Universidad de las Ciencias Informáticas
“Facultad 6”

Título: “Herramienta para la gestión de riesgos en los proyectos productivos de la facultad 6.”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Ailyn Curbelo Entenza

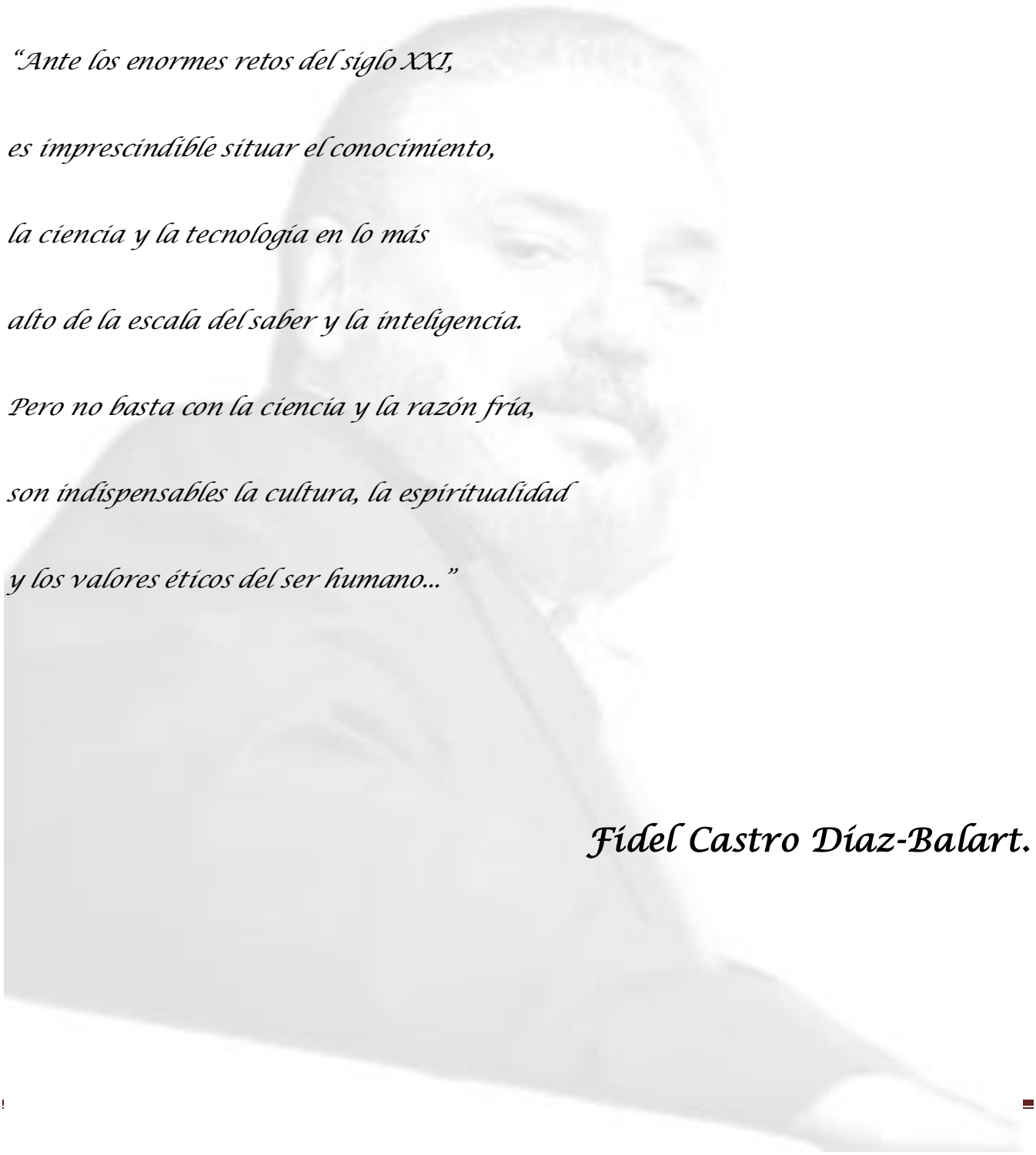
Tutores: Ing. Asnier Enrique Góngora Rodríguez

Ing. Anthony R. Sotolongo León

Ciudad de la Habana, Junio del 2010.

“Año 52 de la Revolución”

PENSAMIENTO



"Ante los enormes retos del siglo XXI,

es imprescindible situar el conocimiento,

la ciencia y la tecnología en lo más

alto de la escala del saber y la inteligencia.

Pero no basta con la ciencia y la razón fría,

son indispensables la cultura, la espiritualidad

y los valores éticos del ser humano..."

Fidel Castro Díaz-Balart.

DECLARACIÓN DE AUTORÍA

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

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

Ailyn Curbelo Entenza

_____

Firma del Autor

Ing. Asnier Enrique Góngora Rodríguez

_____

Firma del Tutor

Ing. Anthony R. Sotolongo León

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Asnier Enrique Góngora Rodríguez
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: agongora@uci.cu

Ing. Anthony Rafael Sotolongo León
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: asotolongo@uci.cu

AGRADECIMIENTOS

Les agradezco a mis tutores por su apoyo incondicional en el desarrollo de esta tesis; a todos los profesores que me ayudaron en mi formación como profesional; a mis compañeros de estudio Salvador, Annia Susel y Marisel por esos momentos de alegría que pasamos juntos. A mis amigos de la FEU Rosado, Azulito, Lien, Pedri, Peña, Freddy, Jorgito, Susy, Pico, Eslavy; con los que pase momentos inolvidables, con los que construí esa gran familia que durante estos 5 años han ocupado un espacio en mi corazón. A todos los que de una forma u otra estuvieron presentes durante estos cinco años y aquellos que me ayudaron en la realización de esta tesis.

DEDICATORIA

Le dedico este trabajo de diploma a mis padres (Cecilia Entenza Rodríguez Y Fidel Curbelo Alonso) las personas que más amo en esta vida, a mi padrastro Arsenio Ponce por cuidar de mi mamá todo este tiempo, a mis abuelos Gerardo, Elsa y Manaca, que desde el cielo me dieron la fuerzas para seguir adelante, a mi abuelita Moringo por sus consejos y apoyo, a mi tía Dorita por estar ahí en todo momento, a mis tías Ibis, Roquelina y Martha Lucia, a mis tíos Miguel Ángel, Carlitos, Papito; en fin a toda mi familia que son lo más lindo que tengo; a Yaneisy, la Piny, Yadira, Greter, Dailin, Ana Niuska, Themis, Ariadna e Iday, que han sido las amigas, hermanas y confesoras. A Serguey por ser el motor impulsor de comenzar esta carrera.

RESUMEN

El desarrollo de un software puede verse afectado por la ausencia o incorrecta gestión de los riesgos. Identificar los riesgos, valorar el impacto que pueden ocasionar al materializarse y la concepción de un plan de contingencia o mitigación, y el monitoreo y control de estos, garantizan en gran medida que se minimice la probabilidad de ocurrencia o sus efectos durante el ciclo de desarrollo de un software. Mantener una estrategia proactiva y no reactiva ante los riesgos es uno de los objetivos fundamentales que todo líder de proyecto debe tener en cuenta. El proceso de informatización del país y de la producción de software para la exportación, llevado a cabo en gran medida por la Universidad de Ciencias Informáticas, no queda exento del impacto arriesgado de los riesgos, de ahí la necesidad de realizar una correcta y oportuna gestión de estos en los proyectos productivos. El presente trabajo de diploma surge teniendo en cuenta investigaciones realizadas en este centro para adaptar el proceso de gestión de riesgos propuesto por la metodología de desarrollo de software Rational Unified Process (RUP), proponiendo la automatización de las áreas de este proceso, entre las que se encuentran: identificación, análisis, priorización, planificación, mitigación y monitoreo de los riesgos.

PALABRAS CLAVE

Gestión de Riesgos, Riesgos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En los últimos años la gestión de proyectos informáticos se ha desarrollado como una disciplina relativamente joven, aunque de importancia creciente en distintos entornos a nivel mundial. Hoy día la gestión de proyectos es un conjunto de principios, dirigidos a ofrecer un enfoque estructural hacia la toma diaria de decisiones que hacen que un negocio funcione de manera adecuada. La gestión de riesgo como parte del proceso de la gestión de proyectos, se ha convertido en un factor muy importante en la industria del software, siendo muy necesaria para lograr los resultados explícitos que figuran en el plan del proyecto, haciendo muy necesario gestionar los riesgos durante el proceso de desarrollo de los software para minimizar los efectos de condiciones desfavorables. Una de las mayores ventajas del análisis y control del riesgo es que permite descubrir oportunidades de proyectos que de otra forma no se llevarían a cabo por ser considerados, demasiado riesgosos.

La gestión de riesgos dentro de la gestión de proyecto es el aumento de los aspectos positivos dentro del proceso de desarrollo del software disminuyendo la probabilidad e impacto de los eventos negativos dentro de este. El proceso de gestión de riesgo está dado por la identificación del riesgo que determina cuáles riesgos podrían llegar a afectar al proyecto y la documentación de las características de cada uno de ellos, el análisis cualitativo del riesgo realizando una evaluación del impacto y probabilidad de los riesgos identificados priorizando los mismos según su potencial impacto sobre el proyecto, el análisis cuantitativo del riesgo que analiza numéricamente la probabilidad de cada riesgo y su consecuencia sobre los objetivos del proyecto, la planificación del riesgo la cual decide como se va a planificar la administración del riesgo en las distintas actividades del proyecto, el planeamiento de la respuesta al riesgo la cual desarrolla opciones y se determinan acciones para mejorar las oportunidades y reducir las amenazas y por último el monitoreo y control de los riesgos donde se lleva a cabo el seguimiento de los riesgos identificados, se detectan aquellos riesgos residuales no identificados con anterioridad y se identifican nuevos riesgos.

Existen estrategias para enfrentar los riesgos; las estrategias reactivas y las proactivas, las estrategias reactivas se manifiestan cuando se deja que los riesgos produzcan sus efectos (en este momento ya no es un riesgo, es una realidad) y entonces se actúa en consecuencia, en estas condiciones lo único que cabe es tomar medidas correctoras, lo que produce mucho tiempo perdido, retraso en el proyecto, gabinete de crisis, etc.; esto no es aconsejable porque pone en grave peligro el desarrollo del proyecto.

Las estrategias proactivas pasan por la evaluación previa y sistemática de todos los riesgos inherentes al proyecto, evaluando sus consecuencias, esto produce la creación de un Plan de Gestión de Riesgos, con sus planes de mitigación, minimizando las consecuencias, planes de contingencia, etc.; en estas condiciones el objetivo es la evasión del riesgo, con menor tiempo de reacción frente a los efectos negativos y una mejor gestión del proyecto en su conjunto: menor tiempo y menor costo.

En la actualidad existen varios métodos para la gestión de riesgos, la mayoría de ellos se consideran como un factor importante dentro de la gestión de proyectos; estos métodos han sido concebidos y creados exclusivamente como un conjunto de pautas a seguir para descubrir y planificar las medidas oportunas para mantener los riesgos bajo control. Los principales modelos de gestión de riesgos propuestos son: Modelo MAGERIT de Gestión de Riesgos en Sistemas adaptado a Proyectos, Modelo de eventos de MAGERIT-Proyectos, Modelo McFarlan, Modelo RiskMan e iniciativa RiskDriver, Modelo DriveSPI, Modelo Eurométodo, Modelo ISPL y Modelo prisa.

Cuba no se encuentra ajena a esta dinámica de cambios en el campo de la informática, actualmente el país se proyecta hacia la búsqueda de nuevos modelos de desarrollo de software informáticos de acuerdo a las peculiaridades nacionales, de forma tal que se garantice un nuevo modelo de gestión empresarial con un elevado desempeño para la empresa estatal socialista. El profesional informático en Cuba debe poseer la capacidad para desarrollar nuevos sistemas de información respondiendo a las necesidades de las diferentes organizaciones dentro de su entorno de trabajo. La ingeniería con la que se construyen los sistemas de información debe ser consistente y permitir su mantenimiento y actualización. En su entorno productivo, el profesional en informática frecuentemente se enfrenta a proyectos de desarrollo de software y requiere los elementos cognitivos para planear, organizar, dirigir y controlar ese tipo de proyecto. El desarrollo de software por deformación profesional induce a buscar e ignorar el riesgo, los desastres en este proceso pueden ser evitados con el tratamiento de aquellos elementos de alto riesgo.

Cuba ha dado un gran impulso a la producción de software nacional, y un ejemplo claro de esto es la Universidad de las Ciencias Informáticas (UCI), la cual es un logro novedoso del país en aras de desarrollar la industria del software y la informatización de la sociedad, es una universidad de nuevo tipo que lleva a la par la formación y preparación de sus estudiantes conjuntamente con la producción de software. En esta universidad de excelencia se aboga porque la línea de producción tenga como prototipo lo planteado por el Capability Maturity Model Integration (CMMI), el cual define una serie de áreas donde

se encuentra incluida la Gestión de Riesgos. Según estudios realizados acerca de cómo se administran los riesgos en la Universidad, se llega a la conclusión de que existen situaciones adversas que se identifican como riesgos, las cuales atentan contra la calidad del producto y con el desarrollo del proyecto, ocasionando: atrasos en los cronogramas, en la entrega del producto y dificultades con la culminación exitosa del software. Actualmente en los proyectos productivos se conocen de cierta forma, los riesgos que podrían afectar el desarrollo, pero no se guían por una taxonomía formal de gestión de riesgos y mucho menos se procede a su análisis o gestión. Por lo que no se tiene medida del nivel de riesgo del proyecto y ocasiona incumplimientos en el término de los objetivos trazados y aumento del costo del mismo.

En la facultad 6 se están llevando a cabo varios proyectos productivos de exportación y nacionales, los cuales se encuentran expuestos a incertidumbres negativas que pueden lograr el fallo o destrucción de elementos claves dentro de la producción; las consecuencias no previstas pueden desarrollarse en el inicio del proyecto o durante la etapa de desarrollo, los fallos comunes están unidos a posiciones incorrectas, las estrategias deficientes tomadas al inicio del desarrollo del proyecto pueden afectar su buena culminación y ocasionar daños a objetos del mismo, ya sea por condiciones de trabajo, pérdida de materiales o equipos, trayendo todo esto aparejado a un déficit en la obtención de mayores ganancias, mejor eficiencia en la producción y en los servicios que brinda nuestra facultad.

Actualmente en la facultad se lleva a cabo el proceso de planificación, ejecución y control de la gestión de riesgos de forma manual. El gran volumen de información y la variedad de actores que intervienen en la gestión de riesgos hacen que este proceso se torne engorroso, lento y poco confiable; haciendo sumamente difícil el proceso de toma de decisiones con vista a mejorar el desarrollo de los proyectos productivos además de realizar análisis muy triviales sobre el tratamiento de los riesgos.

Teniendo en cuenta todo lo anteriormente analizado surge como **problema científico**: ¿Cómo contribuir a la gestión de los riesgos que afectan a los proyectos productivos de la facultad 6?

Por lo cual se tiene como **objeto de estudio:** El Proceso de gestión de riesgos y el **campo de acción** lo conforma: La Gestión de Riesgos dentro de los proyectos productivos de la facultad 6.

Objetivo General: Desarrollar una herramienta para gestionar los riesgos en los proyectos productivos de la facultad 6.

Objetivos Específicos:

1. Diseñar la herramienta para la gestión de riesgos en los proyectos productivos de la facultad 6.
2. Implementar la herramienta para la gestión de riesgos en los proyectos productivos de la facultad 6.
3. Aplicar pruebas a la herramienta para la gestión de riesgos en los proyectos productivos de la facultad 6.

Para dar cumplimiento a los objetivos trazados, se hace necesario que se cumplan las siguientes tareas de investigación:

1. Investigación del proceso de gestión de riesgos en los proyectos productivos de la UCI.
    - Entrevistas con líderes de proyectos.
    - Entrevistas con los especialistas del grupo de calidad de la UCI.
    - Estudio de los documentos: Lista de Riesgo de los Proyectos Productivos y del Plan de Mitigación de Riesgos de los Proyectos Productivos.
2. Estudio de herramientas que realizan la gestión de riesgo.
3. Estudio de las metodologías de desarrollo de software para realizar el análisis y el diseño de la aplicación.
4. Estudio de las tecnologías y herramientas a utilizar en la implementación del sistema.
5. Análisis, Diseño e Implementación del sistema.
6. Con la realización de las tareas investigativas anteriormente expuestas se espera obtener los siguientes resultados:

1. Desarrollar un sistema que brinde las siguientes posibilidades:
    - Gestionar de forma segura los riesgos a los que pueden enfrentarse los proyectos productivos de la Facultad 6.
    - Aumento de la eficiencia en la gestión de riesgos en los proyectos productivos de la Facultad 6.

El diseño teórico ayuda a concretar el objetivo del trabajo de diploma, pues en el mismo quedan definidos todos los elementos necesarios para buscar la respuesta al problema de investigación; pero también es de

vital importancia el diseño metodológico para la ejecución de una investigación, pues define el tiempo necesario para su realización, el costo de la misma y la calidad de los resultados obtenidos, permitiendo así definir la estrategia de investigación más adecuada para cumplir los objetivos propuestos.

Los métodos de investigación científica que se utilizan en esta investigación son:

Métodos Teóricos

- Analítico-Sintético: Se analizan documentos, sitios web, artículos electrónicos, libros, etc., buscando las características de los diferentes modelos y herramientas existentes para la gestión de riesgos, permitiendo la extracción de los elementos más importantes que se relacionan con la gestión de riesgos en los proyectos productivos.

- Análisis Histórico Lógico: Se realiza el análisis histórico lógico de los procesos para la gestión de riesgos que actualmente se llevan cabo en los diferentes proyectos productivos en la Facultad 6.

Métodos Empíricos

- Entrevista: Se realizan entrevistas a varios Jefes de Proyecto de la Facultad 6 para conocer cómo se desarrolla el proceso de gestión de riesgos en los proyectos productivos (métodos, herramientas, etc.). Además con miembros del grupo de investigación de Calidad en la Universidad de Ciencias Informáticas.

Este trabajo de diploma cuenta con la Introducción, 4 capítulos, Conclusiones y Recomendaciones. A continuación se realiza una síntesis de los capítulos a desarrollar en el presente trabajo de diploma:

Capítulo 1: "Fundamentos teóricos". En este capítulo se describe de forma general la gestión de riesgo, realizando un estudio de los procesos, lenguajes, las metodologías para la gestión de los riesgos, además de la descripción de las herramientas a utilizar en el desarrollo de la solución propuesta, basada en la automatización de alguna de las áreas de la gestión de riesgos propuesta por la metodología RUP.

Capítulo 2: "Características del Sistema". En este capítulo se realiza la modelación del negocio, se identifican los requisitos funcionales y no funcionales del sistema propuesto, se realiza la modelación del sistema, generándose los artefactos correspondientes a estos flujos de trabajo según la metodología RUP.

Capítulo 3: "Análisis y Diseño del Sistema". Teniendo como base los artefactos generados durante la modelación del negocio y el sistema, en este capítulo se muestra el diseño de la solución propuesta según la metodología, el framework de desarrollo y la arquitectura seleccionadas. Se presenta además el diagrama de despliegue que muestra la distribución física del sistema para su futura implantación.

Capítulo 4: "Implementación y Prueba". En este capítulo se presentan los artefactos generados durante el flujo de trabajo de implementación entre los que se encuentran los diagramas de componentes que muestran la relación de dependencia entre los ficheros de la aplicación y las pruebas realizadas al sistema en la validación, para el despliegue futuro.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS.

INTRODUCCIÓN

Este capítulo tiene como objetivo plantear el hilo conductor y la estructura del marco teórico referencial del presente trabajo de diploma. En él se realiza una breve descripción sobre la investigación y los pasos para llevar a cabo la gestión de riesgos, haciéndose necesario definir una metodología de trabajo y contar con herramientas de administración de riesgos, para darle solución al problema planteado anteriormente.

1.1. Gestión de proyectos.

La **gestión de proyectos** es la disciplina de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y costo definidos. Muchas veces la gestión de proyectos es responsabilidad de un solo individuo, éste raramente participa de manera directa en las actividades que producen el resultado final. Dentro de las áreas de conocimiento que están inmersas en los procesos definidos por el modelo CMMI, se encuentra la de Gestión de Riesgos, esta es una disciplina compleja y constituye el centro de atención de este trabajo de diploma (1).

