Universidad de Matanzas

Facultad de Ciencias Técnicas

Ingeniería Informática

Trabajo de Diploma para optar el título de Ingeniero Informático

Título: Sistema web para la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los Softwares Educativos.

Autor: Ediel Pérez Alvarez.

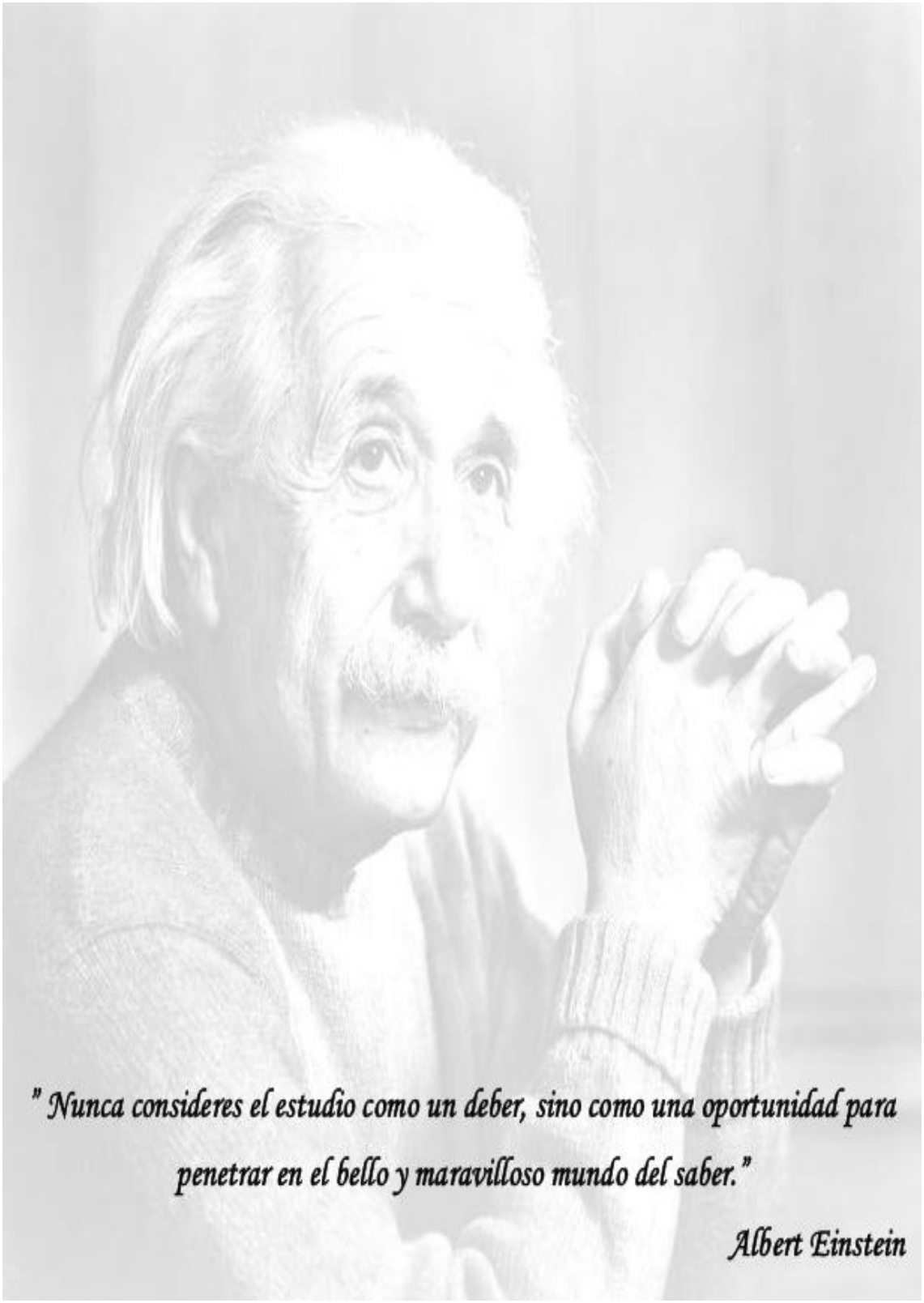
Tutor(es): Dr.C. Walfredo González Hernández.

Ing. Dayana O. Hernández Revilla.

Matanzas, Cuba

Junio, 2018

Pensamiento



Dedicatoria

A mis padres por toda la confianza y apoyo incondicional; por todo el sacrificio que han tenido que hacer y estar a mi lado en todo momento.

A mi hermano Eduardo por estar siempre a mi lado y ayudarme en todo momento.

A mi novia preciosa por existir en mi vida; por su completa dedicación, entrega y amor.

A mis tíos, abuelos, suegros, cuñadas y resto de mi familia por siempre estar ahí para ayudarme en todo momento.

A todos mis amigos, sin mencionar, para que no quede nadie.

A mis tutores que pusieron todo su empeño en este trabajo.

Agradecimientos

Esta se convierte en la página más difícil de completar, cuando han sido muchas las personas que han participado en esta larga travesía, por eso quiero agradecer a todos los que de una forma u otra contribuyeron a que este sueño se convirtiera en realidad.

Un agradecimiento muy especial a mi mamá, mi papá y hermano que me han ayudado en todo momento, por todo el cariño, confianza, todo el sacrificio del mundo para que pudiera culminar mi carrera, siempre les estaré eternamente agradecido.

A mi novia hermosa Yeni que me ha guiado por el buen camino durante todos estos años, por su apoyo incondicional, por todo el esfuerzo realizado y por demostrarme su cariño día a día, le estaré agradecido el resto de mi vida.

A mis tíos, abuelos, primos por su dedicación y el amor que siempre me han demostrado.

A mis tutores: Walfredo González Hernández y Dayana O. Hernández Revilla. Gracias por sus enseñanzas, por sus consejos, por brindar lo mejor de sí y apoyarme en todo momento. Por su dedicación, ayuda y apoyo durante toda la investigación y por su ayuda y empeño en mi formación profesional. Mi eterno agradecimiento.

A todos mis compañeros de aula por compartir estos años de estudio y por la ayuda y apoyo que siempre me han brindado; en especial a Roberto, Marlon, Wuilliam, Yonelkys, Geykel, Alain, Mario, Ernesto, Reniel, Heydi, Zulliam, Lianet y Isselis. Gracias por su ayuda sincera y comprensión en todo momento, por el tiempo que compartimos juntos, los voy a extrañar.

A René Betancourt Perera, Félix Cabrera Ranklim, Yuniel Ramírez, Yohan Hernández vuestra ayuda fue fundamental en este trabajo, muchísimas gracias.

A todos, de corazón, muchas gracias.

Declaración de Autoría

Yo, Ediel Pérez Alvarez, declaro que soy el único autor de este trabajo y autorizo a la Universidad de Matanzas Sede: ¨ Camilo Cienfuegos ¨, especialmente a la Facultad de Ciencias Económicas e Informáticas y al Laboratorio de Tecnología en la Educación, a que hagan el uso que estimen pertinente de él.

Para que así conste firmo la presente a los días del mes de junio del año 2018.

----------	**----------**
Firma del Autor	**Firma del Tutor**
Ediel Pérez Alvarez	**Walfredo González**

Opinión del cliente y tutor del Trabajo de Diploma

TÍTULO DE LA TESIS: Sistema Web para la gestión del riesgo en los procesos de desarrollo de software educativo.

AUTOR. - Ediel Pérez Alvarez.

Tutor(es): Ing. Dayana O. Hernández Revilla, Dr.C. Walfredo González Hernández.

Resumen

El riesgo se halla de forma implícita asociado a toda actividad, por tanto, no es la excepción en el desarrollo de un software educativo. En la actualidad la gestión y control del riesgo en estos proyectos se realizan de forma manual, con el apoyo de documentos basados en Microsoft Excel y Microsoft Word por lo tanto hace el proceso engorroso y lento, tampoco se ha encontrado ningún proyecto que realice de forma automatizada estos procesos de gestión en estos softwares educativos. Debido a esta problemática surge la necesidad de realizar el presente trabajo que tiene como objetivo desarrollar un sistema informático que gestione los riesgos en el proceso de desarrollo de los softwares educativos de forma tal que garantice una mayor protección, confidencialidad de los datos y menor tiempo de trabajo al personal encargado del mismo. Se propone una aplicación capaz de identificar, eliminar o mitigar los riesgos que aparezcan en un software educativo en la etapa de desarrollo. Para la planificación y perfeccionamiento de este proyecto fue utilizada la metodología de desarrollo ágil eXtreme Programming (XP), como gestor de base de datos MySQL 4.7.0, lenguaje PHP 7.1.7, como servidor web Apache 2.4.26, *framework* Symfony 3.3.8, el Entorno de Desarrollo Integrado (IDE) utilizado fue JetBrains PhpStorm 2017.1.4.

Summary

The risk is implicitly associated with any activity, therefore, it is not the exception in the development of educational software. Currently, risk management and control in these projects are carried out manually, with the support of documents based on Microsoft Excel and Microsoft Word; therefore, the process is cumbersome and slow, and no project has been found to perform automated these management processes in these educational softwares. Due to this problem arises the need to carry out the present work which aims to develop a computer system that manages the risks in the development process of educational software in a way that ensures greater protection, confidentiality of data and less time I work for the personnel in charge of it. An application capable of identifying, eliminating or mitigating the risks that appear in an educational software in the development stage is proposed. For the planning and improvement of this project the agile development methodology eXtreme Programming (XP) was used, as MySQL database manager 4.7.0, PHP 7.1.7 language, as Apache web server 2.4.26, Symfony 3.3 framework. 8, the Integrated Development Environment (IDE) used was JetBrains PhpStorm 2017.1.4.

Índice

Introducción

La ingeniería de software ha abierto nuevas puertas a la informática en este siglo. Por esta causa se debe seguir renovando constantemente los conocimientos y los conceptos como base de lo que se conoce como ingeniería de sistemas centrándose en diversos elementos: analizar, diseñar y organizar esos elementos en un sistema que puede ser un producto, un servicio o una tecnología para la transformación o control de información.

En Cuba se está llevando a cabo el proceso de informatización de la sociedad, comenzando por el sistema educacional. Las instituciones educativas se convierten en vías para la preparación de las futuras generaciones para la inserción de los cubanos a un mundo totalmente informatizado. A través de la tecnología educativa como medio de enseñanza basado fundamentalmente técnicas y medios, en la búsqueda de realizar de una forma más interactiva y que resulta “eficiente” para la obtención de los resultados prefijados. El proceso de enseñanza-aprendizaje es apoyado de forma eficiente como medio: el software educativo.

El software educativo es de vital importancia en la actualidad, permitiendo la educación a distancia. Simula las condiciones existentes en un aula y de esta manera el estudiante puede acceder e interactuar con el docente mediante chat, videoconferencias, entre otras; por tanto, proporciona a través de sus actividades información detallada de la realidad al estudiante. Este orienta y regula el aprendizaje de los estudiantes ya sea de forma explícita o implícita facilitando el logro de los objetivos educativos específicos. También motiva a los estudiantes debido a que se suelen incluir elementos para atraer su atención, su interés y enfocarlos hacia los aspectos más importantes de las actividades. Presenta función de evaluadora, debido a la interactividad que contienen permiten responder a las respuestas y acciones de los estudiantes, les hace especialmente adecuados para evaluar el trabajo que van realizando con ellos.

Según Marqués (2014) el software educativo, programa educativo y programa didáctico son sinónimos para designar genéricamente los programas para ordenador creados con la finalidad de ser utilizados como medio didáctico, es decir, para facilitar el proceso de enseñanza y aprendizaje.

En su definición engloba todos los programas con fines didácticos, los programas de Enseñanza Asistida por Ordenador (EAO) y los programas experimentales de

Enseñanza Inteligente Asistida por Ordenador (EIAO), utilizando técnicas propias de los Campos de Inteligencia Artificial y los Sistemas Expertos.

Los programas educativos pueden tratar de diferentes temas (Ingles, Matemática) pero de formas muy diversas y ofrecer un entorno de trabajo sensible a las circunstancias de los estudiantes y con bastantes posibilidades de interacción, pero todos comparten según Marqués (2014) cinco características esenciales:

-Contienen finalidad Didáctica.

-Utilizan Ordenador para que realicen las actividades propuestas.

-Son interactivos, contestan inmediatamente las acciones de los alumnos y permiten un diálogo y un intercambio entre el ordenador y el alumno.

-Individualizan el trabajo de los estudiantes.

-Son fáciles de usar.

A través de las técnicas de la Inteligencia Artificial y el progreso de las tecnologías multimedia, se desea que cada vez los entornos de comunicación sean más intuitivos y que estén próximos a un lenguaje natural que contengan un diálogo más abierto.

Según Marqués (2014) los softwares educativos tienden a tener rasgos básicos y una estructura en general común que contienen rasgos muy disímiles, unos tienden a ser una biblioteca o un laboratorio, otros se limitan a ofrecer una función instrumental del tipo máquina de escribir o calculadora, otros se presentan como un juego o como un libro, bastantes tienen vocación de examen, algunos se creen expertos y, por si no fuera bastante, la mayoría participan en mayor o menor medida de algunas de estas peculiaridades. Para poner orden a esta disparidad, se han elaborado múltiples tipologías que clasifican los programas didácticos a partir de diferentes criterios.

Uno de estos criterios se basa en la consideración del tratamiento de errores que cometen los alumnos, distinguiendo:

- **Programas Tutoriales Directivos,** que hacen preguntas a los alumnos y controlan todo el proceso. A través de estas preguntas el ordenador hace de papel de juez de la verdad y examina al alumno, la respuesta es incorrecta cuando el alumno está en desacuerdo con la que el ordenador tiene como correcta.

- **Programas no Directivos,** en los que el ordenador adopta el papel de un laboratorio o instrumento a disposición de la iniciativa de un alumno que pregunta y tiene una libertad de acción sólo limitada por las normas del programa.

Otra clasificación de estos programas son los que posibilitan realizar modificaciones en los contenidos del software y distingue entre sistemas abiertos y cerrados, que proporcionan un esqueleto, una estructura, sobre la cual los profesores y los alumnos pueden añadir los contenidos que les atraen.

Según Lisandra Morales (2011) "existen otras clasificaciones de softwares educativos (Vázquez 2009):

Sistemas Tutoriales: Sistema basado en el diálogo con el estudiante, adecuado para presentar información objetiva, tiene en cuenta las características del alumno, siguiendo una estrategia pedagógica para la transmisión de conocimientos.

Sistemas Entrenadores: Se parte de que los estudiantes cuentan con los conceptos y destrezas que van a practicar, por lo que su propósito es contribuir al desarrollo de una determinada habilidad, intelectual, manual o motora, profundizando en las dos fases finales del aprendizaje: aplicación y retroalimentación.

Libros Electrónicos: Su objetivo es presentar información al estudiante a partir del uso de texto, gráficos, animaciones, videos, entre otros, pero con un nivel de interactividad y motivación que le facilite las acciones que realiza.

Simuladores: Su objetivo es apoyar el proceso de enseñanza – aprendizaje, semejando la realidad de forma entretenida.

Juegos Educativos: Su objetivo es llegar a situaciones excitantes y entretenidas, sin dejar en ocasiones de simular la realidad.

Sistemas Expertos: Programa de conocimientos intensivo que resuelve problemas que normalmente requieren de la pericia humana. Ejecuta muchas funciones secundarias de manera análoga a un experto, por ejemplo, preguntar aspectos importantes y explicar razonamientos.

Repasadores: Diseñado con el propósito de contribuir a repasar a determinados contenidos estudiados con anterioridad.

Evaluadores: Para producir las evaluaciones y para aplicarlas".

En Cuba se están empleando los softwares educativos en la mayoría de los centros educativos con el fin de apoyar el proceso de Enseñanza-Aprendizaje y servirle de gran apoyo al profesor con el contenido de las asignaturas. Se emplean algunos tipos de estos softwares como son los sistemas tutoriales, simuladores, juegos educativos, entre otros. No obstante, no todos los softwares que se crean con este fin son considerados educativos ya que les falta orientación pedagógica, didáctica, ordenamiento de los contenidos según el programa de estudio, por tanto, a la hora de catalogar un software de este tipo se deben tener un serio proceso de evaluación, selección y que contenga fundamentos pedagógicos.

Se realizaron entrevistas al personal calificado que participan en el desarrollo de los softwares educativos en los centros de Tecnología Educativa, ya sea en la Universidad de Matanzas o en los otros centros del país donde se obtuvo como respuesta que la mayoría desconocía la gestión de riesgo a la hora de desarrollar dichos softwares. Dentro de esta indagación realizada no se había encontrado ningún software o herramienta que permitiera la gestión de los riesgos durante el proceso de desarrollo de los softwares educativos, siendo entrevistada la Directora Nacional de Tecnología Educativa que expresó que desconocía de un software con estas características. También se consultaron el sitio web (Google Académico), el Centro de Información Científico-Técnica de la Universidad de Matanzas (CICT) y la revista nacional de Tecnología Educativa donde no se ha encontrado ningún software que permitiera la gestión de los riesgos durante el desarrollo de los softwares educativos. Por tanto, no se ha determinado un software que permita la gestión de los riesgos en la etapa de desarrollo de los softwares educativos para evitar que gran parte de estos softwares sean un fracaso por no hacer un estudio de los problemas que se pueden evitar, identificar, o las pautas que hay que seguir para encontrar estos problemas y solucionarlos, para que el futuro del producto sea exitoso.

Se especifica los elementos más importantes en el proyecto y se establece cada uno de los contextos (Educacional y Tecnológico), se identifican los riesgos a través de una serie de preguntas. Después de responder cada pregunta y detallar los riesgos posibles se realiza una categorización de cada una de ellas de manera general y específica, después se procede a evaluarlos con resultados matemáticos, para tener un valor específico del riesgo. Al realizar las etapas

anteriores se puede pasar a la toma de decisiones, es decir se toman las medidas pertinentes para su eliminación o al menos mitigar los daños que ocasionarán en el proyecto mediante el Plan de Acción del Riesgo, esta etapa se denomina Tratamiento de los riesgos. Después se realiza el Control y Supervisión de los riesgos donde se mantiene un control estricto sobre el mismo y es supervisado por el jefe del proyecto donde vuelve a evaluar los riesgos con resultados matemáticos y vuelve a realizar el Plan de Acción de Riesgo.

