.

3.3.1Arquitectura del sistema

Los patrones son soluciones a problemas que ocurren en el entorno. Luego de llegar a la solución se encapsulan todas las variables y conforma una guía para resolver una y otra vez el mismo problema. Entre sus características se encuentran: describir el problema de forma sencilla, describir el contexto en el que ocurre, puntualizar los pasos a seguir, hacer énfasis en los puntos fuertes y débiles de la solución, referir otros patrones asociados, entre otras. En la presente investigación se abordan los patrones arquitectónicos y los patrones de diseño que son utilizados para conformar el diseño de la aplicación propuesta.

Los patrones son utilizados generalmente como plantillas que describen la estructura y organización de los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso.

Los aspectos fundamentales para el diseño con patrones son:

- **Control de acceso**: En numerosas situaciones el acceso a datos, características y funcionalidad son limitadas a la definición de los usuarios. Desde un punto de vista arquitectónico, acceder a determinadas partes del software debe tener un riguroso control.
- **Concurrencia**: Muchas aplicaciones deben manejar múltiples tareas de forma que simule el paralelismo. Hay diferentes formas de manejar dicha concurrencia, y cada una puede ser presentada por un patrón arquitectónico diferente.
- **Distribución**: El problema de distribución dirige el problema de forma en que los sistemas o componentes se comunican con otros en un entorno distribuido.
- **Persistencia**: Los datos persistentes son almacenados en bases de datos o archivos y pueden ser leídos o modificados por otros procesos más adelante. En los entornos orientados a objetos esto va más allá, y lo que puede ser accedido o modificable son las propiedades de los objetos.

3.3.2Patrones arquitectónicos

Para el desarrollo de aplicaciones web, existen numerosos patrones arquitectónicos los cuales son ampliamente utilizados. En el caso particular de la solución propuesta y teniendo en cuenta las características del mismo, se analizan dos estilos arquitectónicos fundamentales descritos a continuación:

Patrón arquitectónico Cliente-Servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este

modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

Las principales características de la arquitectura cliente-servidor son:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras y módems.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de servicio, que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente-Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

La siguiente imagen muestra el flujo Cliente-Servidor:

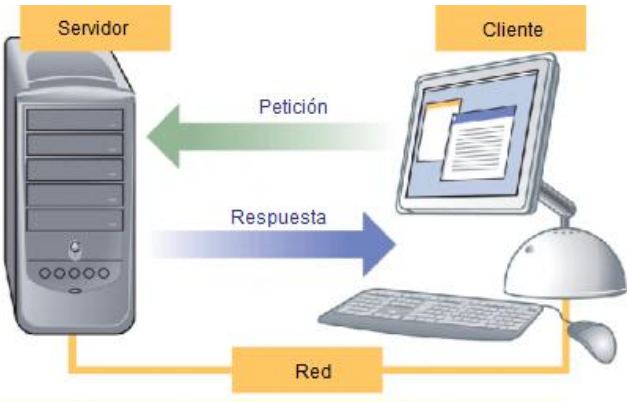


Figura 9: Representación gráfica del flujo Cliente-Servidor.

Patrón arquitectónico Modelo Vista Controlador

El patrón de arquitectura de las aplicaciones de software Modelo Vista Controlador (MVC), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos(MVC 2012)

Modelo: Es la representación de la información que maneja la aplicación, los datos puros, que puestos en contextos del sistema, proveen de información al usuario o a la aplicación en sí. A continuación se muestra la estructura de la clase Tb_riesgo como ejemplo del modelo existente en el sistema.

```
package riesgo

class Tb_riesgo {
    Tb_proyecto proyecto
    String riesgo
    String titulo
    static constraints = {
        titulo(nullable: false, blank: false)
        riesgo(nullable: false, blank: false)
        proyecto(nullable: false, blank: false)
    }
    static  mapping = {
        riesgo  type: 'text'
    }
}
```

Figura 10: Parte del código de la Clase Tb_riesgo.

- **La Vista**: Es la representación del modelo en forma gráfica, disponible para la interacción del usuario. En el caso de una aplicación Web, la vista es una página con

contenido dinámico sobre el cual, el usuario puede realizar operaciones. Se muestra a continuación el paquete que conforma la vista correspondiente a la clase Tb_riesgo en el sistema.

```
<tbody>
<tr>
    <td class="colSepGran" nowrap="nowrap" style="padding-top: 8px;"> Proyecto:
    <td class="colSepGran" nowrap="nowrap" style="border: 0; padding-top:8px;"
    <td  class="colSepGran" style="padding-top: 8px; text-align: left;"> </td>

</tr>
```

Figura 11: Vista de la Clase Tb_riesgo.

- **El Controlador**: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria modificando el Modelo en caso de ser necesario. A continuación se muestra la estructura de la clase RiesgoController en el sistema.

```
package riesgo

import org.springframework.dao.DataIntegrityViolationException

class RiesgoController {
    static allowedMethods = [save: "POST", update: "POST", delete: "POST"]



    def index() {

        redirect(action: "list", params: params)
    }

    def list(Integer max) {

        def riesgo= new ArrayList<Tb_riesgo>()
```

Figura 12: Parte del código de la Clase RiesgoController.

Ventajas del MVC:

- Clara separación entre interfaz, lógica de negocio y de presentación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.

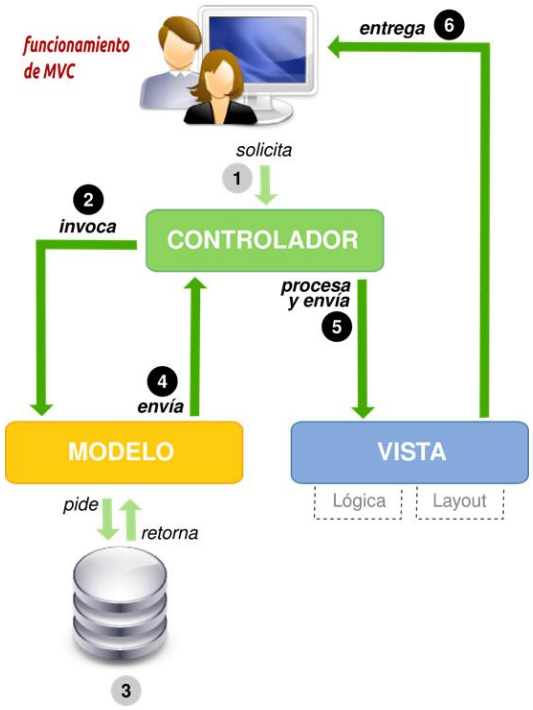


Figura 13: Ciclo de vida del Patrón MVC (Tomado de (MONOGRAFIAS.COM)).

La siguiente figura muestra el funcionamiento de un sistema implementado que usa el patrón arquitectónico Modelo Vista Controlador (MVC).

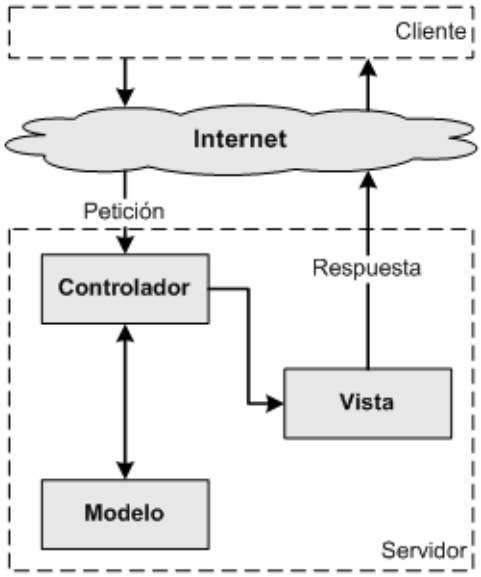


Figura 14: Funcionamiento de un sistema utilizando el patrón MVC (Tomado de (LIBROSWEB)).

A pesar de que ambos patrones cumplen con las necesidades del sistema a desarrollar, se decide emplear el patrón MVC debido a que el lenguaje de programación definido previamente, así como el framework y el IDE de desarrollo implementan dicho patrón.

3.3.3 Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto (ERICK GAMMA). Entre los más conocidos se encuentran los patrones de Asignación de Responsabilidades y los patrones de la Banda de los Cuatro.

Patrones GRASP

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto (ERICK GAMMA) Dentro de los más conocidos se encuentran los patrones de Asignación de Responsabilidades (GRASP) y los patrones de la Banda de los Cuatro (GOF).

Los Patrones Generales de Asignación de Responsabilidades de Software (*GRASP*, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos. Los mismos no introducen ideas novedosas, sino que son la mera codificación de los principios básicos más usados.

Patrones GOF

Dentro de los patrones de la Banda de los Cuatro [*Gang of Four, (GoF* por sus siglas en inglés)], utilizados en el diseño de la presente investigación para el desarrollo de subsistemas más pequeños se encuentra en patrón Solitario o Singleton y el patrón Fachada. Los patrones *GoF*, se encuentran agrupados en 3 divisiones en dependencia de su responsabilidad, estas son: comportamiento, creacionales y estructurales.

