UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática

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

Autores:

Daniel Pelaez Garcia

Antonio Veliz Santos

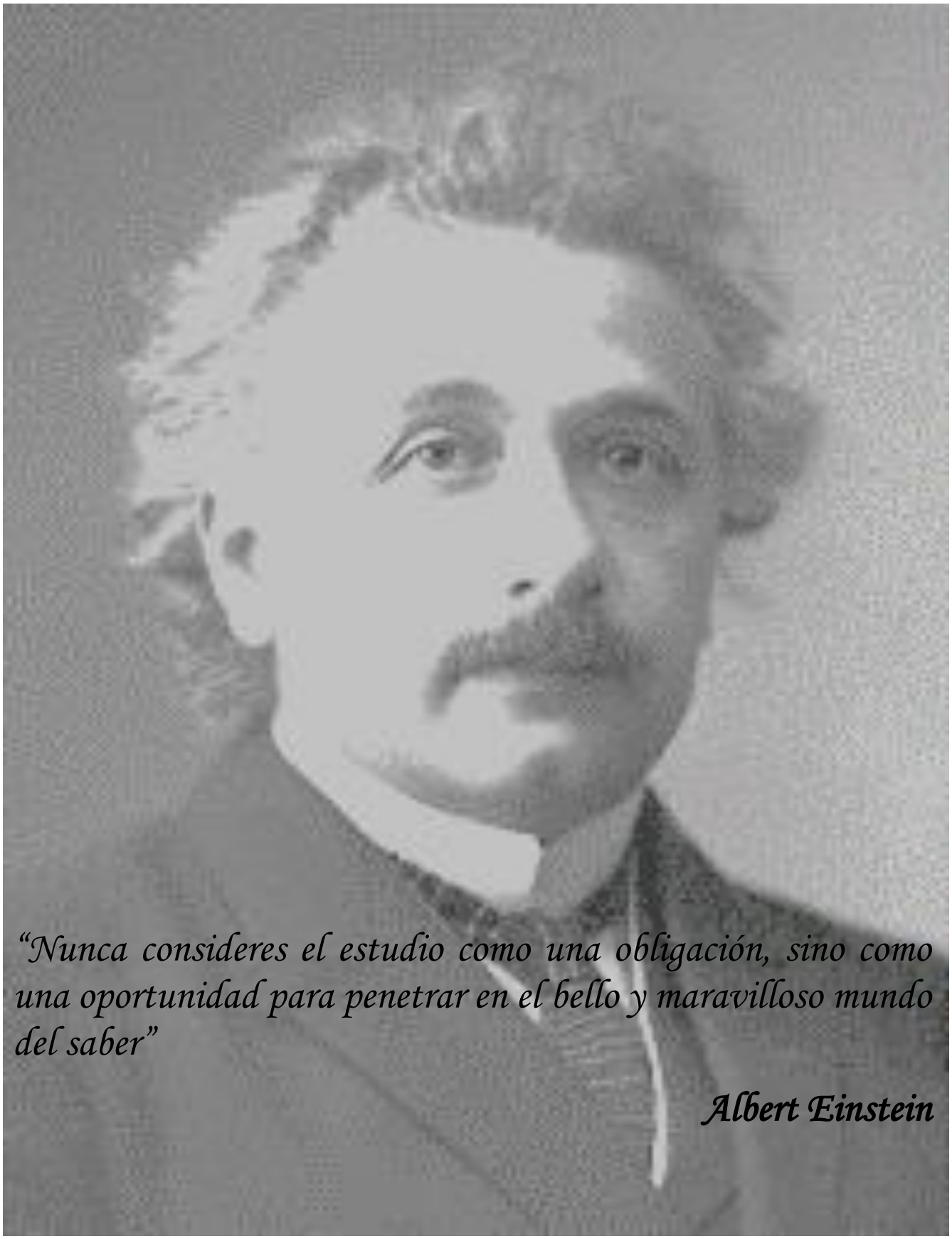
Tutores:

Ing. Delis Ise Morales

Ing. Luis Eduardo Gallardo Concepción

“La Habana, junio 2015”

“Año 57 de la Revolución”



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

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

_____	_____
Firma del autor	*Firma del autor*
Daniel Pelaez Garcia	*Antonio Veliz Santos*
_____	_____
Firma del tutor	*Firma del tutor*
Delis Ise Morales	Luis Eduardo Gallardo Concepción

DATOS DE CONTACTO

Ing. Antonio Veliz Santos

Correo electrónico: aveliz@estudiantes.uci.cu

Sibanicú, Camagüey, Cuba

Daniel Pelaez Garcia

Correo electrónico: dpelaez@estudiantes.uci.cu

Sancti Spíritus, Cuba

Ing. Delis Ise Morales

Correo electrónico: dise@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Ing. Luis Eduardo Gallardo Concepción

Correo electrónico: legallardo@uci.cu

La Habana, Cuba

AGRADECIMIENTOS DE DANIEL

A mi familia, por su apoyo y confianza, en especial a mis padres que lo han dado todo por sus hijos.

A mi compañero de tesis, por estar siempre a mi lado en los buenos y malos momentos de la carrera.

A mis tutores Delis y Gallardo por guiarme en la realización de mi tesis y aconsejarnos en los momentos más difíciles.

A mi profesora guía Yaniselis por ser como una madre para mí, por darme su apoyo y su cariño cuando más lo he necesitado.

A todos los profesores que contribuyeron a mi formación académica.

A todos mis amigos, por los momentos inolvidables que pasamos juntos, por sus momentos de dedicación.

A todos aquellos que de una forma u otra me ayudaron en el desarrollo de este trabajo.

AGRADECIMIENTOS DE ANTONIO

A mamá que aunque no esté físicamente hoy conmigo me acompaña y me protege en todo momento, me cuida y me guía.

A mi papá por ayudarme y siempre estar a mi lado en los momentos que más lo he necesitado.

A mis abuelas Aida y Vidalina a mi tía Aidita por haberse convertido en mis madres.

A mi madrasta Lourdes por todo el apoyo y cariño que me ha dado en el transcurso de la carrera.

A mi abuelo Rogelio que es mi segundo padre y a mi tíos Rider y Elvis

A toda la familia en general.

A mi primos Maydelín y Alejandro por ser los hermanos que no tuve.

A mis tutores Delis y Gallardo doy gracias por toda la ayuda y conocimientos brindados.

A todos los profesores que han colaborado con mi formación.

A cada uno de mis compañeros y compañeras de aula, apartamento y proyecto.

A todos aquellos que de una forma u otra me ayudaron en el desarrollo de este trabajo.

DEDICATORIA DE DANIEL

A mi mamá y papá por el amor y la confianza depositado en mí. Por apoyarme en todos los momentos difíciles de mi vida y por ayudarme a hacer realidad todos mis sueños.

A mi hermana, por estar a mi lado en todo momento, por acompañarme en los momentos difíciles y aconsejarme sobre las malas decisiones que he tomado en mi vida. No me alcanzan las palabras para agradecerte cada momento de mi vida.

A mi hermano por apoyarme siempre y ayudarme a sentirme responsable y seguro de mis decisiones. Por sentirse siempre orgulloso de mí y sobre todo por su cariño.

DEDICATORIA DE ANTONIO

A mis padres, especialmente a mi madre por haberse sacrificado tanto para que yo pudiese llegar hasta aquí y a mi familia en general.

RESUMEN

La realización de auditorías a sistemas informáticos ha tomado auge en la actual era tecnológica. Estas se realizan con el objetivo de identificar las vulnerabilidades que poseen dichos sistemas y por consiguiente, minimizar el riesgo de ataques que atentan contra su seguridad. La Plataforma de Seguridad en Tecnologías de la Información (Xilema-PlatSI) es una solución desarrollada con el objetivo de detectar brechas de seguridad, específicamente a aplicaciones web. Los especialistas que interactúan con dicha plataforma son los encargados de analizar las vulnerabilidades detectadas y comprobar su veracidad. Para realizar dicho análisis, se basan en conocimiento propio e información que proporcionan las bases de datos de vulnerabilidades. Esta búsqueda de información constituye un proceso engorroso, que consume mucho tiempo para los especialistas, debido a la dificultad que trae para los especialistas organizar y obtener dicha información, la cual es necesaria para verificar si las vulnerabilidades representan falsos positivos. Con el objetivo de dar solución a este problema, fue desarrollado el Sistema de Gestión de Vulnerabilidades de Seguridad Informática, el cual permite la gestión de vulnerabilidades de seguridad informática como su nombre lo indica, además de la gestión de categorías y herramientas asociadas a las mismas. Entre los sistemas similares analizados, se seleccionó la Base de Datos de vulnerabilidades: “Vulnerabilidades y Riesgos Comunes (CVE) con el objetivo de poblar la base de datos del sistema. Se seleccionaron además, las tecnologías, metodología, herramientas y lenguajes, en los que se basó el desarrollo de la aplicación. La utilización de la misma, constituye un sistema de apoyo para los especialistas de Xilema-PlatSI en la etapa de análisis de vulnerabilidades.

Palabras Clave: Base de Datos de vulnerabilidades, vulnerabilidades, Xilema-PlatSI.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), ha aumentado considerablemente la automatización de muchos de los procesos de la sociedad, mediante el desarrollo de aplicaciones informáticas. Esto ha traído consigo un aumento de productos informáticos portadores de vulnerabilidades que pueden atentar contra la seguridad de sus datos (1). Por esta razón, la industria de software dirige sus esfuerzos al desarrollo de productos informáticos, que garanticen los principios de confidencialidad, integridad y disponibilidad de la información de la seguridad informática[1].

La seguridad necesita ser un componente íntegro de cada fase de la vida de una aplicación y no un complemento de software (2). De esta manera se pueden aplicar buenas prácticas de seguridad por cada una de las fases por las que transcurre el software, principalmente en la Fase de Implementación, para obtener un sistema con la menor cantidad de vulnerabilidades.

Entre las fases por las que transcurre el software, se encuentra la Fase de Pruebas, esta constituye el eslabón fundamental para comprobar que el mismo cumple con las especificaciones de seguridad deseadas. Entre las pruebas de seguridad más conocidas se encuentran las Pruebas de Intrusión[2].

Para la realización de estas pruebas los especialistas pueden hacerlo de manera dinámica o estática. De modo que, de forma estática incluyen la revisión del código y las configuraciones, enfocándose en los defectos de diseño y de codificación. Mientras que de manera dinámica incluyen la utilización de herramientas de pruebas de seguridad. En este proceso de pruebas, el uso de las Bases de Datos de Vulnerabilidades es de gran importancia, dada la gran cantidad de información que aportan sobre las vulnerabilidades, constituyendo una herramienta con la que pueden contar los especialistas a la hora de minimizar los riesgos y mitigar las posibles brechas de seguridad de los sistemas.

En Cuba los avances alcanzados en los últimos años en la informatización de la sociedad, a partir del incremento de tecnologías de la información en todos los sectores y en particular de las redes informáticas, han requerido la adopción de medidas que garantizan un adecuado nivel de seguridad y protección. La seguridad informática constituye, una de las prioridades a tenerse en cuenta en el proceso de informatización.

[1] Es el estado de cualquier tipo de información que indica que ese sistema está libre de peligro, daño o riesgo.

[2] Prueba de Intrusión: son también conocidas como pruebas de penetración. Son una práctica para poner a prueba un sistema informático, red o aplicación web para encontrar vulnerabilidades que un atacante podría explotar.

En este sentido, una de las instituciones encargadas de la informatización de los sectores de la sociedad es la Universidad de las Ciencias Informáticas (UCI). Entre los centros productivos con los que cuenta, el centro de Telemática (TLM) se especializa en el desarrollo de sistemas y servicios informáticos en las ramas de las Telecomunicaciones y la Seguridad Informática. Específicamente en la rama de la seguridad informática se desarrolló una Plataforma de Seguridad en Tecnologías de la Información (Xilema-PlatSI), que tiene integradas varias herramientas de detección de vulnerabilidades y gestiona el proceso de auditorías a aplicaciones web.

En dicho proceso, se obtiene un informe con los resultados de las vulnerabilidades detectadas por las herramientas de prueba. Obtenido este informe, los especialistas proceden a realizar un análisis manual de cada una de las vulnerabilidades contenidas en dicho informe, con el objetivo de detectar la mayor cantidad de vulnerabilidades que constituyan falsos positivos[3] e identificar las que compartan una misma descripción, debido a que las herramientas de pruebas de seguridad, pueden describir una misma vulnerabilidad de formas diferentes Para llevar a cabo este proceso de análisis, los especialistas realizan una búsqueda de información mediante el uso de internet, en las bases de datos de vulnerabilidades y en sitios que ofrecen reportes sobre vulnerabilidades.

Este proceso de análisis constituye un proceso engorroso, dado que obtener la información necesaria de las vulnerabilidades, para lograr la correcta verificación de falsos positivos y la detección de vulnerabilidades que compartan una misma descripción, consume mucho tiempo para los especialistas, debido a que la información no se encuentra distribuida en un solo sitio. Por esta razón, los especialistas tienen que acceder a las aplicaciones web referenciadas, para de esa forma, completar la información de dichas vulnerabilidades, viéndose afectada su organización y selección. Otra de las dificultades que posee, es el acceso y el uso de la información, debido a las limitantes que traen consigo los sistemas privativos.

Por todo lo anteriormente planteado se define como **problema a resolver:** ¿Cómo facilitar el proceso de análisis que realizan los especialistas a las vulnerabilidades detectadas por Xilema-PlatSI?

Como **objeto de estudio** de la presente investigación se plantea: Proceso de auditorías de seguridad informática.

Para dar solución al problema anteriormente planteado se traza como **objetivo general:** Desarrollar un sistema de gestión de información de vulnerabilidades de seguridad informática que facilite el

[3] Es un error por el cual se informa que un software presenta una vulnerabilidad informática, cuando en realidad el software no presenta la vulnerabilidad informada.

proceso de análisis que realizan los especialistas del centro de Telemática en las auditorías de seguridad informática.

El **campo de acción** se enfoca en el proceso de auditorías de seguridad informática realizado por Xilema-PlatSI, perteneciente al Centro de Telemática.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de la investigación:**

- Caracterización de las Bases de Datos de Vulnerabilidades de acceso libre existentes en el mundo, para establecer las similitudes con la investigación en curso.
- Fundamentación de la metodología de software, las herramientas y tecnologías a utilizar, para el desarrollo del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática.
- Definición de las funcionalidades del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática, para el desarrollo de la solución.
- Selección de patrones de diseño para realizar el análisis y diseño del sistema.

En el desarrollo de la presente investigación se emplearon los siguientes métodos científicos:

Métodos Teóricos:

- Analítico - sintético: Se realizó un estudio de la documentación relacionada con las categorías por las que se pueden agrupar las vulnerabilidades de seguridad informática y se seleccionaron las más relevantes para la investigación.

Métodos empíricos:

- Observación: Se utilizó para observar y analizar el comportamiento de los especialistas de Xilema-PlatSI, en el proceso de análisis de la información de las auditorías de seguridad informática.

