Universidad de las Ciencias Informáticas

Facultad 6

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

Título: *Herramienta Web para la Gestión de Riesgos en los proyectos del Centro GEySED.*

Autor(es): *Tamara Rodríguez Galano*

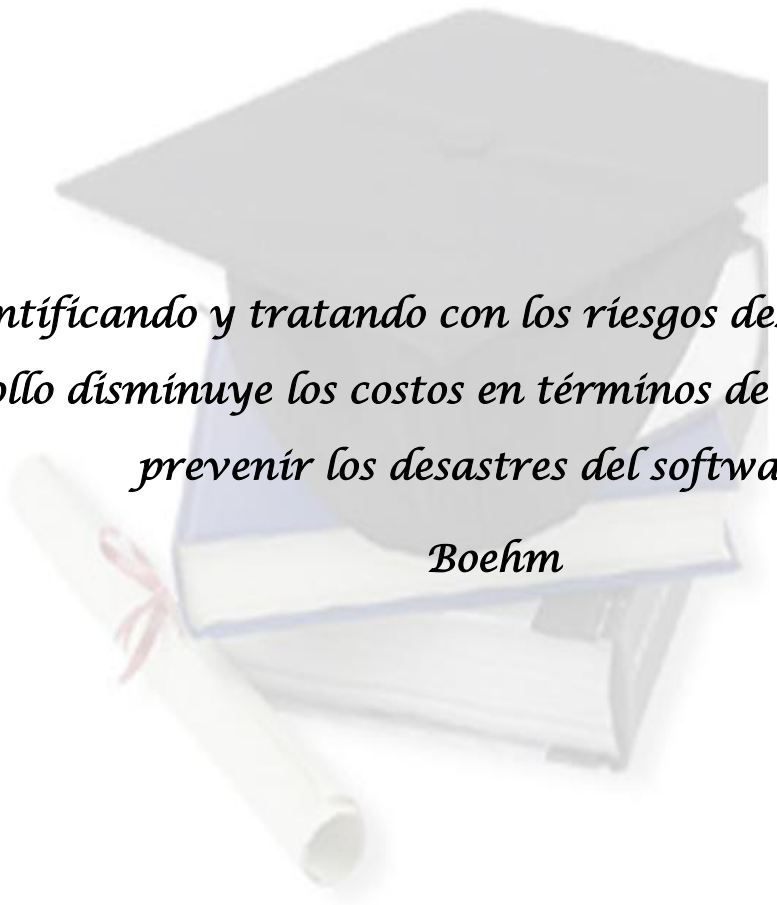
Yaima Pacheco Jorge

Tutores: *Ing. Frank Alain Castro Sierra.*

Ing. Aimé Esther Guzmán Ramírez.

La Habana, Junio 2013

Año 53 de la Revolución



"Identificando y tratando con los riesgos desde el inicio del desarrollo disminuye los costos en términos de tiempo y ayuda a prevenir los desastres del software."

Boehm

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Herramienta Web para la Gestión de Riesgos para los proyectos del Centro GEySED y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Autor
Tamara Rodríguez Galano	**Yaima Pacheco Jorge**
_____	_____
Firma del Tutor	Firma del Tutor
Ing. Frank Alain Castro Sierra	**Ing. Aimé Esther Guzmán Ramírez**

DATOS DE CONTACTO

Nombre y apellidos: Frank Alain Castro Cierra.
Categoría docente: Instructor.
Centro de trabajo: Universidad de las Ciencias Informáticas.
Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.
Año de graduación: 2007.
Años de experiencia:6
Institución donde se graduó: Universidad de las Ciencias Informáticas.
Correo electrónico:fcastro@uci.cu

Nombre y apellidos: Aimé Esther Guzmán Ramírez.
Categoría docente: Instructor.
Centro de trabajo: Universidad de las Ciencias Informáticas.
Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.
Año de graduación: 2009
Años de experiencia: 4
Institución donde se graduó: Universidad de las Ciencias Informáticas.
Correo electrónico: aguzman@uci.cu

AGRADECIMIENTOS

Tamara Rodríguez Galano

Agradezco a los profesores que han ayudado en mi preparación profesional.

A mis padres por apoyarme y ayudarme en el transcurso de la carrera.

A mi profesora y tutora Aimé Esther por guiarme y ayudarme en todo.

A mis hermanas, a mi tía Olga por guiarme y ayudarme en todo lo que me hiciera falta, a mi primo Miguel por sus buenos consejos.

A Manuel Manzor por ser una persona muy especial en mi vida.

A mis amigas Cecilia, Doina, Orisel por soportarme y estar conmigo en las buenas y en las malas.

A mi sobrinito Yan Carlos por ser la personita más especial el mundo para mí.

A todas aquellas personas que compartieron conmigo en la Universidad.

Yaima Pacheco Jorge

Agradezco a todos los que han contribuido con la realización de este trabajo especialmente:
A mi mamita y a mi papito por estar siempre conmigo, apoyarme siempre, por ser los padres más especiales del mundo, por todo el sacrificio que han hecho para que yo pueda estar aquí, por ser los que dan fuerza a mi corazón y porque son lo más importante de mi vida. A mi hermanito que aunque no esté es una de las razones por la que lucho cada día. A mis abuelitos en especial a mi abuelita Paca y Mayi, por su ternura, amor y por ser las abuelitas más buenas del mundo.
A Addiel por quererme tanto, por cuidarme siempre y por estar siempre conmigo en los buenos y malos momentos.
A mi familia en especial a mi tía Tereza, Mercedes, Maribel, Ania a mi tío Juaca, Manolo, a mis primas Lisandra, Liset y Alexander por preocuparse siempre por mí y por todo el apoyo que me han brindad. En fin a toda mi familia porque sé que todos se preocupan por mí y de una forma u otra me han apoyado siempre.

A mis amigas Yari, Elaine, Yeilen, Yissel por haberme aguantado tanto y por no dejarme vivir,

A Alexei por ser tan bueno conmigo y por comprenderme tanto. A Abuelis, Ahylin, Elizabeth, Falcón, Marcial, Edgar, Luisma, Osi, Nelson, Edgar, Osmar, Nelson, Tatiana y Raúl por ser tan lindos conmigo.
A mis vecinos por tenerme presente en todo momento.

A Ramón por ayudarme tanto, a mi tutora Aimé por haberme poyado y ayudado en todo.
En fin a todos los amigos que conocí aquí y que aunque no los vuelva a ver siempre los voy a tener presente por todos los buenos y malos momentos que pasamos juntos.

DEDICATORIA

Tamara Rodríguez Galano

A mi mamita y a mi papito por ser las personas que más quiero y los que me han dado todo su apoyo en el transcurso de la carrera.

A mis hermanos por ayudarme en todo y por darme todo su apoyo incondicional.

Yaima Pacheco Jorge

A mi mamita y a mi papito por ser las personas más importantes de mi vida y porque gracias a ellos hoy estoy aquí.

A mis abuelitos del alma por consentirme y quererme tanto.

A mi hermanito por ser mi tesorito más pequeño.

RESUMEN

En el mundo del software realizar una buena gestión de riesgos en todo el proceso de desarrollo, implica lograr un producto con buena calidad, a un menor costo y que se entregue en tiempo. Comprender los riesgos que se identifican y tomar medidas para evitarlos o gestionarlos, es un elemento clave de una buena gestión de proyecto de software.

El presente trabajo investigativo ha tenido como objetivo general realizar una herramienta web para la gestión de riesgos en los proyectos del centro Geoinformática y Señales Digitales (GEySED), con el fin de obtener una mayor organización de toda la información referente a los riesgos que se manejan en los proyectos a la hora de su identificación y análisis, así como funcionalidades claves dadas a partir de los riesgos, características y necesidades de dicho centro. Para alcanzar este objetivo, se realizó un estudio de los diferentes enfoques de la gestión de riesgos según las metodologías y modelos existentes en el mundo, escogiéndose el modelo MoGeRi, debido a que tiene sus procesos bien definidos y estructurados. Para el desarrollo de la solución se utilizó RUP como metodología de desarrollo, UML como lenguaje de modelado, PHP como lenguaje de programación, Symfony 2.0 como framework de desarrollo, Apache como servidor de aplicaciones web, NetBeans como entorno integrado de desarrollo y PostgreSQL como gestor de base de datos. Lo que facilitó la implementación de un sistema capaz de gestionar todos los procesos que se llevan a cabo en la gestión de riesgos en el centro GEySED a partir de sus particularidades.

Palabras clave: Gestión de Riesgo, Riesgo, identificación de riesgo, análisis de riesgo, MoGeRi.

ÍNDICE

ÌNDICE DE TABLAS

ÌNDICE DE FIGURAS

INTRODUCCIÓN

Un factor clave para el avance de la sociedad ha sido el desarrollo de la informática y las comunicaciones. Esto ha permitido alcanzar una nueva era digital extendiendo el uso del software a las más disímiles esferas. La producción de software se ha hecho cada vez más compleja, por lo que para crear productos con la calidad requerida, en poco tiempo y con el menor coste posible, se ha hecho necesario llevar un control adecuado de los procesos de desarrollo de software.

Cuba, a pesar de ser un país subdesarrollado, está en vías de un desarrollo económico y tecnológico exitoso. Es por ello que la máxima dirección del gobierno cubano ha puesto en práctica un conjunto de acciones con el fin de brindar una mayor accesibilidad a las Tecnologías de la Información y las Comunicaciones (TIC). Como una de estas acciones nace la Universidad de las Ciencias Informáticas (UCI), cuya misión es desarrollar software y servicios informáticos a partir del trabajo conjunto entre estudiantes y profesionales para promover la informatización de la sociedad.

Dicha institución cuenta con 6 facultades donde, en cada una de ellas, existen centros de desarrollo de software (CDS), entre ellas está la facultad 6 que tiene a los centros GEySED (Geoinformática y Señales Digitales) y DATEC (Centro de Tecnologías de Gestión de Datos). Específicamente en el centro GEySED, el incumplimiento de los cronogramas de trabajo por parte de los involucrados en el desarrollo de los proyectos; la necesidad de mayores recursos de hardware; la mala definición de los roles y responsabilidades asignados a los miembros del equipo; las tareas asignadas de último momento que no aparecen en los planes de trabajo tanto individuales, del centro, así como de la facultad y la universidad; la poca o mediana motivación por parte de los miembros del proyecto; los cambios que suceden periódicamente y se modifican sustancialmente, afectan la culminación del proyecto y ocasionan variaciones en los objetivos del mismo que pueden ocasionar consecuencias no previstas.

Además, no tienen definido un proceso para la identificación de riesgos efectivo. Como mínimo, se realizan propuestas de un conjunto de riesgos que son incluidos en la lista de registro de riesgos del proyecto, que es un documento en formato Excel ubicado en el expediente de proyecto. No incluyen un proceso para el seguimiento de los riesgos, pues no actualizan el estado de los mismos después de su definición ni desarrollan acciones encaminadas a disminuir la posibilidad de ocurrencia de estos. Lo antes mencionado provoca situaciones negativas como son, el incumplimiento en los cronogramas, los resultados obtenidos no esperados o alejados de las necesidades reales de los usuarios finales e

insatisfacciones en el equipo de desarrollo a partir de la no identificación y minimización de los riesgos asociados a la estabilidad y aseguramiento de los miembros del proyecto.

Todos estos cambios o eventos inesperados que pueden ocasionar daños se les conocen como riesgos y a los procesos encargados de identificar, planificar y analizar los riesgos, así como de controlar las actividades planificadas al respecto, se les denomina gestión de riesgos (GR).

Lo anterior demuestra la necesidad de una adecuada identificación, análisis, planificación y seguimiento de los riesgos dentro del centro. Por tal razón se plantea la siguiente interrogante como **problema a resolver:** ¿Cómo mejorar la gestión de riesgos en el centro GEySED?

Se define como **objeto de estudio** los procesos de identificación, análisis, planificación, seguimiento y control de los riesgos enmarcado en el **campo de acción** los procesos de identificación, análisis, planificación, seguimiento y control de los riesgos en el centro GEySED.

Para darle solución al problema planteado se define como **objetivo general** desarrollar una herramienta web para la gestión de riesgos en los proyectos del centro GEySED.

Para el cual se plantea la siguiente **idea a defender:**

Con el desarrollo de una herramienta para la gestión de riesgos se mejorará el proceso de identificación, análisis, planificación, seguimiento y control de los riesgos en los proyectos del centro GEySED.

Para dar cumplimiento al objetivo trazado y guiar la presente investigación se proponen las siguientes**tareas de la investigación**:

1- Analizar los conceptos más significativos asociados al tema tratado en la investigación.
2- Caracterizar el proceso de gestión de riesgos en los proyectos productivos del centro GEySED.
3- Analizar el estado del arte y tendencias sobre herramientas que permiten la gestión de riesgos.
4- Determinar los métodos, herramientas y procedimientos factibles para el desarrollo del sistema.
5- Analizar, diseñar e implementar la solución propuesta.
6- Realizar las pruebas funcionales a la aplicación para validar la correspondencia con los requisitos del sistema.

Con la realización de las tareas investigativas anteriormente mencionadas se desean obtener los siguientes **resultados**:

1- Un sistema capaz de ejecutar de manera correcta la gestión de riesgos de los proyectos del centro GEySED.
2- Lograr una mayor organización, control y seguimiento de los riesgos detectados en los proyectos del centro GEySED.

En el transcurso de la investigación se emplean una serie de métodos los cuales permiten de manera significativa el desarrollo de la misma, debido a que contribuyen a definir los elementos necesarios para encontrar la solución del problema a resolver. Para llevar a cabo la presente investigación se utilizarán los siguientes **métodos científicos**.

- **Teóricos**

Histórico-Lógico: El método histórico estudia la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia. El método lógico investiga las leyes generales del funcionamiento y desarrollo de los fenómenos (Hernández León, y otros, 2011).

Este método se utilizó para estudiar la evolución de los conceptos asociados a la gestión de riesgos, herramientas existentes así como los principales modelos de gestión de riesgos conocidos, permitiendo la selección de los conceptos más completos para arribar a conceptos propios.

Analítico-Sintético: Permite la descomposición de un todo complejo en sus partes y cualidades. La síntesis, por su parte, establece la unión entre las partes, previamente analizadas y posibilita descubrir relaciones y características generales entre los elementos de la realidad (Hernández León, y otros, 2011).

Este método se utilizó para la evaluación de soluciones existentes que contribuyan a dar respuesta al problema, así como también las metodologías a estudiar y las herramientas, permitiendo realizar una valoración crítica y detallada de cada una de ellas.

Modelación: Este método ayudó a la modelación de diagramas para lograr un mejor entendimiento de lo que se va a implementar.

- **Empíricos**

Entrevista: La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información(Hernández León, y otros, 2011).

Dentro de los tipos de entrevista utilizados se encuentran:

- Según el modo

Entrevista cara a cara: estas entrevistas se realizan en persona, lo que facilita la toma de nota y una buena calidad en las respuestas.

Entrevista por correo electrónico: a pesar de ser muy impersonales, le dan al entrevistado la posibilidad de planificar cada una de las respuestas a dar.

-En función de la técnica de interrogar

Entrevista estandarizada: esta entrevista, también llamada estructurada, utiliza una lista de preguntas predefinida con anterioridad.
Este método se utilizó para obtener información valiosa acerca del proceso de gestión de riesgos (métodos, herramientas, etc.). Para ello se entrevistaron varios Jefes de Proyecto del centro GEySED y a los miembros del grupo de Calidad del mismo.

