Universidad de Matanzas

Facultad de Ciencias Técnicas

Ingeniería Informática

Trabajo de Diploma para optar el título de Ingeniero Informático

Sistema informático para la gestión de riesgos informáticos en la Fiscalía Provincial de Matanzas

Autor: Conrado Rodríguez Valdés.

Tutor: Dr.C Josue Segura Montero.

Matanzas, Cuba

Junio, 2020

DEDICATORIA

A mis abuelos por todo su amor incondicional y sin los cuales nunca hubiese sido quien soy.

A mi madre por su amor y apoyarme en todo momento.

Agradecimientos

Durante el trayecto de esta etapa académica que culmina, deseo agradecer el apoyo recibido de diversas personas a ellas mis más humildes agradecimientos. Un especial agradecimiento, a mi dúo de estudio, Alejandro Moreno y Omar Guerra, que sin los cuales no hubiese sido tan grato el trayecto recorrido hasta ahora y de los cuales he aprendido mucho. Un eterno agradecimiento a mi tutor Josué por el apoyo que me brindó. Y mi más grande agradecimiento para mi mamá, por ser la persona principal de que yo haya llegado hasta aquí.

Declaración de autoría

Yo, Conrado Rodríguez Valdés, declaro que soy el único autor de este trabajo y autorizo a la Universidad de Matanzas a que haga el uso que estimen pertinente de él.

Y para que así conste, firmo la presente a los 14 días del mes de julio de 2020.

Resumen

La presente investigación se realiza a partir de los problemas existentes en la Fiscalía Provincial de Matanzas en la realización del análisis de riegos informáticos. Actualmente, este proceso se realiza de forma manual, por una entidad externa con el apoyo de documentos basados en Microsoft Excel, generando gastos financieros y el acceso a información clasificada por parte de personal externo a la entidad. Para el desarrollo de este proceso, se emplea la metodología Magerit, que aborda tanto la detección como el análisis de los riesgos informáticos. Como resultado de la investigación realizada, se desarrolló un sistema informático que, adscribiéndose a la metodología utilizada por la entidad, recopila y procesa toda la información necesaria, permitiendo una mayor eficiencia en el tratamiento de los riesgos informáticos detectados, genera el plan de análisis de riesgos y cuenta con una plataforma que permite la toma de decisiones en esta área. Para el desarrollo de dicha solución, se utilizó la metodología ágil de desarrollo de software SCRUM, como IDE de desarrollo Visual Studio, lenguaje de programación C#, framework de desarrollo NetFramework y gestor de base de datos SQL Server. Se obtuvo como resultado favorable la automatización del análisis de riegos en la entidad.

Palabras claves: Análisis, Riesgos, Metodología de Desarrollo, Scrum, Magerit.

Abstract

This investigation is carried out based on the existing problems in the Matanzas Provincial Prosecutor's Office in the analysis of computer risks. Currently this process is carried out manually, by an external entity with the support of documents based on Microsoft Excel, generating financial expenses and access to classified information by personnel external to the entity. For the development of this process, the Magerit methodology is used, which addresses both the detection and analysis of computer risks. As a result of the research carried out, a computer system was developed that, complying with the methodology used by the entity, collects and processes all the necessary information, allowing greater efficiency in the treatment of the computer risks detected, generating the analysis plan for risks and have a platform that allows decision-making in this area. For the development of said solution, the agile methodology of SCRUM software development, NetFramework, C # programming language and SQL Server database manager was used. The automation of the risk analysis in the entity was obtained as a favorable result.

Keywords: Analysis, Risk, Development Methodology, Scrum, MAGERIT

Índice

Índice de tablas

Índice de tablas

Índice de Ilustraciones

Introducción

La seguridad informática juega un papel fundamental en el avance científico-técnico del mundo actual. De una estructura informática basada en sistemas propietarios y grandes servidores, manejada por el personal técnico con una formación específica y alejada del conocimiento común del resto, ha evolucionado a otra más amigable y cercana al usuario final. Ello ha supuesto que los niveles iniciales de conocimiento sean rápidamente adquiridos por cualquier persona interesada, sin especiales conocimientos técnicos en la materia

La seguridad informática a través de los tiempos ha sido un tema crítico dentro de la informática. El propio avance tecnológico de la sociedad, los casos de fraude, robo y acciones poco éticas a través de los medios de comunicación y de la informática, son unos de los factores de gran importancia para la gestión de la seguridad informática. (Voutssas M., 2016)

La seguridad de las organizaciones, sistemas y redes de información están constantemente amenazadas por diversas fuentes que incluyen ataques de distintos tipos y origen: la ocurrencia de catástrofes, errores de operación y negligencias, aumentan los riesgos a que están expuestos los servicios y protocolos utilizados, así como el contenido de la información tratada en dichos sistemas, todo lo cual puede afectar severamente la confidencialidad, integridad y disponibilidad de la información.

La información y los procesos que las apoyan, los sistemas y las redes son bienes importantes de las entidades, por lo que requieren ser protegidos convenientemente frente amenazas que pongan en peligro la integridad, disponibilidad, la confidencialidad de la información, la estabilidad de los procesos, los niveles de competitividad, la imagen corporativa, la rentabilidad y a legibilidad, aspectos necesarios para alcanzar los objetivos de la organización (OSRI, 2010)

Una amenaza se puede definir como el evento que puede desencadenar un incidente de la organización, produciendo daños o pérdidas materiales en sus activos. (Castro Bolaños & Rojas Mora, 2013). Importantes amenazas se concentran en actos de vandalismo, destrucción de datos, destrucción de medios, control de las redes y otros

El incipiente desarrollo que aún tiene Cuba en el empleo a gran escala de las Tecnologías de la Información y la Comunicación (TIC), no presume que el país este expuesto a las mismas amenazas que países con mucha mayor experiencia, tanto en los objetivos e impacto de los ataques a redes, como en las causas y motivaciones de los presuntos atacantes.

En Cuba, a pesar de que aún no se ha logrado alcanzar tales niveles de dependencia de la tecnología informática, las principales autoridades en temas de informatización recomiendan de manera urgente

estudiar la vulnerabilidad de sus sistemas informáticos, todos los cuales sin excepción están expuestos a las más variadas y comprometidas amenazas.

Los avances alcanzados en los últimos años en la informatización de la sociedad con el incremento de tecnologías de la información en todos los sectores, y el impulso orientado por la dirección del país al desarrollo acelerado de programas que multipliquen dichos logros, requieren la adopción de medidas que garanticen un adecuado nivel de seguridad para su protección y ordenamiento.

En nuestro país existen normas tales como el Decreto Ley 360, las resoluciones 128 y 129 de 2019 del MINCOM, que establecen los pasos a seguir en el análisis de riesgos en los medios de la información. Existen entidades autorizadas por el estado a realizar el análisis de los riesgos en otras entidades. Método del cual se apoya la Fiscalía Provincial de Matanzas, contratando a la empresa Desarrollo de Software (DESOFT), para realizar esta actividad. El cual se realiza de documentos basados en Microsoft Excel. Esto conlleva a que personas externas a la entidad tengan acceso a información clasificada, no tener un historial de los análisis anteriormente realizados, gasto monetario cada vez que se realiza la acción, además que el proceso se realiza de forma manual.

A raíz de la situación problémica anteriormente planteada se formula el siguiente **problema científico:** ¿Cómo contribuir al análisis de los riesgos informáticos en la Fiscalía Provincial de Matanzas a través de la informatización?

Por lo que se genera la siguiente **hipótesis**: si se implementa un sistema informático, se logrará una mayor efectividad en la gestión de los riesgos informáticos en la Fiscalía Provincial de Matanzas.

El **objeto de estudio** de esta investigación constituye el análisis de riesgos informáticos. Se define como **campo de acción** la automatización del análisis de riesgos informáticos en la Fiscalía Provincial de Matanzas

Se trazó como **objetivo general** desarrollar un sistema informático que contribuya al análisis de riesgos informáticos en la Fiscalía Provincial de Matanzas.

Para complementar el objetivo general se trazan los siguientes **objetivos específicos**:

- Determinar los antecedentes y tendencias actuales del análisis de riesgos en la Fiscalía Provincial de Matanzas
- Diseñar un sistema informático para la gestión de riesgos informáticos en la Fiscalía Provincial de Matanzas utilizando la metodología de desarrollo Scrum.
- Validar sistema informático para la gestión de riesgos informáticos en la Fiscalía Provincial de Matanzas.

Durante la investigación se utilizaron diversos **métodos y técnicas de metodología de la**

investigación, tales como:

Métodos teóricos

- Análisis histórico – lógico: aportó el análisis de las metodologías, tecnologías y los procesos a informatizar a partir de su desarrollo histórico y las fases en que estuvo compuesto.
- Analítico-sintético: posibilitó la revisión bibliográfica y el análisis de los resultados, permitiendo descomponer ele todo en sus partes y cualidades, la división del todo en sus múltiples relaciones para luego unir las partes analizadas, descubrir las relaciones y características generales entre ellas.
- Inductivo-deductivo: fue utilizado para plantearse las hipótesis de trabajo respecto al análisis de riesgos y la pertinencia del sistema informático que se propone.

Empíricos

- Observación: se observaron los procesos referentes al análisis de riesgos en la entidad objeto de estudio que permitieron detallar las acciones que ejecutaban en la realización del mismo.
- Entrevista: fue útil en distintos momentos de la investigación; fundamentalmente al inicio, cuando se realizó el levantamiento de requisitos para efectuar una exploración preliminar del problema a investigar. Se entrevistaron a los funcionarios y responsables del proceso para dar cumplimiento a las tareas planteadas como vía de obtención de datos.
- Análisis de documentos: Se realizó una revisión de la documentación llevada en el proceso de análisis de riesgos en la entidad, la metodología por la cual se rige, así, como las resoluciones que rigen la seguridad informática en el país.

Entre los **aportes** de la investigación se destacan:

- El teórico-investigativo, al integrar los procedimientos tradicionales más utilizados por autores relacionados con el tema, a través de los diferentes artefactos de la metodología de desarrollo de software que permitió orientar metodológicamente la secuencia de acciones lógicas a desarrollar; y los elementos a tener en cuenta para la continuidad de la investigación.
- El práctico, al desarrollar un sistema automatizado que asista al análisis de riesgos en la entidad.

Atendiendo a lo planteado anteriormente, la tesis queda estructurada en introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas, según sigue.

Estructura de la investigación

- **Capítulo I: "Marco Teórico Referencial":** se plantean las definiciones fundamentales asociadas al campo de acción. Se hace un estudio sobre los antecedentes, las tendencias y

tecnologías que serán usadas. Se exponen las características fundamentales de los lenguajes de programación, los gestores de bases de datos y las características fundamentales de las metodologías ágiles de desarrollo de software.

- **Capítulo II: Análisis, Diseño y Desarrollo de la Solución Propuesta:** se argumenta la solución que se propone al problema de investigación, presentando una planificación inicial del proyecto, con el empleo de la metodología ágil de desarrollo de software Scrum. Se desarrolla la solución propuesta, presentando una planificación por iteraciones.
- **Capítulo III: Validación de la Solución Propuesta:** se realizan pruebas funcionales y se hace un análisis de los resultados obtenidos, basándose en el criterio de los clientes y los propios de desarrollo de software.
- **Conclusiones**
- **Recomendaciones**
- **Bibliografías**
- **Anexos**

1. Capítulo I: Marco Teórico Referencial

1.1. Introducción

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

1.2. Análisis de riesgos

La masificación de las Tecnologías de Información (TI) y su adopción en las empresas, las han convertido también en blanco de ataques, con los riesgos asociados que aumentan y se transforman; por ello, se hace necesario crear y adaptar constantemente los medios y métodos utilizados en la seguridad de la información, para conservar la información crítica o sensible que las organizaciones actualmente necesitan proteger (Arévalo, 2017)

El análisis de riesgos informáticos constituye una parte fundamental en la administración de la seguridad, permitiendo algunos beneficios, tal como, identificar los puntos más débiles de la estructura de las TIC que da soporte a los procesos críticos de la organización. Igualmente, además de ser una guía de selección de medidas de protección de costo adecuado, determina dónde es necesario contar con esquemas de recuperación de desastres y continuidad de negocio y permite realizar políticas de seguridad mejor adaptadas a las necesidades de la organización.