El presente documento está estructurado por capítulos de la siguiente forma:

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA:

En este capítulo se exponen las categorías por las que se pueden agrupar las vulnerabilidades de seguridad informática. Se realiza el estudio de las tendencias de las Bases de Datos de Vulnerabilidades en la seguridad informática. Se describe la metodología y las tecnologías empleadas para el desarrollo del sistema.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA:

En este capítulo se exponen las características del sistema, incluyendo las funcionalidades con las que contará la aplicación, así como algunos de los artefactos generados por la metodología seleccionada, como son las Historias de Usuarios (HU) y la planificación.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA:

En este capítulo se exponen aspectos relacionados con la estructura de la solución propuesta para dar solución a la problemática en cuestión. Se fundamenta la arquitectura empleada, se definen los patrones para la asignación de responsabilidades y se elaboran artefactos como son las tarjetas Clase-Responsabilidad-Colaborador (CRC) y las tareas de la ingeniería por cada HU, las cuales son asignadas a los programadores. Se elabora el modelo de datos y el modelo de componentes. Se definen las estrategias de codificación, estándares y el estilo de código a utilizar.

CAPÍTULO 4. PRUEBAS DEL SISTEMA:

En este capítulo se define la estrategia, los tipos de pruebas realizadas al sistema y los diseños de casos de prueba a utilizar en la validación de la aplicación. Además se analizan los resultados de las pruebas realizadas, para dar una evaluación en cuanto a la calidad de la propuesta de solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se hace referencia a las categorías más relevantes en que se pueden agrupar las vulnerabilidades de seguridad informática. Se realiza una síntesis de los sistemas de información que gestionan vulnerabilidades en el mundo y un análisis de las tecnologías, metodología y herramientas que serán utilizadas para dar solución al problema en cuestión.

1.2. Clasificación de las vulnerabilidades por categorías

Para una mejor gestión, búsqueda y manejo de las vulnerabilidades informáticas, fueron agrupadas por categorías, a continuación se mencionan las más relevantes (3):

- Cross-Site Scripting (XSS): Las vulnerabilidades en esta categoría son introducidas cuando la aplicación web envía los datos proporcionados por el usuario al navegador web, sin realizar ninguna validación o codificación del contenido. Esta vulnerabilidad permite al atacante ejecutar código (scripts) en el navegador de la víctima.
- Cross-Site Request Forgery (CSRF): Las vulnerabilidades en esta categoría son detectadas cuando se fuerza al navegador de la víctima a enviar peticiones pre-autentificadas a una aplicación web vulnerable. Estas peticiones obligan al navegador de la víctima a realizar acciones no deseadas en beneficio del atacante sobre una sesión previamente establecida.
- Configuración: Las vulnerabilidades en esta categoría son típicamente encontradas durante la configuración del software.
- Problemas de Autenticación: Las vulnerabilidades en esta categoría son detectadas cuando las credenciales de acceso y los identificadores de sesión no son protegidos adecuadamente.
- Permisos, Privilegios y Control de Acceso: Las vulnerabilidades en esta categoría son detectadas cuando la gestión de permisos, privilegios y otras características de seguridad que se utilizan para llevar a cabo el control de acceso se encuentran comprometidas.
- Errores de Búfer: Las vulnerabilidades en esta categoría son detectadas cuando se realizan operaciones en un búfer de memoria, en el que se puede leer o escribir en una ubicación de memoria que se encuentra fuera del límite previsto del búfer.
- Almacenamiento criptográfico inseguro: Las vulnerabilidades en esta categoría son detectadas cuando las aplicaciones no utilizan funciones criptográficas adecuadas para la protección de los datos y las credenciales empleadas.

- Inyección SQL: Las vulnerabilidades en esta categoría son detectadas en las aplicaciones cuando los datos proporcionados por el usuario se envían a un intérprete como parte de un comando o consulta para obtener o modificar información.
- Código: Las vulnerabilidades en esta categoría son típicamente encontradas durante el desarrollo de código, incluyendo la especificación, el diseño y la implementación.
- Manejo de datos: Las vulnerabilidades en esta categoría son típicamente encontradas en las funcionalidades que procesan los datos.
- Inapropiada validación de entrada: El producto no es validado o usa una incorrecta validación de entrada, que puede afectar el flujo de control o el flujo de los datos del programa. Cuando el software no valida adecuadamente la entrada, el asaltante puede elaborar una forma de entrar no esperada por la aplicación.

1.3. Sistemas de gestión de información de vulnerabilidades de seguridad informática

Un sistema de gestión de información, es un sistema integrado y automatizado para proveer la información que sostenga las funciones de operatividad, gestión y toma de decisiones en una organización. (4)

A nivel nacional e internacional han sido diseñadas y confeccionadas varias soluciones informáticas, con el fin de fortalecer los procesos de gestión de vulnerabilidades en el área de la seguridad informática. A continuación se muestra una breve reseña de algunos de estos sistemas:

Vulnerabilidades y Riesgos Comunes (CVE[4])

Es un sistema que proporciona un listado de información sobre seguridad de vulnerabilidades y riesgos. Su aspiración es proveer públicamente nombres comunes para problemas conocidos. Trae como meta facilitar la compartición de información, separando las vulnerabilidades por el tipo de capacidad, utilizando una enumeración común. (5)

En CVE se describen las vulnerabilidades con los siguientes atributos:

- ID: Se refiere al identificador único de la vulnerabilidad en el sistema.
- Descripción: Muestra de forma detallada las características de la vulnerabilidad.

[4] Common Vulnerabilities and Exposures por sus siglas en ingles.

- Referencias: Se refiere al identificador que posee la vulnerabilidad y al Localizador de Recursos Uniforme (URL[5]) donde se encuentra información.
- Fecha de creación: Se refiere a la fecha en que fue insertada la vulnerabilidad en la base de datos.
- Fase: Se refiere a la fecha de modificación.
- Comentarios: Se refiere a los votos realizados para la modificación de la vulnerabilidad.
- Propuesto: Se refiere a la fecha en que fue propuesta la vulnerabilidad.

Base de Datos de Vulnerabilidades de Código Abierto (OSVDB[6])

Es una base datos web de vulnerabilidades, independiente y de código abierto, creada por la Fundación de Seguridad Abierta (OSF[7]). El objetivo del proyecto es proporcionar información técnica precisa, detallada, actualizada y objetiva sobre vulnerabilidades de seguridad. El proyecto espera proveer una base de datos de vulnerabilidades con características extendidas para facilitar una mejor búsqueda, clasificación de la información y referencias. (6)

En OSVDB se describen las vulnerabilidades con los siguientes atributos:

- ID: Se refiere al identificador único de la vulnerabilidad en el sistema.
- Fecha: Se refiere a la fecha en la que fue insertada la vulnerabilidad en la base de datos.
- Nombre: Se refiere al nombre de la vulnerabilidad.
- Descripción: Muestra de forma detallada las características de la vulnerabilidad.
- Solución: Se refiere al procedimiento a realizar para erradicar la vulnerabilidad.
- Referencias: Se refiere al identificador que posee la vulnerabilidad y al URL donde se puede encontrar información.
- Productos: Hace referencia a los productos que son dañados por la vulnerabilidad.
- Puntuación CVSSv2: Se refiere a la puntuación decimal en el intervalo de 0 a 1, generada para cada una de las variables siguientes:
    - Tipo de acceso: Local, red adyacente o remota.

[5] URL: secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

[6] Open Sourced Vulnerability Database por sus siglas en inglés.

[7] Open Security Foundation por sus siglas en inglés.

- Complejidad en el acceso: Baja, media o alta.
- Autenticación: Ninguna, una instancia o múltiples instancias.
- Confidencialidad: Ninguna, parcial o completa.
- Integridad: Ninguna, parcial o completa.
- Disponibilidad: Ninguna, parcial o completa.

➢ Comentarios: Se refiere a los comentarios realizados sobre la vulnerabilidad.

Secunia VIM[8]

Es un manejador de vulnerabilidades inteligente en tiempo real. Desarrollado por la empresa de seguridad informática especializada en la gestión de vulnerabilidades, Secunia. El mismo permite ser accedido desde internet por los navegadores web. La base de datos que utiliza el sistema está en constante actualización, ofreciendo compatibilidad con CVE (7). Es portador de gran conocimiento sobre vulnerabilidades y ofrece a sus clientes un rápido manejo de las emergentes amenazas que los pueden afectar.

En Secunia se describen las vulnerabilidades con los siguientes atributos:

➢ ID: Se refiere al identificador único de la vulnerabilidad en el sistema.

➢ Nombre: Se refiere al nombre de la vulnerabilidad.

➢ Fecha de liberación: Se refiere a la fecha en la que fue insertada la vulnerabilidad en la base de datos.

➢ Donde: Se refiere al tipo de acceso.

➢ Impacto: Se refiere a la compañía que afecta.

➢ Estado de solución: Se refiere al estado de la solución.

➢ Sistema operativo: Se refiere al nombre del producto que afecta.

➢ Referencias CVE: Se refiere al URL de la base de datos de CVE.

➢ Descripción: Muestra de forma detallada las características de la vulnerabilidad.

➢ Solución: Se refiere al procedimiento a realizar para erradicar la vulnerabilidad.

➢ Proveedor o descubridor: Se refiere al nombre de la persona que la identificó.

[8] Vulnerability Intelligence Manager por sus siglas en inglés.

- Asesor original: Se refiere al URL del proveedor o descubridor.

Módulo Base de Datos de Vulnerabilidades (MBDV)

Es un módulo desarrollado en el Centro de Telemática de la UCI para Xilema-PlatSI. Este MBDV permite el almacenamiento y la gestión de vulnerabilidades de tipo software de seguridad informática. Actualmente no se encuentra integrado a Xilema-PlatSI.

En MBDV se describen las vulnerabilidades con los siguientes atributos:

- Categoría: Hace referencia a la categoría que pertenece la vulnerabilidad.
- Nombre: Hace referencia al nombre de la vulnerabilidad.
- Descripción: Se explica de forma detallada la vulnerabilidad.
- Impacto: Hace referencia a las consecuencias provocadas por la vulnerabilidad. Se mide en una escala de alto, medio o bajo.
- Solución: Hace referencia al procedimiento a realizar para eliminar la vulnerabilidad, en caso de que exista.
- Fecha de creación: Hace referencia a la fecha en que se inserta la vulnerabilidad en la base de datos.
- Última actualización: Hace referencia a la fecha en que se modifican los datos de la vulnerabilidad en la base de datos.
- Autor: Hace referencia al nombre de la persona que inserta la vulnerabilidad en la base de datos.

Después de realizado el estudio de los sistemas existentes a nivel nacional e internacional se llegó a las siguientes conclusiones:

El sistema CVE, después de ser analizada la forma en que se describen las vulnerabilidades de seguridad informática, se pudo comprobar que la información que provee su base de datos no es suficiente para dar solución al problema de la investigación, pues existen otros atributos que son de gran importancia para el análisis de las vulnerabilidades como son: impacto, categoría y el mismo no los contiene. A pesar de estos inconvenientes, será utilizada su base de datos para poblar el Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática.

El sistema OSVDB describe las características de las vulnerabilidades de una forma similar a las necesarias para dar solución al problema de la investigación, sin embargo la licencia que usa el sistema es de tipo privativa, lo que imposibilita el uso de su base de datos. Además para hacer uso

de la Interfaz de Programación de Aplicaciones (API) con fines comerciales, se debe realizar un contacto con la OSF para discutir la concesión de la licencia. Este sistema solo permite realizar 2 consultas por día y es prohibido y sancionado utilizar cualquier mecanismo que intente descargar las vulnerabilidades de manera automática. Por las razones anteriormente mencionadas, no puede ser utilizado como solución al problema de la investigación.

El proyecto Secunia dado su licencia privativa, no comparte la información de las vulnerabilidades que registra en su base de datos, solo ofrece reportes de vulnerabilidades a los miembros de su comunidad. Para poder hacer uso del sistema Secunia VIM, se debe realizar un contacto directo con los propietarios de la compañía y pagar de acuerdo a los términos de su licencia por cada cuenta registrada en el sistema. Además el acceso al mismo es limitado, solo permite una sesión por cuenta registrada en el sistema. Por las razones anteriormente mencionadas, no puede ser utilizado como solución al problema de la investigación.

El MBDV a pesar de ser una solución bien pensada en el tratamiento de la seguridad informática, actualmente no permite la actualización de las vulnerabilidades, debido a que utiliza la base de datos de OSVDB. Además la forma en que se gestiona la información de las vulnerabilidades en dicho módulo no se corresponde con las necesidades de la investigación, un ejemplo de ello se puede evidenciar cuando se importa una base de datos más actualizada, se sobrescribe toda la información registrada anteriormente, trayendo consigo la posible pérdida de información. Por las razones anteriormente mencionadas, no puede ser utilizado como solución al problema de la investigación, sin embargo el estudio de este sistema dejo bien definido cuáles son las características de las vulnerabilidades con las que debe contar la propuesta de solución.

1.4. Metodología de desarrollo de software

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo han evolucionado durante los años, cada uno con sus propias fortalezas y debilidades (8). Las metodologías de desarrollo de software son clasificadas en tradicionales y ágiles. Las tradicionales son consideradas metodologías pesadas dado el uso exhaustivo de documentación que se lleva a cabo en el ciclo de vida del desarrollo del software. En cambio, las ágiles incluyen poca documentación, siendo su prioridad el desarrollo rápido del software y la capacidad de respuesta a los cambios de requisitos. Para ello incluyen al cliente como un miembro más del equipo de desarrollo.

1.4.1. Metodología: Programación Extrema (XP[9])

Existen muchas metodologías agiles conocidas y utilizadas en el desarrollo de software, pero la Programación Extrema, resalta sobre todas las demás, por su éxito en mantener un buen clima de trabajo, favoreciendo la fluidez de la comunicación entre todos sus participantes (9). Para la selección de esta metodología se tuvieron en cuenta las características que la definen, como son de mencionar:

- Equipo de desarrollo pequeño y programación por parejas: El equipo de desarrollo del sistema está compuesto por 2 programadores.
- Constante comunicación con el cliente: El cliente está conformado por los especialistas de XILEMA-PlatSI, que a su vez intervienen en el presente trabajo como parte del equipo de desarrollo.
- Desarrollo rápido del software: Se cuenta con poco tiempo para la implementación del sistema, siendo la prioridad el desarrollo del sistema y no la generación exhaustiva de documentación.
- Relaciones interpersonales como clave para el éxito del desarrollo del software: La comunicación entre los miembros del equipo de desarrollo es fluida, siempre se encuentra en constante retroalimentación.
- Flexibilidad a la hora de enfrentar cambios: Esto permite la variación del cronograma de actividades y las fechas previstas para las mismas, donde el cambio no afecte el resultado final que se espera obtener en cada iteración.
- Integración continúa de las funcionalidades: Esto permite que por cada entrega de las funcionalidades realizadas al cliente, se obtenga un producto funcional más completo.

1.5. Tecnologías y herramientas seleccionadas

A continuación se describen las herramientas y tecnologías escogidas en el proceso de desarrollo del sistema:

1.5.1. Marco de trabajo Xilema-Base-Web

Es un marco de trabajo desarrollado en el Centro de TLM. Cuenta con Django como framework base y librerías de JavaScript como son: JQuery y BackboneJS. Este marco de trabajo brinda un conjunto de clases y formularios para su reutilización y con las pautas de diseño definidas por la universidad.

[9] Extreme Programming por sus siglas en inglés.

Además Xilema-Base-Web incorpora un módulo para la gestión de usuarios y roles del sistema, esto garantiza que el acceso al sistema sea realizado mediante un proceso de autenticación.