La presente investigación estará conformada por 4 capítulos los que se distribuyen de la siguiente manera:

Capítulo 1: "Fundamentos teóricos". Contendrá todo lo referido al marco teórico de la investigación, conceptos asociados, la revisión y actualización del estado del arte de la gestión de riesgos, los procesos de la gestión de riesgos, los principales modelos de gestión de riesgos existentes.

Capítulo 2:"Tecnologías y Tendencias Actuales". En este capítulo se identifican las herramientas disponibles en la actualidad; la selección del modelo de gestión de riesgos, la metodología, herramientas y tecnologías a utilizar para la realización de la solución propuesta.

Capítulo 3:"Diseño del Sistema". En este capítulo se identifican los requisitos funcionales y no funcionales del sistema propuesto. Se realiza el modelado del sistema, generándose los artefactos correspondientes. Se muestra el diseño de la solución propuesta según la metodología, el framework de desarrollo y la arquitectura seleccionada.

Capítulo 4: "Implementación y Pruebas". En este capítulo se presentan los artefactos generados durante el flujo de trabajo de implementación entre los que se encuentran los diagramas de componentes, el diagrama de despliegue y las pruebas realizadas al sistema para la validación.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

Introducción

Este capítulo tiene como objetivo principal plantear la base y la estructura del marco teórico del presente trabajo de diploma. En él se realiza una breve descripción sobre la investigación: los conceptos asociados a la misma, la revisión y actualización 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. Además, se selecciona de acuerdo a las características, ventajas y desventajas el modelo para la gestión de riesgos indicado para aplicar en la herramienta que se desea desarrollar. Se definen las tecnologías y herramientas, así como el lenguaje de programación a utilizar en el desarrollo de la misma.

1.1. Conceptos básicos asociados

Riesgos

Aunque hay un considerable debate en torno a la definición para el riesgo de software, existe un acuerdo general en que el riesgo siempre involucra dos características (Pressman, 2005):

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

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

Por su 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 (Audrey, 2009)

Robert Charette en su libro sobre análisis y gestión de riesgo, define que:

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).

Después de haber analizado los conceptos de riesgos expuestos por diferentes autores, se puede llegar a la conclusión que la variante más completa es la propuesta por SEI ya que incluye el término referido a la

probabilidad de ocurrencia del riesgo, un factor importante y casi decisivo para su posterior y consecuente tratamiento. Un riesgo se puede definir como la posibilidad de sufrir una pérdida provocando un impacto negativo a un proyecto.

Gestión de Riesgos

Para definir el término Gestión de Riesgos (GR) se ha hecho referencia a diferentes denominaciones realizadas por distintos autores:

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 el proceso formal en que los factores de riesgos son sistemáticamente identificados, evaluados y mitigados; se inicia en la primera etapa de un proyecto de software durante la exploración de conceptos y se desarrolla a lo largo de todo su ciclo de vida (hasta la aceptación del producto del proyecto)(Higuera, 1996).

Romero plantea que la GR en proyectos de software pretende identificar, estudiar y eliminar las fuentes de riesgo antes de que comiencen a amenazar el éxito o la finalización exitosa de un proyecto de desarrollo de software (Romero B., y otros, 2007).

Se puede decir que la gestión de riesgos permite definir en forma estructurada, operacional y organizacional, una serie de actividades para gestionar los riesgos de los proyectos a lo largo de todas las fases de su ciclo de vida de desarrollo de software.

Análisis de Riesgo

El primer paso en la GR es el análisis de riesgo que tiene como propósito determinar los componentes de un sistema que requieren protección, sus vulnerabilidades que los debilitan y las amenazas que lo ponen en peligro, con el fin de valorar su grado de riesgo(Erb, 2008).

Identificación de Riesgos

La identificación del riesgo es un intento sistemático para especificar las amenazas al plan del proyecto (estimaciones, planificación temporal, carga de recursos). Identificando los riesgos conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario(Menéndez, 2002).

1.2. Estrategias de riesgo

En la GR se han considerado dos formas de clasificar las estrategias para controlar los mismos: reactivas y proactivas.

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).

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).

De las anteriores estrategias se utilizará en el presente trabajo de diploma, la gestión de riesgo proactiva, por ser más inteligente y eficaz pues trata el riesgo desde el inicio antes que ocurra lo peor propiciando así que el equipo de software tenga un buen control y seguimiento de los riesgos.

1.3. Procesos y actividades para la Gestión de Riesgos

El proceso de la GR consta de 6 procesos: Planificación de la gestión de riesgos, 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.

Proceso Identificación de Riesgos

La fase de identificación de riesgos consiste en descubrir factores de riesgo antes de que estos lleguen a ser 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).

Proceso Análisis de los riesgos

Este proceso tiene como objetivo fundamental evaluar y categorizar cada riesgo identificado, usando las categorías y los parámetros definidos del riesgo, y determinar su prioridad relativa.

- ✓ Análisis cualitativo de riesgos

Se examina la lista de riesgos obtenida en la identificación y se les asigna una prioridad, registrando el orden final de la lista. Usando esta, se pueden determinar los riesgos más importantes y reservar recursos para planificar y ejecutar una estrategia específica. Se pueden identificar riesgos que, por su poca prioridad, pueden ser eliminados de la lista. A 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 modificará. Pueden que surjan nuevos riesgos y puede que los riesgos más antiguos que han bajado de prioridad se eliminen(Quintero Acosta, y otros, 2009).

Para la realización de esta actividad se deben ejecutar los siguientes pasos:

- Evaluación de la probabilidad e impacto de los riesgos: La evaluación de probabilidad de los riesgos investiga la probabilidad de ocurrencia de cada riesgo inespecífico. La evaluación del impacto de los riesgos investiga el posible efecto sobre un objetivo del proyecto, como tiempo, coste, alcance o calidad, incluidos tanto los efectos negativos por las amenazas que implican, como los efectos positivos por las oportunidades que generan. La probabilidad de un riesgo debe ser mayor que cero o el riesgo no presenta una amenaza para el proyecto. Las probabilidades son difíciles de calcular y aplicar; sin embargo, la mayoría de los equipos de proyectos puede expresar con palabras sus experiencias, interpretar los informes y proporcionar una amplia gama de expresiones de lenguaje natural para indicar rangos de probabilidad. La tabla siguiente describe la escala a utilizar cuando se define la probabilidad.

Probabilidad		Impacto	
Seguro	100%	Muy alto	10
Muy probable	80%	Importante	8

Puede ser	60%	Menor	6
Quizás	40%	Pequeño	4
Poco probable	30%	Mínimo	2
Improbable	10%	No hay efecto	0

Tabla 1: Definición de la escala de probabilidad de ocurrencia

- Categorización de riesgos:

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

  Los Riesgos técnicos (RT): amenazan la calidad y la planificación temporal del software que hay que producir. Si un riesgo técnico llegara a convertirse en realidad, la implementación puede llegar a ser difícil o imposible.

  Riesgos del negocio (RN): ocurren porque el problema es más difícil de resolver de lo que se piensa. Los riesgos del negocio amenazan la viabilidad del software a construir. Estos a menudo ponen en peligro el proyecto o el producto.

- Estimar la exposición al riesgo: Una vez identificados los riesgos, el siguiente paso es analizar cada riesgo para determinar su impacto. Al multiplicar la posibilidad de ocurrencia de un riesgo con el efecto si el mismo ocurre, se obtiene la exposición que tiene el proyecto al mismo. Es decir la vulnerabilidad que tiene el proyecto a que este riesgo lo pueda afectar.

- Priorización del riesgo: La prioridad de cada riesgo es muy importante porque da una visión clara a la hora de realizar el plan de mitigación.

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

✓ Análisis cuantitativo de riesgos

Este análisis se realiza a los riesgos priorizados en el análisis cualitativo, analiza el efecto de estos riesgos y les asigna una calificación numérica. Proponen técnicas para cuantificar los posibles resultados del proyecto y sus probabilidades (por concepto de los riesgos identificados), evaluar la probabilidad de

cumplir un objetivo, cuantificar el riesgo general del proyecto e identificar objetivos viables por concepto de riesgos(Quintero Acosta, y otros, 2009).

Es necesario acotar que el análisis cuantitativo precisa de mucho tiempo para su realización, los cálculos necesarios son muy complejos y sus resultados son expresados en términos monetarios lo que quiere decir que el proyecto debería tener una estimación de costos realizada. Debido a esto para el desarrollo de la herramienta se realiza solamente el análisis cualitativo de los riesgos.

Proceso Planificación

Este proceso tiene como objetivo fundamental desarrollar un plan de mitigación para 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, y otros, 2009).

Entre las actividades de este proceso se encuentran:

- ✓ 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.

Proceso Seguimiento y Control de los riesgos

El seguimiento y el control es una etapa esencial e integral en el proceso de Gestión de Riesgos(Quintero Acosta, y otros, 2009).

En este proceso 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 si(Hechavarria Guibert, y otros, 2008):

- Las asunciones del proyecto aún son válidas.

- El riesgo, según fue evaluado, ha cambiado de su estado anterior, a través del análisis de tendencias.
- Se están siguiendo políticas y procedimientos de Gestión de Riesgos correctos.

1.3.1. Procesos y actividades para la gestión de riesgos en los proyectos del centro GEySED

En el centro GEySED no se realizan todos los procesos y actividades para la gestión de riesgos de sus proyectos. A continuación se dará una descripción de cómo se llevan a cabo.

Proceso Identificación

Describen factores de riesgo antes de que estos lleguen a ser problemas y deriven en daños o pérdidas. Se hace en un documento en formato Excel denominado Plan de Registro y Monitoreo que pertenece al Plan de Mejora del Expediente de Proyecto. Para ello se tiene en cuenta el nombre del riesgo, su tipología, una breve descripción del mismo y las consecuencias que provocaría si llegara a ocurrir.

Proceso Análisis

Evalúan y categorizan cada riesgo identificado, usando las categorías y los parámetros definidos del riesgo y determinan su prioridad relativa. Esto se evidencia en el impacto y la probabilidad de ocurrencia del riesgo.

Proceso Planificación

Realizan el plan de mitigación para los riesgos, pero se tienen en cuenta todos los riesgos del proyecto y no los más importantes dado su criticidad.

Proceso Seguimiento y control

No le dan un seguimiento sistemático a los riesgos identificados, pues no notifican las desviaciones y no actualizan el estado de los riesgos. Además, no se plantean alternativas de cómo controlar los riesgos que puedan aparecer en el entorno a medida que el proyecto avanza, utilizando las estrategias de amenaza (evitar, transferir y mitigar).

1.4. Modelos de Gestión de Riesgos

Existen diferentes modelos orientados a la GR. El análisis de los riesgos es una actividad común para estos modelos, cada uno de ellos plantea la estimación de la probabilidad de ocurrencia de los riesgos y el impacto de estos.

1.4.1. Modelo Gestión de Riesgos en Proyectos de Desarrollo de Software (MoGeRi)

Zulueta, después de haber realizado un estudio profundo del tema de gestión de riesgos en el mundo, en Cuba y, particularmente en la UCI, propone el modelo MoGeRi para la gestión de riesgos en los proyectos de desarrollo de software, que consta de seis procesos: Planificación de la gestión de riesgos; Identificación de los riesgos; Análisis de los riesgos; Planificación de las respuestas; Seguimiento y control y Comunicación de la información sobre los riesgos. Propone un marco para la gestión de riesgos tomando elementos del SEI[1] y de PMI[2], por su reconocimiento, integralidad, actualidad y aplicación internacional y teniendo en cuenta las características propias de la UCI. Para cada proceso definido se describen sus actividades y tareas(Zulueta V., 2007).

Las actividades correspondientes a cada proceso son:

Planificación de la Gestión de los Riesgos.

- Determinación del alcance.
- Planificación de la GR.
- Factibilidad de la GR.
- Comunicación de resultados.

Identificación de los riesgos.

- Selección de herramientas y técnicas a aplicar.
- Identificación de riesgos.
- Comunicación de resultados.

Análisis de los riesgos

- Análisis de los riesgos.
- Priorización de los riesgos.
- Comunicación de resultados.

Respuestas a los riesgos.

- Valoración de la estrategia para enfrentar el riesgo.
- Planificación de las respuestas.
- Comunicación de resultados.

[1] Software Engineering Institute
[2] Project Management Institute

Seguimiento y control de los riesgos.

- Control del cumplimiento de las respuestas a los riesgos.
- Control del cumplimiento de los hitos de GR.
- Aplicación de métricas para valoración de la calidad de procesos, técnicas, herramientas y resultados.

Comunicación de la información sobre los riesgos.

La comunicación debe ser continua desde el inicio de la GR, lo cual puede apreciarse con la inclusión de una actividad al respecto en cada uno de los procesos anteriormente descritos. Pero no es solo un canal para que fluyan datos en el proyecto, la comunicación de manera formal y reutilizable: el mismo proyecto y otros, podrán utilizarla como información histórica y aprender de ella.

MoGeRi ha sido complementado con una Guía de Métricas que permite realizar valoraciones sobre el costo de la GR, la efectividad de las herramientas y técnicas empleadas, las facilidades para desarrollar los procesos y actividades, la idoneidad de la definición de los roles, el nivel de conocimiento con que cuenta el personal de las responsabilidades y actividades que le han sido asignadas, y por supuesto, sobre el desenvolvimiento de la GR en el proyecto de manera general(Zulueta V., 2007).

1.4.2. Modelo Integrado de Capacidad y Madurez (Capability Maturity Model Integration, CMMI)

CMMI se ha convertido en el nuevo estándar a nivel mundial para la medición de la calidad de los procesos de desarrollo de software, presenta como una de sus PA (Áreas de Procesos) fundamentales, de Nivel 3, la Gestión de Riesgos, que es un proceso continuo, de mirar hacia delante que constituye una parte importante de la gestión; debe dirigirse a problemas que pudieran poner en peligro el logro de objetivos críticos; es aplicado para anticiparse eficazmente y mitigar los riesgos que pueden tener un impacto crítico en el proyecto.

El Área de Procesos de Gestión de Riesgos de CMMI describe prácticas específicas para sistemáticamente planear, anticipar y mitigar los riesgos para minimizar de forma proactiva su impacto en el proyecto. Aunque el énfasis primario del Área de Procesos de Gestión de Riesgos es en el proyecto, los conceptos pueden ser aplicados para gestionar los riesgos organizacionales(Valladares Arenas, 2010).

Entre sus fortalezas se destacan:

- Inclusión de las prácticas de institucionalización, que permiten asegurar que los procesos asociados con cada área de proceso serán efectivos, repetibles y duraderos.
- Guía paso a paso para la mejora, a través de niveles de madurez y capacidad (frente a ISO).

- Transición del 'aprendizaje individual' al 'aprendizaje de la organización' por mejora continua, lecciones aprendidas y uso de bibliotecas y bases de datos de proyectos mejorados.

Para llevar a cabo una exitosa gestión de riesgos CMMI plantea las siguientes metas genéricas y específicas.

Tareas Genéricas

- Prepararse para la gestión de riesgos.
- Identificar y analizar los riesgos.
- Mitigar riesgos.
- Análisis y resolución de toma de decisiones.

Tareas Específicas

- Determinar la Fuentes y categorías de los riesgos.
- Definir los parámetros de los riesgos.
- Establecer la estrategia de gestión de riesgos.
- Identificar riesgos.
- Evaluar, categorizar y priorizar los riesgos.
- Desarrollar los planes de la mitigación de los riesgos.
- Implementación del plan de gestión de riesgos.