1.2. Principales Definiciones de la Gestión de Riesgos.

¿Qué es riesgo?

Un riesgo se puede definir como la exposición ante una pérdida o perjuicio, materializándose en un momento determinado dentro del desarrollo del proyecto (2).

Tipos de riesgos.

Cuando se examinan los riesgos es importante medir el acontecimiento que caracteriza al riesgo dando una sinopsis que pueda o no ocurrir, y el grado de pérdida asociado a cada uno, indicando que si el riesgo se convierte en realidad, ocurrirán consecuencias no deseadas en el proyecto. De acuerdo a la ocurrencia que pueda tener un riesgo en el proyecto se consideran diferentes categorías (3):

- **Los riesgos de proyecto:** Son aquellos que amenazan al plan del proyecto. Si estos llegan a hacerse realidad; pueden provocar un retraso en la planificación temporal y los costos pueden aumentar considerablemente. Además de identificarse otros problemas que están enfocados principalmente en el personal (asignación y organización), recursos, los clientes, requisitos y el impacto que pueda causar este en el proyecto.

- **Los riesgos técnicos:** Son los que tienden a amenazar la calidad del producto y la planificación temporal. Si este tipo de riesgo se llega a convertir en realidad, la implementación y otras disciplinas como el diseño, prueba y mantenimiento pueden llegar ser difíciles o imposibles.
- **Los riesgos de negocio:** Ponen en peligro la viabilidad del software a construir. A menudo amenaza al éxito del proyecto o el producto. Existen cinco tipos de riesgos de negocio:

    1. Riesgo de mercado: Es cuando se construye un producto o sistema excelente, que al final no es de interés de nadie en particular.

    2. Riesgo estratégico: Cuando el producto que se construyó no encaja en la estrategia comercial de la compañía.

    3. Riesgo de comercialización: Es aquel producto que se construye, y el departamento de ventas no sabe cómo va a realizar la venta.

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

    5. Riesgo de presupuesto: Perder presupuesto o personal asignado.

- **Los riesgos conocidos:** Son aquellos que se pueden descubrir después de una cuidadosa evolución del plan, dentro del entorno técnico y comercial en el que se desarrolla el proyecto, además de otras fuentes de información fiables. (Ejemplos: fechas de entregas poco realistas, falta de especificación de requisitos o de ámbito de software).
- **Los riesgos predecibles:** Son aquellos que se extrapolan a otros proyectos, por la experiencia adquirida anteriormente. (Por ejemplo: cambio del personal, mala comunicación, disminución del esfuerzo del personal, etc.).
- **Los riesgos impredecibles:** Pueden ocurrir pero son extremadamente difíciles de identificar.

Métodos de resolución de riesgos.

Los métodos de resolución de riesgos surgen para contrarrestar los riesgos que retrasan el éxito de los proyectos y hacen que peligre su calidad, es por ello que han surgido cuatro métodos de resolución de riesgos. Estos métodos son: eliminación, retención, evitación y transferencia (4).

- **Eliminación del Riesgo:** Se trata de eliminar los factores que inducen el riesgo y con ello eliminar la posibilidad de exposición.

- **Retención del Riesgo:** Muchos de los riesgos que se identifican en el proyecto, no es posible eliminarlos y por tanto hay que desarrollarle otro tipo de tratamiento. La retención, es la asunción por parte de los responsables del proyecto, de informar la existencia de un riesgo. El mero hecho de acometer un proyecto, es un riesgo que no puede ser evitado y sólo se debe retener.
- **Evitación del Riesgo:** En estos casos hay que identificar los factores que provocan el riesgo y mantenerlos bajo control, para evitar que el mismo provoque sus efectos. En caso de que el riesgo amenaza permanentemente el proyecto, se debe evitar sus efectos.
- **Transferencias del Riesgo:** Algunos tipos de riesgos (normalmente poco probables pero muy negativos) pueden ser transferidos a terceros, mediante la contratación de seguros o haciendo contratos en los que el cliente o los proveedores asumen este riesgo y liberan al equipo de proyecto de su gestión.

¿Qué es gestión de riesgos?

La Gestión de Riesgos es la práctica compuesta por 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 proactivas en base a determinar constantemente que puede ir mal (riesgos), identificar cuáles son los riesgos más importantes en los cuáles enfocarse e implementar estrategias para gestionarlos (5).

Es fundamental lograr una clara descripción del riesgo de forma tal de que el mismo pueda ser comprendido y manejado adecuadamente; cuando este aparece, no solo debe considerarse el síntoma sino también sus consecuencias. Se puede afirmar que los principales objetivos que engloba la Gestión de Riesgos son: identificar, analizar, controlar y eliminar las fuentes que provocan los mismos antes de que empiecen a afectar el cumplimiento del proyecto.

Proceso de Gestión de Riesgo.

El proceso de gestión de riesgo se basa en la identificación, análisis, priorización, planificación, resolución y/o mitigación y monitoreo de los riesgos en los proyectos productivos. En este trabajo de diploma se crean las bases para contribuir a la gestión de riesgos en los proyectos productivos de la facultad 6 implementando cada paso del proceso de gestión de riesgos (6).

- **Identificación de los Riesgos.**

La identificación de los riesgos no es más que la determinación de los riesgos potenciales, mediante la información disponible acerca de las amenazas que puedan afectar el desarrollo del

proyecto en sus áreas de trabajo, esta etapa se realiza con todo el equipo de trabajo del proyecto, teniendo este la responsabilidad de realizar una revisión detallada de la situación actual del proyecto y las percepciones de estos sobre los riesgos que puedan presentarse. Este paso genera como artefacto de salida la lista de riesgos del proyecto.

Existen diferentes técnicas para la identificación de los riesgos potenciales en el proyecto, estas son las Listas de Comprobación de Riesgos, no es más que el desarrollo de una serie de preguntas que determinan riesgos frecuentes, y Análisis de Supuestos que es una comparación de las diferentes situaciones que se pueden generar. Finalmente, se lista un conjunto de "componentes y controladores del riesgo" junto con sus probabilidades de aparición. Los controladores del rendimiento, el soporte, el coste y la planificación temporal del proyecto se estudian como respuesta a preguntas posteriores (7).

- **Análisis de los Riesgos.**

El análisis de los riesgos es el proceso de examinar los riesgos en detalle para determinar su extensión, interrelación y su importancia, determinando la probabilidad e impacto (o efecto) asignados a cada riesgo (2).

- Impacto: consecuencia que puede traer que se manifieste un riesgo en un proyecto.

Tabla 1: Impacto del Riesgo.

Impacto del Riesgo	Valor	Descripción
Catastrófico	5	Pérdida del sistema. Costo mayor del 50%
Crítico	3	Recuperación de la capacidad operativa. Costo mayor del 20% y menor del 50%
Tolerable	1	Capacidad operativa mermada. Costo mayor del 10% y menor del 20%

- Probabilidad: aspecto que determina la ocurrencia de un riesgo en el proyecto, la cual se clasifica de acuerdo a su descripción en: muy alta, alta, moderada, baja o muy baja.

Tabla 2: Probabilidad del Riesgo.

Tipo de Probabilidad	Valor	Descripción
Muy baja	1	Menor del 10%

Baja	3	Del 10% al 22%
Moderada	5	Del 25% al 75%
Alta	7	Del 75% al 90%
Muy alta	9	Mayor del 90%

Para realizar la modificación del artefacto lista de riesgos del proyecto incluyendo el análisis, se va a realizar una lista ordenada de acuerdo al impacto y probabilidad que posee el riesgo en el proyecto de menor a mayor, dando así la ocurrencia en que pueda ocurrir este.

- **Priorización de los Riesgos.**

La priorización de los riesgos produce una lista ordenada de los riesgos que mayor relevancia tienen en el proyecto. La prioridad se calcula multiplicando la probabilidad por la consecuencia, efecto o impacto y se realiza una equivalencia (8) entre estos aspectos (impacto, probabilidad y exposición al riesgo).

Clasificación de la Prioridad:

- Riesgo Alto: Se requiere una investigación detallada y una planificación a niveles superiores.
- Riesgo Importante: Se requiere una atención del personal superior.
- Riesgo Significativo: Se debe especificar la responsabilidad de gestión.
- Riesgo Bajo: Se maneja mediante procedimientos de rutina.

- **Planificación de los Riesgos.**

Este proceso asegura que los riesgos identificados sean tratados correctamente, la eficacia se determina por el aumento o disminución del riesgo para el proyecto, realizando un cronograma de tratamiento al riesgo con la aplicación de los métodos de tratamiento al riesgo, el cual se plasma en el plan de contingencia.

Existen técnicas de planificación de los riesgos, entre las que se incluyen las listas de comprobación de técnicas de resolución de riesgos y análisis del costo-beneficio (2).

- Estrategias de evitación: Se trata de minimizar la probabilidad de que el riesgo se presente.
- Estrategias de minimización: Se trata de reducir el impacto del riesgo en el producto o en el proyecto.

- Planes de contingencia: Si el riesgo se presenta, el plan de contingencia se encarga de tratar este riesgo, para los efectos negativos del riesgo pero éste ya se ha producido.

- **Mitigación de los Riesgos.**

La mitigación tiene la tarea de implementar las estrategias que se tomaron en la planificación, poniendo en práctica los planes de contingencia, además se definen un conjunto de técnicas, que agilizan este proceso, como son:

1. Poda de requisitos: Se realiza una selección de los requisitos más importantes, en caso que se tenga que realizar una primera versión del producto.
2. Prototipado: Se brinda una breve panorámica acerca de las acciones que se llevarán a cabo para mitigar los riesgos.
3. Desarrollo incremental: Consiste en la utilización de incrementos para aumentar gradualmente el alcance, empezando por las posibilidades más básicas de las aplicaciones y ampliándolas paso a paso.

- **Monitoreo de los Riesgos.**

El monitoreo no es más que el control que se realiza a los riesgos identificados al inicio de un proyecto, y la identificación de otras amenazas durante el progreso del proyecto, corrigiendo las desviaciones de las acciones de los riesgos planificados, estas actividades de control se incluyen en el Plan de Gestión de Riesgos este es un documento que incorpora los objetivos, estrategias y métodos para llevar a cabo la gestión del riesgo de un proyecto. Para cada riesgo se debe comunicar, en caso de ocurrir un aviso ante la ocurrencia del riesgo, la necesidad de aplicar el plan de contingencia, asignando los recursos necesarios para el cumplimiento de este, ya que la efectividad del cumplimiento de este consiste en los mecanismos de comunicación internos y externos entre los encargados del desarrollo de la gestión del riesgo dentro del proyecto.

1.3 Herramientas que realizan la Gestión de Riesgos en el mundo.

En la actualidad existen herramientas de software para la de gestión de riesgos, disponibles en el mercado. Estas herramientas se enfocan sólo 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 WelcomRisk), o bien emplean un mecanismo que no se orienta al uso de taxonomías (ARM – Active Risk Manager).

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 (9).

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 Strategic Thought tiene un costo muy alto.

Technical Risk Identification and Mitigation System (TRIMS).

TRIMS es un componente del sistema PMWS (Programs manager´s workstation), es una herramienta diseñada para identificar y cuantificar los riesgos en un proyecto con el objetivo de reducir o mitigar estos riesgos a niveles aceptables. Esa herramienta puede ser utilizada en todas las fases definidas por RUP para un proyecto de desarrollo de software, además de aplicarse como herramienta de gestión de riesgos durante el ciclo de vida del producto que resulta del proyecto. Es una aplicación desarrollada para Windows que funciona con los sistemas operativos Windows 98, ME, 2000, y XP. Es un software propietario de Best Manufacturing Practices (10).

Risk Trak.

RiskTrak Internacional (RTI) es una empresa que diseña y desarrolla software para la gestión de riesgos de negocio en un proyecto, programa o nivel de empresa. Las herramientas de software que esta produce ayudan en la identificación, definición, estimación y análisis de las incertidumbres o riesgos, lo que mejora su posición competitiva. Esta empresa ha invertido en la investigación y desarrollo para promover aún más RiskTrak ™ como herramienta de gestión de riesgos, la cual ha sido financiada con fondos privados. RiskTrak realiza la identificación, definición, estimación, análisis, evaluación y mitigación de los riesgos, a fin de realizar la gestión de riesgos del producto mediante el empleo de base de datos, ha sido diseñado para ser fácilmente integrada en un proyecto existente, y las metodologías del programa. Este producto se encuentra valorado en un precio de 1495 dólares (11).

Chinchón-Análisis del Riesgo.

Chinchón utiliza XML para especificar el sistema cuyos riesgos se desea analizar. XML es un formato que se puede producir con cualquier procesador de textos y puede ser presentado

directamente por el Internet Explorer de Microsoft, esta herramienta no incluye ninguna facilidad para generar los modelos en formato XML. Es una herramienta para analizar cuantitativamente el riesgo de un sistema de información, sigue el modelo Magerit 1.0 y requiere de Java2 (12).

De acuerdo a lo anteriormente planteado es necesaria la construcción de una herramienta para la gestión de riesgos en la facultad 6 debido a que las existentes están patentizadas por empresas privadas, enfocan su trabajo solamente a una categoría de los riesgos, cuando existen varias categorías y pueden surgir otras nuevas; emplean mecanismo que no se orienta a el uso de taxonomías, y la que pueden adquirirse por ser gratuita no realiza el proceso completo de gestión de riesgos.

1.4 Herramientas y lenguajes para dar la solución al problema planteado.

1.4.1 Lenguaje de modelado.

UML

El Proceso Unificado RUP (Rational Unified Process) utiliza el Lenguaje de Modelado UML (Unified Modeling Language) para preparar todos los esquemas de un sistema software. UML es una parte esencial del proceso unificado, consolidado como el lenguaje estándar en el análisis y diseño de sistemas de computarizados. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. UML recomienda 9 diagramas que son modelados durante las diferentes fases de desarrollo del software, diagrama de casos de uso, diagrama de clases, diagrama de objetos, diagrama de secuencia, diagrama de colaboración, diagrama de estados, diagrama de actividades, diagrama de componentes y diagrama de despliegue.

1.4.2 Herramientas de modelado.

Con el fin de automatizar los aspectos claves del proceso de desarrollo de un sistema se hace necesaria la utilización de las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador).

Rational Rose.

Rational Rose es una herramienta de diseño de software orientado a objeto mediante UML para el modelado visual y la construcción de componentes para las aplicaciones de software. Es la herramienta CASE que comercializan los desarrolladores de UML, que soporta de forma completa la especificación del UML. Propone la utilización de cuatro tipos de vistas para realizar un diseño del sistema: vista de Caso de Uso, vista Lógica, vista de Componentes y la vista de Despliegue;

permitiendo crear y refinar de esta forma un modelo completo que represente el dominio del problema y el sistema de software.

Visual Paradigm-UML.

Visual Paradigm en su versión 3.4 para UML es una de las herramientas CASE del mercado, considerada como muy completa y fácil de usar, es multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo vital completo del desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación; proporciona características tales como la generación de código, ingeniería inversa y generación de informes. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación presenta un diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; posee un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de ingeniería directa (versión profesional) e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; integración con los principales IDEs; como Eclipse, NetBeans, entre otros para soportar las fases de implementación en el desarrollo de un software.

Por todo lo antes planteado se decide utilizar como herramienta de modelado en el desarrollo de la aplicación, Visual Paradigm; la cual presenta una interfaz de usuario de fácil uso y muy amigable que permite realizar los diagramas y artefactos que se generan durante el desarrollo del software.

1.4.3 Lenguajes de programación para la Web.

Entre los distintos lenguajes de programación para la Web que existen en la actualidad, se destacan dos grupos, que se diferencian entre sí por el lugar que ocupan en la arquitectura Cliente / Servidor, característica de los sistemas Web. El primer grupo está formado por los lenguajes que se ejecutan en el servidor (en inglés, *Server Side Languages*), dentro de los que se encuentran PERL, ASP, PHP y Java. Estos lenguajes se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a Bases de Datos y el tratamiento de la información. El segundo grupo lo integran los lenguajes que se ejecutan en el cliente (en inglés, *Client Side Languages*), ejemplos de ellos son: Java Script y el Visual Basic Script, ambos encargados de aportar dinamismo a la aplicación en los navegadores.

Asp.Net.

El ASP.NET es un lenguaje comercializado por Microsoft, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzada al mercado mediante una estrategia de mercado denominada .NET. Creada para resolver las limitantes que brindaba su antecesor ASP, desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Para el funcionamiento de las páginas se necesita tener instalado IIS con el Framework .Net. Microsft Windows 2003 incluye este framework, solo se necesitará instalarlo en versiones anteriores. Este lenguaje se encuentra completamente orientado a objetos, contiene controles de usuario y personalizados, existe una división entre la capa de aplicación o diseño y el código, facilita el mantenimiento de grandes aplicaciones, incrementa la velocidad de respuesta del servidor, para una mayor velocidad y seguridad logrando así un mayor consumo de recursos.

PHP.

Entre los competidores principales de PHP 5.2.5 se puede citar a Perl, Microsoft Active Server Pages (ASP), Asp.Net, Java Server Pages (JSP), Allaire ColdFusion, Ruby, Phyton. Sin embargo en comparación con estos productos, PHP cuenta con muchas ventajas, entre las que se encuentran las siguientes: interfaces para una gran cantidad de sistemas de base de datos diferentes, bibliotecas incorporadas para muchas tareas Web habituales, bajo costo, facilidad de aprendizaje y uso, portabilidad y acceso a código abierto. El uso de ODBC (del inglés Open Database Connectivity Standard, Estándar de conectividad abierta de base de datos) permite establecer una conexión a cualquier base de datos que suministre un controlador ODBC.

Lenguaje Javascript.

Los lenguajes de programación del lado del cliente, como su nombre lo indica, son los que se ejecutan en el cliente o navegador, creado por Brendan Eich en la empresa Netscape Communications para la utilizarse principalmente en páginas web, encomendado para darle dinamismo a la página sin necesidad de enviar información al servidor para realizar las operaciones requeridas. Son lenguajes interpretados que pueden acceder a la información HTML que se muestra en el navegador, modificarla o actualizarla según las necesidades del programador.

Java Script es uno de los lenguajes del lado del cliente más utilizados por ser compatibles con la mayoría de los navegadores modernos. Es un lenguaje con muchas posibilidades,

permite la programación de pequeños scripts así como la de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas. Permite además el acceso a los elementos que componen la página Web, brindándole la posibilidad al programador modificar el contenido de la página dinámicamente.

Para la implementación del sistema se escoge como lenguaje de programación del lado del servidor se escogió PHP, al garantizar con sus características, obtener un producto que se pueda montar en cualquier sistema, ya que desarrollado el sitio en este lenguaje permite portar el sitio de un sistema a otro sin prácticamente ningún trabajo. Agilidad en la elaboración del producto al contar con librerías donde están implementadas las funciones necesarias para dar solución a gran cantidad de funcionalidades del sistema. Los lenguajes de programación del lado cliente son totalmente independiente del servidor, están insertados en la página Web y son interpretados y ejecutados por el navegador, de modo que cuando el navegador recibe una página web, interpreta y da formato al contenido de la página y entra el código de los scripts al programa intérprete correspondiente para lo cual se utiliza en la implementación de la aplicación como lenguaje scripting, Javascript.

1.4.4 Entorno de desarrollo.

NetBeans.

NetBeans es un entorno de desarrollo integrado OpenSource y de distribución gratuita que proporciona herramientas muy cómodas y de fácil uso para el desarrollo de aplicaciones sobre la plataforma JAVA. Bajo este IDE se pueden desarrollar soluciones J2SE, J2ME y J2EE. Entre las características que posee este entorno de desarrollo se encuentra (6):

Desarrollo de aplicaciones multiplataforma sobre: MacOS, Windows, Linux.

1. Add-ons para desarrollo Móvil, integración con SOA, optimización de aplicaciones y desarrollo con C y C++.
2. Cliente CVS integrado.
3. Crecimiento de plataforma por medio de plugins. Entre los plugins que existen se tienen los siguientes herramientas: de java para la mejora de desarrollo de aplicaciones, de modelado UML y XML.
4. Struts, JSF, EJB, WebServices, etc.

Eclipse Editor PHP.

Eclipse Editor PHP es un entorno de desarrollo para PHP, Java, C/C++ y otros lenguajes, dispone de una potente administración de proyectos y ficheros, un gran editor con coloreado del lenguaje,

detección y resaltado de errores sintácticos y de estructuras. Se trata de un entorno IDE cuya principal ventaja es la visualización de los errores de escritura, de inclusión de cabeceras, además dispone del manual de PHP integrado y de rápido acceso.