1.5.2. Herramienta de Ingeniería de Software Asistida por Computación (CASE[10])

Visual Paradigm

Es una herramienta multiplataforma, distribuida de forma comercial o gratuita para desarrolladores independientes. Se utilizó esta herramienta en su versión 8.0, para generar artefactos que apoyen a la propuesta de solución y permitan un mejor entendimiento de la situación problemática de la investigación. Específicamente se modeló el diagrama de negocio del proceso de análisis de vulnerabilidades de seguridad informática, el diagrama de modelo de datos, el diagrama de componentes y los diagramas que representan las capas que integran el patrón arquitectónico Modelo-Plantilla-Vista.

1.5.3. Framework de desarrollo

Django

Es un framework web de código abierto escrito en Python que permite el desarrollo de aplicaciones informáticas. El sitio oficial ofrece documentación variada y actualizada (manuales, ejemplos, buenas prácticas de programación). Se utilizó esta herramienta específicamente en su versión 1.7, debido a que es la versión empleada en el marco de trabajo Xilema-Base-Web. El uso de este framework facilitó a los programadores del sistema, la utilización de vistas genéricas, la obtención de la base de datos mediante el mapeo objeto-relacional, la herencia entre plantillas y la internacionalización en los idiomas inglés y español.

1.5.4. Framework de JavaScript

JQuery

Es una librería JavaScript que facilita el recorrido de DOM[11] (Modelo de Objetos del Documento), el manejo de eventos, las animaciones, y las funcionalidades para trabajar con AJAX[12] (10). Se utilizó esta librería específicamente en su versión 1.9.1, debido a que es la versión empleada en el marco de trabajo Xilema-Base-Web. El uso de la librería JQuery posibilitó la compatibilidad de la aplicación con los navegadores, la creación de interfaces de usuarios y los efectos dinámicos.

---

[10] **C**omputer **A**ided **S**oftware **E**ngineering por sus siglas en inglés. Es la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

[11] Document Object Model por sus siglas en inglés.

[12] Asynchronous JavaScript And XML por sus siglas en inglés.

BackboneJS

Es una librería utilizada para el desarrollo de aplicaciones web que requieran el uso de código JavaScript. Se utilizó esta librería específicamente en su versión 1.1.0, debido a que es la versión empleada en el marco de trabajo Xilema-Base-Web. El uso de esta librería simplificó la creación de las vistas y formularios del sistema, permitiendo que las interfaces de usuario puedan ser renderizadas a una misma página. Además, facilitó el trabajo con los modelos del sistema, debido a que utiliza una estructura de tipo llave/valor.

1.5.5. Entorno de Desarrollo Integrado(IDE[13])

PyCharm

Es un IDE con una interfaz sencilla, que permite el autocompletado y el resaltado de las sintaxis de código. Se utilizó la versión 3.0 como herramienta de desarrollo para el presente trabajo, debido a su capacidad de integración con el framework Django, JQuery y con el lenguaje de plantillas Django Template.

1.5.6. Gestor de Bases de Datos

PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional. Utiliza un modelo cliente/servidor y posee un buen funcionamiento a la hora de trabajar con grandes cantidades de información (11). Para la gestión de los datos de la aplicación se escogió la versión 9.1, debido que su licencia es de código abierto, permitiendo su utilización en cualquier sistema operativo. El mismo ofrece soporte para distintos tipos de datos y compatibilidad con Django para el acceso a los datos de la base de datos.

1.5.7. Lenguaje de programación del lado del cliente

El Lenguaje de Marcas de Hipertexto (HTML[14])

Es un lenguaje utilizado para la estructuración del contenido de las páginas web. Se utilizó la versión 4 en la estructuración de las interfaces del sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática, debido a que esta versión ofrece soporte para diversos lenguajes dentro de un documento. El uso de código HTML en las plantillas del sistema garantizó la compatibilidad con los navegadores web existentes.

JavaScript

[13] Integrated Development Environment por sus siglas en inglés.

[14] Hyper Text Markup Language por sus siglas en inglés.

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas, los programas escritos con este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este lenguaje es ligero e interpretado y orientado a objetos (12). Se utilizó para la confección de tablas, formularios y estructura de cajas en las vistas de la aplicación.

Las Hojas de Estilo en Cascada 3 (CSS[15])

Es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML. Se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista (13). CSS3 es muy usado por los actuales navegadores web y compatible con las versiones anteriores. El mismo se utilizó para el diseño de las interfaces del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática.

1.5.8. Lenguaje de programación del lado del servidor

Python

Es un lenguaje de programación de licencia gratuita y de tipado dinámico[16], se puede manejar con mayor facilidad el cúmulo de información que se gestiona. Se utilizó este lenguaje en su versión 2.7, debido a que es el lenguaje empleado en el framework Django. El uso de este lenguaje simplificó el desarrollo de las funcionalidades del sistema, debido a la gran variedad de librerías y documentación que ofrece.

1.6. Conclusiones parciales

En el presente capítulo se realizó un estudio relacionado con los sistemas de gestión de información de vulnerabilidades de seguridad informática existentes, el mismo tributó los conocimientos necesarios para la definición de una descripción propia de las vulnerabilidades de seguridad informática. El estudio de las soluciones existentes reflejo que el uso de la base de datos de CVE aportará una parte de la información que se requiere para la propuesta de solución. Además se describieron las herramientas, metodología y lenguajes utilizados para el desarrollo del sistema.

[15] Cascading Style Sheets por sus siglas en ingles.

[16] Un lenguaje de programación es dinámicamente tipado si una misma variable puede tomar valores de distinto tipo en distintos momentos. La mayoría de lenguajes de tipado dinámico son lenguajes interpretados, como Python.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En el presente capítulo se realiza una descripción del proceso del negocio y las principales características de la propuesta de solución, así como las funcionalidades y la lista de reserva de productos con las que contará el sistema. Además se muestran los artefactos generados en las fases de Exploración y Planificación de la metodología XP.

2.2. Descripción del Proceso de Negocio Análisis de Vulnerabilidades de Seguridad Informática

El proceso inicia cuando en Xilema-PlatSI se generan los resultados de las vulnerabilidades detectadas por las herramientas de prueba. Una vez obtenidos estos resultados se procede a realizar un análisis de cada una de las vulnerabilidades detectadas, en dicho análisis los especialistas realizan pruebas de concepto y editan las vulnerabilidades, donde especifican un nivel de riesgo y verifican si las vulnerabilidades son falsos positivos. Concluido este análisis los especialistas proceden a agregar las vulnerabilidades que representan una amenaza real. Seguido a esto se finaliza la auditoría y se procede a generar el informe técnico y el informe general. Concluido la generación de los informes, el proceso termina.

En la siguiente figura se muestra el proceso antes mencionado:

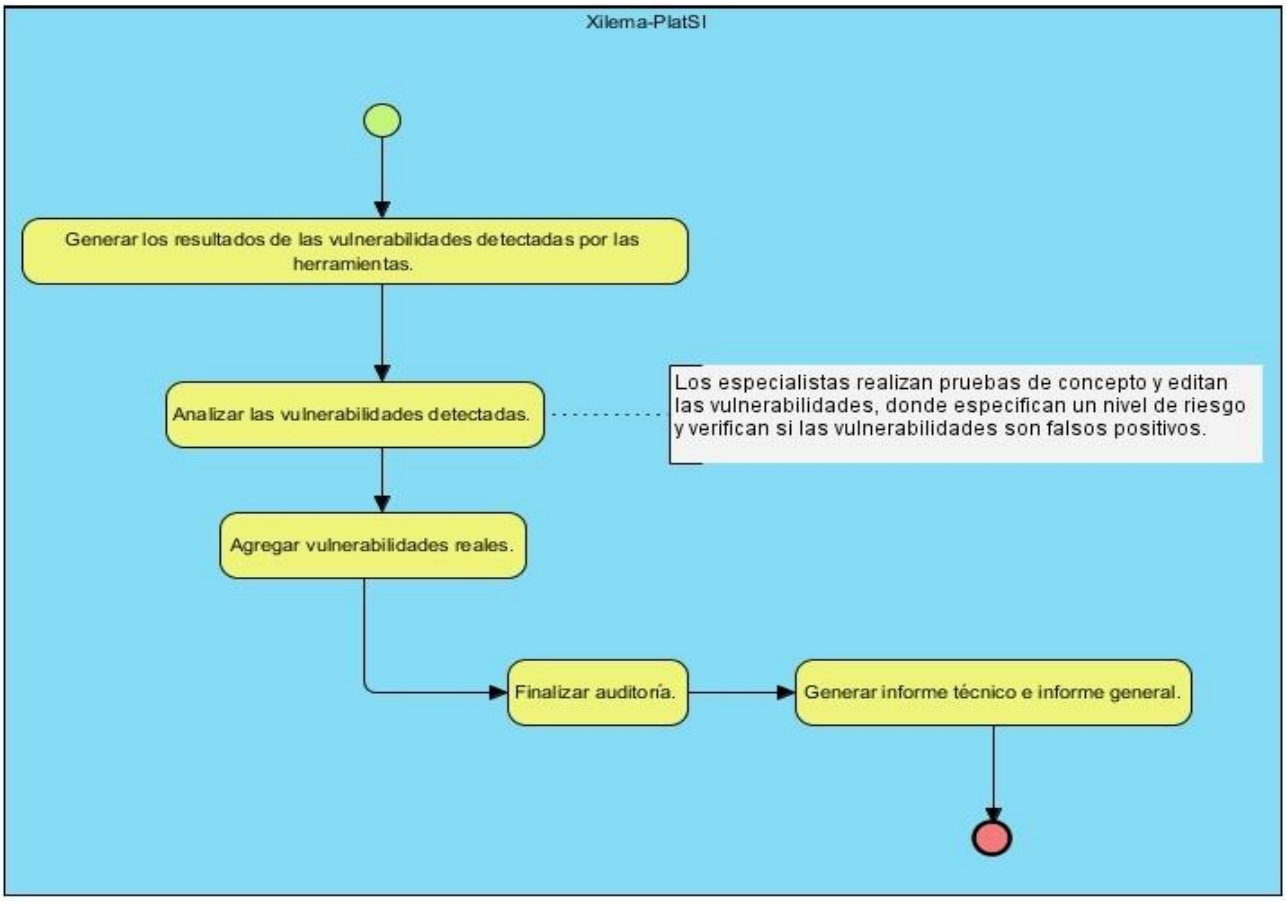


Figura 1. Proceso de Negocio Análisis de Vulnerabilidades de Seguridad Informática.

2.3. Propuesta de solución

Con el objetivo de facilitar el proceso de análisis que realizan los especialistas a las vulnerabilidades detectadas por Xilema-PlatSI, se propone el desarrollo de un Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática que permita a los especialistas la consulta de información de las vulnerabilidades contenidas en la base de datos de CVE. El mismo permitirá:

- La realización de búsquedas avanzadas de vulnerabilidades, para facilitar la verificación de las mismas.
- La inserción de vulnerabilidades que no se encuentren en la base de datos de CVE, para que puedan ser consultadas en posteriores auditorías.
- Exportar la información de la base de datos en un fichero JSON, para que se puedan hacer resguardos de la información contenida en el sistema.
- Importar la base de datos del sistema y la de CVE, para que el sistema esté lo más actualizado posible.

- Clasificar las vulnerabilidades por categorías.
- Asignar las herramientas con las que fueron detectadas las vulnerabilidades y un impacto.
- La introducción de una solución para las vulnerabilidades.
- La gestión de usuarios y roles del sistema, para garantizar los principios de la seguridad informática.

El sistema contará con una Base de Datos estable, la misma será confeccionada y poblada por los desarrolladores del sistema.

2.4. Funcionalidades del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática

El sistema debe permitir:

- Gestionar vulnerabilidades de seguridad informática.
    1. Insertar nueva vulnerabilidad.
    2. Editar vulnerabilidad.
    3. Eliminar vulnerabilidad.
    4. Mostrar detalles de vulnerabilidad.
    5. Buscar vulnerabilidad.
    6. Listar vulnerabilidades aprobadas.
    7. Listar vulnerabilidades no aprobadas.
    8. Aprobar o desaprobar vulnerabilidad.
    9. Asignar herramientas a una vulnerabilidad.
- Gestionar categorías.
    1. Insertar nueva categoría.
    2. Editar categoría.
    3. Eliminar categoría.
    4. Mostrar detalles de categoría.
    5. Buscar categoría.
    6. Listar categorías.

- Gestionar las herramientas de prueba de seguridad.
    1. Insertar nueva herramienta.
    2. Editar herramienta.
    3. Eliminar herramienta.
    4. Mostrar detalles de herramienta.
    5. Buscar herramienta.
    6. Listar herramientas.
- Exportar la Base de Datos de vulnerabilidades de seguridad informática.
- Importar la Base de Datos de vulnerabilidades de seguridad informática.
- Importar la Base de Datos de CVE.

2.5. Lista de Reserva de Producto del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática

Para lograr el correcto funcionamiento del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática se deben tener en cuenta la lista de reserva de productos siguiente:

2.5.1. Interfaz de usuario

- El usuario debe poder acceder a las diferentes opciones que brida el sistema, sin tener que navegar por otras opciones intermedias.

2.5.2. Hardware

- Computadora de 512 MB de RAM o superior.
- Espacio en disco de 2 GB.

2.5.3. Software

Para el cliente:

- Navegador Mozilla Firefox 3.0 o superior.

Para el servidor:

- Gestor de Base de Datos PostgreSQL 9.1.
- Python v2.7 instalado.
- Sistema operativo:

Podrán ser utilizadas las siguientes distribuciones:

- ✓ Ubuntu 12.04 o superior.
- ✓ Debian 6 o 7.
- ✓ Nova.

2.5.4. Seguridad

- El sistema debe permitir que la información solo sea consultada y/o modificada solo por el personal autorizado. El nivel de acceso a las funcionalidades estará dado por el rol del usuario autenticado.
- El sistema debe permitir que la contraseña se almacene de manera encriptada en la base de datos.

2.6. Personas relacionadas con el Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática

Se definen como personas relacionadas con el sistema, al encargado de gestionar los usuarios, roles y permisos (administrador) y a los especialistas en seguridad informática, encargados de interactuar con la propuesta funcional que se obtenga.

Tabla 1: Personas relacionadas con el sistema.

Personas relacionadas con el sistema	
Usuarios	**Responsabilidad**
Administrador	Encargado de la gestión de usuarios, roles y permisos.
Especialista	Encargado de la gestión de las vulnerabilidades en el sistema.

2.7. Fase de Exploración

La fase de Exploración constituye la primera fase de la metodología XP. En esta se trata de dar a conocer a todos los integrantes del proyecto lo que debería hacer el sistema, siendo la primera tarea la confección de las Historias de Usuario (HU).

2.7.1. Historias de Usuario

Las HU es la técnica utilizada por el cliente para especificar los requisitos que debe contener el software. Son escritas en tarjetas, usando un lenguaje común, sin tener que hacer uso de un vocabulario técnico. Ofrecen un tratamiento dinámico y flexible, debido a que en cualquier momento

pueden ser desechadas, modificadas, remplazadas por otras o añadirse nuevas HU. Dada su comprensibilidad los programadores pueden implementarlas en pocas semanas.