1.4.3. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información de las Administraciones Públicas (MAGERIT)

Desarrollada por el Ministerio de Administraciones Públicas de España, es una metodología enfocada a la seguridad del sistema en el desarrollo del mismo. Permite un seguimiento exhaustivo de la seguridad del sistema ajustándose a criterios como los de ITSEC o Criterios Comunes de Evaluación de la Seguridad de los Productos y Sistemas de Información, que permitan la posterior homologación y certificación del sistema de información desde el punto de vista de la seguridad. Consta de tres fases, que incluyen actividades y tareas(Valladares Arenas, 2010). (Ver Figura 1)

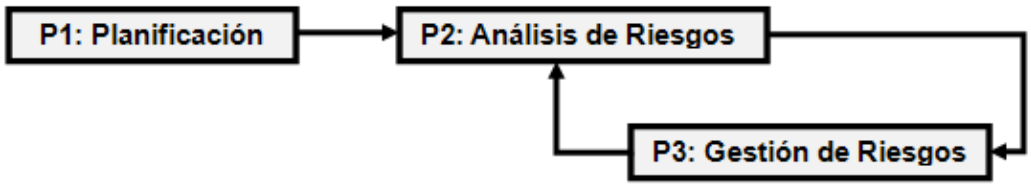


Figura 1: Modelo de procesos MAGERIT

Planificación del Proyecto de Riesgos: Estimaciones iniciales de los riesgos que pueden afectar al sistema de información así como del tiempo y los recursos que su tratamiento conllevará. El objetivo principal de este proceso es establecer el marco general de referencia para todo el proyecto.

Análisis de Riesgos: Se estima el impacto que tendrán los riesgos en la organización.

Gestión de riesgos: Se procesan los impactos y riesgos identificados en el proceso anterior, asumiéndolos y afrontándolos bien, para afrontar los riesgos que se consideren inaceptables se llevará a cabo un plan de seguridad que corrija la situación actual.

1.4.4. Metodología para la Gestión de Riesgos en LPS (Ligerip)

Ligerip se caracteriza por la descripción detallada y paso a paso de cada una de las acciones a realizar durante el desarrollo de los distintos proyectos de software de la LPS, por su fácil implementación y comprensión así como por lo poco extensa y extenuante de su ejecución lo cual asegura la eficacia del proceso, muy a tono con el desarrollo basado en LPS, donde la rapidez de la puesta a punto –sin menguar la calidad- es un factor decisivo(Martell Fernández, 2012). (Ver figura 2)

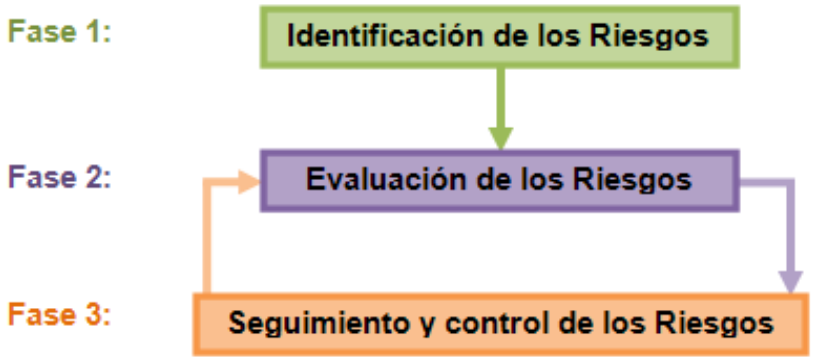


Figura 2: Fases por las que está compuesta la metodología Ligerip

Esta metodología se divide en tres fases: identificación de los riesgos, evaluación de los riesgos, seguimiento y control de los riesgos, las cuales se componen de un conjunto de actividades que a su vez

cuentan con una serie de acciones que son en definitiva el eje impulsor. Las acciones muestran qué hacer en cada momento, de manera que su realización con la calidad requerida y en el tiempo establecido garantizará una gestión de riesgos de acuerdo con las necesidades de la organización y como consecuencia un impacto positivo en el cumplimiento de los objetivos(Martell Fernández, 2012).

Comparación de los Modelos de gestión de riesgos

En las secciones anteriores se hizo referencia a las características de algunos de los principales modelos de Gestión de Riesgos. Todos ellos convergen en que cuentan con un objetivo común, lograr la calidad del producto en las empresas productoras de software. La elección de qué modelo usar se basa fundamentalmente en las diferencias existentes entre ellos, las cuales pueden considerarse ventajas o desventajas, en dependencia del proyecto en que se vayan a utilizar. A continuación se muestra una tabla comparativa para seleccionar el modelo factible para la realización de la herramienta a implementar mediante los procesos que utilizan para la gestión de riesgos. Para realizar esta comparación se tienen en cuenta los siguientes procesos de GR: Planificación (P), Identificación (I), Análisis(A), Planificación de las respuestas(R), Seguimiento y Control (S-C), Análisis cuantitativo de los riesgos(ACN), Análisis cualitativo de los riesgos(ACL). (Ver Tabla 2)

Modelo	P	I	R	S-C	ACN	ACL	Métricas
MoGeRi	X	X	X	X	X	X	Caracterizar los riesgos. Medir los resultados. Mejorar los resultados.
CMMI	X	X	X	X	X	X	Caracterizar los riesgos.
MAGERIT	X	X	X	X			Caracterizar los activos, las amenazas y los
Ligerip		X		X			Caracterizar los riesgos.

Tabla 2: Comparación entre los modelos de GR

Como se puede apreciar, algunos de estos modelos no cumplen con todas las características y necesidades del centro GEySED. MoGeRi es el modelo seleccionado como guía para la realización de la herramienta de GR porque contiene dentro de sus actividades la elaboración de un Plan de Gestión que

permite determinar la viabilidad del proyecto. Tiene sus procesos bien definidos y detallados los cuales cuentan con una serie de actividades que a su vez están conformadas por una serie de tareas permitiendo así una buena comprensión y realización de la GR.Tiene facilidades para desarrollar los procesos y actividades, así como la idoneidad de la definición de roles, el nivel de conocimiento con que cuenta el personal y sobre el desenvolvimiento de la GR en el proyecto de manera general.

1.5. Soluciones informáticas existentes

En la actualidad existen herramientas de software para la de gestión de riesgos disponibles en el mercado, pero en su gran mayoría solo se enfocan en una categoría de riesgos (TRIMS – Technical Risk Identification and Mitigation System) o están orientadas a compañías maduras que poseen una amplia base de datos organizacional que les permite generar información de categorías propias de riesgos (RiskTrak y Welcome Risk).

- A nivel internacional:

Enterprise Risk Management (ERM) Suite - Gestión Riesgos Corporativos

El Soft Expert ERM Suite - Gestión Riesgos Corporativos - permite a las empresas identificar, analizar, evaluar, monitorizar y administrar sus riesgos corporativos de manera integrada. A través de un mismo abordaje son tratados todos los datos relacionados al riesgo - biblioteca reutilizable de riesgos, controles y evaluaciones, eventos de pérdida y no conformidades, indicadores de riesgo, acciones y planes de tratamiento - en una solución única y de alcance.

La solución sirve como base para la gestión de riesgos corporativos de una empresa a través de la habilidad de unificar y soportar diferentes categorías de riesgo - financiero, seguridad, conformidad, planificación - estando en conformidad con otras soluciones relacionadas a la gestión de riesgos como Financial Control Management, Operational Risk Management, IT Risk Management y General Compliance Management (Software ERM).

Risk Advisor

Es una herramienta que asiste paso a paso la implementación del Sistema de Administración Integral del Riesgo (SAIR), en todos los procesos, proyectos u otras actividades críticas para el logro de los objetivos y metas estratégicas de una empresa. Permite identificar el nivel real de exposición de riesgos de la organización y generar diversos mapas de riesgos a diferentes niveles. Combina proactivamente la información sobre las amenazas, vulnerabilidades y medidas correctoras, para determinar con exactitud

qué activos corren verdaderamente riesgo. Despeja las dudas sobre cuándo y dónde se debe centrar esfuerzos de seguridad, con lo que se ahorra tiempo y dinero.

Este sistema provee además una metodología y una guía genérica para la implementación del proceso de gestión de riesgos, considerando todas las etapas: entender el contexto, identificar, analizar, valorar, tratar y monitorear todos los riesgos que puedan afectar los objetivos y metas de la organización.

- **Active Risk Manager (ARM)**

Active Risk Manager (ARM), herramienta creada sobre la concepción de un sistema que gestiona los riesgos de una organización entera, así como las consecuencias que puede ocasionar el efecto de una de las diferentes unidades dentro de la empresa en el desarrollo de esta, considerando todos los componentes humanos y tecnológicos dentro de un negocio.

ARM permite la identificación, registro, adición, evaluación, gerenciamiento y monitoreo de riesgos y oportunidades, de modo que las compañías tengan la información y las estrategias para balancear riesgos contra recompensas. Este software es usado en el mundo por 25 000 usuarios, localizados en 5 continentes, un contrato con su proveedor StrategicThought tiene un costo muy alto.

- **Redmine**

Herramienta web de software libre bajo licencia GPL (GNU General Public License v2) con grandes perspectivas en la gestión de proyectos. Es capaz de soportar distintos tipos de proyectos y los usuarios que acceden tienen distintas funciones según el rol asignado, ya sea como usuario, jefe de proyecto o administrador, todo ello en función de un sistema de permisos. Consta de un foro donde los participantes de la comunidad pueden publicar sus dudas o conocimientos. Presenta una integración con manejadores de configuración de código, este es el caso del SVN (Subversion) y el CVS.

Permite a los líderes de proyectos la GR por lo que es posible identificar los riegos que puedan ocasionar problemas y que estos deriven en daños o pérdidas. Cada proyecto puede llevar asociado, si así lo desea, documentos, archivos o noticias. Además, se establece un sistema de notificaciones para los usuarios mediante correo electrónico ya sea porque se le ha asignado una tarea o porque una parte del proyecto ha sido modificada. Cuenta también, con el seguimiento en tiempo real de cada uno de los proyectos que contiene.

Valoración de los autores

Las herramientas mencionadas y descritas anteriormente posibilitan la gestión de los riesgos de un proyecto, pero están patentizadas por empresas privadas; enfocan su trabajo solamente a una categoría de los riesgos, cuando existen varias categorías y pueden surgir otras nuevas y todas se trabajan de manera diferente, por lo que una sola categoría no sería suficiente para poder tener bien controlado todo el ciclo de vida del riesgo ya que no realiza el proceso completo de la gestión de riesgos.

En el caso específico del Redmine, actual gestor de proyecto de la UCI a pesar de ser una herramienta muy buena ya que realiza diferentes funcionalidades para los proyectos, no determina la situación de los riesgos en los proyecto, así como la idoneidad del plan de contingencia, es decir no pone en práctica ninguna de las métricas existentes para la GR.

Conclusiones del capítulo

Después de haberse realizado un exhaustivo estudio del marco teórico-conceptual y de haber abordado conceptos relacionados con la gestión de riesgos se obtuvo un mayor entendimiento del problema en cuestión. Quedó clara la importancia de la gestión de los riesgos y el alto valor que tiene la utilización de un modelo bien estructurado y detallado para administrar los riesgos en un proyecto, lo que permitió realizar una comparación entre ellos y que el modelo seleccionado fuera MoGeRi, el cual sirvió de guía para tener los procesos y las actividades organizadas y bien definidas.

El análisis y valoración de las soluciones existentes, tanto a nivel internacional como nacional, posibilitó determinar que las mismas no realizan el proceso completo de gestión de riesgos, a excepción del Redmine, pero lo que sí no hace ninguna es determinar la situación de los riesgos, así como la idoneidad del plan de contingencia.

CAPÍTULO 2: TECNOLOGÍAS Y TENDENCIAS ACTUALES

Introducción

En este capítulo se brinda una breve caracterización de las herramientas, tecnologías y metodologías, con el fin de demostrar el porqué de su uso en el desarrollo de la solución propuesta, agrupadas en: lenguaje de modelado, metodología de desarrollo, herramienta case, arquitectura, framework de desarrollo, servidor web, gestor de base de datos y lenguaje de programación, tanto del lado del cliente como del lado del servidor.

2.1. Metodología de desarrollo de software

La metodología 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 (Jacobson, 2005).

RUP

RUP es una metodología guiada por casos de uso, es centrada en la arquitectura, iterativa e incremental. 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(Jacobson, 2005)**.**

RUP propone crear una lista de riesgos en la fase de inicio con el siguiente esquema para facilitar su administración:

Descripción: comienza con una breve descripción y se van añadiendo detalles conforme se va aprendiendo.

Prioridad: se le asigna una prioridad al riesgo: crítico, significativo o rutinario.

Impacto: qué partes del proyecto o del sistema se verán afectados por el riesgo.

Monitor: quién es responsable del seguimiento de un riesgo persistente.

Responsabilidad: qué individuo o unidad de la organización es responsable de eliminar el riesgo.

Contingencia: lo que ha de hacerse en caso de que el riesgo se materialice.

Al explicar los pasos a seguir en cada fase del proyecto, en RUP se perciben las siguientes actividades relacionadas con los riesgos:

Inicio: Identificar los riesgos críticos, es decir, los que afectan la capacidad de construir el sistema y determinar si puede encontrarse una forma de mitigarlos, quizás en una etapa posterior. En esta fase se consideran solo los riesgos que afectan la viabilidad del sistema. Los no críticos son colocados en la lista de riesgos.

Elaboración: Identifica los riesgos significativos, los que podrían perturbar los planes, costes y planificaciones de fases posteriores y los reduce a actividades que pueden ser medidas y presupuestadas.

Construcción: Materializar la monitorización de los riesgos críticos y significativos arrastrados desde las dos primeras fases y su mitigación.

Transición: No se definen tareas relacionadas con los riesgos.

Valoración de los autores

Para la realización de la herramienta se tomó la metodología RUP ya que describe qué se va a producir, las habilidades necesarias requeridas y la explicación paso a paso describiendo cómo se consiguen los objetivos de desarrollo del software. Esta metodología documenta los riesgos, los agrega en su lista de riesgo, les da tratamiento a los críticos y documenta las acciones a tomar. RUP, en cada ciclo entrega diferentes documentos al cliente que permiten evaluar los diferentes aspectos del proyecto uno de estos documentos es el registro de riesgo. Además es la metodología que usa el centro de GEySED de la Universidad de las Ciencias Informáticas para la cual se desarrolla la herramienta.

2.2. Lenguaje de modelado (UML 2.0)

Un lenguaje de modelado es un lenguaje artificial diseñado para expresar modelos. Como los modelos habitualmente se muestran en forma de diagramas por comodidad, UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos (información que es utilizada o producida mediante un proceso de desarrollo de software) de un sistema de software orientado a objetos, que por su potencialidad se ha convertido en un estándar. Proporciona un vocabulario y una regla para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema.

Los objetivos de UML son muchos pero se pueden resumir en sus funciones(Lambert Claramunt, 2012):

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- Elementos: son abstracciones reales o ficticias (objetos, acciones).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

Valoración de los autores

Se selecciona UML como lenguaje de modelado ya que resuelve de forma satisfactoria el viejo problema del desarrollo de software en su modelado gráfico. Además ha llegado a una solución unificada basada en lo mejor que había hasta el momento, lo cual lo hace más excepcional.

2.3. Herramienta CASE

La Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés) son herramientas que permite la automatización de metodologías paso a paso para el desarrollo de software y sistemas. Su objetivo es reducir los niveles de trabajo repetitivos que los diseñadores necesitan hacer, también facilitan la creación de documentación estructurada y la coordinación de los esfuerzos de desarrollo del equipo de trabajo.