En el desarrollo de la presente aplicación se evidencia el empleo de los siguientes patrones de diseño:

Experto:

Este patrón constituye el principio básico de asignación de responsabilidades en diseño orientado a objetos. Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. En la aplicación se pone de manifiesto a través de la siguiente imagen, que muestra como la clase RiesgoController delega la responsabilidad de exportar a la clase PdfRiesgoService, ya que esta reúne la información necesaria para realizar dicha funcionalidad, lo que la hace ser, la clase experta en la información:

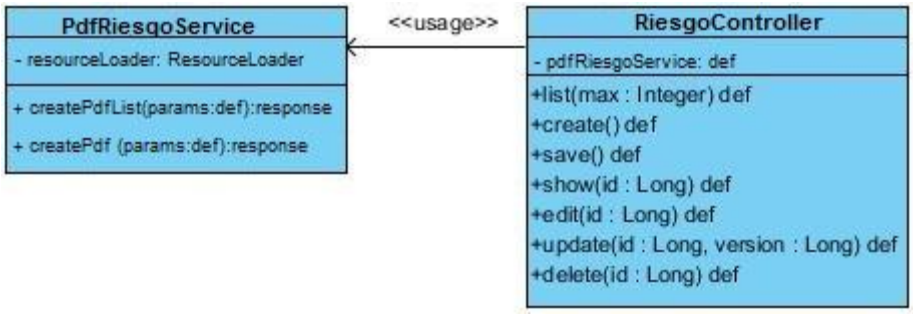


Figura 15: Patrón Experto en la solución.

Creador:

Guía la asignación de responsabilidades relacionadas con la creación de objetos. En el sistema se evidencia mediante la clase RiesgoController, ya que tiene la responsabilidad de crear una instancia de la clase Tb_riesgo, o sea que RiesgoController es un creador de los objetos de Tb_riesgo, como se muestra a través de la siguiente imagen:

Figura 16: Aplicación del patrón Creador.

Controlador:

El patrón controlador se evidencia en el sistema mediante las clases controladoras, que son generadas a su vez por el patrón arquitectónico usado, más específicamente se muestra a través de la siguiente imagen con la clase RiesgoController, que es la intermediaria entre la vista Riesgo y el algoritmo que la implementa, ya que recibe los datos del usuario y 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.

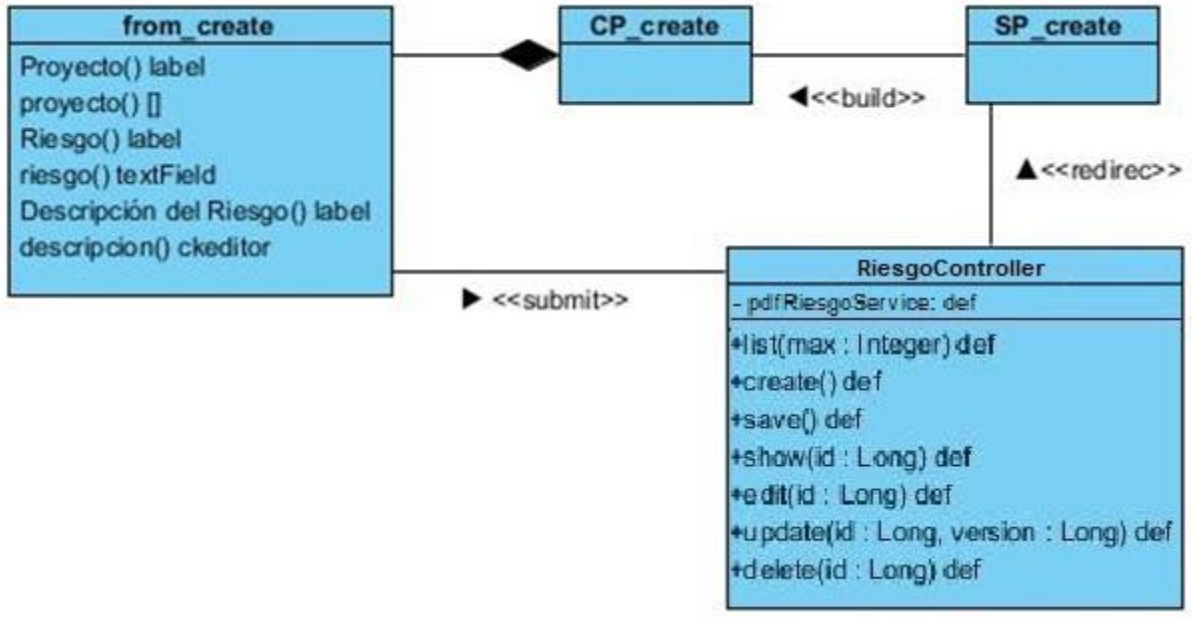


Figura 17: Patrón Controlador.

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. Se evidencia en la clase Tb_Plan_Mitigacion, puesto que está relacionada con una serie de clases de la cuales requiere para crear una instancia de ella como por ejemplo, la clase Tb_Proyecto.

Bajo acoplamiento:

Es la idea de tener las clases lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Los beneficios de este patrón es que no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

3.4 Diagramas de Clases del Diseño (DCD)

Para la realización de los diagramas de clases del diseño se presentan como elementos significativos a tres clases UML fundamentales: server page, client page y form, empleadas para el código servidor, código cliente y formularios respectivamente. Esto permite representar ficheros contenedores de sentencias script.

Descripción de los elementos del Diseño

En la elaboración de los diagramas de clases del diseño se utilizarán los elementos que a continuación se muestran en la siguiente tabla:

Tabla 9: Estereotipos utilizados en el Diagrama de Clases del Diseño.

Clases	Descripción	Representación
Server Page	Representa una página Web que tiene scripts ejecutados por el servidor. Estos scripts interactúan con los recursos que se encuentran al alcance del servidor. Solo puede mantener relaciones con objetos que se encuentren en el servidor.	
Client Page	Representan páginas cliente que son dibujadas por el navegador Web y pueden ser una combinación de algún o algunos lenguajes, scripts del lado del cliente.	
Form	Representa una colección de campos de entrada que forman parte con una página del lado cliente	

Las relaciones existentes entre las clases que integran los diagramas se muestran en la siguiente tabla:

Tabla 10: Relaciones de clases del diseño con estereotipos Web.

Clases/Relaciones	Server Page	Client Page	Form
Server Page	Redirect	Build/Redirect	-
Client Page	Link	Link/Redirect	Agregación
Form	Submit	-	-

Una vez establecidos los estereotipos a utilizar en los diagramas, así como las relaciones entre las clases que interactúan en los diagramas se procede a realizar la representación gráfica de los DCD para los CU Autenticar Usuario y CU Gestionar Riesgos. El resto de los DCD se encuentran en los anexos del presente documento

CU Autenticar Usuario

La Figura 18 muestra las clases utilizadas para la autenticación de los usuarios en el sistema. En este caso la página cliente CI_login muestra el formulario form a través del cual se introduce las credenciales del usuario, para la autenticación utiliza la clase controladora AutenticarController la cual se comunica con la clase entidad Tb_usuario.

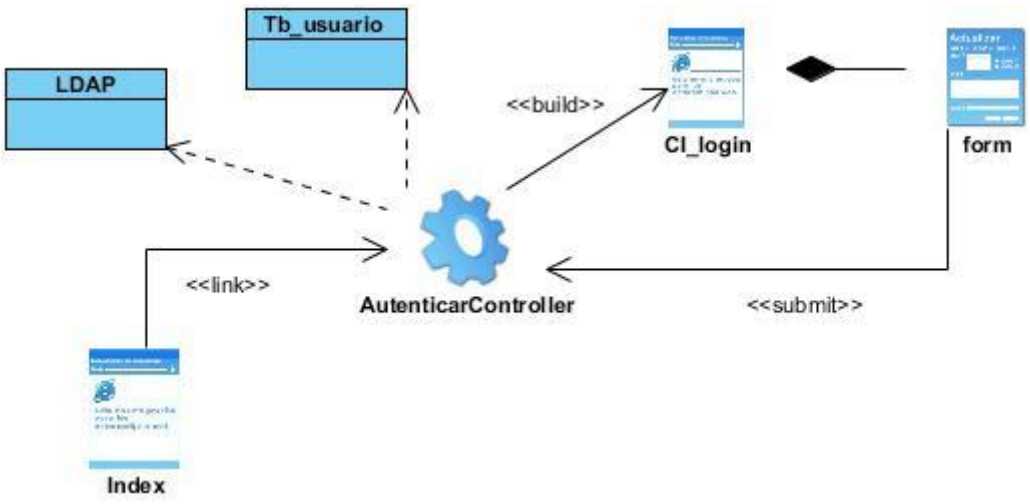


Figura 18: DCD CU: Autenticar Usuario.

CU Gestionar Riesgos

La figura 18 muestra las clases utilizadas para la gestión de los riesgos de un proyecto en el sistema. En este caso la página cliente Create muestra el formulario Form_Create a través del cual se introduce los datos del nuevo riesgo detectado, de igual manera para editar un riesgo ya conocido la página Edit muestra un formulario para editar los datos del riesgo conocido y la página Listar devuelve un listado con todos los riesgos conocidos. La clase controladora utilizada para la gestión de los riesgos es RiesgoController la cual se comunica con la clase entidad Tb_riesgo.