Según Kent Beck[17] cada HU recoge al menos los siguientes aspectos (14):

- Número: Posee el número asignado a la HU.
- Nombre de HU: Atributo que contiene el nombre de la HU.
- Usuario: El usuario del sistema que utiliza o protagoniza la HU.
- Prioridad en el negocio: Evidencia el nivel de prioridad de la HU en el negocio. Se considera Alta en caso de que la HU sea imprescindible en el negocio, Media en caso de que su realización o no lo afecte considerablemente y Baja cuando no se considera una prioridad para el negocio.
- Riesgo de desarrollo: Evidencia el nivel de riesgo en caso de no realizarse la HU. Se considera Alta, cuando el riesgo de no realizar la HU implica en el funcionamiento del sistema. Media cuando el riesgo de no realizarla es medianamente importante y Baja en caso de que no se considere un riesgo el hecho de tardar en la realización de la HU y no implique en el funcionamiento del sistema.
- Puntos estimados: Este atributo no es más que una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal en 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.
- Iteración asignada: Se especifica la iteración a la que pertenece la HU correspondiente.
- Descripción: Posee una descripción de lo que realizará la HU.

A continuación se muestran algunas de las historias de usuario confeccionadas:

Tabla 2: HU #1: Gestionar vulnerabilidad de seguridad informática

Historia de Usuario	
Número: 1	**Usuario:** Especialista
Nombre de HU: Gestionar vulnerabilidad de seguridad informática.	

[17] Ingeniero de software estadounidense. Uno de los creadores de las metodologías de desarrollo de software de programación extrema ((eXtreme Programming o XP))

Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado listar, insertar, editar, eliminar, ver detalles, buscar, aprobar o desaprobar y asignar herramientas de las vulnerabilidades.	

Tabla 3: HU #2 Gestionar categorías

Historia de Usuario	
Número: 2	**Usuario:** Especialista
Nombre de HU: Gestionar categorías.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1,2	**Iteración asignada:** 1
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado listar, insertar, editar, eliminar, ver detalles y buscar categorías de vulnerabilidades.	

Las restantes HU definidas en la investigación podrán consultarse en el Anexo I.

2.8. Fase de Planificación

En esta fase se procede a definir la prioridad de las HU de usuario, estimándose el esfuerzo que les costará a los programadores implementar cada una de ellas. Además se hace un debate para determinar el cronograma y el contenido de la primera entrega, que no debe superar las tres semanas.

2.8.1. Proceso de estimación

Para la realización de la estimación de las HU, los programadores establecerán de 1 a 3 puntos, cada punto equivale a una semana ideal de programación.

2.8.1.1. Estimación de esfuerzo por historias de usuario

Para el desarrollo de la aplicación propuesta se realizó una estimación de esfuerzo según el método juicio de experto basado en la experiencia por cada una de las historias de usuario identificadas. En la siguiente tabla se muestra la estimación del esfuerzo por cada HU propuesta en el proceso de desarrollo de las funcionalidades del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática.

Tabla 4: Estimación de esfuerzo por Historias de Usuario

No	Historia de Usuarios	Puntos de Estimación
1	Gestionar vulnerabilidad de seguridad informática.	2
2	Gestionar categorías.	1,2
3	Gestionar las herramientas de prueba de seguridad.	1,2
4	Exportar la Base de Datos de vulnerabilidades de seguridad informática.	2
5	Importar la Base de Datos de vulnerabilidades de seguridad informática.	2
6	Importar la Base de Datos de CVE.	2

2.8.1.2. Plan de Iteraciones

Luego de identificarse, describirse y estimarse el esfuerzo dedicado para el desarrollo de cada una de las HU, se procede a la planificación de la fase de implementación, para las cuales se establecieron seis iteraciones:

Iteración 1: Se llevó a cabo el desarrollo de la HU # 2 y la HU # 3.

Iteración 2: Se llevó a cabo el desarrollo de la HU # 1.

Iteración 3: Se llevó a cabo el desarrollo de la HU # 4.

Iteración 4: Se llevó a cabo el desarrollo de la HU # 5.

Iteración 5: Se llevó a cabo el desarrollo de la HU # 6.

2.8.2. Plan de duración de las iteraciones

El plan de duración de las iteraciones es el encargado de mostrar el orden en que se desarrollaron las HU, en correspondencia con la iteración a la que pertenezcan y la duración estimada en semanas y días.

Tabla 5: Plan de duración de las iteraciones

Iteración	Orden de la HU a implementar	Duración (Semanas, días)
1	HU # 2: Gestionar categorías. HU # 3: Gestionar las herramientas de prueba de seguridad.	2 semanas y 2 días
2	HU # 1: Gestionar vulnerabilidad de seguridad informática.	2 semanas

3	HU # 4: Exportar la Base de Datos de vulnerabilidades de seguridad informática.	2 semanas
4	HU # 5: Importar la Base de Datos de vulnerabilidades de seguridad informática.	2 semanas
5	HU # 6: Importar la Base de Datos de CVE.	2 semanas

2.8.3. Plan de entregas

En el plan de entrega se muestra la fecha de inicio y finalización de las HU implementadas por cada una de las iteraciones. Obteniéndose un total de cinco entregas.

Tabla 6: Plan de entregas

# HU	Iteración 1 Inicio: 10/3/2015 Fin: 26/3/2015	Iteración 2 Inicio: 26/3/2015 Fin: 9/4/2015	Iteración 3 Inicio: 9/4/2015 Fin: 23/4/2015	Iteración 4 Inicio: 23/4/2015 Fin: 7/5/2015	Iteración 5 Inicio: 7/5/2015 Fin: 21/5/2015
2 y 3	1ra entrega	Finalizada	Finalizada	Finalizada	Finalizada
1	-	2da entrega	Finalizada	Finalizada	Finalizada
4	-	-	3ra entrega	Finalizada	Finalizada
5	-	-	-	4ta entrega	Finalizada
6	-	-	-	-	5ta entrega

2.9. Conclusiones parciales

En este capítulo se describió la propuesta de solución. Se determinaron los requerimientos y los usuarios relacionados con el sistema. De acuerdo con la metodología de desarrollo XP, se identificó la fase inicial de Exploración, en la que se confeccionaron seis HU referentes a las funcionalidades seleccionadas como parte de la propuesta de solución. En la posterior fase de Planificación se estimaron los esfuerzos necesarios para desarrollar cada historia (mediante puntos de estimación) y las iteraciones en las que estarán comprendidas con el objetivo de obtener productos funcionales de forma organizada y en el tiempo estimado, el cual fue especificado en correspondencia con la duración de cada iteración mediante el plan de entrega.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1. Introducción

En el presente capítulo se realiza el modelado de la aplicación, teniendo en cuenta las actividades que define la metodología XP en la fase de diseño. Entre estos elementos se encuentran la definición de la arquitectura y los patrones de diseño a utilizar. La elaboración de las tarjetas Clase-Responsabilidad-Colaborador (CRC[18]), el modelo de datos, el modelo de componentes y las tareas de la ingeniería, para desglosar las actividades que deben realizar los programadores en cada una de la HU. Se definen las estrategias de codificación, estándares y el estilo de código a utilizar.

3.2. Arquitectura

Una arquitectura del software alude a la estructura global del software y a las formas en que esta proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa, la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. (15)

Los patrones arquitectónicos constituyen una estructura, que especifica la forma en que se van a organizar los componentes de un sistema, así como sus responsabilidades. A continuación se muestra el patrón arquitectónico que interviene en el proceso de desarrollo del sistema:

3.2.1. Patrón Arquitectónico Modelo Plantilla Vista (MTV[19])

El patrón de arquitectura MTV utilizado por el framework Django es una modificación del patrón Modelo Vista Controlador (MVC) para obtener un mejor funcionamiento del mismo. Este patrón propone la construcción de tres componentes distintos:

El modelo: Conocido como la capa de acceso a la base de datos. Almacena toda la información referente a los datos, permite realizar las operaciones para acceder, validar, conocer el comportamiento y las relaciones de los datos.

La plantilla: Conocida como la capa de presentación, contiene las decisiones relacionadas a la presentación, permite presentar en el navegador de forma organizada los datos de la vista.

La vista: Conocida como la capa de la lógica de negocios, contiene el conocimiento para acceder al modelo y delegar la información a presentar a las plantillas apropiadas. Es la capa intermedia entre el modelo y la plantilla (16).

[18] Class, Responsabilities and Collaboration por sus siglas en inglés.

[19] Model Template View por sus siglas en inglés.

Figura 2. Patrón Arquitectónico MTV.

En la Figura 2 se puede observar que el cliente realiza las peticiones mediante una interfaz gráfica, las cuales son interceptadas por la Vista. Esta vista envía los datos al Modelo para guardar u obtener información, depende de la solicitud realizada por el cliente; para finalmente enviarlos a la Plantilla a fin de ser mostrados nuevamente al usuario a través de una respuesta.

A continuación se puede observar la aplicación del patrón MTV en el sistema:

La Capa Modelo está compuesta por:

- Vulnerabilidad: Clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las vulnerabilidades.
- Categoria: Clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las categorías.
- Herramienta: Clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las herramientas.

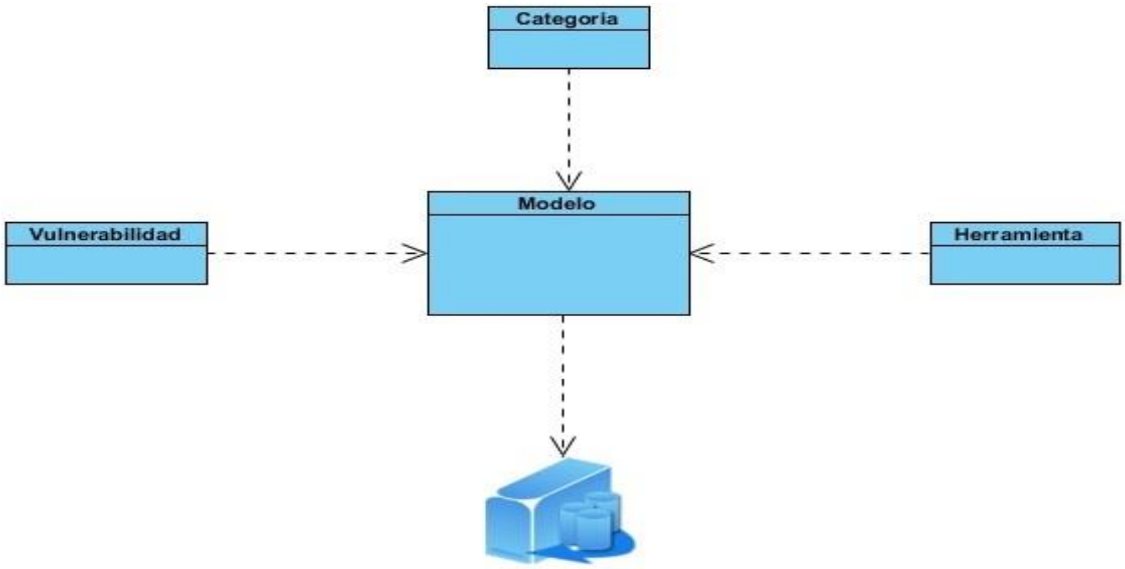


Figura 3: Capa Modelo.

La Capa Plantilla está compuesta por los ficheros categoria, herramienta, importarBDSistema, importarBDCVE, vulnerabilidadAprobada y vulnerabilidadNoAprobada. Los mismos definen las plantillas HTML, CSS y JavaScript.

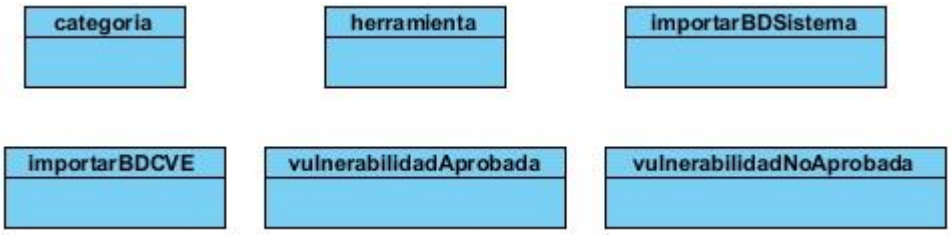


Figura 4: Capa Plantilla.

La Capa Vista está compuesta por:

- ListaCreadaAPI: Clase que contiene el conocimiento para acceder a las clases del modelo (Vulnerabilidad, Categoría y Herramienta) y delegar la información a las plantillas apropiadas (categoria, herramienta, vulnerabilidadAprobada y vulnerabilidadNoAprobada).
- ExportarBDSistema: Clase que contiene el conocimiento para acceder a las clases del modelo (Vulnerabilidad, Categoría y Herramienta).
- ImportarBD: Clase que contiene el conocimiento para acceder a las clases del modelo (Vulnerabilidad, Categoría y Herramienta) y delegar la información a la plantilla apropiada (importarBDSistema).

- ImportarBDCVE: Clase que contiene el conocimiento para acceder a las clases del modelo (Vulnerabilidad, Categoría y Herramienta) y delegar la información a la plantilla apropiada (importarBDCVE).

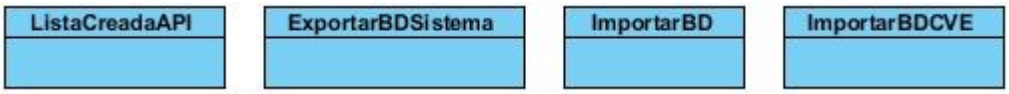


Figura 5: Capa Vista.

3.3. Patrones de diseño

Los patrones de diseño son considerados soluciones ya probadas a problemas de desarrollo de software sujeto a contextos similares. No se utilizan arbitrariamente, se debe tener en cuenta el nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (17)

3.3.1. Patrones generales de software para asignar responsabilidades (GRASP[20])

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen el fundamento de cómo se diseñará el sistema (18).

A continuación se describen los patrones utilizados en el diseño:

Experto

El uso de este patrón posibilitó la asignación de responsabilidades a las clases, de manera que solo contengan la información necesaria para realizar una determinada acción. De este modo los objetos podrán valerse de su propia información para hacer lo que se les pida, favoreciendo así una menor dependencia entre las clases. Una de las clases expertas es AdicionarVulnerabilidad, esta contiene toda la información necesaria de la vulnerabilidad.

Creador

El uso de este patrón posibilitó la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de esta manera los objetos solo puedan ser creados por las clases que contengan la información necesaria para ello. Una de las clases creadoras es AdicionarCategoria, esta clase es la responsable de la creación de las categorías de las vulnerabilidades.

Bajo Acoplamiento

[20] General Responsibility Assignment Software Pattern por sus siglas en inglés.

Este patrón se aplica en todas las clases. Su utilización hace posible que una modificación en una clase tenga poca repercusión en las demás. Esto se logra pues el patrón garantiza la existencia de pocas dependencias entre las clases.

Alta Cohesión

Cada elemento del diseño realiza una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Este patrón es usado en todas las clases ya que permite que los datos y responsabilidades de una clase sean coherentes y estén fuertemente ligados a la misma, en un sentido lógico.