Visual Paradigm8.0

Visual Paradigm, es una herramienta UML profesional fácil de usar. 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 más 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(Lambert Claramunt, 2012).

Dentro de las principales características de la herramienta están:

- Soporte de UML versión 2.0.

- Disponibilidad de integrarse en los principales IDEs.
- Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio.
- Diagramas de flujo de datos.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas.
- Importación y exportación de ficheros.
- Editor de figuras.

Visual Paradigm ofrece un entorno amigable para el usuario, permite crear fácilmente cualquier tipo de diagrama. Incluye gran variedad de estereotipos para la creación de diagramas de fácil entendimiento, los que organiza automáticamente.

Valoración de los autores

Se utiliza la herramienta Visual Paradigm porque es una herramienta multiplataforma. Tiene una interfaz amigable que permite diseñar todos los artefactos que genera, de una forma rápida y con calidad; genera documentación en formatos HTML y PDF, brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas y ofrece la posibilidad de generar código a partir de los diagramas. Permite exportar las imágenes de los diagramas en formato JPG.

2.4. Arquitectura cliente/servidor

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD). Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red. Una representación gráfica de este tipo de arquitectura sería la presentada en la figura 3.

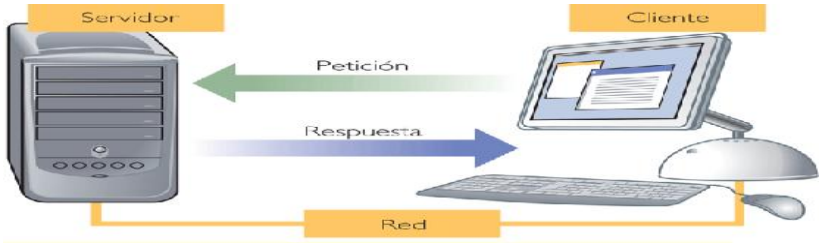


Figura 3: Arquitectura Cliente-Servidor

Este tipo de arquitectura es la más utilizada en la actualidad, debido a que es la más avanzada y la que mejor ha evolucionado en estos últimos años. Se puede decir que esta arquitectura necesita tres tipos de software para su correcto funcionamiento:

1. Software de gestión de datos: este software se encarga de la manipulación y gestión de los datos almacenados y requeridos por las diferentes aplicaciones. Normalmente este software se aloja en el servidor.
2. Software de desarrollo: este tipo de software se aloja en los clientes y sólo en aquellos que se dedique al desarrollo de aplicaciones.
3. Software de interacción con los usuarios: También reside en los clientes y es la aplicación gráfica de usuario para la manipulación de datos, siempre claro a nivel usuario (consultas principalmente) (Alvarez, 2007).

Además de estos, existen más aplicaciones para el correcto funcionamiento de esta arquitectura pero ya están condicionados por el tipo de sistema operativo instalado y el tipo de red en la que se encuentra.

2.5. Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado conocido también como **IDE** (por sus siglas en inglés: Integrated Development Environment), es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consta de en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica o GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones ya existentes. Algunos de estos entornos son compatibles con múltiples lenguajes de programación.

Netbeans 7.3

NetBeans es un IDE desarrollado de código abierto y multiplataforma. Permite diseñar aplicaciones de forma fácil con solo arrastrar objetos a la interfaz de un formulario. Es una plataforma pensada para escribir, compilar, depurar y ejecutar programas. NetBeans no solo permite el desarrollo de aplicaciones de escritorio, también permite el desarrollo de aplicaciones para la web. La programación en este IDE se realiza a través de componentes modulares o módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas ya que estos permiten ser desarrollados independientemente por otros

desarrolladores de software, de ahí que sea una aplicación flexible/extensible(Gonzalez Almora, y otros, 2010)

Valoración de los autores

Se determinó utilizar NetBeans 7.3 ya que es un entorno de desarrollo que tiene una interfaz amigable y fácil de comprender aun cuando los usuarios son inexpertos. Posee herramientas para crear aplicaciones web con PHP 5. Incluye soporte para el trabajo con Symfony y el lenguaje de modelado UML. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix y Linux) como en plataformas Windows.

2.6. Marco de trabajo (Framework)

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, y otros, 2009).

Framework Symfony 2.2

Symfony es un framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Symfony está desarrollado completamente con PHP5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL

Server. Se puede ejecutar tanto en plataformas Unix, Linux, etc. como en plataformas Windows(Milanés López, y otros, 2009).

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.

Valoración de los autores

Se decide utilizar Symfony porque constituye un framework muy maduro, desarrollado completamente con PHP5, que permite migrar de Sistema Gestor de Base de Datos sin hacer cambios en el código fuente de la aplicación, ya que es compatible con la mayoría de los gestores de bases de datos existentes. Además, se puede ejecutar tanto en plataformas Unix como en plataforma Windows. Presenta una maravillosa documentación,sobre todo en español. Por otra parte es el que se utiliza en los proyectos del centro GEySED.

ExtJs

ExtJs es un API escrito en Java Script con la finalidad de asistir el desarrollo de aplicaciones enriquecidas para internet. Entre las principales características de ExtJs resalta un fuerte paradigma basado en componentes soportado por recursos para la programación orientada a objetos en Java Script que facilitan la implementación de extensiones y aplicaciones de gran complejidad, el poseer un amplio conjunto de componentes configurables de alta calidad y una implementación transparente y sencilla para el trabajo con AJAX(Lobo, 2006).

Para el desarrollo del presente trabajo se usará ExtJs v4.1, versión estable de dicho producto. Esta versión está bajo la licencia de la GPL. El uso de la misma permitió tener estos beneficios:

- Un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

Doctrine ORM[3]

Doctrine es un ORM para PHP 5 y posterior. Además de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje DQL (Doctrine QueryLanguage) inspirado en el HQL de Hibernate. (Potencier, Fabien y Zaninotto, François. Symfony la guía definitiva. 2008) Entre muchas otras cosas tienes la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM a tablas de una base de datos(Pérez Mata, 2009).

Con el uso de doctrine el control de datos podrá ser efectuado de manera fácil, permitiendo trasformar el uso de consultas a base datos a un lenguaje sencillo. El subsistema de Administración al usar Doctrine permite manejar consultas a un nivel muy sencillo.

2.7. Lenguajes de programación

Un lenguaje de programación no es más que un idioma artificial diseñado para expresar procesos que pueden ser llevados a cabo por computadoras, que pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación. Está constituido por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Lenguaje de programación del lado del cliente

Los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador. Cada uno de estos tipos tiene por supuesto sus ventajas y sus inconvenientes. Así, por ejemplo, un lenguaje de lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas.

[3] Un ORM o (ObjectRelationMapper) es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y los registros objetos se pueden manejar con facilidad.

JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java, que si es orientado a objeto (OO). Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems(Pérez Mata, 2009).

Lenguaje de programación del lado del servidor

Los lenguajes del lado del servidor son aquellos que se ejecutan en el servidor web, justo antes de que se envíe la página al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos y conexiones en red para crear la página final que verá el cliente.

PHP5

Es un lenguaje de programación que sirve fundamentalmente para páginas Web, ejecutándose el lado del servidor, es independiente del navegador web (browser), pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar este lenguaje. Este cuenta con variables, sentencias, condicionales, ciclos (bucles) y funciones, siendo así un lenguaje robusto y confiable para el desarrollo de aplicaciones web; está cercano a JavaScript, C o a Perl.

Es capaz de combinarse con servidores de bases de datos tales como MySQL, PostgreMSQL, Oracle, ODBC, DB2, Microsoft SQLServer, Firebird y SQLite. Es multiplataforma, con capacidad de ejecutarse en varios sistemas operativos como: UNIX, LINUX, Mac OS y Windows; además permite la interacción con servidores web de mayor uso que existen como: CGI, servidor Apache e ISAPI. Pertenece a la alternativa de código abierto (Open Source), por lo que se presenta como una elección de fácil acceso para todos(Ávila Guerrero, y otros, 2009).

Valoración de los autores

Se determinó utilizar JavaScript como lenguaje de programación del lado del cliente ya que es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez además de que es uno de los más utilizados a nivel mundial. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica. Es compatible con la mayoría de los navegadores modernos. Presenta abundante documentación. Es interpretado por el cliente y no cargado en el servidor antes de ejecutarse en el cliente.
Se decidió utilizar PHP5 como lenguaje de programación del lado del servidor ya que es OpenSource, por lo que no está forzado a pagar actualizaciones anuales para tener una versión que funcione. Dispone de una amplia gama de bibliotecas y agregarle extensiones es muy fácil. Los niveles de seguridad pueden configurarse desde el archivo .ini, aunque esto no depende del lenguaje sino de la manera en que se programe. Además, interactúa muy bien con el gestor de bases de datos PostgrSQL y con el framework Symfony. Al ser utilizado como un módulo de Apache hace que sea extremadamente veloz y que se utilice poca memoria.

2.8. Sistema Gestor de Base de Datos (SGDB)

Un Sistema Gestor de Base de Datos (SGBD, en inglés DBMS: Data Base Management System) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación(BERTINO, E. A., 1995)

PostgreSQL8.4

PostgreSQL8.4proporciona estabilidad, potencia, robustez, facilidad de administración e implementación de estándars lo que lo convierte en el sistema de gestión de bases de datos de código abierto más potente del mercado, funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema, lo cual es una de las características que más se explotan en sistemas de replicación y alta disponibilidad. Con PostgreSQL no se violan los acuerdos de licencia, puesto que no hay costo asociado a la licencia del software. (Silva Cobas, 2010).

Valoración de los autores

Primeramente hay que decir que PostgrSQL pertenece a la familia de software libre que tiene integridad referencial e Interfaces nativas para PHP y otros lenguajes de programación; así como soporte nativo SSL. Además, permite la copia de seguridad continua del servidor; está diseñado con una arquitectura

cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema, por lo que un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

2.9. Servidor web (Apache 2.0)

El servidor HTTP Apache es un software libre, de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual, lo que lo hace prácticamente universal. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor, se puede ver, sin ningún secreto, sin ninguna puerta trasera. Apache es un servidor altamente configurable de diseño modular(León Piñero, y otros, 2009).

Ventajas del servidor HTTP Apache 2.0(Apache, 2012)

1. Nueva interfaz de programación (API) de Apache: La API para los módulos ha cambiado significativamente en la nueva versión. Muchos de los problemas de ordenamiento y prioridad de módulos de la versión 1.3 han desaparecido. Apache 2.0 hace automáticamente mucho de lo que es necesario, y el ordenamiento de módulos se hace ahora por hooks, lo que ofrece una mayor flexibilidad. También se han añadido nuevas llamadas que ofrecen capacidades adicionales sin tener que parchear el núcleo del servidor Apache.
2. Mensajes de error en diferentes idiomas: Los mensajes de error que se envían a los navegadores están ahora disponibles en diferentes idiomas, usando documentos SSI.
3. Actualización de la librería de expresiones regulares: Apache 2.0 incluye la librería de expresiones regulares compatibles con Perl (PCRE).

Valoración de los autores

Por todo lo antes planteado se decide utilizar Apache y además porque trabaja con gran cantidad PHP y otros lenguajes de script. También trabaja con Java y páginas jsp.Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Tiene un alto nivel de configuración en la creación y gestión de registros. Permite la creación de ficheros de registros a medida del administrador, de esta manera se puede tener un mayor control sobre lo que sucede en el servidor y personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Además, es un software libre, de código abierto.

Conclusiones del capítulo

Después de haber analizado cada una de las herramientas y tecnologías estudiadas, se obtuvo una guía que rigió la documentación del proceso de desarrollo de software definida por la metodología de software RUP, además generar los diagramas definidos por dicha metodología empleando el lenguaje de modelado UML y haciendo uso de la herramienta CASE Visual Paradigm. La selección del IDE NetBeans posibilitó viabilizar el desarrollo en la fase de implementación.

CAPITULO 3: DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se muestra el Modelo de Dominio. Se enumeran los requisitos funcionales y los no funcionales; se identifican y describen los casos de uso que guiarán la solución propuesta. Se define, además, cómo se va a desarrollar la aplicación a través de los artefactos de análisis y diseño.

3.1. Modelo de dominio

Un modelo del dominio es una representación de las clases conceptuales significativas para un mayor dominio del problema del mundo real.

La gestión de riesgos define los procesos que se manipulan en el negocio, pero al no tener un cliente definido, no se puede establecer las características que se desarrollan en el entorno de los proyectos productivos en el centro, para la manipulación y control de los riesgos asociados a cada uno de estos. Para la definición del flujo de eventos que se realiza, iniciado por la detección de un riesgo en cada proyecto del centro GEySED, hasta la elaboración de planes de contingencia y mitigación asociados a este, 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.

Debido a que es necesario tener un gran conocimiento del funcionamiento del sistema, con el objetivo de realizar una correcta captura de los requisitos a continuación se exponen todos los conceptos que intervienen en el mismo:

Riesgo: la probabilidad de que una amenaza se materialice, utilizando vulnerabilidad existente de un activo, o un grupo de activos, generándole pérdidas o daños.

Proyecto: es el entorno en el que se realizará la gestión de riesgo.

Registro de GR: documento en el cual se registran todas las características de cada uno de los riesgos.

Plan de Contingencia: es donde se diseña la estrategia, se recogen todas las medidas organizativas, técnicas y se exponen los procedimientos para enfrentarse a la eventualidad de un riesgo que ponga en peligro la continuidad del proyecto.

Plan de Mitigación: conjunto de acciones encaminadas a reducir la ocurrencia de un riesgo.

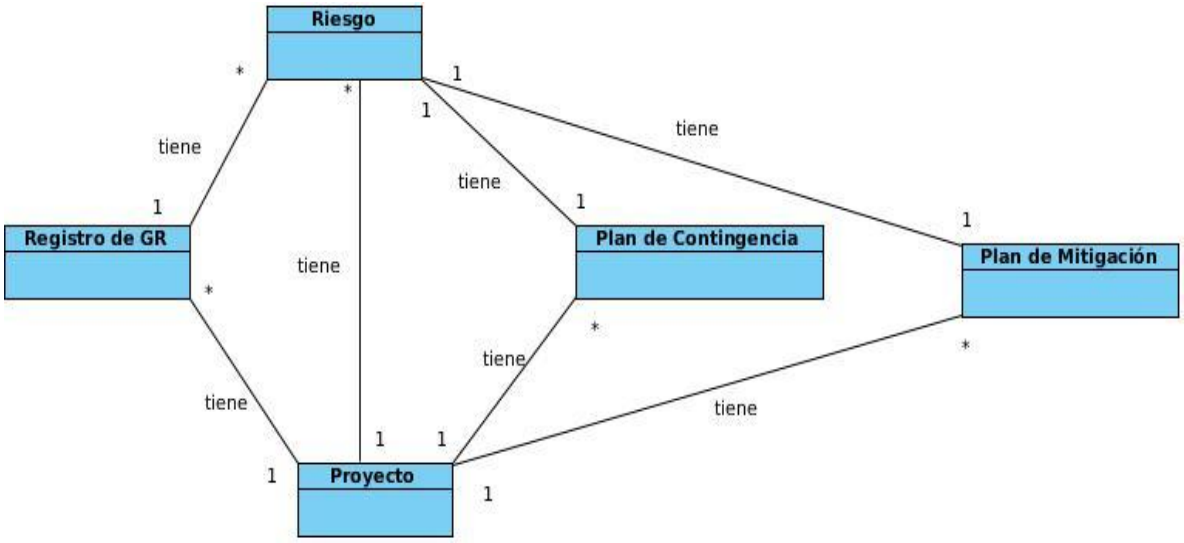


Figura 4: Modelo de dominio

Descripción del modelo de dominio

En un **Proyecto** productivo pueden surgir cambios inesperados que afecten los objetivos del mismo, a estos cambios se le denomina **Riesgo.** Cada proyecto tiene un **Registro de GR** donde se muestran los riesgos asociados a dicho proyecto, además de un **Plan de Contingencia** y un **Plan de Mitigación** para darle respuesta a cada uno de estos riesgos. Cada riesgo a su vez tendrá su propio plan de contingencia y de mitigación.