Zend Studio.

Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Es la propuesta de Zend Technologies para el desarrollo de aplicaciones Web utilizando PHP, actuando Zend Studio como la parte cliente y Zend Platform como la parte servidora. Se trata de un software comercial, lo cual contrasta con el hecho de que PHP es software libre. Zend Studio fue diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Javascript y XML.

Para la solución del sistema se seleccionó el NetBeans, pues posee numerosas características que hacen que el IDE sea atractivo para cualquier desarrollador, incluyendo la amplia integración de las características específicas de la tecnología Java que no se encuentran disponibles en otros conjuntos de herramientas de aplicaciones multiplataforma. NetBeans elimina la necesidad de los equipos de desarrollo que tienen que invertir demasiado tiempo manteniendo los modelos actualizados con revisiones exhaustivas que garanticen la actualización de indicadores y códigos de anotación. Juntas, estas características pueden ahorrar al desarrollador muchas horas de trabajo y acelerar la disponibilidad de un nuevo programa. Entre las características que lo evalúan está su facilidad de uso, su cumplimiento de regulaciones, sus perfiles de rendimiento, además de su flexibilidad entre plataformas.

1.4.5 Framework de desarrollo.

Symfony.

Symfony es un framework para construir aplicaciones web con PHP, en otras palabras, contiene un enorme conjunto de utilidades que simplifican el desarrollo de las aplicaciones web. Es una de las mejores copias para PHP del famoso framework Ruby on Rails. Emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas capas que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás

elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista. Permite utilizar bases de datos de tipo SQLite, MySQL, PostgreSQL, Oracle y SQL Server. Cambiar de base de datos sólo requiere modificar la opción DSN, ya que el resto de la aplicación funciona igual de bien con cualquier sistema gestor de base de datos.

Kumbia.

KumbiaPHP es un framework para aplicaciones web libre, escrito en PHP5. Basado en las prácticas de desarrollo web como DRY y el Principio KISS para software comercial y educativo. Kumbia fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. Este framework intenta proporcionar facilidades para construir aplicaciones robustas para entornos comerciales. El principal objetivo es producir aplicaciones que sean prácticas para el usuario final y no solo para el programador. La mayor parte de tareas que le quiten tiempo al desarrollador deberían ser automatizadas por KumbiaPHP para que se pueda enfocar en la lógica de negocio de su aplicación.

Web4j.

Web4j es un framework que se ha ido construyendo muy lentamente al paso de los años, últimamente ha tomado una madurez que se dice "está listo para un primer tiempo". Entre sus características se destacan: un completo stack de herramientas para aplicaciones Web basadas en Java, tiene una filosofía profunda de minimalismo y simplicidad, se dice que es de los frameworks más pequeños porque su base está conformada por solo 82 clases, no tiene la costumbre archivos xml o anotaciones, no contiene etiquetas personalizadas para el formulario de control (Las formas se llevan a cabo en HTML plano) y permite las declaraciones SQL mediante archivos.

Los Frameworks anteriormente mencionados poseen de forma general grandes mejoras para poder optimizar el trabajo de los desarrolladores en el lenguaje PHP. A la hora de escoger uno de ellos es necesario ver cual tiene mayor cantidad de características que se ajusten específicamente, a la aplicación que se quiere obtener. Teniendo en cuenta lo anteriormente planteado se decide utilizar Symfony, al constituir 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. Además se puede ejecutar tanto en plataformas Unix como en plataforma

Windows. Por último y no menos importante es la maravillosa documentación que posee (sobre todo en español).

1.4.6 Gestor de base de datos.

PostgreSQL.

PostgreSQL en su versión 8.2 es un gestor de base de datos de código abierto que posee una gran escalabilidad, es capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta. Dentro de sus principales características se encuentran aprovechar la potencia de sistemas multiprocesador, gracias a su implementación multihilo, soporta gran cantidad de tipos de datos para las columnas, dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.), gran portabilidad entre sistemas, soporta hasta 32 índices por tabla, gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos, además posee una licencia de tipo BSD.

MySQL.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Por un lado se ofrece bajo la GNU GPL1 para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes. Permite manejar multitud de tipos para columnas y registros de longitud fija o variable así como permite conexiones entre diferentes máquinas con distintos sistemas operativos.

Teniendo en cuenta las características analizadas de los gestores de base de datos expuestos, se opta por usar PostgreSQL en su versión 8.2, la cual incluye mejoras de rendimiento, características de SQL 2003, incluyendo funciones de agregación estadística, sentencias VALUE con múltiples registros, UPDATE RETURNING y funciones de agregación de múltiples columnas, así como índices invertidos generalizados, lo cual constituye una forma más escalable y programable de indexar datos semi-estructurados y texto.

CONCLUSIONES DEL CAPÍTULO

La gestión de riesgos es de vital importancia desde la concepción y planificación de un proyecto informático y hasta su cierre, pues de no tenerse en cuenta podría ser una de las causas potenciales de su fracaso. Esta actividad debe implementarse fundamentalmente de forma proactiva. En este capítulo se propuso asumir la adaptación del modelo definido por RUP para la facultad 6, teniendo en cuenta un grupo de pasos comunes en modelos de gestión de riesgos ya existentes. Dada la necesidad de contribuir a la gestión de riesgos en los proyectos productivos de la facultad 6 se decide implementar un grupo de actividades presentes en la identificación, análisis, planificación resolución y/o mitigación. Para el desarrollo del sistema de gestión de riesgos de los proyectos productivos de la facultad 6 bajo la metodología de desarrollo RUP se decide utilizar el Visual Paradigm versión 3.4 como herramienta CASE, haciendo uso del lenguaje de modelado UML. Para la implementación se propone utilizar como lenguaje de programación PHP 5.2.5 y Java Script, como entorno de desarrollo el NetBeans versión 6.8 utilizando como framework Symfony versión 1.2.7 y como gestor de base de datos PostgreSQL versión 8.2.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

INTRODUCCIÓN

En el presente capítulo, se identifica(n) el(los) proceso(s) del negocio relacionados con la gestión de riegos, una vez descritos permiten identificar las áreas a automatizar. Para la modelación del sistema propuesto se identifican los requisitos funcionales y los requisitos no funcionales asociados a estas áreas, agrupándolos por casos de uso a priorizar y descritos para obtener la documentación necesaria para realizar el diseño del sistema.

2.1. Modelo del Negocio.

En este trabajo, se crea un modelo de negocio, puesto que se describen los procesos existentes y observados para su comprensión. Además se especifica qué procesos de negocio se realiza en el sistema, la identificación de los objetos del negocio y las competencias que se requieren en cada proceso: sus trabajadores, sus responsabilidades y las operaciones que se llevan a cabo.

2.2.1 Actor del Negocio.

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. (13)

En el presente trabajo se define como actor del negocio al Solicitante de Gestión de Riesgo.

Tabla 3: Relación de Actor del Negocio.

Actor del Negocio	Descripción
Solicitante de Gestión de Riesgo	Persona encargada de solicitar que se realice la gestión de riesgos, que puede ser el líder del proyecto, vicedecano de producción o algún estudiante del proyecto asignado para realizar la gestión.

2.2.2 Trabajadores del Negocio.

Un trabajador del negocio representa una abstracción de un ser humano con ciertas capacidades que se requieren en un acaso de uso del negocio. Un trabajador representa el conocimiento y las habilidades que alguien necesita para hacerse cargo del trabajo como trabajador del proyecto (14).

Tabla 4: Relación de Trabajadores del Negocio.

Trabajadores	Descripción
Administrador de Riesgos	Persona encargada de identificar los riesgos, registrar los riesgos, además de analizarlos, priorizarlos y planificarlos en el proyecto.
Revisor de Riesgos	Persona encargada de realizar el monitoreo/control y reporte de los riesgos en el proyecto.
Administrador de Usuarios	Persona encargada de realizar la gestión de los usuarios que accedan al sistema, dándole los privilegios a cada uno.
Usuario	Actor genérico que puede realizar las funcionalidades de los actores que heredan de él, además de ser el encargado de realizar la entrada al sistema.
Gestor de Riesgos	Actor genérico que puede realizar las funcionalidades del administrador de riesgos y el revisor de riesgos.

2.2.3 Caso de uso del Negocio.

Un diagrama de casos de uso (CU) del negocio representa gráficamente a los procesos del negocio y su interacción con los actores del negocio (15).

Fig. 1: Diagrama de Caso de Uso del Negocio.

Diagrama de Actividades.

El diagrama de actividades es un grafo (grafo de actividades) que contiene estados en que puede hallarse una actividad, puede contener bifurcaciones, así como divisiones de control en hilos concurrentes. Describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio. Es similar a un diagrama de estados en el cual todos o la mayoría de los estados son estados de actividad y en la cual todas o la mayoría de las transiciones se disparan al completarse las acciones en los estados fuentes precedentes (15).

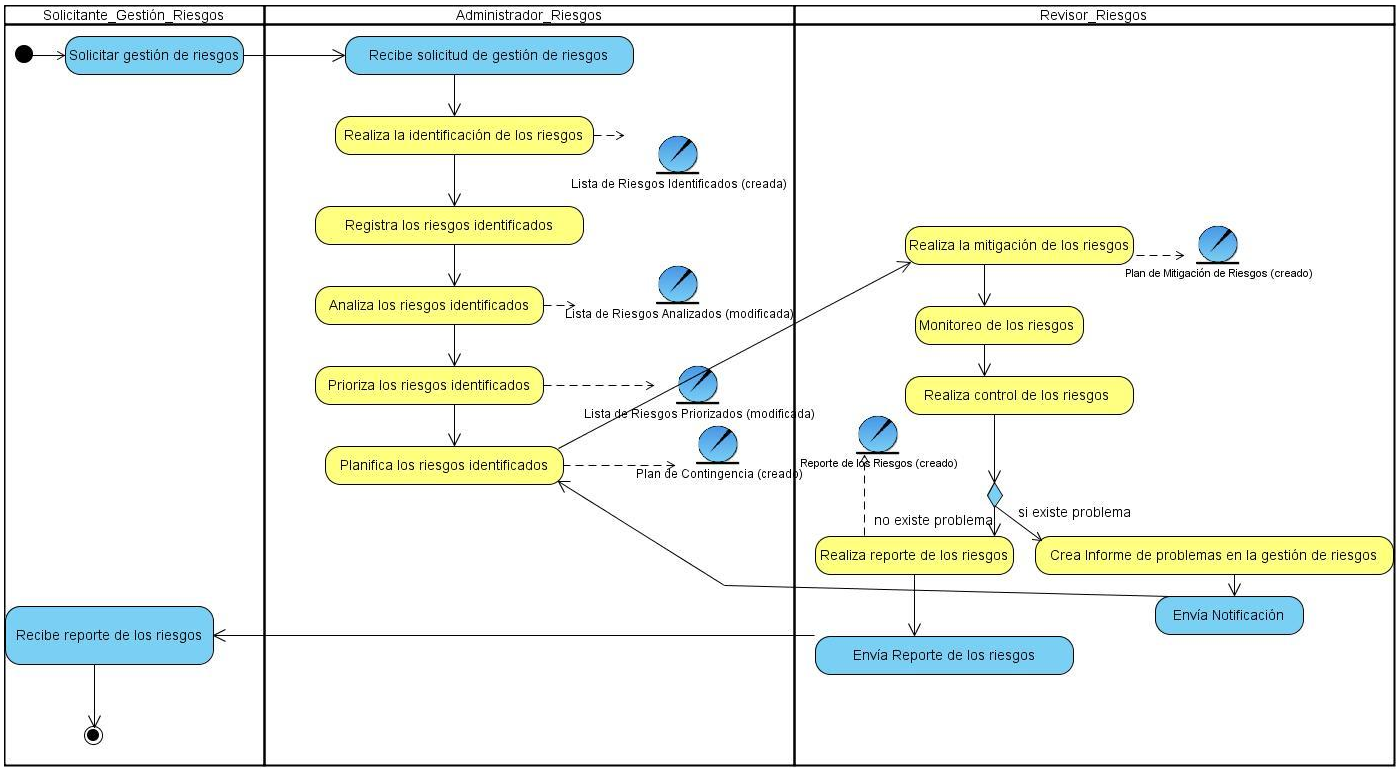


Fig. 2: Diagrama de Actividades del Negocio.

2.2.4 Modelo de Objeto del Negocio.

Modelo de objetos del negocio es aquel que describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo de trabajo del proceso de negocio (15).

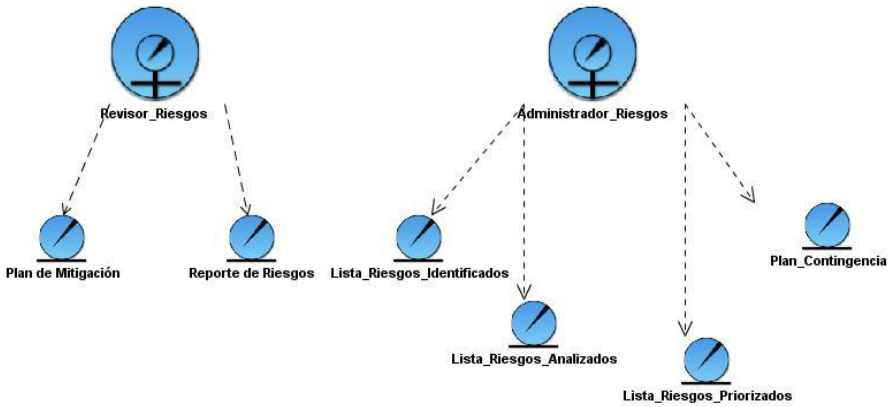


Fig. 3: Diagrama de Objetos del Negocio.

2.2.5 Reglas del Negocio.

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio.

Reglas de estructura

- **Término:**

1. El número del efecto se le asigna el valor de 1 (riesgo tiene un impacto tolerable), 3 (riesgo tiene un impacto crítico) y 5 (riesgo tiene un impacto catastrófico).
2. El número de la probabilidad se le asigna el valor de 1 (riesgo posee una probabilidad de ocurrencia muy baja), 3 (riesgo posee una probabilidad de ocurrencia baja), 5 (riesgo tiene una probabilidad de ocurrencia moderada), 7 (riesgo tiene una probabilidad de ocurrencia alta) y 9 (riesgo tiene una probabilidad de ocurrencia muy alta).

Reglas de acción

- **Restricciones de operaciones:**

1. Para llevar a cabo el proceso de análisis de los riesgos el administrador de los riesgos debe contar con los riesgos identificados en el proyecto.
2. Para llevar a cabo el proceso de priorización y planificación de los riesgos el administrador de los riesgos debe contar con los riesgos analizados en el proyecto.
3. Para hacer el proceso de mitigación de los riesgos el revisor de los riesgos debe contar con los riesgos planificados en el proyecto.
4. Para realizar el proceso de monitoreo de los riesgos el revisor de los riesgos debe contar con los riesgos mitigados en el proyecto.

2.2. Modelo del Sistema.

2.2.1 Solución Informática.

En el presente epígrafe se hace un estudio en el cual RUP realiza una propuesta de series de pasos para la gestión de riesgos en los proyectos productivos de la facultad 6. Para el cumplimiento de los objetivos trazados se propone una aplicación que automatice los procesos de identificación, análisis, priorización, planificación, mitigación, monitoreo y/o control que propone RUP. La solución informática permitirá generar una serie de reportes que permitirá al usuario conocer los resultados del comportamiento de los riesgos en el proyecto en cuestión.

2.2.2 Requerimientos del sistema.

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

Requisitos Funcionales.

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir, estos se mantienen invariables sin importar con que propiedades o cualidades se relacionen. (15) De acuerdo con los objetivos trazados para darle solución al sistema, este debe cumplir con:

RF 1: Adicionar proyecto.
RF 2: Modificar proyecto.
RF 3: Eliminar proyecto.
RF 4: Visualizar proyecto.
RF 5: Generar lista de riesgos.
RF 6: Generar plan de mitigación de los riesgos.
RF 7: Adicionar riesgos.
RF 8: Modificar riesgos.
RF 9: Eliminar riesgos.
RF 10: Visualizar riesgos.
RF 11: Autenticar usuario.
RF 12: Adicionar usuario.
RF 13: Modificar usuario.
RF 14: Eliminar usuario.
RF 15: Visualizar Usuario.
RF 16: Generar plan de contingencia.
RF 17: Identificar riesgos en los proyectos productivos.
RF 18: Analizar riesgos en los proyectos productivos.
RF 19: Priorizar riesgos de los proyectos productivos.
RF 20: Planificar riesgos de los proyectos productivos.
RF 21: Mitigar riesgos de los proyectos productivos.
RF 22: Monitorear riesgos de los proyectos productivos.
RF 23: Generar reporte del seguimiento de los riesgos.

Requisitos no Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (15).

Categorías de los Requerimientos no Funcionales.

Existen disímiles categorías para clasificar los requisitos funcionales, a continuación se mencionan aquellos que son imprescindibles para el desarrollo del sistema a realizar:

- Requerimientos de Software.

Para la PC de Aplicación: Se debe disponer de sistemas operativos GNU/Linux, Windows XP (Servipack 3) o superior para la instalación de la aplicación. Debe tenerse instalado el navegador web Mozilla/Firefox versión 3.0 o superior.

Para la PC servidora: Se debe disponer de una PC con sistema operativo GNU/Linux o Windows XP (Servipack 3) o una versión superior para soportar la base de datos de la aplicación y debe tener instalado el PostgreSQL (versión 8.2 o superior) y EMS_PostgreSQL_Manager, servidor web Wamp 5 versión 1.7.4 o superior y framework de desarrollo Symfony en su versión 1.2.7. Debe tenerse instalado el navegador web Mozilla/Firefox versión 3.0 o superior.

- Requerimientos de Hardware.

Para la PC cliente debe tenerse prestaciones mínimas de 256 de RAM, 1.7 Hz de velocidad y tener tarjeta de red.

Para la PC servidora debe tenerse prestaciones mínimas de 512 RAM, 2.4 Hz de velocidad y 6 GB disco duro y tener tarjeta de red.

- Requerimientos de apariencia o interfaz externa.

La interfaz del sistema será a través de una aplicación Web la cual debe contar con una interfaz amigable, facilidades de uso, que permita al usuario una navegación sugerente, con un diseño que sea agradable a la vista de los usuarios. Resaltar contenidos o secciones de mayor importancia al usuario, de forma intencional permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.

- Requerimientos de Seguridad.

Los mecanismos de seguridad están soportados por los servicios de autenticación y autorización de acuerdo a los permisos asignados permitiendo así la confidencialidad, integridad y disponibilidad del sistema.

- Requerimientos de Usabilidad.

La aplicación está orientada a usuarios que tengan al menos un mínimo de conocimientos sobre la gestión de riesgos y manejo de un ambiente web. El sistema debe informar sobre los riesgos que afecten el funcionamiento del mismo, como: los servicios de red, caída del servicio del servidor de datos, entre otras, le ofrecerá al usuario la posibilidad de realizar reportes para tener conocimiento de la información y analizar los resultados de las mismas. En este sentido se centra el diseño de la aplicación y en específico de las interfaces que harán posible el intercambio de datos de manera que le resulte al usuario de fácil entendimiento.

- Requerimientos de Soporte.

El producto debe recibir mantenimiento ante cualquier fallo que ocurra. Este mantenimiento deberá ser realizado por la entidad que lo realice o personal capacitado por la misma entidad. Este es de fácil instalación. Terminada su realización debe será sometida a pruebas con el objetivo de comprobar su funcionalidad.

- Requerimientos de Portabilidad.

La herramienta propuesta podrá ser usada bajo cualquier sistema operativo GNU/Linux o Windows XP (Servipack 3) o una versión superior, para su implementación se usaron Herramientas de Programación (NetBeans 6.8 como entorno de desarrollo y Symfony versión 1.2.7 como framework de desarrollo), Servidor Web Wamp5 versión 1.7.4 y Sistema Gestor de Bases de Datos PostgreSQL versión 8.2, las cuales son herramientas multiplataforma.

- Requerimientos de Rendimiento.

Debido a la importancia de la información que procesamos en nuestra aplicación, el sistema deberá de ser lo más estable y confiable posible. La respuesta a las solicitudes de los usuarios del sistema debe ser en un período de tiempo relativamente breve, para evitar la acumulación de trabajo por parte de los responsables. El sistema deberá de ser lo más estable y confiable posible. La actualización de los datos obtenidos como resultado de la consulta a los elementos activos de la red, debe suceder en rangos de tiempo de muy cortos (25 a 35 segundos).