3.4. Tarjetas Clase – Responsabilidad – Colaborador

Las Tarjetas CRC constituyeron una forma de representar las clases que intervendrán en la confección del sistema. Cada una de las tarjetas representa a una única clase. En su confección se describieron en la parte superior el nombre de la misma, en la parte izquierda se describieron las responsabilidades o funciones destinadas a realizar y en la parte derecha las clases colaboradoras que le sirven de soporte para su funcionamiento. A continuación se muestran algunas de las Tarjetas CRC correspondientes al sistema:

Tabla 7: Tarjeta CRC: Clase Categoria

Datos de la Clase	
Nombre de la clase: Categoria	
Responsabilidades	**Colaboradores**
Listar categorías	Categoria
Insertar nueva categoría	Categoria AdicionarCategoria
Editar categoría	Categoria EditarCategoria
Eliminar categoría	Categoria
Mostrar detalles de categoría	
Buscar categoría	

Tabla 8: Tarjeta CRC: Clase Vulnerabilidad

Datos de la Clase	
Nombre de la clase: Vulnerabilidad	
Responsabilidades	**Colaboradores**

Listar Vulnerabilidades Aprobadas	Vulnerabilidad
Listar Vulnerabilidades No Aprobadas	
Insertar nueva vulnerabilidad	Vulnerabilidad AdicionarVulnerabilidad
Editar vulnerabilidad	Vulnerabilidad EditarVulnerabilidad
Eliminar vulnerabilidad	Vulnerabilidad
Mostrar detalles de vulnerabilidad	Vulnerabilidad Categoria Herramienta
Buscar Vulnerabilidad	Vulnerabilidad Categoria Herramienta
Aprobar o desaprobar vulnerabilidad	Vulnerabilidad
Asignar herramientas a una vulnerabilidad	Vulnerabilidad Herramienta

Las restantes tarjetas CRC definidas en la investigación podrán consultarse en el Anexo II.

3.5. Tareas de Ingeniería

Las tareas de ingeniería de la metodología XP constituyeron la forma de representar gráficamente las responsabilidades asignadas a cada miembro del equipo de desarrollo. Estas surgieron producto del desglose de cada una de las HU, para luego ser implementadas en una iteración. Las tareas de la ingeniería están compuestas por los siguientes atributos:

- Número de la tarea: Se refiere al número consecutivo con el que se identificó la tarea.
- Número de Historia de Usuario: Se refiere al número de la historia de usuario a la que pertenece.
- Nombre de la tarea: Se refiere al nombre con el que se va identificar la tarea.
- Tipo de Tarea: Se refiere a si es una tarea de desarrollo, corrección, mejora o especificar otra.
- Puntos estimados: Se refiere al tiempo estimado en semanas en que se desarrolló la tarea.
- Fecha inicio: Se refiere a la fecha en que inició el desarrollo de la tarea.

- Fecha fin: Se refiere a la fecha en que finalizó el desarrollo de la tarea.
- Programador responsable: Se refiere al nombre y apellidos del programador responsable que desarrolló la tarea.
- Descripción: Se refiere a una breve descripción de la tarea.

A continuación se muestran un conjunto de tareas de ingeniería correspondientes a la primera iteración:

Tabla 9: Tarea #1: Listar categorías.

Tarea de Ingeniería de Usuario	
Número de la tarea: 1	**Número de Historia de Usuario:** 2
Nombre de la tarea: Listar categorías.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 10/3/2015	**Fecha fin:** 11/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, el sistema muestra un listado de categorías y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles y Buscar categorías).	

Tabla 10: Tarea #2: Insertar nueva categoría

Tarea de Ingeniería de Usuario	
Número de la tarea: 2	**Número de Historia de Usuario:** 2
Nombre de la tarea: Insertar nueva categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 11/3/2015	**Fecha fin:** 12/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, la opción “Adicionar categoría”, el sistema muestra los campos nombre, descripción y categoría padre, se introducen los datos y se escoge una categoría padre sí posee. Seguido a esto se selecciona el botón aceptar, para registrar la nueva categoría en la Base de Datos de Vulnerabilidades. Si se selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 11: Tarea #3: Editar categoría

Tarea de Ingeniería de Usuario	
Número de la tarea: 3	**Número de Historia de Usuario:** 2
Nombre de la tarea: Editar categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 12/3/2015	**Fecha fin:** 13/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, seguidamente selecciona una categoría del listado de categorías y escoge la opción “Editar categoría”, luego el sistema muestra los datos de la categoría, brindando la posibilidad de actualizarlos. Terminado los cambios, se selecciona el botón editar para actualizar la información en la Base de Datos de Vulnerabilidades. En caso de que no se requiera modificar los datos, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Las restantes Tareas de Ingeniería definidas en la investigación podrán consultarse en el Anexo III.

3.6. Modelo de Datos

Para un mejor entendimiento del diseño de la base de datos, se puede observar en el siguiente diagrama el modelo de la base de datos del sistema, siendo sus elementos esenciales las entidades, los atributos y las relaciones entre las entidades.

3.6.1 Diagrama de Modelo de Datos

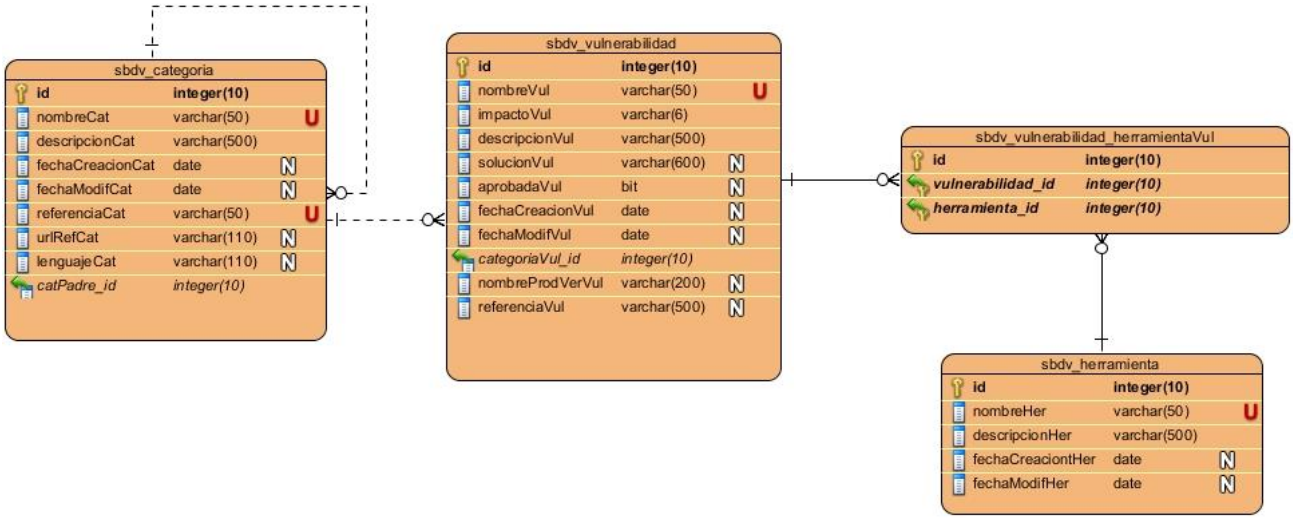


Figura 6. Modelo de Datos del sistema.

3.6.2 Descripción de las tablas de la base de datos

A continuación se muestran algunas de las descripciones de las tablas del Modelo de datos, para ver las restantes descripciones consultar el Anexo IV.

Tabla 12: Descripción de la tabla de la base de datos sbdv_vulnerabilidad

Tabla de la base de datos		
Nombre: sbdv_vulnerabilidad		
Descripción: Almacena toda la información de la vulnerabilidad.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la vulnerabilidad. Es la llave primaria de esta tabla.
nombreVul	varchar	Nombre único de la vulnerabilidad.
impactoVul	varchar	Nivel de daño que puede ocasionar.
descripcionVul	varchar	Descripción de la vulnerabilidad.
solucionVul	varchar	Representa la solución o soluciones para darle tratamiento a la vulnerabilidad.
aprobadaVul	bit	Estado en que se encuentra la vulnerabilidad.
fechaCreacionVul	date	Fecha en la que fue registrada.
fechaModifVul	date	Fecha en la que fue actualizada.
categoriaVul_id	integer	Id de la categoría a la que pertenece.
nombreProdVerVul	varchar	Productos que son afectados por la vulnerabilidad.
referenciaVul	varchar	Referencias en las que se pueden encontrar información de la vulnerabilidad.

Tabla 13: Descripción de la tabla de la base de datos sbdv_categoria

Tabla de la base de datos		
Nombre: sbdv_categoria		
Descripción: Almacena toda la información de la categoría		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la categoría. Es la llave primaria de esta tabla.

nombreCat	varchar	Nombre único de la categoría.
catPadre_id	integer	Categoría superior a la que pertenece.
descripcionCat	varchar	Descripción de la categoría.
fechaCreacionCat	date	Fecha en que se registró en el sistema.
fechaModifCat	date	Fecha en que se actualizó en el sistema.
referenciaCat	varchar	Nombre de la referencia.
urlRefCat	varchar	URL en la que se puede encontrar la referencia.
lenguajeCat	varchar	Lenguajes en los que se puede encontrar.

3.7. Diagrama de componentes

Los diagramas de componentes son utilizados para modelar las organizaciones y dependencias lógicas entre los componentes del software. Normalmente contienen componentes, interfaces y relaciones entre ellos, pero también pueden ser utilizados los paquetes, para lograr la agrupación de los elementos en el modelo (19).

En la siguiente figura se muestran los principales componentes del sistema siguiendo el patrón arquitectónico MTV utilizado por Django. El contenedor Model muestra los componentes correspondientes al modelo de los datos, el View los correspondientes a las vistas que se encuentran definidas y el Template los correspondientes a las plantillas que definirán como se mostrarán las vistas.

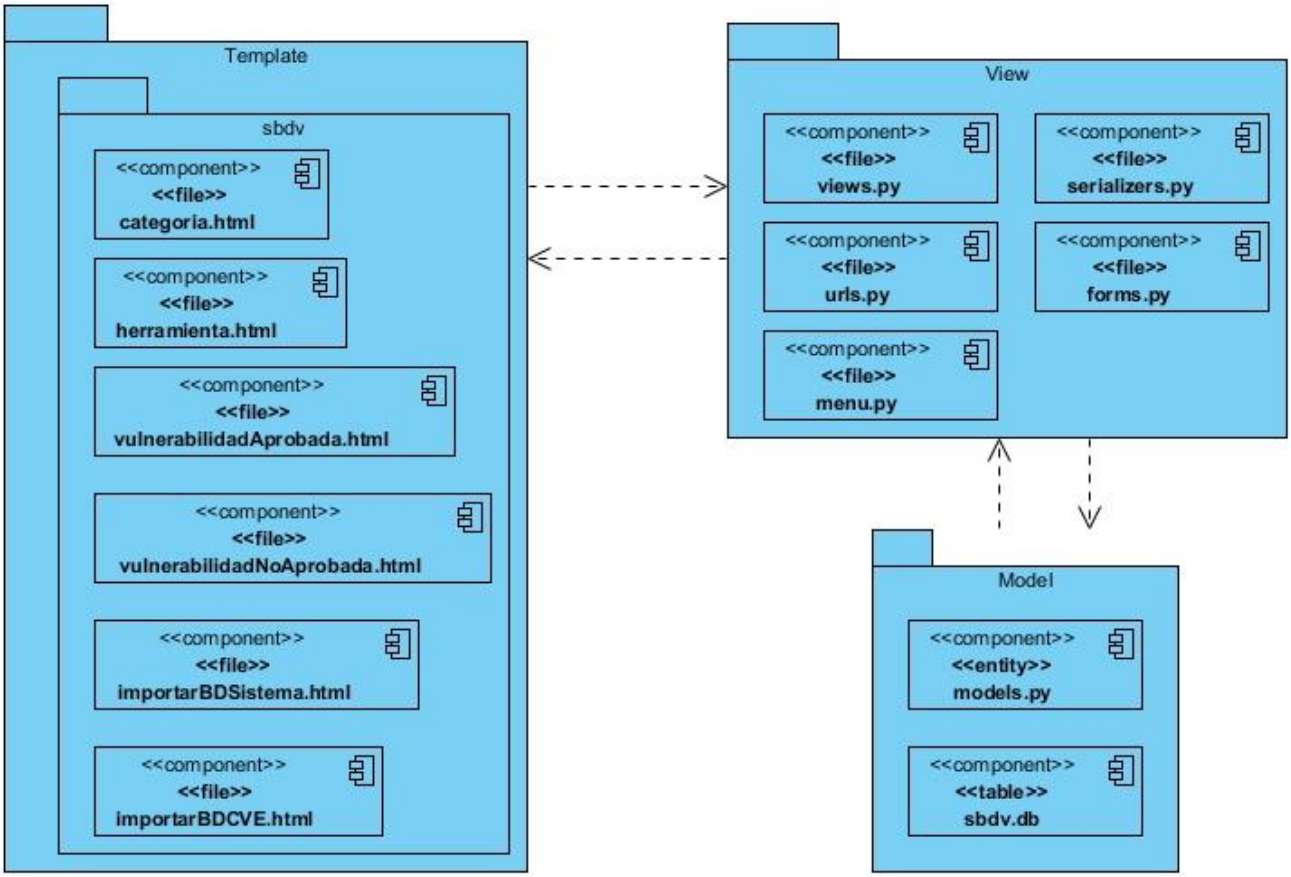


Figura 7: Diagrama de componentes.

3.8. Estándar de codificación

Las estrategias de codificación representan el conjunto de reglas que deben seguir los desarrolladores para escribir el código fuente de un programa. Las mismas permiten que el código sea legible por cualquier desarrollador del equipo de trabajo, garantizando una mayor agilidad en cuanto a reutilización y mantenimiento.

Para el desarrollo del sistema propuesto se utilizó el estándar de codificación Notación CamelCase para denotar variables, parámetros y métodos. Este estándar define, que si el identificador es una palabra simple, se escribe todo con minúscula, en caso de que sea compuesta, se escribe con minúscula la primera letra del identificador y las que vienen a continuación con mayúscula.

```
14      var editarHerramienta={
15          nombreHer:{title: gettext("Nombre de la Herramienta"), validators: ['required']},
16          descripcionHer:{title:gettext("Descripción de la Herramienta"),type:'TextArea'}
17      };
```

A continuación se muestran algunas reglas para la nomenclatura, basadas en el estándar a utilizar:

Identación:

Se emplearon cuatro espacios como unidad de identación.

```
10      var addHerramienta={
11          nombreHer:{title: gettext("Nombre de la Herramienta"), validators: ['required']},
12          descripcionHer:{title:gettext("Descripción de la Herramienta"),type:'TextArea'}
13      };
```

Líneas en blanco:

Se debe dejar 2 líneas en blanco antes y después de la declaración de una clase.

```
46  class ImportBDB(View):...
83
84
85  class ImportBDCVE(View):...
```

Se debe dejar una línea entre la implementación de funciones.

```
87      def get(self, *args, **kwargs):...
92
93      def post(self, *args, **kwargs):...
```

Espacios en blanco:

Se evitaron los espacios en blanco adicionales en expresiones y sentencias.

```
93      def post(self, *args, **kwargs):
94          form = UploadForm( self.request.POST, self.request.FILES)
95          if self.request.POST and form.is_valid():
96              import_file = self.request.FILES['archivo']
97              tree = ET.parse(import_file)
98              root = tree.getroot()
99              for prueba in root:
```

Comentarios:

Se utilizaron los comentarios de bloque limitados por """ <Comentarios> """.

```
42  """Las clases a continuación se encargan de importar las base de datos del
43  Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática y de CVE"""
```

Se utilizó el símbolo # para comentarios de una línea.

```
43      # Atributos de la clase Producto
44      class Producto(models.Model):
45          vulnerabilidadProd = models.ForeignKey(Vulnerabilidad)
46          nombreProd = models.CharField(max_length=200)
47          version = models.CharField(max_length=200)
```

Declaraciones:

Los métodos y variables se declararon con nombres asociados a la función por la cual fueron creados.

```
76      var editarVulnerabilidad = {
77          nombreVul: {title: gettext("Name"), validators: ['required']},
78          categoriaVul: {title: gettext("Category"),
79              type: 'Select',
80              options: new SelecionarCategorias(), validators: ['required']},
```

3.9. Conclusiones parciales

En este capítulo se definió como patrón de arquitectura a utilizar, el MTV. Se identificó como patrones de diseño GRASP el Creador, Controlador, Experto, Alta Cohesión y Bajo Acoplamiento. Se modelaron las tarjetas CRC de acuerdo con el diseño de clases especificado, además, se confeccionaron las tareas de la ingeniería correspondiente a las HU. La confección de las tarjetas CRC, permitió especificar las responsabilidades de las clases, así como las clases colaboradoras. Se obtuvieron las tablas del modelo de datos y se definió el estándar de codificación empleado en el desarrollo del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática.