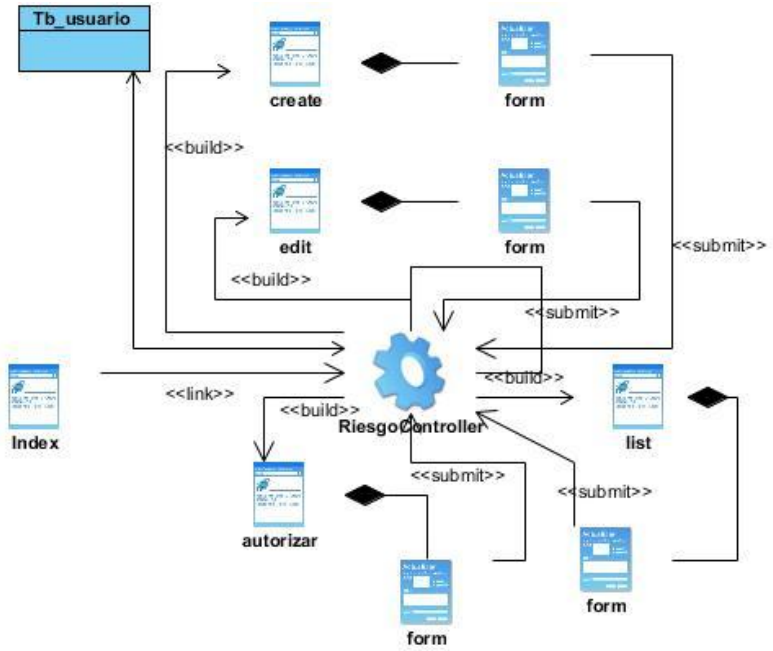


Figura 19: DCD CU: Gestionar Riesgo.

3.5 Diagramas de interacción

Los Diagramas de interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado. Existen dos tipos de diagramas de interacción: diagramas de colaboración y diagramas de secuencias. Durante el análisis de la aplicación será utilizado este último dado que es un tipo de diagrama usado para modelar interacción entre objetos. Un diagrama de secuencia (DS) muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso.

A continuación se muestran los diagramas de secuencia (DS) de los casos de uso Autenticar Usuario y Gestionar Usuario.

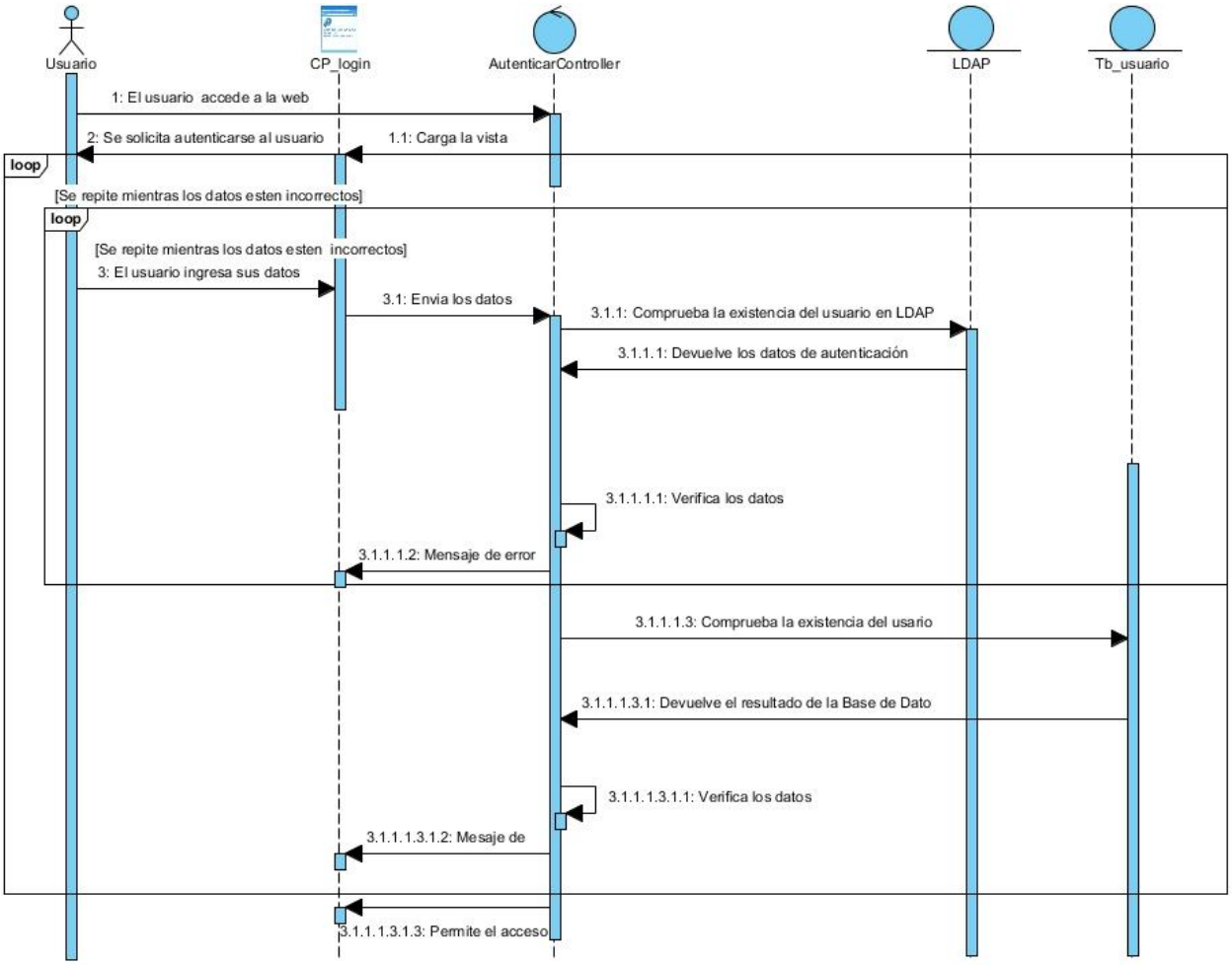


Figura 20: DS: Autenticar Usuario.

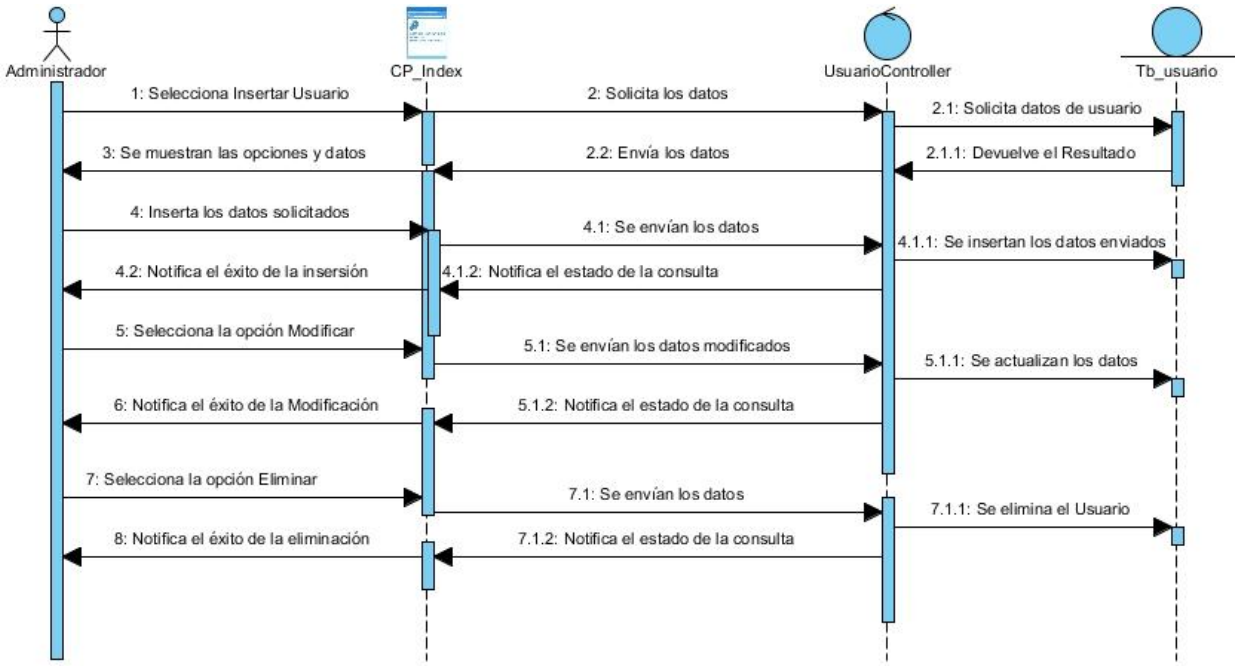


Figura 21: DS: Gestionar Usuario.

3.6 Diseño de la Base de Datos

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos correctamente diseñada permite obtener acceso a la información exacta y actualizada.

