UNIVERSIDAD DE MATANZAS

FACULTAD DE CIENCIAS TÉCNICAS

CARRERA DE INGENIERÍA INFORMÁTICA

Trabajo para optar por el Título de Ingeniero en Informática.

Título: “Sistema de Gestión para los controles de Seguridad Informática en la Fiscalía Provincial Matanzas”

Autor: Yadián Pérez Bernal

Tutor: Ing. Yanser Falcón Alfonso

Consultor: Ing. Ángel Gabriel Valdés Sarduy

Julio, 2020

Dedicatoria

A mi familia por estar siempre presente, su amor incondicional y por guiarme por el camino correcto.

Gracias por creer en mí en todo momento y darme fuerzas para seguir adelante, por ayudarme a luchar por mis sueños. Simplemente le doy las gracias por existir. Los quiero mucho

AGRADECIMIENTOS

Quiero agradecerle en primer lugar a mi esposa, por asumir una gran carga en este período, por entender mi ausencia y por alentarme en los momentos necesarios.

A mi hijo que es mi motor impulsor, ya que quiero convertirme en su modelo a seguir.

A mi madre, porque con su ejemplo y educación supo conducirme por el camino correcto.

A mis amigos que supieron esclarecer esas dudas complejas en los momentos en que más hacían falta.

Al profesor Ángel que logró sembrar en mi lo importante y útil de la programación Web.

A mi tutor Yanser que me llevo por el camino de los victoriosos.

A la Universidad de Matanzas, donde me estoy formado como profesional.

A TODOS los que hicieron posible que este sueño se hiciera realidad.

Declaración de autoría

Yo, Yadián Pérez Bernal, declaro ser autor de la presente tesis y reconozco a la Universidad de Matanzas los derechos patrimoniales de la misma. Para que así conste firmo la presente a los ____ días del mes de _____ del 2020

_____ _____

Firma del Autor **Firma del Tutor**

Yadián Pérez Bernal

Opinión del usuario del Trabajo de Diploma

Opinión del Tutor del Trabajo de Diploma

Resumen

El proceso de gestión de la seguridad informática, por su complejidad, requiere del establecimiento de una gran variedad de controles, la diversidad de los mismos, conlleva a que su confección y actualización se convierta en un procedimiento engorroso y en muchas ocasiones inefectivo. Como resultado de la presente investigación se desarrolla un sistema informático para la gestión de los controles de seguridad informática (GDSI), en el cual se definen e implementan los procedimientos establecidos en el Plan de Seguridad Informática (PSI) de la Fiscalía Provincial de Matanzas, se propone como objetivo contribuir al aumento de la eficiencia y eficacia del control de los medios informáticos, además del análisis y toma de decisiones. Para su descripción y construcción fue utilizada la metodología ágil de desarrollo de software XP, como ambiente de programación Asp.NetCore 3.1 con el lenguaje de programación C# 7 y gestor de base de datos Postgres.

CONTENIDO

Indice de Tablas

Indice de Ilustraciones

INTRODUCCIÓN

En el mundo de hoy las empresas y organizaciones son completamente dependientes de la tecnología para llevar a cabo sus objetivos. Las informaciones son almacenadas, procesadas y transmitidas en formato digital. En un entorno en que el desarrollo tecnológico ha posibilitado la conexión a Internet desde cualquier lugar y mediante múltiples dispositivos electrónicos.

Entre los ataques que se producen con mayor frecuencia se pueden mencionar los programas malignos, la denegación de servicios, la desfiguración de sitios web, los ataques de ingeniería social, entre otros. En las encuestas anuales realizadas por el prestigioso *Computer Security Institute* (CSI) (1), los ataques con mayor incidencia han sido históricamente los programas malignos, afectando a más del 60% de las instituciones participantes en dichas encuestas.

Los diferentes ataques informáticos pueden provocar la pérdida de la disponibilidad, confidencialidad o integridad de la información; lo cual generalmente implica graves consecuencias para las organizaciones y en muchos casos se ocasionan daños irreparables. Según datos estadísticos hasta el 2016 (2), las pérdidas promedio de las grandes empresas a nivel mundial debido a incidentes de seguridad informática son valoradas en 5,9 millones de dólares al año.

Teniendo en cuenta, además, el crecimiento exponencial de los programas malignos, los cuales aumentan por decenas de miles diariamente (3) , (4), las más de ocho mil nuevas vulnerabilidades de sistemas operativos y aplicaciones descubiertas anualmente (5), (6), y la organización cada vez más estructurada de los atacantes informáticos (7); es evidente la necesidad de garantizar la seguridad informática de las instituciones, mediante un adecuado proceso de gestión de todas las medidas y controles necesarios.

Actualmente existen varios modelos, estándares, recomendaciones y regulaciones relacionados con la gestión de la seguridad de la información. Entre ellos se destaca el estándar *ISO/IEC* 27001, el cual posee gran aceptación a nivel internacional y constituye una norma certificable. En nuestro país también existen normas que regulan la seguridad informática como es la Resolución 128/2019 y 129/2019. De manera general todas plantean que el proceso de gestión implica la realización de varias acciones que conforman un ciclo cerrado para la mejora continua del sistema de seguridad informática. Como parte de este proceso

se recomiendan más de 100 controles[1] que no son solo de corte técnico, sino que involucran también aspectos organizativos y de recursos humanos.

La gran cantidad y variedad de controles que es necesario implementar, en un entorno donde la tecnología evoluciona a una gran velocidad, hace que la gestión de la seguridad informática sea un proceso complejo, en el que hay que lograr una adecuada armonía entre tecnologías, personas y procedimientos (8).

Por otra parte, se presentan también los siguientes retos a la hora de gestionar la seguridad informática:

- La cantidad y variedad de sistemas a asegurar: la mayoría de las organizaciones posee una gran cantidad de aplicaciones, sistemas operativos y dispositivos de red que necesitan ser protegidos, cada uno con sus diferentes mecanismos de soporte y configuraciones de seguridad.
- La necesidad de responder rápidamente a nuevas amenazas: las organizaciones necesitan frecuentemente reconfigurar las aplicaciones o instalar parches, para eliminar nuevas vulnerabilidades que son descubiertas constantemente y utilizadas por los atacantes. Cada minuto perdido en la implantación de un parche de seguridad significa la exposición a amenazas concretas y un gran riesgo para las instituciones.
- La falta de interoperabilidad e integración de los sistemas de seguridad: es necesario utilizar varias herramientas de seguridad informática que emplean formatos propietarios y nomenclaturas diferentes; lo cual no permite su interoperabilidad y una gestión integrada de las mismas.

Para ejemplificar un poco lo anteriormente expresado, se brindan algunos datos de la Fiscalía Provincial de Matanzas (FPMZ), siendo esta entidad la seleccionada para realizar los estudios de la presente investigación. En la misma, se encuentran interconectadas en red nacional más de 250 computadoras y 62 tiene conexión a internet, encontrándose estas en una red aislada de la red ordinaria y alrededor de 20 servidores. Entre los sistemas operativos utilizados se encuentran *Microsoft Windows* (7,10, *Server* 2012, *Server* 2016), NOVA[2], Ubuntu. Por otra parte, existe una gran variedad de aplicaciones que soportan los diferentes servicios telemáticos: *Apache*, *IIS*, *Zimbra*, entre otras. Esto hace que la "superficie de ataque" sea grande y por tanto es necesario dar seguimiento a los parches y vulnerabilidades de una amplia gama

[1] Control: medida, salvaguarda.

[2]NOVA: Distribución cubana de GNU/Linux desarrollada en la UCI.

de sistemas operativos, aplicaciones, soportes de almacenamiento y servicios a autorizados a los usuarios de la red.

Los aspectos mencionados anteriormente, unidos a temas de preparación de los recursos humanos y una deficiente gestión, provocan que en muchas ocasiones el sistema de seguridad informática de las instituciones resulte inefectivo. En Cuba, por ejemplo, el 60,7% de las inspecciones realizadas a los sistemas de seguridad informática de diferentes instituciones cubanas, han obtenido la calificación de "Vulnerable" o "Muy Vulnerable" (9). Teniendo en cuenta lo planteado anteriormente se identificó el siguiente **problema científico**: ¿Cómo contribuir a la gestión de los controles de Seguridad Informática en la Fiscalía Provincial de Matanzas (FPMZ)?

Como **hipótesis** se establece que, el desarrollo de una aplicación informática contribuirá a la gestión eficiente y eficaz de los controles de seguridad informática, así como el análisis de la misma.

Para solucionar el problema planteado se consideró como **objeto de estudio** el proceso de gestión de la seguridad de la información, y como **campo de acción** la informatización en la gestión de controles de seguridad informática.

El **objetivo general** de la presente investigación consiste en desarrollar un sistema para la gestión informatizada de controles de seguridad informática, que contribuya a aumentar la efectividad de los mismos y a disminuir la complejidad de la gestión de la seguridad informática.

Para alcanzar el objetivo general se plantean los siguientes **objetivos específicos**:

1. Realizar una revisión bibliográfica que establezca el estado del arte y la fundamentación teórica de la investigación.
2. Determinar los controles de seguridad informática que son automatizables.
3. Analizar metodologías de desarrollo de software para documentar el ciclo de vida de la aplicación informática, así como herramientas apropiadas para su diseño e implementación
4. Diseñar e implementar un sistema que permita la gestión informatizada de controles de seguridad informática.
5. Validar el sistema mediante su aplicación en la FGR y pruebas de software.

Métodos Teóricos:

- **Método histórico**: Se investigó el comportamiento de la gestión de seguridad informática a lo largo de los años en la FPMZ.
- **Histórico lógico**: Fue utilizados para el estudio crítico de los trabajos anteriores en esta temática.

Métodos Empíricos:

- **Entrevista y Encuesta***:* Se entrevistaron a los máximos encargados de la gestión de Seguridad Informática y a los Jefe de los departamentos Informática de la FPMZ, para obtener información de interés.
- **Análisis de Documentos**: Fueron analizados los planes de Seguridad Informática, normas y resoluciones vigentes del tema.

Para exponer los resultados de la investigación, este documento está dividido en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. El contenido de los capítulos se distribuye de la siguiente forma:

- **Capítulo I: "Marco teórico- referencial"**

Contiene una explicación detallada de los referentes teóricos que argumentan la propuesta y permiten un acercamiento al objeto de estudio. Se comenta sobre el estudio del estado del arte y las tendencias y tecnologías actuales que serán usadas para el desarrollo de la aplicación informática.

- **Capítulo II: "Solución teórica del problema científico"**

En dicho capítulo se describe el proceso de gestión de la Seguridad Informática y la propuesta de solución. Se especifican los procesos que darán solución a la problemática y se definen los requisitos funcionales y no funcionales de la aplicación.

- **Capítulo III: "Resultados del trabajo desarrollado. Elementos de la validación práctica de la propuesta de solución del problema científico."**

Se muestran los diagramas de despliegue y secuencia los cuales guían todo el proceso de desarrollo, también se explica el tratamiento que se le da a seguridad, se realiza un análisis del esfuerzo y el costo de la solución propuesta y finaliza con las pruebas de caja blanca y caja negra.

CAPITULO 1: Marco Teórico- Referencial

En el presente capítulo se presenta una valoración de los principales modelos, estándares, recomendaciones y regulaciones que, tanto a nivel internacional como nacional, están relacionados con la gestión de la seguridad de la información. Se realiza, además, una valoración y comparación de los principales trabajos e investigaciones existentes en el campo de la informatización de controles de seguridad informática y se analizan un grupo de sistemas que, desde el punto de vista práctico, pueden ser empleados para lograr este propósito. También se analizan las principales tecnologías a utilizar entre las que destacan los lenguajes de programación, *framework*[3], gestores de bases de datos (BD) y las metodologías de desarrollo de software.

1.1. Antecedentes del trabajo

En la entidad en estudio Fiscalía Provincial de Matanzas previamente no existe un trabajo previo a esta investigación por lo que hay que hacer referencia a sistemas existentes en el mundo. Existe un grupo de software que realizan trabajos relacionados con el proceso de la gestión de la seguridad Informática, pero ninguno a nivel de documentación sino de la gestión de activos informáticos solamente. Este es un problema al cual se le dará solución durante el transcurso de esta investigación, puesto que es el núcleo central de la misma. Se procede a la investigación de los diferentes *softwares* a nivel nacional e internacional que realizan la gestión de la seguridad informática. De estas se decidió realizar una selección de las plataformas más utilizadas actualmente conjuntamente con una breve caracterización de los seis más importantes.