A todo ello se le denomina Gestión de Riesgos que se define como "… una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre. Un proyecto de software puede estar lleno de problemas. Un riesgo es un problema potencial (puede ocurrir o no). Pero sin tener en cuenta el resultado, es necesario identificarlo, evaluar su probabilidad de aparición, estimar su impacto, y establecer un plan de contingencia por si ocurre el problema" (Morales 2011).

Peter Drucker (1975) expresó" … aunque sea fútil intentar eliminar el riesgo, y cuestionable intentar minimizarlo, es esencial que los riesgos tomados sean los riesgos correctos" (pág-3). Antes de poder identificar los riesgos correctos que se van a tomar durante un proyecto de software, es importante identificar todos los que son obvios para gerentes y profesionales.

Según Lisandra Morales (2011), la gestión de riesgos definida por Pressman es el conjunto de elementos, medidas y herramientas dirigidas a la intervención de la amenaza o la vulnerabilidad, con el fin de disminuir o mitigar los riesgos existentes, ya que los riesgos son difíciles de eliminar totalmente.

Aunque existe un gran debate acerca de la definición adecuada para la gestión de riesgo existe un acuerdo en general en que los riesgos siempre involucran dos características: incertidumbre (el riesgo puede o no ocurrir, es decir, no hay riesgos 100 por ciento probables) y pérdida (si el riesgo se vuelve una realidad, ocurrirán consecuencias o pérdidas no deseadas (Higuera 1995)).

Cada riesgo se documenta individualmente usando hoja de información de riesgo (HIR) (Williams 1997). En la mayoría de los casos, la HIR se mantiene utilizando un sistema de base de datos, por lo que la creación y entrada de información, ordenación por prioridad, búsquedas y otros análisis pueden ser realizadas con facilidad. Pressman (2010) coincide con estos autores con esta necesidad y utiliza el formato que proponen ellos.

Determinando al riesgo de esta forma existen soluciones para mitigar los riesgos en los softwares en general. No obstante, al realizar un análisis en diferentes investigaciones realizadas sobre softwares educativos (Cataldi, Lage et al. 2006, Miranda Juana 2009, Brioli, Amaro et al. 2011, Giacomone and González 2012, Fernández López, Zermeño et al. 2016) no se encontraron los modelos a seguir para la gestión de riesgo en este tipo de software en específico. En las metodologías para el desarrollo de los softwares educativos creadas por (Azamar 2006, Marqués 2010, Cataldi 2013) en ninguna de las etapas empleadas se identifican los riesgos, ni se analizan, ni mitigan esos riesgos por lo que tampoco se realiza un plan de contingencia para dichos riesgos. En estas metodologías se abordan de forma muy general el proceso de desarrollo de los softwares educativos a pesar de que emplean muchas etapas no se les da el tratamiento a la gestión de riesgos que puede afrontar este tipo de software en su etapa de desarrollo.

De ahí que se determine como **problema de investigación**: ¿Cómo gestionar la información asociada a los riesgos durante el proceso de desarrollo de un software educativo?

La gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos constituye el **objeto de estudio** de esta investigación.

La presente investigación tiene como **Objetivo General**: Desarrollar un sistema informático que gestione la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.

Se define como **campo de acción** la informatización de la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.

Se ha determinado las siguientes **Preguntas Científicas**:

- ¿Cuál es el marco teórico referencial sobre la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos?
- ¿Cómo implementar un sistema web para la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos?

- ¿Cuán válido será un sistema web para la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos?

Para complementar el objetivo general se trazan las siguientes **tareas de investigación:**

- Elaborar el marco teórico referencial sobre la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.
- Implementar un sistema web que facilite la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.
- Validar el sistema web para la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.

En la presente investigación se utilizaron diversos **métodos de la investigación científica**. Entre los **métodos teóricos** se utilizaron:

- Histórico-lógico.
- Análisis-Síntesis.
- Inducción-Deducción.

Por otra parte, los **métodos empíricos** analizados fueron los de:

- Entrevista.
- Encuesta.

Estructura de la Investigación

La investigación se estructuró en 3 capítulos como se indica a continuación:

Capítulo 1: Marco teórico referencial

Este capítulo estará dividido en dos momentos:

1-Recopila los conceptos principales asociados al dominio del problema.

2-Realiza un estudio de las tendencias y tecnologías actuales sobre las que se apoya la propuesta y analiza las soluciones existentes.

Capítulo 2: Análisis, diseño y desarrollo de la solución propuesta

Presenta una propuesta de solución para el sistema, donde se describen las reglas y los elementos del negocio, una planificación inicial del proyecto con el

empleo de la metodología de Ingeniería del Software *eXtreme Programming* (XP). La solución que se propone se basa en el análisis y diseño del sistema; tomando en cuenta los intereses originados por el cliente, los cuales se describen mediante las Historias de Usuario.

Capítulo 3: Validación de la solución propuesta

Se hace el análisis de los resultados obtenidos. También se realizan las pruebas al software con el objetivo de entregarle al cliente un producto totalmente funcional, cumpliendo con todos los requisitos demandados por el mismo y satisfaciendo sus necesidades.

Capítulo 1: Marco teórico referencial

Introducción

Este capítulo contiene los conceptos fundamentales relacionados con el objeto de estudio, se abordan también los diferentes métodos de investigación empleados, como las herramientas, tecnologías y metodologías empleadas en la investigación.

1.1- Objeto de estudio

En esta investigación el objeto de estudio está dirigido a la gestión de los riesgos durante el proceso de desarrollo de los softwares educativos, por lo cual se hace una descripción más detallada de este elemento que se pretende realizar.

1.1.1- Definiciones de Riesgo

Aunque hay un considerable debate (Huang 2015, Dey, Khan et al. 2016, Domingues, Baptista et al. 2017) acerca de la definición de riesgo de software, existe un acuerdo general que el riesgo siempre involucra pérdida o incertidumbre. El riesgo se caracteriza, según Pressman (2011) como la presencia de pérdidas o desastres y averías en general supone la previa existencia de determinadas condiciones de riesgo.

Según Robert Charette (1989) se caracteriza al riesgo como un elemento que en primer lugar afecta los futuros acontecimientos, en segundo lugar, que involucra cambios y en tercer lugar el riesgo implica elección y la incertidumbre que ella conlleva. Por otro lado, el SEI (Software Engineering Institute) define al riesgo como “la posibilidad de sufrir una pérdida”.

Norma ISO 31000: 2009 para la gestión del riesgo.

Principios básicos para la gestión de riesgos según Mauricio Castro (2013)

La norma ISO 31000: 2009 establece los principios y directrices de carácter genérico sobre la gestión del riesgo. Para una mayor eficacia, la gestión del riesgo en una organización debe tener en cuenta los siguientes principios:

a) Crea valor
b) Está integrada en los procesos de la organización
c) Forma parte de la toma de decisiones
d) Trata explícitamente la incertidumbre
e) Es sistemática, estructurada y adecuada
f) Está basada en la mejor información disponible
g) Está hecha a medida

h) Tiene en cuenta factores humanos y culturales
i) Es transparente e inclusiva
j) Es dinámica, iterativa y sensible al cambio
k) Facilita la mejora continua de la organización

El enfoque está estructurado en tres elementos claves para una efectiva gestión de riesgos:

1. Los principios de gestión de los riesgos.
2. El marco de trabajo (framework) para la gestión de riesgos.
3. El proceso de gestión de riesgos.

La relación entre los principios de gestión, el marco de referencia y el proceso de gestión de riesgo desarrollado en la norma se resume en el **Anexo I.**

Beneficios:

La norma ISO 31000 está diseñada para ayudar a las organizaciones a:

- Aumentar la probabilidad de lograr los objetivos.
- Fomentar la gestión proactiva.
- Ser conscientes de la necesidad de identificar y tratar el riesgo en toda la organización.
- Mejorar en la identificación de oportunidades y amenazas.
- Cumplir con las exigencias legales y reglamentarias pertinentes, así como las normas internacionales.
- Mejora la información financiera.
- Mejora la gobernabilidad.
- Mejora la confianza de los grupos de interés (stakeholder)
- Establecer una base confiable para la toma de decisiones y la planificación.
- Mejorar los controles.
- Asignar y utilizar con eficacia los recursos para el tratamiento del riesgo.
- Mejorar la eficacia y eficiencia operacional.
- Mejorar la salud y seguridad, así como la protección del medio ambiente.
- Mejora la prevención de pérdidas, así como la gestión de incidentes.
- Minimizar las pérdidas.
- Mejorar el aprendizaje organizacional.
- Mejorar capacidad de recuperación de la organización.

En el presente trabajo se asume como riesgo de software como un evento o acontecimiento que afecta futuros sucesos, cambios en el proyecto y la posibilidad de sufrir pérdidas.

1.1.2- Definición de Software Educativos

El software educativo se refiere a los programas educativos o programas didácticos, conocidos también, como programas por ordenador, creados con la finalidad específica de ser utilizados para facilitar los procesos de enseñanza y aprendizaje. Se excluyen de este tipo de programas, todos aquellos de uso general utilizados en el ámbito empresarial que también se utilizan en los centros educativos con funciones didácticas o instrumentales como: procesadores de texto, gestores de base de datos, hojas de cálculo, editores gráficos, entre otros (Marques Graell 2007)

"Con la expresión 'software educativo' se representa a todos los programas educativos y didácticos creados para computadoras con fines específicos de ser utilizados como medio didáctico, para facilitar los procesos de enseñanza y de aprendizaje" (Marqués 2014;pág-1)

Según Piedra (2010) se definen los términos de los softwares educativos como "…aquellos programas que permiten cumplir y apoyar funciones educativas. En esta categoría entran tanto los que dan soporte al proceso de enseñanza y aprendizaje (un sistema para enseñar matemáticas, ortografía, contenidos o ciertas habilidades cognitivas), como los que apoyan la administración de procesos educacionales o de investigación (ej., un sistema que permita manejar un banco de preguntas)."

De otra manera se consideran que son "… los programas de computación realizados con la finalidad de ser utilizados como facilitadores del proceso de enseñanza y consecuentemente del aprendizaje, con algunas características particulares tales como: la facilidad de uso, la interactividad y la posibilidad de personalización de la velocidad de los aprendizajes" (Cataldi 2013)

El software educativo es un conjunto de programas que son utilizadas para la instrucción, formación o enseñanza. Por lo tanto, este tipo de software es utilizado para educar al usuario, por lo que se puede decir que es una herramienta pedagógica, que por la eficacia de las características que tiene, facilita y ayuda a adquirir conocimientos y a desarrollar habilidades de todo tipo.

Existen básicamente dos tendencias de software educativo: la primera se enfoca

a la instrucción asistida mediante la computadora y la otra hacia un software educativo abierto. (RUMANCELA 2017)

Para otro autor es "… un programa o conjunto de programas computacionales que se ejecutan dinámicamente según un propósito determinado. Se habla de software educativo cuando los programas incorporan una intencionalidad pedagógica, incluyendo uno o varios objetivos de aprendizaje" (Guerra 2016;pág-3)

El autor asume el criterio de Marqués (2014) como software educativo a todos los programas educativos y didácticos creados para computadoras con fines específicos de ser utilizados como medio didáctico, para facilitar los procesos de enseñanza y de aprendizaje, ya que esta metodología integra la mayoría de las aristas que se deben tener en cuenta a la hora de desarrollar cualquier software con estas características: las cuestiones pedagógicas y tecnológicas. Además, la mayoría de las metodologías especificadas con anterioridad se basan en las etapas de esta metodología por lo que la hace extensible y fácilmente modificable.

Para desarrollar un software educativo, como cualquier otro software, es necesario realizar un conjunto de tareas en dependencia de los rasgos que contienen cada uno de estos softwares denominadas metodologías (Hains-Wesson 2014, Chittaro and Buttussi 2015, Aslan 2016). En estos procesos de desarrollo se debe tener en cuenta el riesgo y la gestión del mismo para que el producto se realice de forma exitosa, por eso se hace un análisis y las metodologías correspondientes.

1.1.3- Metodologías para el desarrollo de software educativo

A continuación, se presenta una metodología propuesta por Pere Marqués (2010):

Esta Metodología para la elaboración de software educativa propone facilitar el proceso de diseño y desarrollo de software educativo, esta metodología contiene 11 etapas, cada una de las cuales se pueden dividir en etapas más específicas. Las etapas son las génesis de la idea donde se expresa la idea inicial de un programa que constituye una intuición global de lo que se quiere crear, contiene la semilla del qué se quiere trabajar y del cómo. Otra de las etapas más importante es pre-diseño o diseño funcional después de tener una idea inicial. El pre diseño constituye un primer guión del programa: contenidos, objetivos,

estrategia didáctica, entre otras. También se encuentran las etapas estudio de viabilidad y marco del proyecto, dosier completo de diseño o diseño orgánico, programación y elaboración del prototipo alfa-test, redacción de la documentación del programa, evaluación interna, ajuste y elaboración del prototipo beta-test, evaluación externa, ajuste y elaboración de la versión 1.0 y por última etapa se encuentra la publicación y mantenimiento del producto. A pesar que posee muchas etapas y lo abarca todo desde el punto de vista educativo no posee en ninguna de las etapas de esta metodología acciones relacionadas con los riesgos durante el proceso de desarrollo de estos softwares.

En la Metodología para el Desarrollo de Software Educativo planteada por Azamar (2006) se basa en 13 pasos fundamentales, en los cuales se toman aspectos de la Ingeniería de Software, Educación, Didáctica y Diseño Gráfico, entre otros. La finalidad de la metodología es la creación de productos de softwares creativos, pero que vayan de la mano con los planteamientos de una materia, método didáctico y tipo de usuario específico, ya que no todos los aprendizajes pueden, ni deben, ser planteados de la misma forma, porque las capacidades de los usuarios varían según la edad, medio ambiente y propuesta educativa. Los pasos que se llevan a cabo son:

1-Determinar la necesidad de un Software Educativo: Expresa que el Software Educativo deberá poder cubrir los aspectos primordiales del área o materia de estudios en cuestión y que la necesidad de realizar este producto permita entregar toda la información y las técnicas didácticas al Ingeniero de Software.

2- Formación del equipo de trabajo: Diversos autores plantean que se debe conformar un gran equipo de trabajo para poder desarrollar un Software Educativo completo.

3-Análisis y delimitación del tema: Se analizan las necesidades presentadas por las personas que requieren el software, deben permitir establecer el ámbito de la materia y determinar los temas específicos ya que se debe delimitar la amplitud de los temas a cubrir.

4-Definición del usuario: Basados en la definición del nivel de enseñanza al cual va dirigido el software educativo, deben determinarse las características del usuario.

5-Estructuración del contenido: Se deben definir los conceptos a considerar para establecer los contenidos temáticos que se abarcan en el Software Educativo. El

experto en el tema y los redactores, definen la amplitud de los contenidos temáticos específicos que deberán ser mostrados a los alumnos.

6-Elección del tipo de software a desarrollar: En el momento de elegir el tipo de software a desarrollar es preciso tener presente los niveles de complejidad de las áreas de aprendizaje.

7-Diseño de interfaces: La interfaz es un punto crucial, ya que a través de ella se lleva a cabo la comunicación entre el usuario y la computadora. Se tienen en cuenta las especificaciones definidas hasta el momento y las consideraciones didácticas expuestas en la definición de necesidades. El desarrollador debe mostrar maquetas al equipo de trabajo.

8-Definición de las estructuras de evaluación: Lograr que los alumnos aprendan los contenidos establecidos dentro de la planeación didáctica del curso y debe ir a la par las formas de evaluación de los contenidos para que el maestro pueda evaluar los aprendizajes.

9-Elección del ambiente de desarrollo: La delimitación del campo de aplicación del Software Educativo esté perfectamente definida, ya que cada desarrollador deberá buscar la herramienta que le permita involucrar todas las peticiones de los usuarios potenciales.

10- Creación de una versión inicial: Se debe comenzar a planificar los aspectos de implementación y realizar la implementación en sí.

11-Prueba de campo: La primera versión del sistema debe ser puesta a prueba frente al equipo de trabajo para su evaluación y rectificación de características. Una vez que se detecten los posibles errores u omisiones, debe realizarse nuevamente una versión con los problemas corregidos.

12-Mercadotecnia: En este caso que el producto se ha diseñado para comercializarlo, en este paso debe hacerse un recuento de características de mercadotecnia que harán que el producto sea vendible.

13-Entrega del producto final: Debe presentarse un producto final a los usuarios potenciales, el cual debe tener de apoyo documentado en características de instalación, operación.

En la metodología desarrollada por Figueroa (2009), propone un ciclo de vida dividido en dos etapas. En la primera etapa se contempla la definición de requisitos y el análisis y diseño preliminar, durante los cuales se determinan en forma global las características que se pretende alcanzar con el producto, los

requisitos pedagógicos, de comunicación y la arquitectura sobre la cual se construirá el software. Esta primera etapa se concluye con un plan de iteraciones las cuales se programan teniendo cuidado de que el producto que se libera al término de cada una está didácticamente completo, es decir que cubre completamente algunos de los objetivos didácticos del software. Una vez establecidos estos lineamientos, inicia la segunda etapa, en la cual se procede a desarrollar el producto, de modo que el equipo toma cada iteración, la diseña, la construye, la prueba y la implementa, evaluando al final la conveniencia de proseguir con subsecuentes iteraciones hasta obtener un producto completo.

Las fases propuestas para la etapa de definición son: la fase conceptual, durante la cual se identifican los requerimiento del sistema, se conforma el equipo de trabajo y se elabora el plan de desarrollo; la fase de análisis y diseño inicial, en la que se propone la arquitectura que servirá de base para la solución del problema y se establecen las características pedagógicas y de comunicación que regirán el desarrollo del software; finalmente la fase de plan de iteraciones, en la cual se divide el proyecto en partes funcionales que permitan mejor control en su desarrollo. En la etapa de desarrollo se tienen: la fase de diseño computacional, en la que se realizará un diseño computacional detallado de un incremento específico del software; la fase de desarrollo, durante la cual se implementa la arquitectura en forma incremental (iteración por iteración); y la fase de despliegue, donde se realiza la transición del producto ejecutable al usuario final. Estas tres últimas etapas se repiten iterativamente para cada incremento del software.