3.2. Requisitos del sistema

Luego de conocer los conceptos asociados al objeto de estudio del problema se comienza a modelar el sistema que se va construir, además se analiza, ¿qué debe hacer el sistema para que se cumplan los objetivos planteados? Para lo mismo se identifican los requisitos funcionales (RF), los no funcionales RNF) y se modelan los RF en representaciones de casos de uso del sistema.

3.2.1 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, 1988)

RF1 Autenticar usuario.: El sistema debe permitir que los usuarios se autentiquen para poder realizar las funcionalidades de la herramienta.

RF2 Insertar usuario: El sistema debe permitir insertar un usuario.

RF3 Modificar usuario: El sistema debe permitir modificar un usuario.

RF4 Eliminar usuario: El sistema debe permitir eliminar un usuario.

RF5 Insertar proyecto: El sistema debe permitir insertar un proyecto.

RF6 Modificar proyecto: El sistema debe permitir modificar un proyecto.

RF7 Cerrar proyecto: El cerrar debe permitir eliminar un proyecto.

RF8 Insertar riesgo: El sistema debe permitir insertar un riesgo

RF9 Modificar riesgo: El sistema debe permitir modificar un riesgo

RF10 Eliminar riesgo: El sistema debe permitir eliminar un riesgo.

RF11 Mostrar listado de los riesgos: El sistema debe permitir mostrar un listado con los riesgos asociados a cada proyecto.

FR12 Mostrar listado de riesgos que requieran respuestas a corto plazo: El sistema debe permitir mostrar un listado con los riesgos más importantes del proyecto ordenados de mayor a menor de acuerdo a la exposición al riesgo, los riesgos más importantes serán los diez primeros y estarán de color rojo.

RF13 Determinar exposición del riesgo: El sistema debe permitir dado el impacto y la probabilidad calcular la exposición al riesgo.

RF14 Insertar plan de mitigación: El sistema debe permitir insertar un plan de mitigación

RF15 Modificar plan de mitigación: El sistema debe permitir modificar un plan de mitigación

RF16 Eliminar plan de mitigación: El sistema debe permitir eliminar un plan de mitigación.

RF17 Insertar plan de contingencia: El sistema debe permitir insertar un plan de contingencia.

RF18 Modificar plan de contingencia: El sistema debe permitir modificar un plan de contingencia**.**

RF19 Eliminar plan de contingencia: El sistema debe permitir eliminar un plan de contingencia.

RF20 Insertar acciones de mitigación: El sistema debe permitir insertar acciones de mitigación.

RF21 Modificar acciones de mitigación: El sistema debe permitir modificar las acciones de mitigación

RF22 Eliminar acciones de mitigación: El sistema debe permitir eliminar las acciones de mitigación

RF23 Insertar acciones de contingencia: El sistema debe permitir insertar acciones de contingencia.

RF24 Modificar acciones de contingencia: El sistema debe permitir modificar las acciones de contingencia**.**

RF25 Eliminar acciones de contingencia: El sistema debe permitir eliminar las acciones de contingencia**.**

RF26: Generar reportes: El sistema debe permitir generar un reporte con todos los riesgos identificados en el proyecto.

RF27 Mostrar grafica de análisis de riesgos: El sistema debe permitir mostrar una gráfica donde se muestre el análisis de los riesgos de acuerdo a la exposición al riesgo (impacto* probabilidad).

RF28 Buscar riesgo: El sistema debe permitir que dado cualquier dato que coincida con alguno de los datos almacenados en la BD el usuario pueda buscar cualquier riesgo o proyecto.
RF29 Determinar la situación de los riesgos en el proyecto: El sistema debe permitir que el usuario pueda ver la situación actual de los riesgos en el proyecto.
RF30 Determinar la idoneidad del plan de contingencia: El sistema debe permitir que el usuario verifique si el plan de contingencia desarrollado fue o no efectivo.

Los RF quedaron agrupados en los siguientes casos de usos.
CU1 Autenticar usuario: RF1.
CU2 Gestionar usuario: RF2, RF3., RF4.
CU3 Gestionar proyectos: RF5, RF6. RF7.
CU4 Gestionar riesgos: RF8, RF9, RF10.
CU5 Mostrar listado de los riesgos: RF11.
CU6 Mostrar listado de riesgos que requieran respuestas a corto plazo: RF12.
CU7 Determinar exposición del riesgo: RF13
CU8 Gestionar plan de mitigación: RF14, RF15, RF16.
CU9 Gestionar plan de contingencia: RF17, RF18, RF19.
CU10 Gestionar acciones de mitigación: RF20, RF21, RF22.
CU11 Gestionar acciones de contingencia: RF23, RF24, RF25.
CU12 Generar reportes: RF26.
CU13 Mostrar grafica de análisis de riesgos: RF27.
CU14 Buscar riesgo: RF28.
CU15 Determinar la situación de los riesgos en el proyecto: RF29.
CU16 Determinar la idoneidad del plan de contingencia: RF30.

3.2.2 Requisitos no Funcionales

Los RNF especifican propiedades del sistema que el sistema debe cumplir, como restricciones 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 usuarios(Pérez Quintero, y otros, 2008).

✓ **Apariencia o interfaz externa**

-Diseño sencillo y fácil de usar, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.

-Debido al gran uso que deberá tener la aplicación, es necesario que se combinen correctamente los colores, tamaño y tipo de letra.

✓ **Usabilidad**

-Tipo de Aplicación Informática: aplicación WEB.

-Finalidad: Esta aplicación web tiene como objetivo automatizar el proceso de gestión de riesgo en cada uno de los proyectos que tiene el centro de GEySED.

-La aplicación podrá ser manipulada por toda persona con conocimientos básicos de ambientes Web

✓ **Portabilidad**

-La herramienta debe ser capaz de ejecutarse sobre el sistema operativo Linux.

✓ **Diseño e implementación**

-La aplicación será desarrollada por el framework de desarrollo Symfony2.1.8usando PHP 5.3 como lenguaje del lado del servidor y Java Script del lado del cliente.
-El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte.
-El estilo arquitectónico es el MVC para la separación de los datos, interfaz de usuario y lógica de negocio.

✓ **Seguridad**

Confidencialidad

-Garantizar que la información sea vista únicamente por los usuarios autenticados en el sistema.

✓ **Hardware**

Cliente

-Se recomienda un navegador Mozilla Firefox 3.5.5, pero se puede hacer uso de otro navegador que cumpla con los estándares W3C (World Wide Web Consortium).
-Computadora con al menos 256 MB de memoria RAM, al menos 100 MB de disco duro.

Servidor

-Se requiere un servidor que tenga como mínimo 512 MB de memoria RAM, 500 MB de espacio libre.

✓ **Software**

-El lenguaje de programación es PHP, versión 5.2.
-El servidor Web es Apache Server 2.0, compatible con GPL.
-El servidor de Base de Datos es PostgreSQL8.0.
-El framework de desarrollo es Symfony en su versión 2.1.8.

3.3 Propuesta de solución

Haciendo uso de las habilidades y facilidades que brinda UML y luego de realizar la captura de los RF del sistema, se representará el diagrama de casos de uso del sistema. Teniendo en cuenta lo definido anteriormente, se identifican cuáles serán los actores que van a interactuar con el sistema y los casos de uso que van a representar las diferentes funcionalidades.

3.3.1 Actores del sistema

Los actores del sistema definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que interactúan con el mismo intercambiando información. A continuación se detallan los actores del sistema. (Ver tabla 3)

Actor	Descripción
Administrador	Es la persona que se encarga de administrar todo el sistema informático, y de gestionar los usuarios del sistema.
Usuario	Es la persona que pretende ingresar al sistema. Si el proceso de autenticación es correcto, recibe permisos en dependencia del rol que desempeña. El ingreso al sistema es mediante el usuario y contraseña. Se beneficia de cierta forma con la información que está almacenada en el sistema, puede solicitar reportes y exportar estos reportes
Equipo de GR	El equipo del proyecto es el responsable de realizar todas

las tareas para controlar y darle seguimiento a los riesgos, gestionar y clasificar correctamente los mismos, además de gestionar los proyectos y de tener acceso pleno a la información que se maneja en el sistema.

Tabla 3: Descripción de los actores del sistema

3.3.2 Diagrama de casos de uso del sistema

Los casos de uso del sistema (CUS) son un conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor. Con otras palabras, son fragmentos de funcionalidad que el sistema ofrece a los actores que interactúan con el mismo, reportándoles beneficios.

Luego de haber definido los RF del sistema de acuerdo al flujo de trabajo de requerimientos de la metodología seleccionada a continuación se muestra el diagrama de casos de uso que responde a dichos requisitos.

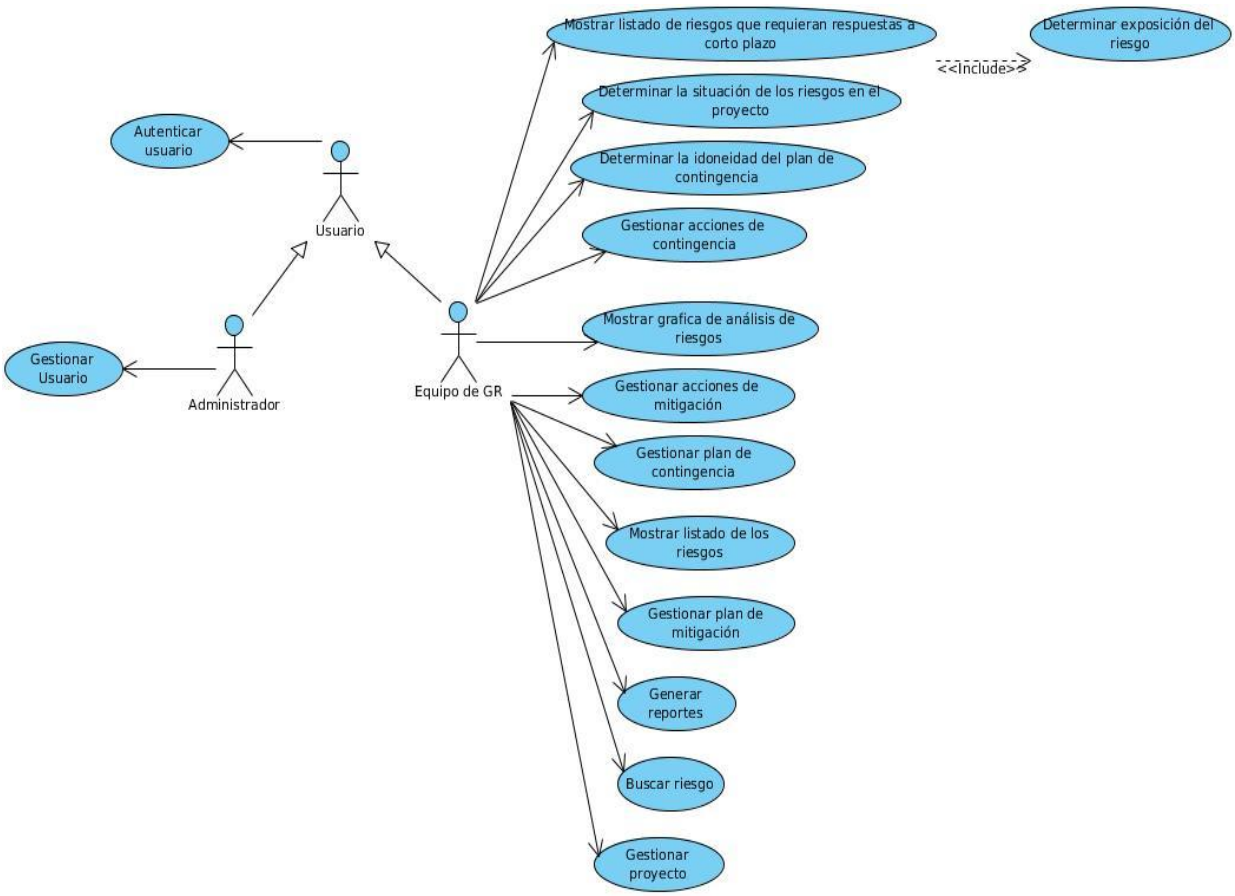


Figura 5: Diagrama de CU

En la Figura 5 se muestra los actores que interactúan con el sistema y cada una de las funcionalidades que estos realizan.

Un usuario puede ser un administrador, un equipo de GR y este a su vez puede ser líder del proyecto. Todos los usuarios para poder ingresar en el sistema deben estar debidamente autenticados. El administrador se encargará de gestionar los proyectos del centro GEySED. El equipo de GR podrá gestionar planes de mitigación y de contingencia, además de gestionar el conjunto de acciones que se llevarán a cabo en dichos planes. Podrá mostrar un listado con los riegos críticos del proyecto, una gráfica con el análisis de dichos riegos, además de verificar la idoneidad con que se realizó el plan de contingencia y la situación actual de los riesgos en el proyecto. El líder del proyecto se encargará de identificar los riesgos asociados al proyecto, además puede realizar las acciones que el equipo de gestión de riesgo realiza.

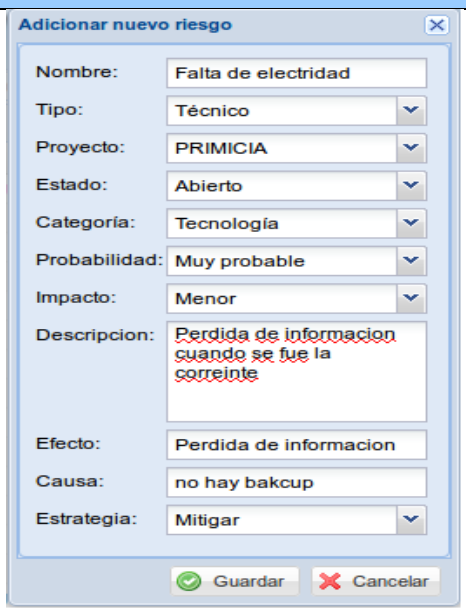
3.3.3 Descripción de los casos de uso del sistema

Para un mayor entendimiento de los CUS, es necesario hacer una descripción de cada uno de los procesos que se realizan. Esta descripción contiene el conjunto de pasos que se deben cumplir para llevar cada CU a su fin.

CU4: Gestionar riesgo

Objetivo	Gestionar los riesgos del proyecto.
Actores	Líder del proyecto
Resumen	El CUS se inicia cuando el líder de proyecto selecciona la opción Listado de riesgos, luego selecciona el tipo de gestión que desea realizar si es adicionar un nuevo riesgo, modificarlo, o eliminar uno ya existente, introduce los datos en caso que sea necesario, el sistema realiza la acción seleccionada y termina el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El usuario con el rol de líder de proyecto se debe haber autenticado con anterioridad.