GRHS (Gestor de recursos de *hardware* y *software*) Sistema que realiza el inventario de la información de los componentes de hardware y software en una red de computadoras. Facilita detectar cambios en dichos componentes, clasificarlos en Autorizados o No autorizados, según se haya establecido en la entidad, así como tomar acciones de control (apagar, hibernar, suspender, reiniciar, enviar notificación por correo) ante los cambios no autorizados. Posee una interfaz web que permite la visualización y administración de la información obtenida de las computadoras. (10)

[3] Framework o marco de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto

CIMET Control Informático de Medios Técnicos. *Software* implementado en los Joven Club de Computación y Electrónica del país. Creado por Desoft en trabajo conjunto con Copextel. Soló es usado en los Joven Club a nivel nacional. Tiene módulos donde se recogen las principales características de los equipos que son atendidos por Copextel. No está implementado en las empresas del país a no ser las mencionadas anteriormente. No incluye módulos de inspecciones a los equipos, no controla las deficiencias, ni incidencias, ni acciones a desarrollar en caso de existir problemas.

GLPI (acrónimo: en francés, *Gestionnaire Libre de Parc Informatique*) es una solución libre de gestión de servicios de tecnología de la información (ITSM), un sistema de seguimiento de incidencias y de solución *service desk*. Este *software* de código abierto está editado en PHP y distribuido bajo una licencia GPL.

Es una aplicación web que ayuda las empresas con la gestión de su sistema de información. Entre sus características, esta solución es capaz de construir un inventario de todos los recursos de la organización y de gestionar tareas administrativas y financieras. Las funcionalidades de este *software* ayudan a los administradores de red a crear una de base de datos de activos técnicos, así como a gestionarla y proporciona un historial de las intervenciones de mantenimiento. (11)

ASSETS NS El control de los medios básicos o activos fijos informáticos de la Estación Experimental de Pastos y Forrajes “Indio Hatuey” se hace a través del Sistema ASSETS NS en el área económica de la institución. Este un Sistema de Gestión Integral, que permite el control de una serie de procesos entre los cuales se encuentran Inventario y Activos Fijos. (12)

El Sistema ASSETS NS tiene la particularidad que solo es manejado por los trabajadores del área económica. Las demás áreas del centro no acceden a este sistema, por seguridad. No obstante, la información que se obtiene de ellos no es la más completa. No se llevan controles de fichas técnicas de los equipos informáticos, no se recoge el historial de las inspecciones que se les realiza a los equipos, las incidencias, el control de software, el control de salvas de información que se les realiza a los equipos cuya información es de alta importancia.

SGSI (Sistema de Gestión de Seguridad de la Información) El SGSI consiste básicamente en un conjunto de políticas para definir, construir, desarrollar y mantener la seguridad del equipo basado en hardware y recursos de software (*ISO*/*IEC* 27001, 2005); estas políticas, muestran la manera en que los recursos del computador pueden ser utilizados. Es importante definir el enfoque que se tiene para la evaluación del estado de seguridad de información en una organización, de ahí la razón de definir un marco jerárquico para priorizar lo que se tiene. (13)

OCS (*Open Computer and Software*) es un *software* libre que permite a los Administradores de TI (Tecnología de Información) gestionar el inventario de sus activos de TI*. OCS-NG* recopila información sobre el hardware y software de equipos que hay en la red que ejecutan el programa de cliente OCS ("agente OCS de inventario"). OCS puede utilizarse para visualizar el inventario a través de una interfaz web. Además, OCS comprende la posibilidad de implementación de aplicaciones en los equipos de acuerdo a criterios de búsqueda. Además, tiene muchas opciones más como escanear la red por medio del *IPDiscovery*, o instalar aplicaciones remotamente creando *Builds*. (14)

Resumiendo, tanto las aplicaciones nacionales como las internacionales su objeto principal es el control de los medios básicos ninguno de estos sistemas cuenta con el registro de aperturas, movimientos o de bajas de los medios o componentes informáticos que es nuestro objeto principal de investigación por lo que podemos llegar a la conclusión que no se adaptan a los requerimientos de nuestro cliente.

En resumen, estos sistemas investigados su objetivo fundamenta se basa en la gestión del *hardware* y *software,* unos en mayor profundidad que otro, en el sistema que se pretende desarrollar tendrá incluido este módulo, pero para resolver al 100% la problemática existente de la gestión de la documentación del PSI necesitamos desarrollar un nuevo sistema, ya que ninguno de los sistemas estudiados presentan módulos desarrollados que cumplan con las exigencia vigentes de los modelos propuestos en la resolución 129 para el desarrollo del PSI.

1.2. Principales estándares y regulaciones sobre seguridad informática tanto en el ámbito nacional como el internacional

En el contexto de la seguridad de la información existen estándares que constituyen normas certificables, marcos de trabajo que representan una recopilación de mejores prácticas y regulaciones que son de obligatorio cumplimiento en determinadas naciones.

1.3.1 Decretos y Resoluciones Nacionales

En nuestro país recientemente se ha estado organizando esta actividad de la seguridad informática por lo que a continuación se mencionan los decretos y resoluciones aprobados recientemente en el 2019: (15) (16)

1- Decreto-Ley No. 370 del 17 de diciembre de 2018 Sobre la informatización de la sociedad cubana. Esta es una norma jurídica de rango superior del país en materia de informatización, y busca elevar la soberanía tecnológica en beneficio de la sociedad, la economía, la seguridad y la defensa nacional,

contrarrestar las agresiones cibernéticas, salvaguardar los principios de seguridad de nuestras redes y servicios, así como defender los logros alcanzados por nuestro Estado socialista.

2- Decreto No. 359 del 5 de junio de 2019 Sobre el desarrollo de la industria de programas y aplicaciones informáticas. La norma jurídica complementa el Decreto-Ley de informatización de la sociedad, ya que establece las regulaciones generales aplicables a la determinación del alcance de la Industria cubana de programas y aplicaciones informáticas para promover, perfeccionar e incrementar la producción nacional y las exportaciones de los productos de dicha industria, así como contribuir con la sustitución de importaciones.

3- Decreto No. 360 del 5 de junio de 2019 Sobre el establecimiento de la seguridad de las tecnologías de la informatización y la comunicación y la defensa del ciberespacio nacional. La norma complementa el Decreto-Ley No. 370 al regular el empleo seguro de las Tecnologías de la Información y la Comunicación (TIC) para la informatización de la sociedad, la defensa del ciberespacio nacional y establece la seguridad de las TIC y de los servicios y las aplicaciones que soportan, así como de las infraestructuras críticas de las TIC, con la finalidad de contar con una estrategia de fortalecimiento y sostenibilidad.

4- Acuerdo 8611 del Consejo de Ministros para la implementación de la estrategia de desarrollo de la banda ancha en cuba. Este organiza, regula y traza las líneas para el desarrollo integral de la banda ancha nacional, que sirva de guía a las entidades nacionales y a la población, en el desarrollo, la explotación y utilización de los servicios de comunicaciones, así como encarga al Ministerio de las Comunicaciones (MINCOM) con el control de su implementación.

5- Resolución 124 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba el reglamento para las reglas de producción y evaluación de las aplicaciones informáticas nacionales. La norma regula las reglas básicas para la producción de programas y aplicaciones informáticas, la evaluación del proceso de desarrollo de estas y el proceso para la evaluación de la calidad de las mismas, solicitada por los desarrolladores o los comercializadores de productos nacionales o importados, o por cualquier persona interesada en adquirirlos.

6- Resolución 125 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba la inscripción de aplicaciones informáticas comercializables. La norma establece el sistema de inscripción de productos de software con el propósito de ordenar los procesos de producción y comercialización en la industria de programas y aplicaciones informáticas, así como ordenar, controlar, almacenar y mantener actualizada la información sobre estos productos existentes en el país.

7- Resolución 126 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba el reglamento para el control de las redes informáticas. Regula las medidas de control y los tipos de herramientas de seguridad que se implementan en las redes privadas de datos, inscritas en el Control Administrativo Central Interno del Ministerio de Informática y Comunicaciones (MIC).

8- Resolución 127 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba el reglamento sobre proveedores de servicios de hospedaje y alojamiento. Destinada a las personas jurídicas, la norma regula la organización, el funcionamiento y la expedición de licencias de operación del proveedor de servicios públicos de alojamiento y hospedaje en el entorno de internet en el territorio nacional.

9- Resolución 128 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba el reglamento de seguridad de las Tecnologías de la Información y la Comunicación. Este reglamento complementa las disposiciones del Decreto No. 360 en materia de seguridad de las TIC y establece las funciones de los sujetos que intervienen en esta, así como garantiza con un respaldo legal que responda a las condiciones y necesidades del proceso de informatización de la sociedad.

10- Resolución 129 del 24 de junio de 2019 del Ministro de Comunicaciones que aprueba la metodología sobre la gestión de la seguridad informática en todo el país.

Esta resolución es la que rige la elaboración del plan de seguridad informática en todas las entidades del país, en ella se establecen todos los aspectos a seguir para su confesión.

La entrada en vigor de todas estas normativas fue el pasado 4 de julio del 2019, y se derogan las normas jurídicas siguientes:

- Acuerdo No. 5586 del Consejo de Ministros que aprueba los Lineamientos para el desarrollo en Cuba del Comercio Electrónico de 26 de diciembre de 2005.
- Acuerdo No. 6058 del Comité Ejecutivo del Consejo de Ministros que aprueba los Lineamientos de Seguridad de las Tecnologías de la Información de 9 de julio de 2007.
- Resolución No. 127 del Ministro de la Informática y las Comunicaciones (MIC) de 24 de julio de 2007, que aprobó el Reglamento de Seguridad de las Tecnologías de la Información.
- Resolución No. 33 del Ministro de la Informática y las Comunicaciones, de fecha 24 de enero de 2008, que estableció el Sistema de Inscripción de Productos de Software, con el propósito de ordenar los procesos de producción y comercialización en esa industria.

- Resolución No. 55 del Ministro de la Informática y las Comunicaciones de fecha 9 de marzo de 2009, que regula la organización, el funcionamiento, registro y expedición de licencias de operación para los proveedores de servicios públicos de Alojamiento, Hospedaje y Aplicaciones.
- Resolución No. 104 del Ministro de la Informática y las Comunicaciones de fecha 16 de junio de 2011, que modifica la Resolución No. 55 del Ministro de la Informática y las Comunicaciones de fecha 9 de marzo de 2009.
- Resolución No. 192 del Ministro de Comunicaciones de 20 de marzo de 2014, que aprueba el reglamento para contrarrestar el envío de mensajes masivos dañinos a través de las redes de telecomunicaciones.

1.3.2 **Normas Internacionales ISO/IEC**

La Organización Internacional para la Estandarización (ISO) y la Comisión Electrotécnica Internacional (IEC) han elaborado un grupo de estándares que constituyen una referencia a nivel mundial en la temática de seguridad de la información. La serie ISO/IEC 27000 está destinada completamente a la seguridad informática, donde los estándares ISO/IEC 27001 y 27002 abordan el tema de manera general, y el resto tratan temas específicos que complementan a los anteriores.

Como se puede apreciar, el estándar propone un enfoque de procesos donde la seguridad de la información no es un estado que se alcanza en determinado instante de tiempo y permanece invariable, sino que es un proceso continuo que necesita ser gestionado. Se proponen varias acciones que forman un ciclo cerrado para la mejora continua del sistema. La norma certificable ISO/IEC 27001 (17) especifica, 133 controles de seguridad divididos en 11 dominios y 39 objetivos de control. Estos controles son explicados detalladamente en la guía de buenas prácticas ISO/IEC 27002 (18), donde pueden apreciarse un grupo de controles técnicos, otros más relacionados con los recursos humanos y un grupo de controles de tipo organizativo.

1.3.3 NIST SP 800-53

La división de seguridad del Instituto Nacional de Estándares y Tecnologías de EEUU (NIST) posee gran prestigio a nivel internacional y desarrolla un grupo de especificaciones técnicas, procedimientos y estándares sobre seguridad de la información. Las publicaciones especiales (SP) en su serie 800 están destinadas completamente a esta temática. En ellas se presentan varías guías y estándares resultantes de un trabajo colaborativo entre la industria, el gobierno y organizaciones académicas.

En particular el estándar NIST SP 800-53 (19) especifica los controles de seguridad a implementar en las agencias federales de los EEUU. El estándar está asociado a una ley federal establecida en el año 2002 conocida como FISMA (*Federal Information Security Management* Act), la cual es de obligatorio cumplimiento y establece, a través de los estándares FIPS 199 y 200 (*Federal Information Processing Standards*), los requerimientos mínimos de seguridad en dependencia del tipo de organización y de la información que se procese. Según la categoría de la institución y el nivel de impacto de la misma se seleccionan los controles descritos en NIST SP 800-53 que deben ser implementados. A pesar de que el estándar NIST SP 800-53 constituye una regulación solo en EEUU, esta publicación es muy utilizada y referenciada a nivel internacional por el prestigio que normalmente poseen los estándares del NIST. El estándar contiene una recomendación de controles de seguridad informática que pueden servir en general para diferentes organizaciones y toda la documentación pueden ser obtenida sin costo alguno[4]. En este caso se especifican 18 dominios de seguridad y 198 controles, clasificados en técnicos, operacionales y de gestión.