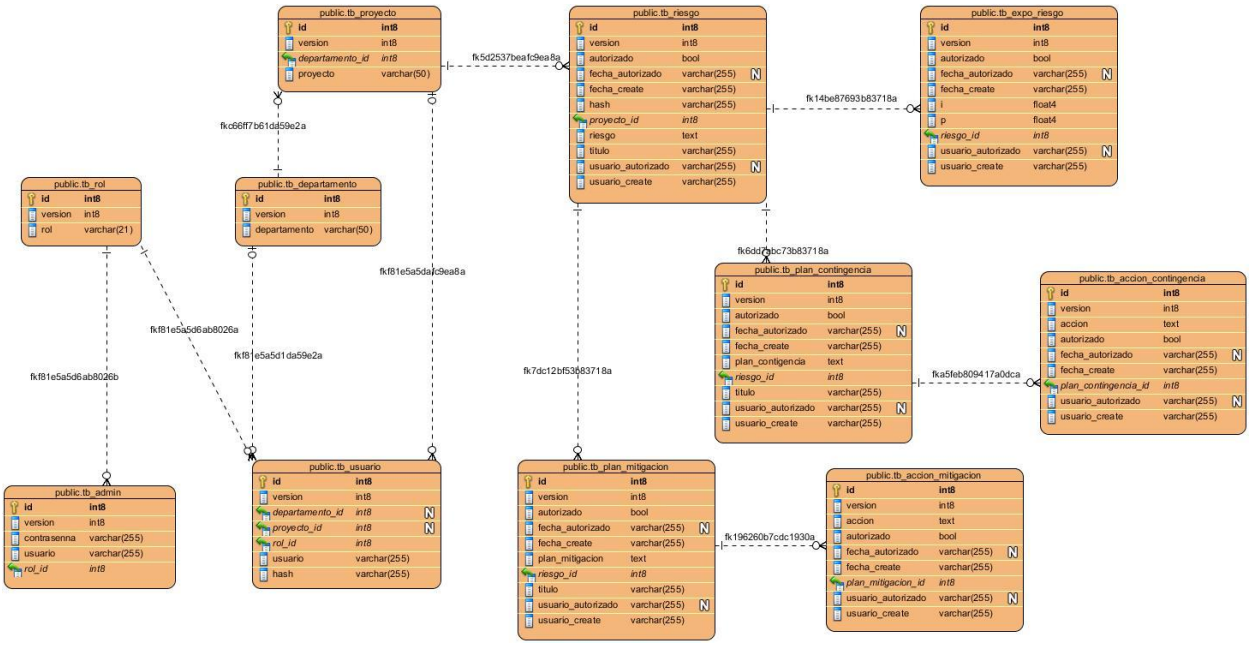


Figura 22: Modelo de Datos

3.7 Diagrama de despliegue

En el diagrama de despliegue se muestra la configuración en funcionamiento del sistema incluyendo su software y su hardware. Para cada uno de los componentes del diagrama es necesario documentar las características técnicas requeridas, el tráfico de la red, el tiempo de respuesta. Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. En los diagramas de despliegue se representan a los nodos y sus relaciones.

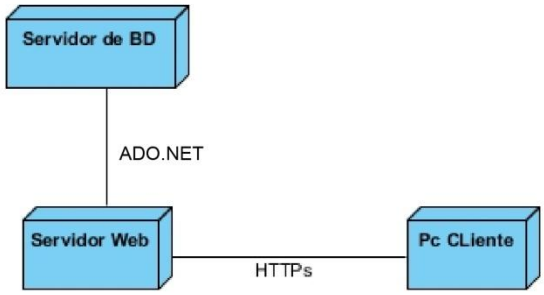


Figura 23: Diagrama de Despliegue.

Descripción de los nodos físicos del sistema

- ✓ **PC Cliente**: El nodo representa una PC cliente desde la cual se podrá acceder al sistema por medio de un navegador Web e interactuar con todas las funcionalidades que este brinda.
- ✓ **Servidor Web:** El nodo representa el servidor Web donde estará alojado el sistema así como los componentes almacenados en el mismo.
- ✓ **Servidor de Base de Datos:** El nodo representa el servidor de Base de Datos PostgreSQL en el que estará alojada la base de datos del sistema.

Descripción de los protocolos utilizados:

- ✓ **HTTPS:** Protocolo para Transferencia de Hipertexto seguro. Extensión del HTTP para la autenticación y encriptación de datos entre un servidor Web y un navegador Web.
- ✓ **ADO.NET:** Es una parte de la biblioteca de clases base que están incluidas en el Microsoft .NET Framework. Es comúnmente usado por los programadores para acceder y para modificar los datos almacenados en un Sistema Gestor de Bases de Datos Relacionales, aunque también puede ser usado para acceder a datos en fuentes no relacionales. ADO.NET es a veces considerado como una evolución de la tecnología ActiveX Data *Objects* (ADO), pero fue cambiado tan extensivamente que puede ser concebido como un producto enteramente nuevo.

3.8 Conclusiones Parciales

- Después de realizar el análisis y diseño de la propuesta de solución mediante la metodología OpenUp generaron un grupo de artefactos necesarios para el desarrollo del mismo, se describieron los conceptos de los patrones de diseño que fueron utilizados en este flujo, asociándolos a un conjunto de ejemplos relacionados con el trabajo en cuestión.
- Entre los artefactos generados en este flujo de trabajo se encuentran: los diagramas de clases de diseño e interacción de los principales casos de uso, así como el diseño de la BD y el diagrama de despliegue lo cual sienta las bases para la implementación del sistema propuesto.
- Apoyándose en las descripciones de los casos de usos generados en el capítulo anterior se hizo posible la realización de los diagramas de secuencia de cada uno de estos, modelando así la interacción entre los objetos de la aplicación.

Capítulo 4: Implementación y Análisis de los resultados

4.1 Introducción

El presente capítulo tiene como objetivo especificar el modelo de implementación para con esto precisar la estructura y organización del sistema. Se implementan las clases y subsistemas hallados durante el diseño representado en los diagramas de componentes. Además se realiza un análisis de los resultados obtenidos durante el desarrollo del sistema. Así como una ejemplificación de las principales interfaces del sistema con las que interactúa el usuario.

4.2 Modelo de implementación

El modelo de implementación es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros de código fuente, y otros tipos de ficheros necesarios para la implantación y despliegue del sistema. Describe también, como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación en el lenguaje o lenguajes de programación utilizados, y como dependen de los componentes unos de otros (CARABALLO 2003).

4.2.1 Diagrama de componentes

Un diagrama de componentes muestra las dependencias entre los componentes de software. Incluye los clasificadores que los especifican (por ejemplo, clases de implementación) y los artefactos que los implementan, tales como, archivos de código fuente, archivos de código binario, archivos ejecutables, scripts (PATRICIO 2010).

Los diagramas de componentes describen los elementos físicos del sistema ya sean ejecutables, librerías, clases y las relaciones entre ellos. Para la modelación del diagrama que a continuación se propone primeramente se identificaron los componentes, luego se agruparon los mismos por la arquitectura, en este caso Modelo Vista Controlador (MVC) y se representaron las relaciones entre ellos.

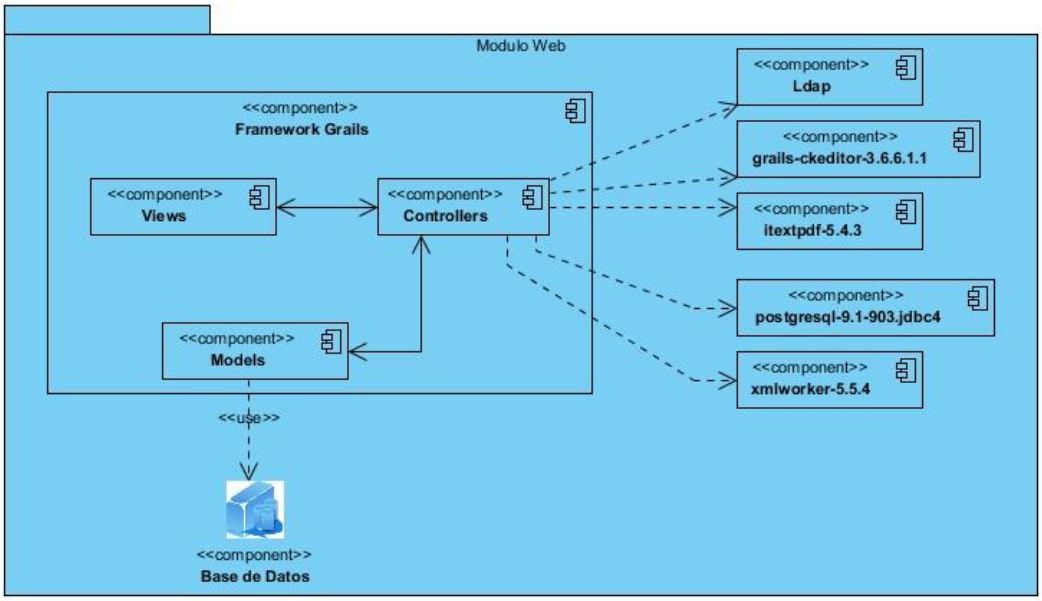


Figura 24: Diagrama de Componentes del sistema.

Descripción de los componentes representados en el diagrama anterior:

Componente	***Propósito***
Grailss-ckeditor-3.6.6.1.1	Librería externa de Grails que se utiliza para la edición de texto.
Itexpdf-5.4.3	Librería externa de Grails que se utiliza para exportar documentos a formato PDF.
Postgresql-9.1-903.jdbc4	Librería externa de Grails que se utiliza para la conexión a la base de datos.
Xmlworker-5.5.4	Librería externa de Grails que se utiliza para la conversión de código html a texto plano.
LDAP	Clase Java que permite la autenticación con el LDAP de la UCI.
Views	Contiene las páginas Web que son vistas por el usuario.
Models	Son clases de Groovy que se diseñan para trabajar con información de la Base de Datos.
Controllers	Contiene las clases que determinan cómo se manejan las solicitudes HTTP, y permiten la lógica del negocio.
Base de Datos	Representa la Base de Datos.