- Requerimientos Legales.

El sistema debe estar acorde con las políticas preestablecidas en la universidad.

- Requerimientos de confiabilidad.

El sistema debe tener una garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario para evitar entradas inadecuadas. Este debe estar disponible las 24 horas del día. La respuesta de mantenimiento y soporte técnico deben ser dadas con la mayor brevedad posible.

2.2.3 Modelo de Casos de Uso del Sistema.

Actores del sistema.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema (15).

Tabla 5: Actores del Sistema.

Actores	Descripción
Administrador de Riesgos	Persona encargada de identificar los riesgos, registrar los riesgos, además de analizarlos, priorizarlos y planificarlos en el proyecto.
Revisor de Riesgos	Persona encargada de realizar el monitoreo/control y reporte de los riesgos en el proyecto.
Administrador de Usuarios	Persona encargada de realizar la gestión de los usuarios que accedan al sistema, dándole los privilegios a cada uno.
Usuario	Actor genérico que puede realizar las funcionalidades de los actores que heredan de él, además de ser el encargado de realizar la entrada al sistema.
Gestor de Riesgos	Actor genérico que puede realizar las funcionalidades del administrador de riesgos y el revisor de riesgos.

Diagrama de Casos de Uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores (10).

Para una mejor precisión del diagrama de casos de usos del sistema se llevó a cabo el uso de los patrones de casos de usos, para concretar los requisitos funcionales de la aplicación proporcionando un

mejor entendimiento de la solución del sistema, utilizando concordancia, múltiples actores- roles comunes y CRUD completo (Creating, Reading, Updating, Deleting), estos patrones están enfocados al diseño y modelos de alta calidad; la subsecuencia de acciones en el caso de uso gestionar riesgos de proyecto que incluye identificar riesgos en los proyectos, analizar riesgos en los proyectos, priorizar riesgos de los proyectos, planificar riesgos de los proyectos, mitigar riesgos de los proyectos, monitorear riesgos de los proyectos y generar reporte del seguimiento de los riesgos; la creación, lectura, actualización y eliminación (CRUD Completo) en la modelación de la información de los riesgos y los proyectos y además de los actores administrador de usuarios y gestor de riesgos, cumpliendo el rol de usuario, y a su vez el administrador de riesgos y el revisor de riesgos realizando el rol de gestor de riesgos.

A continuación se hace referencia a la relación que existe entre los requisitos funcionales con los casos de uso del sistema identificados; CU gestionar_proyecto agrupa los RF: 1, 2, 3 y 4, CU gestionar_riesgos con los RF: 7, 8, 9 y 10, CU generar_lista_riesgos con el RF: 5, CU generar_plan_contingencia con el RF: 16, CU gestionar_usuario con los RF: 12, 13, 14 y 15 CU generar_plan_mitigación con el RF: 6, CU autenticar_usuario con el RF: 11, CU gestionar_riesgos_proyectos con los RF: 17, 18, 19, 20, 21, 22 y 23, CU identificar_riesgos_proyecto con el RF 17, CU analizar_riesgos_proyecto con el RF 18, CU priorizar_riesgos_proyecto con el RF 19, CU planificar_riesgos_proyecto con el RF 20, CU mitigar_riesgos_proyecto con el RF 21, CU monitorear_riesgos_proyecto con el RF 22 y CU generar_reportes_seguimiento_riesgos_proyecto con el RF 23.

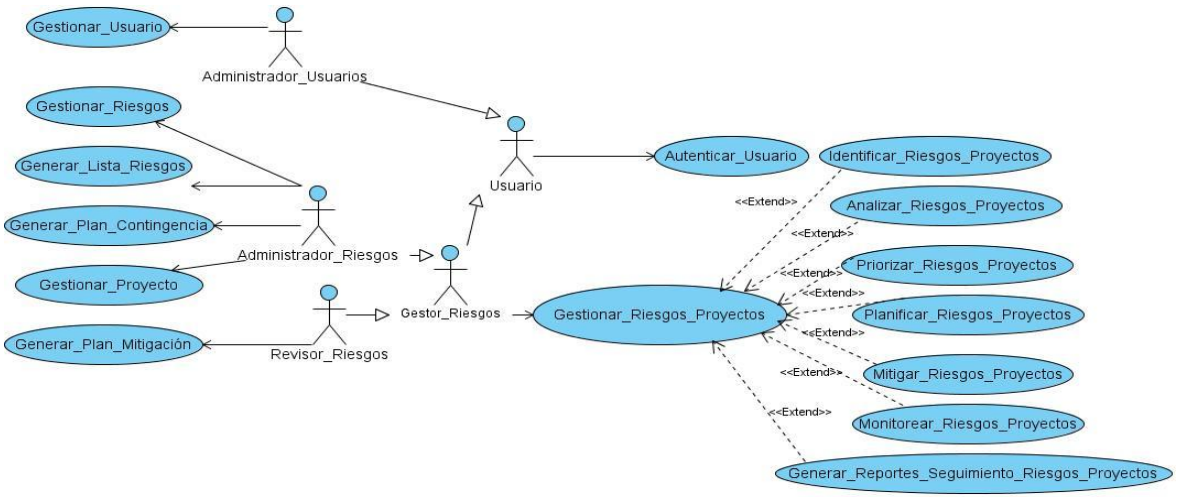


Fig. 4: Diagrama de Casos de Uso del Sistema.

A continuación se realiza la descripción de unos de los casos de uso del sistema. En el anexo 2 se pueden observar la demás descripciones referentes al caso de uso gestionar riesgos del proyecto. (Ver Anexo 2).

Tabla 6: Descripción del Caso de Uso del Sistema "Gestionar Riesgos del Proyecto".

Caso de Uso:	Gestionar Riesgos del Proyecto
Actores:	Administrador_Riesgos, Revisor_Riesgos
Resumen:	El caso de uso se inicia cuando se decide realizar la gestión de riesgos del proyecto, terminando con la devolución de los reportes, referente al comportamiento de los riesgos en el proyecto.
Referencias	RF 17, 18, 19, 20, 21, 22 y 23
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El caso de uso comienza cuando el usuario selecciona la opción gestionar riesgos del proyecto y las opciones para la gestión de riesgos, de acuerdo al rol que cumpla (administrador de riesgos o revisor de riesgos): a. Identificación. b. Análisis. c. Priorización. d. Planificación. e. Mitigación. f. Monitoreo. g. Reportes.	El sistema muestra la interfaz para cualquiera de las siguientes opciones: a. Si el administrador de riesgos desea identificar los riesgos del proyecto, selecciona la opción: Identificación y va al caso de uso Identificar riesgos del Proyecto. b. Si el administrador de riesgos desea analizar los riesgos del proyecto, selecciona la opción: Análisis y va al caso de uso Analizar riesgos del Proyecto. c. Si el administrador de riesgos desea priorizar los riesgos del proyecto, selecciona la opción: Priorización y va al caso de uso Priorizar riesgos del Proyecto. d. Si el administrador de riesgos desea planificar los riesgos del proyecto, selecciona la opción: Planificación y va al caso de uso Planificar los riesgos del Proyecto. e. Si el revisor de riesgos desea mitigar los riesgos del proyecto, selecciona la opción: Mitigación y va al caso

	de uso Mitigar los riesgos del Proyecto. f. Si el revisor de riesgos desea monitorear los riesgos del proyecto, selecciona la opción: Monitoreo y va al caso de uso Monitoreo de los riesgos del Proyecto. g. Si el revisor de riesgos desea realizar reportes de los riesgos del proyecto, selecciona la opción: Reportes de los riesgos del Proyecto y va al caso de uso Generar Reportes de los riesgos del proyecto. h. El caso de uso finaliza cuando se realizan todas las opciones anteriormente descritas.

CONCLUSIONES DEL CAPÍTULO

Durante la modelación del negocio se identifican dentro del proceso Gestionar Riesgos, diez actividades posibles a automatizar: identificar, analizar, priorizar, planificar, monitorear y reportar comportamiento de los riesgos. Dentro de estas actividades se precisan 23 requisitos funcionales que fueron agrupados en 15 casos de uso. Para el mejor entendimiento del diagrama de casos de uso del sistema se utilizan los patrones de casos de uso: CRUD en los casos de uso del sistema gestionar proyectos, usuarios, riesgos y gestionar riesgos del proyecto, y múltiples actores - rol común en el caso de uso autenticar usuario.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

INTRODUCCIÓN

En este capítulo se presentan los artefactos correspondientes al flujo de Trabajo de Análisis y Diseño según la metodología RUP, teniendo en cuenta el lenguaje de programación, framework de desarrollo y los patrones de arquitectura y diseño empleados. Se realiza el modelado de los diagramas de clases del análisis y diseño y los diagramas de interacción mediante los diagrama de secuencias para el análisis y colaboración para el diseño. El modelo lógico y físico de los datos, y la disposición física de los dispositivos, mediante el diagrama de despliegue y el Modelo Vista Controlador utilizado en él desarrollo del sistema; además de la seguridad y tratamiento de errores del sistema.

3.1 Análisis y Diseño.

En este epígrafe se realiza una traducción de los requerimientos del sistema a una especificación de cómo implementar el sistema, transformando los requerimientos al diseño del futuro sistema, desarrollar la arquitectura candidata para el sistema, para lograr un diseño consistente con el entorno de implementación, para el rendimiento de la aplicación.

3.1.1 Modelo de Análisis del sistema.

En el análisis del sistema se realiza el modelado de las clases del análisis en función de los requerimientos funcionales, especificando el comportamiento funcional del sistema, independientemente de los aspectos relativos al ambiente en el que va a ser finalmente implementado.

Diagrama de Clases del Análisis.

Los diagramas de clases del análisis son una representación de la interacción entre las clases interfaz, controladora y entidad definida a partir de los requisitos funcionales identificados en el sistema.

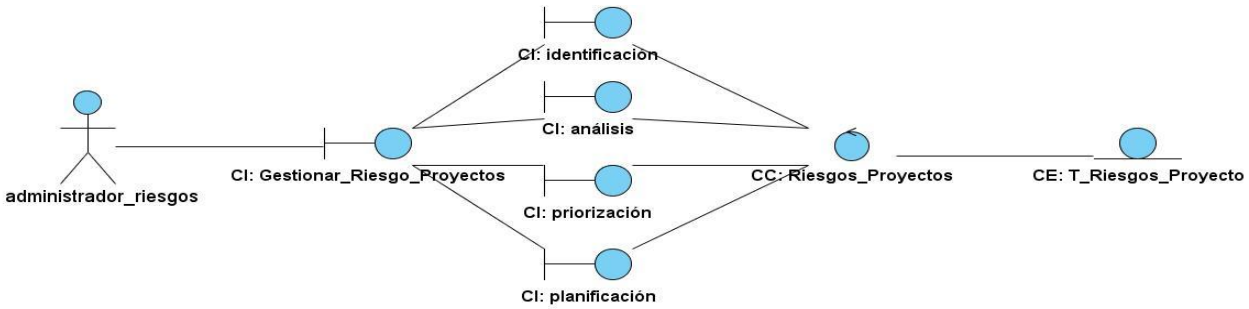


Fig. 5: Diagrama de Clases del Análisis Gestionar Riesgos de Proyectos (I).

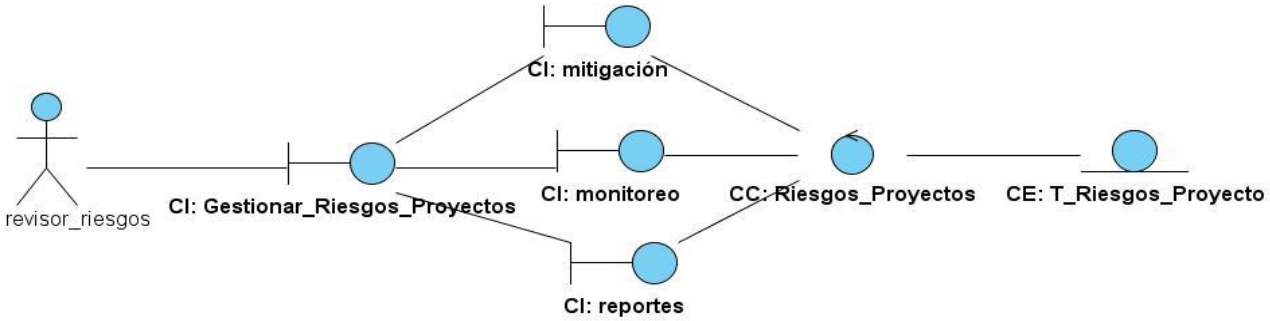


Fig. 6: Diagrama de Clases del Análisis Gestionar Riesgos de Proyectos (II).

3.1.2 Diseño del sistema.

El modelo de diseño es una descripción profunda de los requisitos no funcionales. El modelo análisis es un paso de entrada al diseño que se realiza para lograr su refinamiento, y orientar al desarrollador como la aplicación debe ser realizada en la implementación, el diseño es crear una entrada apropiada y un punto de partida para la implementación.

Diagrama de Clases de Diseño.

Los diagramas de clases del diseño son una representación de la solución del sistema, especificando la visibilidad de los atributos y operaciones a implementarse a partir del lenguaje de programación y framework de desarrollo, existiendo una correspondencia entre el diseño y la implementación del sistema.

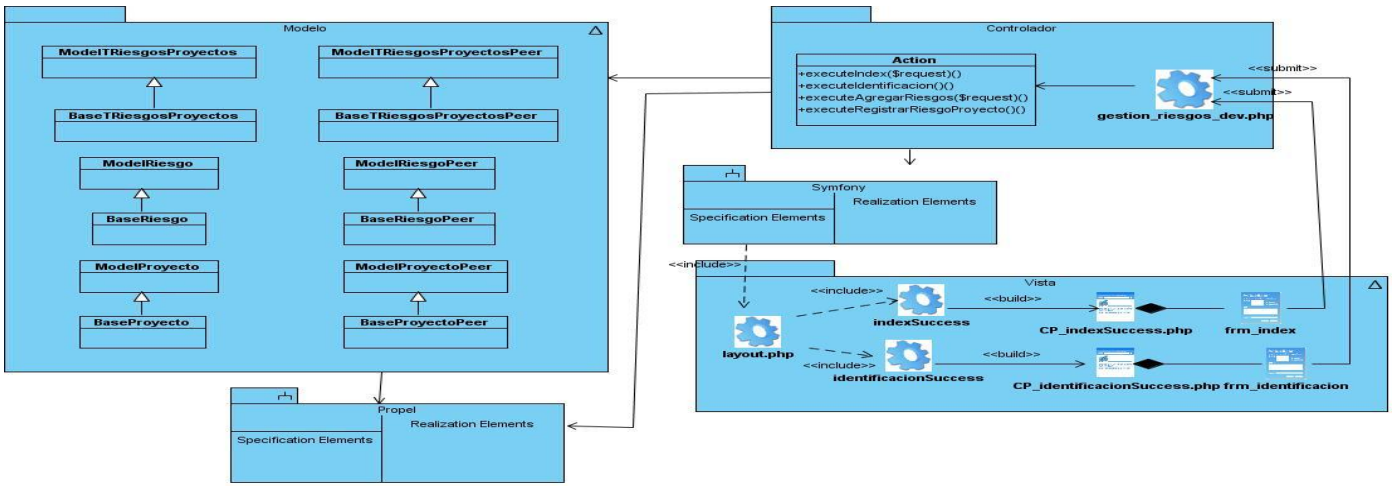


Fig. 7: Diagrama de Clases del Diseño Identificar Riesgos del Proyecto.

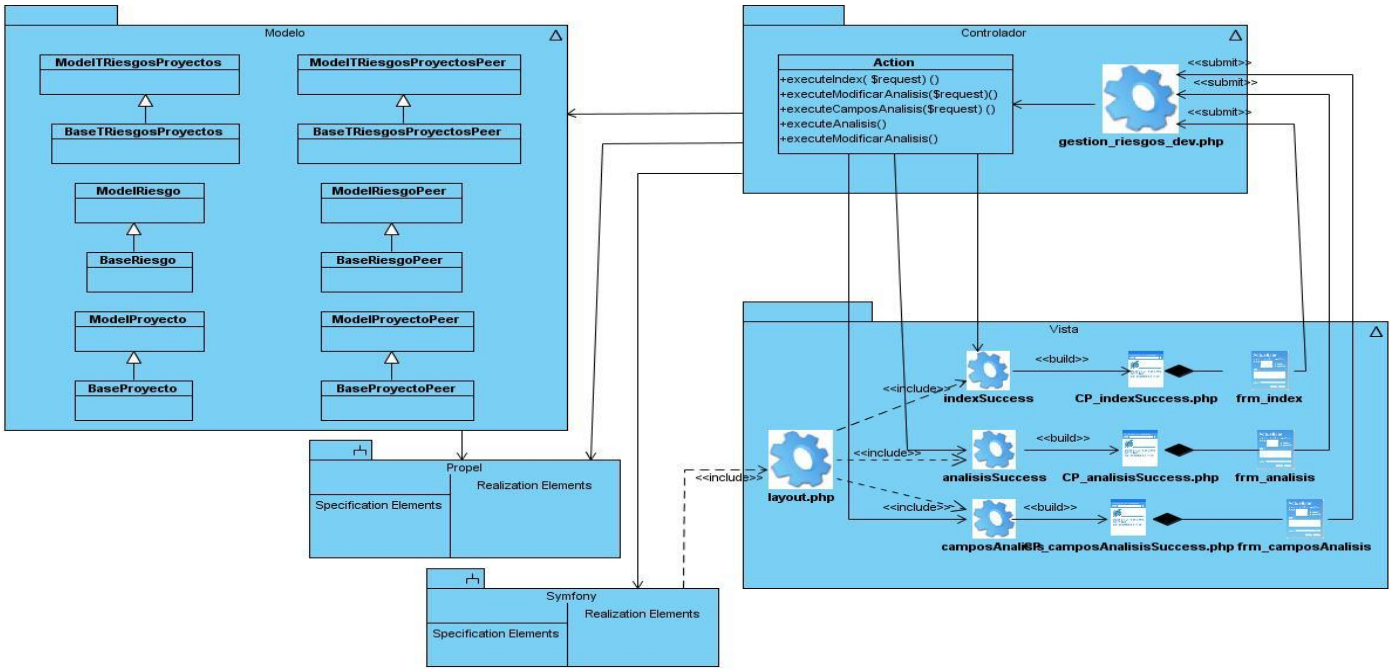


Fig. 8: Diagrama de Clases del Diseño Análisis Riesgos del Proyecto.

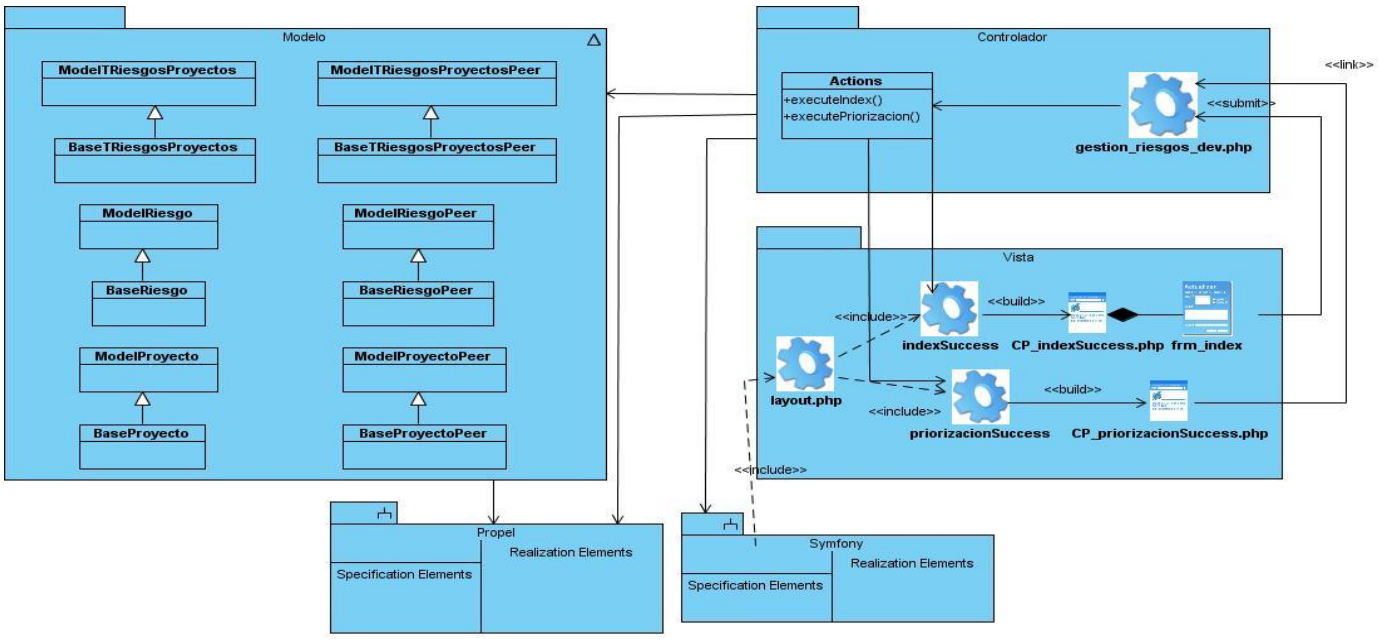


Fig. 9: Diagrama de Clases del Diseño Priorización Riesgos del Proyecto.