La aparición de la ISO 31000:2009 supuso un gran cambio en la forma de entender el análisis de riesgos. Este estándar, orienta el análisis de riesgos al negocio en su conjunto y establece una metodología que permite unificar los criterios y comparar los diferentes riesgos

Los principios básicos de la gestión del riesgo que se describen en la norma ISO 31000 corresponden a: crear y proteger el valor, es una parte integral de todos los procesos de la organización, es parte de la toma de decisiones, trata explícitamente la incertidumbre, es sistémica, estructurada y oportuna, se basa en la mejor información disponible, es adaptable, integra los factores humanos y culturales, es transparente y participativa, es dinámica, iterativa, responde a los cambios y facilita la mejora continua de la organización.

Los sistemas informáticos de una organización están constantemente sometidos a amenazas, que involucran desde fallos técnicos hasta acciones no deseadas. Una metodología de análisis de riesgos informáticos permite determinar que probabilidad existe de que estas amenazas ocurran y del impacto que producirían si sucedieran, además de la vulnerabilidad que los sistemas informáticos pueden presentar.

1.2.1. Metodología de análisis riesgos

En el ámbito de la seguridad informática, las metodologías de análisis de riesgos conforman una disciplina que se articula desde los Sistemas de Gestión de Seguridad Informática SGSI en las organizaciones, realizando unos importantes escaneos de vulnerabilidades mediante el uso de una serie de modelos y procesos para, así, proponer una forma más segura de cuidar la información y los recursos de TIC.

Algunos de los objetivos de las metodologías de análisis de riesgos corresponden a: planificación de la reducción de riesgos, prevención de accidentes, visualización y detección de las debilidades existentes en los sistemas y ayuda en la toma de las mejores decisiones en materia de seguridad de la información. En la seguridad de la información existen diversas metodologías de análisis de riesgos dentro de las que sobresalen:

- **ITIL**: Information Technology Infraestructure Library “proporciona un planteamiento sistemático para la provisión de servicios de TI con calidad”.
- **COBIT 5**: Significa (Objetivos de control para la información y tecnologías relacionadas) es una metodología publicada en 1996 por el Instituto de Control de TI y la ISACA (Asociación de Auditoría y Control de Sistemas de Información).
- **CRAMM:** “CCTA Risk Assessment and Management Methodology” fue originalmente desarrollado para uso del gobierno de UK pero ahora es propiedad de Siemens;
- **ISO TR 13335:** fue el precursor de la ISO/IEC 27005;
- **MAGERIT** “Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información”.
- **OCTAVE**: “Operationally Critical Threat, Asset, and Vulnerability Evaluation” Metodología de Análisis y Gestión de Riesgos desarrollada por el CERT;
- **NIST SP 800-39** “Gestión de Riesgos de los Sistemas de Información, una perspectiva organizacional”;
- **NIST SP 800-30:** Guía de Gestión de Riesgos de los Sistemas de Tecnología de la Información, es gratuito;
- **Mehari:** Método de Gestión y Análisis de Riesgos desarrollado por CLUSIF (Club de la Sécurité de l'Information Français);
- **AS/NZS:** Norma de Gestión de Riesgos publicada conjuntamente por Australia y Nueva Zelanda y ampliamente utilizada en todo el mundo.

La metodología a usar en la investigación es la Magerit, por ser utilizada en la empresa DESOFT encargada de desarrollar el análisis de riesgo en la Fiscalía Provincial de Matanzas.

1.3. Antecedentes

Con el objetivo de investigar la existencia de aplicaciones informáticas que guardan semejanza con la idea que se propone desarrollar en el trabajo de diploma, se consultaron diversas herramientas para la gestión de análisis de riesgo del ámbito nacional e internacional, a saber:

- **RiesgosTI**: es una herramienta que permite la evaluación y gestión de los riesgos en Tecnologías de la Información, a partir de la aplicación de la Resolución 127 del Ministerio de la Informática y las Comunicaciones de Cuba. Está implementada en el lenguaje de programación Java por su potencia para desarrollar aplicaciones en cualquier ámbito, su dinamismo y su propósito general, además de incorporar una base de datos en MySql. (Orta Cruz. & Rodríguez Cárdenas, 2013)
- **DIÓGENES**: Dado el hecho de que el Ministerio de la Informática y las Comunicaciones tiene como atribución estatal la ejecución de inspecciones en materia de Seguridad a las Tecnologías de la Información, es que ha sido concebido un sistema automatizado para la gestión de riesgos en Tecnologías de la Información (DIÓGENES), que implementa los contenidos esenciales que aparecen en el Reglamento de Seguridad para las Tecnologías de la Información y que facilita tanto a la empresa (en términos de autoevaluación), como a los inspectores estatales (al realizar una evaluación), el conocimiento del grado de cumplimiento del mencionado reglamento y el plan de acciones a tomar. Esta herramienta informática se ejecuta sobre el sistema operativo Windows. (Orta Cruz. & Rodríguez Cárdenas, 2013)
- **SoftExpert Riesgo**: contempla todos los aspectos del proceso de gestión de riesgos, desde la identificación inicial del riesgo, pasando por la evaluación y análisis, hasta la mitigación y el monitoreo, administrando los incidentes y garantizando la ejecución de las acciones y la debida comunicación. El software se adecúa en varios departamentos de la organización, lo que significa que la gestión de riesgo automáticamente estará presente en los proyectos, procesos y estrategias de la empresa; en el desarrollo de productos; medio ambiente, salud y seguridad; en las prácticas de gobernanza; en la gestión de TI; y muchos otros. (SoftExpert Riesgo, n.d.). Es software propietario
- **KAWAK:** es un software para centralizar la información y para organizar y focalizar su empresa en la calidad y en la mejora continua y sostenible. Cuente con un Sistema Integral de Gestión e indicadores alineados con la estrategia organizacional. Atienda de manera sencilla la normatividad de Seguridad y Salud en el trabajo y del Talento Humano, las PQRS de su

empresa y sus Auditorías. KAWAK cubre las normas ISO 9001, ISO 14001, ISO 27001, ISO 31000, OHSAS 18001, RSE, GP1000, MECI, entre otros. (KAWAK®, Software para la administración de Sistemas integrales de Gestión, s.f.). Es software propietario

- **RiskyProject:** es un conjunto completo de software de análisis y gestión de riesgos de proyectos en un paquete integrado que es fácil de usar, se integra con Microsoft Project, Primavera y otras herramientas de planificación y planificación, y cubre todo el ciclo de vida del riesgo. RiskyProject incluye cronograma cuantitativo y cuantitativo de Monte Carlo y análisis de riesgo de costos. RiskyProject también incluye un completo Registro de riesgos. Con RiskyProject, ya no necesita dos o más aplicaciones para realizar el análisis de riesgos del proyecto Monte Carlo y administrar los riesgos de su proyecto. Ahora se encuentran en un software de escritorio fácil de usar y asequible. (RiskyProject: software de análisis de riesgos del proyecto, s.f.). Es software propietario
- **PILAR**: es un conjunto de herramientas cuya función es el análisis y la gestión de riesgos de un sistema de información siguiendo la metodología Magerit (Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información) y que ha sido desarrollada en coordinación con el CCN y parcialmente financiada por el mismo. (València, 2019). Es software propietario.

Luego de consultar diversas aplicaciones utilizadas para el análisis de riesgos informáticos, no se encontró una solución que cumpliera todos los requisitos necesarios establecidos por la entidad. Algunos de estos no presentaban los requerimientos funcionales definidos o están obsoletos y otros que a pesar de cumplir las expectativas necesarias son de uso propietario o de conexión en línea. Siendo necesario el desarrollo de la solución propuesta.

1.4. Herramientas y tecnologías

Para dar cumplimiento al objetivo general se estudiaron diversas herramientas, tecnologías y metodologías de desarrollo. Seguidamente, se mencionan elementos esenciales de las mismas:

- **Aplicaciones de Escritorio**: sistema informático instalado en el ordenador del usuario, que es ejecutado directamente por el sistema y cuyo rendimiento depende generalmente del hardware del equipo en el que se ejecuta. (clinicbox, 2018)

  Los formularios Windows Forms se utilizan para desarrollar aplicaciones en las que se espera que el cliente maneje una parte significativa de la carga de trabajo de la aplicación. Entre ellas, se incluyen las aplicaciones clásicas de escritorio Win32 que solían desarrollarse en versiones anteriores de Visual Basic y Visual C++. Ejemplo de ello son las aplicaciones gráficas o de diseño, los sistemas de entrada de datos, los sistemas de punto de venta y los juegos.

Todas estas aplicaciones tienen en común el hecho de que dependen de la potencia del equipo de escritorio para el procesamiento y la presentación de contenidos de alto rendimiento. Algunas aplicaciones de formularios Windows Forms pueden estar completamente autocontenidas y ejecutar todo el proceso de la aplicación en el equipo del usuario. A menudo, los juegos se escriben de este modo. Otras pueden ser parte de un sistema mayor y, fundamentalmente, utilizan el equipo de escritorio para procesar los datos proporcionados por el usuario.

Características de las Aplicaciones de Escritorio.

- **Implementación:** Permiten su implementación "sin contacto": las aplicaciones se pueden descargar, instalar y ejecutar directamente en los equipos de los usuarios, sin ninguna alteración del Registro.
- **Gráficos:** Incluyen GDI+, que permite utilizar gráficos sofisticados para juegos y otros entornos gráficos extremadamente ricos.
- **Capacidad de respuesta:** Pueden ejecutarse por completo en el equipo cliente; pueden proporcionar la respuesta más rápida para aquellas aplicaciones que necesiten un alto grado de interactividad.
- **Formularios y control de flujo de texto:** Las funciones de posición de cuadrícula de formularios Windows Forms proporcionan un preciso control bidimensional (coordenadas x e y) para la colocación de controles. Para mostrar texto en Windows Forms, es necesario insertarlo primero en algún control como **Textbox** o **RichTextBox** (las posibilidades de formato son limitadas).
- **Plataforma:** Los formularios Windows Forms requieren que .NET Framework se ejecute en el equipo cliente.
- **Acceso a recursos locales (sistema de archivos, Registro de Windows):** Las aplicaciones, si se les permite, pueden tener acceso total a los recursos del equipo local. Si es preciso, se pueden restringir los permisos de la aplicación para que no utilice recursos específicos.
- **Modelo de programación:** Los formularios Windows Forms se basan en un modo de suministro de mensajes Win32 en el cliente, donde el desarrollador crea, utiliza y descarta instancias de componentes.
- **Seguridad:** Los formularios Windows Forms utilizan permisos granulares en su implementación de seguridad de acceso a código a fin de proteger los recursos del equipo y la información reservada. Esto permite exponer cuidadosamente la funcionalidad al mismo tiempo que se conserva la seguridad. Por ejemplo, el permiso de impresión, que en un nivel sólo permitiría

imprimir en la impresora predeterminada, en otro nivel podría permitir la impresión en cualquier impresora.

Ventajas:

- Hay un mejor aprovechamiento de los recursos del equipo en el cual se ejecuta.

Desventajas:

- Requiere instalación en cada cliente.
- Se debe utilizar algún sistema de control de versiones para actualizar cada estación de trabajo y en ocasiones actualizar de forma manual computador por computador.
- La mayoría del código se ejecuta en el pc del cliente y en muchas ocasiones ralentizando el rendimiento de este.
- No puede ejecutarse en cualquier equipo debido a su arquitectura.

Se eligió para el desarrollo de este sistema una aplicación de escritorio ya que el sistema procesa información clasificada, por lo cual debe ejecutarse en una PC aislada de la red.

Microsoft Visual Studio es un entorno de desarrollo integrado, creado por la compañía Microsoft y disponible para sistemas operativos Windows, Linux y macOS, y la vez es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET, fue lanzado en 1997, cuenta con versiones gratis y de venta. (Escobar, 2020).

Se escogió como IDE de desarrollo Microsoft visual Studio 2019 Comunity ya que es libre, es desarrollado por Microsoft una de las compañías más grandes e importantes en temas de informatización. Presenta buena seguridad y se mantiene en constante actualización. Es uno de los IDE más utilizados a nivel mundial. Presenta una interfaz amigable y muchos atajos que ayudan a el desarrollo

.NET Framework: Un framework consiste en una serie de estructuras y tecnología definidas que básicamente facilita la programación. Cuando hablamos de .NET Framework, estamos hablando de este conjunto de estructuras y tecnologías que proporciona Microsoft para una programación más sencilla orientada a las redes e internet, con independencia de la plataforma hardware utilizada. Para programar en .NET existen hoy en día más de 20 de lenguajes de programación, pero C# y Visual Basic son los más populares (no existe un lenguaje de programación propio .NET). Otros lenguajes de programación que soportan .NET son Delphi (Object Pascal), C++, F#, Python, J# Fortran, Perl, Prolog.