Postcondiciones	• Riesgo adicionado a la Base de Datos. • Riesgo modificado en la Base de Datos. • Riesgo eliminado de la Base de Datos.
Objetivo	Gestionar los datos de los riesgos del proyecto.
Flujo de eventos	
Flujo básico <Gestionar Riesgos>	
Actor	**Actor**
1-El líder de proyecto selecciona la opción Riesgos.	1.1-El sistema muestra un listado con los riesgos identificados y permite al líder del proyecto realizar las siguientes opciones: • Insertar nuevo riesgo • Modificar riesgo • Eliminar riesgo
Sección 1: "Insertar nuevo riesgo"	
1-El líder de proyecto selecciona la opción "Adicionar" de la sección Listado de riesgos.	1.1-El sistema muestra un formulario con los campos necesarios para que el líder de proyecto registre los datos del riesgo. • Nombre • Tipo • Proyecto • Descripción • Efecto • Impacto • Probabilidad • Causa • Categoría • Estrategia
2-El líder de proyecto inserta los datos en el formulario.	2.1-El sistema valida que todos los campos estén llenos. 2.2-El sistema verifica que el riesgo no esté registrado

	en la Base de Datos. 2.3-El sistema guarda todos los datos entrados por el líder de proyecto y termina el caso de uso.
Prototipo de interfaz de usuario	
	
Flujos Alternos	
Actor	**Actor**
	2.1-El sistema muestra un mensaje “Este campo es obligatorio”, y retorna a la acción 2.
Sección 3: “Modificar riesgo”	
Actor	**Actor**

1-El líder de proyecto selecciona el riesgo que va a modificar y selecciona la opción “Editar”.	1.1-El sistema localiza los datos del riesgo y los muestra en un formulario. • Nombre • Tipo • Proyecto • Descripción • Efecto • Impacto • Probabilidad • Causa • Categoría • Estrategia
2-El líder de proyecto realiza las modificaciones deseadas	2.1-Se verifica que todos los campos estén llenos. 2.2 El sistema verifica que todos los datos modificados sean correctos. 2.3-El sistema actualiza la información entrada por el líder de proyecto en la base de dato y finaliza el caso de uso.
Flujos Alternos	
Actor	**Sistema**
	2.1-En caso de que el líder del proyecto deje campos vacíos, el sistema muestra un mensaje: “Este campo es obligatorio” y retorna a la acción 2. 2.2 Si los datos que se desean modificar no se entran correctamente el sistema muestra un mensaje de error, da la posibilidad de insertar los datos nuevamente y retoma al flujo normal de eventos a partir de la acción 2.
Sección 3: “Eliminar riesgo”	

Actor	**Sistema**
1-El líder de proyecto selecciona el riesgo que desea eliminar y da clic en la opción “Eliminar”.	2-El sistema elimina el riesgo seleccionado de la base de dato y termina el CU.
Flujos Alternos	
Actor	**Sistema**
Requisitos Funcionales	RF8, RF9, RF10

Tabla 4: Descripción del caso de uso Gestionar riesgo

3.4 Arquitectura de software

3.4.1 Estilos arquitectónicos

Un Estilo Arquitectónico es una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes(Camacho, y otros, 2004).

Para el desarrollo de la herramienta web se hace uso de la familia de estilos de llamada y retorno

Llamada y Retorno

Reflejan la estructura del lenguaje de programación. Permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Se basan en la bien conocida abstracción de procedimientos/funciones/métodos. Son utilizados en grandes sistemas de software y persiguen escalabilidad y modificabilidad (Nava, 2009).

3.4.2 Patrones arquitectónicos

Los patrones arquitectónicos ofrecen soluciones bien establecidas a problemas arquitecturales dentro de la Ingeniería del software; describen los elementos y las relaciones junto a especificaciones de cómo pueden ser usados. La Arquitectura del Software comprende los primeros pasos de las decisiones del

diseño para un sistema, por lo tanto, se acude a la separación de capas para contrarrestar las consecuencias de las modificaciones futuras(Martínez Vergara, 2010).

Modelo Vista Controlador

El modelo vista controlador es un patrón de arquitectura de software que se ve comúnmente en aplicaciones web donde la vista es la página HTML y el código que suministra de datos dinámicos a la página, el modelo es el sistema de gestión de base de datos y el controlador representa la lógica del negocio(Martínez Vergara, 2010).

Aunque se pueden encontrar diferentes implementaciones de MVC, los flujos más comunes son generalmente los siguientes(Barnett Villalobos, 2010):

1. El Modelo Pasivo
2. El Modelo Activo

El modelo pasivo es empleado cuando un Controlador manipula el Modelo exclusivamente. El Controlador modifica al Modelo y luego notifica a la Vista que dicho Modelo ha sido cambiado.

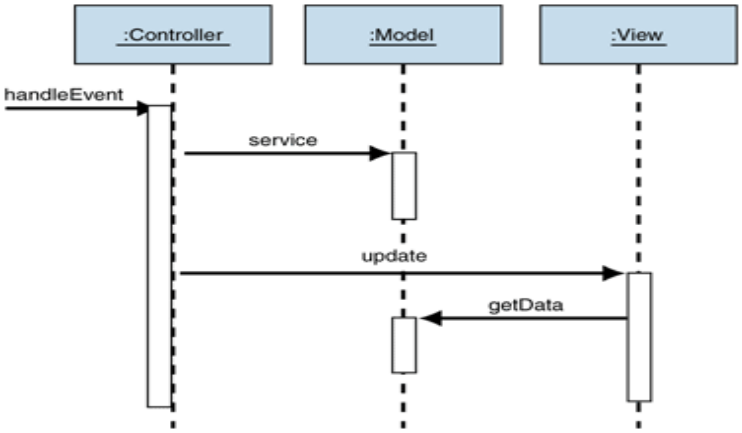


Figura 6: Modelo pasivo

El modelo activo es usado cuando el Modelo cambia de estado sin la intervención del Controlador.

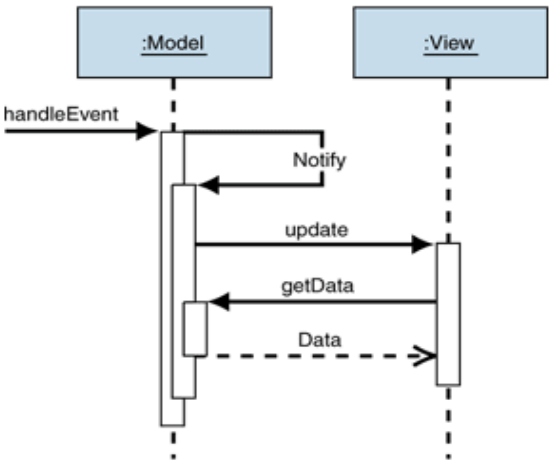


Figura 7: Modelo activo

La herramienta web propuesta presenta como parte de la línea base de su arquitectura el patrón Modelo Vista Controlador, el cual separa los datos de la aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos: el modelo para la administración de los datos; la vista que muestra las interfaces de usuario y el controlador para la gestión de las acciones del usuario, invocando cambios tanto en la vista como en el modelo. Con el uso de este patrón se garantiza la no dependencia entre la vista y el modelo, por lo que un cambio en la vista trae consigo poco o ningún impacto en los datos o en la lógica de negocio.

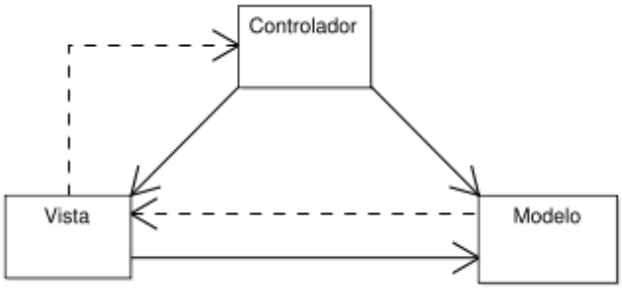


Figura 8: Patrón arquitectónico MVC

3.5 Modelo de análisis

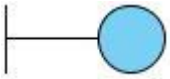
El modelo de análisis, es la primera representación técnica de un sistema y establece una base para la creación del diseño del software. Estructura los requisitos de un modo que facilita su comprensión, preparación, modificación y en general su mantenimiento en un formato que se corresponda con el área de conocimiento de los diseñadores del software. Este constituye una abstracción del modelo de del

diseño y está conformado por las clases interfaz, clases controladoras y clases entidad(Jacobson, y otros, 2000).

3.5.1 Diagrama de Clases del Análisis

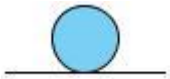
El análisis consiste en transformar los requisitos funcionales en un diseño de clases en el cual se vean las relaciones e interacción que existe entre ellas, teniendo en cuenta en este proceso una arquitectura robusta que permita adaptar el sistema al entorno de implementación que se está desarrollando. Además en este flujo se obtiene una visión del sistema que se preocupa de ver que hace, de tal forma que se preocupa solo por los requisitos funcionales*(*PérezNúñez*,* 2007*)*.

Las clases del análisis siempre encajan con tres estereotipos básicos.



Las clases interfaz se utilizan para modelar la interacción entre el sistema y sus actores.

Las clases controladoras (control) representan coordinaciones, secuencias, transacciones, y control de objetos y se utilizan con frecuencia para encapsular el control de un caso de uso en concreto.



Las clases entidades se utilizan para modelar información que posee una vida larga y que es a menudo persistente. La mayoría de estas clases se derivan de una entidad del negocio.

A continuación se presenta el diagrama de clases del análisis del caso de uso más significativo de la herramienta web para gestión de riesgos en los proyectos del centro GEySED (Figura 9), para ver el resto dirigirse al Anexo 1.

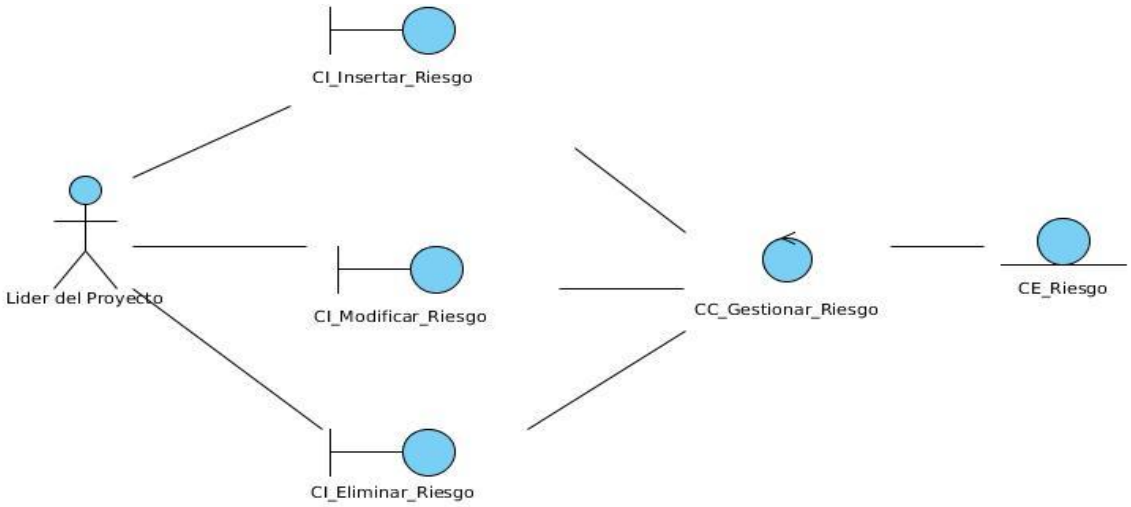


Figura 9: Diagrama de clase del análisis (Gestionar riesgos)

3.6 Modelo de diseño

Es un modelo de objetos que describe como los casos de uso influyen en el sistema. Es también una abstracción de la implementación. Cada subsistema o clase del diseño representa una abstracción con una correspondencia con la implementación. Los casos de uso se realizan por clases de diseño, lo cual se representa por colaboraciones en el modelo del diseño(Lobaina Rodríguez, y otros).

3.6.1 Diagrama de clases del Diseño

En el diagrama de clases del diseño se especifica la estructura de las clases del sistema, incluyendo las relaciones entre clases(Balmaseda Acosta, y otros, 2007).

A continuación se presenta el diagrama de clase del diseño del caso de uso Gestionar riesgo (Figura 10).

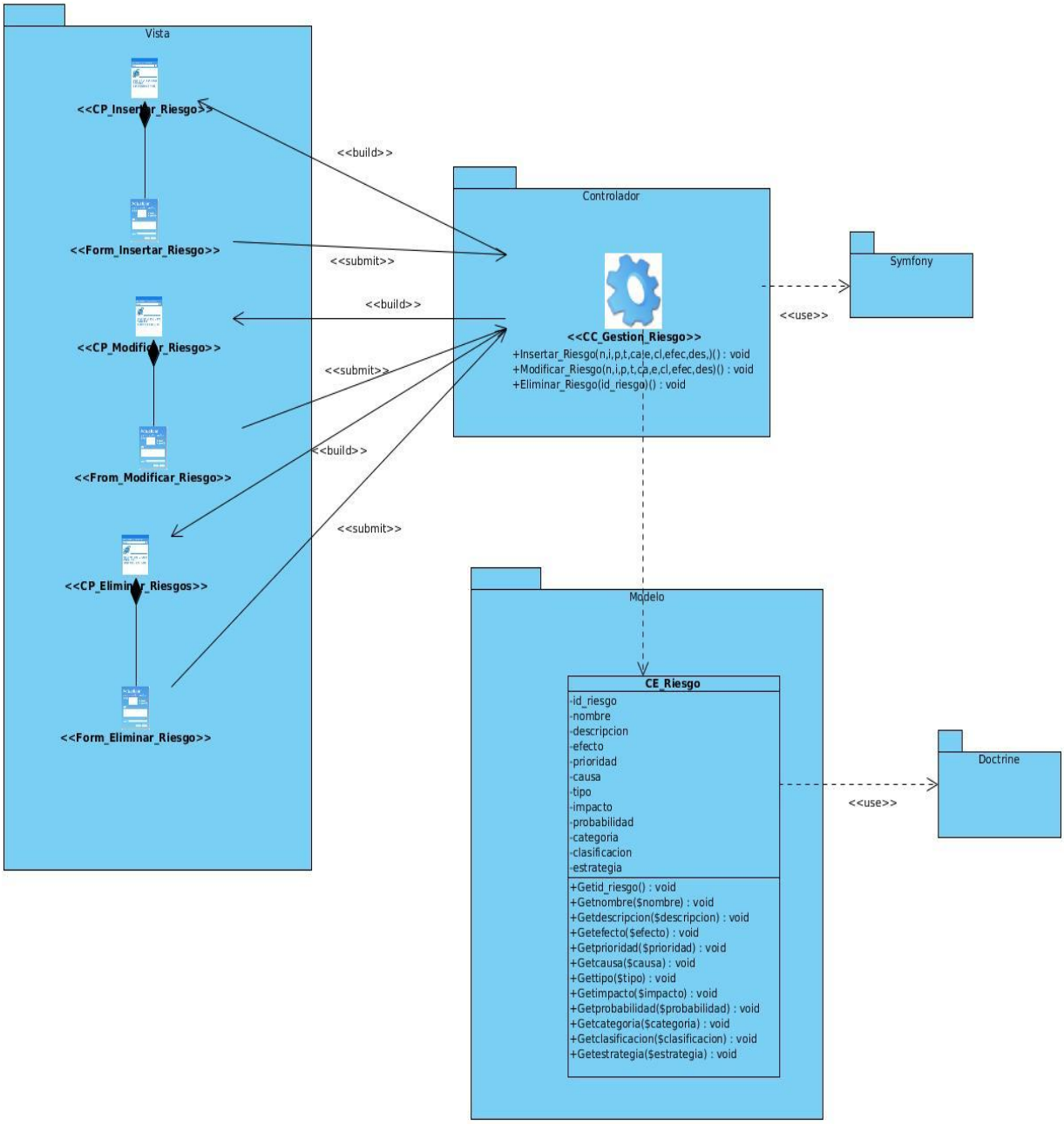


Figura 10: Diagrama de diseño del CU Gestionar riesgo

3.7 Patrones de diseño

Los patrones de diseño son definidos como una solución reusable para un problema que comúnmente ocurre durante el diseño del software. Una solución para que sea considerada como patrón debe cumplir

con determinadas características como son su efectividad resolviendo problemas, reusabilidad(Martínez Vergara, 2010).