CAPÍTULO 4: PRUEBAS DEL SISTEMA

4.1. Introducción

En el presente capítulo se realiza la descripción de la estrategia de pruebas aplicada al Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática con el objetivo de medir la calidad y el buen funcionamiento del mismo. Se describieron los resultados de la ejecución de las pruebas para lograr la evaluación del producto obtenido.

4.2. Estrategia de pruebas

Para la creación de la estrategia de pruebas se tuvo en cuenta que la metodología XP propone realizar la mayor cantidad de pruebas posibles, con el objetivo de detectar la mayor cantidad de problemas en las funcionalidades del sistema. Las pruebas a realizar son las definidas por dicha metodología, las pruebas unitarias y las pruebas de aceptación.

4.3. Pruebas unitarias

Constituyen una de las etapas imprescindibles en la metodología XP. Este tipo de pruebas son aplicadas y ejecutadas constantemente ante cada modificación del sistema. Son diseñadas por los programadores, para verificar que todo el código escrito funcione correctamente y posteriormente pueda ser publicado. En este sentido, el sistema y el conjunto de pruebas puede ser mejor utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo. (20)

Las pruebas unitarias fueron realizadas al concluir la implementación de cada una de las funcionalidades de las HU. Las mismas se aplicaron en dos iteraciones, mediante el uso del módulo unittest que ofrece el framework Django, reduciendo así el tiempo y el esfuerzo en esta fase.

A continuación se muestran los resultados de la primera iteración:

- Luego de terminada la implementación de la HU #1 se detectaron un total de dos no conformidades, tales como parámetros incorrectos en las funcionalidades Adicionar Vulnerabilidad y Editar Vulnerabilidad.
- Luego de terminada la implementación de la HU #2 se detectaron un total de dos no conformidades, tales como parámetros incorrectos en las funcionalidades Adicionar Categoría y Editar Categoría.
- Luego de terminada la implementación de la HU #3 no se detectó ninguna no conformidad.
- Luego de terminada la implementación de la HU #4 no se detectó ninguna no conformidad.

- Luego de terminada la implementación de la HU #5 se detectaron un total de seis no conformidades, tales como parámetros incorrectos, retorno de datos incorrectos en la funcionalidad Importar Base de Datos del Sistema.
- Luego de terminada la implementación de la HU #6 se detectaron un total de ocho no conformidades, tales como parámetros incorrectos, retorno de datos incorrectos en la funcionalidad Importar Base de Datos de CVE.

Después de ser aplicada la segunda iteración de las pruebas unitarias, no se detectó ninguna no conformidad. A continuación se ilustran los resultados de las pruebas unitarias en la siguiente gráfica:

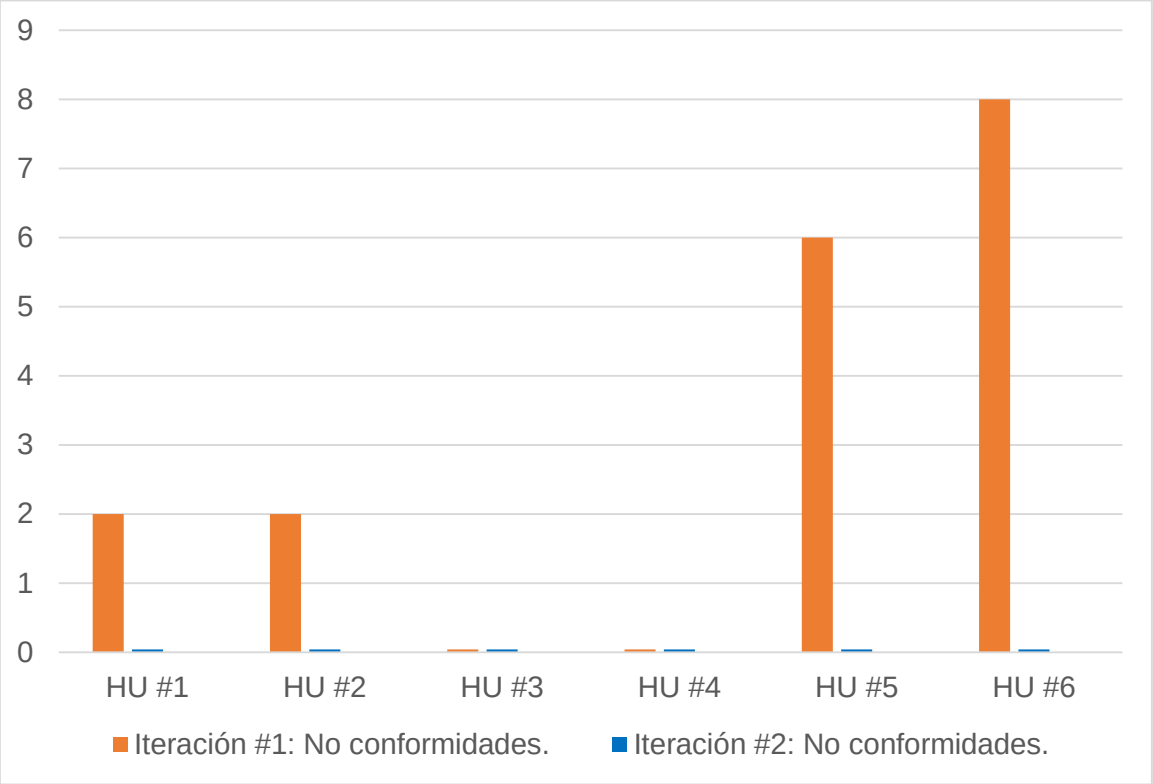


Figura 8: Resultado de las dos iteraciones de las pruebas unitarias.

A continuación se muestran los resultados obtenidos de las pruebas unitarias aplicadas a tres funcionalidades, mediante el uso del IDE Pycharm:

Test: se refiere a la ubicación donde se encuentra la prueba.

Time elapsed: se refiere al tiempo en que se demora ejecutar la prueba.

Results: se refiere al resultado de la prueba, devuelve uno si el código no tiene ningún error y cero en caso contrario.

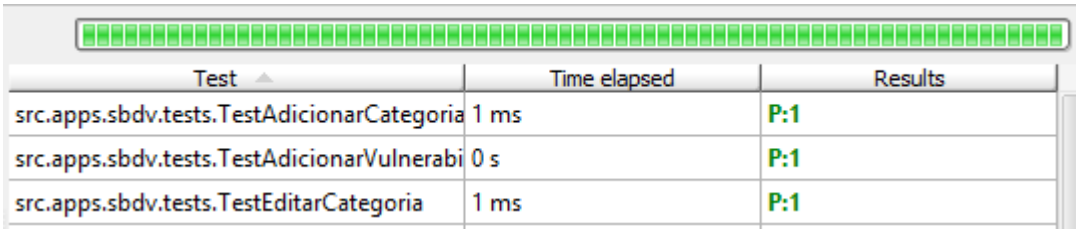


Figura 9: Resultados de la las pruebas unitarias obtenidas mediante el IDE Pycharm.

En la siguiente figura, se muestra la ubicación de la clase que contiene las restantes pruebas unitarias aplicadas al sistema:

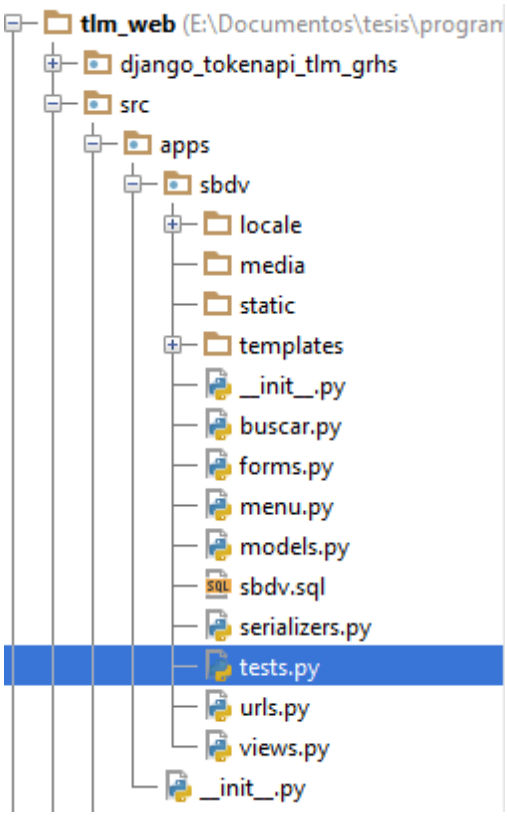


Figura 10: Ubicación de la clase que contiene las pruebas unitarias.

4.4. Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente es el responsable de especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. En caso de que fallen varias pruebas, el cliente es el responsable de indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta

información. Además las pruebas de aceptación son consideradas como Pruebas de Caja Negra[21] y son tan importantes como las pruebas unitarias, dado que significan la satisfacción del cliente con el producto desarrollado. (20)

Las pruebas de aceptación fueron planificadas en dos iteraciones, en la primera iteración se detectaron un total de quince no conformidades. A continuación se muestran los resultados de la primera iteración:

- Al culminar las pruebas de aceptación realizadas a la HU #1 se obtuvo como resultado cinco no conformidades, tales como errores al validar los datos introducidos de la vulnerabilidad y al realizar las búsquedas de vulnerabilidades.
- Al culminar las pruebas de aceptación realizadas a la HU #2 se obtuvo como resultado cuatro no conformidades, tales como errores al validar los datos introducidos de la categoría y al realizar las búsquedas de categorías.
- Al culminar las pruebas de aceptación realizadas a la HU #3 se obtuvo como resultado cuatro no conformidades, tales como errores al validar los datos introducidos de la herramienta.
- Al culminar las pruebas de aceptación realizadas a la HU #4 no se obtuvo ninguna no conformidad.
- Al culminar las pruebas de aceptación realizadas a la HU #5 se obtuvo como resultado una no conformidad, no se notificaba al especialista de que la operación de importar la base de datos del sistema puede demorar varios minutos.
- Al culminar las pruebas de aceptación realizadas a la HU #6 se obtuvo como resultado una no conformidad, no se notificaba al especialista de que la operación de importar la base de datos de CVE puede demorar varios minutos.

A continuación se ilustran los resultados de las pruebas de aceptación de la primera iteración en la siguiente gráfica:

[21] Pruebas de Caja Negra: permiten obtener un conjunto de condiciones de entrada que ejerciten todos los requisitos funcionales de un software.

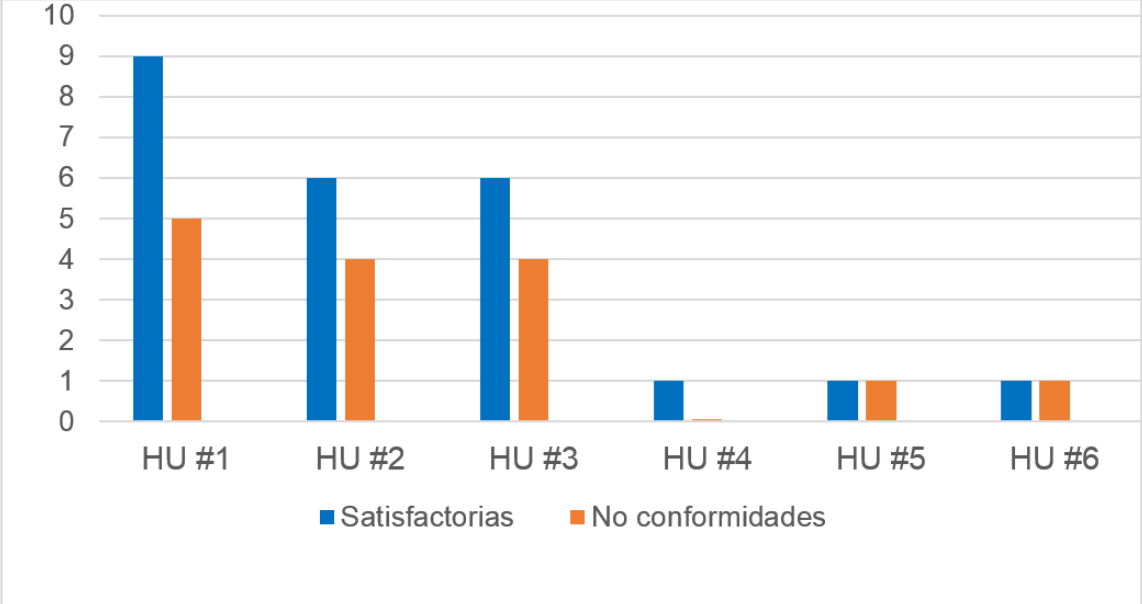


Figura 11: Resultado de la primera iteración de las pruebas de aceptación.

Las no conformidades detectadas se resolvieron en la segunda iteración, siendo los resultados obtenidos, los esperados por el cliente.

A continuación se muestran tres de los casos de prueba de aceptación realizados para la HU #2:

Tabla 14: Caso de prueba de aceptación: HU2_Tarea1

Caso de prueba de aceptación	
Código: HU2_Tarea1.	**Historia de Usuario:** Gestionar categorías.
Nombre: Listar categorías.	
Descripción: Muestra un listado con las categorías existentes en el sistema.	
Condiciones de ejecución: Existencia de categorías en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría.	
Resultado de la prueba: El sistema muestra un listado de categorías y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles y Buscar categorías).	
Evaluación de la prueba: Satisfactoria.	

Tabla 15: Caso de prueba de aceptación: HU2_Tarea2

Caso de prueba de aceptación	
Código: HU2_Tarea2.	**Historia de Usuario:** Gestionar categorías.
Nombre: Insertar nueva categoría.	
Descripción: Permite insertar una nueva categoría en la base de datos.	
Condiciones de ejecución: Llenar todos los campos obligatorios de la categoría.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría. ➢ El usuario presiona el botón “Adicionar categoría”. ➢ El usuario llena los campos de la categoría. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se insertará la categoría en la base de datos y se mostrará un mensaje que indica que se ha insertado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Las restantes pruebas de aceptación definidas en la investigación podrán consultarse en el Anexo V.