1.3.4 ISF-SoGP. Recomendación de buenas prácticas de seguridad.

El Fórum para la Seguridad de la Información (ISF por sus siglas en inglés), compuesto por más de 300 organizaciones, ha desarrollado lo que denominan el Estándar de Buenas Prácticas (SoGP) para la seguridad informática. Esta recomendación aborda el tema de la seguridad de la información desde la perspectiva del negocio, enfocándose en la implementación de los controles necesarios para mitigar los riesgos asociados con los sistemas de información críticos (20).

El estándar está dividido en seis aspectos fundamentales, para los cuales se establecen 36 objetivos y 166 controles. Los aspectos son: gestión de la seguridad informática, aplicaciones de negocio críticas, instalaciones de sistemas, redes, desarrollo de sistemas y entorno del usuario final.

El SoGP posee gran aceptación entre los miembros del ISF, pero a nivel internacional su aplicación es **limitada, debido a que predominan otros estándares con mayor** aceptación como ISO/IEC 27001.

1.3.5 ISM3. Modelo de madurez para la gestión de la seguridad informática.

El consorcio ISM3, compuesto por varias empresas y organizaciones, ha desarrollado el Modelo de Madurez de Gestión de la Seguridad Informática (ISM3 por sus siglas en inglés).

[4] Serie NIST SP 800: http://csrc.nist.gov/publications/PubsSPs.html

Este modelo pretende extender los principios de calidad establecidos en ISO 9001 a un SGSI. En lugar de estar orientado a controles, se enfoca en los procesos de seguridad informática que pueden ser comunes a todas las organizaciones (21) (22) (23).

1.3.6 Estándares y regulaciones para sectores específicos: PCI DSS, HIPAA.

Existen otros estándares que son muy utilizados a nivel internacional, pero que abordan la temática de la seguridad de la información para sectores específicos como la salud o el sector comercial. Tales son los casos de HIPAA y PCI DSS respectivamente.

El Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago (*PCI DSS* por sus siglas en inglés) ha sido desarrollado por el *Payment Card Industry Security Standards Council* (*PCI SSC*), como una guía que ayude a proteger los datos de tarjetas de débito y crédito durante su procesamiento, almacenamiento y transmisión, con el fin de prevenir los fraudes relacionados con las tarjetas de pago. Las compañías que procesan, guardan o trasmiten datos de tarjetas deben cumplir con el estándar o arriesgan la pérdida de sus permisos para procesar las tarjetas. La versión actual del estándar (24) especifica 12 requisitos mínimos organizados en 6 objetivos de control.

Por otra parte, la Regla de Seguridad del HIPAA (*Health Insurance Portability and Accountability Act*) (25) es también muy conocida en el ámbito de la seguridad informática y constituye una regulación de obligatorio cumplimiento para las instituciones de salud de EEUU, con el objetivo de proteger los datos electrónicos de los pacientes. La norma establece 18 dominios y 42 controles agrupados en tres categorías: administrativos, físicos y técnicos.

Es importante señalar que tanto PCI DSS como HIPAA proponen una menor cantidad de controles de seguridad informática, lo cual está asociado a que tienen un propósito más específico y están orientados a determinados sectores.

1.3.7 Estándares y regulaciones de alcance regional.

Existen también otras guías y regulaciones que tienen un carácter regional. En general la mayoría de los países poseen regulaciones relacionadas con la seguridad informática. Algunas recomendaciones son conocidas fuera de sus fronteras porque constituyen guías de buenas prácticas aplicables a diferentes organizaciones. Cabe mencionar aquí al IT *Baseline Protection Catalog* (IT-*Grundschutz*) de Alemania y el Manual de Seguridad de la Información (ISM) de Australia.

1.3.8 CAG. 20 controles críticos de seguridad

Como puede apreciarse, los estándares y recomendaciones analizados anteriormente especifican una gran cantidad de controles diferentes que deben ser implementados para garantizar la seguridad de la información. Esto resulta realmente complejo para los responsables de la gestión de la seguridad informática, ya que deben garantizar la implementación y el seguimiento de un gran número de controles. En este sentido se hace necesaria una labor de síntesis, donde se identifiquen los controles más críticos de acuerdo al análisis de riesgos, así como las acciones elementales dentro de los mismos para garantizar un nivel básico de seguridad de la información.

1.3.9 Resumen de estándares, guías y regulaciones de seguridad informática.

En la Tabla 1 se ofrece un resumen de los estándares, recomendaciones y regulaciones mencionados previamente. Se ha realizado una clasificación con respecto al alcance, enfoque y características fundamentales de cada uno.

Tabla 1: Resumen de estándares y regulaciones de seguridad informática

	Clasificación	**Alcance**	**Enfoque**	**Características**	**Procesos**
ISO/IEC 27001	Estándar. Norma certificable.	Internacional General.	Orientado a procesos.	Ciclo PDCA, 4 fases, 7 procesos, 30 acciones.	Establecer, implementar, operar, monitorizar, revisar, mantener, mejorar.
ISO/IEC 27002	Mejores prácticas.	Internacional General.	Orientado a controles.	11 dominios, 39 objetivos, 133 controles.	-
NIST SP 800-53	Regulación.	Regional (EEUU). General.	Orientado a controles.	18 dominios, 198 controles.	-

ISF-SoGP	Mejores prácticas.	Internacional General.	Orientado a controles.	6 dominios, 36 objetivos, 166 controles.	-
ISM3	Modelo.	Internacional General.	Orientado a procesos.	3 categorías, 45 procesos, 5 niveles de madurez.	Gestión estratégica, táctica y operacional.
CoBIT 4.1	Mejores prácticas.	Internacional General (TI).	Orientado a procesos.	4 fases, 8 macroprocesos, 34 procesos, 210 controles.	Planificar y organizar, adquirir e implementar, entrega y soporte, monitorear y evaluar.
ITIL v3	Mejores prácticas.	Internacional General (TI).	Orientado a procesos.	5 fases, 86 tópicos fundamentales.	Estrategia, diseño, transición, operación, mejora continua.
PCI DSS v2	Regulación.	Internacional Sectorial (tarjetas de pago)	Orientado a controles.	6 objetivos de control, 12 requisitos.	-
HIPAA	Regulación.	Regional (EEUU) Sectorial (servicios médicos)	Orientado a controles.	3 categorías, 18 dominios, 42 controles.	-
IT-Grundschutz	Mejores prácticas.	Regional (Alemania). General.	Orientado a controles	5 dominios, 46 objetivos, + 500 controles.	-

ISM	Regulación.	Regional (Australia) General.	Orientado a controles.	5 dominios, 20 objetivos, 91 controles.	-
Res.128/2019 MIC	Regulación.	Regional (Cuba). General.	Orientado a controles.	11 dominios, 89 controles.	-
CAG v3	Mejores prácticas.	Internacional General	Orientado a controles	20 controles	-

1.3. **Efectividad de los controles. Métricas de seguridad informática.**

La gestión de la seguridad informática implica realizar un grupo de acciones que conforman un ciclo cerrado para la mejora continua del sistema. Aunque la terminología de los procesos propuestos por los diferentes estándares varía, todos plantean una fase de revisión y evaluación como parte de la gestión, de tal manera que pueda determinarse la efectividad del sistema de seguridad informática con respecto a las políticas y los objetivos de seguridad establecidos. Esto se logra mediante la adecuada definición de métricas que permitan medir el desempeño del sistema de seguridad informática.

Las métricas son el resultado de la recopilación, análisis y reporte de datos relevantes del sistema de seguridad informática que permiten la toma de decisiones (26). Es importante señalar que el término más utilizado en el contexto de la seguridad informática es métricas, pero también se emplean otros como indicadores o medidas (*measures* en inglés) en la literatura especializada. A los efectos de este trabajo los términos mencionados poseen el mismo significado.

La publicación NIST IR 7564 (*Directions in Security Metrics Research*) demuestra que, este es un campo que todavía se encuentra en investigación y desarrollo. La mayoría de lo que se ha escrito sobre métricas de seguridad informática son recomendaciones y guías para la correcta definición de las mismas. Sin embargo, poco se ha reportado sobre métricas que han probado ser útiles en la práctica (27). En (28) se ofrecen varias razones que argumentan que la medición de la seguridad informática de un sistema es un problema complejo.

Entre las recomendaciones más reconocidas en esta temática se encuentran ISO/IEC 27004 (29) y NIST SP 800-55 (19), que explican cómo llevar a cabo un programa de medición en seguridad informática. Sin

embargo, en estos documentos solo se brindan algunos ejemplos de métricas, dejando la completa definición de las mismas a las organizaciones.

1.4. Tipos de métricas

La madurez de una organización y de su sistema de seguridad informática determinan los tipos de métricas que pueden utilizarse. Esta madurez está determinada por la existencia e institucionalización de procesos y procedimientos en la organización. A medida que un sistema de seguridad informática se desarrolla y evoluciona, las políticas se tornan más detalladas y mejor documentadas, los procesos se vuelven más estandarizados y repetibles; de manera tal que existe una mayor cantidad de datos que pueden ser usados para medir la efectividad del sistema. Las métricas de seguridad informática se clasifican entonces de la siguiente manera (26):

- Métricas de implementación: evalúan la implantación de controles de seguridad informática, de acuerdo a las políticas de seguridad definidas. Responden a la pregunta: ¿qué controles de seguridad informática se encuentran implementados? Este tipo de métricas permite evaluar el cumplimiento de regulaciones y normas establecidas.
- Métricas de efectividad: permiten determinar si los controles de seguridad informática implementados realizan la función para la que fueron concebidos. Este tipo de métricas permite determinar si el sistema de seguridad informática diseñado es efectivo o si necesita ser revisado y mejorado.
- Métricas de impacto en el negocio: permiten determinar el impacto que la seguridad informática ha tenido en el cumplimiento de la misión de la organización. Este tipo de métricas ofrece una visión más global, considerando la seguridad informática como un medio para alcanzar los objetivos de la organización y no como un fin en sí misma.

1.5. Tendencias Tecnológicas

Las aplicaciones *Web* tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de *software* tradicionales. Las tecnologías ASP.NET y AJAX permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales. (30) (31)

No se requieren complicadas combinaciones de *Hardware/Software* para utilizar estas aplicaciones. Solo una computadora con un buen navegador *Web*. Además, estas aplicaciones son fáciles de usar, ya que no

requieren conocimientos avanzados de computación. Otro beneficio de usar aplicaciones *Web* es que, al residir y correr en los servidores del proveedor, usan en muchos casos la memoria de las computadoras donde corren, dejando más espacio para correr múltiples aplicaciones al mismo tiempo sin incurrir en frustrantes deterioros del rendimiento. Además, estas aplicaciones pueden ser utilizadas por múltiples usuarios al mismo tiempo, por lo que se facilita el trabajo a distancia.

La arquitectura general de una aplicación *Web* es la de un sistema cliente/servidor.

Una de sus ventajas más significativas es su forma de instalación y distribución. Normalmente instalar una aplicación *Web* consiste en configurar los componentes del lado del servidor en la red y no es necesaria una instalación o configuración en el lado cliente.

1.6. **Arquitectura cliente- servidor:**

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información, donde las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. (32)

Los clientes interactúan con el usuario, frecuentemente se comunican con procesos que se encargan de establecer la conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de seguridad.

Los servidores deben manejar fundamentalmente los interbloqueos (varios clientes accediendo a un mismo recurso de la aplicación) y la recuperación ante las fallas que puedan existir. Por lo que el ordenador asociado con los servidores debe ser más poderoso que el de los clientes. Estos servidores deben manejar entre otros servicios la administración de la red, el control y la administración de la entrada a la aplicación.

En esta arquitectura existe una división en cuanto a capacidad del proceso entre los clientes y los servidores que deben estar conectados entre sí mediante una red, esto trae ventajas a nivel organizativo porque se centraliza la gestión de la información y se separan las responsabilidades, lo que facilita y clarifica el diseño del sistema (33).

Entre las principales características de la arquitectura cliente/servidor se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.

- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Esta arquitectura cliente/servidor ofrece varias ventajas:

- Aumenta la productividad ya que los usuarios pueden utilizar herramientas que le son familiares, como hojas de cálculo y herramientas de acceso a bases de datos. Además, una interfaz gráfica de usuario consistente reduce el tiempo de aprendizaje de las aplicaciones.
- Menores costos de operación permitiendo un aprovechamiento mejor de los sistemas existentes y proporcionando un mejor acceso a los datos. El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas. Mejora en el rendimiento de la red porque parte del proceso se reparte con los clientes y este se conecta al servidor cuando es necesario, obtiene los datos y cierra la conexión optimizando con esto el tráfico de la red al devolver solamente los datos que la aplicación necesita.

1.7. Herramientas de Trabajo

C# y .Net: Se seleccionó este lenguaje y esta plataforma de trabajo, por ser la estudiada en clases durante la carrera, por su gran sencillez y otras características que se plantea a continuación**:** (34) (35)

- **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
    - El código escrito en C# es **autocontenido**, lo que significa que no necesita de ficheros adicionales al propio código fuente tales como ficheros de cabecera o ficheros IDL
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.