3.7.1 Patrones GRASP

En los componentes definidos fueron aplicados los patrones GRASP (General Responsibilit y Assignment Software Patterns), Se buscó que cada clase incluida fuese responsable de los aspectos que le conciernen.

Patrón Experto: Plantea que se debe asignar determinada responsabilidad al experto en información, de otra forma, a la clase que tiene la información necesaria para realizar la responsabilidad. Es uno de los más usados en Symfony ya que garantiza que se creen las clases con las funcionalidades asociadas a las entidades de la base de datos. En la solución propuesta este patrón se utiliza en las clases entidades ya que estas son las encargadas de acceder únicamente a los datos del sistema.

Patrón Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. Por lo tanto el patrón creador propone que una clase cree instancias de otra dadas determinadas condiciones. En el sistema este patrón es aplicado por las clases controladoras, debido a que son las encargadas de controlar el sistema creando objetos de otras clases para tener acceso a estas.

Controlador: El patrón de diseño controlador está diseñado para asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Este patrón se evidencia en las clases controladoras, pues estas tienen la responsabilidad de recibir o manejar un evento del sistema. En la aplicación se dividen los eventos del sistema en varias clases controladoras para lograr disminuir el acoplamiento y aumentar la cohesión.

Alta Cohesión: Su principio se basa en que cada clase debe tener responsabilidades moderadas en un área funcional y colaborar con otras para llevar a cabo las tareas(Molina Luis, 2007). Por tanto la aplicación de este patrón posibilita, que las clases sean más fáciles de comprender, de reutilizar y conservar. Se tuvo en cuenta este patrón porque cada clase cuenta con las funcionalidades imprescindibles para su buen funcionamiento. Este patrón permite fomentar la reutilización y genera bajo acoplamiento.

Bajo Acoplamiento: Plantea que las dependencias entre las clases deben ser en el menor grado posible(Molina Luis, 2007). En el diseño propuesto existen pocas dependencias entre las clases, esto permite que en caso de producirse una modificación en una de ellas, se tenga el mínimo desenlace

posible en el resto de clases por lo que se ha logrado que el acoplamiento sea bajo y las posibilidades de poder reutilizar el diseño aumenten.

3.7.2 Patrones GOF (Gang of Four)

Singleton: Singleton o solitario, este patrón es utilizado para garantizar que una clase sólo tenga una única instancia, proporcionando un punto de acceso global a la misma. En el diseño propuesto la clase sfRouting se encuentran en la capa Controlador del framework, la misma es muy usada porque es la encargada de enrutar todas las peticiones que se hagan a la aplicación.

Decorador: Responde a la necesidad de añadir funcionalidades dinámicamente a un objeto, lo que permite no crear sucesivas clases que hereden de la primera incluyendo la funcionalidad, sino otras que la implementan y se asocian a la primera. Symfony implementa este patrón ya que posee un archivo layout.php (plantilla global) el cual almacena el código HTML común a todas las páginas de la aplicación, evitando así las repeticiones de código innecesarias. De esta manera el layout decora las plantillas de forma dinámica(Potencier, y otros, 2008).

3.8 Modelo de Datos

El Modelo de datos describe de forma abstracta las entidades y sus características, además de las relaciones entre estas dentro de la base de datos.

Las entidades son objetos que guardan información necesaria para el sistema. Su símbolo es un rectángulo. Los atributos son características de una entidad, se representan colocando su nombre dentro del rectángulo de la entidad. Los atributos se clasifican en: obligatorios, opcionales, claves foráneas y claves primarias (estas se dividen en simples y compuestas). Gráficamente la clave primaria se representa con un signo de suma y la foránea con un símbolo de número. Las relaciones muestran la asociación entre dos entidades, representadas por una línea que une a las entidades involucradas(Fuentes Escoba, y otros, 2010).

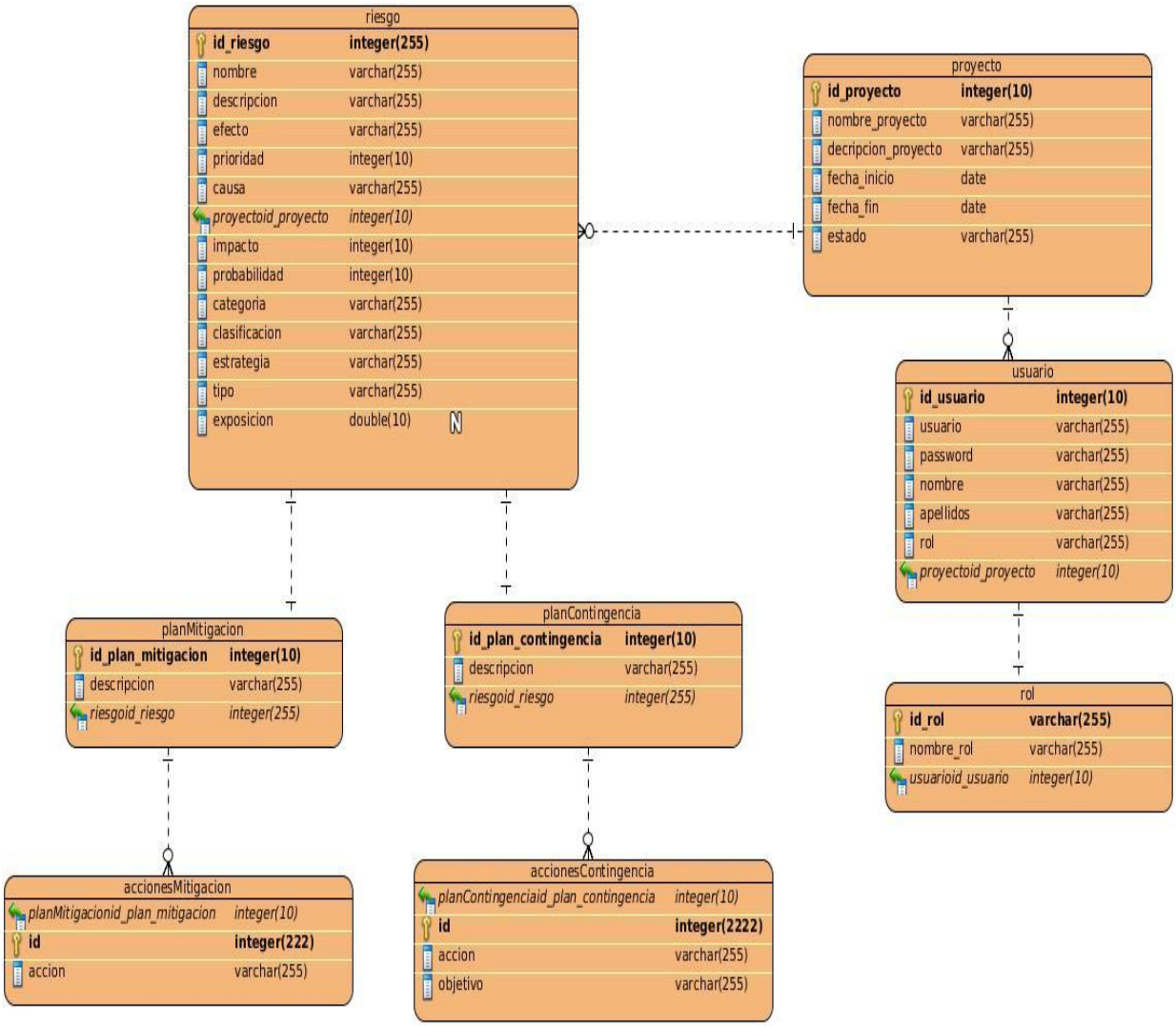


Figura 11: Modelo de Datos

Descripción del modelo.

En el modelo de datos que se muestra en la figura 11, se representa la relación entre las clases persistentes**.** Entre la tabla proyecto y riesgo existe una relación de "uno a mucho" por lo que su llave primaria pasa hacer llave foránea de riesgo, está a su vez se relaciona con la tabla planMitigación con una relación de "uno a uno", lo misma pasa con la tabla planContingencia. La tabla planMitigación tiene una relación de uno mucho con la tabla accionesMitigación igual que la tabla planContingencia con accionesMitigacion. La tabla proyecto tiene otra relación de "uno a mucho" con la tabla usuario por lo que

la llave primaria de proyecto pasa como llave foránea para la tabla usuario y esta tiene una relación con la tabla rol de “uno a uno”.

Conclusiones del capítulo

En el desarrollo de este capítulo se analizaron los principales conceptos que se manejan en el dominio del problema lo que permitió la identificación de las funcionalidades del sistema y las capacidades que este debe tener. Estas funcionalidades fueron encapsuladas en los casos de uso para definir el flujo de eventos presentes en la GR. La selección de patrones arquitectónicos y de diseño permitió obtener un modelado del diseño basado en soluciones ya probadas, lo que define una representación refinada de las clases para cada flujo de eventos. La representación de estos diagramas se utilizó como entradas en la disciplina de implementación lo que hizo posible un desarrollo consistente del sistema de GR.

CAPITULO 4: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En el presente capítulo se hará énfasis en dos aspectos fundamentales durante el desarrollo de un software: implementación y pruebas al sistema. Se explicará cómo está concebido el sistema a través del diagrama de despliegue y se describirán los casos de prueba para garantizar la calidad de la herramienta que será desarrollada

4.1. Diagrama de Componentes

Un diagrama de componentes representa los segmentos de aplicaciones, controladores embebidos y otros elementos que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción mucho más elevado que un diagrama de clase. Usualmente un componente se Implementa por una o más clases en tiempo de ejecución. Eventualmente un componente puede comprender una gran porción de un sistema(Mora Sánchez, 2010).

A continuación se muestra el diagrama de componente correspondiente a los casos de uso más significativos del sistema.

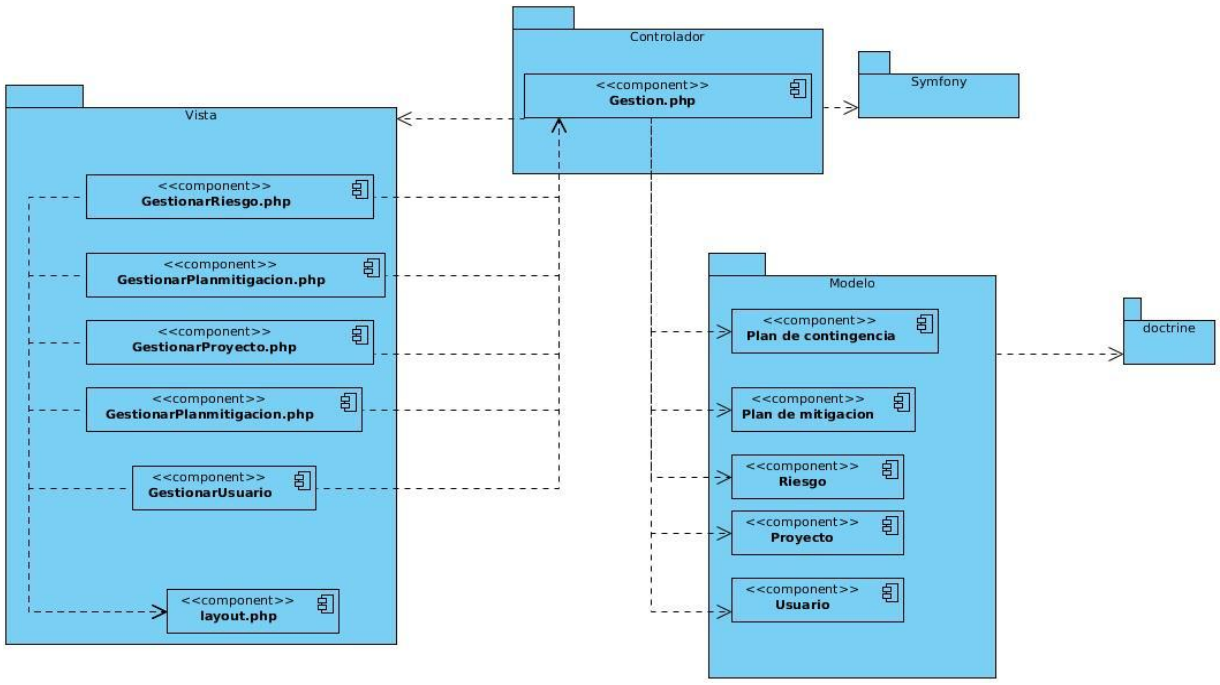


Figura 12: Diagrama de componente para los CU más significativos

Modelo de Despliegue

El Modelo de Despliegue muestra la disposición física de los distintos nodos que componen la herramienta web para la gestión de riesgos en los proyectos del centro GEySED y el reparto de los componentes sobre dichos nodos. Dicho modelo está conformado por una pc cliente para la visualización de las peticiones realizadas por el usuario. La pc cliente se comunica con el servidor web mediante el protocolo HTTP. A su vez este nodo se comunica con el servidor de base de datos que es el encargado de manejar toda la información referente a los riegos, a los proyectos y a los usuarios. Esta relación se realiza mediante el protocolo de comunicación TCP por el puerto 5432.

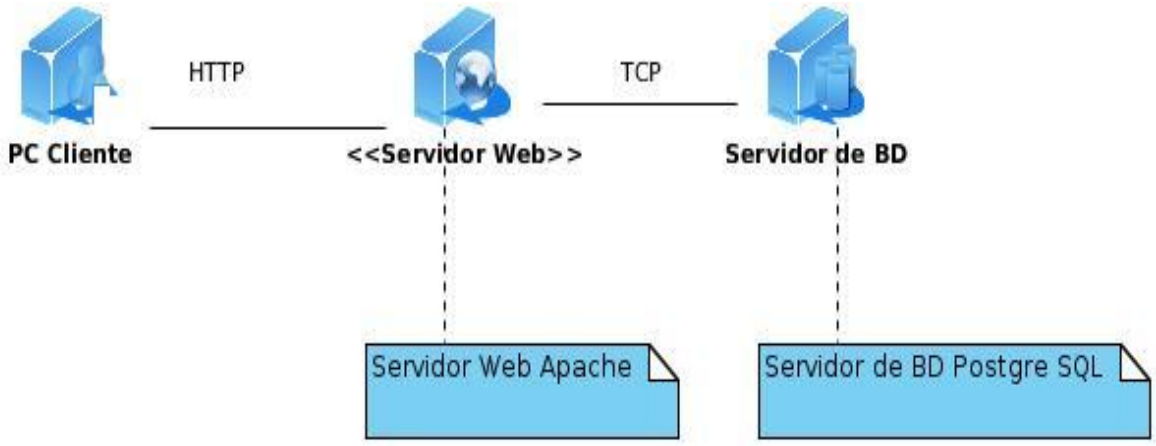


Figura 13: Modelo de Despliegue

4.2. Pruebas de software

Las pruebas de software son un factor importante para asegurarse de que se va por el camino correcto para alcanzar la calidad esperada. Son los procesos que permiten verificar y revelar la calidad de un producto. Estas se integran dentro de las diferentes fases del ciclo de vida del software. Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación(Pressman, 2005).