4.5. Conclusiones parciales

En este capítulo se comprobaron las funcionalidades del sistema mediante las pruebas de unidad y de aceptación. En dichas pruebas se detectaron un conjunto de no conformidades, a las cuales se les dieron solución en las posteriores iteraciones. De esta forma quedaron evaluadas las funcionalidades del sistema, lográndose así la satisfacción del cliente con el software realizado.

CONCLUSIONES

Con la realización de este trabajo investigativo se dio cumplimiento al objetivo propuesto y por tanto se arribó a las siguientes conclusiones:

- Se realizó un estudio de los sistemas que gestionan información referente a las vulnerabilidades de seguridad informática confirmándose que los mismos no brindan una solución al problema planteado.
- La selección de la metodología de desarrollo, lenguaje de programación, tecnologías y herramientas posibilitaron al equipo de desarrollo cumplir con el objetivo general de la investigación.
- Se realizó la implementación del Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática, cumpliéndose con los objetivos trazados por el equipo de desarrollo y con las especificaciones del cliente.
- Se registraron en el Sistema de Gestión de Información de Vulnerabilidades de Seguridad Informática las vulnerabilidades contenidas en la base de datos de CVE.
- Se realizaron pruebas unitarias y de aceptación, para verificar que el sistema informático propuesto reúne las condiciones tecnológicas para su uso.

RECOMENDACIONES

- Se recomienda realizar la futura integración con Xilema-PlatSI, para que una vez integrado se desarrollen funcionalidades que garanticen de manera automática la obtención de la información de las vulnerabilidades contenidas en el Sistema de Gestión de Vulnerabilidades de Seguridad Informática.
- Se recomienda la descarga periódica de la base de datos de vulnerabilidades de CVE, para que se mantenga la información del sistema lo más actualizada posible.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cisco Systems.** *Informe anual de seguridad.* 2014.

2. **Mark G. Graff, Kenneth R. van Wyk.** *Secure Coding: Principles & Practices.* [ed.] O'Reilly. 2003. pág. 224.

3. **MITRE.** Common Weakness Enumeration. [En línea] 2015. [Citado el: 16 de Febrero de 2015.] http://cwe.mitre.org/.

4. **Gordon Bitter Davis, Margrethe H. Olson.** *Management information systems: conceptual foundations, structure, and development.* s.l. : McGraw-Hill, 2007. 0070158282.

5. **Mitre.** CVE. [En línea] 2014. [Citado el: 12 de Diciembre de 2014.] http://cve.mitre.org/about/faqs.html.

6. **Open Sourced Vulnerability Database .** OSVDB. [En línea] 2014. [Citado el: 10 de Diciembre de 2014.] http://www.osvdb.org/.

7. **Secunia ApS.** Secunia. [En línea] 2014. [Citado el: 10 de Diciembre de 2014.] http://secunia.com/vulnerability_intelligence/faq/.

8. **INTECO (Instituto Nacional de Tecnologías de la Comunicación).** [aut. libro] Laboratorio Nacional de Calidad del Software. *Ingeniería del Software: Metodologías y Ciclos de Vida.* España : s.n., 2009.

9. **Patricio Letelier, Carmen Penadés.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n., 2015.

10. **Jquery.** Jquery. [En línea] 2015. [Citado el: 10 de Enero de 2015.] http://jquery.com/.

11. **PosgreSQL-es.** PostgreSQL-es. [En línea] 2013. [Citado el: 10 de Enero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

12. **Eguiluz, Javier.** LIBROSWEB. *LIBROSWEB.* [En línea] [Citado el: 15 de Febrero de 2015.] http://librosweb.es/javascript/capitulo_1.html.

13. **Hogan, Brian P.** *HTML5 and CSS3.* s.l. : Pragmatic Programmers, LLC, 2010.

14. **Victor Angel Fong Rios, Roger Armando González Rodríguez.** *Sistema de gestión de información para los resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2013.

15. **Pressman, Roger.** *Ingeniería de Software "Un Enfoque Práctico".* Quinta Edicion. Parte III. Capítulo 13.4.4: Arquitectura del software. s.l. : Mc Graw Hill, 2005. pág. 226.

16. **Adrian Holovaty, Jacob Kaplan Moss.** *The Definitive Guide to Django. Web Development Done Right.* United States of America : s.n., 2008.

17. **Tedeschi, Nicolás.** Microsoft. [En línea] 2015. [Citado el: 4 de Mayo de 2015.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

18. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

19. **Visual Paradigm.** VP Gallery. [En línea] 2015. [Citado el: 9 de Junio de 2015.] http://www.visual-paradigm.com/VPGallery/diagrams/Component.html.

20. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming.* 2008.

BIBLIOGRAFÍA

1. **Adrian Holovaty, Jacob Kaplan Moss.** *The Definitive Guide to Django. Web Development Done Right.* United States of America : s.n., 2008.
2. **Alegsa, Leandro.** ALEGSA. [En línea] 2014. [Citado el: 5 de Octubre de 2014.] http://www.alegsa.com.ar/Dic/vulnerabilidad.php.
3. **123 Innovation Group**. Auditoria Sistemas. [En línea] 2014. [Citado el: 6 de Enero de 2015.] http://www.auditoriasistemas.com/auditoria-informatica/.
4. **Backbone.org.** Backbone.org. [En línea] [Citado el: 1 de Junio de 2015.] www.backbone.org.
5. **Beck, Kent.** *Extreme programming explained: embrace change.* Estados Unidos : Addison-Wesley, 2000. 201-61641.
6. **Cisco Systems.** *Informe anual de seguridad.* 2014.
7. **Django Software Foundation.** Django, La guia Definitiva. 2012
8. **Eguiluz, Javier.** LIBROSWEB. *LIBROSWEB.* [En línea] [Citado el: 15 de Febrero de 2015.] http://librosweb.es/javascript/capitulo_1.html.
9. **Gordon Bitter Davis, Margrethe H. Olson.** *Management information systems: conceptual foundations, structure, and development.* s.l. : McGraw-Hill, 2007. 0070158282.
10. **Hipertextual.** Bitelia. [En línea] 2014. [Citado el: 13 de Enero de 2014.] http://bitelia.com/2013/05/entendiendo-html5-guia-para-principiantes.
11. **Hogan, Brian P.** *HTML5 and CSS3.* s.l. : Pragmatic Programmers, LLC, 2010.
12. **INTECO (Instituto Nacional de Tecnologías de la Comunicación).** [aut. libro] Laboratorio Nacional de Calidad del Software. *Ingeniería del Software: Metodologías y Ciclos de Vida.* España : s.n., 2009.
13. **Jeffries, Ronald E.** XPrograming. [En línea] 2015. [Citado el: 14 de Enero de 2015.] http://xprogramming.com/book/whatisxp/.
14. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming.* 2008.
15. **Jquery.** Jquery. [En línea] 2015. [Citado el: 10 de Enero de 2015.] http://jquery.com/.
16. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

17. **Mark G. Graff, Kenneth R. van Wyk.** *Secure Coding: Principles & Practices.* [ed.] O'Reilly. 2003. pág. 224.

18. **MITRE.** Common Weakness Enumeration. [En línea] 2015. [Citado el: 16 de Febrero de 2015.] http://cwe.mitre.org/.4.

19. **MITRE.** CVE. [En línea] 2014. [Citado el: 12 de Diciembre de 2014.] http://cve.mitre.org/about/faqs.html.

20. **Open Sourced Vulnerability Database .** OSVDB. [En línea] 2014. [Citado el: 10 de Diciembre de 2014.] http://www.osvdb.org/.

21. **OWASP Foundation.** *Guía de pruebas de OWASP.* 2008.

22. **OWASP Foundation.** *Testing Guide Foreword.* 2014.

23. **Patricio Letelier, Carmen Penadés.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n., 2015.

24. **PosgreSQL-es.** PostgreSQL-es. [En línea] 2013. [Citado el: 10 de Enero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

25. **Pressman, Roger.** *Ingeniería de Software "Un Enfoque Práctico".* Quinta Edicion. Parte III. Capítulo 13.4.4: Arquitectura del software. s.l. : Mc Graw Hill, 2005. pág. 226.

26. **Secunia ApS.** Secunia. [En línea] 2014. [Citado el: 10 de Diciembre de 2014.] http://secunia.com/vulnerability_intelligence/faq/.

27. **Tedeschi, Nicolás.** Microsoft. [En línea] 2015. [Citado el: 4 de Mayo de 2015.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

28. **Veracode.** VERACODE. [En línea] 2014. [Citado el: 20 de Noviembre de 2014.] http://www.veracode.com/security/vulnerability-management.

29. **Victor Angel Fong Rios, Roger Armando González Rodríguez.** *Sistema de gestión de información para los resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2013.

30. **Visual Paradigm.** VP Gallery. [En línea] 2015. [Citado el: 9 de Junio de 2015.] http://www.visual-paradigm.com/VPGallery/diagrams/Component.html.

GLOSARIO DE TÉRMINOS

Ajax: Del inglés Asynchronous JavaScript And XML, en español JavaScript asíncrono y XML. Conjunto de técnicas de desarrollo web para crear aplicaciones interactivas.

Aplicación Web: Es una aplicación de software que se codifica en un lenguaje soportado por los navegadores Web.

API: Del inglés Application Programming Interfaces, en español Interfaz de Programación de Aplicaciones. Es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

BPMN: Del inglés Business Process Modeling Notation, en español Notación para el Modelado de Procesos de Negocio, se utilizó para modelar el proceso de negocio en el análisis de las vulnerabilidades de seguridad informática.

DOM: Del inglés Document Object Model, en español Modelo de Objeto de Documento. Es la estructura de objetos que genera el navegador cuando se carga un documento y se puede alterar mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página.

Framework: Generadores de aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados. Además tienen la capacidad para promover la reutilización del código del diseño y el código fuente.

Herramienta CASE: Del inglés Computer Aided Software Engineering, en español Ingeniería de Software Asistida por Computadora. Son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

HTML: Del inglés Hyper Text Markup Language, en español Lenguaje de marcación de Hipertexto, se puede definir como el lenguaje que se utiliza para la elaboración de páginas Web.

IDE: Entorno de Desarrollo Integrado, es un programa compuesto por un conjunto de herramientas para un programador, además está compuesto por un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

JavaScript: Es un lenguaje de programación que permite a los desarrolladores crear efectos atractivos y dinámicos en las páginas web.

JSON: Del inglés JavaScript Object Notation, en español Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos.

LRP: Lista de Reserva del Producto, se refiere a la actividad de describir los requerimientos no funcionales del software.

MTV: Del inglés Model-Template-View, en español Modelo-Plantilla-Vista, patrón utilizado en el diseño y desarrollo web.

Open Source: Término con el que se conoce al software distribuido y desarrollado libremente.

Plugin: Pequeño programa que proporciona alguna funcionalidad específica a otra aplicación mayor o más compleja.

Seguridad Informática: Disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinadas a conseguir un sistema de información seguro y confiable.

UML: Del inglés Unified Modeling Language, en español Lenguaje Unificado de Modelado. Utilizado para modelar procesos y artefactos dentro del desarrollo de un software.

URL: Secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

Vulnerabilidades: Son puntos débiles del software que permiten que un atacante comprometa la integridad, disponibilidad o confidencialidad del mismo.

XML: Del inglés Extensible Markup Language; lenguaje de descripción de páginas de Internet, diseñado con la intención de reemplazar al estándar actual HTML.

XP: Es una metodología de desarrollo ágil centrada en potenciar las relaciones interpersonales como clave para obtener el éxito en el desarrollo de software, promoviendo el trabajo en equipo, la comunicación con el cliente, proporcionando a cada uno de estos un buen clima de trabajo.

ANEXOS

Anexo I: Historias de Usuario.

Tabla 16: HU #3 Gestionar las herramientas de prueba de seguridad

Historia de Usuario	
Número: 3	**Usuario:** Especialista
Nombre de HU: Gestionar las herramientas de prueba de seguridad.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1,2	**Iteración asignada:** 1
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado insertar, editar, eliminar, ver detalles y buscar herramientas de prueba de seguridad.	

Tabla 17: HU #4 Exportar la Base de Datos de vulnerabilidades de seguridad informática

Historia de Usuario	
Número: 4	**Usuario:** Especialista
Nombre de HU: Exportar la Base de Datos de vulnerabilidades de seguridad informática.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado exportar en un fichero JSON, la información contenida en la Base de Datos de Vulnerabilidades.	

Tabla 18: HU #5 Importar la Base de Datos de vulnerabilidades de seguridad informática

Historia de Usuario	
Número: 5	**Usuario:** Especialista
Nombre de HU: Importar la Base de Datos de vulnerabilidades de seguridad informática.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado importar mediante un fichero JSON los datos del sistema a la Base de Datos de Vulnerabilidades.	

Tabla 19: HU #6 Importar la Base de Datos de CVE

Historia de Usuario	
Número: 6	**Usuario:** Especialista
Nombre de HU: Importar la Base de Datos de CVE.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 5
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: Permite al usuario autenticado importar mediante un fichero XML los datos de las vulnerabilidades contenidos en la Base de Datos de Vulnerabilidades de CVE.	

Anexo II: Tarjetas CRC.

Tabla 20: Tarjeta CRC: Clase Herramienta

Datos de la Clase	
Nombre de la clase: Herramienta	
Responsabilidades	**Colaboradores**
Listar herramientas	Herramienta
Insertar nueva herramienta	Herramienta AdicionarHerramienta
Editar herramienta	Herramienta EditarHerramienta
Eliminar herramienta	Herramienta
Mostrar detalles de herramienta	
Buscar herramienta	

Tabla 21: Tarjeta CRC: Clase ExportarBDSistema

Datos de la Clase	
Nombre de la clase: ExportarBDSistema	
Responsabilidades	**Colaboradores**
Exportar la Base de Datos del sistema	Vulnerabilidad Categoria Herramienta

Tabla 22: Tarjeta CRC: Clase ImportBD

Datos de la Clase	
Nombre de la clase: ImportBD	
Responsabilidades	**Colaboradores**
Importar la Base de Datos del sistema.	FormularioImportar Vulnerabilidad Categoria Herramienta

Tabla 23: Tarjeta CRC: Clase ImportBDCVE

Datos de la Clase	
Nombre de la clase: ImportBDCVE	
Responsabilidades	**Colaboradores**
Importar la Base de Datos de CVE al sistema.	FormularioImportar Vulnerabilidad

Anexo III: Tareas de Ingeniería.

Tabla 24: Tarea #4: Eliminar categoría

Tarea de Ingeniería de Usuario	
Número de la tarea: 4	**Número de Historia de Usuario:** 2
Nombre de la tarea: Eliminar categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 13/3/2015	**Fecha fin:** 16/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, seguidamente selecciona una categoría del listado de categorías y escoge la opción “Eliminar categoría”, luego el sistema muestra un mensaje preguntando si desea eliminar la categoría especificada, si selecciona el botón aceptar, se elimina la categoría de la Base de Datos de Vulnerabilidades y si selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 25: Tarea #5: Mostrar detalles de categoría

Tarea de Ingeniería de Usuario	
Número de la tarea: 5	**Número de Historia de Usuario:** 2
Nombre de la tarea: Mostrar detalles de categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 16/3/2015	**Fecha fin:** 17/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, seguidamente selecciona una categoría del listado de categorías y escoge la opción “Ver detalles de categoría”, luego el sistema muestra los datos de la categoría.	