En el Anexo 1 del presente documento se encuentran los diagramas con los componentes generados.

4.3 Interfaces del sistema

Las interfaces de la aplicación mostradas son capturas de pantalla tomadas durante el funcionamiento del mismo. Estas imágenes muestran parte de los resultados obtenidos con el desarrollo de la presente investigación (Ver Anexo 2).

4.4 Pruebas

Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

En todo proceso de desarrollo de aplicaciones es indispensable la presencia de un proceso de Pruebas de Software para garantizar así el buen funcionamiento y la calidad del producto final. Estas tienen como objetivo general verificar y validar un software, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo.

4.4.1 Pruebas de seguridad

Las pruebas de seguridad tienen como objetivo hacer un análisis con el fin de encontrar vulnerabilidades en el sistema, o sea fallos de seguridad tanto en el diseño como en la implementación de la aplicación. Además buscan medir la confidencialidad, integridad y disponibilidad de los datos. Las pruebas de seguridad del sistema se realizaron con la herramienta Acunetix en su versión 8.0. A continuación se muestran los resultados de las 3 iteraciones de pruebas realizadas, las imágenes mostradas fueron tomadas de la propia herramienta utilizada.

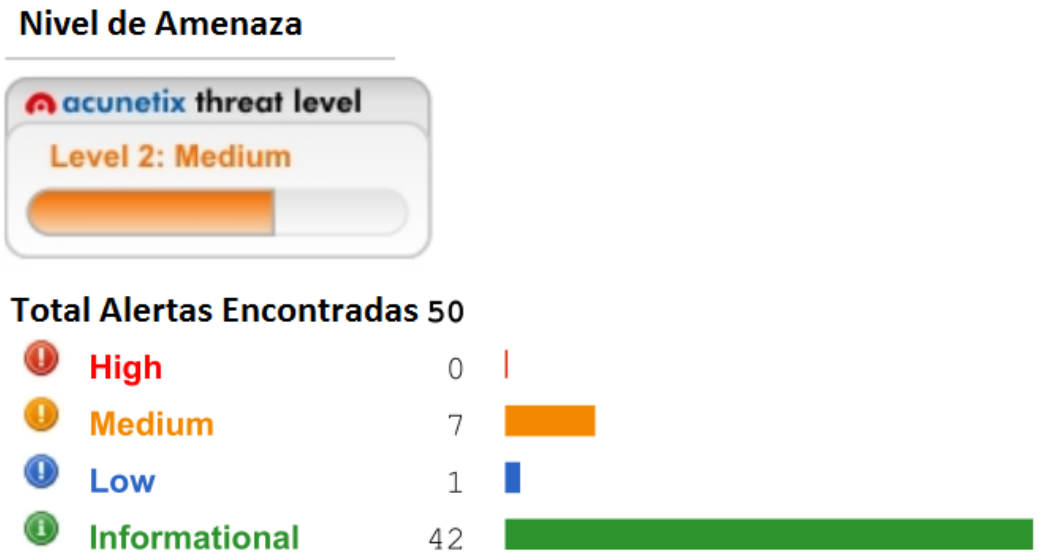


Figura 25: Primera iteración de pruebas.

Después de realizar la primera iteración se encuentran un total de 50 alertas (Ver Figura 30). Luego de corregidas, se realiza la segunda iteración cuyos resultados se muestran en la Figura 31.

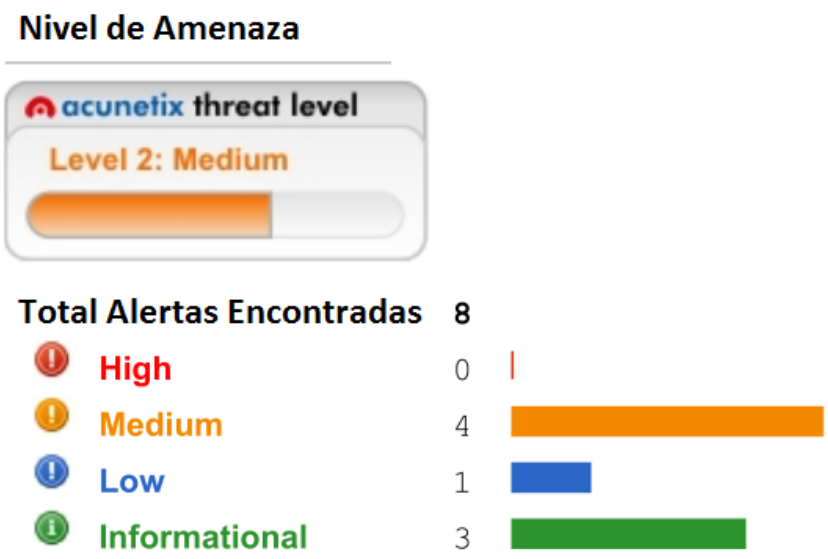


Figura 26: Segunda iteración de pruebas.

Luego de realizada la segunda iteración la herramienta arroja 8 no conformidades (Ver Figura 31), las cuales son corregidas para posteriormente realizar una tercera iteración la cual arroja los resultados mostrados en la Figura 32.

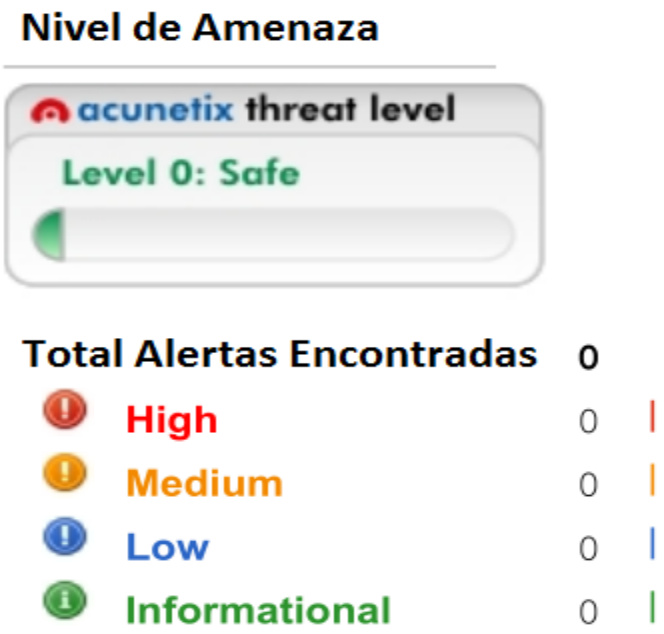


Figura 27: Tercera iteración de pruebas.

4.4.2 Pruebas de Caja Negra

Son las que realizan pruebas de forma que se compruebe que cada función del sistema es operativa. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la

ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione bien.

Se intenta encontrar con estas pruebas:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Se decide realizar este método de prueba ya que se llevan a cabo sobre la interfaz del software. O sea, pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene (Ver Anexo 3).

4.5 Conclusiones parciales

- Una vez concluida la fase de implementación, se obtuvieron los diagramas de componentes.
- Una vez codificado el sistema, se presentaron las principales interfaces de la herramienta desarrollada.
- Se constituyeron escenarios de prueba a la interfaz de usuario con el objetivo de corroborar el correcto funcionamiento del sistema.
- Después de la implementación de los escenarios de pruebas, se pudo obtener un sistema que funciona correctamente.

Conclusiones generales

Se culmina la presente investigación logrando cumplir los objetivos trazados con técnicas y metodologías para cada uno de ellos. En vista de lograr la mejor aproximación a estos, se puede arribar a los siguientes resultados:

1. Se investigó la existencia de soluciones como punto de partida para definir la adopción de alguna de ellas o la implementación de una propia.
2. Para el cumplimiento de los objetivos y en concordancia con las necesidades expuestas por el cliente, se requirió hacer un estudio de la situación actual del proceso de gestión y seguimiento de riesgos en el centro DATEC de la Universidad de las Ciencias Informáticas, para con ello elevar la eficiencia del proceso en cuestión.
3. El uso de patrones de diseño y arquitectónicos brindó como resultado un diseño sólido y flexible para la codificación de la aplicación.
4. La correcta selección de la metodología de desarrollo, el lenguaje de modelado y la herramienta CASE a utilizar propició la correcta representación del sistema a través de los diferentes modelos.
5. Se insistió en la utilización de software y herramientas libres atendiendo a una sana política de independencia y libertad de acción futura previendo, una vez implementado el sistema, la incorporación paulatina de mejoras en posteriores versiones del mismo.

Resultado de la presente investigación se obtuvo una herramienta informática que contribuye a la gestión, el diagnóstico y seguimiento de riesgos en el centro DATEC de la UCI.

Recomendaciones

Como recomendaciones al finalizar el proceso de desarrollo del presente sistema se tienen:

- Extender su uso a otras áreas de la Universidad y del país con el fin de contribuir a mejorar el proceso de diagnóstico y seguimiento de riesgos no solo a nivel UCI sino también en las entidades estatales cubanas.
- Continuar la investigación relacionada con el diagnóstico y seguimiento de riesgos con el fin de agregarle nuevas funcionalidades y reportes a la herramienta según las necesidades que surjan.
- Se recomienda realizar el manual de usuario para la solución presentada.

Bibliografía

ABDUL-JAWAD, B. *Groovy and Grails Recipes*. New York, NY, Apress, Inc, 2009. p.