Existen muchas herramientas que utilizan el .Net Framework para desarrollar apps para móviles, como Xamarin, que permite a los desarrolladores escribir código en C# bajo un entorno .Net Framework y que el mismo sea traducido para ejecutarse en dispositivos Android, IOS o Windows Phone. (Robledano, 2019)

Se escogió como framework .NET Framework ya que es multiplataforma, compatible con el IDE de trabajo. Posee un conjunto de estructuras y tecnologías para una programación más sencilla, organizada y segura.

Visual Paradigm: es una herramienta de diseño y administración poderosa, multiplataforma y sin embargo fácil de usar para sistemas de TI. Visual Paradigm. Proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad más rápido, mejor y más barato. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes que excede todo su proceso de desarrollo Model-Code-Deploy en esta solución integral. (Sierra, 2007)

Se seleccionó como herramienta de desarrollo Visual Paradigm ya que es multiplataforma. De muy fácil trabajo e intuitiva. Compatible con el gestor de base de datos utilizado en el sistema informático desarrollado.. Además de ser uno de los más usados a nivel mundial.

Lenguajes de programación: Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para, de esa forma, crear programas que controlen el comportamiento físico y lógico de una máquina.

C#: (en inglés es pronunciado como “C Sharp”, en español como “C Almohadilla”), es un lenguaje de programación diseñado por la conocida compañía Microsoft. Fue estandarizado en hace un tiempo por la ECMA e ISO dos de las organizaciones más importantes a la hora de crear estándares para los servicios o productos. El lenguaje de programación C# está orientado a objetos.

C# es considerado como una evolución y necesidad de ciertas circunstancias. Evolución por sus lenguajes antecesores que son el C y el C++ y necesidad a la hora en que la compañía tuvo problemas con la empresa creadora del lenguaje Java. Es por lo anterior que C Sharp presenta los atributos positivos de C++, Java y Visual Basic y los mejora otorgando un lenguaje fuerte y actualizado para los tiempos actuales. (Rivera, 2018).

Se eligió C# como lenguaje de programación porque además de ser orientado a objeto es un lenguaje de alto nivel. Es compatible con el IDE y Framework de trabajo. Evolución de sus lenguajes antecesores que son el C y el C++.

SQLServer:

Microsoft SQL Server es la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos. Es un sistema de gestión de base de datos relacional desarrollado como un servidor que da servicio a otras aplicaciones de software que pueden funcionar ya sea en el mismo ordenador o en otro ordenador a través de una red (incluyendo Internet).

Se escogió Sql Server porque fue desarrollado por Microsoft, es un potente servidor de base de datos. Es compatible con el framework y el lenguaje de programación. Además de estar integrado al IDE de trabajo. Este gestor es además un requisito no funcional exigido por el cliente.

1.5. Metodologías de desarrollo

Una metodología de desarrollo es un conjunto de métodos que especifican quién debe hacer qué cosa, cuándo debe hacerse, estableciendo un conjunto de roles (el quién), actividades (la cosa) y un ciclo de vida que establece las diferentes fases (González-Hernández & Coloma-Carrasco, 2018).
Durante el transcurso de los años han surgido diferentes metodologías. Algunas hacen más énfasis en el control del proceso, imponiendo rigurosamente las actividades y los artefactos que se deben producir, las herramientas y notaciones a utilizar, las cuales se denominan metodologías tradicionales, que son utilizadas para grandes proyectos; y otras denominadas metodologías ágiles que se centran más en las personas que en el proceso, dan mayor valor al individuo, al equipo de desarrollo y a su colaboración con el cliente; proponiendo un desarrollo creciente del software con iteraciones muy cortas y la documentación necesaria.

Metodología ágil: proceso que permite al equipo dar respuestas rápidas e impredecibles a las valoraciones que reciben sobre su proyecto. Crea oportunidades de evaluar la dirección de un proyecto durante el ciclo de desarrollo. Los equipos evalúan el proyecto en reuniones regulares, llamadas sprints o iteraciones.

El método ágil es un proceso de empoderamiento que ayuda a las empresas a diseñar y crear el producto idóneo. El proceso de gestión es muy beneficioso para las compañías de software porque les permite analizar y mejorar su producto durante el desarrollo del mismo. Esto da a las empresas la capacidad de fabricar un producto valioso, de manera que se mantengan competitivas en el mercado. (Gonçalves, 2020)

La Alianza Ágil elaboró un conjunto de doce principios comunes a las metodologías ágiles de desarrollo que se enuncian a continuación:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
- El software que funciona es la principal medida del progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica enaltece la agilidad.
- La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

  La utilización de todas las buenas prácticas enumeradas en el manifiesto ágil no implica ser ágil, sin embargo, el hecho de incumplir una de ellas te transforma en no ágil. (Pérez Pérez, 2012)

El manifiesto ágil tiene cuatro valoraciones importantes:

- El foco se debe poner más sobre las personas y las interacciones que sobre los procesos y herramientas
- El software funcionando es más importante que la documentación extensiva
- La colaboración con el cliente es más importante que la negociación contractual
- El proceso debería responder ante el cambio, en lugar de seguir un plan

XP: La metodología XP o Programación Extrema es una metodología ágil y flexible utilizada para la gestión de proyectos. Extreme Programming se centra en potenciar las relaciones interpersonales del equipo de desarrollo como clave del éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo. Esta metodología pone el énfasis en la retroalimentación continua entre cliente

y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes (Calvo, 2018)

Características

- Se considera al equipo de proyecto como el principal factor de éxito del proyecto.
- Software que funciona por encima de una buena documentación.
- Interacción constante entre el cliente y el equipo de desarrollo.
- Planificación flexible y abierta.
- Rápida respuesta a cambios.

Kanban: Esta metodología busca conseguir un proceso productivo, organizado y eficiente. Se creó en Toyota (Japón) y se utiliza para controlar el avance del trabajo en una cadena de producción. Forma parte de la metodología Lean Manufacturing basada en la utilización de técnicas just-in-time (JIT). El principal objetivo del sistema Kanban es asegurar una tasa de producción sostenible para evitar exceso de producto terminado, cuellos de botella y retrasos en la entrega de pedidos. Los trabajos en curso deben organizarse en función de la capacidad del centro de trabajo y equipos. Requiere una comunicación en tiempo real sobre la capacidad y una transparencia del trabajo total. (Lendínez, 2019)

Scrumban: En esta metodología el trabajo en equipo se organiza en pequeñas iteraciones y se monitorea con la ayuda de un tablero visual. Las reuniones de Planificación bajo demanda se llevan a cabo cuando es necesario determinar qué Historias de usuarios y tareas se completarán en la próxima iteración. Para mantener las iteraciones cortas, se utiliza el límite de trabajo en progreso (WIP/ Work in Progress for sus siglas en Inglés). Cuando WIP cae por debajo de un nivel predeterminado, se establece un activador de planificación bajo demanda para que el equipo sepa cuándo planificar a continuación. (GERMANOV, 2019)

De Scrum

- Roles: Cliente, equipo (con los diferentes perfiles que se necesiten).
- Reuniones: reunión diaria.
- Herramientas: pizarra

De Kanban

- Flujo visual • Hacer lo que sea necesario, cuando sea necesario y solo la cantidad necesaria.
- Limitar la cantidad de trabajo (WIP)
- Optimización del proceso.

Scrum

> Marco de trabajo o framework que se utiliza dentro de equipos que manejan proyectos complejos. Es decir, se trata de una metodología de trabajo ágil que tiene como finalidad la entrega de valor en períodos cortos de tiempo y para ello se basa en tres pilares: la transparencia, inspección y adaptación. (Abellán, 2020)

Tabla Comparativa de metodologías de desarrollo. Ver anexo 1

Analizando el siguiente cuadro comparativo proporcionado por (Pérez Pérez, 2012) y teniendo en cuenta las necesidades del cliente se ha determinado que la metodología más apropiada para el desarrollo del proyecto es SCRUM.

1.6. Metodología Scrum

Marco de trabajo o framework que se utiliza dentro de equipos que manejan proyectos complejos. Es decir, se trata de una metodología de trabajo ágil que tiene como finalidad la entrega de valor en períodos cortos de tiempo y para ello se basa en tres pilares: la transparencia, inspección y adaptación. (Abellán, 2020)

¿En qué se basa la metodología Scrum?

Al estar enmarcada dentro de las metodologías agiles, Scrum se basa en aspectos como:

- La flexibilidad en la adopción de cambios y nuevos requisitos durante un proyecto complejo.
- El factor humano.
- La colaboración e interacción con el cliente.
- El desarrollo iterativo como forma de asegurar buenos resultados.

1.6.1. Pilares de la metodología scrum:

1. Transparencia

Con el **método Scrum** todos los implicados tienen conocimiento de qué ocurre en el proyecto y cómo ocurre. Esto hace que haya un entendimiento “común” del proyecto, una visión global.

2. Inspección

Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera auto-organizada.

3. Adaptación

Cuando hay algo que cambiar**,** el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir el éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

1.6.2. Roles en el equipo Scrum

Los roles en Scrum se dividen en dos grupos principales, los comprometidos y los implicados. Al grupo de los comprometidos, sin que tenga alguna connotación negativa, también se le llaman “cerdos”, y al grupo de los implicados se le llaman “gallinas”. (Mesa, 2018)

1. Product owner:

Es el responsable de maximizar el valor del trabajo del equipo de desarrollo. La maximización del valor del trabajo viene de la mano de una buena gestión del **Product Backlog**, el cual explicaremos más adelante.

El **Product owner** es el único perfil que habla constantemente con el cliente, lo que le obliga a tener muchos conocimientos sobre negocio.

Para finalizar, un equipo Scrum debe tener solo un Product Owner y este puede ser parte del equipo de desarrollo.

2. Scrum Master:

Es el responsable de que las técnicas Scrum sean comprendidas y aplicadas en la organización. Es el manager de Scrum, un líder que se encarga de eliminar impedimentos o inconvenientes que tenga el equipo dentro de un sprint (que ya revisaremos en detalle más adelante), aplicando las mejores técnicas para fortalecer el equipo de marketing digital.

Dentro de la organización, el Scrum Master tiene la labor de ayudar en la adopción de esta metodología en todos los equipos.

3. Equipo de desarrollo:

Son los encargados de realizar las tareas priorizadas por el Product Owner. Es un equipo multifuncional y auto-organizado. Son los únicos que estiman las tareas del product backlog, sin dejarse influenciar por nadie.

Los equipos de desarrollo no tienen sub-equipos o especialistas. La finalidad de esto es transmitir la responsabilidad compartida si no se llegan a realizar todas las tareas de un sprint.

1.6.3. Herramientas principales de scrum:

1. Product backlog

El **Product Backlog o pila de producto** en un proyecto que sigue la metodología Scrum consiste en una lista con todos los requerimientos iniciales del producto que se va a desarrollar. Se trata de una lista dinámica, que irá evolucionando a medida que lo hace el producto y el entorno del proyecto. La finalidad de crear esta lista no es otra que identificar las necesidades del producto para lograr su máxima utilidad.

Esta **lista de Product Backlog** contiene la descripción de las tareas y subtareas que se van a realizar para la ejecución de cada requisito. Tareas que se organizarán en función de sus prioridades. Además, la pila de producto también indica una estimación del tiempo en la que cada tarea se va a desarrollar y el valor que cada una le da al producto. (EALDE, 2019)

2. Sprint backlog

Contiene todo el trabajo que el Equipo de Desarrollo se compromete a llevar a cabo dentro de un Sprint. Este trabajo se pasa del Product Backlog al Sprint Backlog.

El Sprint Backlog es un artefacto que permite visualizar todo el trabajo incluido en el Sprint en curso y es responsabilidad del Equipo de Desarrollo. Su objetivo es dar transparencia al estado del desarrollo durante el Sprint. Por ese motivo, una de las mejores formas de representarlo es mediante una pizarra Kanban, con columnas por estados, para ver cómo progresa y evoluciona el trabajo.

Los ítems incluidos en el Sprint (sobre todo historias o tareas técnicas de cierto tamaño) se rompen normalmente en tareas más pequeñas, lo que facilita la asignación del trabajo entre los miembros del equipo, que trabajará de forma auto organizada. (TAMARIT, 2019)

1.6.4. Ventajas y desventajas de la metodología Scrum

Ventajas

- Scrum es muy fácil de aprender: los roles, hitos y herramientas son claros y tienen un objetivo por lo que es un método muy relacionado con nuestra manera diaria de trabajar.
- El cliente puede comenzar a usar el producto rápidamente.
- Se agiliza el proceso, ya que la entrega de valor es muy frecuente.
- Menor probabilidad de sorpresas o imprevistos, porque el cliente está viendo frecuentemente el proyecto.