Estas metodologías a pesar de ser bastantes completas no poseen en ninguna de sus etapas un proceso que es fundamental en el de desarrollo de cualquier software como es el caso de la gestión de los riesgos, un proceso que permite tener un control tanto del proceso administrativo como de la elaboración del software y de tener documentado los posibles problemas que puedan aparecer como la solución a estos.

1.1.4- Definiciones y caracterización de la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares

El autor se basa para realizar esta investigación en la metodología empleada por Lisandra Morales (2011) que está compuesta por 5 etapas y contiene una serie de actividades específicas para los softwares educativos. Estas etapas son:

1. **Establecer los contextos del software:** En esta etapa se especifican los elementos de mayor importancia que fueron seleccionados en la etapa de planificación, dentro de los cuales no deben faltar los aspectos didácticos y pedagógicos, después de una buena comunicación entre los integrantes del equipo de desarrollo del software educativo y el cliente, con el propósito de desarrollar una estructura para la identificación del riesgo posteriormente. El contexto se dividirá en dos tipos: *contexto educacional y tecnológico.*
2. **Identificar y categorizar los riesgos:** Teniendo bien definidos los objetivos de la organización y el proyecto, estableciendo correctamente los contextos definidos en la etapa anterior, se realizará el proceso de identificación de los riesgos posibles: donde pueden aparecer riesgos, saber qué tanto repercute en el proyecto para categorizarlos cualitativamente uno a uno.
3. **Evaluar y proyectar los riesgos:** Se debe calcular el factor de riesgo con la fórmula correspondiente teniendo el valor de su probabilidad y el costo de su consecuencia en caso de que ocurra, para luego representarlo en el nivel de referencia y en la línea de corte para tener una visión aún más clara de cuánto puede afectar al proyecto.
4. **Tratamiento de los riesgos:** Después de saber cuáles son los riesgos que representan un peligro inminente en el proyecto se debe tratar de mitigar llevando a cabo un Plan de Acción para cada riesgo.
5. **Control y supervisión de los riesgos:** Cuando se escogen los mecanismos y acciones adecuadas, este proceso debe controlarse y ser supervisado a través de todo el proceso de planificación del proyecto.

Para cuantificar el nivel de incertidumbre y el grado de pérdidas se analizan los riesgos, esto se realiza considerando las diferentes categorías existentes, ya que estos pueden aparecer en cualquier etapa de desarrollo del proyecto. Aunque la categorización de los riesgos parece un proceso sencillo, algunos de ellos son difíciles de predecir. Algunas de estas categorías son:

Los **riesgos técnicos** identifican potenciales problemas de diseño, implementación, verificación, interfaz y mantenimiento. Si estos problemas llegasen a volverse una realidad, la implementación puede volverse difícil o imposible.

Los **riesgos del proyecto** amenazan el plan del proyecto, si llegara a volverse una realidad el calendario del proyecto varía y el costo aumenta. Identifican potenciales problemas de presupuesto, personal, calendario, recursos, participantes y requisitos.

Los **riesgos empresariales(negocio)** amenazan la viabilidad del software que se va a construir y con frecuencia ponen en peligro el proyecto.

Otra categorización general de los riesgos es la propuesta por Charette (1989). Los **riesgos conocidos** son aquellos que pueden descubrirse después de una evaluación del plan de proyecto, del entorno empresarial o técnico donde se desarrolla el proyecto. Los **riesgos predecibles** se extrapolan de la experiencia de proyectos anteriores. Los **riesgos impredecibles** pueden ocurrir, pero son extremadamente difíciles de identificar por adelantado.

Según Pressman (2010) la identificación de los riesgos es un intento sistemático para especificar amenazas al plan del proyecto. Existen dos tipos distintos de riesgos para cada una de las categorías que se presentaron anteriormente:

Riesgos genéricos: son una amenaza potencial a todo proyecto de software.

Riesgos específicos del producto: pueden identificarse solamente por quienes tienen clara comprensión de la tecnología, el personal y el entorno específico del software que se construye.

Un método para identificar los riesgos es realizando una lista de los elementos de los riesgos. De esta forma se puede enfocarse sobre algún subconjunto de riesgos conocidos y predecibles en las siguientes subcategorías genéricas según plantea Pressman (2010):

Tamaño del producto: Son los riesgos asociados con el tamaño global del software que se va a construir o modificar.

Impacto empresarial: Son aquellos riesgos que están asociados con restricciones impuestas por la administración o por el mercado.

Características de los participantes: Son los riesgos asociados con la sofisticación de los participantes y con la habilidad de los desarrolladores para comunicarse con los participantes de forma oportuna.

Definición del proceso: Son los riesgos asociados con el grado que se definió el proceso de software y la manera como continua la parte de la organización desarrolladora.

Entorno de desarrollo: Son los riesgos asociados con la disponibilidad y calidad de las herramientas por usar para construir el producto.

Tecnología por construir: Son los riesgos vinculados con la complejidad del sistema que se va a construir y con la tecnología que se incluye en el sistema.

Tamaño y experiencia del personal: Son los riesgos asociados con la experiencia técnica y de proyecto global de los ingenieros de softwares que realizaran el trabajo.

El punto de vista de la Fuerza Aérea de los Estados Unidos (AFC 1988), requiere de un gestor de proyecto para que identifique los controladores del riesgo que afectan los componentes de riesgo de software: costo, rendimiento, soporte y calendario. Definen los componentes de riesgo de la siguiente forma:

Riesgo en rendimiento: El grado de incertidumbre con el que el producto encontrará sus requisitos y se adecue al uso pretendido.

Riesgo de costo: Grado de incertidumbre que mantendrá el presupuesto del proyecto.

Riesgo de soporte: El grado de incertidumbre del software para corregirse, adaptarse y ser reformado.

Riesgo de calendario: El grado de incertidumbre de que el calendario del proyecto se mantendrá y de que el producto se entregue a tiempo.

El impacto de cada controlador de riesgo sobre el componente de riesgo se divide en cuatros categorías de impacto: despreciable, marginal, crítico y catastrófico. Las siguientes preguntas fueron obtenidas mediante una entrevista a diferentes gestores de proyectos experimentados en lugares diferentes del mundo (Keil 1998). Las preguntas están ordenadas de acuerdo a la importancia para el proyecto. Pressman (2010) en su metodología hace referencia a estas preguntas:

¿Los gestores de software y de cliente se reunieron formalmente para apoyar el proyecto?

¿Están completamente entusiasmados los usuarios finales con el proyecto y con el sistema/proyecto a construir?

¿El equipo de ingeniería del software y sus clientes entienden por completo los requisitos?

¿Los clientes se involucran plenamente en la definición de los requisitos?

¿Los usuarios finales tienen expectativas realistas?

¿Es estable el ámbito del proyecto?

¿Tiene el ingeniero de software el conjunto adecuado de habilidades?

¿Son estables los requisitos del proyecto?

¿Tiene experiencia el equipo del proyecto con la tecnología a implementar?

¿Es adecuado el número de personas del equipo del proyecto para realizar el trabajo?

¿Están de acuerdo todos los clientes/usuarios en la importancia del proyecto y en los requisitos del sistema/producto a construir?

Estas preguntas tienen un peso importante ya que si algunas de estas preguntas se responden de manera negativa se debe establecer sin falta pasos de mitigación, gestión y monitoreo, es decir mientras mayor sean las respuestas negativas mayor es el grado de riesgo en el proyecto.

Según Pressman (2010) la proyección del riesgo, la denomina también como estimación de riesgo, intenta calificar el riesgo de dos formas: la probabilidad de que el riesgo sea real y la consecuencia de los problemas asociados con el riesgo, en caso de que ocurra. En sus estudios plantea que debería ser utilizado para la tabla de riesgo un modelo de una hoja de cálculo, y así facilitar el manejo y ordenación de las entradas. La tabla se estructura de la siguiente manera: Se listan los riesgos en la primera columna, cada uno de ellos es categorizado en la segunda columna (por ejemplo: PS implica riesgo en el tamaño del proyecto). En la siguiente columna se coloca la probabilidad de aparición del riesgo (este valor debe realizarse individualmente por cada integrante del equipo). A continuación, se valora el impacto de cada riesgo, usando las categorías antes descritas, por la Fuerza Aérea de Estados Unidos. Por último, agregar una columna etiquetada RSGR, que contiene una referencia que apunta hacia un Plan de reducción, supervisión y gestión del riesgo o alternativamente a un informe del riesgo desarrollado para todos los que se encuentran por encima de la línea de corte.

La exposición al riesgo en general, se realiza usando la siguiente fórmula (Hall 1998):

ER= P X C

ER= Exposición del riesgo en general.

P= Probabilidad de ocurrencia.

C= Costo en caso de que ocurra.

Según Lisandra Morales (2011), expresó " Tho (2005) describe la exposición del riesgo como el producto de la probabilidad y la magnitud del resultado indeseable de la relación, como se expresa en la siguiente ecuación.

RE = Pr (UO) * L (UO)

Donde RE es la exposición de riesgo; **Pr (UO)** es la probabilidad de un resultado indeseable; y **L (UO)** es la magnitud de pérdida debido al resultado indeseable. Considerado las variables en la ecuación, si la probabilidad (de pérdida) sostuvo valores constantes, la exposición de riesgo es proporcional a la pérdida, y viceversa. Sin embargo, ni inconstante (la magnitud de pérdida ni la probabilidad de pérdida) es constante encima de un periodo de tiempo, porque son dinámicos, es decir varían en el tiempo. Con el tiempo, la exposición de riesgo total se representa por la ecuación siguiente:

Total risk exposure = Total (Probability of loss *Magnitude of loss).

(Valor del riesgo total) = Total (Probabilidad de pérdida * Magnitud de la pérdida)".

Una forma de **refinamiento del riesgo** es la propuesta por Gush (1994) y Pressman (2010) es la que tiene el formato condición-transición-consecuencia (CTC), es decir el riesgo se enuncia de la siguiente manera:

Dado que <condición> entonces hay preocupación porque (posiblemente) <consecuencia>. Esta condición puede ser refinada por subcondiciones, las condiciones de las mismas se mantienen iguales, pero el refinamiento ayuda aislar los riesgos subyacentes y pueden conducir a análisis y respuestas más sencillas.

Todo este análisis de riesgo hasta el momento solo tiene una meta que es ayudar al equipo de proyecto a tomar acciones para lidiar con el riesgo. Una estrategia efectiva debe ser evitar el riesgo, monitorearlo, y planificar el plan de contingencia. Conforme avanza el proyecto debe seguir realizándose el monitoreo de riesgos. El gestor de proyecto monitorea si el riesgo se vuelve más o menos probable, también debe dar seguimiento a la efectividad de los pasos de mitigación del riesgo. Las planificaciones de contingencia suponen que los esfuerzos de mitigación fracasaron y que el riesgo se convirtió una realidad.

Según Pressman (2010) el **plan de mitigación, monitoreo y manejo de riesgo** (MMMR) documenta todo el proceso realizado como parte del análisis de riesgos y el gestor de proyecto lo usa como parte del plan de proyecto general. Plantea que algunos equipos de software no desarrollan un documento MMMR formal.

El autor realizó algunos cambios en estas etapas ya que decidió que era necesario agregar algunos puntos que no fueron tratados en esta metodología como es el caso de las preguntas realizadas para la identificación de los riesgos donde no se tenían en cuenta a fondo todo lo referente a la parte pedagógica y otras preguntas que no eran necesarias, también se incorporó al proceso la clasificación de los riesgos para tener mejor especificado todo el proceso además de la categorización y en dependencia de las categorías y las clasificaciones se va a ver cuánto repercute estos riesgos en el futuro del proyecto. En el anexo II se muestran todas las preguntas que se decidieron establecer.

1.2- Métodos de la investigación

1.2.1- Métodos teóricos empleados

-**Histórico-Lógico:** Permitió esclarecer en un orden cronológico las etapas de desarrollo del problema a solucionar y como llevar a cabo la gestión de los riesgos durante el proceso de desarrollo de los softwares educativos, su definición, características y desarrollo.

-Análisis-Síntesis: Este se precisó durante la revisión bibliográfica, permitiendo descomponer lo complejo del análisis de los riesgos en pedazos y cualidades, la división mental del todo en sus múltiples relaciones para luego unir las partes analizadas, descubrir las relaciones y características generales entre ellas.

-Inducción-Deducción: Tuvo un papel significativo en el proceso de verificación de la supuesta partida desde las definiciones de riesgos hasta la definición de riesgo en el software educativo. En el momento de decidir que metodologías o herramientas emplear para realizar la investigación presente.

1.2.2- Métodos empíricos empleados

-Encuestas: Aportó los datos esenciales en la investigación principalmente al inicio a la hora de determinar cómo realizan los procesos de gestión de los

riesgos en varios centros tecnológicos que se dedican a desarrollar softwares educativos y a la hora de realizar el criterio de expertos en la última etapa.

-Entrevistas: Se permitió conocer las sugerencias del personal calificado con respecto al tema y al inicio a la hora del levantamiento de requisitos para efectuar una exploración preliminar del problema a investigar, de esta forma desarrollarla a gusto del cliente. También se obtuvo información relevante realizar de manera correcta el proceso de gestión de los riesgos en estos softwares y al realizar el criterio de expertos.

1.3- Herramientas, tecnologías y metodologías de desarrollo

Para el desarrollo de esta investigación, se hace necesario el estudio y selección de herramientas, tecnologías y metodologías de desarrollo con el propósito de darle cumplimiento al objetivo general. A continuación, se describen aspectos a tener en cuenta.

1.3.1- Metodología de desarrollo empleada

Extreme Programming (XP):

Extreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, entre otras) aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías “clásicas” es que son demasiado burocráticas (Armendáriz Barreno and Guaraca 2013). Hay tanto que hacer para seguir la metodología que, a veces, el ritmo entero del desarrollo se retarda. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de “Metodologías Ágiles”. Estos nuevos métodos buscan un punto medio entre la ausencia de procesos y el abuso de los mismos, proponiendo un proceso cuyo esfuerzo sea recompensado por un sistema con calidad en tiempo. Los valores que defiende la metodología fueron definidos por Kent Beck (2004) de la siguiente forma:

Comunicación: Muchos de los problemas que existen en proyectos de software (así como en muchos otros ámbitos) se deben a problemas de comunicación entre las personas. La comunicación permanente es fundamental en XP. Dado que la documentación es escasa, el diálogo frontal, cara a cara, entre

desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.

Simplicidad: XP, como metodología ágil, apuesta a la sencillez, en su máxima expresión. Sencillez en el diseño, en el código, en los procesos, entre otras. La sencillez es esencial para que todos puedan entender el código, y se trata de mejorar mediante recodificaciones continuas.

Retroalimentación: La retroalimentación debe funcionar en forma permanente. El cliente debe brindar retroalimentación de las funciones desarrolladas, de manera de poder tomar sus comentarios para la próxima iteración, y para comprender, cada vez más, sus necesidades. Los resultados de las pruebas unitarias son también una retroalimentación permanente que tienen los desarrolladores acerca de la calidad de su trabajo.

Coraje: Cuando se encuentran problemas serios en el diseño, o en cualquier otro aspecto, se debe tener el coraje suficiente como para encarar su solución, sin importar que tan difícil sea. Si es necesario cambiar completamente parte del código, hay que hacerlo, sin importar cuanto tiempo se ha invertido previamente en el mismo. (Joskowicz 2008)

Las características principales de la programación extrema son las siguientes:

- Desarrollo iterativo incremental.
- Los requerimientos se expresan como escenario (también conocidos como historias de usuarios), los cuales se dividen e implementan como una serie de tareas.
- Se describen pruebas unitarias para cada tarea. No se considerará una unidad de código como correcto hasta que no pase todos los test escritos. Se recomienda antes que el código de la tarea.
- Los desarrolladores trabajan en parejas. Se da prioridad a un código de mayor calidad discutido y analizado desde dos puntos de vista distintos, que a una mayor productividad.
- Integración del equipo de programación con el cliente. Se recomienda que un representante trabaje de forma conjunta con el equipo de desarrollo.
- Refactorización del código, es decir, reescribir parte del código, mantenibilidad y capacidad de reutilización.

- Propiedad compartida del código, es decir, que la responsabilidad del desarrollo de cada módulo se divide entre varios grupos de trabajo, de forma que cualquier miembro del equipo puede modificar e introducir mejoras en cualquier parte. Gracias a esto la mayoría de los errores acaban detectándose.
- Simplicidad en el código. La mejor manera de que algo funcione es que sea simple, al menos en su inicio. Más adelante, cuando todo funciona bien, se pueden añadir funciones más complejas.
- Comunicación. Es una premisa de desarrollo ágil, una buena comunicación entre los integrantes de un equipo de desarrollo evita mal entendidos que puedan afectar posteriormente el desarrollo del proyecto.

Justificación de uso en el proyecto:

Para el desarrollo del presente trabajo el autor decidió aplicar la Metodología de XP, ya que brinda una gran garantía a la hora de implementar el sistema, esta se centra en realizar un producto con calidad y lo más rápido permitido, también se puede realizar cambios durante el proceso de desarrollo y que el cliente forma parte indiscutible en el equipo de desarrollo.

1.3.2-Tendencias tecnológicas a considerar

Arquitectura cliente-servidor: esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un Sistema Gestor de Base de Datos (SGBD). Por otro lado, los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red (Peña 2007)

Modelo-Vista-Controlador: Es un patrón de arquitectura de software que separa los datos y la lógica del negocio de una aplicación, de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (Tedeschi 2013)

De manera genérica, los componentes de MVC se podrían definir como sigue:

- El **Modelo:** Es la representación de la información con la cual el sistema opera. Gestiona todos los accesos a dicha información, tanto consultas como actualizaciones. Implementa también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la vista aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.
- El **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). Se podría decir que el controlador hace de intermediario entre la vista y el modelo.
- La **Vista:** Presenta el modelo (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho modelo la información que debe representar como salida. (González 2017)

1.3.3- Tendencias y herramientas

Lenguajes de Programación Web: Es un lenguaje que está diseñado para realizar procesos que pueden ser ejecutados por cualquier computadora. Estos lenguajes tienen sus propias especificaciones, están formados por dos grandes grupos los del lado del cliente que son los encargados de interactuar directamente con el cliente a través del navegador y los del lado del servidor que son los que se ejecutan en el servidor web, justo antes de que se envíe la página a través de la red al cliente, es decir tienen la tarea de dar respuesta a esas peticiones del cliente.