Para garantizar la calidad del producto, las pruebas de software se deben planificar y ejecutar durante todo el ciclo de vida del software; es por ello que existen diferentes tipos de pruebas para las distintas etapas del desarrollo de software.

Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas. Estas pruebas se centran principalmente en los requisitos funcionales del software.

Para garantizar la calidad de la herramienta web se le realizan las pruebas de caja negra, porque se basa fundamentalmente en la interfaz de usuario, Para la implementación de la misma no se realizaron métodos complejos por lo tanto validando solo el cumplimiento de los requerimientos funcionales del software se verifica si la herramienta cumple con los objetivos esperados.

Para la realización de dichas pruebas se utilizó la técnica de partición equivalente, esta técnica divide el campo de entrada de un sistema en clases de datos que permitan derivar casos de prueba(Gómez Adán, 2012).

4.3.1 Diseño de casos de prueba

Un caso de prueba ideal descubre de forma inmediata una clase de errores, reduciendo así el número de casos de prueba a desarrollar para detectar un error genérico (Gómez Adán, 2012).

Caso de uso: Gestionar riesgo

Descripción General: El CUS se inicia cuando el líder de proyecto selecciona la opción Listado de riesgos, luego selecciona el tipo de gestión que desea realizar si es adicionar un nuevo riesgo, modificarlo, o eliminar uno ya existente, introduce los datos en caso que sea necesario, el sistema realiza la acción seleccionada y termina el CU.

Condiciones de Ejecución: El usuario con el rol de líder de proyecto se debe haber autenticado con anterioridad.

Secciones a probar en el Caso de Uso

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 2 Insertar nuevo riesgo.	EC1.1: Insertar los riesgos con éxito.	El usuario accede a la opción riesgo, le da clic a la opción “Adicionar” de la sección Listado de riesgos y el sistema muestra un formulario con varios campos en los cuales se van a introducir nombre, tipo, proyecto, descripción, efecto, impacto, probabilidad, prioridad, causa, categoría, y estrategia, después de introducir los datos, le da clic al botón “Guardar” y el sistema almacena el nuevo riesgo.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Adicionar.
	EC1.2: No introduce el nombre.	El usuario presiona el botón crear sin introducir el nombre y el sistema muestra un mensaje de error “Este campo es obligatorio”.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Adicionar.
	EC 1.3: No introduce el tipo.	El usuario presiona el botón crear sin introducir el tipo y el sistema muestra un mensaje de error “Este campo es obligatorio”.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Adicionar.

	EC 1.4: No introduce la descripción, el efecto, el proyecto, el impacto, la probabilidad, la causa, la categoría, la estrategia.	El usuario presiona el botón “Adicionar” sin introducir los datos de algunos de los campos siguientes: descripción, efecto, proyecto impacto, probabilidad, causa, categoría, y estrategia., y el sistema muestra un mensaje de error “Este campo es obligatorio”.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Adicionar.
SC 2 Modificar riesgo.	EC 2.1: Modificar riesgo con éxito.	El usuario selecciona el riesgo a modificar presiona la opción “Editar” y modifica cualquier campo. El sistema guarda los cambios correspondientes.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Editar.
	EC 2.2 Deja algún campo vacío	El usuario deja algún campo vacío, el sistema muestra un error “Este campo es obligatorio” indicando que debe llenar los campos que estén vacíos.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Editar.
	EC 2.3: Presiona el botón “Cancelar”	El sistema cancela la acción realizada y muestra el listado de riesgo.	Sistema Gestión de Riesgo / Opción Listado de riesgos / Editar.
SC 3: Eliminar	EC 3.1: Eliminar	El usuario selecciona un riesgo y le da a la opción	Sistema Gestión de Riesgo / Opción

riesgo.	riesgo con éxito.	"Eliminar" El sistema elimina del listado el riego seleccionado.	Listado de riesgos / Eliminar.

Tabla 5: Secciones a probar en el Caso de Uso

Descripción de variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto.	No	Nombre contiene solo letras.
2	Tipo	Seleccionable	No	El tipo contiene solo letras.
3	Descripción	Campo de texto	No	Descripción contiene solo letras.
4	Efecto	Campo de texto.	No	Efecto contiene solo letras.
5	Impacto	Seleccionable	No	Impacto contiene solo letras.
6	Probabilidad	Seleccionable	No	Probabilidad contiene solo letras.
7	Proyecto	Seleccionable	No	Proyecto contiene solo letras.
8	Causa	Campo de texto.	No	Causa contiene solo letras.
9	Categoría	Seleccionable	No	Categoría contiene solo letras.
10	Estrategia	Seleccionable	No	Estrategia contiene solo letras.

Tabla 6: Descripción de variables

SC 1 Insertar riesgo

Escenario	Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 1.1: Adicionar riesgo con éxito.	V/(pérdida de información)	Guarda el nuevo riesgo adicionado.	Satisfactorio
EC 1.2: No introduce el nombre.	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.3 No introduce el tipo	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.4 No introduce la descripción	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.5 No introduce el efecto	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.6 No introduce impacto	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio

EC 1.7 No introduce la probabilidad	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.8 No introduce el proyecto	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.9 No introduce la causa	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.10 No introduce la categoría	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.11 No introduce la estrategia.	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio

Tabla 7: SC 1 Insertar riesgo

SC 2 Modificar riesgo

Escenario	**Nombre**	**Respuesta del Sistema**	**Resultado de la Prueba**

EC 1.1: Adicionar riesgo con éxito.	V/(pérdida de información)	Guarda el nuevo riesgo adicionado.	Satisfactorio
EC 1.2: No introduce el nombre.	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.3 No introduce el tipo	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.4 No introduce la descripción	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.5 No introduce el efecto	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.6 No introduce el impacto	/vacío	Muestra un cartel: “Este campo es obligatorio”, indicando que tiene que llenar el campo vacío.	Satisfactorio

EC 1.7 No introduce la probabilidad	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.8 No introduce el proyecto	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.9 No introduce la causa	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.10 No introduce la categoría	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio
EC 1.11 No introduce estrategia.	/vacío	Muestra un cartel: "Este campo es obligatorio", indicando que tiene que llenar el campo vacío.	Satisfactorio

Tabla 8: SC 2 Modificar riesgo

SC 3 Eliminar riesgo

Escenario	**Respuesta del Sistema**	**Resultado de la Prueba**

EC 3.1: Eliminar riesgo con éxito.	Luego de que el usuario de clic en “Eliminar” el riesgo seleccionado, el sistema lo borra de la lista de riesgos.	Satisfactorio

Tabla 9: SC 3 Eliminar riesgo

Análisis de resultados de las pruebas

Para la realización de las pruebas estas se dividieron en 3 iteraciones: en la primera se encontraron 5 No Conformidades, en la segunda 3 No Conformidades y ya para la tercera y última no encontró ninguna, demostrándose de esta manera el resultado satisfactorio de la aplicación.

Conclusiones del capítulo

En este capítulo se realizó la encapsulación de las clases del diseño en los componentes de código fuente lo que permitió analizar las dependencias entre estos componentes. Luego se analizó la distribución en los nodos de cómputo definidos en el diagrama de despliegue. Se realizó una concepción del modelo de datos, definiendo las relaciones de las tablas creadas por la herramienta web para la gestión de riesgos en los proyectos del centro GEySED lo que permitió reflejar su nivel de integración. Se realizaron las pruebas al sistema donde se obtuvieron 5 no conformidades en la primera iteración y 3 no conformidades en la segunda iteración lo que hizo posible validar los requerimientos funcionales del sistema.

CONCLUSIONES

Para el desarrollo del presente trabajo de diploma se realizó un análisis y una valoración de las herramientas que gestionan los procesos de gestión de riesgo, tanto a nivel internacional como nacional, posibilitando determinar que las mismas no soportan en su totalidad el proceso de gestión de riesgo. Se efectuó una comparación de los procesos más importantes de los modelos existentes para la GR, donde se escogió el modelo MoGeRi, por tener los procesos bien definidos y detallados sirviendo de guía para regir dichos procesos. El uso conjunto de las metodologías, modelos, herramientas y tecnologías facilitó la implementación de un sistema capaz de gestionar todos los procesos que se llevan a cabo en la gestión de riesgos. El desarrollo del sistema mediante el patrón arquitectónico MVC, garantizó la flexibilidad y extensibilidad del mismo, evitando de esta forma la rápida obsolescencia.

Gracias a esto se logró el desarrollo de una herramienta fácil de usar para la gestión de los riesgos basada en el modelo MoGeRi en los proyectos productivos del centro GEySED y la utilización de la misma permite obtener una mayor organización en la información manejada durante el proceso de gestión de riesgos, validado por el resultado satisfactorio de las pruebas realizadas.

RECOMENDACIONES

Se recomienda

- Aplicar el sistema de GR como herramienta de gestión de riesgos en los proyectos del centro GEySED.
- Analizar las posibles vías para que en futuras versiones se implemente en la herramienta desarrollada el registro de usuario usando el archivo de usuario LDAP de la UCI.

REFERENCIAS BIBLIOGRÁFICAS

Alvarez, Sara. 2007. [En línea] 30 de Agosto de 2007. [Citado el: 25 de Mayo de 2013.] www.desarrolloWeb.com/articulos/arquitectura-cliente-servidor.html.

Apache. 2012. Apache. [En línea] 06 de Febrero de 2012. [Citado el: 29 de Abril de 2013.] http://httpd.apache.org/docs/2 .0/es/new_features_2_0.html..

Audrey, D. 2009.*Rethinking Risk Management,Paper presented at the NDIA Systems Engineering.* 2009.

Ávila Guerrero, Alexei Ansberto y Rodríguez Ruiz, Yusimy. 2009.*Sistema para la Gestión de Imágenes Libres para los Proyectos Productivos de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

Balmaseda Acosta, Liliam y Reina Lugo, Jean. 2007.*Sistema de Reportes de Programas Malignos capturados en Segurmática.* La Habana : s.n., 2007.

Barnett Villalobos, Ing. Randall. 2010. Cómo crear tu primerMVC. WebPart en Sharepoint. [En línea] 2010. [Citado el: 30 de Mayo de 2013.] www.sharepoint.com.

BERTINO, E. A. (1995). *Sistemas de bases de datos orientadas a objetos* (Díaz de Santos ed.).

Camacho, Erika, Cardesio, Fabio y Nuñez, Gabriel. 2004.*Arquitectura de Software.* 2004.

Charette, R. N. 1989.*Software Engineering Risk Analuysis and Management.* s.l. : McGraw-Hill, 1989.

Erb, Markus. 2008.*Gestión de Riesgo en la Seguridad Informática.* España : s.n., 2008.

Fuentes Escoba, Claudia y Miló Pérez, Joselín. 2010.*Desarrollo de las consultas de cristalino y aprobación de cirugía del Sistema de Información Hospitalaria alas HIS.* La Habana : s.n., 2010.

Gómez Adán, Víctor. 2012.*Software Testing and ALM done right.* 2012.

Gonzalez Almora, Aymelis y Abigantus Perez, Pedro. 2010.*Implementacion de servicios para la gestion de informacion en el polo PETROSOFT.* La Habana : s.n., 2010.

Gonzalez Cedeño, Yusleimi. 2008.*Gestión de Riesgos del proyecto Sistema de Gestión Penitenciaria.* La Habana : s.n., 2008.

Hechavarria Guibert, Lisandra y Mayol Zaldivar, Ariadna Barbara. 2008.*Gestión de riesgos en el proyecto Sistema de Gestión Fiscal.* La Habana : s.n., 2008.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011.*El proceso de la investigación científica.* 2011.

Higuera. 1996.*Software Risk Management.* 1996.

Jacobson, Booch. 2005.*El proceso unificado de desarrollo de software.* s.l. : Addison Wesley, 2005.

Jacobson, Booch, Rumbaugh, James y otros. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Lambert Claramunt, Javier. 2012.*Implementación de un prototipo funcional para automatizar el proceso de pruebas de Caja Blanca del departamento Señales Digitales del Centro GEySED.* La Habana : s.n., 2012.

León Piñero, Roberlay y Carrodeguas Oliva, Rogelio. 2009.*Automatización del módulo de Referencia de la Biblioteca de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

Lobaina Rodríguez, Danis y Alvarez Almanza, Frank.*Análisis y diseño de la Herramienta Generador y Administrador de Portales Web v2.0.* La Habana : s.n

Lobo, Armando Robert. 2006.*Tutorial Básico de ExtJS.* La Habana : s.n., 2006.

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

Martell Fernández, Vladimir. 2012.*Metodología para la gestión de riesgos en líneas de productos de Software.* La Habana : s.n., 2012.

Martínez Vergara, Yadir. 2010.*Diseño e implementación del componente de configuración de las planillas evaluativas del proceso de evaluaciones.* La Habana : s.n., 2010.

Menéndez, Rafael. 2002.*Gestión de riesgos en ingeniería del software.* 2002.

Milanés López, Aldo y Montano García, Reinier. 2009.*Sistema para la Gestión de la Tecnología en la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

Molina Luis, Reinel. 2007.*Diseño del Sistema Integral para el Calculo de índices de precios al Consumidor.* La Habana : s.n., 2007.

Mora Sánchez, Israel. 2010.*Diseño e implementación del componente configuración del proceso evaluativo.* La Habana : s.n., 2010.

Nava, Mayli. 2009.*Sistemas de Llamada y Retorno.* Venezuela : s.n., 2009.

Pérez Mata, Manuel. 2009. [En línea] 2009. [Citado el: 26 de Mayo de 2013.] www.tecnoretales.com/programacion/que Es doctrine orm/.

Pérez Núñez, Yenly. 2007.*Análisis y diseño de un editor de sonido.* La Habana : s.n., 2007.

Pérez Quintero, Lisett y Verdecia Guerra, Carlos Manuel. 2008.*Estrategia para el desarrollo de requisitos no funcionales de software en proyectos productivos de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2008.

Potencier, Fabien y Zaninotto, François. 2008.*Symfony la guía definitiva.* 2008.

Pressman, Roger S. 2005.*Ingenieria de Software: un enfoque practico.* 5a. La Habana : McGraw Hill, 2005.

Pressman, S. Roger. 1988.*Ingenieria de Sotfware.* s.l. : Mc Graw-Hill, 1988.

—. 2005.*Ingeniería del Software: un enfoque práctico.* 5a edición. La Habana : McGraw Hill, 2005.

Quintero Acosta, Eddy y Blanco Brito, Josue Hugo. 2009.*Gestión de Riesgos para el Proyecto de Gestión Académica Akademos v2.0.* La Habana : s.n., 2009.

Romero B., Alfonso, Lovera D., Daniel y Yaringaño Y., Simeón. 2007.*Gestión de riesgo con CMMI, RUP e ISO en la ingeniería de software.* 2007.

Silva Cobas, Roxana. 2010.*Propuesta de solución de réplica multi-maestra asincrónica para bases de datos.* La Habana : s.n., 2010.

Software ERM. Software ERM | Gestión Riesgos Corporativos. [En línea] [Citado el: 22 de 03 de 2013.] http://www.softexpert.es/gestion-riesgos-corporativos.php.

Valladares Arenas, Ana Silvia. 2010.*Proceso de Gestión de riesgos para proyectos de desarrollo de software de Softel.* La Habana : s.n., 2010.

Zulueta V., Yeleny. 2007.*Tendencias actuales de la Gestión de Riesgos.* La Habana : s.n., 2007.