- No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (**.**) acceder a miembros de espacios de nombres (**::**)

- **Modernidad:** C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción *foreach* que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico *string* para representar cadenas o la distinción de un tipo *bool* específico para representar valores lógicos.

- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

  C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

  En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores *public*, *private* y *protected*, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

  Respecto a la herencia -a diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

  Por otro lado, y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario

de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros)
- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de *.NET* tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active -ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción *using*.
- **Seguridad de tipos:** C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:
    - Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (*downcasting*) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.
    - No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del código fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

- Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.
- Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación)
- A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.
- Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

- **Instrucciones seguras:** Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un *switch* ha de terminar en un *break* o *goto* que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.
- **Sistema de tipos unificado:** A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada *System.Object,* por lo que dispondrán de todos los miembros definidos en esta clase (es decir, serán "objetos")

  A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de *boxing y unboxing* con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- **Extensibilidad de tipos básicos:** C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.
- **Extensibilidad de operadores:** Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

  Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y -- el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postifja; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta (+=) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir !=)

  También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

  Extensibilidad de modificadores: C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

- **Versionable:** C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoque errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

  Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

  - Se obliga a que toda redefinición deba incluir el modificador *override*, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría *override*. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con *override* no existe en la clase padre se producirá un error de compilación.
  - Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador *unsafe*) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.
- **Compatible:** Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados *Platform Invocation Services (PInvoke)*, la posibilidad de acceder a código nativo escrito como funciones

sueltas no orientadas a objetos tales como las DLLs de la API *Win32*. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

ASP.NET: Se seleccionó por formar parte de plataforma de trabajo de Visual *Studio* que estudiamos durante la carrera y por tener mayores habilidades. ASP es un entorno de secuencias de comandos en el lado del servidor que puede utilizar para crear y ejecutar aplicaciones de servidor Web dinámicas, interactivas y de alto rendimiento. ASP.Net es un ambiente de programación construido sobre el entorno NGWS (*New Generation Windows Services*, o sea, Servicios de la Nueva Generación de *Windows*), que permite crear poderosas aplicaciones de Internet. Puede aprovechar las ventajas del enlace anticipado, la optimización nativa y los servicios de caché desde el primer momento por lo que aporta un mejor rendimiento. Debido a que permite el uso de una gran variedad de lenguajes de programación tiene gran flexibilidad. Este ambiente facilita la realización de tareas comunes y con servicio de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados. Con la autenticación de *Windows* integrada a la configuración por aplicaciones, se puede tener la completa seguridad de que las aplicaciones están a salvo. Permite la creación de componentes reutilizables a través de la creación de Controles de Usuario (*User Controls*). Un control de usuario sigue la misma estructura que un formulario web, excepto que los controles derivan de la clase *System.Web.UI.UserControl*, y son almacenados en archivos ASCX. Como los archivos ASPX, un ASCX contiene etiquetas HTML o XHTML, además de etiquetas para definir controles Web y otros controles de usuario. (31)

BOOTSTRAP es un *framework* en su origen creado por Twitter, que permite crear interfaces Web con de *Cascading Style Sheets* (CSS) y *JavaScript*, cuya particularidad es la de adaptar la interfaz del sitio Web al tamaño del dispositivo en que se visualice. Esta técnica de diseño y desarrollo se conoce como *responsive design* (en inglés) o diseño adaptativo (García, 2015). (36)

El beneficio de usar *responsive design* en un sitio Web, es que el sitio Web se adapta de forma automática al dispositivo desde donde se acceda. Lo que se usa con más frecuencia es el uso de Media Queries (consultas), que es un módulo CSS3 que permite la representación de contenido para adaptarse a condiciones como la resolución de la pantalla y si trabajas las dimensiones de tu contenido en porcentajes, puedes tener una Web muy fluida capaz de adaptarse a casi cualquier tamaño de forma automática.

JavaScript se utiliza para crear páginas web dinámicas. Una página *web* dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y

ventanas con mensajes de aviso al usuario. En modo técnico, *JavaScript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con *JavaScript* se pueden probar de forma directa en cualquier navegador sin necesidad de procesos intermedio. (37)

ITextSharp: *ITextSharp* es una biblioteca para crear y manipular archivos PDF, RTF, XML, RTF y HTML entre otros, fue escrita por Bruno *Lwagie*, Paulo *Soares*, entre otros, la misma está disponible sin costo alguno. Al crear un documento con *iTextSharp*, el mismo puede ser exportado en múltiples formatos, o múltiples instancias del mismo formato. Mover páginas de un PDF a otro, rellenar formularios entre otras funciones. Es ideal para crear PDF en aplicaciones de forma dinámica, pasando los distintos parámetros necesarios, adicionar encabezados, títulos, número de páginas, etc. Por lo que es muy útil a la hora de crear documentos que generen tablas, listas, textos. Se puede usar tanto en Java como en .NET. (38)

Lenguaje de Marcas de Hipertexto (HTML)

Se utiliza para organizar el contenido de una página web. Tiene gran accesibilidad y adaptabilidad. Es un lenguaje de programación que describe el formato que tendrá el contenido de un documento. Sirve para la elaboración de páginas web, se define una estructura básica y un código para las mismas. Su última versión permite la reproducción interna de videos, audios y juegos sin necesidad de utilizar programas adicionales. Es compatible con los navegadores web más populares de la actualidad como Internet Explorer, Mozilla Firefox, Safari y *Google Chrome*, además de funcionar de forma correcta con *smartphones* y *tablets*. (39)

CSS es una poderosa herramienta que transforma la presentación de un documento o una colección de documentos, y se ha extendido a casi todos los rincones de la *web* y en muchos entornos que en forma aparente no son web. (40) El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que se le aplica a: (41)

- Un web entero, de modo que se puede definir la forma de todo el web de una sola vez.
- Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, se aplican estilos visibles en un trozo de la página.
- Una etiqueta en concreto, se llega incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en la programación.

1.8. **Programación en Capas**

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario. (42)

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables.

LDAP: El Protocolo de *Acceso Ligero* a Directorio, mejor conocido como *LDAP* (por sus siglas en inglés), está basado en el estándar X.500, pero significativamente más simple y más realmente adaptado para satisfacer las necesidades del usuario. A diferencia de X.500 *LDAP* soporta *TCP*/*IP*, que es necesario para el acceso a Internet. El núcleo de las especificaciones *LDAP* está totalmente definido en las *RFCs* -- una lista completa de las *RFCs*. (43)

¿ Qué hay en un nombre ?

Todas las entradas almacenadas en un directorio *LDAP* tienen un único "*Distinguished Name*," o DN. El DN para cada entrada está compuesto de dos partes: el Nombre Relativo Distinguido (*RDN* por sus siglas en ingles, *Relative Distinguished Name*) y la localización dentro del directorio *LDAP* donde el registro reside. El *RDN* es la porción de tu DN que no está relacionada con la estructura del árbol de directorio. La mayoría de los *items* que almacenas en un directorio *LDAP* tendrá un nombre, y el nombre es almacenado

frecuentemente en el atributo cn (*Common Name*). Puesto que casi todo tiene un nombre, la mayoría de los objetos que almacenarás en *LDAP* utilizarán su valor cn como base para su *RDN*. Si estoy macenando un registro para mi receta favorita de comida de avena, estaré utilizando cn=*ComidaDeAvena Deluxe* como el *RDN* de mi entrada.

Gestor de Base de Datos

Postgres: Este gestor de base de datos seleccionado por el cliente es un proyecto *Open Source*, esto significa que se puede obtener el código fuente , usar y modificar el programa sin las restricciones que comúnmente encuentra con el *software* propietario. Está libremente disponible bajo los términos de la BSD *License*. Existen versiones comerciales de *PostgreSQL* que ofrecen soporte en instalación, configuración y administración. (44) "No hay diferencia práctica entre la versión *Open Source* y la versión Comercial de *PostgreSQL*". Declara el sitio oficial.

- DBMS Objeto-Relacional.
- Altamente Extensible.
- Soporte SQL.
- Integridad Referencial.
- Cliente/Servidor.

1.9. Metodologías de Desarrollo

Scrum es un *framework* de proceso que ha sido utilizado para el trabajo en complejos productos desde los inicios de 1990. (45) Scrum no es un proceso, técnica o método definitivo. Más bien, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. Scrum establece de modo claro la eficiencia relativa de la gestión del producto y las técnicas de trabajo de manera que se puede mejorar de forma continua el producto, el equipo y el entorno de trabajo. Está compuesto por equipos y los roles asociados, eventos, artefactos y reglas. Cada componente dentro del *framework* tiene un propósito específico y es esencial para Scrum y su uso. Scrum plantea un desarrollo iterativo e incremental, esto significa que el producto se va desarrollado en pequeñas partes que se prueban como unidad y también integradas al producto ya desarrollado, esta forma de avanzar permite una fácil adaptación ante los posibles cambios que pueden surgir en el transcurso del proceso. Scrum también plantea la necesidad de involucrar al cliente desde el principio de manera que se puedan detectar posibles errores o cambios deseados. Estas dos características hacen de Scrum una metodología apta para el desarrollo de soluciones cuando no se dispone de parte del cliente de un dominio total del negocio. (45)

Las características más marcadas que se logran notar en Scrum serían: gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos,

productividad y calidad, alineamiento entre cliente y equipo, por último, equipo motivado. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar. (46)

Beneficios de Scrum por los que se seleccionó esta metodología. (47)

- Cumplimento de expectativas: el cliente establece sus expectativas e indica el valor que le aporta cada requerimiento, el equipo técnico los estima y con esta información el dueño del producto establece su prioridad. En las demostraciones de la fase el dueño del producto comprueba que se cumplen de modo efectivo los requerimientos y se brindan sugerencias correspondientes al equipo.
- Flexibilidad a cambios: alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- Resultados anticipados: el cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado el software por completo.
- Mayor calidad del software: la metodología de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a tener un software de calidad superior.
- Mayor productividad: se elimina la burocracia y se motiva al equipo, se da autonomía a las personas para auto-organizarse.
- Maximiza el retorno de la inversión: producción de software con solo las prestaciones que aportan mayor valor al negocio.
- Predicciones de tiempos: mediante esta metodología se conoce la velocidad media del equipo por iteración, conocidos como puntos historia, con lo que es posible estimar de forma fácil cuando se dispondrá de una determinada funcionalidad que todavía está en la lista de requerimientos.
    - Reducción de riesgos: se reducen riesgos al conocer el tiempo medio que tarda el equipo de desarrollo y las entregas funcionales de mayor valor al inicio del proyecto.

Las principales razones del uso de un ciclo de desarrollo iterativo e incremental de tipo scrum para la ejecución de este proyecto son:

- Sistema modular. Las características del sistema GDSI permiten desarrollar una base funcional mínima y sobre ella ir incrementando las funcionalidades o modificando el comportamiento o apariencia de las ya implementadas.
- Entregas frecuentes y continuas al cliente de los módulos terminados, de forma que puede disponer de una funcionalidad básica en un tiempo mínimo y a partir de ahí un incremento y mejora continua del sistema.
- Previsible inestabilidad de requisitos.
- Es posible que el sistema incorpore más funcionalidades de las inicialmente identificadas.
- Es posible que durante la ejecución del proyecto se altere el orden en el que se desean recibir los módulos o historias de usuario terminadas.
- Para el cliente resulta difícil precisar cuál será la dimensión completa del sistema, y su crecimiento puede continuarse en el tiempo suspenderse o detenerse.

Conclusiones parciales.

En este Capítulo se han sentado las bases para la elaboración de la propuesta de investigación, el desarrollo del trabajo. Por lo que se llega a las siguientes conclusiones:

- *S*e encontró herramientas informáticas en el campo de acción de la investigación, pero al analizarlas se confirmó que no son las herramientas adecuadas para esta institución por sus características y necesidades particulares.
- El problema debe ser resuelto a partir de la implementación de un primer módulo dedicado a la adecuación y rotación de variedades.
- El ambiente de programación ASP.NET Core 3.0 en el entorno de desarrollo *Visual Studio* .NET 2019, con el lenguaje C# y con el uso de Vue.js permite el desarrollo de aplicaciones Web, 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.

CAPITULO 2: SOLUCIÓN TEÓRICA DEL PROBLEMA CIENTIFICO

Introducción

Se plantean las funcionalidades del sistema, se exponen las etapas del proceso de ingeniería de *software* por las que se transita durante el desarrollo de un proyecto y se detalla el proceso objeto de estudio. Además, abordará sobre estimación de costo de esta aplicación.

2.1. Propuesta de la aplicación

La solución propuesta es una aplicación web de gestión que estará compuesta por 6 módulos fundamentales, destinados a la gestión de la información relacionada con los:

Nomencladores: En este módulo se podrán llenar todos los campos de gestión de la aplicación teniendo acceso solamente los que posean el rol de (Administración).

Medios Informáticos: En este módulo se podrán introducir todos los medios informáticos, así como los componentes internos de una PC, teniendo acceso solamente los que posean el rol de (Administración o Esp. Principal).