Del Lado del Cliente

Lenguaje de Marcas de Hipertexto 5 (HTML5): El “Hypertext Markup Language”, más conocido como HTML, es un lenguaje de programación que es interpretado en la parte del cliente, que describe el formato que tendrá el

contenido del documento. Es un estándar que es referencia para la elaboración de páginas web en todas sus versiones definiendo una estructura básica de lo que será el documento, estableciendo su contenido como son textos, párrafos, imágenes y formularios. HTML desde sus orígenes en 1990, y hasta la actualidad está lejos del estancamiento por lo que su evolución constante no es para ser ignorada, y así como todo en la vida debe apartarse para dar su lugar a las cosas nuevas que se avecinan, llegó HTML5 para desplazar a HTML4 e imponerse en el mundo de la web con sus nuevas ventajas y componentes de estreno, para resolver problemas tradicionales de la web.

HTML5 a pesar de basarse en gran parte sobre su predecesor, tiene novedades que lo hacen ser insustituible hasta el momento en el mundo que corre actualmente en el maquetado de páginas web. Su capacidad para reducir la dependencia de *plug-ins* en los navegadores para visualizar determinadas páginas, y un ejemplo práctico de esto es la mejora que brindó para los dispositivos que de manera nativa no soportaban la tecnología de animaciones gráficas FLASH, ni *plug-ins* para esta función. A la lista de novedades se suman también las funciones de geolocalización, las etiquetas para audio y video, que como antes se planteaba resuelve de forma nativa la de pendencia de *plug-ins*, y otra característica que por muy pequeña que parezca no lo es y son las funcionalidades del *drag* y el *drop*, las cuales juegan un papel importante en la creación y edición de los mapas conceptuales, para la colocación de los conceptos. También toma la lista de etiquetas existente en el HTML y le suma a esta, nuevas que permiten un mayor nivel de organización y brindan más facilidades a la hora de crear componentes en la web a los desarrolladores. (Perera 2016)

Hojas de Estilo en Cascada 3 (CSS3): Las hojas de estilo en cascada es un lenguaje que permite el diseño gráfico. Oficialmente CSS nada tiene que ver con HTML5. CSS no es parte de la especificación y nunca lo fue. Este lenguaje es, de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. Al comienzo, atributos dentro de las etiquetas HTML proveían estilos esenciales para cada elemento, pero a medida que el lenguaje evolucionó, la escritura de códigos se volvió más compleja y HTML por sí mismo no pudo más satisfacer las demandas de diseñadores. En consecuencia, CSS pronto fue adoptado como la forma de separar la estructura de la presentación.

Desde entonces, CSS ha crecido y ganado importancia, pero siempre desarrollado en paralelo, enfocado en las necesidades de los diseñadores y apartado del proceso de evolución de HTML. CSS es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, entre otros. En este momento las nuevas incorporaciones de CSS3 están siendo implementadas en las últimas versiones de los navegadores más populares. (Gauchat 2012)

JavaScript (JS): Javascript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio. Esta mejorada capacidad permitió superar viejas limitaciones de rendimiento y confirmar el lenguaje Javascript como la mejor opción para la web. Para aprovechar esta plataforma de trabajo ofrecida por los nuevos navegadores, Javascript fue expandido en relación con portabilidad e integración. A la vez, interfaces de programación de aplicaciones (APIs) fueron incorporadas por defecto en cada navegador para asistir al lenguaje en funciones elementales. Estas nuevas APIs (como Web Storage, Canvas, y otras) son interfaces para librerías incluidas en navegadores. La idea es hacer disponible funciones a través de técnicas de programación sencillas y estándares, expandiendo el alcance del lenguaje y facilitando la creación de programas útiles para la web.(Gauchat 2012)

Del lado del Servidor

Preprocesador de Hipertexto(PHP): PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. PHP es un acrónimo de "Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP. (Achour 2007)

Justificación en el uso del Proyecto:

El autor decidió que para el desarrollo de la solución propuesta este lenguaje ya que permite la posibilidad de usar Programación Orientada a Objetos. Además, es uno de los lenguajes más utilizados en el desarrollo de las aplicaciones web con acceso a información almacenada bases de datos con una extensa capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destacando su conectividad con MySQL. Es multiplataforma debido a que puede ser utilizado en la mayoría de los sistemas operativos y servidores web actuales y, debido sus características, hace posible que el cliente interactúe con una página rápida, eficiente y segura, capaz de mostrar y procesar información de forma eficiente.

Framework: Es un ambiente de trabajo para desarrollo, es una herramienta para facilitar el trabajo a los programadores, pueden incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Estos representan una arquitectura de software que modela las relaciones generales de las entidades.

Symfony: Es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Una de estas características es la separación de la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, entre otras.) como en plataformas Windows (Potencier 2008)

Justificación del uso del proyecto:

El autor seleccionó Symfony de acuerdo las características que presenta:

▪ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)

▪ Independiente del sistema gestor de bases de datos

▪ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos

▪ Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional

▪ Sigue la mayoría de mejores prácticas y patrones de diseño para la web

▪ Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo

▪ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo

▪ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros (Potencier 2008)

Bootstrap: Es un framework que fue desarrollado por Twitter para diseñar sitios y aplicaciones web. Permite crear interfaces web con CSS, Javascript, este de código abierto. Es compatible con todos los tipos de navegadores, contiene estilos que son muy fácil de configurar ya sean menús desplegables, formularios, JQuery para ofrecer ventanas, entre otras opciones. Bootstrap logra que los diseños sean iguales en cualquiera de ellos. Por estas razones y la bondad de poseer predefinidos un conjunto de elementos de suma importancia a la hora de ahorrar tiempo y esfuerzo cuando se crean las vistas de la aplicación. (Raible 2013)

Doctrine: Es un asignador objeto relacional (ORM) para PHP 5.3.0 que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL por Data Base Abstraction Layer). La principal tarea de los asignadores objeto relacionales es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos. Una de las características clave de Doctrine es la opción de escribir las consultas de base de datos en un dialecto SQL propio orientado a objetos llamado Lenguaje de Consulta Doctrine (DQL por ***Doctrine Query Language***), inspirado en Hibernates HQL. (Perera 2016)

Entorno de desarrollo integrado: Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el

desarrollo de software. Un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador.

JetBrains PhpStorm: Es un ambiente de desarrollo integrado para el desarrollo de programas informáticos. Una de sus características que lo hacen imponerse es el resaltado de sintaxis del lenguaje, ya que facilita el trabajo al permitir distinguir fácilmente los diferentes Tokens que tiene el lenguaje, como sus funciones, comentarios y variables, esta es una ayuda visual necesaria para un mayor entendimiento del código usado por el desarrollador. También el completamiento de código es otro factor a tener en cuenta, porque cuándo se desarrolla un producto de software la velocidad de escritura es vital pero también las ayudas contextuales de escritura que brinde el IDE, puesto que evita tener que escribir más de lo necesario, y es mucho mejor si también incluyen plantillas para la documentación de código en comentarios. Además, la facilidad de búsqueda y remplazo de código ahorran un tiempo precioso a la hora de encontrar funciones o segmentos de códigos, los cuales pueden ser remplazados de una vez o en todas sus apariciones si así es deseado. Al poseer una gran cantidad de facilidades es necesario utilizarlo como apoyo para desarrollar esta aplicación. (Perera 2016)

El autor seleccionó esta herramienta teniendo en cuenta la gran cantidad de facilidades que contiene para el desarrollo de este software.

Servidor web Apache: El servidor web HTTP seleccionado para la aplicación es Apache, es usado principalmente para enviar páginas web estáticas y dinámicas en la WWW, así como PHP y MySQL, también es liberado bajo licencia Open Source y es multiplataforma entre ellas las más populares como Unix, Microsoft Windows y Macintosh. Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, entre otras.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP y la noción de sitio virtual.(MORINE 2013)

Gestor de base de datos: Es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. La función principal es tener bien detallado el conjunto de datos.

MySQL: Es un gestor de base de datos sencillo de usar e increíblemente rápido, sólido y flexible. Es idóneo para la creación de bases de datos con acceso desde páginas web dinámicas, así como para la creación de cualquier otra solución que implique el almacenamiento de datos posibilitando realizar múltiples y rápidas consultas. Está desarrollado en C y C++, facilitando su integración en otras aplicaciones desarrolladas también en esos lenguajes. Es un sistema cliente/servidor, por lo que permite trabajar como servidor multiusuario y de subprocesamiento múltiple, o sea, cada vez que se crea una conexión con el servidor, establece un proceso para manejar la solicitud del cliente, controlando así el acceso simultáneo de un gran número de usuarios y asegurando el acceso a solo los autorizados. Es uno de los sistemas gestores de bases de datos más utilizado en la actualidad, utilizado por grandes corporaciones como Yahoo! Finance, Google, Motorola, entre otras. Es gratis para aplicaciones no comerciales. Dentro de las principales características se encuentra, la gran portabilidad entre sistemas, soporta gran cantidad de tipos de datos para las columnas y hasta 32 índices por tabla, manteniendo un buen nivel de seguridad en los datos gestiona los usuarios y contraseña, dispone de API's en gran cantidad de lenguajes (Python, C, C++, Java, PHP) y aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multi-hilo. (Schwartz 2012)

Conclusiones

Durante el desarrollo de este capítulo se determinaron los fundamentos teóricos acerca de los softwares educativos; la metodología planteada para la gestión de

la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.

Se realizó un análisis de las principales metodologías para el desarrollo de softwares educativos en las cuales no se detectaron actividades relacionadas con la identificación, análisis y mitigación de los riesgos.

El estudio de las metodologías, herramientas, y tecnologías que pudieran ser emplearon para la solución del problema permitió seleccionar las siguientes: XP como metodología de desarrollo, Apache como servidor web, PHP como Lenguaje de programación, Bootstrap y Symfony como framework de desarrollo, JavaScript como lenguaje de programación del lado del cliente y MySQL como gestor de bases de datos.

Capítulo 2. Descripción de la solución propuesta.

Introducción

En este capítulo, apoyándose en la metodología de desarrollo de software Programación Extrema (XP), se afrontan los elementos necesarios para describir la solución propuesta. Debido a la metodología se han usado las Historias de Usuarios para el levantamiento de requisitos, con la cual también se realiza la planificación inicial del proyecto.

Este capítulo tiene como objetivo describir la solución obtenida a la problemática planteada en la introducción, con la ventaja de la incorporación del cliente como un miembro del equipo de desarrollo.

2.1- Descripción de la solución

Se propone el desarrollo de un sistema web que permita la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos. Los usuarios que interactúan con el sistema tendrán los siguientes privilegios:

Administrador del sistema: tiene la posibilidad de visualizar las trazas de cada usuario registrado, como de crear nuevos usuarios y configurar el sistema propuesto.

Gestor de Proyecto: tiene la posibilidad en caso de ser necesario cualquier cambio en cuanto, a las clasificaciones, las categorías, preguntas, entre otras.

Evaluador: tiene la posibilidad de acceder a los proyectos para realizar la evaluación en función de observar los posibles riesgos que puedan tener los proyectos.

2.2- Planificación

Para tener una solución del problema es importante realizar una planificación, que permita poseer desde el momento inicial posibles soluciones y organizar el modelado del sistema. Se emplean las historias de usuarios para modelar y organizar las ideas del cliente, obteniendo un punto de partida para el resto de la planificación del proyecto. Es necesario estimar el tiempo y una planificación inicial de las entregas, hay que tener en cuenta que pueden surgir nuevos requisitos, esto puede afectar esa planificación inicial.

2.3- Grupo de trabajo y Roles

Para que el trabajo se realice correctamente es necesario delegar responsabilidades y XP cubre esta función mediante la asignación de roles, los

cuales determinan el papel que desempeña cada persona en el trascurso del propósito. Existen diferentes roles con sus responsabilidades correspondientes, el jefe de proyecto es el responsable indiscutible de la dirección y organización en toda la vida de desarrollo del proyecto, el cliente, parte indisoluble del grupo de desarrollo, dentro del cual juega un papel indispensable, porque el objetivo principal de XP es la satisfacción del mismo y por esto ellos tienen voz y voto sólido en su desarrollo de software. Por otra parte, el programador será el encargado de la implementación de las funcionalidades del software que darán solución al problema a informatizar, quien tendrá que redactar junto al cliente las historias de usuarios, base de la solución brindada por la metodología XP.

En la siguiente tabla 1, quedan expuestos los roles asignados en el proyecto:

Miembros	Roles
Dr.C. Walfredo González **Ing. Dayana O. Hernández Revilla**	Jefe de Proyecto, tester, cliente
Ediel Pérez Alvarez	Programador, tester

Tabla 1:Miembros y roles del proyecto. *Fuente: Elaboración del autor.*

2.4- Diagrama de proceso del negocio

Permite el modelado de procesos de negocio, en un formato de flujo de trabajo (*workflow*):

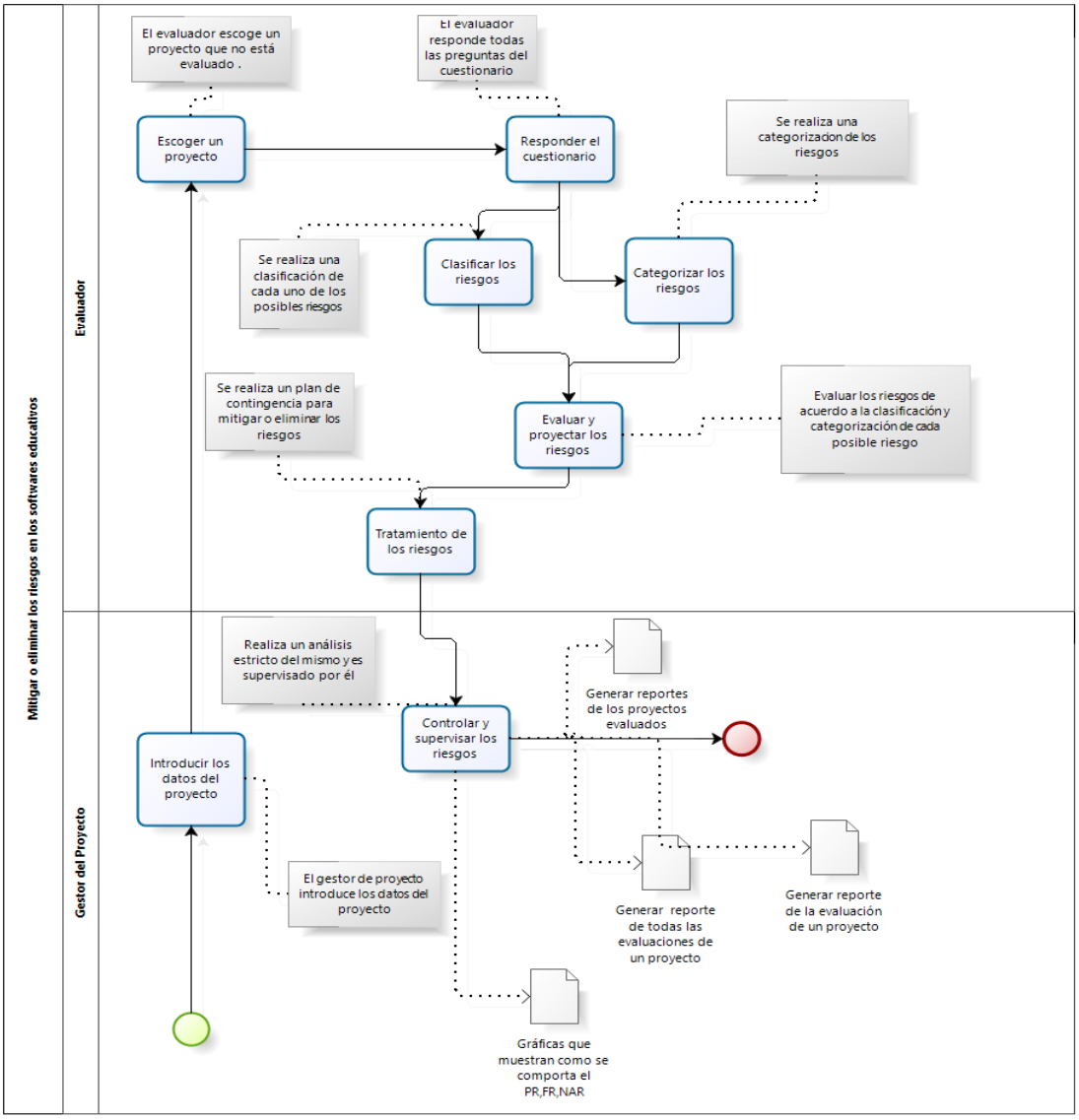


Figura 1: Diagrama de proceso del negocio. Fuente: Elaboración del autor.

2.5- Requerimientos Funcionales

Los requerimientos funcionales de un software son las respuestas del sistema a la acción de los usuarios. A continuación, se expondrán los requisitos funcionales referentes al proyecto:

1. Diseño y Creación de la Base de Datos.
   - Diseño de la Base de Datos.
   - Creación de la Base de Datos.
2. Autenticarse.
3. Gestionar Usuarios.

- Mostrar usuarios.
- Insertar usuarios.
- Editar usuarios.
- Eliminar usuarios.

4. Trazas.
    - Listar trazas.
    - Buscar trazas.
5. Gestionar clasificación de los riesgos.
    - Listar clasificación de los riesgos.
    - Insertar clasificación de los riesgos.
    - Editar clasificación de los riesgos.
    - Eliminar clasificación de los riesgos.
6. Gestionar categorías de los riesgos.
    - Listar categorías de los riesgos.
    - Insertar categorías de los riesgos.
    - Editar categorías de los riesgos.
    - Eliminar categorías de los riesgos.
7. Gestionar evaluaciones.
    - Listar evaluaciones.
    - Insertar evaluaciones.
8. Gestionar plan de mitigación.
    - Insertar plan de mitigación.
    - Listar plan de mitigación.
9. Gestionar proyectos.
    - Listar proyectos.
    - Insertar proyectos.
    - Editar proyectos.
    - Eliminar proyectos.
    - Buscar proyectos.
10. Gestionar tipos de proyectos.
    - Listar tipos de proyectos.
    - Insertar tipos de proyectos.
    - Editar tipos de proyectos.