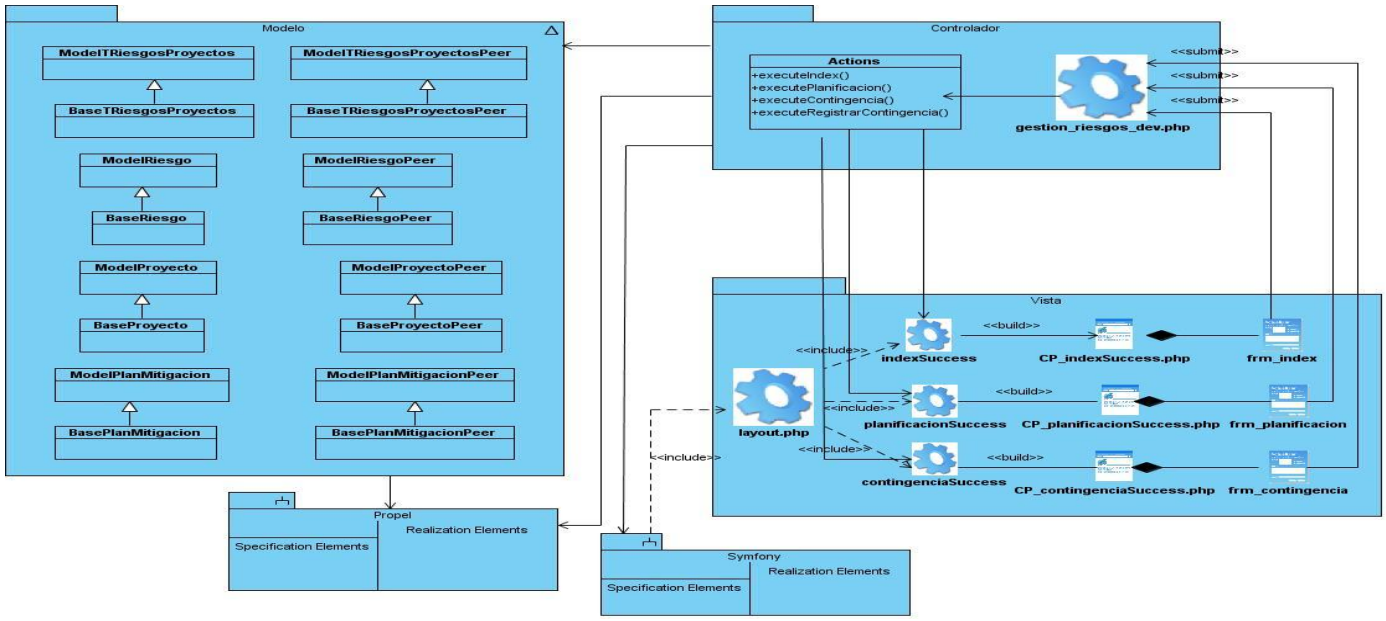


Fig. 10: Diagrama de Clases del Diseño Planificación Riesgos del Proyecto.

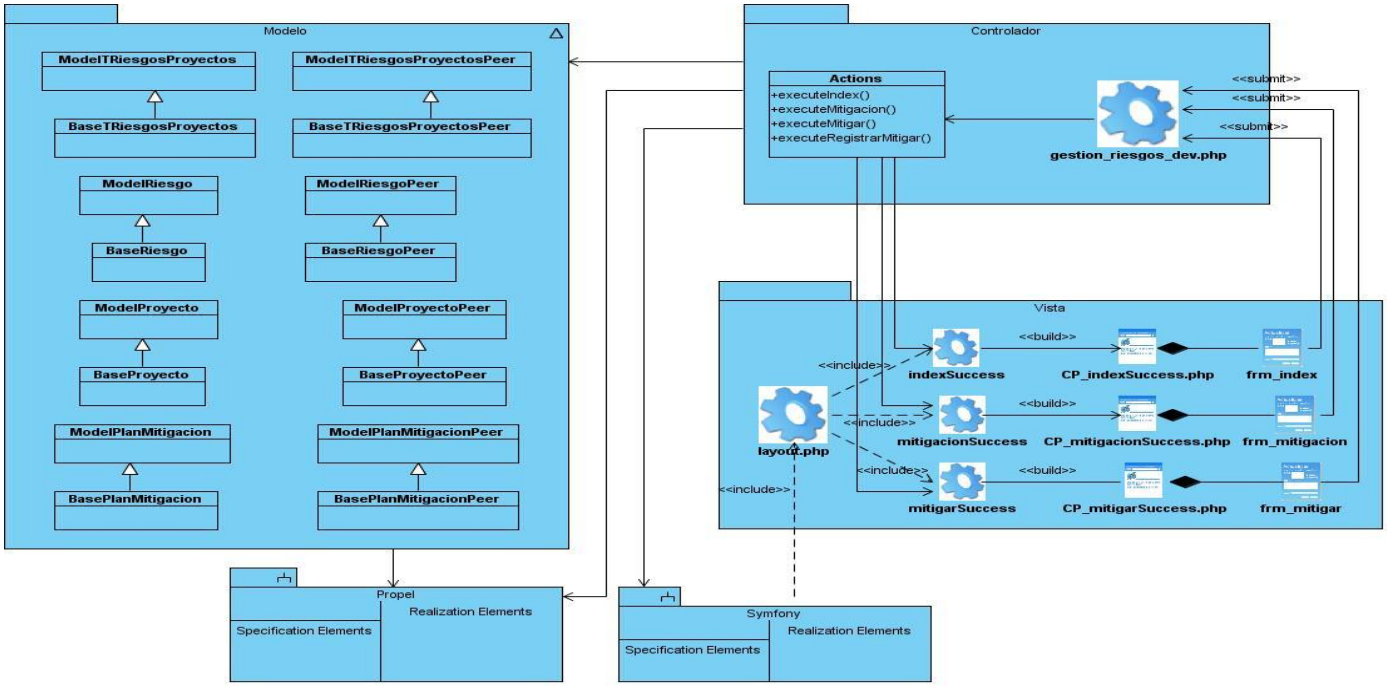


Fig. 11: Diagrama de Clases del Diseño Mitigación Riesgos del Proyecto.

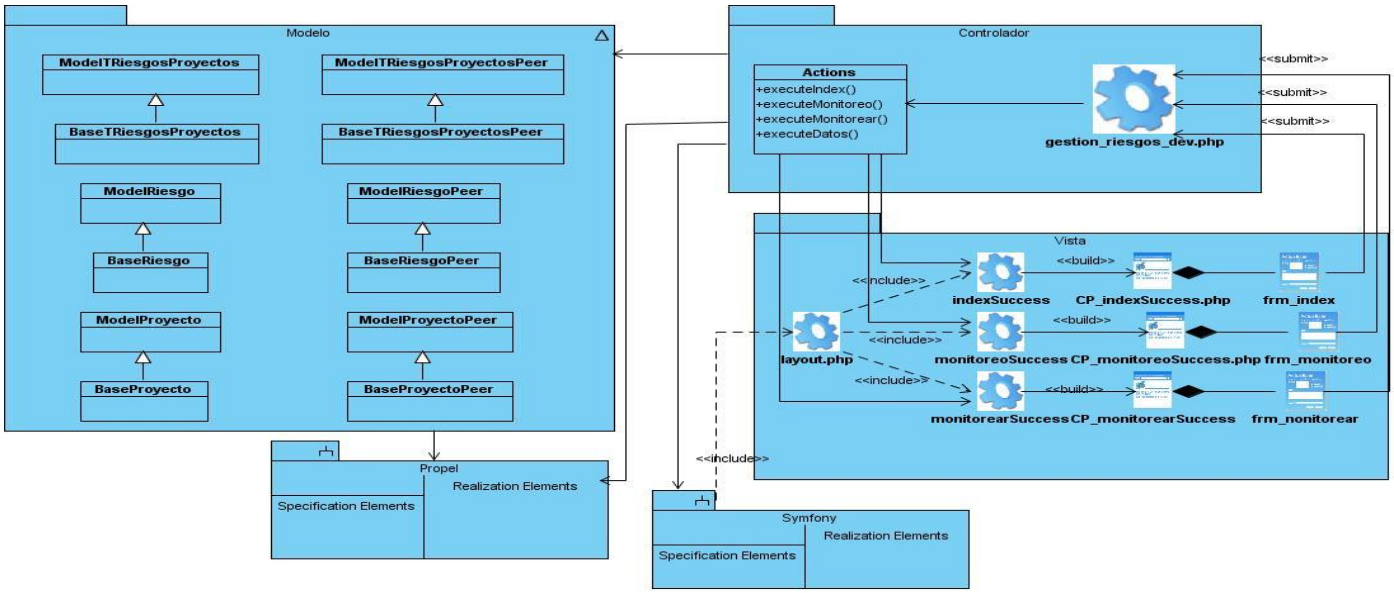


Fig. 12: Diagrama de Clases del Diseño Monitoreo Riesgos del Proyecto.

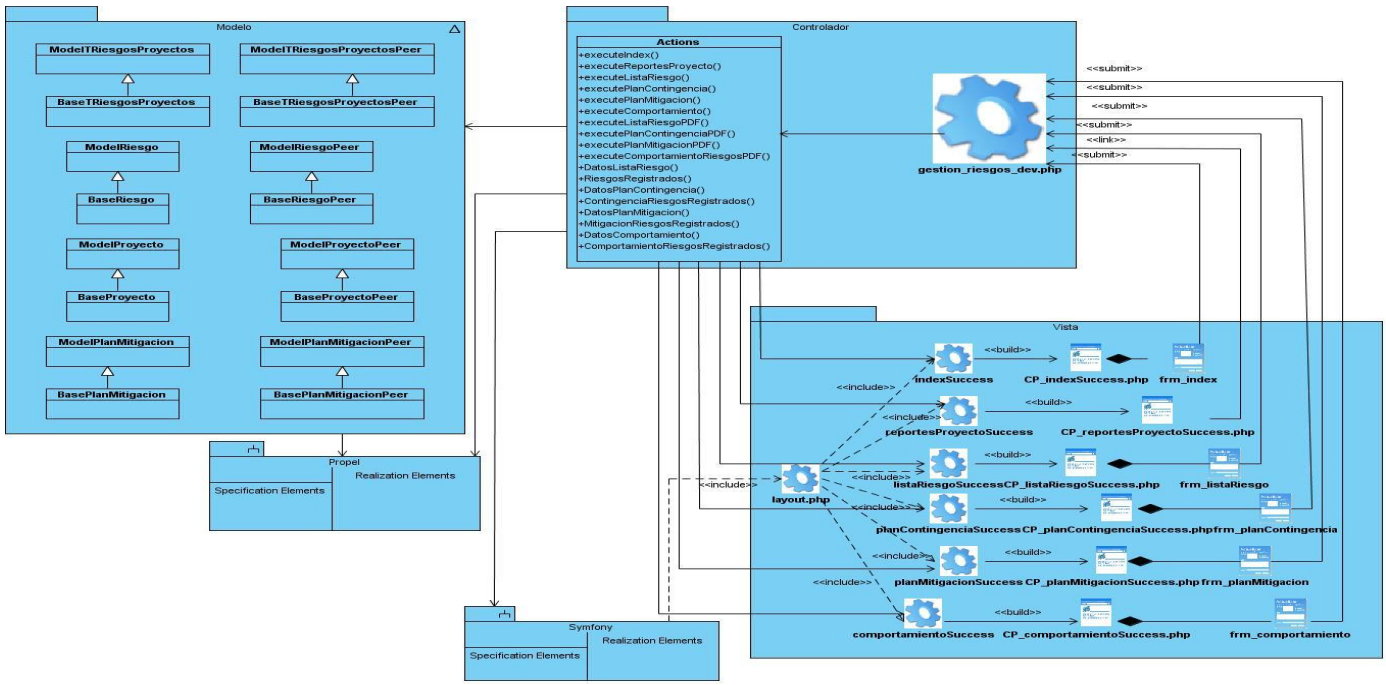


Fig. 13: Diagrama de Clases del Diseño Reportes Riesgos del Proyecto.

3.1.3 Diagramas de Interacción.

Los diagramas de interacción es la forma de representar los aspectos dinámicos del sistema, lo que trae consigo modelar instancias concretas de las interfaces, los componentes y nodos, junto con los mensajes enviados entre ellos; todo el contexto de un escenario que ilustra un comportamiento. Estos diagramas se expresan en diagramas de colaboración; muestran las relaciones entre los objetos y los mensajes que intercambian y, en diagramas de secuencia, que ilustran las interacciones expresadas en función de secuencias temporales.

Diagramas de Colaboración.

Los diagramas de colaboración describen cómo trabajan en común instancias específicas de las clases para lograr un objetivo en común, mediante una secuencia de mensajes que se intercambian entre un objeto y otro.

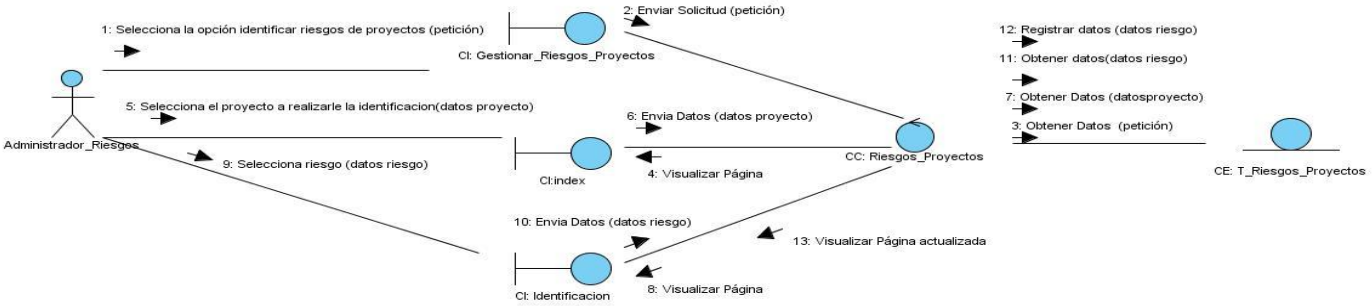


Fig. 14: Diagrama de Colaboración Identificar Riesgos del Proyecto.

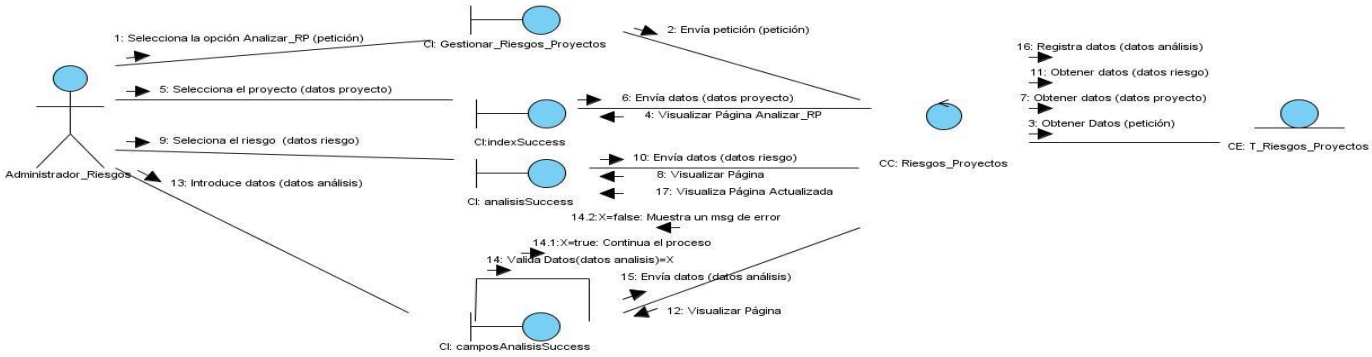


Fig. 15: Diagrama de Colaboración Analizar Riesgos del Proyecto.

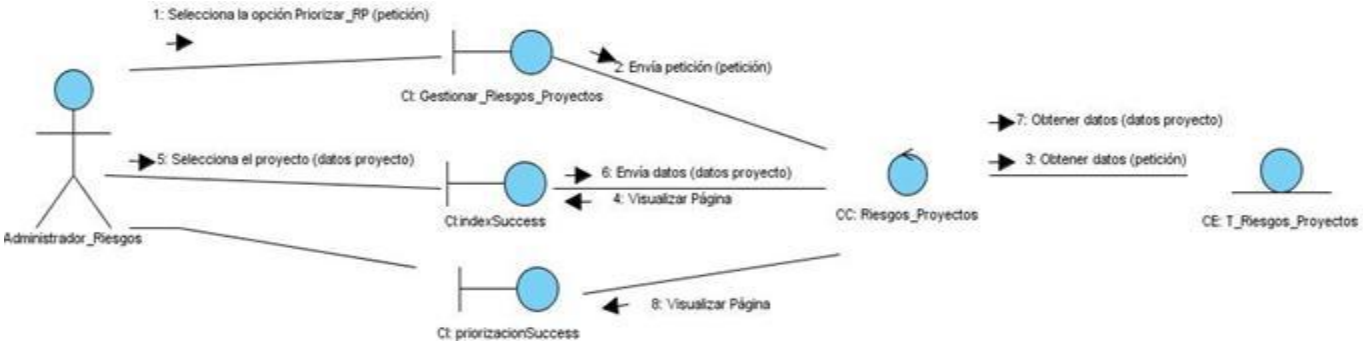


Fig. 16: Diagrama de Colaboración Priorizar Riesgos del Proyecto.

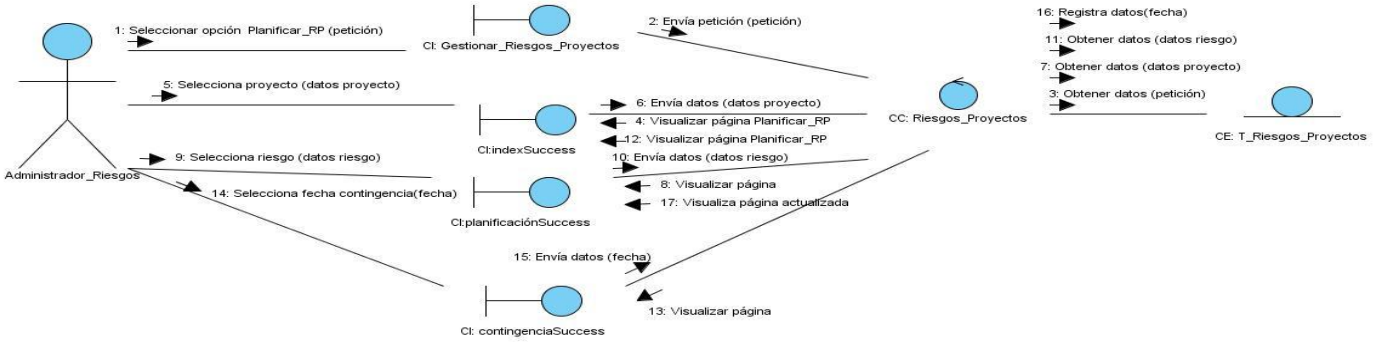


Fig. 17: Diagrama de Colaboración Planificar Riesgos del Proyecto.