ALCOLEA, R. P. *Pruebas de unidad con Spock Testing Framework*, 2012. [5 de Marzo de 2015]. Disponible en: http://grails2.spirobyte.com/blog/index.php?entryid=2

ASENSIO, R. M.-B. *Gestión de Riesgos en ingeniería del software*, Universidad de Murcia España, 2005. [Disponible en: http://www.um.es/docencia/barzana

BOOCH, J. *El proceso unificado de desarrollo de software*. Addison Wesley, 2005. p. ISBN: 84-7829-036-2

CAMPOVERDE VÉLEZ, F. *Administración de los Riesgos Empresariales*, 2011.

CARABALLO, S. *Prolegómenos Sobre el Lenguaje de Modelado Unificado (UML)*. ,[Online]. [Consultado: Noviembre 11 de 2014] Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/prolego.pdf 2003.

CARRILLO P., I. P. G., RODRIGO; RODRÍGUEZ M., AURELIANO D. *Metodología de Desarrollo del Software*, Disponible en: http://tecnicaytecnologiasc.wikispaces.com/file/view/Metodologias%20de%20desarrollo.pdf/409623082/Metodologias%20de%20desarrollo.pdf

2014.

CAVSI. *Computer Audio Video Systems Integrator*, 2014. [Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/

CHARETTE, R. N. *Software Engineering Risk Analuysis and Management*. New York, EEUU: McGraw-Hill, 1989. p.

CHRISTOPHER M. JUDD, J. F. N., JAMES SHINGLER. *Beginning Groovy and Grails. From Novice to Professional*. New York, NY, Apress, Inc 2008. p.

CONTRALORÍA. *Ley de la Contraloría General de la República*. Habana, 2009. Ley_No.107/09.

CONTRALORÍA. *Normas del Sistema de Control Interno*. Habana, 2011. Resolución No. 60/11.

COSO. *Gestión de Riesgos Corporativos. Técnicas de Aplicación*. 2004. p.

DANISOFT. *Las novedades de ASP.Net*, Disponible en: http://www.danysoft.com/free/aspnet.pdf, 2014.

DICKINSON, J. *Grails 1.1 Web Application Development. Birmingham*. Packt Publishing, 2009. p.

DRIVE, G. JVM Web Framework Matrix, 2010.

DURÁN A., M. V. Y. A. H., M. *Metodología para el proceso Identificación de Riesgos V Encuentro Internacional de Contabilidad, Auditoría y Finanzas.* La Habana, Consultoría BISE S.A, 2007. p.

EPF. *OpenUp Basic. Eclipse Foundation*, 2011. [Disponible en: http://epf.eclipse.org/wikis/openupsp/

ERICK GAMMA, R. H., RALPH JONHSON, JONH VLISSIDES. *Patrones de Diseño: elementos de software orientados a objetos reutilizables*. Addison Wesley, p. *Traducido de: Design Patterns: Elements of Reusable Object-Oriented Software.*

EUI-FI. *Introducción a Herramientas CASE y System Architect*, 2014.

FREEDOWNLOADMANAGER. *Visual Paradigm for UML*, 2014. [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

GARCÍA, J. *Lenguaje de Modelado*, 2014. [Disponible en: http://www.ingenierosoftware.com/analisisydiseno/uml.php

GONZALEZ CEDEÑO, Y. *Gestión de Riesgos del proyecto Sistema de Gestión Penitenciaria*. La Habana, 2008. p.

GONZÁLEZ, L. *Cuadernos de Trabajo de Hegoa*. LANKOPI S.A, Universidad del País Vasco, 2000.

HECHAVARRIA GUIBERT, L. Y. M. Z., ARIADNA BARBARA. *Gestión de riesgos en el proyecto Sistema de Gestión Fiscal*. La Habana, 2008. p.

HERNÁNDEZ SÁNCHEZ, J. A. *Ambiente de aprendizaje interactivo en Internet, basado en la tecnología JSP para la Educación Ambiental*, 2014. [Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/hernandez_s_ja/capitulo4.pdf

INTELLIJIDEA. *The Most Intelligent Java IDE*, 2014. [Disponible en: http://www.jetbrains.com/idea/

ISO, I. S. O. *Gestión de riesgos – Terminología – Líneas directrices para el uso en las normas*. ISO/CEI 73. 2004, 2004. p.

J MARCELO COCHO, R. A. M. *Estudio exploratorio sobre los métodos de gestión de proyectos de alto riesgo. Primer Congreso Soporte del Conocimiento con la Tecnología*. Valencia España, 2003. p.

LANG, J. P. *Traducido de: Redmine Main features*, [online]. 2014. [Disponible en: http://www.redmine.org/projects/redmine/wiki/Features

LIBROSWEB. Disponible en: http://librosweb.es/libro/symfony_1_4/capitulo_2/el_patron_mvc.html

LOCKE, J. *Meet Apache wicket*, 2014. [Disponible en: https://wicket.apache.org/meet/introduction.html

MANIASI, S. D. *Identificación de riesgos de proyectos de software en base a taxonomías*. 2005. p.

MAP. *El proyecto Eurométodo. Ejercicio de Validación de EM v0*. Bilbao España, Comisión Europea, 1996. p.

MARTÍNEZ CARRERA, R. *Situación actual y perspectivas de la Administración de Riesgos en Cuba*. *Intervención en el 1er Seminario Nacional sobre Administración de Riesgos*, La Habana, Cuba, 1998.

MAVEN. *Groovy Tapestry*, 2004. [Disponible en: http://groovestry.sourceforge.net/

MILANÉS LÓPEZ, A. Y. M. G., REINIER. *Sistema para la Gestión de la Tecnología en la Universidad de las Ciencias Informáticas*. La Habana, 2009. p.

MONOGRAFIAS.COM. Disponible en: https://www.google.com.cu/search?q=patron+mvc&biw=1280&bih=675&tbm=isch&tbo=u&source=univ&sa=X&ei=IpX-VMbXD7SBsQTMiYHQCw&ved=0CCYQsAQ#imgdii=_&imgrc=BMIo2Es9ru3IyM%253A%3BANmbt95Bd0czaM%3Bhttp%253A%252F%252Fwww.monografias.com%252Ftrabajos89%252Fpoo-y-mvc-php%252Fimage008.png%3Bhttp%253A%252F%252Fwww.monografias.com%252Ftrabajos89%252Fpoo-y-mvc-php%252Fpoo-y-mvc-php2.shtml%3B548%3B734

MVC. Microsoft, 2012. [2014]. Disponible en: http://msdn.microsoft.com/es-es/library/bb972251.aspx#M19

MYSQL. *MySQL 5.0 Reference Manual*, [online]. 2014. [Disponible en: http://dev.mysql.com/doc/refman/5.0/en/

NETBEANS. *Bienvenido a NetBeans*, 2014. [Disponible en: https://netbeans.org/index_es.html

OPENSUSE. *Apache*, 2014. [Disponible en: http://es.opensuse.org/Apache

PATRICIO, L. *"Rational Unified Process (RUP)"*. [Consultado el: 7 de Noviembre de 2014] Disponible en: http://es.scribd.com/doc/51852648/Introduccion-a-RUP, Universidad Politécnica de Valencia, 2010.

PELEGRIN PÉREZ, E. L. *La administración de los Riesgos, su impacto en la empresa*. Cuba, UPR, 2006. p.

PENADÉS, M. C. C., JOSÉ HILARIO. *Metodologías ágiles en el desarrollo del software*, Disponible en: http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf, 2014.

PÉREZ QUINTERO, L. Y. V. G., CARLOS MANUEL. *Estrategia para el desarrollo de requisitos no funcionales de software en proyectos productivos de la Universidad de las Ciencias Informáticas*. La Habana, 2008. p.

PHP. *Manual de PHP*, 2014. [Disponible en: http://www.php.net/manual/es/intro-whatis.php

PMI. *Guía de los Fundamentos para la Dirección de Proyectos*. Newtown Square Pensilvania, Project Management Institute, 2008. p. 1-930699-73-5

PRESEDO, C., DOLADO, J. JAVIER AND AGUIRREGOITIA, AMAI. *Estudio de métricas para el control de proyectos software*. Bilbao, España, Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, 2010. p. 1, 2010. 1988-3455

PRESSMAN, R. S. *Ingenieria de Software*. New York, EEUU: Mc Grow-hil, 1998. p. ISBN: 9786071503145

PROGRAMACIÓNWEB.NET. *¿Que se puede decir del PHP?*, 2014. [Disponible en: http://www.programacionweb.net/articulos/articulo/que-se-puede-decir-php/

QUINTERO ACOSTA, E. Y. B. B., JOSUE HUGO. *Gestión de Riesgos para el Proyecto de Gestión Académica Akademos v2.0*. La Habana, 2009. p.

RAE. *Real Academia Española*, 2014. [Disponible en: http://lema.rae.es/drae/?val=riesgo

RODRÍGUEZ CARRAZANA, Y. *Modelo de identificación de los riesgos de control interno para la actividad empresarial*, 2009.