- Eliminar tipos de proyectos.
- Buscar tipos de proyectos.

11. Gestionar preguntas.
    - Listar preguntas.
    - Insertar preguntas.
    - Editar preguntas.
    - Eliminar preguntas.
12. Gestionar rangos.
    - Listar rangos.
    - Insertar rangos.
    - Editar rangos.
    - Eliminar rangos.
13. Gestionar impactos.
    - Listar impactos.
    - Insertar impactos.
    - Editar impactos.
    - Eliminar impactos.
14. Gestionar probabilidades.
    - Listar probabilidades.
    - Insertar probabilidades.
    - Editar probabilidades.
    - Eliminar probabilidades.
15. Listar notificaciones.
16. Gestionar valoraciones.
    - Listar valoraciones.
    - Insertar valoraciones.
    - Editar valoraciones.
    - Eliminar valoraciones.
17. Generar reporte de los proyectos evaluados.
18. Generar reporte de las evaluaciones de un proyecto.
19. Generar reporte de la evaluación de un proyecto.
20. Generar reporte por Peso de Riesgo.
21. Generar reporte por Factor de Riesgo.

22. Generar reporte por Nivel de Atención al Riesgo.

2.6- Requisitos no funcionales

Son aquellos requisitos que se encargan de especificar características como aquella que necesita el software para su funcionamiento, costo, usabilidad y rasgos de manera general establecidos por el cliente.

2.6.1- Requisitos de Hardware

Servidor:

- El servidor donde se instale la aplicación debe contar con sistema operativo Windows 7 o superior, o Linux, pero preferentemente Debian 8, por lo que para Windows necesita 512 MB de RAM como mínimo.

Cliente:

- Display con resolución 1366 x 768.
- Procesador Pentium IV o superior, 512 MB de RAM como mínimo.
- Windows 7 o superior, cualquier versión de Linux.
- Conexión de red.

2.6.2- Requisitos de Software

- Servidor con Apache 2.4.10 o superior, PHP 5.6.29-0+deb8u1 (cli) build dec 14 2016 o superior.
- Servidor de Base de Datos: MYSQL 14.14 o superior.
- Mozilla Firefox (versión 57 o superior).

2.7- Estimación por Puntos de Función

Puntos de Función

Es una métrica que permite traducir en un **número** el tamaño de la funcionalidad que brinda un producto de software desde el punto de vista del usuario, a través de una suma ponderada de las características del producto.

Componentes:

EI: Procesos en los que se introducen datos y que suponen la actualización de cualquier archivo interno.

EO: Procesos en los que se envía datos al exterior de la aplicación.

EQ: Procesos consistentes en la combinación de una entrada y una salida, en el que la entrada no produce ningún cambio en ningún archivo y la salida no contiene información derivada.

ILF: Grupos de datos relacionados entre sí internos al sistema.

EIF: Grupos de datos que se mantienen externamente.

Una vez obtenidos los diferentes elementos del sistema se utilizan las tablas reflejadas en el **anexo III** para asignar pesos en función del número de atributos que tengan y el número de archivos a los que afecte.

Componentes	Cantidad de Componentes por su Peso	Total
EI (Entradas externas)	22 * 3	66
EO (Salidas externas)	6 * 4	24
EQ (Consultas)	15* 3	45
ILF (Ficheros Lógicos Internos)	19 * 10	190
ILF (Ficheros Lógicos Externos)	0	0

Tabla 2: Resultado de los Componentes por su peso. Fuente: Elaboración del autor.

Cálculo de los Puntos de Función sin Ajustar (PFSA)

Los PFSA se calculan como la suma de los productos de cada componente por su peso determinado en la tabla correspondiente.

PFSA=PFTe+PFTo+PFTq+PFtif+PFTef

PFSA = 66+ 24+ 45+ 190+ 0 = 325

Cálculo de los Puntos de Función Ajustados (PFA)

PFA = PFSA * [0.65 + (0.01 * ACT)]

Cálculo del Ajuste de Complejidad Técnica (ACT)

Para calcular el ACT se le va dando un valor entre 0 y 5 a cada Factor de Ajuste como se muestra en el **Anexo IV**. Cuando cada Factor tenga un valor, se suman todos y así obtenemos el ACT. A continuación, se muestra una **Tabla 3** como se refleja dicho procedimiento.

No. de Factor	Factor de Ajuste	0<valor<5
1	Comunicación de Datos	2
2	Proceso Distribuido	4
3	Objetivos de Rendimiento	3
4	Configuración de Explotación Compartida	2
5	Tasa de transacciones	2
6	Entrada de Datos en Línea	2

7	Eficiencia con el Usuario Final	3
8	Actualizaciones en Línea	1
9	Lógica de Proceso Interno Compleja	3
10	Reusabilidad del Código	2
11	Conversión e Instalación contempladas	2
12	Facilidad de Operación	3
13	Instalaciones Múltiples	2
14	Facilidad de Cambios	2
Total	**Ajuste de Complejidad Técnica(ACT)**	**33**

Tabla 3: Obtención del ACT Fuente: Elaboración del autor.

PFA=PFSA*[0.65+(0.01*ACT)]

PFA=325*[0.65+(0.01*33)]

PFA=325*0.98

PFA=318.5

Cálculo del Esfuerzo

Líneas de Código (LC)

LC = PFA * (Líneas por PF)

Para calcular las Líneas por PF o Líneas por Puntos de Función nos apoyamos en el **Anexo V**.

LC = 318.5 * 20

LC = 6370

Esfuerzo en Horas/Personas (E)

E=PFA/ (1/8 persona/hora)

E=318.5/0.125

E=2544 horas/personas

Tomando 24 días laborales en el mes y 8 horas productivas al día se obtiene 192 horas laborables al mes

Duración del proyecto en meses

2544 horas/personas/1 persona=2544 horas

DM=2544 horas/192 horas=13.25 meses aproximadamente 14 meses por el margen de error de la estimación por puntos de función.

Costo total del proyecto

CT=sueldo de 1 persona* cantidad de personas*DM

CT=250*1*14

CT=3500

Este cálculo permite al autor definir el plan de entregas y de iteraciones como se mostrará en el epígrafe que sigue.

2.8- Plan de Iteraciones

Teniendo las historias de usuario del sistema declaradas y la estimación del esfuerzo para implementar cada una de ellas se procede a realizar la planificación de la etapa de implementación del proyecto. Atendiendo a lo mencionado con anterioridad se decide realizar cinco iteraciones durante el desarrollo del proyecto. A continuación, se muestra el plan de iteraciones.

Figura 2: Plan de Iteraciones *Fuente: Elaboración del autor.*

2.9- Reuniones

La planeación en XP requiere de una revisión continua del plan de trabajo, a pesar de que en esta no se presenta una documentación extensa, se hace

hincapié en la elaboración de todas sus actividades para tener bien organizado el trabajo. Semanalmente se realizará una reunión con el equipo de trabajo para ver los avances de cada iteración y en caso de algún inconveniente se realizará otra reunión en la misma semana.

2.10- Plan inicial de entregas

Las entregas se realizan al terminar cada iteración para tener una mejor comunicación con el cliente, permitir un desarrollo del sistema más eficiente. En cada una de estas iteraciones se entregarán funcionalidades para ir corrigiendo los posibles errores que se encuentren.

En la planificación de un proyecto es difícil tener en cuenta todos los detalles desde un inicio y en consecuencia ocurren modificaciones al respecto. La poca experiencia del equipo de trabajo en la organización y los contratiempos aparecidos durante su desarrollo inciden en la programación de las iteraciones iniciales y por consiguiente en el plan de las entregas preliminar pactado con el cliente.

Este proyecto no quedó exento ante esta problemática pues sobre la marcha surgieron necesidades de modificar la base de datos en algunas ocasiones, también en el transcurso del proceso de desarrollo de la aplicación hubo variaciones que provocaron la inclusión de otras que no se tomaron en cuenta desde un principio. Ello demuestra que no es posible una planificación inicial de entregas inalterable. A partir de esto queda aclarado que el cliente es de gran importancia durante todo el proceso para lograr el éxito del proyecto, ya que a través de la comunicación continua se soluciona rápidamente cualquier duda relacionada con los requisitos.

2.11- Historias de Usuario

Las historias de usuarios son una técnica empleadas por XP (eXtreme Programming) para dejar claro los requisitos del software. Estas permiten ser usadas para el cálculo de la estimación del tiempo de desarrollo de la funcionalidad en punto. Se trata también de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

A continuación, se presentan las escalas equivalentes a la **prioridad en el negocio**: (IZQUIERDO 2014)

Alta: Asignada a las Historias de Usuario que corresponden a funcionalidades esenciales en el desarrollo del proyecto, a las que el cliente define como primordiales.

Media: Dada a las Historias de Usuario que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación directa sobre el proyecto que se esté desarrollando.

Baja: Se le otorga a las Historias de Usuario que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el proyecto en desarrollo.

Escala Nominal de Riesgo en Desarrollo: (Pressman 2010)

Alta: Cuando para la implementación de la Historia de Usuario se considera la posible existencia de errores que llevan a inoperatividad del código.

Media: Cuando pueden aparecer errores en la implementación de la Historia de Usuario que puedan retrasar la entrega de la versión.

Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

A continuación, se presenta en la Tabla 4 un resumen de las Historias de Usuario de la aplicación planificadas inicialmente, en las que queda definido el nivel de prioridad con el que deben darle solución a las HU, teniendo en cuenta el riego en desarrollo. Se define también la estimación del esfuerzo requerido (en semanas), que no es más que el tiempo en el que se concibió inicialmente el desarrollo de cada HU.

No.	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración	Entrega
1	Diseño y creación de la base de datos.	Alta	Alto	1	1	**1**
2	Autenticarse.	Media	Medio	1	1	**1**
3	Gestionar Usuarios	Media	Medio	0.5	1	**1**
4	Trazas	Media	Bajo	0.5	1	**1**

5	Gestionar clasificación de los riesgos	Alta	Medio	1	1	**1**
6	Gestionar categorías de los riesgos	Alta	Alto	1	2	**2**
7	Gestionar evaluación	Alta	Medio	0.5	2	**2**
8	Gestionar plan de mitigación.	Alta	Medio	0.5	2	**2**
9	Gestionar proyectos	Alta	Medio	1	2	**2**
10	Gestionar tipos de proyectos	Alta	Medio	1	2	**2**
11	Gestionar preguntas.	Alta	Medio	1	3	**3**
12	Gestionar rangos	Alta	Medio	1	3	**3**
13	Gestionar impactos	Alta	Medio	1	3	**3**
14	Gestionar probabilidades	Alta	Medio	1	3	**3**
15	Listar notificaciones	Alta	Medio	1	3	**3**
16	Gestionar valoraciones	Alta	Medio	1	4	**4**
17	Generar reporte de los proyectos evaluados	Alta	Alto	1	4	**4**
18	Generar reporte de las evaluaciones de un proyecto	Alta	Alto	1	4	**4**

19	Generar reporte de la evaluación de un proyecto	Alta	Alto	1	4	**4**
20	Generar reporte por Peso de Riesgo	Alta	Alto	1	5	**5**
21	Generar reporte por Factor de Riesgo	Alta	Alto	1	5	**5**
22	Generar reporte por Nivel de Atención al Riesgo	Alta	Alto	1	5	**5**
Totales				**20**	**5**	**5**

Tabla 4: Resumen de las Historias de Usuario. Fuente: Elaboración del autor.

A continuación, se presenta una de las Historias de Usuario de mayor importancia en el proyecto debido a la extensión de la cantidad de Historias de Usuario, el resto se podrán encontrar en la documentación del sistema.

Historia de Usuario	
Número: 6	**Usuario:** Gestor de Proyecto.
Nombre historia: Gestionar categorías de los riesgos.	
Prioridad en negocio: alta	**Riesgo en desarrollo:** alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Ediel Pérez Alvarez.	
Descripción: Se accederá mediante el menú a la funcionalidad Gestionar Riesgos y se podrá listar, insertar, editar y eliminar las categorías de los riesgos.	
Observaciones: Solo usuarios con privilegios correspondientes a esta funcionalidad la puede ejecutar y se debe verificar la integridad de los datos.	

Tabla 5: Gestionar categorías de los riesgos. Fuente: Elaboración del autor.

2.12- Resumen de tareas generadas por HU.

No	Nombre de la Historia de Usuario	Tareas Asignadas
1	Diseño y creación de la base de datos.	• Diseño de la base de datos. • Creación de la base de datos.
2	Autenticarse.	• Autenticarse.
3	Gestionar Usuarios	• Listar usuarios. • Insertar usuarios. • Editar usuarios. • Eliminar usuarios.
4	Trazas	• Listar traza. • Buscar traza.
5	Gestionar clasificación de los riesgos	• Listar clasificación de los riesgos. • Insertar clasificación de los riesgos. • Editar clasificación de los riesgos. • Eliminar clasificación de los riesgos.
6	Gestionar categorías de los riesgos	• Listar categorías de los riesgos. • Insertar categorías de los riesgos. • Editar categorías de los riesgos. • Eliminar categorías de los riesgos.
7	Gestionar evaluaciones.	• Listar evaluaciones. • Insertar evaluaciones.

8	Gestionar plan de mitigación.	• Insertar plan de mitigación. • Listar plan de mitigación.
9	Gestionar proyectos	• Listar proyectos. • Insertar proyectos. • Editar proyectos. • Eliminar proyectos. • Buscar proyectos.
10	Gestionar tipos de proyectos	• Listar tipos de proyectos. • Insertar tipos de proyectos. • Editar tipos de proyectos. • Eliminar tipos de proyectos. • Buscar tipos de proyectos.
11	Gestionar preguntas.	• Listar preguntas. • Insertar preguntas. • Editar preguntas. • Eliminar preguntas.
12	Gestionar rangos	• Listar rangos. • Insertar rangos. • Editar rangos. • Eliminar rangos.
13	Gestionar impactos	• Listar impactos. • Insertar impactos. • Editar impactos. • Eliminar impactos.
14	Gestionar probabilidades	• Listar probabilidades. • Insertar probabilidades. • Editar probabilidades. • Eliminar probabilidades.
15	Listar notificaciones.	• Listar notificaciones.

16	Gestionar valoraciones.	• Listar valoraciones. • Insertar valoraciones. • Editar valoraciones. • Eliminar valoraciones.
17	Generar reporte de los proyectos evaluados.	• Generar reporte de los proyectos evaluados.
18	Generar reporte de las evaluaciones de un proyecto.	• Generar reporte de las evaluaciones de un proyecto.
19	Generar reporte de la evaluación de un proyecto.	• Generar reporte de la evaluación de un proyecto.
20	Generar reporte por Peso de Riesgo.	• Generar reporte por Peso de Riesgo.
21	Generar reporte por Factor de Riesgo.	• Generar reporte por Factor de Riesgo.
22	Generar reporte por Nivel de Atención al Riesgo.	• Generar reporte por Nivel de Atención al Riesgo.

Tabla 6: Resumen de tareas generadas. Fuente: Elaboración del autor

2.12.1- Especificación de las tareas del sistema.

Debido a la extensión de la cantidad de tareas generadas en el sistema se mostrará las tares correspondiente a la Historia de Usuario utilizada, el resto se podrán encontrar en la documentación del sistema. Los puntos estimados estarán dados por días de trabajo.

Tarea	
Número de Tarea: 14	**Número de Historia:** 6
Nombre de la Tarea: Listar categorías de los riesgos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.25
Programador responsable: Ediel Pérez Alvarez.	
Descripción: Listar categorías de los riesgos será una opción del menú Gestionar Riesgos, esta deberá mostrar en una tabla todos los datos referentes en la base de datos.	

Tabla 7: Listar categorías de los riesgos. Fuente: Elaboración del autor.

Tarea	
Número de Tarea: 15	**Número de Historia:** 6
Nombre de la Tarea: Insertar categorías de los riesgos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.25
Programador responsable: Ediel Pérez Alvarez.	
Descripción: Se le permite al usuario insertar las categorías de los riesgos.	

Tabla 8: Insertar categorías de los riesgos. Fuente: Elaboración del autor.

Tarea	
Número de Tarea: 16	**Número de Historia:** 6
Nombre de la Tarea: Editar categorías de los riesgos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.25
Programador responsable: Ediel Pérez Alvarez.	
Descripción: Se le permite al usuario editar las categorías de los riesgos.	

Tabla 9: Editar categorías de los riesgos. Fuente: Elaboración del autor.

Tarea	
Número de Tarea: 17	**Número de Historia:** 6
Nombre de la Tarea: Eliminar categorías de los riesgos.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.25
Programador responsable: Ediel Pérez Alvarez.	
Descripción: Se le permite al usuario eliminar las categorías de los riesgos.	

Tabla 10: Eliminar categorías de los riesgos. Fuente: Elaboración del autor.

2.13- Diseño de tarjetas de Clase, Responsabilidad y Colaboración (CRC)

Una de las principales funcionalidades que tiene las tarjetas CRC es mostrar las colaboraciones que se establecen entre las clases que componen el sistema,

donde cada tarjeta representa una clase con su nombre en la parte superior. A continuación, se muestran algunas de las tarjetas que se crearon las otras se encuentran en el la Documentación del Sistema.

Tarjeta CRC	
Clase: usuario	
Superclase :	
Subclase:	
Descripción: En esta clase se guardan los datos relacionados con los usuarios del sistema	
Atributos	
Id	Int(11)
Rol	Int(11)
username	Varchar(20)
Foto	Varchar(255)
nombre	Varchar(20)
Ci	Varchar(11)
passsword	Varchar(255)
Salt	Varchar(255)
apellido1	Varchar(20)
Apellido2	Varchar(20)

Tabla 11: Tarjeta CRC: usuario *Fuente: Elaboración del autor.*

Tarjeta CRC	
Clase: riesgo	
Superclase :	
Subclase:	
Descripción: En esta clase se guardan los datos relacionados con los riesgos.	
Atributos	
Id	Int(11)
nombre	Varchar(30)
descripcion	longtext

Tabla 12: Tarjeta CRC: riesgo *Fuente: Elaboración del autor.*

2.14- Análisis de los Costos (Modelo matemático COCOMO II)

El modelo de estimación de software COCOMO II fue propuesto y elaborado por Barry Boehm, este modelo es uno de los modelos de estimación más utilizados y con una gran cantidad de documentación. El modelo permite obtener el esfuerzo y el tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto.