Desventajas

- Aunque Scrum sea fácil de aprender, es muy difícil implementarlo. Esto supone una predisposición y un cambio de cultura de la organización que debe ir desde los altos mandos hasta los clientes.
- La necesidad de tener equipos multidisciplinares puede ser un problema, ya que es difícil encontrar personas que sean capaces de hacer todo el trabajo de un equipo.
- El equipo puede tender a realizar el camino más corto para conseguir el objetivo de un sprint, el cual no siempre ofrece resultados de calidad.

En definitiva, **Scrum** es especialmente interesante para proyectos en los que el objetivo es la entrega de valor continua al cliente para poder empezar a ver resultados lo antes posibles. Además, esta metodología permite agilizar procesos, practicar la transparencia y motivar al equipo a través de la autonomía y la independencia.

1.7. Conclusiones parciales

Se encontraron herramientas informáticas en el campo de acción de la investigación, pero al analizarlas se confirmó que no son las herramientas adecuadas para esta entidad por sus características y necesidades particulares. El problema debe ser resuelto a partir de la implementación de un primer módulo dedicado al análisis de riesgos. El ambiente de programación Windows Form en el entorno de desarrollo Visual Studio 2019, con el lenguaje C# permite el desarrollo de aplicaciones de escritorio, lo que se ajusta para llevar a cabo el objetivo general de esta investigación. La metodología programación Scrum es adecuada para la planificación y construcción de esta aplicación, por poseer una alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente, lo que permite corregir problemas y mitigar riesgos de forma temprana.

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

2.1. Captura de requisitos

Requisitos Funcionales: Son funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reaccionará a determinados insumos. Cuando hablamos de las entradas, no necesariamente hablamos sólo de las entradas de los usuarios. Pueden ser interacciones con otros sistemas, respuestas automáticas, procesos predefinidos. En algunos casos, los requisitos funcionales de los sistemas también establecen explícitamente lo que el sistema no debe hacer. Es importante recordar esto: un RF puede ser también una declaración negativa. Siempre y cuando el resultado de su comportamiento sea una respuesta funcional al usuario o a otro sistema, es correcto. Y más aún, no sólo es correcto, sino que es necesario definirlo. (Blog, 2018).

- Gestionar base de datos.
- Autenticación.
- Gestionar usuarios.
- Gestionar análisis de riesgos.
- Gestionar plan de contingencia.
- Generar reportes.
- Módulo administrador.

Requisitos no Funcionales: describen otras prestaciones, características y limitaciones que debe tener el sistema para alcanzar el éxito. Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso, presupuestos, tiempo de entrega, documentación, seguridad y auditorías internas.

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (rvillarroel16, 2017)

Usabilidad: El sistema será utilizado por especialistas en informática. El software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

Rendimiento: Los tiempos de respuestas del sistema deben ser rápidos, al igual que la velocidad de procesamiento de la información para lograr respuestas rápidas del mismo.

Requerimiento de Ayuda y Documentación: El sistema contará con una ayuda general en la página principal, que guíe al usuario a trabajar en el sistema. Estará disponible en cada una de las interfaces, de manera que los usuarios tengan conocimiento de las funcionalidades del mismo y logren hacer un mejor uso de ellas.

Seguridad; El usuario se autenticará antes de entrar al sistema.

Confiabilidad: La información que se maneje en el sistema estará protegida de acceso no autorizado y divulgación, a partir de los diferentes módulos de los usuarios que empleen el sistema.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra corrupción.

Disponibilidad: La información se encontrará disponible en todo momento para aquellos usuarios autorizados a acceder al sistema.

Interfaz: El software debe presentar una interfaz amigable y sencilla.

Software:

- Sistema operativo Wndows 10
- SQLServer
- NetFramework 4.7.2 o superior
- Acrobar Reader o cualquier otro visor de PDF

Hardware:

- Impresora
- Procesador Pentium IV a 2.0GHz y 2Gb de memoria RAM.

2.2. Etapa de planificación

La etapa de planificación es la etapa inicial de todo el proyecto. Se realiza con el objetivo de lograr una eficiente organización del prototipo inicial del problema y proporcionar así un buen comienzo a una solución eficaz. Con este objetivo y según las ideas del cliente sobre el software se desarrollarán las Historias de Usuario, mediante la cual se obtendrá un punto de partida para el resto de la planificación del proyecto. De igual forma se realizará un estimado de cada una de las entregas del proyecto y del tiempo, que se basan en que la planificación inicial se podría afectar debido a cambios que pudiesen sufrir estos aspectos durante el desarrollo del proyecto.

2.2.1. Definición de equipo

La metodología SCRUM define tres roles que rigen el proceso de desarrollo, ellos son:

Roles	Miembros
Cliente o Dueño del Producto (Product Owner)	Fiscalía Provincial Matanzas
SCRUM Máster o SCRUM Manager	Dr C Josue Segura Montero
Equipo (Team)	Conrado Rodríguez Valdés

Tabla 2.1: Definición de roles. Elaboración Propia.

2.2.2. Historias de usuarios

Las historias de usuarios fueron desarrolladas por el equipo de trabajo durante el proceso de levantamiento de requisitos para la implementación del proyecto. Para la estimación de los datos se tomó los siguientes criterios:

- Prioridades en el Negocio (PN): Se medirá en función al rango de: Alta, Media y Baja, las cuales, serán asignadas por el Product Owner.
- Importancia del Desarrollo (ID): Se asignarán valores con ponderaciones del 1 al 100 entre el Product Owner y los miembros del equipo Scrum, donde:
    - Todos los elementos con importancia >=100 deben estar incluidos en el Sprint 1, por ser considerados de extrema importancia para el proyecto.
    - Todos los elementos de importancia de 99-50 deberán estar incluidos en el Sprint 2, pero eso depende de la velocidad del Sprint.
    - Los elementos con importancias de 49-25 se pueden incluir en el último Sprint, según el avance del equipo ya que son requisitos que no alteran el desarrollo del mismo o funcionalidades del mismo.
- Tiempo Estimado (TS): Se asignará por medio de cartas con ponderaciones del 1 al 20 entre el Product Owner y los miembros del equipo Scrum.

Así mismo las historias de usuario se han dividido por módulos para hacer más fácil la programación de cada una de las tareas concernientes a cada uno de los mismos, las cuales son:

- Módulo de Base de Datos: Es el módulo inicial donde se creará la Base de Datos del sistema.
- Módulo Cliente: Es el módulo que contendrá todas las funcionalidades que van a interactuar con los usuarios del sistema.

- Módulo Acceso al Sistema: Es parte esencial del sistema, el cual, consistirá en registrar o autenticar a los usuarios y permitirá el acceso a sus funcionalidades autorizadas acorde a su rol asignado.
- Módulo Administrador: Es el módulo que contendrá todas las funcionalidades que van a ser utilizadas por el administrador del sistema.

Historia de usuario	
ID: HU01	**Usuario:** Administrador del sistema
Nombre Historia: Gestionar Base de Datos	
Prioridad en el Negocio: Alta	**Importancia del Desarrollo:** 100
Tiempo Estimado: 12	**Modulo Asignado:** Base de Datos
Descripción: Se creará el esquema de base de datos para la carga de información teniendo en cuenta las relaciones existentes entre las tablas además de validar la carga de la información y la recuperación de la misma para las transacciones que se realizaran entre la misma y el sistema web.	
Observaciones: Las tablas deben contener toda la data y nomenclatura que manejan en la empresa.	

Tabla 2.2: Historia de Usuario HU01. Elaboración Propia.

Historia de usuario	
Id: hu02	**Usuario:** clientes
Nombre historia: autenticación	
Prioridad en el negocio: alta	**Importancia del desarrollo:** 99
Tiempo estimado: 8	**Modulo asignado:** acceso al sistema
Descripción: para el login se usará un usuario y una contraseña registrada en la base De datos del sistema, para poder tener acceso.	
Observaciones: la interfaz del login será de forma intuitiva.	

Tabla 2.3: Historia de Usuario HU02. Elaboración Propia.

Historia de usuario	
Id: hu03	**Usuario:** administrador del sistema
Nombre historia: gestionar usuarios	
Prioridad en el negocio: media	**Importancia del desarrollo:** 85
Tiempo estimado: 7	**Modulo asignado:** administrador
Descripción: • El usuario podrá crear un nuevo cliente con toda la información requerida como nombre, usuario, rol y contraseña • Se podrá editar un cliente y actualizar su información ya existente. • El usuario no se podrá eliminar, solo deshabilitar. • Los usuarios con rol de digitador no se le podra asignar rol de validación y viseversa.	
Observaciones: los clientes solo deben ser registrados una vez.	

Tabla 2.4: Historia de Usuario HU03. Elaboración Propia.

Historia de usuario	
Id: hu04	**Usuario:** clientes
Nombre historia: gestionar análisis de riesgos	
Prioridad en el negocio: alta	**Importancia del desarrollo:** 98
Tiempo estimado: 20	**Modulo asignado:** cliente
Descripción: • El usuario podrá crear un nuevo plan de análisis de riesgos. • Se podrá editar, eliminar y validar planes ya existentes.	
Observaciones: los planes ya validados no se podran eliminar.	

Tabla 2.5: Historia de Usuario HU04. Elaboración Propia.

HISTORIA DE USUARIO	
ID: HU05	**Usuario:** Clientes
Nombre Historia: Gestionar Plan de Contingencia	
Prioridad en el Negocio: Alta	**Importancia del Desarrollo:** 95
Tiempo Estimado: 18	**Modulo Asignado:** Cliente
Descripción: • El usuario podrá crear un nuevo Plan de Contingencia. • Se podrá editar, eliminar y validar planes ya existente.	
Observaciones: Los planes ya validados no se podrán eliminar.	

Tabla 2.6: Historia de Usuario HU05. Elaboración Propia.

Historia de usuario	
Id: hu06	**Usuario:** clientes
Nombre historia: generar reportes	
Prioridad en el negocio: media	**Importancia del desarrollo:** 75
Tiempo estimado: 12	**Modulo asignado:** cliente
Descripción: el usuario podrá: • Generar el plan de contingencia. • Generar el plan de análisis de riesgos. • Listar todos los planes de contingencia y de análisis de riesgos realizados.	

Tabla 2.7: Historia de Usuario HU06. Elaboración Propia.

Historia de usuario	
Id: hu07	**Usuario:** administrador
Nombre historia: módulo administrador	
Prioridad en el negocio: alta	**Importancia del desarrollo:** 90
Tiempo estimado: 15	**Modulo asignado:** administrador
Descripción: • El módulo administrador permitirá cambiar los establecidos por la metodología. • Realizar la salva de base de datos. • Chequear las trazas de seguridad	
Observaciones: deberá tener todas las opciones para facilitar la interacción.	

Tabla 2.8: Historia de Usuario HU07. Elaboración Propia.

2.2.3. Pila del producto (product backlog)

Es el instrumento metodológico del marco de trabajo **Scrum**, que se usa para listar las características (**Features**) o funcionalidades del software a desarrollar, para priorizarlas de acuerdo a las necesidades del área de negocio. Su contenido se desarrolla a partir de las historias de usuario identificadas por el dueño de producto (**Product Owner**).

La pila de producto permitirá tener visualización de las funcionalidades a desarrollar, priorizar las características del software según las necesidades del negocio, dejar registrado el esfuerzo necesario para desarrollar la historia y asignarla a una iteración (**Sprint**). Para ello, se deben seguir las **reglas de administración de la pila de producto**.

Hay que tener en cuenta que la lista de objetivos/requisitos priorizada representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto. El cliente es el responsable de crear y gestionar la lista (con la ayuda del Facilitador y del equipo, quien proporciona el costo estimado de completar cada requisito). Dado que se deben reflejar las expectativas del cliente, esta lista permite involucrarlo en la dirección de los resultados del producto o proyecto.

ID	Módulo	Historia	Prioridad	Importancia	Tiempo estimado (días)
1	Base de Datos	Gestionar Base de Datos	Alta	100	12
2	Acceso al Sistema	Autenticación	Alta	99	8
3	Cliente	Gestionar Análisis de Riesgos	Alta	98	20
4	Cliente	Gestionar Plan de Contingencia	Alta	95	18
5	Administrador	Módulo Administrador	Alta	90	15
6	Administrador	Gestionar usuarios	Media	85	7
7	Cliente	Generar Reportes	Media	75	11

Tabla 2.9: Pila del Producto. Elaboración Propia.

2.2.4. Definición de los sprints

Se define la velocidad de desarrollo de cada Sprint según la importancia de las historias de usuario y el tiempo de trabajo del equipo de Scrum para el proyecto y la dedicación que se le dará al mismo.