RUMBAUGH, J. J., IVAR; BOOCH, GRADY. *El Lenguaje Unificado de Modelado*. Addison Wesley, 2000. p. ISBN: 84-7829-037-0

SCRIBID. *Manual del usuario de PostgreSQL*, 2014. [Disponible en: http://es.scribd.com/doc/5703210/Manual-del-usuario-de-PostgreSQL

SEI. *CMMI Benefits. Software Engineering Institute*, Carnegie Mellon, 2012. [Disponible en: http://www.sei.cmu.edu/cmmi/why/benefits/index.cfm

SEI. *RISK Managemente Overview*, 2004. [Disponible en: http://www.sei.cmu.edu/programs/sepm/risk/

SOFTEXPERT. *Enterprise Risk Management (ERM). Minimice Riesgos. Maximice Oportunidades.*, Disponible en: http://www.softexpert.es/catalogos/catalogo-SE-ERM.pdf, 2013.

STG. *Strategic Thougth Group. Active Risk Manager (ARM) from Strategic Thought delivers advanced*, 2010.

TECHNET, M. *Información General del Servidor Web IIS*, [online]. 2012. [Disponible en: http://technet.microsoft.com/es-es/library/hh831725.aspx

UNESCO. *UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION*, 2008. [Disponible en: http://unesdoc.unesco.org/images/0018/001874/187429s.pdf

URCELAY, S. *Análisis de riesgo*. Santiago de Chile, Facultad de Ciencias Veterinarias y Pecuarias, Universidad de Chile, 2009. p.

Anexos

Anexo 1

Diagrama de componentes

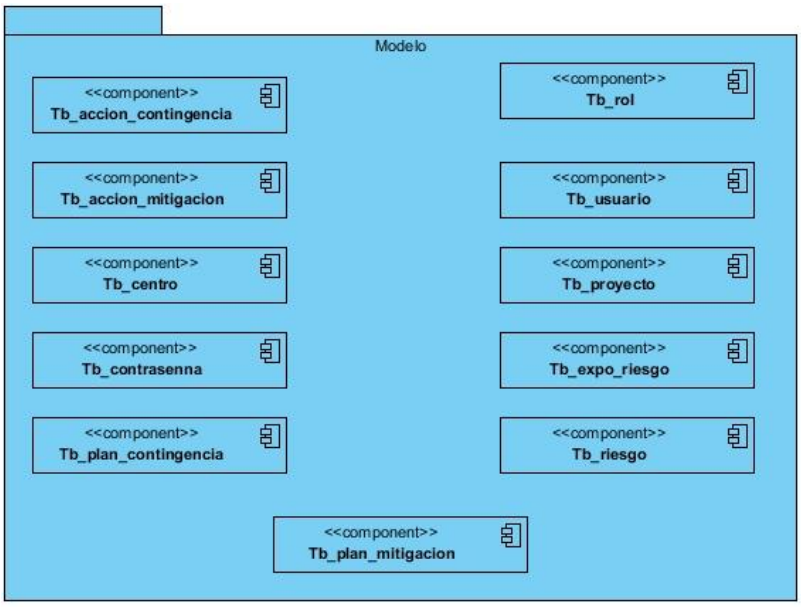


Figura 28: Componentes de la capa Modelo.

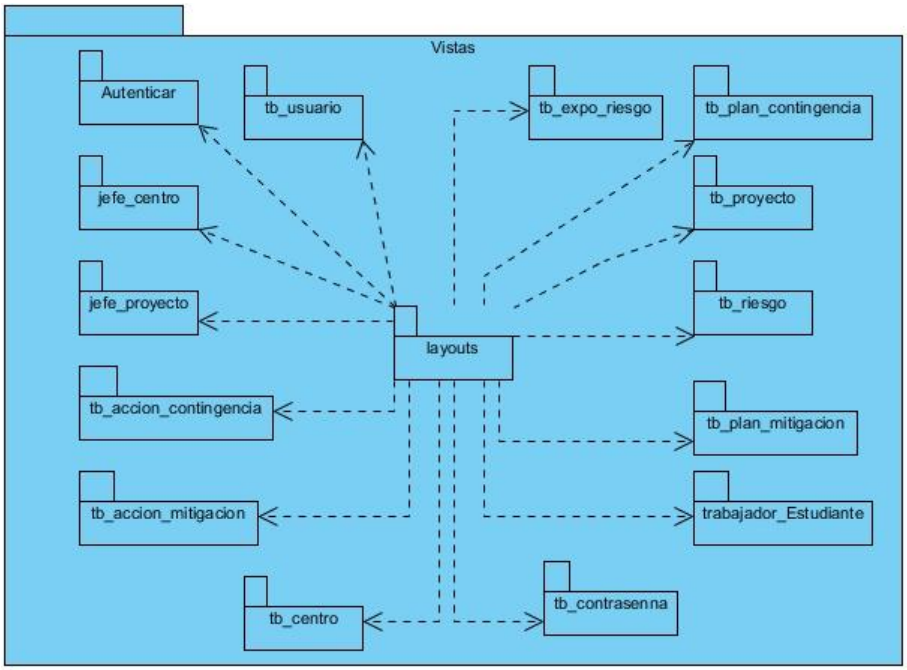


Figura 29: Componentes de la capa Vista.

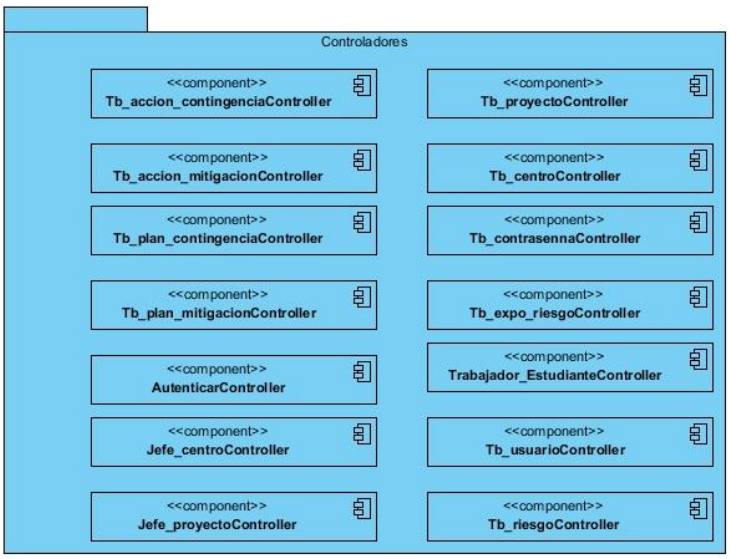


Figura 30: Componentes de la capa Controlador.

Anexo 2

Interfaces del sistema desarrollado

En la Figura 31 se muestra la interfaz de autenticación de usuario.

Figura 31: Pantalla de autenticación del sistema.

En la figura 32 se muestran los del Plan de Contingencia de un proyecto de prueba. El sistema permite editar el plan existente, eliminarlo y exportar el documento en formato pdf.

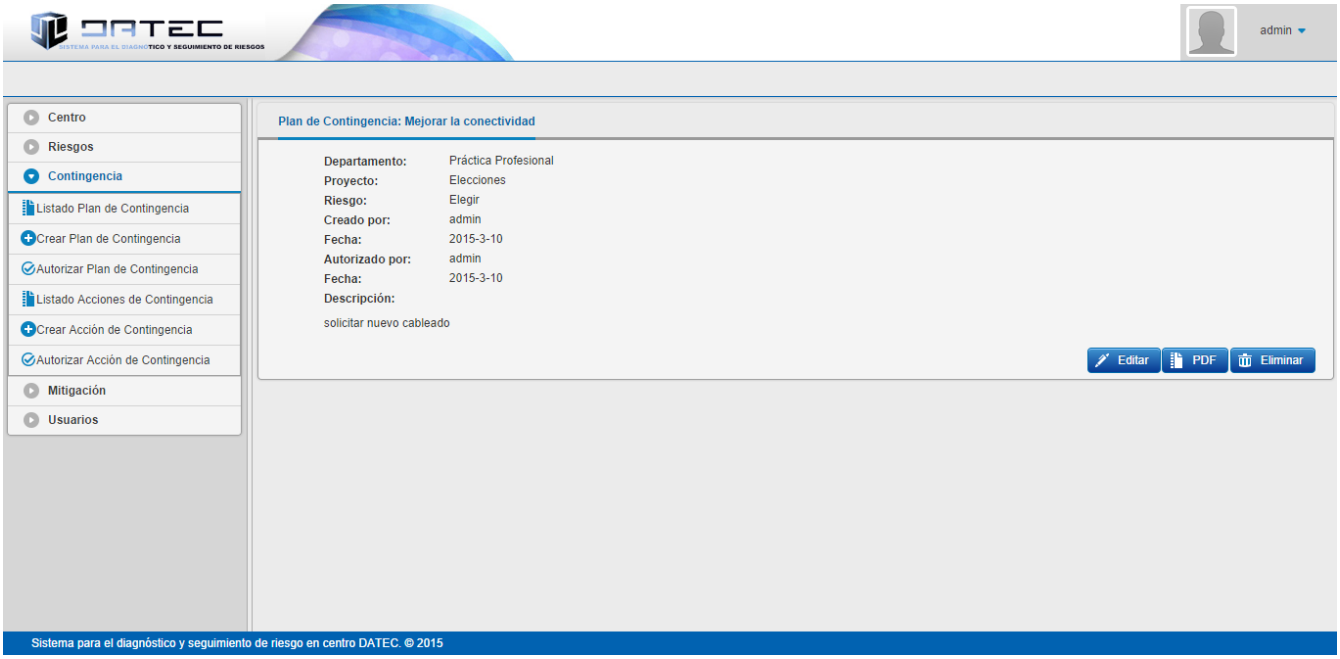


Figura 32: Descripción del Plan de Contingencia: Mejorar la conectividad.