Este modelo consta de tres etapas que se presentan a continuación: (Pressman 2010)

- **Modelo de composición de aplicación**: se usa en las primeras etapas de proyecto, cuando son primordiales la elaboración de prototipos de las interfaces de usuario, la consideración de la interacción del software y el sistema, la valoración del rendimiento y la evaluación de la madurez de la tecnología.
- **Modelo de etapa temprana de diseño**: Se usa una vez estabilizados los requisitos y establecida la arquitectura básica del software.
- **Modelo de etapa postarquitectónica**: Se usa durante la construcción del software.

En este proyecto se realizó mediante el **modelo de etapa temprana de diseño**. En esta etapa se tiene poca información, se conoce muy poco del tamaño del producto a ser desarrollado, de la naturaleza de la plataforma, del personal a ser incorporado al proyecto o aspectos específicos del proceso a utilizar. Ese nivel de detalle en este modelo es consistente con el nivel general de información disponible y con nivel general de estimación detallada que es necesaria en estas etapas, lo que concuerda con el uso de Puntos de Función. Para estimar tamaño usa Puntos de Función No Ajustados como métrica de medida y un número reducido de factores de costo.

<u>Etapa 1</u>

Estimación de esfuerzo

E = NOP/PROD

E: Esfuerzo personas-mes

NOP: New Object Point NOP

PROD: razón de productividad

- Paso 1: Identificación de los tipos de objetos, las ventanas, los informes, los componentes de tercera generación, en la siguiente tabla se muestra los valores de complejidad según el tipo de objeto.

Tipo de objeto	Peso de complejidad		
	Simple	Medio	Difícil
Pantalla	1	2	3
Reporte	2	5	8
Componente 3GL			10

Figura 3: Complejidad de los tipos de objetos.

- Paso 3: Cálculo de los puntos de objetos
  OP = 8*3+10 = 34
- Paso 4: Cálculo de los nuevos puntos de objetos
  NOP = OP*(100 - %reuso) /100 = 34*(100-10) /100 = 30.6
- Paso 5 = Determinar la razón de productividad PROD
  PROD = NOP/persona-mes
  PROD = 30.6/3
  PROD = 5.1
- Paso 6: Cálculo del esfuerzo
  Esfuerzo estimado = NOP/PROD
  =30.6/5.1
  = 6 aproximadamente, más un mes de prueba por lo que serían 7 meses.

2.15- Beneficios tangibles e intangibles

Situar en funcionamiento este sistema web favorece en gran medida a los centros educativos cubanos que su objetivo sea desarrollar estos softwares educativos, porque informatiza todo el procedimiento de evaluación y control de los riesgos durante el desarrollo de esos proyectos educativos, brindándole al personal encargado diversos beneficios como tener un control de gran parte del proceso directivo, organización y conocer de posibles problemas que puede ir apareciendo durante el desarrollo de estos softwares y permite realizar un

diagnóstico del mismo y evitar que puedan ocurrir estos problemas, además , permite un trabajo más rápido y con una mayor calidad.

2.16- Diseño de la base de datos

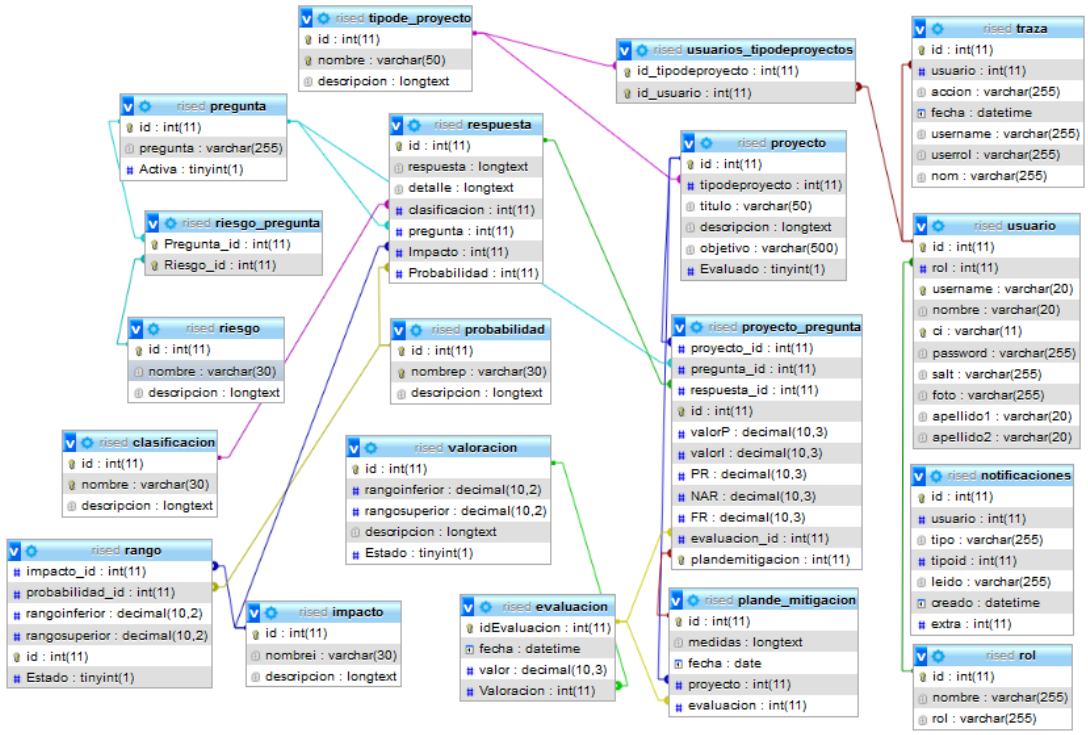


Figura 4: Diseño de la Base de Datos. *Fuente: Elaboración del autor.*

2.17- Construcción de la propuesta.

La construcción de la solución propuesta se llevó a cabo de forma iterativa e incremental. Luego de terminar las historias de usuario, se obtuvo una visión clara de lo que implicaría el diseño final e implementación de la solución. Se utilizaron las prioridades dadas en la planificación a cada caso de uso y sus riesgos para componer el contenido de cada incremento.

2.18- Despliegue.

El sistema informático es un sistema web para la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los softwares educativos.

Para la puesta en marcha de este sistema es necesario seguir y no violar los pasos descritos en el Manual de Instalación del sistema. Siguiendo esto con detalles, la instalación no resultará un proceso complejo.

Conclusiones del Capítulo

En el presente capítulo se explicó como este sistema web permite la gestión de la información asociada a los riesgos durante el proceso de desarrollo de los

softwares educativos, para que de esta forma el personal de los centros de tecnologías educativas tenga la posibilidad de conocer los problemas antes de que se les presenten y tener en caso de que ocurra una solución para ello. Para una adecuada planificación se realizó un levantamiento de requisitos funcionales y no funcionales, permitiendo realizar las historias de usuario del proyecto según la metodología XP, junto al plan de iteraciones, es decir, empleando los artefactos. El análisis del costo fue calculado por etapa, una primera al comenzar el proyecto que fue por Puntos de Función y una segunda parte en el intermedio del proyecto con COCOMO II arrojando el costo del proyecto como el tiempo que tardaría.

Capítulo 3: Validación de la solución propuesta.

Introducción

En este capítulo se realizan las pruebas al software permitiendo comprobar la calidad de este producto, lo que constituye uno de los pasos más importantes en el diseño e implementación de un sistema. No debe existir ninguna característica en el programa que no haya sido probada con la intención de mostrar un error no descubierto hasta entonces y con el fin de verificar la fiabilidad y calidad de la aplicación como un todo.

3.1 Pruebas al software

El proceso de pruebas es el instrumento más adecuado para determinar el status de la calidad de un producto. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos o si es el software que se quería desarrollar. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de Prueba.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. Los niveles de trabajo en los cuales se pueden realizar las pruebas son:

- Prueba de Unidad.
- Prueba de Integración.
- Prueba de Sistema.
- Prueba de Aceptación.
- Prueba de Seguridad.

3.1.1 Plan de pruebas

Al desarrollar el plan de pruebas, se puede obtener información sobre los errores, defectos o fallas que tiene el prototipo, así se realizan las correcciones pertinentes, según el caso y se asegura la calidad del producto que se está entregando al cliente.

A continuación, se muestra en la **Tabla 13** el plan de pruebas.

No.	Historia de Usuario	Pruebas a realizar
1	Diseño y Creación de la base de datos.	• Test base de datos.
2	Autenticarse.	• Test de autenticarse.

3	Gestionar usuarios.	• Test de listar usuario. • Test de insertar usuario. • Test de editar usuario. • Test de eliminar usuario.
4	Trazas.	• Test de listar traza. • Test de buscar traza.
5	Gestionar clasificación de los riesgos.	• Test de listar clasificación de los riesgos. • Test de insertar clasificación de los riesgos. • Test de editar clasificación de los riesgos. • Test de eliminar clasificación de los riesgos.
6	Gestionar categorías de los riesgos.	• Test de listar categorías de los riesgos. • Test de insertar categorías de los riesgos. • Test de editar categorías de los riesgos. • Test de eliminar categorías de los riesgos.
7	Gestionar evaluaciones.	• Test de listar evaluaciones. • Test de insertar evaluaciones.
8	Gestionar plan de mitigación.	• Test de insertar plan de mitigación. • Test de listar plan de mitigación.
9	Gestionar proyectos.	• Test de listar proyectos. • Test de insertar proyectos. • Test de editar proyectos. • Test de eliminar proyectos.

		• Test de buscar proyectos.
10	Gestionar tipos de proyectos.	• Test de listar tipos de proyectos. • Test de insertar tipos de proyectos. • Test de editar tipos de proyectos. • Test de eliminar tipos de proyectos. • Test de buscar tipos de proyectos.
11	Gestionar preguntas.	• Test de listar preguntas. • Test de insertar preguntas. • Test de editar preguntas. • Test de eliminar preguntas.
12	Gestionar rangos.	• Test de listar rangos. • Test de insertar rangos. • Test de editar rangos. • Test de eliminar rangos.
13	Gestionar impactos.	• Test de listar impactos. • Test de insertar impactos. • Test de editar impactos. • Test de eliminar impactos.
14	Gestionar probabilidades.	• Test de listar probabilidades. • Test de insertar probabilidades. • Test de editar probabilidades. • Test de eliminar probabilidades.
15	Listar notificaciones.	• Test de listar notificaciones.
16	Gestionar valoraciones.	• Test de listar valoraciones. • Test de insertar valoraciones. • Test de editar valoraciones. • Test de eliminar valoraciones.

17	Generar reporte de los proyectos evaluados.	• Test de generar reporte de los proyectos evaluados.
18	Generar reporte de las evaluaciones de un proyecto.	• Test de generar reporte de las evaluaciones de un proyecto.
19	Generar reporte de la evaluación de un proyecto.	• Test de generar reporte de la evaluación de un proyecto.
20	Generar reporte por Peso de Riesgo.	• Test de generar reporte por Peso de Riesgo.
21	Generar reporte por Factor de Riesgo.	• Test de generar reporte por Factor de Riesgo.
22	Generar reporte por Nivel de Atención al Riesgo.	• Test de generar reporte por Nivel de Atención al Riesgo.

Tabla 13: Plan de Pruebas. *Fuente: Elaboración del autor.*

3.2.2 Pruebas de aceptación

Las Pruebas de Aceptación (PA) son las realizadas por el cliente y usuarios finales de la aplicación. En estas serán probadas las funcionalidades definidas por el cliente y descritas en las historias de usuario, además de los aspectos de seguridad requeridos. Luego de haber superado las pruebas de aceptación podrá considerarse que la aplicación es apta para el uso.

A continuación, se muestran algunas de estas pruebas:

Prueba de Aceptación	
Número de Caso de Prueba:13	**Número de Historia de Usuario: 6**
Nombre de Caso de Prueba: Test de listar categorías de los riesgos.	
Responsable: Ediel Pérez Alvarez.	
Descripción: Verificar que se permita listar todas las categorías de los riesgos.	
Condiciones de ejecución: Estar conectado a la Base de Datos.	
Entrada: Acceder a la vista donde se muestra el listado de todas las categorías de los riesgos.	
Resultado esperado: El sistema debe mostrar en una tabla todas las categorías de los riesgos en la base de datos.	

Evaluación: Satisfactoria.

Tabla 14: Test de listar categorías de los riesgos. Fuente: Elaboración del autor.

Prueba de Aceptación	
Número de Caso de Prueba:14	**Número de Historia de Usuario: 6**
Nombre de Caso de Prueba: Test de insertar categorías de los riesgos.	
Responsable: Ediel Pérez Alvarez.	
Descripción: Verificar que se permita insertar todos los datos de las categorías de los riesgos. Se insertarán correctamente los datos para comprobar que los datos son almacenados en la base de datos. También se insertarán con datos incorrectos y campos en blancos teniendo en cuenta que no se almacenarán en la base de datos.	
Condiciones de ejecución: Estar conectado a la Base de Datos.	
Entrada: -Insertar los datos correctamente. -Insertar los datos incorrectamente. -Dejar campos en blanco.	
Resultado esperado: El sistema cuando se inserten los datos correctamente debe almacenarlos en la base de datos y mostrarlos. Cuando se inserten datos incorrectos o campos en blanco mostrar una alerta.	
Evaluación: Satisfactoria.	

Tabla 15: Test de insertar categorías de los riesgos. Fuente: Elaboración del autor.

Prueba de Aceptación	
Número de Caso de Prueba:15	**Número de Historia de Usuario: 6**
Nombre de Caso de Prueba: Test de editar categorías de los riesgos.	
Responsable: Ediel Pérez Alvarez.	

Descripción: Verificar que se permita editar todos los datos de las categorías de los riesgos. Se modifican correctamente los datos para comprobar que los datos son almacenados en la base de datos. También se modificarán con datos incorrectos y campos en blancos teniendo en cuenta que no se almacenarán en la base de datos.
Condiciones de ejecución: Estar conectado a la Base de Datos.
Entrada: -Editar los datos correctamente. -Editar los datos incorrectamente. -Dejar campos en blanco.
Resultado esperado: El sistema cuando se editen los datos correctamente debe almacenarlos en la base de datos y mostrarlos. Cuando se editen los datos incorrectos o campos en blanco mostrar una alerta.
Evaluación: Satisfactoria.

Tabla 16: Test de editar categorías de los riesgos. Fuente: Elaboración del autor.

Prueba de Aceptación	
Número de Caso de Prueba:16	**Número de Historia de Usuario: 6**
Nombre de Caso de Prueba: Test de eliminar categorías de los riesgos.	
Responsable: Ediel Pérez Alvarez.	
Descripción: Verificar que se permita eliminar de la tabla el registro deseado.	
Condiciones de ejecución: Estar conectado a la Base de Datos.	
Entrada: -Se eliminará la categoría dando aceptar en el mensaje que se muestra.	
Resultado esperado: Se elimina la categoría del riesgo del sistema.	
Evaluación: Satisfactoria.	

Tabla 17: Test de eliminar categorías de los riesgos. Fuente: Elaboración del autor.

3.2.3 Pruebas Funcionales:

Insertar categorías de los riesgos

Atributos:

- nombre
- descripcion

1-Clases de equivalencia por cada atributo:

Atributo	Válida	Representante	Inválida	Representante
nombre	1-Cualquier combinación de letras que comience con Mayúscula y tenga al menos 3 caracteres.	Pedagógico	2-Que este vacía	NULL
			3- Cadena que contenga menos de 3 caracteres	as
			4-Cadena que no comience con mayúscula.	software
			5-Cadena que contenga números o caracteres extraños.	12123{¨¨·
descripcion	6- Cualquier combinación de letras	El grado de incertidumbre	7-Que este vacía	NULL

	y números al menos 40 caracteres.	de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean	8-Cadena que contenga menos de 40 caracteres	El grado de incertidumbre.

Tabla 18: Clases de equivalencia. *Fuente: Elaboración del autor.*

2-Definir casos de Pruebas:

No.	Clase Equiv.	nombre	descripción	Result. Esperado
1	1,6	Pedagógico	El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de	Resultado esperado

			desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean	
2	1,7	Pedagógico	NULL	Descripción no puede estar vacía.
3	1,8	Pedagógico	El grado de incertidumbre.	Descripción es muy breve, tiene que contener al menos 40 caracteres.
4	2,6	NULL	El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo	El nombre no puede estar vacío.

			de la tecnología que allí posean	
5	3,6	as	El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean	Nombre está muy corto, debe tener al menos 3 caracteres.
6	4,6	software	El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean	El nombre debe comenzar con mayúscula.

7	5,6	12123{¨¨	El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean	El nombre no puede contener caracteres extraños ni números.

Tabla 19: Casos de Prueba *Fuente: Elaboración del autor.*

3-Tabla de Prueba por Caso de Prueba:

No.	2
Requerimiento	Insertar riesgo
Objetivo	Probar la acción de insertar riesgo (clases equivalentes 1,7)
Tipo de Prueba	Funcional
Hardware	Un procesador Core i5 , disco duro de 750GB, memoria RAM de 6GB
Software	Sistema Operativo Windows 7 o superior, Base de Datos Xampp, Navegador Mozilla Firefox 3.5 o superior
Personal	Ingeniero de Pruebas
Casos de Prueba	
Datos de Entrada	Nombre: Pedagógico Descripción: NULL

Resultados Esperados	Mensaje: ¨La descripción no puede estar vacía ¨		
Resultados Obtenidos	Si(x) No()		
Casos de Excepción:		**Comentarios:**	
Aprobado por: Ediel Pérez Alvarez	**Cargo:** Líder de pruebas	**Fecha:**	

Tabla 20: Caso de Prueba #2 *Fuente: Elaboración del autor.*

No.	1	
Requerimiento	Insertar riesgo	
Objetivo	Probar la acción de insertar riesgo (clases equivalentes 1,6)	
Tipo de Prueba	Funcional	
Hardware	Un procesador Core i5 , disco duro de 750GB, memoria RAM de 6GB	
Software	Sistema Operativo Windows 7 o superior, Base de Datos Xampp, Navegador Mozilla Firefox 3.5 o superior	
Personal	Ingeniero de Pruebas	
Casos de Prueba		
Datos de Entrada	Nombre: Pedagógico Descripción: El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean.	
Resultados Esperados	Mensaje: ¨El riesgo se insertó correctamente. ¨	
Resultados Obtenidos	Si(X) No()	
Casos de Excepción:		**Comentarios:**

Aprobado por: Ediel Pérez Alvarez	**Cargo:** Líder de pruebas	**Fecha:**

Tabla 21: Caso de Prueba #1 Fuente: Elaboración del autor.