Modelos: En este módulo se podrán realizar todos los modelos del PSI informatizados como las incidencias, aperturas y movimientos básico de medios, teniendo acceso solamente los que posean el rol de (Administración, Esp. Principal o Técnico).

Reportes: En este módulo se podrán emitir todos los reportes en cuanto a informaciones solicitadas de la FGR o información almacenada en base de datos, teniendo acceso solamente los que posean el rol de (Administración, Esp. Principal o Reportes).

Auditor: En este módulo se podrán auditar todas las acciones realizadas en la aplicación, teniendo acceso solamente los que posean el rol de (Administración, o Auditor).

Seguridad: En este módulo se podrán asignar roles y permisos a los usuarios, teniendo acceso solamente los que posean el rol de (Administración).

El sistema posee una autentificación mediante LDAP (43), donde solo se permite el acceso a usuarios que pertenezcan al dominio de la FPMZ, asegurando así que sean usuarios reales. La seguridad es basada en roles, donde se predefinen los que a continuación se detallan:

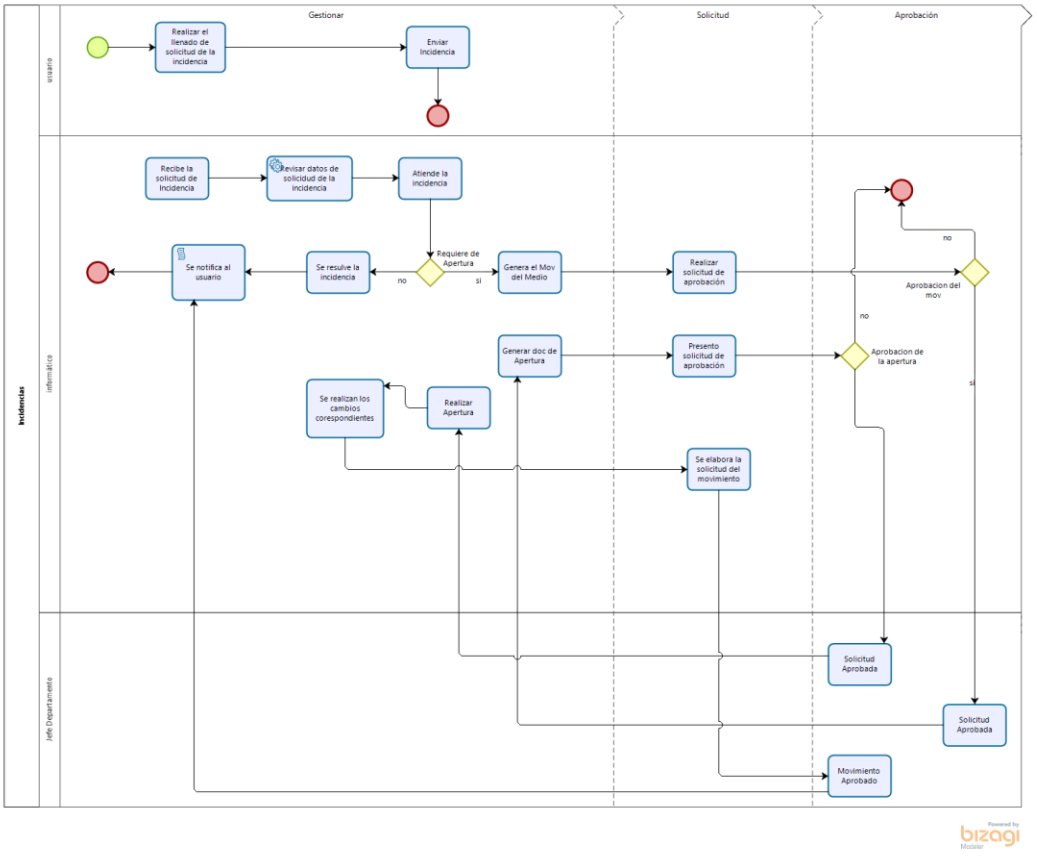
- Administrador. Que permite cambiar cualquier elemento del sistema.

- Gestor de modelos. Que permite realizar aperturas, movimientos, bajas de medios básicos, así como la gestión de incidencias de usuarios, aunque el sistema permite personalizar los permisos de los roles.
- Reportes. Que permite realizar los reportes que brinda el sistema.
- Auditor. Que permite evaluar todos los detalles de seguridad analizando las traza del sistema.

2.2. **Modelo de Proceso**

En la figura 1 se puede observar representado el proceso de creación de una incidencia como punto inicial, donde se recopilan los datos del usuario, área y problema de dicha incidencia. Procediendo a realizar movimientos y aperturas en caso de ser necesario para la ejecución de la incidencia.

Ilustración 1.Diagrama BPMN del Negocio



2.1. **Definición del Equipo**

Los roles principales en Scrum son el Scrum Master, que mantiene los procesos y trabaja de forma similar al director de proyecto, el *Product Owner*, que representa a los *stakeholders* (interesados externos o internos), y el *Team* que incluye a los desarrolladores (46). Los Equipos Scrum son auto organizados y multifuncionales. Los equipos auto organizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad. Los Equipos Scrum entregan productos de forma iterativa e incremental, donde se maximizan las oportunidades de obtener retroalimentación. (45) Las entregas incrementales de producto "Terminado" aseguran que siempre estará disponible una versión útil y funcional del producto.

Tabla 2. Equipo del proyecto

Rol	Miembro
Dueño del producto	Departamento de Informática de FPMZ
Equipo de desarrollo	Yadián Pérez Bernal
Scrum master	Yadián Pérez Bernal

Fuente: Elaboración Propia

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 Página Inicio: Es donde se muestra el software y de los productos que ofrece, además de proporcionar información de contacto y características de las funcionalidades.
- 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.

2.3. **Pila del Producto (*Product Backlog*)**

La Pila del Producto es una lista ordenada de todo lo que se conoce es necesario para el producto. Es la única fuente de requerimientos y cambios para el producto. En ella se describen todas las funcionalidades y características del sistema y puede cambiar a medida que se desarrolla el producto y el cliente valora el progreso con respecto al objetivo deseado, que sería su ideal del producto terminado. Cabe destacar que solo el cliente (*Product Owner*) puede cambiar la pila del producto. (45)

De las entrevistas realizadas se obtuvieron los siguientes requerimientos:

- Gestionar los Nomencladores
- Gestionar los medios informáticos
- Buscar medios informáticos por inventario
- Gestionar modelos del PSI
- Emitir reportes
- Visualizar eventos del sistema
- Autenticación con el dominio de la FPMZ
- Gestionar los permisos de usuarios como gestores y administradores

2.4. **Planeación de las Entregas (*Sprints*)**

En la planeación de las entregas se establecen los plazos dentro de los cuales se debe desarrollar las funcionalidades especificadas. Cada entrega se puede dividir en una serie de historias, que a su vez están compuestas por tareas. Esta fragmentación permite asignar y desarrollar múltiples tareas al mismo tiempo. Las fechas de las entregas pueden variar a medida que aparecen cambios en los componentes de la pila del producto o algún otro tipo de complicaciones en el desarrollo. En la siguiente tabla de muestran las fechas planificadas en que se terminaron las entregas.

2.5. **Descripción de las Entregas**

A continuación, se detallan la composición de las entregas en historias y tareas.

Tabla 3. Composición de las entregas en historias y tareas

Sprint	**Fecha de Inicio**	**Fecha de Fin**
Diseño de la Base de Datos	07 enero 2020	11 enero 2020
Arquitectura inicial	14 enero 2020	4 febrero 2020
Módulo Incidencias	24 marzo 2020	28 marzo 2020
Módulo Medios Informáticos	12 marzo 2020	25 marzo 2020
Módulo de Modelos	5 abril 2020	15 abril 2020
Módulo de reportes	15 abril 2020	20 abril 2020
Módulo de Seguridad	1 de marzo 2020	28 de marzo 2020

Tabla 4. Diseño de la Base de Datos

Historia	Tarea	Nombre
1		**Diseño de la Base de Datos**
	1	Analizar y definir los tipos de datos
	2	Definir las relaciones
	3	Normalizar la base de datos
	4	Generar la base de datos
	5	Cargar datos y realizar pruebas

Tabla 5. Arquitectura inicial

Historia	Tarea	Nombre
1		**Crear la Arquitectura inicial**
	1	Implementar requisitos de cada iteración
	2	Definir la estructura y las funciones de los componentes que forman parte de la arquitectura
	3	Realizar pruebas a la arquitectura validando el diseño contra los requisitos actuales y los que pudieran existir en el futuro

Tabla 6. Módulo Incidencias

Historia	Tarea	Nombre
1		**Crear los *endpoints* en el servidor**
	1	Crear el modelo y los métodos GET
	2	Crear los *endpoint* POST / PUT
	3	Crear el *endpoint* DELETE
2		**Crear las vistas de gestión**
	1	Crear el modelo y el servicio para conectar con el servidor
	2	Crear la vista para listar y eliminar
	3	Crear el formulario para crear / editar
3		**Integrar las categorías (AUF)**

	1	Crear el modelo y servicio para conectar con el servidor
	2	Agregar las categorías al menú de navegación

Tabla 7. Módulo Medios Informáticos

Historia	**Tarea Nombre**	**Historia**
1		**Crear los *endpoints* en el servidor**
2	1	Crear el modelo y los métodos GET
	2	Crear los *endpoint* POST / PUT
	3	Crear el *endpoint* DELETE
		Crear las vistas de gestión
	1	Crear el modelo y el servicio para conectar con el servidor
	2	Crear la vista para listar y eliminar
	3	Crear el formulario para crear / editar
3		**Integrar las categorías (AUF)**
	1	Crear el modelo y servicio para conectar con el servidor
	2	Agregar las categorías al menú de navegación

Tabla 8. Módulo Modelos

Historia	Tarea Nombre	Historia
1		**Crear los *endpoints* en el servidor**
2	1	Crear el modelo y los métodos GET
	2	Crear los *endpoint* POST / PUT
	3	Crear el *endpoint* DELETE
		Crear las vistas de gestión
	1	Crear el modelo y el servicio para conectar con el servidor
	2	Crear la vista para listar y eliminar
	3	Crear el formulario para crear / editar
3		**Integrar las categorías (AUF)**
	1	Crear el modelo y servicio para conectar con el servidor
	2	Agregar las categorías al menú de navegación

Tabla 9. Módulo Reportes

Historia	Tarea Nombre	Historia
1		**Crear los *endpoints* en el servidor**
2	1	Crear el modelo y los métodos GET
	2	Crear los *endpoint* POST / PUT

	3	Crear el *endpoint* Buscar
		Crear las vistas de gestión
	1	Crear el modelo y el servicio para conectar con el servidor
	2	Crear el formulario para buscar y generar
3		**Integrar las categorías (AUF)**
	1	Crear el modelo y servicio para conectar con el servidor
	2	Agregar las categorías al menú de navegación
1		**Crear los *endpoints* en el servidor**

Tabla 10. Módulo Seguridad

Historia	**Tarea** **Nombre**	**Historia**
1		**Crear la configuración en el servidor**
	1	Crear el *endpoint* de autenticación
	2	Crear los servicios para chequear las peticiones y los roles
	3	Crear el servicio para usuarios bloqueados
2		**Implementar la integración con LDAP**
	1	Implementar la función de autenticación con LDAP
	2	Implementar la función de verificar un usuario en LDAP
	3	Modificar el *endpoint* de autenticación para utilizar LDAP

3		**Implementar la autenticación (AG)**
	1	Crear el servicio de autenticación y la vista
	2	Implementar los servicios de autenticación de peticiones y manejo de errores
4		**Implementar la autenticación (AUF)**
	1	Crear el servicio de autenticación y la vista
	2	Implementar los servicios de autenticación de peticiones y manejo de errores

2.6. **Planificación**

La estimación del costo de un software es el proceso de predecir la cantidad de esfuerzo requerido para el desarrollo del sistema y el tiempo para ello (48). Para la estimación del costo se siguió el método de Puntos de Casos de Uso, su resolución se detalla a continuación.

Factor de Peso de los Actores sin ajustar (UAW)

$$\text{UAW} = \Sigma(Cantidad\ Tipo\ Actor_i \cdot Peso_i)$$

$$\text{UAW} = 2 \cdot 3$$ (Dos actores complejos: Técnicos y Usuarios)

$$\text{UAW} = 6$$

Factor de Peso de los Casos de Uso sin ajustar (UUCW)

$$\text{UUCW} = \Sigma(Tipo\ de\ Caso\ de\ Uso_i \cdot Peso_i)$$

$$\text{UUCW} = \big((5 \cdot 5) + (10 \cdot 1)\big)$$ (Ver anexo 1)

$$\text{UUCW} = 35$$

Puntos de Caso de Uso sin ajustar (UUCP)

$$UUCP = UAW + UUCW$$

$$UUCP = 41$$

Factor de Complejidad Técnica (TCF)

$$TCF = 0{,}6 + 0{,}01 \cdot \Sigma(Peso_i \cdot Valor_i)$$ (Ver anexo 2)

$$TCF = 0{,}6 + 0{,}01 \cdot 25{,}5$$

$$TCF = 0{,}855$$

Factor de ambiente (EF)

$$EF = 1{,}4 - 0{,}03 \cdot \Sigma(Peso_i \cdot Valor_i)$$ (Ver anexo 3)

$$EF = 1{,}4 - 0{,}03 \cdot 21{,}5$$

$$EF = 0{,}655$$

Puntos por casos de uso ajustados (UCP)

$$UCP = UUCP \cdot TCF \cdot EF$$

$$UCP = 22{,}96$$

Después de analizar la puntuación de los elementos que afectan los Factores de Ambiente se obtuvo como valor 2, y acorde a la metodología se utiliza el factor de conversión (CF) 20 horas-hombre/Punto de Casos de Uso.