El tiempo del equipo de trabajo está dado dentro de las jornadas laborales de 8 horas a la semana de lunes a viernes 8 horas durante 6 meses, de los cuales, se obtiene como resultado la cantidad de días de trabajo dedicados al proyecto por cada Sprint.

Equipo Scrum	Jornada Laborar (h)	Horas de trabajo al proyecto por día (h)	Horas de trabajo al proyecto por semana (h)	Semanas de Trabajo por mes (u)	Total de horas (h)	Total de días laborables para el proyecto (días)
Conrado Rodríguez	8	8	40	4	160	20

Tabla 2.10: Tiempo para cada Sprint. Elaboración Propia.

Debido al tiempo de dedicación que se le dará al proyecto y las horas asignadas dentro de horario de trabajo se esperan tener algunas distracciones e impedimentos pero que están dentro de las estimaciones para el proyecto, por lo cual, el Product Owner da un factor de dedicación del 90 % del

tiempo comprendido para el mismo. Según lo indicado se procederá a calcular la velocidad estimada para el desarrollo de los Sprints, la cual es: 18 días.

De acuerdo a la velocidad obtenida para la ejecución de cada Sprint y al tomar en cuenta el nivel de importancia definido por cada historia de usuario, se procede a agrupar las mismas y determinar la cantidad de Sprints para el proyecto, en donde se obtiene:

Módulo	**Historia**	**Prioridad**	**Importancia**	**Tiempo estimado (días)**
Base de Datos	Gestionar Base de Datos	Alta	100	12
Acceso al Sistema	Autenticación	Alta	99	8
Total de días del Sprint		20 días		

Tabla 2.11: Estimación del Sprint N° 1. Elaboración Propia.

Módulo	**Historia**	**Prioridad**	**Importancia**	**Tiempo estimado (días)**
Cliente	Gestionar Análisis de Riesgos	Alta	98	20
Total de días del Sprint		20 días		

Tabla 2.12: Estimación del Sprint N° 2. Elaboración Propia.

Módulo	**Historia**	**Prioridad**	**Importancia**	**Tiempo estimado (días)**
Cliente	Gestionar Plan de Contingencia	Alta	95	18
Total de días del Sprint		18 días		

Tabla 2.13: Estimación del Sprint N° 3. Elaboración Propia.

Módulo	**Historia**	**Prioridad**	**Importancia**	**Tiempo estimado (días)**
Administrador	Módulo Administrador	Alta	90	15
Administrador	Gestionar usuarios	Media	85	7
Total de días del Sprint		22 días		

Tabla 2.14: Estimación del Sprint N° 4. Elaboración Propia.

Módulo	**Historia**	**Prioridad**	**Importancia**	**Tiempo estimado (días)**
Cliente	Generar Reportes	Media	75	11
Total de días del Sprint		11 días		

Tabla 2.15: Estimación del Sprint N° 5. Elaboración Propia.

De acuerdo a la velocidad estimada de por cada Sprint el desarrollo de la aplicación se ejecutará en 5 Sprint, los mismos que han sido organizados por la importancia de cada una de las historias de usuario y por el tiempo de duración de cada una de las mismas.

2.2.5. Planificación de los sprints

Para el desarrollo de cada Sprint se han planificado revisiones y entregas para validar los avances obtenidos del desarrollo programado y así generar de manera retrospectiva las acciones de mejora para los siguientes desarrollos.

Por cada desarrollo de Sprint se mostrarán los avances a través del TaskBoard, donde se apreciaran las actividades en desarrollo, pendientes y finalizadas por cada historia de usuarios; además de mostrar el Burndown para ver la velocidad de desarrolla en la que se está dando el proyecto y determinar cuáles son las historias o actividades que están demandando mucho tiempo al desarrollo del proyecto o si las historias de usuario tiene pocas actividades de desarrollo y se están perdiendo recursos en ello.

Para validar la funcionalidad o conformidad de la elaboración de cada historia de usuario se realizarán pruebas de funcionalidad por cada historia de usuario y ver los aciertos y desaciertos de los mismo, los cuales, se verán reflejados en el informe de cierre del Sprint. Se procede a detallar la planificación de cada Sprint, indicando las fechas de revisión e historias de usuario comprendidas.

Sprint No.1	
Fecha inicio	23-12-2019
Fecha fin	22-1-2020
Revisión de los avances	Las revisiones se realizarán semanalmente. Las fechas de revisión serán las siguientes: 27-12-2020 3-1-2020 10-1-2020

	17-1-2020 22-1-2020
Tareas a desarrollar	Gestionar base de datos Autenticación

Tabla 2.16: Planificación de entregas Sprint 1. Elaboración Propia.

Sprint No.2	
Fecha inicio	23-1-2020
Fecha fin	19-2-2020
Revisión de los avances	Las revisiones se realizarán semanalmente. Las fechas de revisión serán las siguientes: 31-1-2020 7-2-2020 14-2-2020 19-2-2020
Tareas a desarrollar	Gestionar análisis de riesgos

Tabla 2.17: Planificación de entregas Sprint 2. Elaboración Propia.

Sprint No.3	
Fecha inicio	20-2-2020
Fecha fin	16-3-2020
Revisión de los avances	Las revisiones se realizarán semanalmente. Las fechas de revisión serán las siguientes: 28-2-2020 6-3-2020 16-3-2020

Tareas a desarrollar	Gestionar plan de contingencia

Tabla 2.18: Planificación de entregas Sprint 3. Elaboración Propia.

Sprint No.4	
Fecha inicio	17-3-2020
Fecha fin	15-4-2020
Revisión de los avances	Las revisiones se realizarán semanalmente. Las fechas de revisión serán las siguientes: 20-3-2020 27-3-2020 6-4-2020 10-4-2020 15-4-2020
Tareas a desarrollar	Módulo administrador Gestionar usuarios

Tabla 2.19: Planificación de entregas Sprint 4. Elaboración Propia.

Sprint No.5	
Fecha inicio	16-4-2020
Fecha fin	30-4-2020
Revisión de los avances	Las revisiones se realizarán semanalmente. Las fechas de revisión serán las siguientes: 24-4-2020 30-4-2020
Tareas a desarrollar	Generar reportes

Tabla 2.20: Planificación de entregas Sprint 5. Elaboración Propia.

2.3. Estudio de factibilidad

Una de las cuestiones a tener en cuenta en el momento de desarrollar un software, es validar los beneficios y ventajas de éste con respecto a su costo, por lo que es necesaria una estimación del costo del software, además de un análisis de los beneficios tangibles e intangibles que reportará el proyecto.

2.3.1. Estimación de costos

El **análisis de costos** se define, en economía, como la medida de la relación costo-producción. Es decir, los economistas se preocupan por determinar el costo en el que se incurre al contratar los insumos, y qué tan bien se pueden reorganizar para aumentar la productividad de la empresa.

En otras palabras, el análisis de costos se refiere a la determinación del valor monetario de los insumos (mano de obra, materia prima), denominado como el costo general de producción, que ayuda a decidir el nivel óptimo de producción.

Por tanto, el análisis de costos es fundamental en la toma de decisiones de negocios, ya que debe entenderse cuidadosamente el costo incurrido en la entrada y salida de producción, antes de planificar la capacidad de producción de la empresa.

A menudo se le denomina análisis de costo-beneficio o análisis de costo-efectividad. Un análisis de costos requiere de habilidades específicas para llevarlo a cabo, y es una herramienta útil para varios aspectos de la planificación empresarial. (Corvo, 2019)

¿Qué es el método de puntos de función?

Es una técnica de estimación de software desarrollada originalmente por Allan Albrecht en 1979 mientras trabajaba para IBM, quien definió conceptos para medir el software a partir de valoraciones de funcionalidades entregadas al usuario y no a partir de aspectos técnicos, con la intención de producir valoraciones independientes de la tecnología y fases del ciclo de vida utilizado.

El trabajo de Albrecht fue continuado por el grupo internacional de usuarios de puntos de función, quienes plasmaron sus conceptos en el método IFPUG-FPA.

IFPUG-FPA realiza las valoraciones a partir de la funcionalidad del sistema, primero clasificándolas, luego asignando una complejidad y ponderación a cada una según unas tablas predefinidas, determinando así el valor de puntos de función.

Sumando los puntos de todas las funcionalidades se obtiene la valoración de todo el proyecto y finalmente se puede aplicar un factor de ajuste, que puede depender de características generales del sistema como por ejemplo requerimientos no funcionales como el rendimiento, reusabilidad, facilidad de instalación y operación entre otros aspectos.

Los puntos de función permiten traducir el tamaño de funcionalidades de software a un número, a través de la suma ponderadas de las características que este tiene.

Una vez que tenemos los puntos de función, podemos traducirlos en horas hombre o días de trabajo, según factor de conversión que dependería de mediciones históricas de nuestra productividad. Con las horas hombre, podemos determinar el costo y presupuesto de los proyectos. (PMOinformatica.com, 2015)

Terminologías

- Entrada Externa (EI).
- Salida Externa (EO).
- Consulta Externa (EQ).
- Archivo Lógico Interno (ILF).
- Archivo de Interfaz Externo (EIF).
- Puntos de Función Sin Ajustar (PFSA).
- Puntos de Función Ajustado (PFA).
- Factor de complejidad Técnica (FCT).
- Esfuerzo (E).
- Costo Tota (CT).

Tipo/Complejidad	**Baja**	**Media**	**Alta**
Entrada Externa (EI)	3	4	6
Salida Externa (EO)	4	5	7
Consulta Externa (EQ)	3	4	6
Archivo Lógico Interno (ILF)	7	10	15
Archivo de Interfaz Externo (EIF)	5	7	10

Tabla 2.21: Tabla de Valores.

Tipo/Complejidad	**Baja**	**Media**	**Alta**	**Total**
Entrada Externa (EI)	3*9=27	4*1=4	6*2=12	43
Salida Externa (EO)	4*7=28	-	7*4=28	56
Consulta Externa (EQ)	3*9=27	4*1=4	6*2=12	43
Archivo Lógico Interno (ILF)	-	-	15*1=15	15
Archivo de Interfaz Externo (EIF)	-	-	-	0
				PFSA=157

Tabla 2.22: Tabla de Valores Completa. Elaboración Propia.

Luego de obtener los puntos de función sin ajustar, debemos calificar cada uno de los factores de valor de ajuste, utilizando una escala del 0 al 5:

No.	**Factores de valor de ajuste**	**Valor**
1	Comunicación de datos	0
2	Proceso distribuido de datos	0
3	Desempeño	1
4	Configuración	1
5	Volumen de transacciones	1
6	Captura de datos en línea	0
7	Eficiencia del usuario final	4
8	Actualización de datos en línea	0
9	Complejidad	3
10	Reusabilidad	2
11	Facilidad de instalación	1

12	Facilidad de operación	0
13	Instalación múltiple	0
14	Facilidad de cambio	3
	FCT	**16**

Tabla 2.23: Tabla de Factores de valor de ajuste. Elaboración Propia.

Calculo de Puntos de Función Ajustados

$$\mathbf{PFA} = \mathbf{PFSA} * [\mathbf{0.65} + (\mathbf{0.01} * \mathbf{FCT})]$$

$$\mathbf{PFA} = \mathbf{157} * [\mathbf{0.65} + (\mathbf{0.01} * \mathbf{16})]$$

$$\mathbf{PFA} = \mathbf{157} * [\mathbf{0.65} + \mathbf{0.16}]$$

$$\mathbf{PFA} = \mathbf{157} * \mathbf{0.81}$$

$$\mathbf{PFA} = \mathbf{127.17}$$

Calculo Esfuerzo [hora/persona]

$$\mathbf{E} = \frac{\mathbf{PFA}}{\frac{\mathbf{1}}{\mathbf{8}}\mathbf{persona/hora}}$$

$$\mathbf{E} = \frac{\mathbf{127.17}}{\frac{\mathbf{1}}{\mathbf{8}}\mathbf{persona/hora}}$$

$$\mathbf{E} = \mathbf{1017.36\ horas/persona}$$

Tomando 22 días laborables en el mes y 8 horas productivas al día, obtenemos 174 horas laborables al mes.

Duración de proyecto en horas = 1017.36 horas/persona × 1 persona = 1017.36 horas

Duración en meses□= 1017.36 horas / (174 horas/mes) = **5.85 meses**

Cálculo del Presupuesto del Proyecto

Suponiendo un sueldo de 600.00 MN

Costo Total = sueldo de 1 participante * cantidad de participantes *Tiempo de desarrollo

Costo Total = 600 * 1 * 5.85

Costo Total = 3508.1 MN

2.3.2. Beneficios tangibles e intangibles

El sistema permite la automatización del análisis de riesgos. La cual permite la posterior confección del plan de seguridad informática. Generando un plan de contingencia que permite la toma de decisiones por parte del personal encargado de dicho proceso

2.3.3. Análisis de costos y beneficios.

Con estos beneficios y la estimación de costos realizados anteriormente, se evidencia que la solución propuesta presenta una buena relación entre costo y beneficios. Constatándose la oportunidad que representa para la entidad contar con esta valiosa herramienta.