Anexo 3

Tabla 11: Caso de Prueba Autenticar Usuario.

Nombre	Escenarios	Descripción	Flujo central
SC1: Gestionar usuario	EC 1.1: Gestionar usuario satisfactoriamente.	El sistema realiza la acción según la operación de gestión (Asignar, Modificar o Eliminar), seleccionada por el usuario.	• Seleccionar en el menú usuarios la opción “Asignar Rol”.
SC2: Asignar Rol	EC 2.1: Asignar Rol satisfactoriamente.	El sistema asigna el rol al usuario, mostrando un mensaje que confirma dicha	• Seleccionar en el menú usuarios la opción “Asignar Rol”. • Llenar los datos correspondientes.

		acción.	• Seleccionar la opción “Crear”.
	EC 2.2: Se dejan campos en blanco.	El sistema muestra el mensaje “Rellene este campo”.	• Seleccionar en el menú usuarios la opción “Asignar Rol”. • Llenar los datos correspondientes. • Seleccionar la opción “Crear”.
	EC 2.3: No son válidos los campos	El sistema muestra el mensaje: “Ajústese al formato solicitado”.	• Seleccionar en el menú usuarios la opción “Asignar Rol”. • Llenar los datos correspondientes. • Seleccionar la opción “Crear”.
	EC 2.4: Duplicado de Usuario	El sistema muestra el mensaje: “El campo Usuario debe ser único”.	• Seleccionar en el menú usuarios la opción “Asignar Rol”. • Llenar los datos correspondientes. • Seleccionar la opción “Crear”.
SC3: Modificar usuario	EC 3.1: Modificar usuario satisfactoriamente.	El sistema modifica el usuario, mostrando un mensaje que confirma dicha	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea modificar.

		acción.	• Modificar los datos correspondientes. • Seleccionar la opción “Actualizar”.
	EC 3.2: Se dejan campos en blanco.	El sistema muestra el mensaje “Rellene este campo”.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea modificar. • Modificar los datos correspondientes. • Seleccionar la opción “Actualizar”.
	EC 3.3: No son válidos los campos	El sistema muestra el mensaje: “Ajústese al formato solicitado”.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea modificar. • Modificar los datos correspondientes. • Seleccionar la opción “Actualizar”.
	EC 3.4: Duplicado de Usuario	El sistema muestra el mensaje: “El campo Usuario debe ser único”.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea modificar. • Modificar los datos

			correspondientes. • Seleccionar la opción “Actualizar”.
SC4: Eliminar usuario	EC 4.1: Eliminar usuario satisfactoriamente.	El sistema elimina el usuario, mostrando un mensaje que confirma dicha acción.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea eliminar. • Seleccionar la opción “Eliminar”. • Confirmar la eliminación del usuario (seleccionar “sí” en el mensaje de confirmación).
	EC 4.2: No se confirma la eliminación del usuario.	El sistema oculta el mensaje de confirmación y mantiene el usuario registrado en el sistema.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea eliminar. • Seleccionar la opción “Eliminar”. • Confirmar la eliminación del usuario (seleccionar “sí” en el mensaje de confirmación).
	EC 4.3: Cerrar mensaje de confirmación.	El sistema oculta el mensaje de confirmación y mantiene el usuario registrado en el sistema.	• Seleccionar en el menú usuarios la opción “Listado de usuario”. • Seleccionar el usuario que se desea eliminar.

			• Seleccionar la opción “Eliminar”. • Confirmar la eliminación del usuario (seleccionar “sí” en el mensaje de confirmación).

Tabla 12: Descripción de variables.

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Usuario	• Campo de texto (En la SC Adicionar usuario).	No	Este campo no debe estar en blanco.
2	Rol	Lista desplegable	No	Se debe seleccionar un rol.

Matriz de Datos

Tabla 13: SC 2 Adicionar usuario.

Id del escenario	Escenario	V1	V2	Respuesta del sistema.	Resultado de la prueba.
EC 2.1	Asignar Rol satisfactoriamente.	*V* *maiker*	*V* *rol*	El sistema asigna el rol al usuario, mostrando un mensaje que confirma dicha acción.	Satisfactorio
EC 2.2	Se dejan campos en blanco.	*I* *( )*	*I* *( )*	El sistema muestra el mensaje “Rellene este campo”.	Satisfactorio
		/ *()*	*V* *rol*		
		V *maiker*	*/* *()*		
EC 2.3	No son válidos los campos	*I* *ma1ker*	*I* *rol*	El sistema muestra el mensaje: “Ajústese al formato	Satisfactorio

				solicitado".	
EC 2.4	Duplicado de Usuario	/ *(//)*	*V* *maiker*	El sistema muestra el mensaje: "El campo Usuario debe ser único".	Satisfactorio

Tabla 14: SC 3 Modificar usuario.

Id del escenario	Escenario	V1	V2	Respuesta del sistema.	Resultado de la prueba.
EC 3.1	Modificar usuario satisfactoriamente.	*V* *maiker*	*V* *Rol*	El sistema modifica el usuario, mostrando un mensaje que confirma dicha acción.	Satisfactorio
EC 3.2	Se dejan campos en blanco.	*I* *( )*	*I* *( )*	El sistema muestra el mensaje "Rellene este campo".	Satisfactorio
		/ *()*	*V* *Rol*		
		V *maiker*	/ *()*		
EC 3.3	No son válidos los campos	*I* *ma1ker*	*I* *Rol*	El sistema muestra el mensaje: "Ajústese al formato solicitado".	Satisfactorio
EC 3.4	Duplicado de Usuario	/ *(//)*	*V* *maiker*	El sistema muestra el mensaje: "El campo Usuario debe ser único".	Satisfactorio

Tabla 15: SC 4 Eliminar usuario.

Id del escenario	Escenario	V1	V2	Respuesta del sistema.	Resultado de la prueba.
EC 4.1	Eliminar usuario satisfactoriamente.	*V* *maiker*		El sistema elimina el usuario, mostrando un mensaje que	Satisfactorio

				confirma dicha acción.	
EC 4.2	No se confirma la eliminación del usuario.	*NA*		El sistema oculta el mensaje de confirmación y mantiene el usuario registrado en el sistema.	Satisfactorio
EC 4.3	Cerrar mensaje de confirmación.	*NA*		El sistema oculta el mensaje de confirmación y mantiene el usuario registrado en el sistema.	Satisfactorio

A continuación se muestra una gráfica resumen de las tres iteraciones de pruebas realizadas:

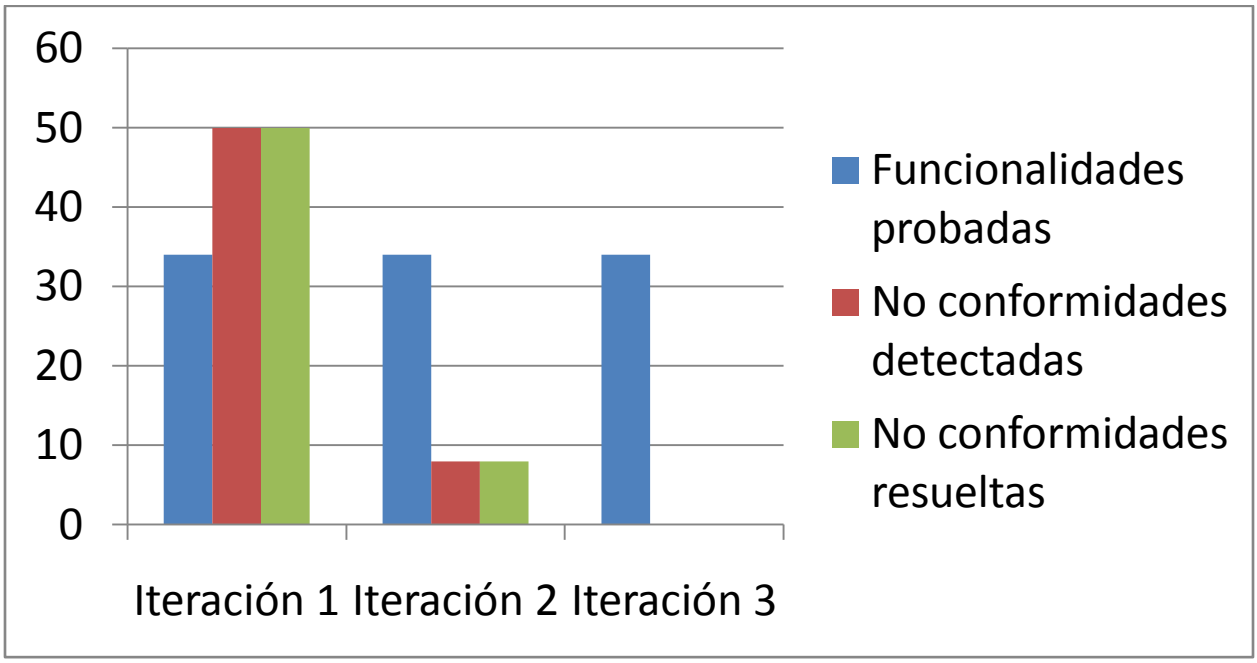


Figura 33: Iteraciones de pruebas de caja negra.