Estimación del esfuerzo para los casos de uso (E)

$$E = UCP \cdot CF$$

$$E = 22{,}96 \cdot 20 \text{ horas} - \text{hombre}$$

$$E = 459{,}2 \text{ horas} - \text{hombre}$$

Para obtener el esfuerzo total del proyecto, se puede realizar un nuevo ajuste que consiste en sumar a la estimación de esfuerzo obtenida por UCP, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo del *software* (Ver anexo 4).

Esfuerzo total (ET)

$$ET = \Sigma\big((UCP \cdot \%)/(CF \cdot 10)\big)$$

$$ET = 2\,548 \text{ horas} - \text{hombre}$$

Tiempo de desarrollo (TD)

$TD = ET/CH$ CH - Cantidad de hombres total (1 u).

$TD = 1\,148 \text{ horas}$

Se consideró que se trabajan ocho horas diarias:

$$TD = \frac{TD}{8\ h - día}$$

$TD = 143{,}5\ días\ (4{,}7\ meses)$

Costo total (CT)

Acorde a las Resoluciones No. 108/05 (Ministerio de Trabajo y Seguridad Social (MTSS), 2005 a) y No. 30/05 (MTSS, 2005 b) referentes a la escala salarial y sus clasificadores, los informáticos clasifican en los grupos del X al XII descritos en el anexo 5. A los tres grupos se les considera con la capacidad técnica para realizar el proyecto, al ser graduados de nivel superior, se estimó la tarifa horaria en base al promedio de sus salarios básicos (358,00 CUP·mes^{-1}), para 168 h mensuales de trabajo.

$CT = ET \cdot CH \cdot TH$ TH – Tarifa horaria (CUP·h^{-1})

$CT = 1\,148\ h \cdot 1 \cdot 2{,}292\ CUP \cdot h^{-1}$

$CT = 2\,631{,}2\ CUP$

2.7. Beneficios tangibles e intangibles

El control de la Seguridad Informática tiene muchos beneficios económicos. La protección apropiada de los medios informáticos contribuye a no tener pérdidas económicas por motivos de sustracción de piezas y medios informáticos, mejora la estructura del empresarial, reduce los riesgos y puede dar como resultado mayores rendimientos. A su vez, una mejor estructura del informática mejora el rendimiento informático (49).

Tangibles

- Reducción cantidad horas/hombre empleadas para la certificación de los medios en 3 días.
- Reducción del espacio físico empleado para almacenar la documentación del PSI.

Intangibles

- Reducción del tiempo promedio para buscar información de medios informáticos, realizar análisis de comportamiento de incidencias, aperturas y movimientos.
- Menor número de errores; letra.
- Generar información más precisa y confiable, que sirva de apoyo a la toma de decisiones.
- Mayor y mejor aprovechamiento de los recursos tecnológicos
- Incremento en la imagen institucional.

2.8. **Análisis de costos y beneficios**

Su uso no solo reportaría ventajas por concepto de una adecuada gestión de la seguridad Informática, sino también un ahorro en tiempo para la entidad ya que se puede gestionar con mayor rapidez y eficiencia el PSI, en la fiscalía para realizar un control del total de los medios se requieren como media de cuatro especialistas para certificar todos los medios de la entidad en una semana; en contraposición, al aplicar la herramienta, solo se requiere de un especialista y 4 hora de trabajo para cumplir la tarea, pudiendo los otros tres especialistas dedicarle su atención a otras materias del departamento.

Conclusiones Parciales

Como corresponde a la metodología de desarrollo Scrum, se plantearon las etapas necesarias para desarrollar el software con la excepción de las pruebas funcionales. Se definió el equipo de trabajo. Se crearon las historias de usuarios, planificadas en cada Sprint. La planificación del proyecto y la estimación de los costos se llevaron a cabo mediante el modelo de Puntos de Función. Los elementos tratados en este capítulo sirvieron para llegar a un acuerdo entre las partes interesadas en el diseño y la estructura de la aplicación a través de la implementación de las funcionalidades que permitan realizar la gestión de la seguridad Informática en la Fiscalía Provincial de Matanzas.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

En este capítulo se presentan los resultados que se obtienen al aplicar las metodologías explicadas en el capítulo 2, así como las pruebas al software realizadas, que permiten conocer el grado de calidad del producto, y de esta forma, comprobar si el sistema es capaz de realizar todas las funcionalidades detallas anteriormente. Se prueban las características más La importantes de la aplicación con el fin de verificar la fiabilidad y calidad de la aplicación como un todo.

3.1. Interfaz de usuario

La interfaz del sistema desarrollado, visualizándose un diseño sencillo, destacando siempre la barra de navegación a la izquierda de forma minimalista, garantizando que el contenido a mostrar ocupe la mayor parte de la pantalla. Predominan los colores claros, generalmente entre el azul claro y el gris claro, según lo indica el manual de Identidad del órgano en estudio, como son la navegación a la izquierda o los botones para crear. La información referente al usuario que está autenticado está en la parte superior derecha de la aplicación, creando así una forma fácil de acceder a ella ya que, aunque el contenido una página determinada pueda exceder el tamaño de la pantalla, la barra superior siempre va a estar visible. En la barra superior también se puede encontrar la barra de búsqueda, esta se puede utilizar para filtrar en las vistas que contienen listas, para buscar el o los elementos deseados.

3.2. Pruebas al software

Las pruebas de software son una parte integral del ciclo de vida de desarrollo del mismo ya que identifican defectos, fallas y errores en la aplicación. Son de naturaleza iterativa e incremental (48). Son una garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. Es el proceso de ejecución de un programa con la intención de descubrir errores. Todo software debe probarse desde dos perspectivas diferentes: al explicar la lógica interna del programa y al comprobar el cumplimiento de los requisitos del sistema. Para lograr este objetivo se llevan a cabo técnicas de diseño de casos de prueba de “caja blanca” y “caja negra” (50).

El proceso de pruebas es el instrumento más adecuado para determinar el status de la calidad de un producto. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos

o si es el software que se quería desarrollar. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de Prueba.

Una estrategia de pruebas integra los métodos de diseño de los casos de prueba para lograr un *software* eficaz. La prueba es un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Una estrategia de pruebas debe incluir tanto pruebas de alto como de bajo nivel. Son parte de la Verificación y Validación incluidas en el aseguramiento de la calidad del *software* (51).

Verificación: Comprobar que el software está de acuerdo con su especificación, donde se debe comprobar que satisface tanto los requerimientos funcionales como los no funcionales.

Validación: El objetivo es asegurar que el *software* satisface las expectativas del cliente.

El plan de pruebas de software se elabora con el fin de especificar qué elementos o componentes se van a probar para que el grupo de trabajo pueda realizar el proceso de Validación y Verificación de los requerimientos funcionales y no funcionales. Además, a través del plan de pruebas se puede continuar con la trazabilidad de los requerimientos, con lo cual el grupo de trabajo, identifica el porcentaje de avance que se ha logrado hasta cierto momento. Al desarrollar el plan de pruebas, se puede obtener información sobre los errores, defectos o fallas que tiene el prototipo, así se realizan las correcciones pertinentes, según el caso y se asegura la calidad del producto que se va a entregar al cliente (50).

En el presente epígrafe, se presentan los procedimientos empleados para la realización de pruebas al sistema una vez concluido su ciclo de desarrollo.

3.3. **Plan de Pruebas**

El plan de pruebas de *software* se elabora con el fin de especificar qué elementos o componentes se van a probar para que el grupo de trabajo pueda realizar el proceso de Validación y Verificación de los requerimientos funcionales y no funcionales. Además, a través del plan de pruebas se puede continuar con la trazabilidad de los requerimientos, con lo cual el grupo de trabajo, identifica el porcentaje de avance que se ha logrado hasta cierto momento.

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

Tabla 11. Plan de Pruebas

No.	Descripción	Tareas Asignadas	Iteración
1	Diseño y Creación de la Base de Datos	*Test* de la Base de datos	
2	Diseño y creación de la interfaz	*Test* de las Interfaces	
3	Gestionar Modelos	*Test* Insertar Modelos *Test* Editar Modelos *Test* Detalles de Modelos *Test* Generar documentos Modelos *Test* Listar Modelos *Test* Listar Modelos sin imprimir	
4	Buscar Modelos	*Test* Buscar Modelos	
6	Gestionar Incidencias	*Test* Insertar Incidencias *Test* Actualizar Incidencias	
8	Imprimir Incidencias y Modelos	*Test* Imprimir Incidencias y Modelos	
9	Reportes	*Test* de Reportes	

3.4. **Pruebas unitarias**

Son las pruebas diseñadas por los programadores y están enfocadas al código, consisten en verificar de manera manual o automatizada, si una parte específica del código, funciona de acuerdo con los requisitos del sistema. Deben ser definidas antes de realizar el código y repetirse hasta eliminar todos los errores para aumentar la calidad del desarrollo (52).

3.5. **Pruebas de Integración**

En las pruebas de integración se examinan las interfaces entre grupos de componentes o subsistemas para asegurar que son llamados cuando es necesario y que los datos o mensajes que se transmiten son los requeridos. El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados de forma unitaria con el fin de comprobar que interactúan de modo correcto a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes (53).

3.6. **Pruebas de seguridad**

Las pruebas de seguridad buscan medir la Confidencialidad, Integridad y Disponibilidad de los datos, desde la perspectiva del aplicativo, es decir, se parte de identificar amenazas y riesgos desde el uso o interface de usuario final. Una vez ejecutadas las pruebas de seguridad es posible medir y cuantificar los riesgos a los cuales se ven expuestos los aplicativos tanto en la infraestructura interna como externa (50).

La realización de las pruebas de seguridad fue con el objetivo de detectar vulnerabilidades, al tomar en cuenta las técnicas de ataque más comunes en los entornos web, por ejemplo: SQL *Injection*, XSS, entre otros (54).

3.7. **Pruebas de carga y estrés**

Las pruebas de carga se realizan de modo general para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. La carga está determinada por el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de acciones durante el tiempo que dura la carga. Esta prueba muestra los tiempos de respuesta de todas las acciones importantes de la aplicación (54).

Por otra parte, las pruebas de estrés se utilizan forma habitual para romper la aplicación. Al doblar el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada (54).

3.8. **Pruebas funcionales o caja negra**

Las pruebas de caja negra comprueban si el software funciona según los requerimientos o especificaciones establecidas. Se conocen de esta forma porque se ejecutan las pruebas sin conocer la lógica interna, sino que se enfoca en las salidas generadas a partir de las entradas seleccionadas y las condiciones de ejecución. Este tipo de pruebas permite conocer si el *software* hace lo que se espera. Las especificaciones funcionales son la fuente de información empleada, por lo que se requiere una especificación detallada de las mismas . Estas se enfocan en buscar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Ilustración 2. Esquema de Prueba de caja

Tabla 12. Caso de prueba de Gestionar Nomencladores

Caso de Uso	Gestión de Nomencladores
Caso de Prueba	Insertar o editar una apertura y dejar los campos vacío
Resultado Esperado	Se envía un mensaje al usuario indicándole que el campo requiere datos
Resultado Obtenido	Se produce un error y se emite una advertencia
% Cumplimiento	100 %

Tabla 13. Caso de prueba de Gestionar Incidencias

Caso de Uso	Gestionar Incidencias
Caso de Prueba	Insertar o editar una incidencia y dejar el campo del problema de la incidencia vacío.
Resultado Esperado	Se le envía un mensaje al usuario indicándole que el campo del problema de la incidencia requiere datos.
Resultado Obtenido	Se produce un error y se levanta una advertencia
% Cumplimiento	100 %

Tabla 14. Caso de prueba de Gestionar Aperturas

Caso de Uso	Gestionar Apertura
Caso de Prueba	Insertar o editar una apertura y dejar los campos vacío
Resultado Esperado	Se le envía un mensaje al usuario indicándole que los campos requieren datos
Resultado Obtenido	Se produce un error y se levanta una advertencia
% Cumplimiento	100 %

Tabla 15. Caso de prueba de Gestionar Movimientos

Caso de Uso	Gestionar Movimientos
Caso de Prueba	Insertar o editar un movimiento y dejar los campos vacío
Resultado Esperado	Se le envía un mensaje al usuario indicándole que el campo requiere datos
Resultado Obtenido	Se produce un error y se levanta una advertencia
% Cumplimiento	100 %

Tabla 16. Caso de prueba de Aprobación de Modelos

Caso de Uso	Aprobación de Modelos
Caso de Prueba	Aprobación o Denegación de modelos si se deja son activar la aprobación
Resultado Esperado	Se envia un mensaje al usuario indicándole que el campo requiere ser activado.