2.4. Diseño de base de datos

Uno de los pasos cruciales en la construcción de un software que maneje una base de datos, es sin duda, el diseño de la misma pues tiene como propósito asegurar que los datos persistentes sean almacenados consistente y eficientemente. En el diseño de la base de datos los modelos de datos tienen un papel significativo pues se encargan de proveer una vista de las entidades lógicas de datos y sus relaciones, si los modelos no son definidos apropiadamente, podemos tener muchos problemas al momento de ejecutar consultas a la base de datos. Ver anexo 1

2.5. Conclusiones parciales

El levantamiento de los requisitos funcionales y su descripción es fundamental para lograr un diseño apropiado para la realización del proyecto. Se elaboraron los elementos descriptivos y artefactos fundamentales correspondientes a las fases de planificación, diseño e implementación que propone la metodología de desarrollo SCRUM. Se pudo comprobar la organización y rapidez que ofrece el trabajo con la metodología de desarrollo SCRUM. Se pudo determinar los beneficios tangibles e intangibles así, como la estimación de los costos del sistema informático.

3. Capitulo III: validación de la solución propuesta

3.1. Introducción

En este capítulo se muestran las pruebas realizadas permitiendo comprobar cada una de las funcionalidades añadidas y detectando los posibles errores no detectados en el momento de desarrollo. En el capítulo también se muestran el análisis de los resultados obtenidos.

3.2. Descripción de la solución

Se propone el desarrollo de una aplicación de escritorio que les permita a los miembros designados por la dirección de la fiscalía al análisis de riesgos informáticos mejorar la gestión y almacenado de la información de los procesos que realizan. De esta manera facilitará el acceso a la información, la seguridad, la rapidez y eficiencia de los datos. Permitirá tener un historial, de fácil acceso, de los planes elaborados y aprobados.

Para garantizar la seguridad y confiabilidad en la información que se procesa, es de gran importancia la aplicación cuenta con un sistema de autentificación para identificar los usuarios que utilizan el programa. Además, cuenta un sistema de auditoria permitiendo tener un registro de todos los procesos que se ejecuten en dicho software.

Existe un usuario único administrador del sistema que es quien crea el resto de los usuarios, asigna módulos, importa y exporta la Base de Datos y además puede cambiar los valores de los parámetros establecidos por la metodología MAGERIT. La figura1 muestra la interfaz gráfica de esta pantalla del sistema.

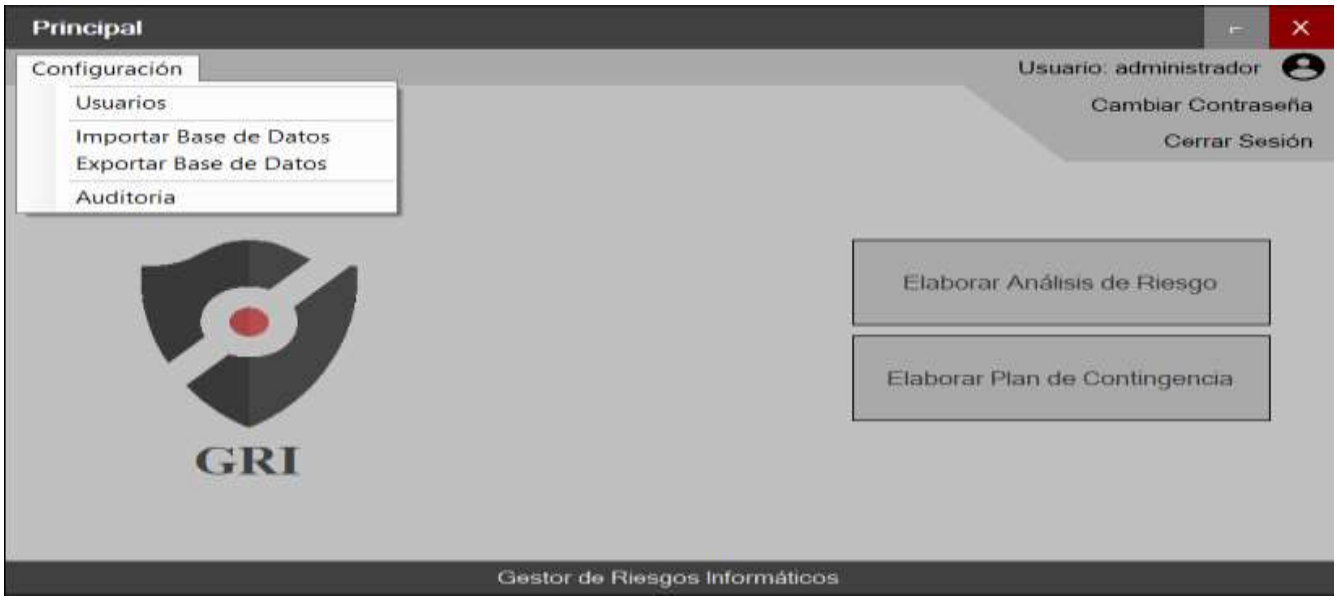

Ilustración 1: Página principal administrador del sistema. Elaboración Propia

Los usuarios que interactúan con la aplicación se le podrán asignar los siguientes módulos:

- Digitador de Plan de análisis de Riesgos: es quien puede introducir todo lo relacionado con el análisis de riesgos y además quien lo genera. El usuario que tenga este módulo no se le puede asignar el de aprobar
- Digitador Plan de Contingencia: es quien puede introducir todo lo relacionado con el Plan de Contingencia y además quien lo genera. El usuario que tenga este módulo no se le puede asignar el de aprobar. En La ilustración 2 se puede observar la interfaz gráfica del usuario digitador de planes de análisis de riesgos y contingencia.

Ilustración 2: Página principal digitador del sistema

- Visor de Reportes de Plan de Análisis de Riesgos: puede revisar los planes de análisis de
- riesgos generados.
- Visor de Reportes de Plan de Contingencia: puede revisar los planes de Contingencias generados.
- Aprobar Plan de Análisis de Riesgos: puede revisar los planes de análisis de riesgos generados y además aprobarlos. El usuario que tenga este módulo no se le puede asignar el de digitador
- Aprobar Plan de Contingencia: puede revisar los planes de contingencias generados y además aprobarlos. El usuario que tenga este módulo no se le puede asignar el de digitador. En La ilustración 3 se puede observar la interfaz gráfica del usuario que aprueba los planes de análisis de riesgos y contingencia.

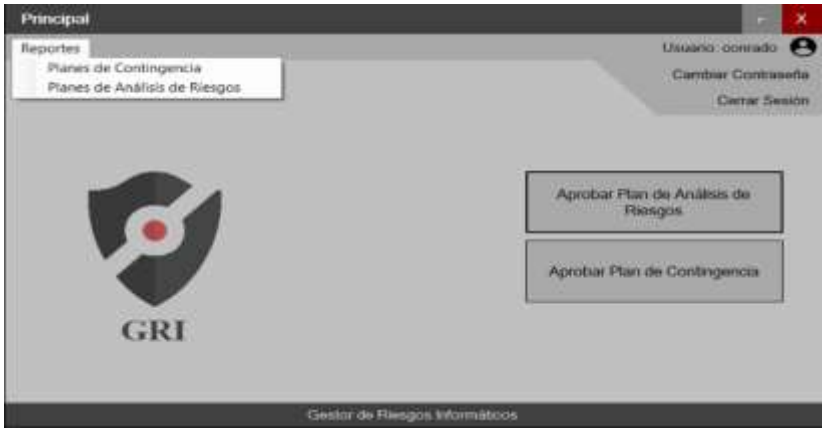

Ilustración 3: Página principal validador del sistema. Elaboración Propia.

3.3. Algoritmos

Se presentan fragmentos de algunos de los algoritmos principales usados en el desarrollo del sistema.

- En este algoritmo se encarga de validar la autenticación y el registro de usuario.

```
static public void AutenticarUsuario(string usuario, string contraseña, int total, fLogin fLogin)
{
    if (usuarioNuevoLogin != usuario)
    {
        total = 0;
        fLogin.TotalLoginFallidos = 0;
        usuarioNuevoLogin = usuario;
    }
    UsuarioAutenticado = Context.Instance.Usuarios.Include("UsuarioModulos")
        .Where(x => x.Usuario == usuario).FirstOrDefault();

    if (UsuarioAutenticado == null)
    {
        throw new Exception("El usuario o la contraseña son incorrectos.");
    }
    if (UsuarioAutenticado.Delete != null)
    {
        throw new Exception("El usuario está deshabilitado.");
    }
    if (UsuarioAutenticado.Contraseña != Encriptar.EncriptarTexto(contraseña))
    {
        if(total<2)
            throw new Exception("El usuario o la contraseña son incorrectos.");
        else
        {
            Mensajes.Informacion("El usuario o la contraseña son incorrectos.");
            fLogin.TotalLoginFallidos = 0;
            UsuarioAutenticado.Delete = DateTime.Now;
            ControladorAuditoria.Log("El usuario ha sido deshabilitado.");
            Context.Instance.SaveChanges();
            throw new Exception("El usuario ha sido deshabilitado.");
        }
    }
    ControladorAuditoria.Log("Se ha autenticado correctamente.");
    Context.Instance.SaveChanges();
}
```

Ilustración 4: Algoritmo1

```
public static void ValidarUsuario(string nombreApellidos, string usuario,
    string contraseña, string confirmarContraseña, int id)
{
    if (nombreApellidos == string.Empty | usuario == string.Empty | contraseña == string.Empty
        | confirmarContraseña == string.Empty)
    {
        throw new Exception("No pueden haber campos vacios");
    }
    Validaciones.SoloTexto(nombreApellidos, "Nombre y Apellidos");
    if (id == -1)
        ValidarUsuarioSinRegistrar(nombreApellidos, usuario, contraseña, confirmarContraseña);
    else
        ValidarUsuarioRegistrado(usuario, contraseña, confirmarContraseña, id);
}
1 referencia
public static void ValidarUsuario(string nombreApellidos, string usuario, int id)
{
    if (nombreApellidos == string.Empty | usuario == string.Empty )
    {
        throw new Exception("No pueden haber campos vacios");
    }
    Validaciones.SoloTexto(nombreApellidos, "Nombre y Apellidos");
    if (id == -1)
        ValidarUsuarioSinRegistrar(nombreApellidos, usuario);
    else
        ValidarUsuarioRegistrado(usuario, id);
}
```

Ilustración 5: Algoritmo2

```
public static void ValidarUsuarioSinRegistrar(string nombreApellidos, string usuario,
string contraseña, string confirmarContraseña)
{
    if (Context.Instance.Usuarios.FirstOrDefault(x => x.NombreApellidos == nombreApellidos) != null)
    {
        throw new Exception("Esa persona ya esta registrada");
    }
    if (Context.Instance.Usuarios.FirstOrDefault(x => x.Usuario == usuario) != null)
    {
        throw new Exception("Esa usuario ya existe");
    }
    if (contraseña != confirmarContraseña)
    {
        throw new Exception("Las contraseñas no coinciden");
    }
    Validaciones.Contraseña(contraseña);
}
1 referencia
public static void ValidarUsuarioRegistrado(string usuario,
    string contraseña, string confirmarContraseña, int id)
{
    Usuarios Usuario = Context.Instance.Usuarios.FirstOrDefault(x => x.Id==id);
    if (Usuario != null & Usuario.Id != id)
    {
        throw new Exception("Esa persona ya esta registrada");
    }

    Usuario = Context.Instance.Usuarios.FirstOrDefault(x => x.Usuario == usuario);
    if (Usuario != null && Usuario.Id != id)
    {
        throw new Exception("Esa usuario ya existe");
    }
    if (contraseña != confirmarContraseña)
    {
        throw new Exception("Las contraseñas no coinciden");
    }
    Validaciones.Contraseña(contraseña);
}
```

Ilustración 6: Algoritmo3

```
public static void ValidarUsuarioSinRegistrar(string nombreApellidos, string usuario)
{
    if (Context.Instance.Usuarios.FirstOrDefault(x => x.NombreApellidos == nombreApellidos) != null)
    {
        throw new Exception("Esa persona ya esta registrada");
    }
    if (Context.Instance.Usuarios.FirstOrDefault(x => x.Usuario == usuario) != null)
    {
        throw new Exception("Esa usuario ya existe");
    }
}

1 referencia
public static void ValidarUsuarioRegistrado(string usuario, int id)
{
    Usuarios Usuario = Context.Instance.Usuarios.FirstOrDefault(x => x.Id == id);
    if (Usuario != null & Usuario.Id != id)
    {
        throw new Exception("Esa persona ya esta registrada");
    }

    Usuario = Context.Instance.Usuarios.FirstOrDefault(x => x.Usuario == usuario);
    if (Usuario != null && Usuario.Id != id)
    {
        throw new Exception("Esa usuario ya existe");
    }
}
```

Ilustración 7: Algoritmo4

3.4. Pruebas

La etapa de pruebas del software es la encargada de descubrir si la aplicación desarrollada funciona como fue prevista. Los tipos de pruebas realizadas sobre la aplicación fueron bajo los principios de las pruebas de caja blanca y pruebas de caja negra que a continuación se explican.