No.	4	
Requerimiento	Insertar riesgo	
Objetivo	Probar la acción de insertar riesgo (clases equivalentes 2,6)	
Tipo de Prueba	Funcional	
Hardware	Un procesador Core i5 , disco duro de 750GB, memoria RAM de 6GB	
Software	Sistema Operativo Windows 7 o superior, Base de Datos Xampp, Navegador Mozilla Firefox 3.5 o superior	
Personal	Ingeniero de Pruebas	
Casos de Prueba		
Datos de Entrada	Nombre: NULL Descripción: El grado de incertidumbre de la preparación de los profesores para trabajar con el software, el nivel de desarrollo de los estudiantes, el nivel educacional y la institución donde se trabajará con el software ya terminado y el nivel de desarrollo de la tecnología que allí posean.	
Resultados Esperados	Mensaje: ¨El nombre no puede estar vacío. ¨	
Resultados Obtenidos	Si(X) No()	
Casos de Excepción:		**Comentarios:**
Aprobado por: Ediel Pérez Alvarez	**Cargo:** Líder de pruebas	**Fecha:**

Tabla 22: Caso de Prueba #4 Fuente: Elaboración del autor.

En el **anexo VI** se muestran varias figuras de cómo se muestran los errores del insertar, editar y eliminar de los riesgos en las pruebas realizadas.

3.3 Herramientas de Pruebas.

También se emplearon herramientas para hacer pruebas tal es el caso del software Subgraph Vega y Webstress obteniendo los siguientes resultados:

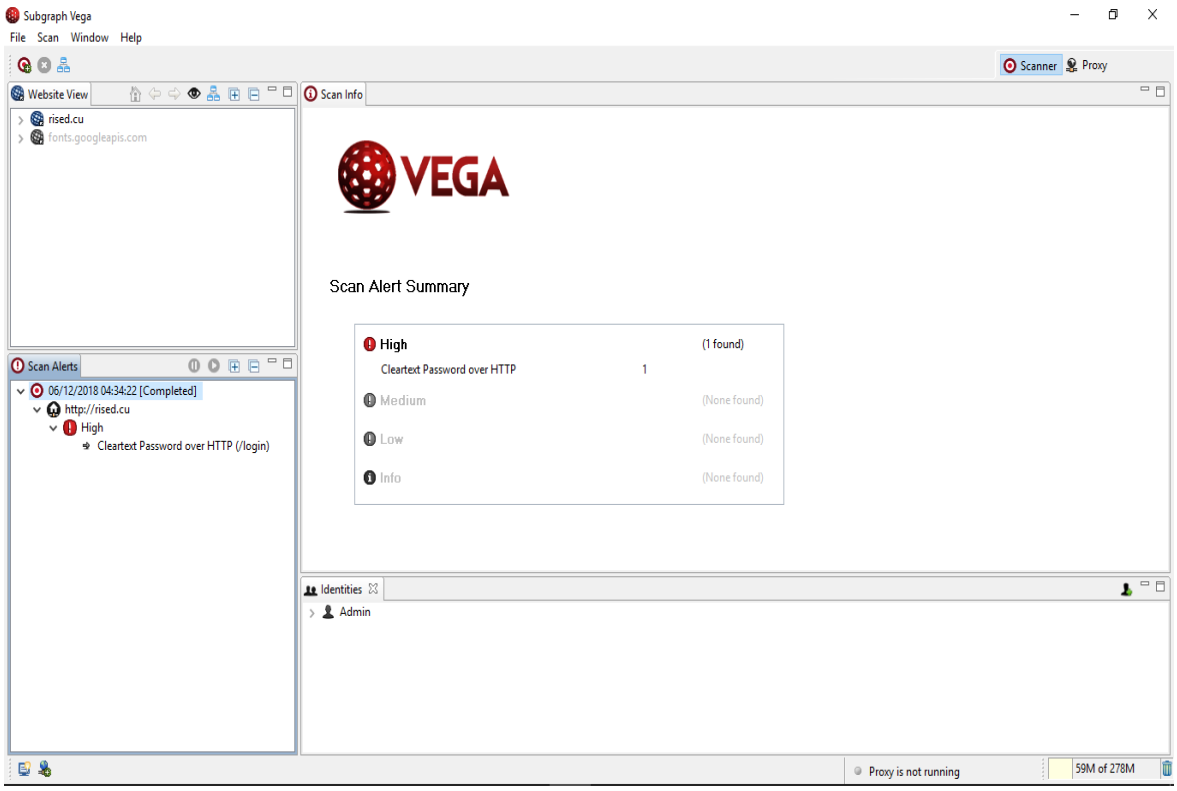


Figura 5: Resultado de una prueba realizada con Subgraph Vega.

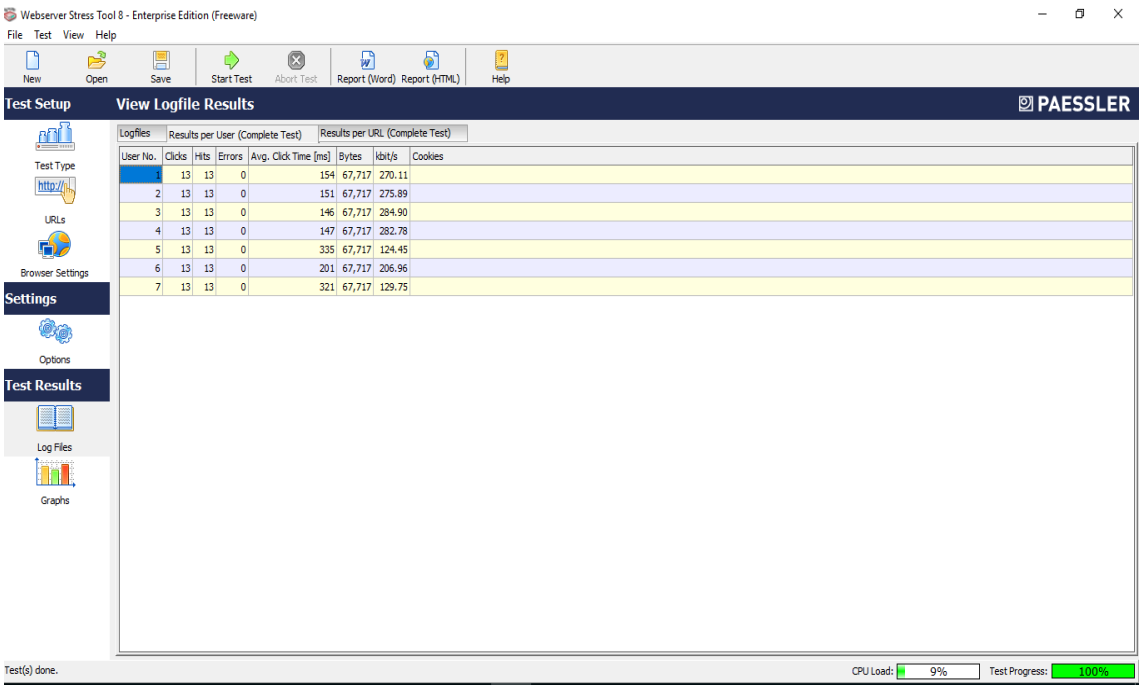


Figura 6: Resultado de una prueba realizada con Webstress.

3.4 Análisis de los resultados obtenidos.

Al concluir el desarrollo del proceso de pruebas se lograron resultados satisfactorios en el progreso del sistema con un nivel medio de sencillez. Las pruebas fueron desplegadas por cada una de las historias de usuario. Los elementos de pruebas abordados, permitieron validar el funcionamiento de la aplicación y los resultados satisfactorios de dichas pruebas.

Conclusiones del Capítulo.

Las pruebas realizadas utilizando las técnicas anteriormente planteadas fueron de gran importancia para demostrar el buen funcionamiento del software y el cumplimiento de los requisitos planteados por el cliente. Una vez realizadas las pruebas se logró brindarle al cliente una versión del software completamente funcional que facilitara la informatización de la gestión de los riesgos en los softwares educativos.

Conclusiones Generales

En esta investigación quedaron satisfechos los objetivos planteados y de esta forma se determinaron las siguientes conclusiones:

- Se determinaron el marco teórico referencial sobre la gestión de la información de los riesgos durante el proceso de desarrollo de los softwares educativos que se exponen a continuación:
  - Se normalizaron los fundamentos teóricos que permitan la creación de un sistema web que gestione la información asociada a los riesgos durante el desarrollo de los softwares educativos.
  - Se determinó la metodología eXtreme Programming (XP) para el desarrollo del sistema web debido a sus características.
  - Se determinaron como herramientas más viables para la creación del proyecto el gestor de base de datos MySQL 4.5.1, lenguaje PHP 5.6.21, como servidor web Apache 2.4.17, *framework* Symfony 2.7.3, el Entorno de Desarrollo Integrado (IDE) utilizado fue JetBrains PhpStorm 2016.1
- Se implementó una aplicación web que facilite la gestión de los riesgos en el desarrollo de los softwares educativos.
- Se aplicó un sistema de pruebas que permitió detectar errores cometidos durante los procesos anteriores del desarrollo del sistema web para la gestión de los riesgos en el desarrollo de los softwares educativos y corregirlos.

Recomendaciones

Desde el punto de vista del alcance del presente trabajo y teniendo en cuenta el momento de desarrollo del mismo, se proponen las siguientes recomendaciones:

1. Aprovechar las posibilidades de información almacenadas en la aplicación e implementar funcionalidades que respondan a nuevos intereses.
2. Agregarle nuevos reportes que sean de interés para los usuarios que interactúen con la aplicación.
3. También se recomienda poner en funcionamiento el sistema web para automatizar la gestión de la información de los riesgos durante el proceso de desarrollo de los softwares educativos.
4. Implantar el sistema en el repositorio nacional para generalizar su introducción en todos los CES del país.

Bibliografía

Achour, M. (2007). Manual de PHP.

AFC, U. S. A. F. (1988). "Software Risk Abatement." Pamphlet.

Armendáriz Barreno, G. A. and M. G. S. Guaraca (2013). Adaptación de las metodologias ágiles scrum y extreme game development en una metodologia para desarrollo de videojuegos en android. Caso práctico: Desarrollo de un videojuego. Escuela de Ingeniería en Sistemas, Facultad de Informatica y Electrónica. Riobamba – Ecuador, Escuela Superior Politécnica Del Chimborazo. **Tesis de Grado previa a la Obtencion del Título de: Ingenieros en Sistemas Informáticos:** 332.

Aslan, S. (2016). Digital Educational Games: Methodologies for Development and Software Quality. Blacksburg, Virginia, USA, Virginia Polytechnic Institute and State University. **Doctor of Philosophy in Computer Science and Applications:** 152.

Azamar, S. G. P. C. B. L. (2006). "Metodología para el Desarrollo de Software Educativo (DESED)."

Beck, K. (2004). Extreme Programming Explained: Embrace Change.

Brioli, C., et al. (2011). "Referente Teórico y Metodológico para el Diseño Instruccional de Entornos Virtuales de Enseñanza y Aprendizaje (EVEA)." Docencia Universitaria **XII**(2).

Cataldi, Z. (2013). Una metodología para el diseño, desarrollo y evaluación de software educativo. Tesis para el Magister de Automatización de Oficinas.

Cataldi, Z., et al. (2006). "Metodología extendida para la creación de software educativo desde una visión integradora." Revista Latinoamericana de Tecnología Educativa **2**(1).

Charette, R. N. (1989). "Software Engineering Risk Analysis and Management."

Chittaro, L. and F. Buttussi (2015). "Assessing Knowledge Retention of an Immersive Serious Game vs. a Traditional Education Method in Aviation Safety." IEEE Transactions on Visualization and Computer Graphics **21**(4): 529-538.

Dey, P. P., et al. (2016). "Managing interacting software project risks." Pressacademia **2**(1): 427-427.

Domingues, M. S. Q., et al. (2017). "Engineering complex systems applied to risk management in the mining industry." International Journal of Mining Science and Technology **27**(4): 611-616.

Drucker, P. (1975). "Management."

Fernández López, J. A., et al. (2016). "Desarrollo de las Políticas de Uso de un Repositorio Digital para una Institución de Nivel Superior en Situación de Desventaja Tecnológica." Archivos analíticos de políticas educativas **24**(10).

Figueroa, M. C. M. A. A. (2009). "MeISE: Metodología de Ingeniería de Software Educativo " **2**.

Gauchat, J. D. (2012). El Gran libro de HTML5,CSS3 y Javascript.

Giacomone, M. B. and S. B. González (2012). El problema de apolonio provoca incertidumbre y genera nuevos conocimientos. Actas III Jornadas de Enseñanza e Investigación Educativa en el campo de las Ciencias Exactas y Naturales. U. N. d. L. Plata. La Plata, Argentina, Facultad de Humanidades y Ciencias de la Educación. .

González, Y. R. (2017). Aplicación web para la gestión de los equipos de la reserva estatal en la División de la Empresa de Atención a Equipos Matanzas. Informática, Matanzas.

Guerra, A. A. (2016). "Software educativo para el trabajo con matrices."

Gush, D. P. (1994). "A Construct for Describing Software Development Risk . Software Engineering."

Hains-Wesson, R. (2014). "Towards a self-report methodology for STEM research when developing and implementing a blended learning faculty-wide curriculum design framework." Contemporary Approaches to Research in Mathematics, Science, Health and Environmental Education **1**.

Hall, E. M. (1998). "Managing Risk: Methods for Software Systems Development." Addison Wesley.

Higuera, R. P. (1995). "Team Risk Management." pag 2-4.

Huang, Q. (2015). Research on Risk Analysis and Management in the Software Development Process. 5th International Conference on Education, Management, Information and Medicine (EMIM 2015). New York, USA, Atlantis Press**:** 1294-1298.

IZQUIERDO, L. R. (2014). Aplicacion Web para la evaluacion y control de la gestion tecnologica e innovadora en las empresas cubanas., Universidad de Matanzas.

Joskowicz, J. (2008). "Reglas y Prácticas en Extreme Programming."

Keil, M. (1998). "A Framework for Identifying Software Project Risks."

Marques Graell, P. (2007). "El software educativo." from http://tecno-educativa.blogspot.com/2007/03/software-definicin-y-caractersticas.html.

Marqués, P. (2010). Metodología para la elaboración de software educativo. Barcelona, Editorial Estel.

Marqués, P. (2014). "El software educativo. ." Universidad Autónoma de Barcelona.

Mauricio Castro, G. (2013). "El nuevo estandar ISO para la Gestión del riesgo."

Miranda Juana, M. (2009). Iniciativas de acceso abierto para la conformación de repositorios institucionales. 2a. etapa : Propuesta de implementación de un espacio de la UNaM en la web". Misiones, Argentina, Universidad Nacional de Misiones.

Morales, L. (2011). METODOLOGÍA PARA LA GESTIÓN DE RIESGOS EN EL PROCESO DE PLANIFICACIÓN DEL SOFTWARE EDUCATIVO. Facultad de Ciencias de la Economía y la Informática, Universidad de Matanzas.

MORINE, R. J. I. (2013). "Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android."

Peña, M. Á. (2007). "Arquitectura cliente-servidor.".

Perera, R. B. (2016). Sistema web para la creación de mapas conceptuales que contribuya al autoaprendizaje de los estudiantes. Informática, Matanzas.

Piedra, M. V. L. F. G. M. A. M. R. (2010). "Softwares educativos."

Potencier, F., & Zaninotto,F. (2008). Symfony la guía definitiva.

Pressman, R. (2011). Ingeniería del software: Un Enfoque Práctico. New York, USA, McGRAW-HILL Higher Education.

Pressman, R. S. (2010). Ingeniería de Software.Enfoque Práctico.

Raible, M. (2013). "Refreshing Your UI with HTML5,Bootstrap and CSS3."

RUMANCELA, H. D. G. (2017). INFLUENCIA DEL SOFTWARE "MICROSOFT MATHEMATIC" EN EL PROCESO Y DESARROLLO DE APRENDIZAJE DE LA MATEMÁTICA EN LOS ESTUDIANTES DEL NOVENO AÑO DE EDUCACIÓN BÁSICA, DE LA UNIDAD EDUCATIVA "11 DE NOVIEMBRE" CANTÓN GUANO, PROVINCIA DE CHIMBORAZO. FACULTAD DE CIENCIAS DE LA EDUCACIÓN, HUMANAS Y TECNOLOGÍAS, UNIVERSIDAD NACIONAL DE CHIMBORAZO.

Schwartz, B. (2012). "High Performance MySQL. ."

Tedeschi, N. (2013). "¿Qué es un Patrón de Diseño? ."

Tho, I. (2005). "Managing the Risks of IT(Ian Tho)." USA.

Vázquez, A. (2009). Material utilizado en la maestría de Ciencias Pedagógicas. Tipologías de software educativo. Cuba.

Williams, R. C. W., J.A ; Dorofee, A.J (1997). "Putting Risk Management into Practice. IEEE Software."