Resultado Obtenido	Se produce un error y se levanta una advertencia
% Cumplimiento	100 %

3.9. **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 de modo general son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción. Esta termina de definir el nivel de calidad de la aplicación (32).Las pruebas de aceptación se llevarán a cabo mediante la redacción de los casos de prueba, al tomar en cuenta el orden de las HU y la prioridad que ha sido asignada a las funcionalidades. Luego se hará la planificación con el cliente de cuándo y cuáles pruebas serán llevadas a cabo, para así reunir los miembros del proyecto seleccionados para realizarlas. Al final, se completarán cada uno de los campos de las tablas de las pruebas de aceptación con el resultado de la prueba. Después de haber superado las pruebas de aceptación podrá considerarse que la aplicación es apta para el uso y despliegue dentro del proyecto (55).

Se realizó estas pruebas para cada uno de los requisitos funcionales del software y a continuación se presentan los resultados obtenidos:

Campos descritos:

Código: Sirve para identificar la prueba realizada y al mismo tiempo sugiere su nombre.

HU: Nombre de la Historia de Usuario a la que hace referencia

Nombre: Nombre de la prueba

Descripción: Describe la funcionalidad que se desea probar

Condiciones de Ejecución: Condiciones a cumplirse para poder llevar a cabo el caso de prueba

Entradas / Pasos de ejecución: Descripción de cada uno de los pasos durante el desarrollo de la prueba

Resultado esperado: Descripción del resultado que se espera obtener con la prueba realizada

Evaluación: Evaluación emitida acorde al resultado de la prueba realizada **Satisfactorio:**

El resultado es completamente el esperado por el usuario

Parcialmente Bien: El resultado no es completamente el esperado por el cliente o usuario de la aplicación

Mal: El resultado de la prueba genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contextos.

Pasos a seguir para realizar pruebas de aceptación:

- Redacción de los casos de prueba teniendo en cuenta el orden de las HU y los niveles de prioridad dados anteriormente a las funcionalidades.
- Planificación con el cliente de cuales pruebas se llevarán a cabo y en qué momento.
- Completar cada uno de los campos de las tablas de las pruebas de aceptación con el resultado de la prueba.

Ilustración 3: Prototipo del formulario de Autentificación

Tabla 17. Prueba de autenticación

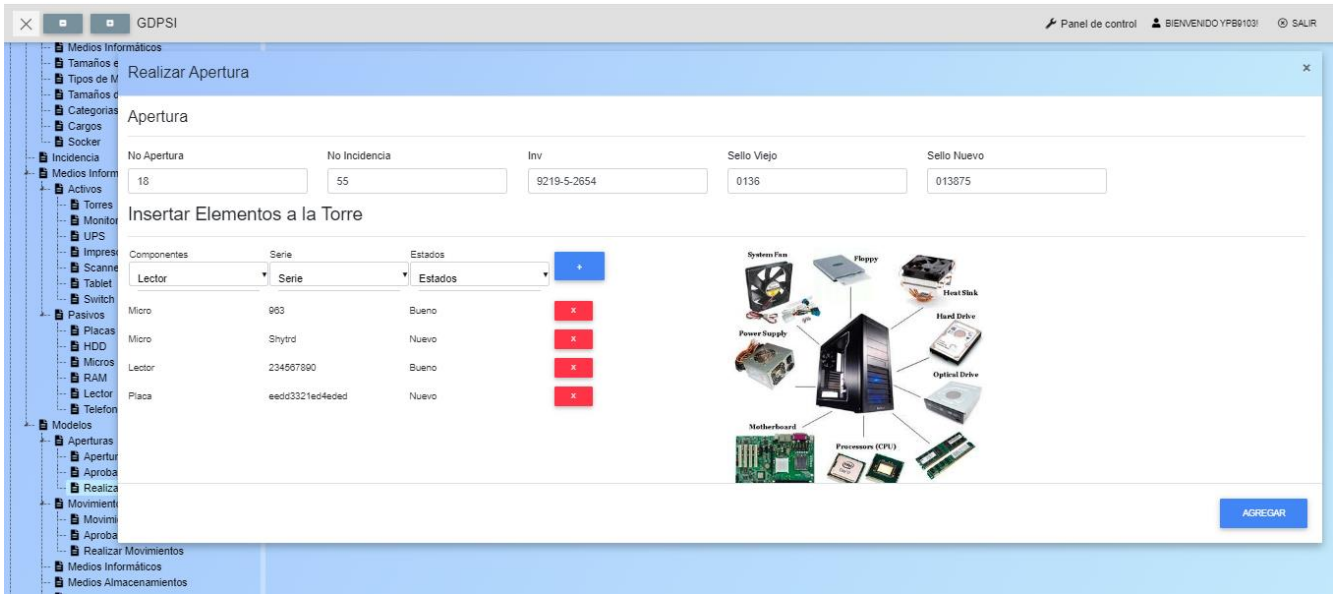
Prueba de Aceptación	
Número del caso de prueba: 1	**Número de Historia de Usuario: 1**
Nombre del Caso de Prueba: Autenticarse	
Descripción: Comprobar que el sistema permite el acceso al sitio web solo a los usuarios que estén registrados en el LDAP de la Entidad. El sistema debe verificar que los datos obligatorios para entrar al sistema sean correctos y no estén vacíos. En estos casos se muestra un mensaje de error impidiendo el acceso al sitio hasta que los datos cumplan con sus requerimientos.	
Condiciones de ejecución: Que el sistema esté conectado con el LDAP, comprobando además el rol que tenga asignado este usuario	
Entrada / Pasos de ejecución: El usuario introduce su nombre de usuario y su contraseña y pulsa el botón entrar.	
Resultado esperado: El sistema debe aceptar y entrar al sistema en el caso de que el nombre de usuario y la contraseña introducidos coincida con el LDAP de la entidad, de intentar introducir un valor erróneo o nulo deberá mostrar una alerta.	
Evaluación de la prueba: Satisfactorio	

Tabla 18. Prueba Generar Aperturas

Prueba de Aceptación	
Número del caso de prueba: 2	**Número de Historia de Usuario: 2**
Nombre del Caso de Prueba: Generar Apertura	

Descripción: El sistema debe mostrar los datos de la pc que se desee abrir y comprobar que esta apertura tenga asignada una incidencia. En estos casos se muestra un mensaje de error impidiendo el acceso al sitio hasta que los datos cumplan con sus requerimientos.
Condiciones de ejecución: Que el sistema esté conectado con BD, comprobando además el rol que tenga asignado este usuario. Esta Apertura debe estar Aprobada.
Entrada / Pasos de ejecución: El usuario introduce los datos correspondientes y da en el botón generar. .
Resultado esperado: El sistema comprobara el rol del usuario si tiene permisos a este módulo y validara los datos del formulario, de intentar introducir un valor erróneo o nulo deberá mostrar una alerta.
Evaluación de la prueba: Satisfactorio

Ilustración 4. Prueba Generar Apertura



3.10. **Análisis de los resultados obtenidos**

Después de desarrollar todo un proceso de pruebas se logró resultados satisfactorios, pues tras la detección de diferentes errores, obtenidos fundamentalmente con las pruebas de caja negra, se solucionaron varios problemas que impedían el cumplimiento de los requisitos fundamentales del sistema. Las pruebas relevaron algunos errores, principalmente en validaciones y entradas de datos, permitiendo así corregirlos para un mejor funcionamiento del sistema, por lo que se pudo afirmar que las pruebas son satisfactorias. Además, se logró obtener un *software,* cuyas funciones se encuentra, en correspondencia con las especificaciones acordadas y que además cumple con los requerimientos funcionales, de comportamiento y de rendimiento.

Conclusión Parciales

Al terminar este capítulo se puede concluir que la planificación presentada en el capítulo 2 fue acertada, pues permitió el desarrollo del sistema según el cronograma y se cumplió con las exigencias del cliente al iniciar esta investigación. Además, las pruebas realizadas mediante el uso de las técnicas descritas con anterioridad fueron de gran importancia para demostrar el buen funcionamiento del software y el cumplimiento de los requerimientos del cliente.

Como resultado final se obtiene una aplicación Web con una apariencia atractiva y fácil de usar, con todas las funcionalidades requeridas, y que satisface todas las expectativas del cliente.

Conclusiones

Como resultado de esta investigación se dio cumplimiento a los objetivos trazados con lo que se arribó a las siguientes conclusiones:

- El estudio realizado sobre los antecedentes, el estado actual de la temática, la bibliografía y 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 utilizó herramientas de software adecuadas para dar solución al problema detectado.
- Se realizó la estimación del costo de implementación del sistema y el estudio de factibilidad, lo que arrojó como resultado la factibilidad de la realización del sistema informático.
- La realización de pruebas aplicadas al sistema permitió la detección de errores y la pronta corrección de los mismos.
- La implementación del sistema y la aplicación de las pruebas de validación con resultados satisfactorios, logró demostrar que el software elaborado cumple con los requerimientos especificados por el cliente, y que posee un nivel de calidad y estabilidad suficientes para la explotación del mismo.

Recomendaciones

- Continuar el desarrollo de las funcionalidades que garanticen la integralidad para seguridad informática y toma de decisiones en la FPMZ.
- Implementar un agente para instalarlo en cada PC y automatizar el control de los componentes y supervisar los cambios de *hardware* y administrarlos desde nuestra aplicación.
- Implementar a modo de prueba en un grupo de entidades de la FPMZ de muestra para su validación.

REFERENCIAS BIBLIOGRÁFICAS

1. **Richardson, Robert.** *CSI Computer Crime & Security Survey.* s.l. : CSI, 2011.

2. **Study, Benchmark.** *Second Annual Cost of Cyber Crime Study.* s.l. : U.S. Companies, 2011.

3. **Sophos.** Sophos security threat report. [En línea] [Citado el: 17 de 4 de 2018.] http://www.sophos.com/en-us/security-news-trends/security-trends/security-threatreport-2018.aspx.

4. **Kaspersky Lab.** Kaspersky Security Bulletin. Statistics. [En línea] [Citado el: 17 de 4 de 2018.] http://www.securelist.com/en/analysis/204792216/Kaspersky_Security_Bulletin_Statisti cs_2018.

5. **s21sec.** [En línea] [Citado el: 17 de 3 de 2017.] http://www.s21sec.com/descargas/Informe%20Vulnerabilidades%202011.pdf..

6. **Secunia.** Secunia yearly report 2017. [En línea] [Citado el: 21 de 3 de 2018.] http://secunia.com/?action=fetch&filename=Secunia_Yearly_Report_2017.pdf.

7. **Nitesh Dhanjani, Billy Rios & Brett Hardin.** *Hacking: The next generation.* s.l. : Sebastopol (CA), 2009.

8. **Rodrigo Werlinger, Kirstie Hawkey and Konstantin Beznosov.** *An integrated view of human, organizational, and technological challenges of IT security management.* Canada : University of British Columbia, 2008.

9. **Garcias, Geraldo.** *Principales deficiencias en las redes nacionales.* Cuba : s.n., 2015.

10. **UCI.** Gestor de Recursos de Harware y Software. [En línea] [Citado el: 4 de 5 de 2019.] https://www.uci.cu.

11. **GLPI.** Aplicación de código abierto para gestionar el inventario. [En línea] [Citado el: 15 de 6 de 2018.] https://www.tecnologiapyme.com/.

12. **ASSETS.** Sistema de Gestión Integral. [En línea] [Citado el: 4 de 5 de 2016.] https://www.assets.co.cu.

13. **SGSI.** Sistema de Gestión de Seguridad de la Información. [En línea] [Citado el: 21 de 11 de 2019.] http://firma-e.com.

14. **Elda, Gisel.** ABC INVENTORY SOFTWARE. [En línea] [Citado el: 21 de 11 de 2018.] https://prezi.com/dkjay5-smjgk/abc-inventory-software/.

15. **Juventud Revelde.** Juventud Revelde. [En línea] [Citado el: 24 de 4 de 2020.] http://www.juventudrebelde.cu/suplementos/informatica/2019-07-03/informatizacion-ordenada-cuba-emite-diez-nuevas-normativas-juridicas.