3.4.1. ¿Qué son las pruebas de software?

Las pruebas, vistas desde el marco de un proceso de desarrollo de software, son los diferentes procesos que se deben realizar durante un desarrollo, con el objetivo de asegurar que este completo, correcto, tenga calidad, entre otros factores de gran importancia.

Consisten en llevar a cabo la verificación dinámica de un componente, programa o sistema, mediante el uso de métodos, técnicas y herramientas especializadas, las cuales permiten detectar y corregir errores, problemas e inconsistencias durante el proceso de desarrollo.

Estas, al contrario de lo que muchas personas creen, no se deben dejar para el final de la etapa de construcción del software. Las pruebas se deben empezar a realizar desde la misma etapa de análisis de los requerimientos, ya que desde un principio se puede caer en malas interpretaciones de las "reglas del negocio", lo que finalmente tendrá como consecuencia incongruencia entre lo que el cliente quiere y lo que se ha desarrollado. (D-bag_Christoph, 2008)

3.4.2. Tipos de pruebas de software

Todos los tipos de pruebas de software que existen, básicamente, se pueden agrupar en dos grupos: las pruebas funcionales y las pruebas no funcionales.

Sin embargo, seguramente has escuchado hablar de más tipos de pruebas, por ejemplo, pruebas unitarias, pruebas de integración o pruebas de aceptación, pero estos tipos se pueden agrupar dentro de los dos grupos anteriores.

Dentro de las pruebas funcionales tenemos:

- Pruebas unitarias.
- Pruebas de aceptación.
- Pruebas de integración.
- Pruebas de regresión.

Las pruebas no funcionales son:

- Pruebas de carga.
- Pruebas de estrés.
- Pruebas de escalabilidad.
- Pruebas de portabilidad.

Tal vez hayas oído hablar de las pruebas de caja blanca y las pruebas de caja negra, pero lo que ocurre con las mismas es que no son tipos de pruebas, sino técnicas de pruebas de software. (Jiménez, 2019)

3.5. Pruebas de aceptación

Las pruebas de aceptación son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas. Estas pruebas generalmente son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción.

Las pruebas de aceptación son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software.

Estas pruebas se realizan una vez que ya se ha probado que cada módulo funciona bien por separado, que el software realice las funciones esperadas y que todos los módulos se integran correctamente. (Training, 2017).

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

- **Número**: servirá como identificador de la prueba realizada.
- **EPP**: tendrá el nombre del elemento de la pila de producto al que hace referencia la prueba a realizar.
- **Nombre**: nombre que se le da a la prueba a realizar.
- **Descripción**: se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución**: mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- **Entradas**: se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado esperado**: se hará una breve descripción del resultado que se espera obtener con la prueba realizada.
- **Evaluación de la prueba**: acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los tres resultados que a continuación se describen:
    - **Satisfactoria**: cuando el resultado de la prueba es exactamente el esperado por el usuario.
    - **Parcialmente satisfactoria**: cuando el resultado no es completamente el esperado por el cliente o usuario de la aplicación y muestra resultados erróneos o fuera de contexto.
    - **No satisfactoria**: cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida el EPP.

Se ha tomado una muestra al azar de las pruebas de aceptación, el resto se encuentran en el documento del proyecto.

PRUEBAS DE ACEPTACIÓN	
Número: 1	**No EPP: 1**
Nombre Caso de Prueba: Test de integridad a la base de datos	
Descripción: Verificar que la inserción, edición, eliminación de datos y las relaciones entre los mismo sean correctas	
Condiciones de ejecución: Servicio de SQL Server ejecutándose.	
Entradas: Datos de Pruebas	
Resultado esperado: Todos los conjuntos de datos probados han sido correctamente procesados	
Evaluación: Prueba satisfactoria	

Tabla 3.1: Prueba se Aceptación 1. Elaboración Propia.

PRUEBAS DE ACEPTACIÓN	
Número: 2	**No EPP: 2**
Nombre Caso de Prueba: Test de autenticación de usuario	
Descripción: Autenticar un usuario	
Condiciones de ejecución: El usuario tiene que estar agregado en la base de datos y activo	
Entradas:	

Nombre de usuario y contraseña
Resultado esperado: El usuario es autenticado y según el rol que desempeña en la aplicación, se mostrará la vista de inicio con sus accesos correspondientes
Evaluación: Prueba satisfactoria

Tabla 3.2: Prueba se Aceptación 2. Elaboración Propia.

PRUEBAS DE ACEPTACIÓN	
Número: 2	**No EPP: 3**
Nombre Caso de Prueba: Test a agregar un usuario.	
Descripción: Verificar que el usuario se agregue correctamente	
Condiciones de ejecución: El administrador esté autenticado.	
Entradas: Nombre de usuario existente.	
Resultado esperado: Mensaje: Usuario registrado satisfactoriamente	
Evaluación: Prueba No Satisfactoria	

Tabla 3.3: Prueba se Aceptación 3. Elaboración Propia.

3.6. Análisis de los resultados obtenidos en las pruebas

Al concluir el desarrollo del proceso de pruebas se lograron resultados satisfactorios en el progreso del sistema. Las pruebas fueron desplegadas por cada una de las historias de usuario. Los elementos de

pruebas abordados, permitieron erradicar las incongruencias respecto al funcionamiento de la aplicación, obteniendo resultados satisfactorios de dichas pruebas.

3.7. Conclusiones parciales

El proceso de pruebas es de vital importancia, permitiendo encontrar errores existentes y dar fin al proceso de desarrollo. Las pruebas de aceptación permiten al usuario familiarizarse con el software y sugerir cambios no pensados anteriormente. El cliente y el equipo de desarrollo confirman en conjunto la culminación del proyecto. Las pruebas se realizaron con los expertos de la Oficina de Seguridad para las Redes Informáticas (OSRI), manifestaron validez de la solución propuesta

Conclusiones

Como resultado de esta investigación se cumplieron los objetivos trazados arribando a las siguientes conclusiones:

- El estudio realizado sobre los antecedentes, el estado actual del análisis de riesgos, la consulta de la literatura nacional como internacional, así, como documentos relacionados con el objeto de estudio, permitió contar con los elementos necesarios para dar solución a la problemática planteada.
- Los sistemas automatizados encontrados, vinculados al tema no le dan solución al problema planteado por lo que no es factible su utilización.
- Se utilizaron las herramientas de software más factibles para la construcción de la solución adecuándolas a las características del sistema informático de la entidad.
- Se realizó la estimación de costo de implementación del sistema y el estudio de factibilidad, arrojando como resultado la factibilidad de la realización del sistema informático.
- La realización de las pruebas funcionales a este sistema informático permitió detectar errores en el sistema y la rápida corrección de los mismos.

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:

- Ampliar las funcionalidades del sistema en aras de generar el Plan de Seguridad Informática.
- Generar reportes en forma de gráficas para una mejor comprensión de los mismos.

1. Continuar perfeccionando el sistema con e objetivo de añadirle nuevas funcionalidades que puedan resultar de interés tanto al cliente.
2. Desplegar el sistema en otras entidades.

Bibliografía

Abellán, E. (2020, 03 05). *Metodología Scrum: qué es y cómo funciona*. Retrieved from https://www.wearemarketing.com: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html

Araque, M. (2017, 2 8). *Metodología Scrum: qué es y cómo funciona*. Retrieved from wearemarketing: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html

B., G. (2019, 5 13). *¿Qué es MySQL? Explicación detallada para principiantes*. Retrieved from hostinger: https://www.hostinger.es/tutoriales/que-es-mysql/

Blog, R. (2018, 04 20). *Requerimientos Funcionales y No Funcionales, ejemplos y tips*. Retrieved from https://medium.com/: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Borges, S. (2019, 11 19). *Servidor PostgreSQL*. Retrieved from blog.infranetworking: https://blog.infranetworking.com/servidor-postgresql/

Calvo, D. (2018, 4 7). *Metodología XP Programación Extrema (Metodología ágil)*. Retrieved from diegocalvo: https://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/

CASTRO BOLAÑOS, D. E., & ROJAS MORA, Á. D. (2013). RIESGOS, AMENAZAS Y VULNERABILIDADES DE LOS SISTEMAS DE INFORMACIÓN GEOGRÁFICA. BOGOTÁ, COLOMBIA.

Corvo, H. S. (2019). *Análisis de costos: cómo se hace, para qué sirve y ejemplos*. Retrieved from www.lifeder.com/analisis-de-costos: https://www.lifeder.com/analisis-de-costos/

D-bag_Christoph. (2008, 11 8). *Pruebas+de+Software*. Retrieved from http://clases3gingsof.wikifoundry.com: http://clases3gingsof.wikifoundry.com/page/Pruebas+de+Software

EALDE. (2019, 8 27). *DIRECCIÓN DE PROYECTOS*. Retrieved from ealde.es: https://www.ealde.es/product-backlog-sprint-backlog/

Equipo de Expertos. Universidad Internacional de Valencia. (2018, 3 21). *Lenguaje de alto nivel, los más utilizados*. Retrieved from Universidad Internacional de Valencia: https://www.universidadviu.com/lenguaje-alto-nivel-los-mas-utilizados/

Escobar, A. R. (2020, Junio 16). *Concepto de Microsoft Visual Studio, ¿Qué es y para qué sirve Microsoft Visual Studio?* Retrieved from www.espaciohonduras.net:

https://www.espaciohonduras.net/microsoft-visual-studio-concepto-y-que-es-y-para-que-sirve-microsoft-visual-studio

Garcerant, I. (2018, 8 23). *¿Qué es un lenguaje de bajo nivel? ¿Cuáles son los que existen?* Retrieved from Quora: https://es.quora.com/Qu%C3%A9-es-un-lenguaje-de-bajo-nivel-Cu%C3%A1les-son-los-que-existen

García, J. M. (2015, 12 11). *Descubre lo último en innovación y tendencias digitales*. Retrieved from arsys: https://www.arsys.es/blog/programacion/que-es-laravel/

GERMANOV, I. (2019, 8 12). *Kanban vs Scrum vs Scrumban: Cuales son las diferencias?* Retrieved from ora: https://ora.pm/es/blog/scrum-vs-kanban-vs-scrumban

Jiménez, J. B. (2019, 6 27). *Tipos de Pruebas de Software*. Retrieved from https://openwebinars.net/: https://openwebinars.net/blog/tipos-de-pruebas-de-software/

Lendínez, L. C. (2019, 3 14). *Kanban. Metodología para aumentar la eficiencia de los procesos*. Retrieved from 3C Tecnología: http://ojs.3ciencias.com/index.php/3c-tecnologia/article/view/766

Marco, B. S. (2020, 1 7). *Visual Studio Code*. Retrieved from mclibre: https://www.mclibre.org/consultar/informatica/lecciones/vsc.html

Mesa, A. R. (2018, 12 18). *Los roles de Scrum*. Retrieved from openwebinars: https://openwebinars.net/blog/roles-scrum/

Orta Cruz., L. D., & Rodríguez Cárdenas, J. (2013). *Herramienta informática para la gestión de riesgos en tecnologías de la información*. Retrieved from monografias: https://www.monografias.com/trabajos99/herramienta-informatica-gestion-riesgos-tecnologias-informacion/herramienta-informatica-gestion-riesgos-tecnologias-informacion.shtml

Parada, M. (2019, 11 23). *Qué es SQL Server*. Retrieved from openwebinars.net: https://openwebinars.net/blog/que-es-sql-server/

Pérez Pérez, M. J. (2012). Guía Comparativa de Metodologías Ágiles . *Guía Comparativa de Metodologías Ágiles* . Valladolid , Valladolid , España.