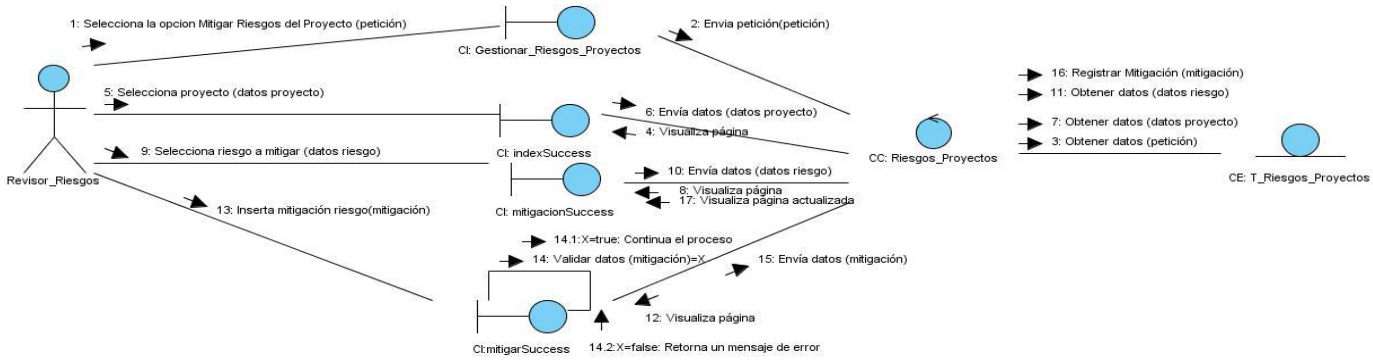


Fig. 18: Diagrama de Colaboración Mitigar Riesgos del Proyecto.

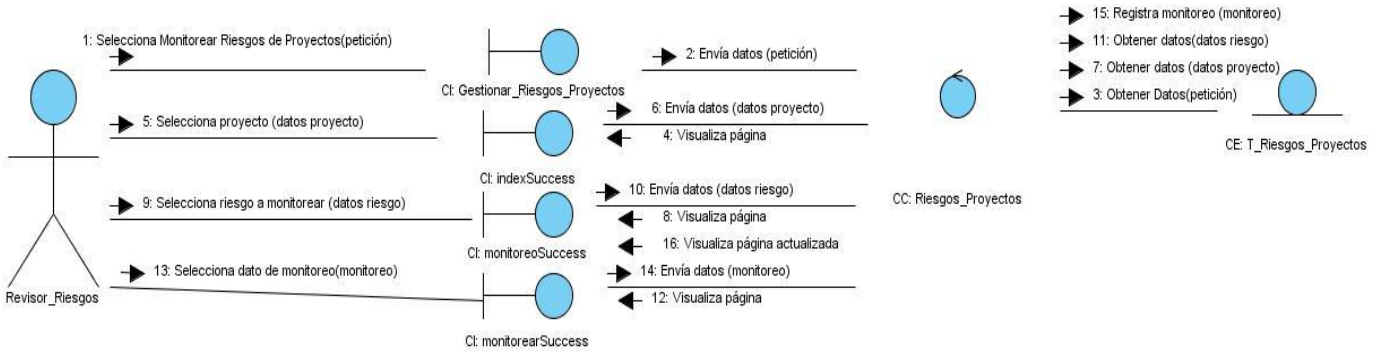


Fig. 19: Diagrama de Colaboración Monitorear Riesgos del Proyecto.

Diagramas de Secuencia.

La secuencia de mensajes intercambiados en un determinado tiempo, expresando restricciones entre los objetos y clases, se define como un diagrama de secuencia.

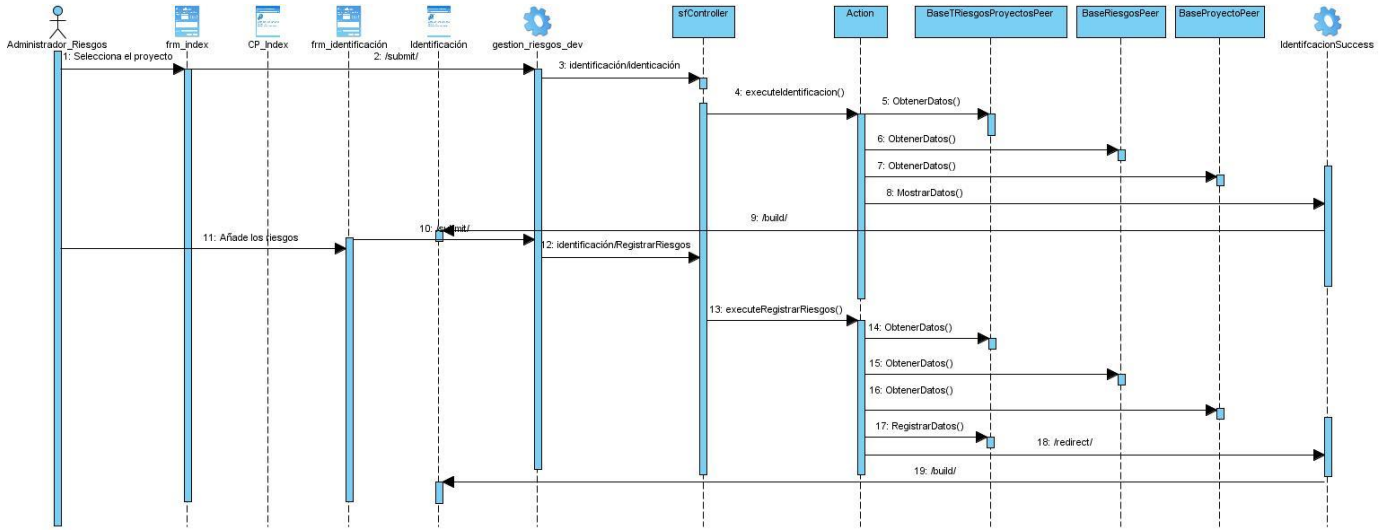


Fig. 20: Diagrama de Secuencia Identificar Riesgos del Proyecto.

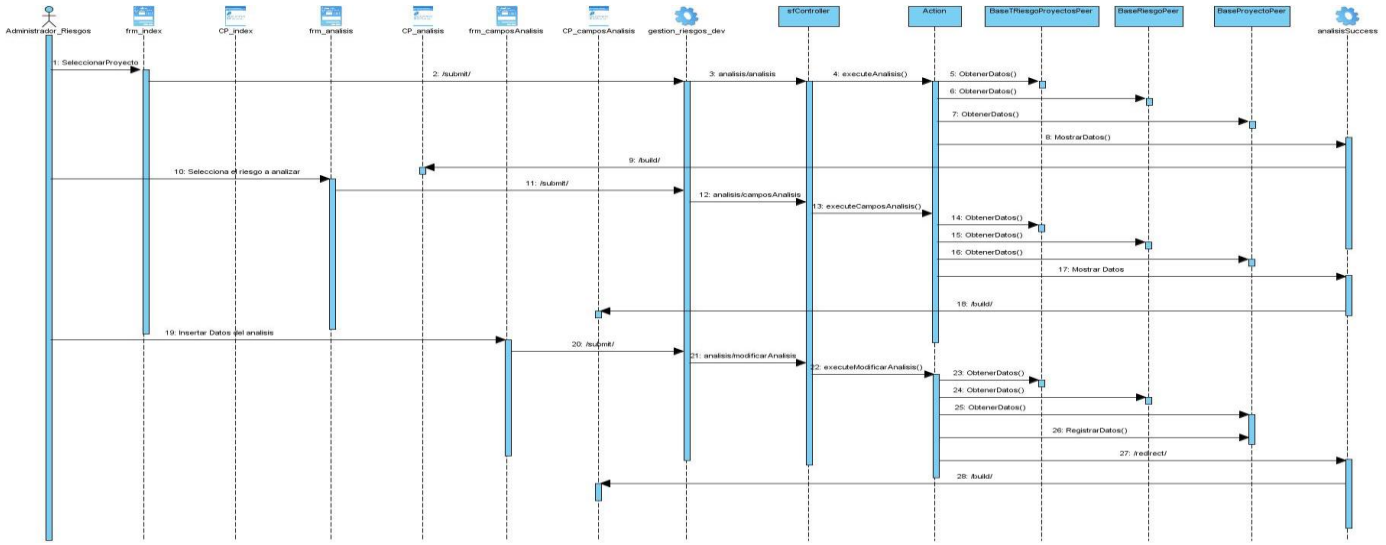


Fig. 21: Diagrama de Secuencia Analizar Riesgos del Proyecto.

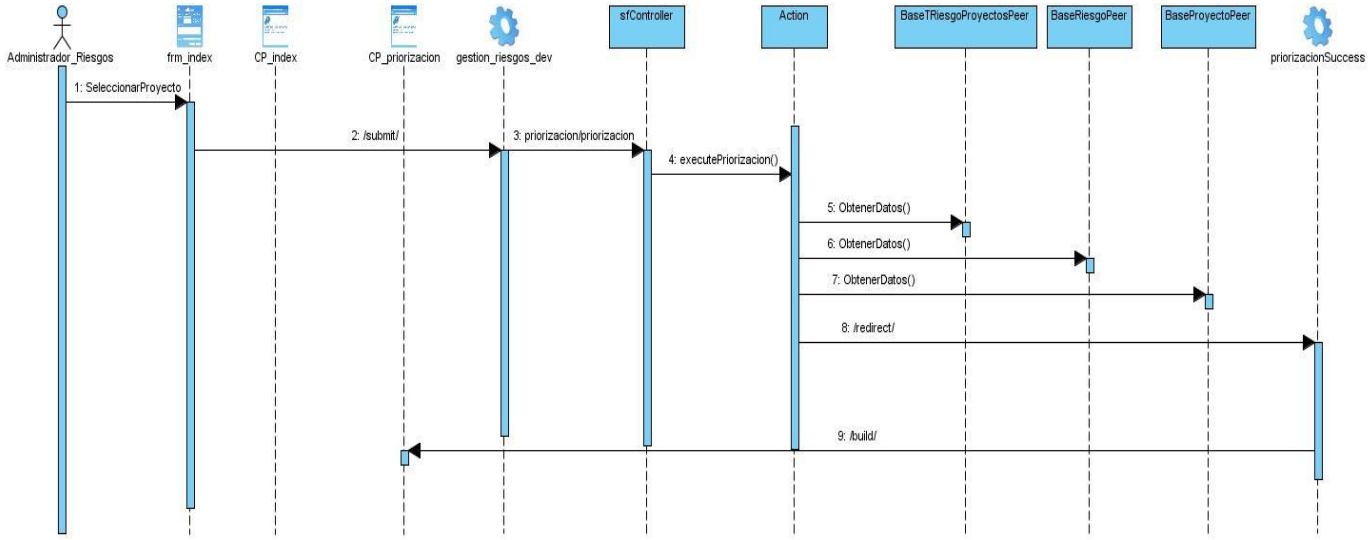


Fig. 22: Diagrama de Secuencia Priorizar Riesgos del Proyecto.

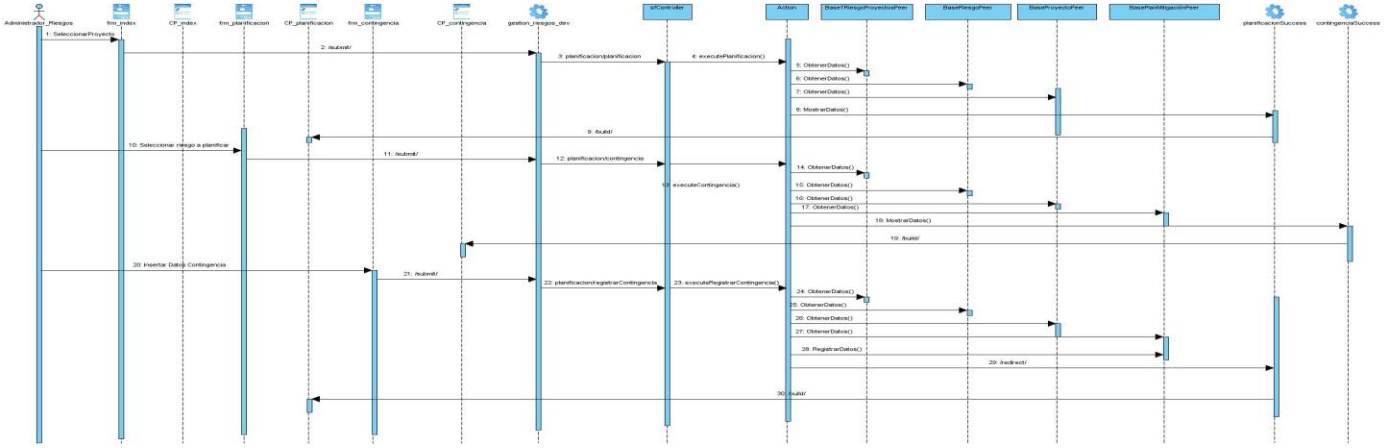


Fig. 23: Diagrama de Secuencia Planificar Riesgos del Proyecto.

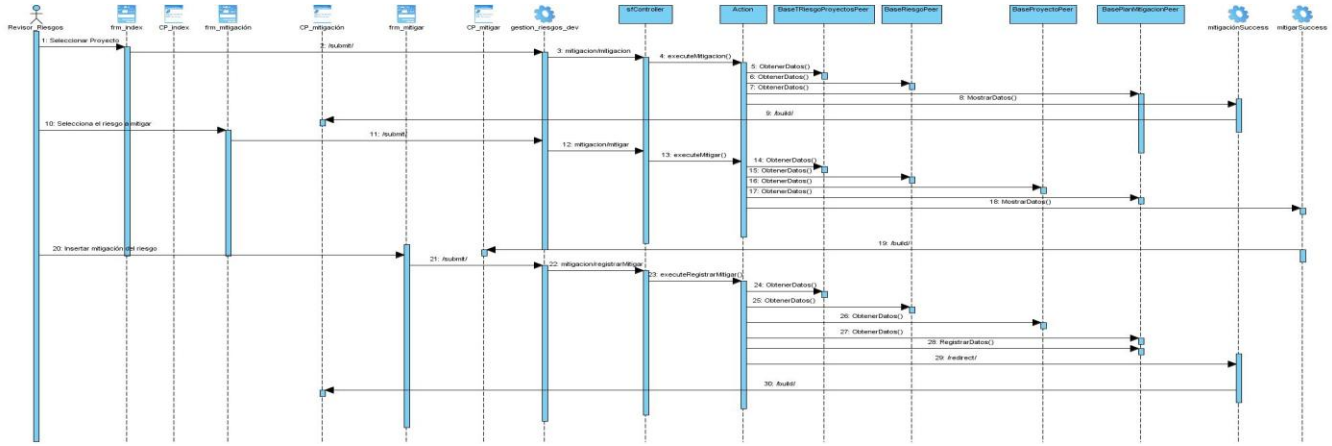


Fig. 24: Diagrama de Secuencia Mitigar Riesgos del Proyecto.

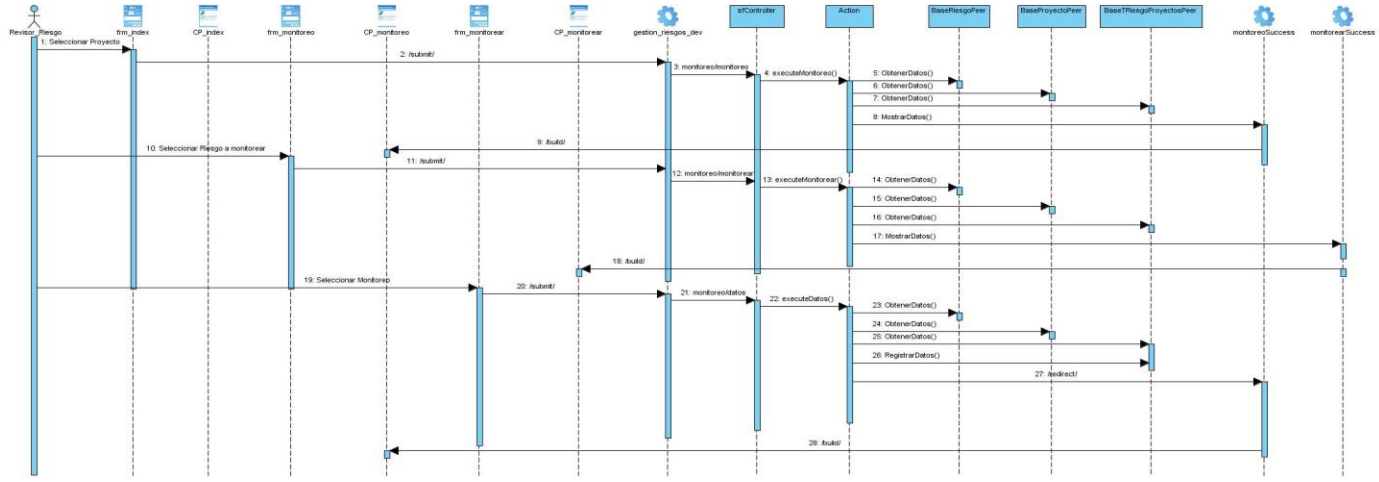


Fig. 25: Diagrama de Secuencia Monitorear Riesgos del Proyecto.

3.2 Patrón arquitectónico. Modelo Vista Controlador del framework Symfony.

Un patrón arquitectónico caracteriza los componentes de acuerdo a la responsabilidad, relación y forma en que colaboran los sistemas informáticos para proporcinarle solución a un problema dentro del desarrollo de software.

La arquitectura del Modelo Vista Controlador (MVC) está determinada por tres niveles:

- Modelo: información con la que trabaja la aplicación (lógica de negocio), haciendo que la vista y las acciones sean independientes.
- Vista: realiza la transformación del modelo de una página web de forma tal que le permite al usuario interactuar con ella (presentación).
- Controlador: es el encargado de procesar la interacción del usuario, aislando al modelo de la vista.

3.2.1. Descripción del MVC del framework Symfony.

Controlador: La capa del controlador en Symfony se encuentra dividido en dos partes: el controlador frontal y las acciones. La única entrada a la aplicación, es a través del controlador frontal, donde se carga la configuración y determina la acción a ejecutarse. La lógica de las páginas ocurre mediante las acciones, realizando la verificación de la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.

Vista: La vista en Symfony está compuesta por las plantillas, en la cual se realiza la presentación de los datos de las acciones que se están ejecutando, y el layout, que contiene el código HTML común a todas las páginas de la aplicación.

Modelo: La capa de tipo Object/Relational Mapping (ORM) es el componente que se encarga de gestionar el modelo en Symfony mediante el proyecto Propel. El acceso y la modificación de los datos almacenados en la base de datos en proyectos desarrollados bajo el framework Symfony se realiza mediante objetos, lo cual permite un alto nivel de abstracción y una fácil portabilidad. Para la abstracción de las base de datos Symfony utiliza el PHP Data Objects (PDO).

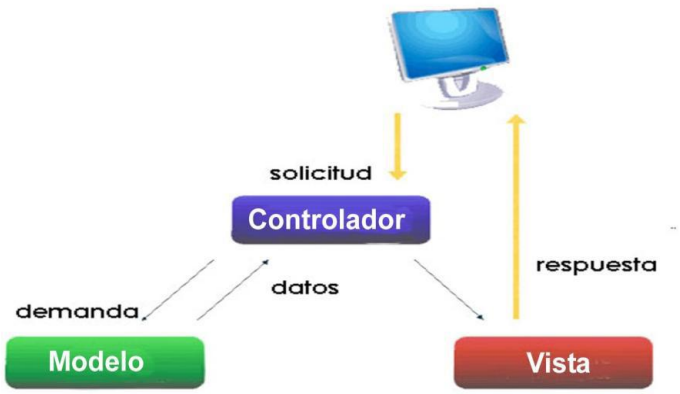


Fig. 26: Patrón MVC de Symfony.

3.3 Diseño de la Base de Datos.

Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido, y recuperar la información, es por ello que su diseño es muy importante en su desarrollo. Al diseñar una base de datos, se refleja la estructura del problema en el mundo real, evita el almacenamiento de información redundante y permite además representar todos los datos esperados, incluso con el paso del tiempo. Para lograr diseñar la Base de Datos se partió de la definición de las clases persistentes, lo cual podemos definir como la capacidad de un objeto de mantener su valor en el espacio y tiempo. Después del refinamiento y clasificación de estas clases y sus atributos se realizó la construcción del diagrama de entidades persistentes.

3.3.1 Modelo lógico de datos.

El modelo lógico de datos representa la información que maneja el sistema, siendo una fuente de información para el modelo físico. A continuación se muestra el modelo lógico de datos diseñado para la base de datos de la aplicación de este trabajo de diploma, el cual no es más que el diagrama de clases persistentes. El diagrama de clases persistentes es un diagrama de clases en el que, como bien indica su nombre, solo aparecen las clases persistentes, de las que hay que expandir detalles estructurales.