16. **Comunicaciones, Ministerio de.** Gaceta Oficial. [En línea] 4 de 7 de 2019. [Citado el: 14 de 5 de 2020.] https://www.gacetaoficial.gob.cu/es/gaceta-oficial-no-45-ordinaria-de-2019.
17. **ISO/IEC.** ISO/IEC 27001. [En línea] [Citado el: 25 de 4 de 2020.] https://iso.org.
18. —. ISO/IEC 27002. [En línea] [Citado el: 25 de 4 de 2020.] https://iso.org.
19. **NIST.** National Institute of Standards and Tecnology. [En línea] [Citado el: 26 de 4 de 2020.] https://medium.com.
20. **ISP.** Standard of Good Pratice for Information Security 2018. [En línea] [Citado el: 27 de 4 de 2020.] https://www.securityforum.org.
21. **Solms, Bevon.** *Information Security – The Fourth Wave.* s.l. : Computers & Security, 2012.
22. **Montesino, Raydel.** *Information security automation: how far can we go?* Cuba : s.n., 2011.
23. **Cano, Jaime.** *Un concepto extendido de la mente segura.* s.l. : Criptored, 2010.
24. **PCI SSC.** *Payment Card Industry Data Security Standard.* 2012.
25. **HHS.** *Health Insurance Portability and Accountability.* s.l. : HHS, 2015.
26. **Fovino, Manuel Masera and I. N.** *Security metrics for cyber security assessment and testing.* s.l. : Centre of the European Commission, 2015.
27. **Jaquith, Antonio.** *Security metrics: replacing fear.* s.l. : Addison-Wesley, 2017.
28. **Hayden, Laida.** *IT security metrics a practical framework for measuring security & protecting data.* New York : McGraw Hill, 2014.
29. **ISO/IEC, "ISO/IEC 27004: Information technology - Security techniques - Information security management systems – Measurements.** *International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).* 2009.
30. **Barrios, Eduardo.** DEV. [En línea] 10 de Diciembre de 2019. [Citado el: 4 de Febrero de 2020.] https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0-5goj.
31. **Roth, Daniel , Anderson, Rick y Luttin, Shaun.** Microsoft. *Introducción a ASP.NET Core.* [En línea] Microsoft, 11 de Noviembre de 2019. [Citado el: 21 de Enero de 2020.] https://docs.microsoft.com/es-es/aspnet/core/?view=aspnetcore-3.1.
32. **Sotolongo Alonso, Keythy .** *Sistema informático para la tramitación de documentos en la Secretaría General de la Universidad de Matanzas.* Facultad de Ciencias Técnicas. Departamento de Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2018. Tesis de Grado.

33. **Mesa Pedroso , Keila .** *Sistema Web para facilitar el desarrollo de funcionalidades de la Federación Estudiantil Universitaria (FEU) y la Unión de Jóvenes Comunistas (UJC) en la sede de la Universidad de Matanzas, Camilo Cienfuegos.* Facultad de Ciencias Técnicas. Departamento de Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
34. **Shahzad, Khuram.** C# Corner. [En línea] 24 de Octubre de 2018. [Citado el: 31 de Diciembre de 2019.] https://www.c-sharpcorner.com/article/command-mediator-pattern-in-asp-net-core-using-mediatr2/.
35. **Carlos, Universidad Rey San Juan.** C#. Qué es y para qué se utiliza. [En línea] [Citado el: 17 de 4 de 2020.] https://negociosyestrategia.com/blog/que-es-csharp/.
36. **Carpenter, J y Bithell, J.** *Bootstrap confidence intervals: when, which, what?. A practicalguide for medical statisticians .* s.l. : Statistics in medicine, 2015.
37. **Pérez, J.** *Introducción a JavaScript.* 2018.
38. **Platform, Open-Source and Business Software.** SourceForge. [En línea] [Citado el: 17 de 6 de 2020.] https://sourceforge.net/articules/itextsharp/.
39. **Galileo.** Galileo. [En línea] 4 de Mayo de 2018. [Citado el: 2019 de Enero de 6.] http://elearningmasters.galileo.edu/2018/05/04/html5-en-cursos-virtuales/.
40. **Martos, M.** *Manual de CSS.* 2018.
41. **Meyer , Eric A. y Weyl, Estelle .** *CSS: The De€nitive Guide.* Fourth Edition. Sebastopol : O'Reilly Media, Inc., 2018. 978-1-449-39319-9.
42. *Programación en Capas.* **Santiago Domingo Moquillaza, Hugo vega Huerta.** 7, s.l. : Universidad San Marcos, 2010, Investigacion de Sistemas Informáticos, Vol. 2, págs. 57-67.
43. **Donnelly, Michael.** *Introducion a LDAP.* 2012.
44. **PostgreSQL.** *The PostgreSQL Global Development Group.* s.l. : Copyright, 2019.
45. **Schwaber, Ken y Sutherland, Jeff.** *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.* s.l. : Org and ScrumInc, 2016.
46. *Integrando el Scrum a la planIfIcacIón de proyectoS por cadena crítIca.* **González Sajiúm, Derby y Marcano De La Rosa, José.** 2, julio-diciembre de 2019, Ciencia, Ingenierías y Aplicaciones, Vol. 2, págs. 81-130. 2636-2171 .
47. **Paniagua González, Pablo César.** *Estudio sobre la administración efectiva de proyectos de software, utilizando la metodología scrum, dirigido a empresas de sistemas.* FACULTAD DE CIENCIAS QUÍMICAS Y FARMACIA , UNIVERSIDAD DE SAN CARLOS DE GUATEMALA. San Carlos : Universidad de San Carlos de Guatemala, 2017. Tesis de Mestria.

48. **Yanes De la Cruz , Alejandro .** *Activate, sistema para la difusion de eventos.* Facultad de Ciencias Técnicas. Departamento de Informática, Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
49. *Web based Crop Rotation Recommendation in Agrarian Society.* **A. Sudha , Christina , M., Rajalakshmi y K. , Sathya .** 17, 2017, International Journal of Engineering Research & Technology (IJERT), Vol. 5, págs. 1-5. 2278-0181.
50. **Scull Echeverría , Lizlaine.** *Sistema Informático para la gestión de los procesos de Órdenes de Trabajos en la Asociación Económica Internacional Aguas Varadero.* Facultad de Ciencias Técnicas. Departamento Ingeniería Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2018. Tesis de Grado.
51. **Padrón González , Andy.** *Aplicación web para la solicitud de ariticulos para los clientes de LaBiofam en Matanzas.* Facultad de ciencias Técnicas. Departamento de Informática, Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
52. **Rico Rodríguez, Frank David .** *Videojuego serio para ejercitar el inglés.* Facultad de Ciencias Técnicas. Departamento de Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
53. **Gilo Carrasco, Lilianet de Lourdes.** *Sistema informático para la tramitación de las consultas médicas en el Hospital Provincial Faustino Pérez de Matanzas.* Facultad de Ciencias Técnicas. Departamento de Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2018. Tesis de Grado.
54. **Lara Sobrino, John de Jesús.** *Aplicación web para la gestión de solicitudes de almacén en la Dirección Provincial de Bufetes Colectivo Matanzas.* Facultad de CienciasTécnicas. Departamento de Informática, Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
55. **Peña Daniel, Luis Alberto.** *Sistema para la gestión del plan de capacitación de la empresa de perforación y reparación capital de pozos de petróleo y gas.* Facultad de Ciencias Técnicas. Departamento de Informática , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.
56. **Microsoft.** *Microsoft SQL Server 2019 Technical white paper.* s.l. : Microsoft Corporation, 2018.
57. **Arias Herrera , Jairo Fernando .** *Adaptación de la metodología de desarrollo de software scrum a equipos de un solo programador. Caso: departamento de sistemas de la cooperativa de ahorro y crédito*

andina ITDA. UNIVERSIDAD TÉCNICA DE COTOPAXI. Latacunga : Universidad Técnica De Cotopaxi, 2016. Tesis de Grado.

58. *Informatización y dirección de empresas en Cuba: evolución y desafíos.* **Blanco, L.** 1, 2017, COFIN, Vol. 11, págs. 1-13.

59. **Blanco González, Lorna Ylenia.** *Sistema web para la gestión de los Programas de Formación Doctoral.* Facultad de Ciencias Técnicas , Universidad de Matanzas. Matanzas : Universidad de Matanzas, 2019. Tesis de Grado.

60. **Carrión Abollaneda, Victor Hugo.** *Desarrollo de una aplicación web basada en el modelo vista controlador para la gestión de las historias clínicas de los pacientes en el centro de salud de San Jerónimo.* FACULTAD DE INGENIERÍA , UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS. Andahuaylas : Universidad Nacional José María Arguedas, 2015. Tesis de Grado.

61. *Resultados preliminares más significativos tras cuatro años de aplicación de la metodología SCRUM en las prácticas de laboratorio.* **Castillo Vidal, Luis .** 1, Enero de 2018, ReVisión, Vol. 11, págs. 53-64. 1989-1199.

62. **Fotache, M. y Strimbei, C.** *SQL and data analysis. Some implications for data analysits and higher education.* s.l. : ELSEVIER, 2016.

63. **Gamma, Erich .** *Visual Studio Code Tips & Tricks.* Zurich : Microsoft Deutschland GmbH, 2016.

64. **Hernández, R., Fernández, C. y Baptista, M.** *Metodología de la investigación.* Quinta. Distrito Federal de México : McGraw-Hill-Interamericana, 2010. pág. 613 .

65. **Mendoza, D y Baquero, L.** *Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso. .* La Habana : s.n., 2016.

66. *Dinámica ambiental y económica en la localidad de Puente Aranda en Bogotá.* **Parra, C. y Muñoz, J.** 2, 2014, Bioética, Vol. 14, págs. 28-37.

67. *Impacto de las tecnologías de la información y la comunicación en la agricultura.* **Pérez, A., Milla, M. y Mesa, M.** 1, 2006, Cultivos Tropicales, Vol. 27, págs. 11 - 17.

68. *Impacto del uso de patrones de diseño en la industria del software en Costa Rica.* **Picado Corao , Francisco y Pérez Vanegas, Mariana .** 6, Julio – Diciembre de 2019, Tecnología Vital, Vol. 2, págs. 44-51.

ANEXOS

Anexo 1. Factor de peso de casos de uso sin ajustar.

CU	Peso
Acceso al sistema	5
Administrar Incidencia	5
Administrar Reportes	5
Administrar Modelos	5
Administrar Nomencladores	5
Diseñar Medios para Bajas	5
Determinar Equipos a Reparar	5
	35

Fuente: Elaboración Propia.

Anexo 2. Factor de complejidad técnica.

Factor	Descripción	Peso	Influencia	Resultado	Comentario
F1	Sistema distribuido	2	0	0	El sistema es centralizado
F2	Tiempo de respuesta y desempeño	1	1	1	La velocidad es limitada por las entradas provistas por el usuario
F3	Eficiencia respecto al usuario final	1	1	1	Escasas restricciones de eficiencia
F4	Procesamiento interno complejo	1	1	1	No hay cálculos complejos
F5	Código reutilizable en otras aplicaciones	1	2	2	Existen microservicios que pueden ser reutilizables
F6	Facilidad en la instalación	0,5	0	0	No requiere instalación
F7	Usabilidad	0,5	5	2,5	No requiere conocimiento especializado
F8	Portabilidad	2	0	0	No se requiere que el sistema sea portable al ser web
F9	Facilidad de mantener	1	2	2	Se requiere un costo medio a bajo de mantenimiento
F10	Accesos simultáneos	1	5	5	Alta probabilidad de múltiple concurrencia al ser web
F11	Incluye objetos especiales de seguridad	1	3	3	Seguridad normal
F12	Provee acceso directo a terceros	1	5	5	Los usuarios web tiene acceso directo
F13	Se requiere facilidades especiales de entrenamiento a usuarios	1	3	3	Sistema intuitivo y fácil de usar, pero no requiere de conocimiento especializado
				25,5	

Anexo 3. Factor de ambiente.

Factor	Descripción	Peso	Valor	Resultado
E1	Familiaridad con el modelo de proyecto utilizado	1,5	3	4,5
E2	Experiencia en la aplicación	0,5	2	1
E3	Experiencia en orientación a objetos	1	3	3
E4	Capacidad del analista líder	0,5	2	1
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	5	10
E7	Personal part-time	-1	0	0
E8	Dificultad del lenguaje de programación	-1	3	-3
				21,5

Fuente: Elaboración Propia.

Anexo 4. Esfuerzo total.

Actividad	Porcentaje (%)	ET
Análisis	10	114,8
Diseño	20	229,6
Programación	40	459,2
Pruebas	15	172,2
Sobrecarga (otras actividades)	15	172,2
Total	**100**	**1 148**

Fuente: Elaboración Propia.

Anexo 5. Escala salarial de informáticos.

Grupo	Descripción	Requerimiento	Nivel de utilización	Salario (Peso Cubano (CUP))
X	Especialistas en Ciencias Informáticas C	Graduados de nivel superior	En todas las entidades	325,00
XI	Especialistas en Ciencias Informáticas B	Graduados de nivel superior	Uniones, grupos empresariales, empresas con categoría I y II, unidades presupuestadas de similar complejidad	365,00
XII	Especialistas en Ciencias Informáticas A	Graduados de nivel superior	Organismos	385,00
Promedio				358,00

Fuente: Elaboración Propia.

Anexo 6 Diagrama Entidad-Relación