PMOinformatica.com. (2015, 4 6). *Estimación de proyectos de software por puntos de función*. Retrieved from http://www.pmoinformatica.com: http://www.pmoinformatica.com/2015/04/estimacion-puntos-funcion-introduccion.html

Raffino, M. E. (2020, 2 12). *Métodos de Investigación*. Retrieved from concepto.de: https://concepto.de/metodos-de-investigacion/

Robledano, Á. (2019, 7 22). *Qué es NET Framework*. Retrieved from Open Webinars: https://openwebinars.net/blog/que-es-net-framework/

rvillarroel16. (2017, 1 20). *Requerimientos Funcionales y No Funcionales*. Retrieved from ingenieriadesoftwareutmachala: https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/20/requerimientos-funcionales-y-no-funcionales/

Sierra, C. D. (2007, Noviembre 15). *Visual Paradigm For Uml*. Retrieved from SlideShare: https://es.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml

SoftExpert Riesgo. (n.d.). Retrieved from softexpert: https://www.softexpert.com/es/produto/gestion-riesgos-controles/

Solano, A. A. (2019, 1 1). *Qué es PHP: Características y usos*. Retrieved from openwebinars: https://openwebinars.net/blog/que-es-php/

TAMARIT, R. G. (2019, 3 12). *PRODUCT BACKLOG Y SPRINT BACKLOG*. Retrieved from muyagile.com: https://muyagile.com/product-backlog-y-sprint-backlog/

Training, L. A. (2017, 8 23). *que-son-las-pruebas-de-aceptacion/*. Retrieved from https://losandestraining.com: https://losandestraining.com/2017/08/23/que-son-las-pruebas-de-aceptacion/

Voutssas M., J. (2016). Preservación documental digital y seguridad informótica. *Publicando*, 127-155.

Anexos

Anexo 1. Tabla Comparativa Metodologías Ágiles

Metodologías ágiles					
		Orientada al desarrollo de software	Orientada a la gestión de proyectos		
		Xp	Scrum	Kanban	Scrumban
¿Por qué utilizar un método ágil?	Respeto de las fechas de entrega	Falso	Verdadero	Falso	Falso
	Cumplimiento de los requisitos	Verdadero	Verdadero	Verdadero	Verdadero
	Respeto al nivel de calidad	Falso	Falso	Falso	Falso
	Satisfacción del usuario final	Falso	Verdadero	Falso	Falso
	Entornos turbulentos	Verdadero	Verdadero	Verdadero	Verdadero
	Favorable al off shoring	Falso	Verdadero	Falso	Verdadero
	Aumento de la productividad	Verdadero	Verdadero	Verdadero	Verdadero
¿Cuál es la parte de agilidad incluida en el método?	Iteraciones cortas	Verdadero	Verdadero	Verdadero	Verdadero
	Colaboración	Verdadero	Verdadero	Verdadero	Verdadero
	Centrado en las personas	Verdadero	Verdadero	Verdadero	Verdadero
	Refactoring político	Verdadero	Falso	Falso	Falso

.

Anexo 2. Diagrama de Diseño de Base de Datos

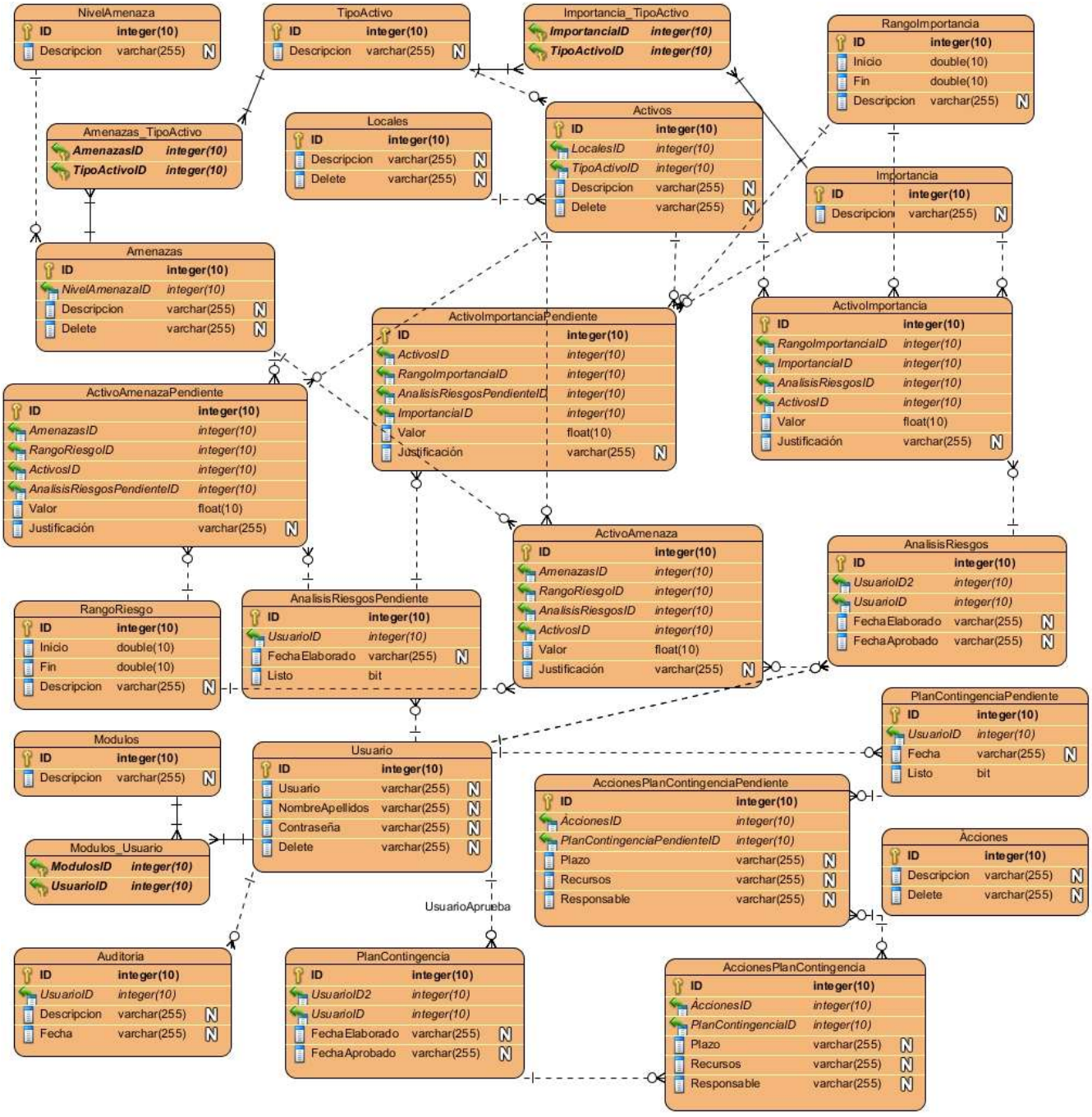

Ilustración 8: Diagrama de Base de Datos

Anexo 4. Interfaz de la pantalla del sistema crear usuario

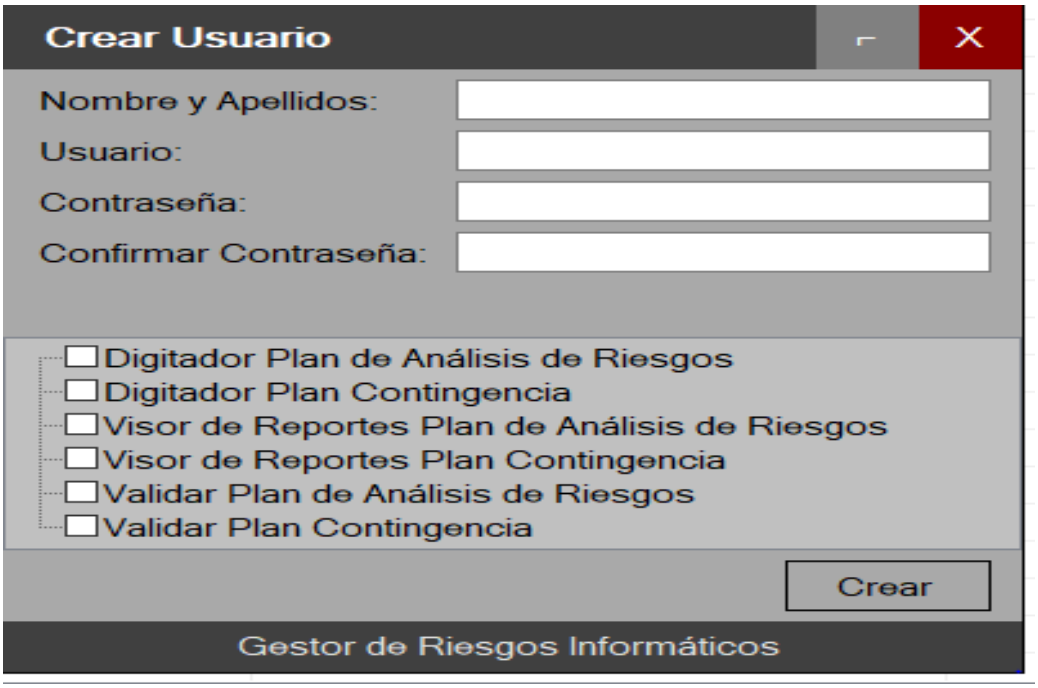

Ilustración 9: Interfaz Crear Usuario

Anexo 5. Interfaz de la pantalla del sistema crear amenaza

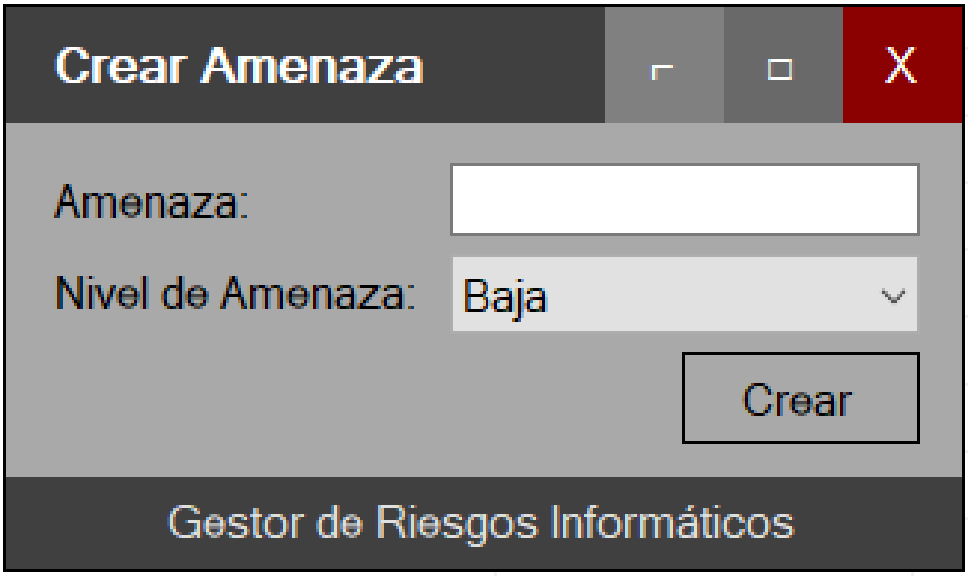

Ilustración 10: Interfaz Crear Amenaza

Anexo 7. Conceptos y normas asociados al problema

- **Activo**: es todo elemento o material digital, físico o humano que puede ser afectado y que requiere protección (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **Riesgo**. Es la posibilidad de que se produzca un impacto determinado a un activo. (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **Impacto:** Es la consecuencia para un activo de la materialización de una amenaza.
- **Amenaza**. Es el evento que puede desencadenar un incidente de la organización, produciendo daños o pérdidas materiales en sus activos. (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **Vulnerabilidad**. Es la debilidad de un activo que puede ser explotado por una amenaza para materializar una agresión sobre dichos activos, se clasifica en alta, media y baja. (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **Evaluación de riesgo:** Se identifican las amenazas, vulnerabilidades y riesgos, sobre la plataforma tecnológica de una organización, con el fin de generar un plan de implementación de los controles que aseguren un ambiente informático seguro, bajo los criterios de disponibilidad, confidencialidad e integridad. Se consideran los siguientes puntos:
    - La probabilidad de una amenaza
    - La magnitud del impacto sobre el sistema

  (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **ISO/IEC 27001:** es la norma internacional auditable que da los parámetros para definir los requisitos para gestionar la seguridad de la información (SGSI). (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **ISO/IEC 27005:** es la norma que proporciona las directrices para la gestión de riesgos en la seguridad de la información. (CASTRO BOLAÑOS & ROJAS MORA, 2013)
- **Resolución 128/2019**
- **Resolución 129/2019**
- **Decreto Ley 360**

Anexos