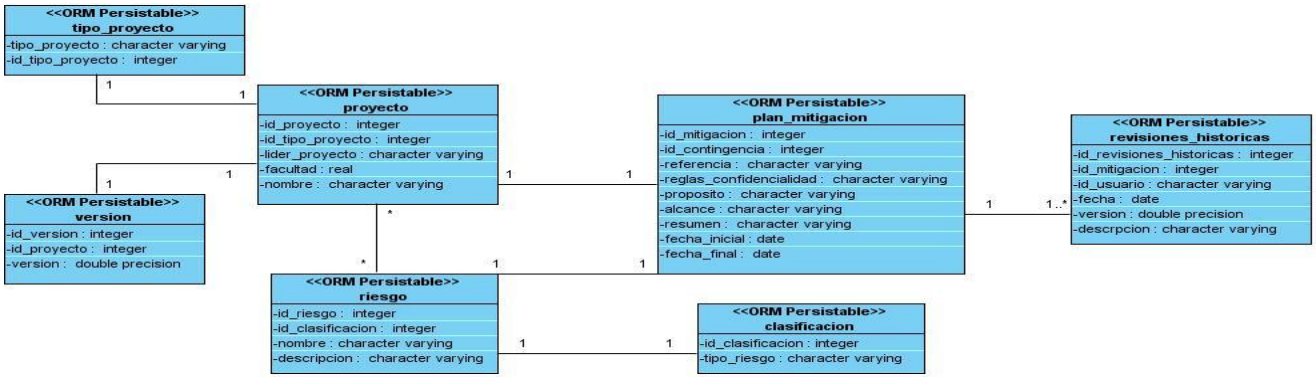


Fig. 27: Diagrama de Clases Persistente.

3.3.2 Modelo físico de datos.

El modelo físico de datos se utiliza para describir cómo los datos se almacenan en un sistema computacional; formatos de registros, estructuras de archivos y métodos de acceso. El diagrama de entidades de las clases persistentes del sistema es el llamado modelo físico de los datos para la aplicación, dando este una descripción detallada de cómo quedarán almacenados los datos.

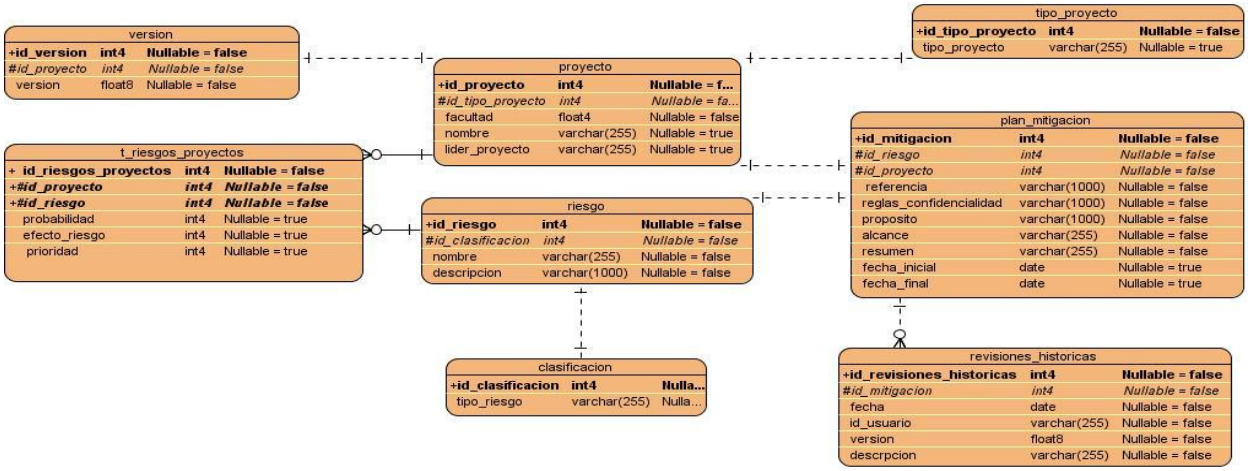


Fig. 28: Diagrama de Entidades de las Clases Persistentes.

3.4 Modelo de despliegue.

El Modelo de Despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

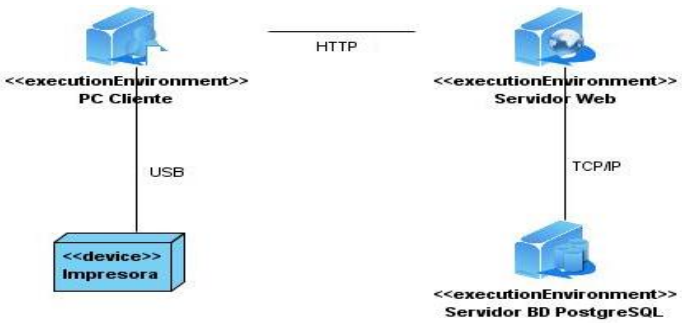


Fig. 29: Modelo de Despliegue.

3.5 Tratamiento de Errores.

En las aplicaciones web es muy usual realizar el tratamiento de errores mediante la validación del lado del cliente o del servidor, o en ambas; lo cual evita que algunos errores que puedan presentarse durante el funcionamiento de la aplicación no ocasionen problema en su rendimiento. La validación de los formularios de la aplicación se desarrolla del lado del cliente con lenguaje javascript para aumentar la experiencia del usuario, y del lado del servidor para no corromper los datos en la base de datos con funcionalidades del framework symfony.

Mediante la interfaz Web se evita que el usuario asuma un papel activo en la introducción de la información, para esto se cuenta con cuadros de opción, menú de selección lo cual facilita la entrada de datos. La información que requiere ser adicionada por el usuario, se validara mediante funciones que garanticen que el cuadro de texto no esté vacío si es obligatorio llenarlo.

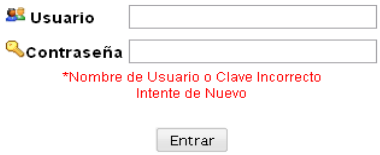


Fig. 30: Ejemplo de Validación.

3.6 Seguridad del Sistema.

La seguridad en Symfony maneja automáticamente las sesiones del usuario, mediante los mecanismos de manejo de sesiones incluido en PHP con una mejora para hacerlo más configurable y fácil de usar. La autenticación de los usuario se controlan a través del controlador frontal mediante filtros de seguridad a través de peticiones; el objeto de sesión de usuarios se accede con el método getUser(), que es una clase

de la clase sfUser, la cual tiene un contendor de parámetros que permite guardar cualquier atributo del usuario en el, estos atributos pueden guardar cualquier tipo de información (cadenas de texto, arreglos y arreglos asociativos).

El manejo de sesiones de Symfony se encarga de gestionar automáticamente el almacenamiento de los IDs de sesión tanto en el cliente como en el servidor. Sin embargo, si se necesita modificar este comportamiento por defecto, es posible hacerlo. Se trata de algo que solamente lo necesitan los usuarios más avanzados. En el lado del cliente, las sesiones son manejadas por cookies. Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

- Las acciones seguras requieren que los usuarios estén autenticados.
- Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

3.7 Ayuda del Sistema.

La ayuda del sistema se realiza con el objetivo de brindarle al usuario una breve descripción del funcionamiento de la aplicación, dando a conocer datos que deben ingresarse en cada uno de los pasos del proceso de gestión de riesgos, para un mayor entendimiento por parte de los usuarios principiantes en el trabajo con el sistema.

CONCLUSIONES DEL CAPÍTULO

El uso del patrón MVC del framework Symfony, al igual que la aplicación de patrones de diseño, ha hecho más fácil el diseño de las clases permitiendo que se obtengan clases mejor diseñadas, modulares, flexibles y reutilizables. A partir del diseño de clases persistentes y la estructuración del diagrama de clases persistentes se obtuvo el modelo de datos, lo que posibilita el diseño de las tablas de la base de datos. El modelado del diagrama de despliegue propicia una visión de como se encuentra organizado el sistema en cuanto a la disposición física de los nodos. El empleo de funciones del framework Symfony facilita el tratamiento de errores a través de las validaciones de formulario.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA.

INTRODUCCIÓN

En este capítulo se aborda el flujo de trabajo de Implementación, donde se describen sus principales artefactos. En este flujo de trabajo se implementa el sistema en términos de componentes a partir de las clases del diseño. Se hace una prueba de lo que se ha implementado; probando los componentes por separado antes de integrarlos al sistema para que funcione como un todo.

4.1 Diagrama de componentes.

Un componente es el empaquetamiento físico de los elementos de un modelo, como es las clases en el modelo de diseño. El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros.

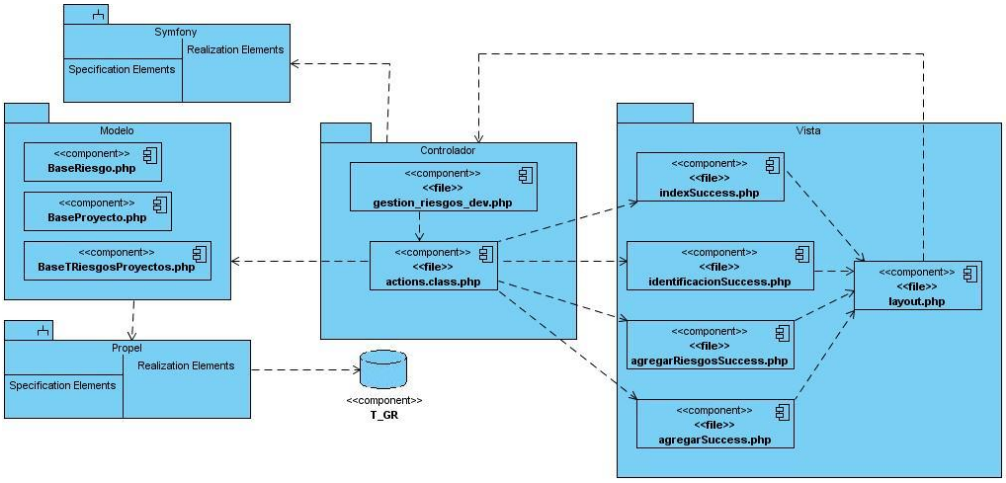


Fig. 31: Diagrama de Componentes Identificar Riesgos del Proyecto.

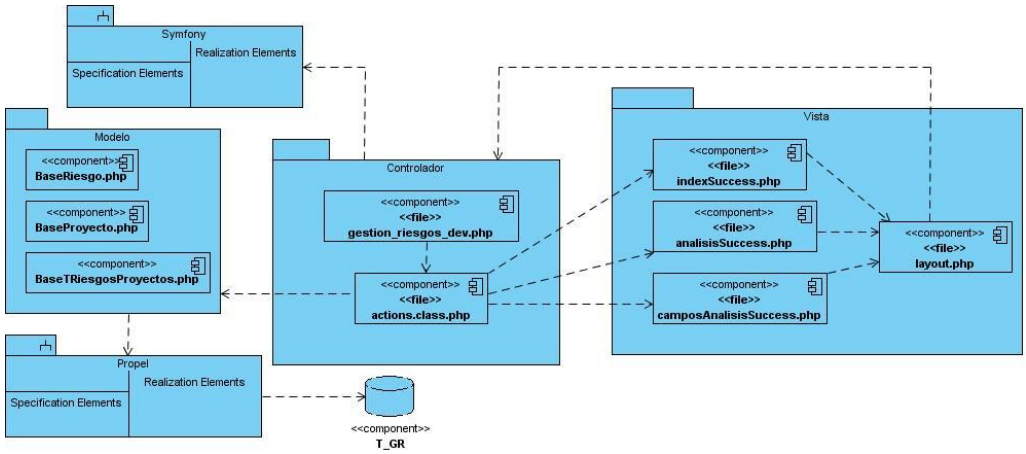


Fig. 32: Diagrama de Componentes Analizar Riesgos del Proyecto.

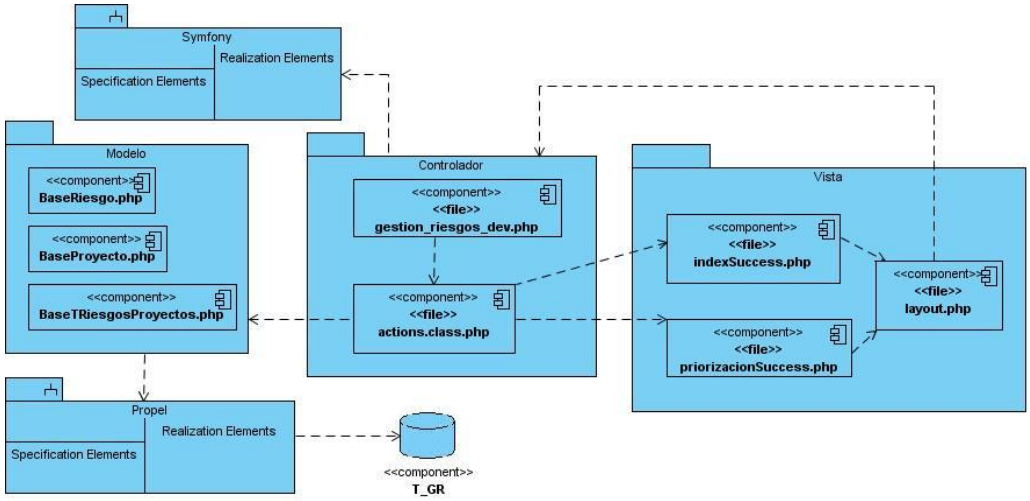


Fig. 33: Diagrama de Componentes Priorizar Riesgos del Proyecto.

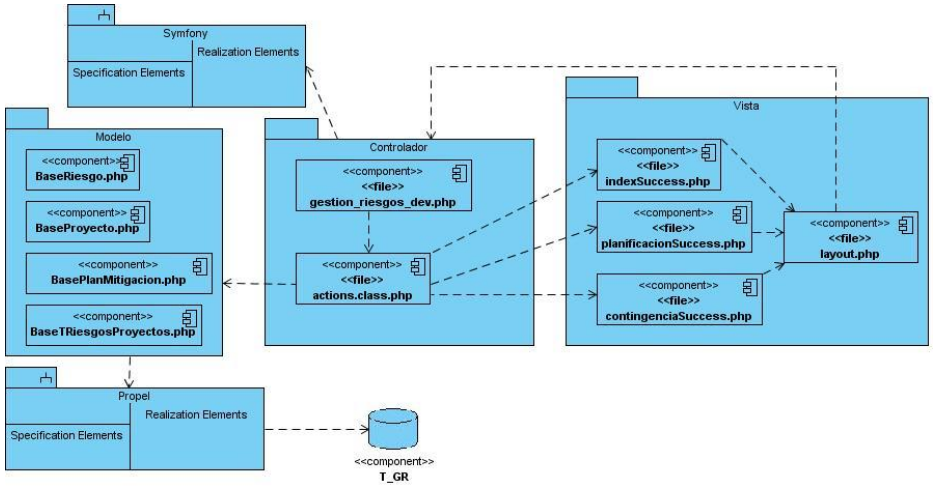


Fig. 34: Diagrama de Componentes Planificar Riesgos del Proyecto.

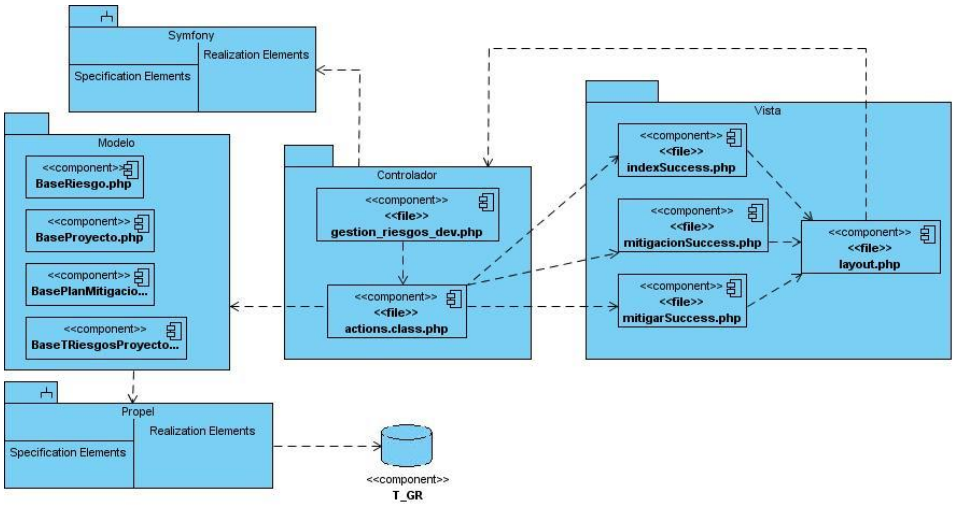


Fig. 35: Diagrama de Componentes Mitigar Riesgos del Proyecto.

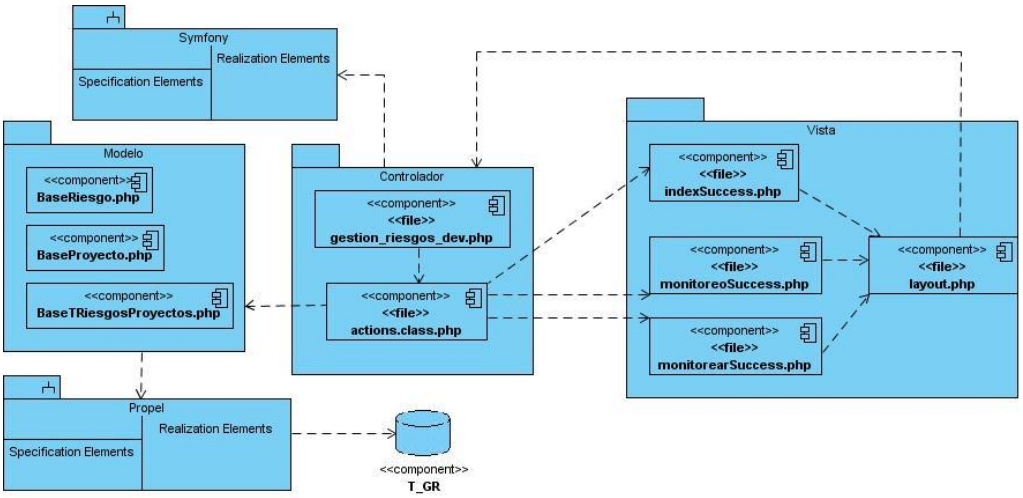


Fig. 36: Diagrama de Componentes Monitorear Riesgos del Proyecto.

4.2 Pruebas de Software.

El desarrollo de software implica una serie de actividades de producción en las que las posibilidades de que surjan errores pueden empezar a darse desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea e imperfecta en cada una de las fases; lo que hace que el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad. El flujo de trabajo definido por RUP, Pruebas, garantiza que las actividades en la cual un sistema o componente de este es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, a partir de la cual se realiza una evaluación hecha algún aspecto del sistema.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación (16).

4.4.1 Estrategias de Prueba.

Niveles de Prueba.

De acuerdo a los objetivos trazados y escenarios de trabajo identificados para el desarrollo de la herramienta para la gestión de riesgos de la facultad 6, se lleva a cabo los siguientes niveles de prueba: Prueba de Integración y Prueba de Sistema. La Prueba de integración la podemos definir como el proceso que asegura que los modelos de implementación funcionen correctamente al ser ejecutados para darle cumplimiento a los casos de uso del sistema y en concordancia con el diseño trazado. La Prueba de Sistema es aquella que se realiza para verificar que el sistema una vez puesto en marcha funciona como se ha previsto.

Tipos de Prueba.

De acuerdo a las dimensiones de calidad: Funcionalidad, Fiabilidad y Performance (Rendimiento) se desarrollarán los siguientes tipos de prueba:

- **Función:** para verificar el funcionamiento de la validación de las acciones, servicios y casos de uso del sistema.
- **Stress:** evaluar la respuesta del sistema de acuerdo a la sobrecarga de acciones, insuficiente memoria, servicios y hardware.
- **Carga:** aceptabilidad de acciones del sistema bajo carga de trabajo variable, en una permanencia constante.

4.4.2 Herramientas de Prueba.

Casos de Prueba.