Anexos

Anexo **I:** Diagramas del proceso de Gestión de Riesgo de la Norma ISO 31000

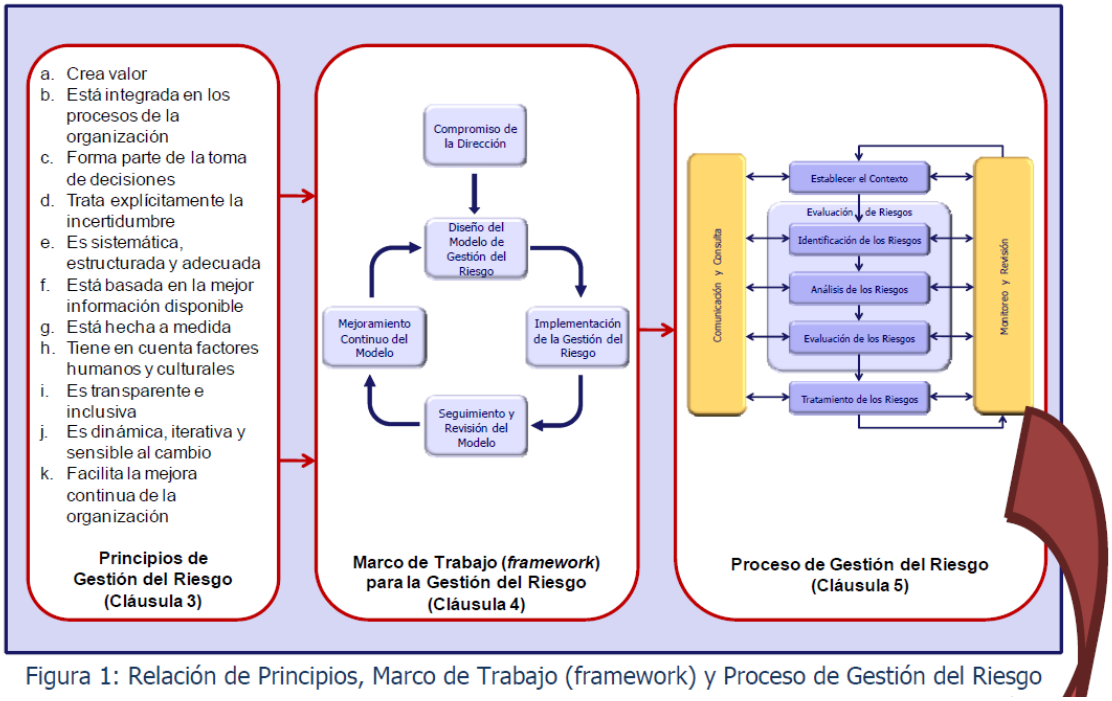


Figura 1: Relación de Principios, Marco de Trabajo (framework) y Proceso de Gestión del Riesgo

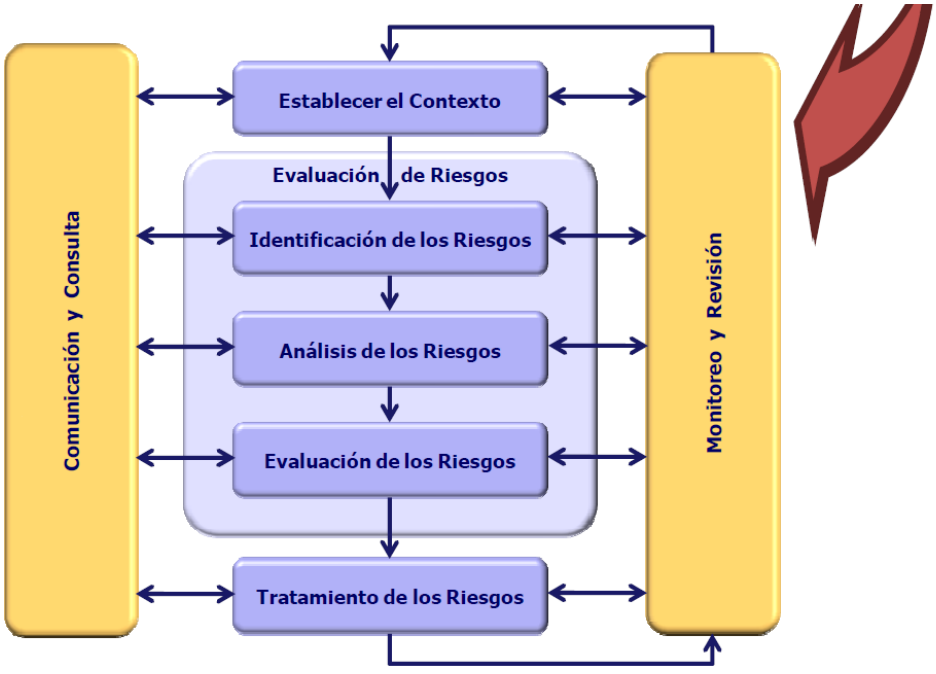


Figura 2: Proceso de Gestión del Riesgo

Anexo II: Las preguntas que se establecieron para el proceso de gestión son:

1. ¿Tiene el cliente una idea formal de lo que se requiere?
2. ¿El cliente está dispuesto a participar en la etapa de desarrollo del proyecto?
3. ¿El cliente especificó la tecnología que se requiere para la elaboración?

4. ¿Se acordaron los términos del proyecto, es decir, el costo, tiempo, entre otros?
5. ¿Está de acuerdo el equipo de desarrollo en los términos acordados?
6. ¿Tiene el equipo de ingenieros de software el conjunto adecuado de habilidades y la experiencia necesaria?
7. ¿Está definido el equipo de desarrollo en especialidad?
8. ¿Es adecuado el número de personas del equipo del proyecto para la realización del trabajo?
9. ¿Son estables los requerimientos del proyecto?
10. ¿Mientras se desarrolla el producto hay requerimientos cambiantes?
11. ¿Cuál es el efecto que provoca en el sistema los requerimientos cambiantes?
12. ¿Hay requerimientos faltantes o están incompletas las especificaciones?
13. ¿Tiene gran impacto esos requerimientos en el sistema?
14. ¿Existen requerimientos no claros o que necesitan interpretación?
15. ¿Los requerimientos conducirán al producto que fue planteado?
16. ¿Los requerimientos no son factibles desde un punto de vista analítico?
17. ¿Los requerimientos especifican un producto más grande o más complejo o requiere una organización mayor de lo que está acostumbrado la entidad?
18. ¿Aplicó algún método de estimación?
19. ¿Tipo de Base de Datos?
20. ¿Se han comprendido cuáles son los objetivos del software a construir y para quién va dirigido?
21. ¿Se adecua el software con el sistema de asignaturas del nivel educacional a quien va dirigido?
22. ¿El nivel de desarrollo de los estudiantes es suficiente para trabajar con el software?
23. ¿Los profesores están preparados totalmente para trabajar correctamente con el software y darle el uso adecuado?
24. ¿Todos los estudiantes tienen acceso a las computadoras para trabajar de manera adecuada con el software?
25. ¿Existe un especialista del tema a disposición del equipo de desarrollo para que le explique en caso de dudas?

26. ¿Es difícil de implementar el diseño?
27. ¿Las interfaces externas están bien diseñadas y controladas?
28. ¿Hay requerimientos rigurosos de tiempo de respuesta?
29. ¿El hardware limita la habilidad de alcanzar algún requerimiento?
30. ¿Está siendo reutilizado algún software no desarrollado por el equipo?
31. ¿Tienen conocimiento de la metodología que van a emplear?
32. ¿Se dispone del tiempo suficiente para la realización del software?
33. ¿Ha sido exitosa la comunicación entre los integrantes del equipo de desarrollo?
34. ¿Se realizaron las estimaciones por cada iteración correspondiente?
35. ¿Los niveles y tiempos asignados a las pruebas de unidad son adecuados?
36. ¿El lenguaje de programación es apropiado para producir el software?
37. ¿Se realizó algún tipo de prueba?
38. ¿Descoordinada integración del sistema, pobre definición de interfaces o instalaciones inadecuadas?
39. ¿La implementación será difícil de entender o mantener?
40. ¿El personal de mantenimiento estuvo involucrado en el diseño?
41. ¿Conocen la metodología a emplear para realizar el mantenimiento?
42. ¿Se está usando más de un modelo de desarrollo?
43. ¿Las relaciones entre los roles están claras?

Anexo III: Tablas de ponderaciones para EI, EQ, EO, ILF, EIF

CLASIFICACION DE ENTRADAS Y CONSULTAS	1-4 Atributos	5-15 Atributos	Más de 15 Atributos
0 o 1 ficheros accedidos	BAJA 3	BAJA 3	MEDIA 4
2 ficheros accedidos	BAJA 3	MEDIA 4	ALTA 6
Más de 2 ficheros accedidos	MEDIA 4	ALTA 6	ALTA 6

CLASIFICACION DE SALIDAS	1-5 Atributos	6-19 Atributos	Más de 19 Atributos
0 o 1 ficheros accedidos	BAJA 4	BAJA 4	MEDIA 5

2 o 3 ficheros accedidos	BAJA 4	MEDIA 5	ALTA 7
Más de 3 ficheros accedidos	MEDIA 5	ALTA 7	ALTA 7

FICHEROS LÓGICOS INTERNOS	1-19 Atributos	20-50 Atributos	Más de 50 Atributos
1 Entidad o registro lógico	BAJA 7	BAJA 7	MEDIA 10
2 - 5 Entidades o registros lógicos	BAJA 7	MEDIA 10	ALTA 15
Más de 5 Entidades o registros lógicos	MEDIA 10	ALTA 15	ALTA 15

FICHEROS LÓGICOS EXTERNOS	1-19 Atributos	20-50 Atributos	Más de 50 Atributos
1 Entidad o registro lógico	BAJA 5	BAJA 5	MEDIA 7
2 - 5 Entidades o registros lógicos	BAJA 5	MEDIA 7	ALTA 10
Más de 5 Entidades o registros lógicos	MEDIA 7	ALTA 10	ALTA 10

Anexo IV: Tabla para obtener el ACT

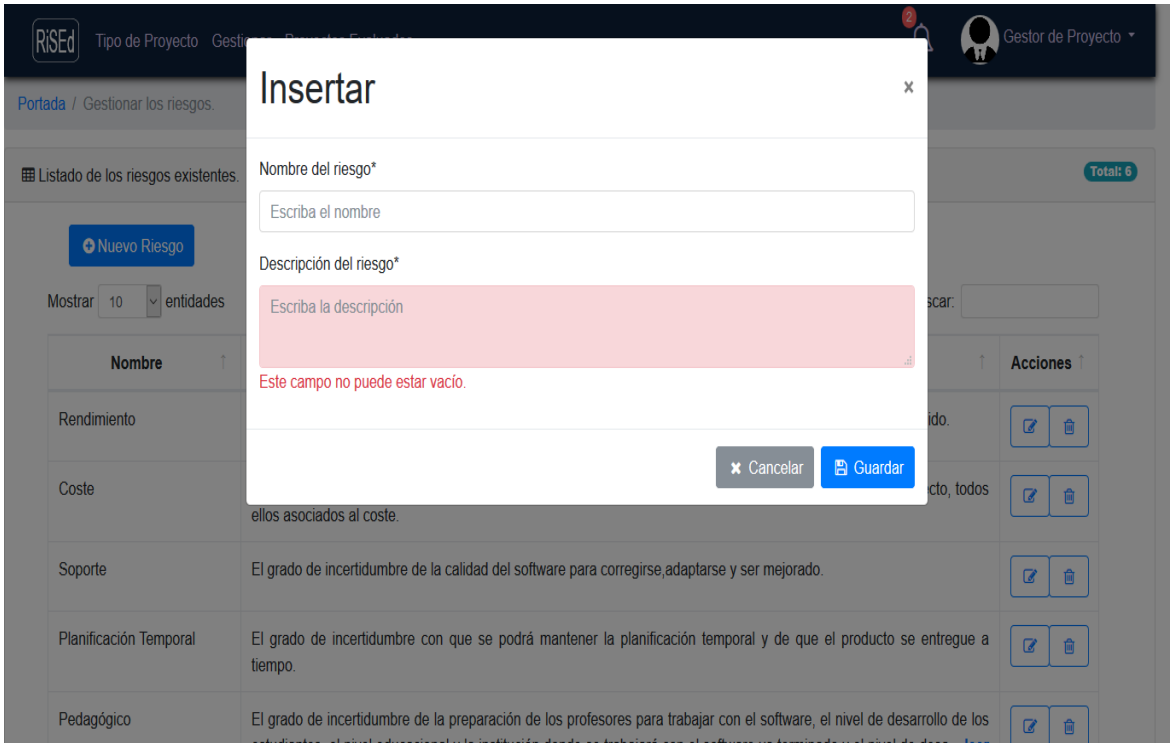
No.de Factor	Factor de Ajuste	Min	Max
1	Comunicación de Datos	0	5
2	Proceso Distribuido	0	5
3	Objetivos de Rendimiento	0	5
4	Configuración de Explotación Compartida	0	4
5	Tasa de transacciones	0	5
6	Entrada de Datos en Línea	0	5
7	Eficiencia con el Usuario Final	0	5
8	Actualizaciones en Línea	0	5
9	Lógica de Proceso Interno Compleja	0	5
10	Reusabilidad del Código	0	5
11	Conversión e Instalación contempladas	0	5

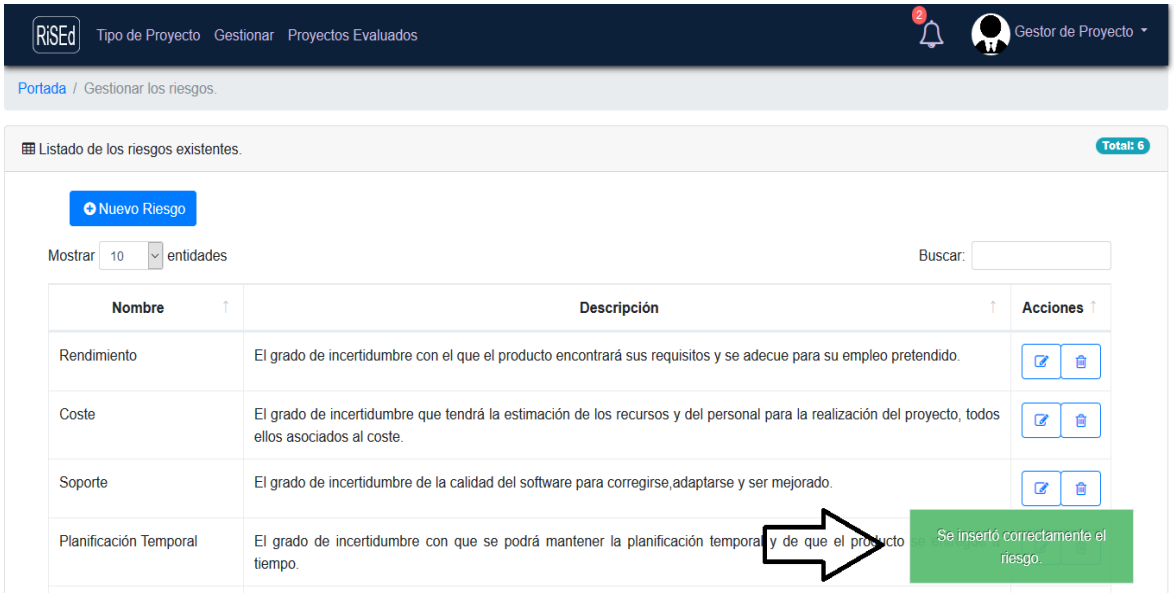
12	Facilidad de Operación	0	5
13	Instalaciones Múltiples	0	5
14	Facilidad de Cambios	0	5

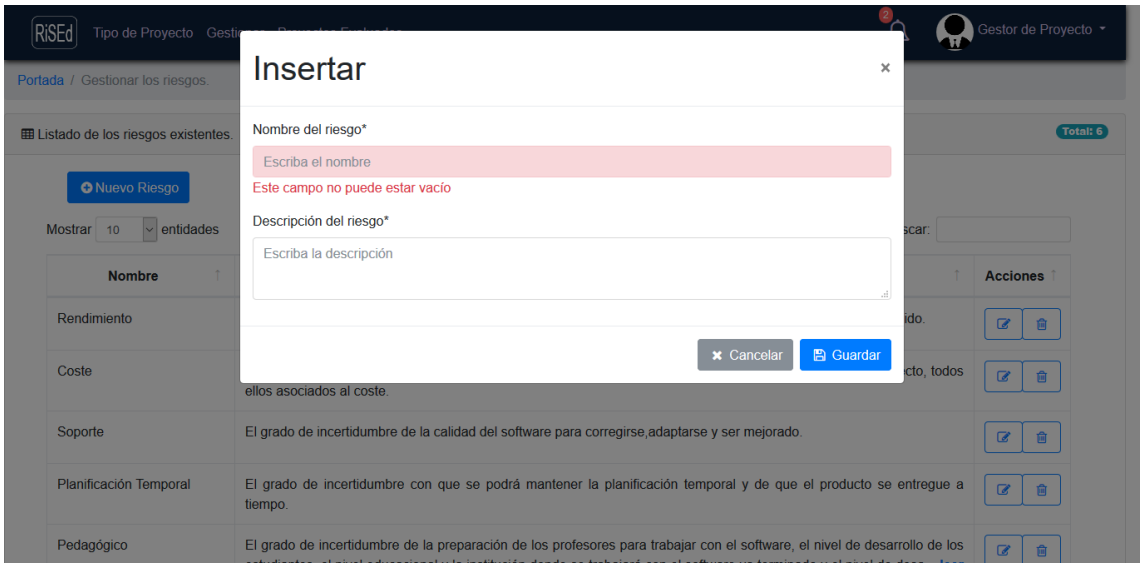
Anexo V: Tabla para obtener las Líneas por Puntos de Función

Entorno y Lenguaje	Líneas de Código por PF	Horas por PF
Lenguajes 2GL: Ensamblador, C,…	300	20 a 30
Lenguajes 3GL: Cobol	100	10 a 20
Lenguajes 4GL: VisualXX	20	5 a 10

Anexo VI: Figuras relacionadas con los casos de prueba.




RiSEd
Tipo de Proyecto
Gestionar
Proyectos Evaluados
Gestor de Proyecto
Portada / Gestionar los riesgos.
Listado de los riesgos existentes.
Total: 6
Nuevo Riesgo
Mostrar 10 entidades
Buscar:
Nombre
Descripción
Acciones
Rendimiento
El grado de incertidumbre con el que el producto encontrará sus requisitos y se adecue para su empleo pretendido.
Coste
El grado de incertidumbre que tendrá la estimación de los recursos y del personal para la realización del proyecto, todos ellos asociados al coste.
Soporte
El grado de incertidumbre de la calidad del software para corregirse,adaptarse y ser mejorado.
Planificación Temporal
El grado de incertidumbre con que se podrá mantener la planificación temporal y de que el producto
tiempo.
Se insertó correctamente el riesgo.


Insertar
Nombre del riesgo*
Escriba el nombre
Este campo no puede estar vacío
Descripción del riesgo*
Escriba la descripción
Cancelar
Guardar