Los casos de prueba pretenden demostrar que las funciones que se les realiza al software son operativas, además la entrada que se realiza al sistema se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Tabla 7: Secciones de Casos de Prueba.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Identificar Riesgos del Proyecto.	EC 1.1: Agregar riesgos al proyecto exitosamente.	Se agrega nuevos riesgos al proyecto.
SC 2: Analizar Riesgos	EC2.1: Insertar datos para el	Se agregan los datos

del Proyecto.	análisis del riesgo.	correspondientes al análisis del riesgo.
	EC2.2: No se agregan los campos correctos.	No se agregan los campos incorrectos.
SC 3: Priorizar Riesgos del Proyecto.	EC 3.1: Calcular la prioridad de los riesgos del proyecto.	Se priorizan los riesgos en el proyecto.
SC 4: Planificar Riesgos del Proyecto.	EC 4.1: Seleccionar los datos para la planificación del riesgo.	Se agregan los datos correspondientes a la planificación del riesgo.
	EC 4.2: No se seleccionan los campos correctos.	No se agregan los campos incorrectos.
SC 5: Mitigar Riesgos del Proyecto.	EC 5.1: Insertar datos para la mitigación del riesgo.	Se agregan los datos correspondientes a la mitigación del riesgo.
	EC 5.2: No se agregan los campos correctos.	No se agregan los campos incorrectos.
SC 6: Monitorear Riesgos del Proyecto.	EC 6.1: Seleccionar los datos para monitorear del riesgo.	Se realiza el monitoreo de los riesgos en el proyecto.
	EC 6.2: No se seleccionan los campos correctos.	No se agregan los campos incorrectos.
SC 7: Reportes de los Riesgos del Proyecto.	EC 7.1: Insertar los datos para realizar el reporte de los riesgos.	Se realiza el reporte de los proyectos.
	EC 7.2: No se insertan los campos correctos.	No se realizan los reportes.

Herramienta JMeter.

JMeter es una herramienta Java desarrollada dentro del proyecto Jakarta que dispone de una variedad de componentes para facilitar la elaboración de los escenarios de prueba, simulando para cada escenario diversos usuarios, permite realizar pruebas de rendimiento distribuidas entre

distintos ordenadores y pruebas funcionales sobre Aplicaciones Web; test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC, y WebServices (en Beta).

4.4.3 Resultados de las Pruebas.

Las pruebas realizadas al sistema se arrojaron 34 no conformidades de aplicación (21 de funcionalidad, 5 de opciones que no funcionan, 2 de validación, 1 de errores en la interfaz y 5 de excepciones), 4 de documentación (todas de otros errores) y 2 de comunes para ambos artefactos (de ortografía y errores de idiomas). A continuación se realiza un resumen de las no conformidades encontradas en cada iteración.

Tabla 8: Resumen de No Conformidades.

Iteración	Cantidad de No Conformidades	Resueltas	No Resueltas
Primera	23	21	2
Segunda	17	14	3
Tercera	5	5	0

Al aplicar la herramienta JMeter a la aplicación para realizarle las pruebas de carga y stress está no pudo acceder a las funcionalidades de la herramienta por la seguridad de autenticación que posee esta, pero los resultados de estas pruebas fueron dados de acuerdo a estudios estadísticos realizados por el grupo de calidad de nuestra universidad de acuerdo a los requisitos funcionales del sistema este puede aceptar hasta 55 usuarios de forma concurrente, por lo que la aplicación soporta la cantidad de usuarios que tendrán acceso a ella.

CONCLUSIONES DEL CAPÍTULO

Durante el desarrollo de este capítulo se dieron a conocer aspectos fundamentales de acuerdo a los flujos de trabajo implementación y prueba, empleando el diagrama de componentes para estructurar el modelo de implementación en términos de subsistemas de implementación, así como mostrar las relaciones entre sus elementos. Se dieron a conocer los resultados de las pruebas.

CONCLUSIONES

Con el desarrollo del presente trabajo de diploma se logra el diseño de una herramienta fácil de usar para la gestión de los riesgos en los proyectos productivos de la facultad 6, desarrollar una herramienta para gestionar los riesgos de los proyectos productivos de la facultad 6 a partir de la implementación de los procesos de la gestión de riesgos y Se verifica la herramienta para el despliegue posterior.

RECOMENDACIONES

1. Aumentar los riesgos almacenados en la base de datos para que constituya un archivo histórico en los proyectos.

2. Aplicar técnicas de inteligencia artificial y minería de datos para realizar estudios de tendencia y reportes estadísticos del comportamiento de los riesgos en los proyectos productivos.

3. Implementar nuevas herramientas para mejorar el proceso de monitoreo de los riesgos en los proyectos productivos de la facultad 6.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cabrera, Armando, Castillo, Luis Enriquez y Cuenca, Luis Eduardo.** *Gestión del Riesgo.* [En línea]. [Citado el: 24 de octubre de 2009]. http://www.slideshare.net/lecastillox/gestion-del-riesgo.

2. *RIESGOS DEL SOFTWARE*. [En línea] [Citado el: 16 de octubre de 2009.] www.sitios.uach.cl/caminosfor/CristianSalazar/SIE/RS.html - .

3. *07. Gestión de Riesgos.* [En línea] [Citado el: 18 de octubre de 2009.] www.di.uniovi.es/aquilino/Asignaturas/ProyectosInformatica/Documentos/07-GestionRiesgos.pdf.

4. **López Cabrera, Yanisleidy; Alvarez Lamas, Tailys.** *Propuesta para la Gestión de Riesgo en los proyectos productivos de la UCI.* [En línea] 2007. [Citado el: 10 de noviembre de 2009.] http://bibliodoc.uci.cu/TD/TD_0466_07.pdf.

5. **SEI. 2004.** *Software Engineering Institute.* 2004 [Consultado] [Citado el: 20 de octubre de 2009.].

6. *Gestión de riesgos en Ingeniería de Software* [En línea] [Citado el: 9 de marzo de 2010.]http://www.wikilearning.com/curso_gratis/gestion_de_riesgos_en_ingenieria_del_software/3620-3

7. **Oropesa Pupo, Tania y Rengifo Hardy, Yamilé. 2009.** *Sistema informático para la gestión de los riesgos en los proyectos productivos.* [Consultado] [Citado el: 20 de octubre de 2009.].

8. *Active Risk Manager* [En línea] [Citado: 19 de enero de 2010] http://www.overseasbr.com/es/riskmanagement/risksolutons/arm.asp.

9. *TRIMS.* [En línea] [Citado: 8 de octubre de 2009] http://www.bmpcoe.org/pmws/trims.html.

10. **Kulik, Peter and Catherine Weber. 2001.** *Software Risk Management Practices.* 2001[Consultado] [Citado el: 20 de octubre de 2009.].

11. **ANSI/IEEE 1042, 1987.** *IEEE Guide to Software Configuration Management (ANSI/IEEE STD 1042-1987).* http://www.standards.ieee.org/.

12. *Conferencia 5: Fase de Inicio. Modelo del Negocio.* [En línea] [Citado: 24 de octubre de 2009].http://eva.uci.cu/file.php/102/Curso_2009-2010/Semana_4/Conferencia_5/Materiales_Basicos/Conferencia_5.doc

13. *Riesgos del Software.* [En línea] [Citado el: 24 de octubre de 2009.] www.sitios.uach.cl/caminosfor/CristianSalazar/SIE/RS.html - .

14. *Fase de Inicio. Modelo del Negocio.* [En línea] 2007-2008. [Citado el: 6 de febrero de 2010.] http://eva.uci.cu/mod/resource/view.php?id=11553.

15. *Conferencia 6: Fase de Inicio. Disciplina de Requisitos.* [En línea] [Citado: 6 de febrero de 2010]. http://eva.uci.cu/file.php/102/Curso_2009-2010/Semana_7/Conferencia_6/Materiales_Basicos/Conferencia_6.doc

16. *Conferencia # 7: Fase de Construcción. Disciplina Prueba.* [En línea] [Citado: 8 de mayo de 2009]. http://eva.uci.cu/file.php/259/Curso_2009-2010/Conferencia_7/Materiales_basicos/Conferencia_7_Disciplina_Prueba.doc.

BIBLIOGRAFÍA

RIESGOS DEL SOFTWARE. [En línea] [Citado el: 16 de octubre de 2009.] www.sitios.uach.cl/caminosfor/CristianSalazar/SIE/RS.html - .

07. Gestión de Riesgos. [En línea] [Citado el: 18 de octubre de 2009.] www.di.uniovi.es/aquilino/Asignaturas/ProyectosInformatica/Documentos/07- GestionRiesgos.pdf.

SEI. 2004. Software Engineering Institute. 2004 [Consultado] [Citado el: 20 de octubre de 2009.].

CONNELL, S. 1997. Desarrollo y Gestión de Proyectos Informáticos. 1997 [Consultado] [Citado el: 20 de octubre de 2009.].

Kulik, Peter and Catherine Weber. 2001. Software Risk Management Practices. 2001[Consultado] [Citado el: 20 de octubre de 2009.].

Gestión de riesgos en Ingeniería de Software [En línea] [Citado el: 9 de marzo de2010.]http://www.wikilearning.com/curso_gratis/gestion_de_riesgos_en_ingenieria_del_software/3620-3

Oropesa Pupo, Tania y Rengifo Hardy, Yamilé. 2009. Sistema informático para la gestión de los riesgos en los proyectos productivos. [Consultado] [Citado el: 20 de octubre de 2009.].

Conferencia 5: Fase de Inicio. Modelo del Negocio. [En línea] [Citado: 24 de octubre de 2009].http://eva.uci.cu/file.php/102/Curso_2009-2010/Semana_4/Conferencia_5/Materiales_Basicos/Conferencia_5.doc

Riesgos del Software. [En línea] [Citado el: 24 de octubre de 2009.] www.sitios.uach.cl/caminosfor/CristianSalazar/SIE/RS.html - .

Fase de Inicio. Modelo del Negocio. [En línea] 2007-2008. [Citado el: 6 de febrero de 2010.] http://eva.uci.cu/mod/resource/view.php?id=11553.

Conferencia 6: Fase de Inicio. Disciplina de Requisitos. [En línea] [Citado: 6 de febrero de 2009]. http://eva.uci.cu/file.php/102/Curso_2009-2010/Semana_7/Conferencia_6/Materiales_Basicos/Conferencia_6.doc

Cabrera, Armando, Castillo, Luis Enriquez y Cuenca, Luis Eduardo. Gestión del Riesgo. [En línea] [Citado el: 19 de octubre de 2009.] http://www.slideshare.net/lecastillox/gestion-del-riesgo.

López Cabrera, Yanisleidy; Álvarez Lamas, Tailys. Propuesta para la Gestión de Riesgo en los proyectos productivos de la UCI. [En línea] 2007. [Citado el: 19 de octubre de 2009.] http://bibliodoc.uci.cu/TD/TD_0466_07.pdf.

INDECOPI. 2006. Norma Técnica peruana NTP-ISO/IEC 12207. Peru: s.n., 2006.

CMMI. 2002. Capability Maturity Model Integration. 2002.

Gracia, J. 2005. CMM – CMMI Volumen, II. 2005.

ANEXOS

ANEXO 1: Entrevista con líderes de proyectos.

1. Cree usted que el proceso de gestión de riesgos que se está realizando en los proyectos, cumple los objetivos de este proceso.
2. Se logra efectividad en el proyecto con el manejo de los riesgos de la forma establecida hasta el momento.
3. ¿Cuáles serían sus argumentos sobre la creación de una aplicación para el proceso de Gestión de Riesgos?

ANEXO 2: Descripción de Casos de Uso del Sistema.

Caso de Uso:	Identificar Riesgos del Proyecto
Actores:	Administrador_Riesgos
Resumen:	El caso de uso se inicia la identificación de riesgos del proyecto, terminando al registrar todos los riesgos encontrados en el proyecto
Referencias	RF 17
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Identificar riesgos del proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
El administrador de riesgos selecciona la opción identificar riesgos en el proyecto.	El sistema muestra la interfaz para que el administrador de riesgos seleccione el proyecto a realizar la identificación de los riesgos.
Selecciona el proyecto y acepta.	El sistema una interfaz con todos los riesgos que se encuentran en la base de datos
El administrador de riesgos selecciona los riesgos a introducir en el proyecto y presiona el botón añadir.	El sistema muestra los riesgos que se van introduciendo en el proyecto, y los agrega a la base de datos, finalizando así el caso de uso.
Poscondiciones	Se introducen los riesgos del proyecto en la base de datos del sistema.

Tabla 9: Descripción del Caso de Uso del Sistema "Identificar Riesgos del Proyecto".

Caso de Uso:	Analizar Riesgos del Proyecto
Actores:	Administrador_Riesgos
Resumen:	El caso de uso se inicia cuando se decide analizar los riesgos identificados

	en el proyecto, terminando con la devolución de los reportes, referente al comportamiento de los riesgos en el proyecto.
Referencias	RF 18
Prioridad	Crítico
Sección "Analizar riesgos del Proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
El administrador de riesgos selecciona la opción analizar riesgos en el proyecto.	El sistema muestra la interfaz para que el administrador de riesgos seleccione el proyecto a realizar la identificación de los riesgos.
Selecciona el proyecto y acepta.	El sistema una interfaz con todos los riesgos que se encuentran en la base de datos identificados en el proyecto.
El administrador de riesgos selecciona el riesgo a analizar introducir en el proyecto.	El sistema muestra la interfaz para introducir los datos referentes al efecto y la probabilidad del riesgo en el proyecto.
El administrador de riesgos inserta los datos y presiona el botón aceptar.	El sistema una interfaz con todos los riesgos que se encuentran en la base de datos identificados en el proyecto finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador de riesgos oprime el botón cancelar.	* El sistema cierra la interfaz.
Poscondiciones	Se introducen los datos en la base de datos del sistema.

Tabla 10: Descripción del Caso de Uso del Sistema "Analizar Riesgos del Proyecto".

Caso de Uso:	Priorizar Riesgos del Proyecto
Actores:	Administrador_Riesgos
Resumen:	El caso de uso se inicia cuando se decide priorizar los riesgos del proyecto, terminando con la devolución de los riesgos identificados y analizados en el proyecto priorizados.
Referencias	RF 19
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Priorizar riesgos del Proyecto**"**	

Acción del Actor	Respuesta del Sistema
1. El administrador de riesgos selecciona el proyecto a priorizar sus riesgos y presiona el botón aceptar.	2. El sistema muestra la interfaz con todos los riesgos identificados y analizados en el proyecto, priorizados finalizando así el caso de uso.
Poscondiciones	Se introducen todos los datos en la base de datos del sistema.

Tabla 11: Descripción del Caso de Uso del Sistema "Priorizar Riesgos del Proyecto".

Caso de Uso:	Planificar Riesgos del Proyecto
Actores:	Administrador_Riesgos
Resumen:	El caso de uso se inicia cuando se decide planificar los riesgos del proyecto, terminando con la inserción de la planificación de los riesgos de un proyecto.
Referencias	RF 20
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Planificar los riesgos del Proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
1. El revisor de riesgos selecciona la opción planificar los riesgos en el proyecto.	2. El sistema muestra la interfaz con los proyecto.
3. El revisor de riesgos selecciona el proyecto a realizarle la planificación a los riesgos y presiona el botón aceptar.	4. El sistema muestra la interfaz para que el revisor de riesgos seleccione el riesgo a planificar.
5. El revisor de riesgos selecciona el riesgo a realizarle la planificación y presiona el botón aceptar.	6. El sistema muestra la interfaz para que el revisor de riesgos introduzca la planificación del riesgo.
7. El revisor de riesgos realiza la planficación del riesgo y presiona el botón aceptar.	8. Inserta los datos introducidos en la base de datos y muestra la interfaz con los demás riesgos sin planificar hasta que culmine con todos, finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El administrador de riesgos oprime el botón cancelar.	* El sistema cierra la interfaz.
Poscondiciones	Se introducen todos los datos en la base de datos del sistema.

Tabla 12: Descripción del Caso de Uso del Sistema "Planificar Riesgos del Proyecto".

Caso de Uso:	Mitigar Riesgos del Proyecto
Actores:	Revisor_Riesgos
Resumen:	El caso de uso se inicia cuando se decide mitigar los riesgos del proyecto.
Referencias	RF 21

Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Mitigar los riesgos del Proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
1. El revisor de riesgos selecciona la opción mitigar los riesgos en el proyecto.	2. El sistema muestra la interfaz con los proyecto, a realizarle la mitigación a los riesgos.
3. El revisor de riesgos selecciona el proyecto a realizarle la mitigación y presiona el botón aceptar.	4. El sistema muestra la interfaz para que el revisor de riesgos seleccione el riesgo a mitigar.
5. El revisor de riesgos selecciona el riesgo a realizarle la mitigación y presiona el botón aceptar.	6. El sistema muestra la interfaz para que el revisor de riesgos introduzca la mitigación del riesgo.
7. El revisor de riesgos realiza la mitigación del riesgo y presiona el botón aceptar.	8. Inserta los datos introducidos en la base de datos y muestra la interfaz con los demás riesgos sin mitigar hasta que culmine con todos, finalizando el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El revisor de riesgos oprime el botón cancelar.	* El sistema cierra la interfaz.
Poscondiciones	Se introducen todos los datos en la base de datos del sistema.

Tabla 13: Descripción del Caso de Uso del Sistema "Mitigar Riesgos del Proyecto".

Caso de Uso:	Monitorear Riesgos del Proyecto
Actores:	Revisor_Riesgos
Resumen:	El caso de uso se inicia cuando se decide monitorear los riesgos del proyecto.
Referencias	RF 22
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Monitorear los riesgos del Proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
1. El revisor de riesgos selecciona la opción monitorear los riesgos en el proyecto.	2. El sistema muestra la interfaz con los proyecto, a realizarle el monitoreo a los riesgos.
3. El revisor de riesgos selecciona el proyecto a realizarle el monitoreo y presiona el botón aceptar.	4. El sistema muestra la interfaz para que el revisor de riesgos seleccione el riesgo a l monitorear.
5. El revisor de riesgos selecciona el riesgo a realizarle el monitoreo y presiona el botón aceptar.	6. El sistema muestra la interfaz para que el revisor de riesgos seleccione el monitoreo del riesgo.

7. El revisor de riesgos selecciona el monitoreo del riesgo y presiona el botón aceptar.	8. Inserta los datos en la base de datos y muestra la interfaz con los demás riesgos sin monitorear hasta que culmine con todos, finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El revisor de riesgos oprime el botón cancelar.	* El sistema cierra la interfaz.
Poscondiciones	Se introducen todos los datos en la base de datos del sistema.

Tabla 14: Descripción del Caso de Uso del Sistema "Monitorear Riesgos del Proyecto".

Caso de Uso:	Reportes Comportamiento de los Riesgos del Proyecto
Actores:	Revisor_Riesgos
Resumen:	El caso de uso se inicia cuando se decide realizar la gestión de riesgos del proyecto, terminando con la devolución de los reportes, referente al comportamiento de los riesgos en el proyecto.
Referencias	RF 23
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Reportes de los riesgos del Proyecto**"**	
Acción del Actor	**Respuesta del Sistema**
1. El revisor de los riesgos selecciona la opción Reportes de los riesgos del Proyecto.	2. El sistema muestra una interfaz con los proyectos.
3. Selecciona el proyecto y oprime el botón aceptar.	4. Muestra una interfaz con el comportamiento de los riesgos en la fecha seleccionada, finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
* El revisor de riesgos oprime el botón cancelar.	* El sistema cierra la interfaz.

Tabla 15: Descripción del Caso de Uso del Sistema "Reportes Comportamiento de los Riesgos del Proyecto".

GLOSARIO DE TÉRMINOS

CMMI: Capability Maturity Model Integration, modelo integrado de capacidad y madurez para la definición, implantación, evaluación, mejora y optimización de los procesos de desarrollo y mantenimiento de sistemas y productos de software.

Taxonomía: Clasificaciones ordenadas de elementos de acuerdo a sus relaciones presumidas; y pueden emplearse como herramientas de suma utilidad en diferentes ramas de la ciencia y la industria donde se pretende organizar y facilitar el acceso a un número importante de elementos que se encuentran mutuamente relacionados de alguna manera relevante.

RUP: Proceso Unificado (Rational Unified Process) metodología para el desarrollo de sistemas informáticos, dirigidos por casos de uso.

Object/Relational Mapping (ORM): es una técnica de programación para 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.

Plugins: Función o utilidad generalmente muy específica. Se adiciona a algún programa para ser ejecutado. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, etc.

ODBC: Open DataBase Connectivity, es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, su objetivo es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos los almacene.

XML: Siglas en ingles de eXtensible Markup Language, es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.