Tabla 26: Tarea #6: Buscar categoría

Tarea de Ingeniería de Usuario	
Número de la tarea: 6	**Número de Historia de Usuario:** 2
Nombre de la tarea: Buscar categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 17/3/2015	**Fecha fin:** 18/3/2015

Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.
Descripción: El especialista selecciona del menú Gestionar Categorías el submenú Categoría, seguidamente se le muestra el listado de categorías y un campo para que introduzca el criterio de búsqueda. Llenado este, se procede a presionar la tecla “Enter”, mostrándose las categorías correspondientes con el criterio de búsqueda.

Tabla 27: Tarea #7: Listar herramientas

Tarea de Ingeniería de Usuario	
Número de la tarea: 7	**Número de Historia de Usuario:** 3
Nombre de la tarea: Listar herramientas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 18/3/2015	**Fecha fin:** 19/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, el sistema muestra un listado de herramientas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles y Buscar herramientas).	

Tabla 28: Tarea #8: Insertar nueva herramienta

Tarea de Ingeniería de Usuario	
Número de la tarea: 8	**Número de Historia de Usuario:** 3
Nombre de la tarea: Insertar nueva herramienta.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 19/3/2015	**Fecha fin:** 20/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, la opción “Adicionar herramienta”, el sistema muestra los campos nombre y descripción, se introducen los datos y se selecciona el botón aceptar, para registrar la nueva herramienta en la Base de Datos de Vulnerabilidades. Si se selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 29: Tarea #9: Editar herramienta

Tarea de Ingeniería de Usuario

Número de la tarea: 9	**Número de Historia de Usuario:** 3
Nombre de la tarea: Editar herramienta.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 20/3/215	**Fecha fin:** 23/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, seguidamente selecciona una herramienta del listado de herramientas y escoge la opción “Editar herramienta”, luego el sistema muestra los datos de la herramienta, brindando la posibilidad de actualizarlos. Terminado los cambios, se selecciona el botón aceptar para actualizar la información en la Base de Datos de Vulnerabilidades. En caso de que no se requiera modificar los datos, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Tabla 30: Tarea #10: Eliminar herramienta

Tarea de Ingeniería de Usuario	
Número de la tarea: 10	**Número de Historia de Usuario:** 3
Nombre de la tarea: Eliminar herramienta.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 23/3/2015	**Fecha fin:** 24/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, seguidamente selecciona una herramienta del listado de herramientas y escoge la opción “Eliminar herramienta”, luego el sistema muestra un mensaje preguntando si desea eliminar la herramienta especificada, si selecciona el botón aceptar, se elimina la herramienta de la Base de Datos de Vulnerabilidades y si selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 31: Tarea #11: Mostrar detalles de herramienta

Tarea de Ingeniería de Usuario	
Número de la tarea: 11	**Número de Historia de Usuario:** 3
Nombre de la tarea: Mostrar detalles de herramienta.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 24/3/2015	**Fecha fin:** 25/3/2015

Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, seguidamente selecciona una herramienta del listado de herramientas y escoge la opción “Ver detalles de herramienta”, luego el sistema muestra los datos de la herramienta.

Tabla 32: Tarea #12: Buscar herramientas

Tarea de Ingeniería de Usuario	
Número de la tarea: 12	**Número de Historia de Usuario:** 1
Nombre de la tarea: Buscar herramientas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 25/3/2015	**Fecha fin:** 26/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Herramientas el submenú Herramienta, seguidamente se le muestra el listado de herramientas y un campo para que introduzca el criterio de búsqueda. Llenado este, se procede a presionar la tecla “Enter”, mostrándose las herramientas correspondientes con el criterio de búsqueda.	

Tabla 33: Tarea #13: Listar vulnerabilidades aprobadas

Tarea de Ingeniería de Usuario	
Número de la tarea: 13	**Número de Historia de Usuario:** 1
Nombre de la tarea: Listar vulnerabilidades aprobadas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 26/3/2015	**Fecha fin:** 27/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas, el sistema muestra un listado de vulnerabilidades aprobadas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles, Buscar, Desaprobar y Asignar herramientas de las vulnerabilidades).	

Tabla 34: Tarea #14: Listar vulnerabilidades no aprobadas

Tarea de Ingeniería de Usuario

Número de la tarea: 14	**Número de Historia de Usuario:** 1
Nombre de la tarea: Listar vulnerabilidades no aprobadas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 27/3/2015	**Fecha fin:** 30/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades No Aprobadas, el sistema muestra un listado de vulnerabilidades no aprobadas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles, Buscar, Aprobar y Asignar herramientas de las vulnerabilidades).	

Tabla 35: Tarea #15: Insertar nueva vulnerabilidad

Tarea de Ingeniería de Usuario	
Número de la tarea: 15	**Número de Historia de Usuario:** 1
Nombre de la tarea: Insertar nueva vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 30/3/2015	**Fecha fin:** 31/3/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguido a esto escoge la opción “Adicionar vulnerabilidad”, el sistema muestra los campos nombre, impacto, descripción, solución, producto, versión y categoría, se introducen los datos y se selecciona el botón aceptar, para registrar la nueva vulnerabilidad en la Base de Datos de Vulnerabilidades. Si se selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 36: Tarea #16: Editar vulnerabilidad

Tarea de Ingeniería de Usuario	
Número de la tarea: 16	**Número de Historia de Usuario:** 1
Nombre de la tarea: Editar vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 31/3/2015	**Fecha fin:** 1/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	

Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguidamente selecciona una vulnerabilidad del listado de vulnerabilidades y escoge la opción “Editar vulnerabilidad”, luego el sistema muestra los datos de la vulnerabilidad, brindando la posibilidad de actualizarlos. Terminado los cambios, se selecciona el botón aceptar para actualizar la información en la Base de Datos de Vulnerabilidades. En caso de que no se requiera modificar los datos, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.

Tabla 37: Tarea #17: Eliminar vulnerabilidad

Tarea de Ingeniería de Usuario	
Número de la tarea: 17	**Número de Historia de Usuario:** 1
Nombre de la tarea: Eliminar vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 1/4/2015	**Fecha fin:** 2/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguidamente selecciona una vulnerabilidad del listado de vulnerabilidades y escoge la opción “Eliminar vulnerabilidad”, luego el sistema muestra un mensaje preguntando si desea eliminar la vulnerabilidad especificada, si selecciona el botón aceptar, se elimina la vulnerabilidad de la Base de Datos de Vulnerabilidades y si selecciona el botón cancelar, no ocurre ningún evento en el sistema.	

Tabla 38: Tarea #18: Mostrar detalles de vulnerabilidad

Tarea de Ingeniería de Usuario	
Número de la tarea: 18	**Número de Historia de Usuario:** 1
Nombre de la tarea: Mostrar detalles de vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 2/4/2015	**Fecha fin:** 3/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguidamente	

selecciona una vulnerabilidad del listado de vulnerabilidades y escoge la opción “Ver detalles de vulnerabilidad”, luego el sistema muestra los datos de la vulnerabilidad.

Tabla 39: Tarea #19: Buscar vulnerabilidades

Tarea de Ingeniería de Usuario	
Número de la tarea: 19	**Número de Historia de Usuario:** 1
Nombre de la tarea: Buscar vulnerabilidades.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 3/4/2015	**Fecha fin:** 7/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguidamente se le muestra el listado de vulnerabilidades y un campo para que introduzca el criterio de búsqueda. Llenado este, se procede a presionar la tecla “Enter”, mostrándose las vulnerabilidades correspondientes con el criterio de búsqueda.	

Tabla 40: Tarea #20: Asignar herramientas a una vulnerabilidad

Tarea de Ingeniería de Usuario	
Número de la tarea: 20	**Número de Historia de Usuario:** 1
Nombre de la tarea: Asignar herramientas a una vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 7/4/2015	**Fecha fin:** 8/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona el menú Gestionar Vulnerabilidades el submenú Vulnerabilidades Aprobadas o el submenú Vulnerabilidades No Aprobadas, seguidamente selecciona una vulnerabilidad del listado de vulnerabilidades y escoge la opción “Asignar herramientas a vulnerabilidad”, luego el sistema muestra las herramientas asignadas, brindando la posibilidad de quitar o asignar herramientas, seguido a esto se escoge el botón aceptar para actualizar los cambios en la Base de Datos de Vulnerabilidades. En caso de no quererse realizar ningún cambio, se escoge el botón cancelar.	

Tabla 41: Tarea #21: Aprobar o desaprobar vulnerabilidad

Tarea de Ingeniería de Usuario

Número de la tarea: 21	**Número de Historia de Usuario:** 1
Nombre de la tarea: Aprobar o desaprobar vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 8/4/2015	**Fecha fin:** 9/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista superior selecciona del menú Gestionar Vulnerabilidades el submenú Vulnerabilidades No Aprobadas o Vulnerabilidades Aprobadas, seguidamente selecciona una vulnerabilidad del listado de vulnerabilidades y escoge la opción “Aprobar vulnerabilidad” o “Desaprobar vulnerabilidad” depende del estado en que se encuentre la vulnerabilidad, luego el sistema muestra el nombre de la vulnerabilidad y el estado en que se encuentra, brindando la posibilidad de cambiar el estado. Terminado los cambios, se selecciona el botón aceptar para actualizar la información en la Base de Datos de Vulnerabilidades. En caso de que no se requiera modificar el estado, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Tabla 42: Tarea #22: Exportar la Base de Datos de vulnerabilidades de seguridad informática

Tarea de Ingeniería de Usuario	
Número de la tarea: 22	**Número de Historia de Usuario:** 4
Nombre de la tarea: Exportar la Base de Datos de vulnerabilidades de seguridad informática.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 9/4/2015	**Fecha fin:** 23/4/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Exportar Base de Datos el submenú Exportar Base de Datos, seguidamente el sistema muestra los campos nombre y ubicación en la que se va a guardar la información, se especifican los campos y se escoge el botón aceptar, almacenando en un fichero JSON toda la información. En caso de que no se desee exportar la Base de Datos de Vulnerabilidades, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Tabla 43: Tarea #23: Importar la Base de Datos de vulnerabilidades de seguridad informática

Tarea de Ingeniería de Usuario	
Número de la tarea: 23	**Número de Historia de Usuario:** 5

Nombre de la tarea: Importar la Base de Datos de vulnerabilidades de seguridad informática.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 23/4/2015	**Fecha fin:** 7/5/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Importar Base de Datos el submenú Importar Base de Datos del Sistema, seguidamente el sistema muestra el campo ubicación, que es donde se va a encontrar el fichero a cargar, especificada la ubicación se selecciona el botón aceptar, actualizando toda la información del sistema. En caso de que no se desee importar, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Tabla 44: Tarea #24: Importar la Base de Datos de CVE

Tarea de Ingeniería de Usuario	
Número de la tarea: 24	**Número de Historia de Usuario:** 6
Nombre de la tarea: Importar la Base de Datos de CVE.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 7/5/2015	**Fecha fin:** 21/5/2015
Programador responsable: Antonio Veliz Santos y Daniel Pelaez Garcia.	
Descripción: El especialista selecciona del menú Importar Base de Datos el submenú Importar Base de Datos de CVE, seguidamente el sistema muestra el campo ubicación, que es donde se va a encontrar el fichero a cargar, especificada la ubicación se selecciona el botón aceptar, actualizando toda la información del sistema. En caso de que no se desee importar, se selecciona el botón cancelar, ocurriendo ningún evento en el sistema.	

Anexo IV: Descripción de las tablas del Modelo de Datos.

Tabla 45: Descripción de la tabla de la base de datos sbdv_herramienta

Tabla de la base de datos		
Nombre: sbdv_herramienta.		
Descripción: Almacena toda la información de la herramienta.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la herramienta. Es la llave primaria de esta tabla.
nombreHer	varchar	Nombre único de la herramienta.
descripcionHer	varchar	Descripción de la herramienta.
fechaCreacionHer	date	Fecha en que se registró en el sistema.
fechaModifHer	date	Fecha en que se actualizó en el sistema.

Tabla 46: Descripción de la tabla de la base de datos sbdv_vulnerabilidad_herramientaVul

Tabla de la base de datos		
Nombre: sbdv_vulnerabilidad_herramientaVul.		
Descripción: Almacena las llaves de la relación existente entre las tablas sbdv_vulnerabilidad y sbdv_herramienta.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de sbdv_vulnerabilidad_herramientaVul.
vulnerabilidad_id	integer	Identificador de la tabla sbdv_vulnerabilidad. Es llave primaria de esta tabla.
herramienta_id	integer	Identificador de la tabla sbdv_herramienta. Es llave primaria de esta tabla.

Anexo V: Casos de pruebas de aceptación.

Tabla 47: Caso de prueba de aceptación: HU2_Tarea3

Caso de prueba de aceptación	
Código: HU2_Tarea3.	**Historia de Usuario:** Gestionar categorías.
Nombre: Editar categoría.	
Descripción: Permite editar una categoría de la base de datos.	
Condiciones de ejecución: Existencia de la categoría en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría. ➢ El usuario selecciona la categoría. ➢ El usuario presiona el botón “Editar categoría”. ➢ El usuario modifica los datos de la categoría. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se actualizarán los datos modificados de la categoría en la base de datos y se mostrará un mensaje que indica que se ha editado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 48: Caso de prueba de aceptación: HU2_Tarea4

Caso de prueba de aceptación	
Código: HU2_Tarea4.	**Historia de Usuario:** Gestionar categorías.
Nombre: Eliminar categoría.	
Descripción: Permite eliminar una categoría de la base de datos.	
Condiciones de ejecución: Existencia de la categoría en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría. ➢ El usuario selecciona la categoría. ➢ El usuario presiona el botón “Eliminar categoría”.	

➢ El usuario presiona el botón Aceptar del mensaje de confirmación.
Resultado de la prueba: Se borrará la categoría de la base de datos y se mostrará un mensaje que indica que se ha eliminado satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 49: Caso de prueba de aceptación: HU2_Tarea5

Caso de prueba de aceptación	
Código: HU2_Tarea5.	**Historia de Usuario:** Gestionar categorías.
Nombre: Mostrar detalles de categoría.	
Descripción: Permite mostrar los datos de una categoría de la base de datos.	
Condiciones de ejecución: Existencia de la categoría en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría. ➢ El usuario selecciona la categoría. ➢ El usuario presiona el botón “Ver detalles de categoría”.	
Resultado de la prueba: Se mostrará toda la información de la categoría seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla 50: Caso de prueba de aceptación: HU2_Tarea6

Caso de prueba de aceptación	
Código: HU2_Tarea6.	**Historia de Usuario:** Gestionar categorías.
Nombre: Buscar categoría.	
Descripción: Permite buscar las categorías especificando un criterio de búsqueda.	
Condiciones de ejecución: Existencia de categorías en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Categorías. ➢ El usuario presiona la opción Categoría. ➢ El usuario selecciona el criterio de búsqueda. ➢ El usuario llena el campo de búsqueda y presiona la tecla “Enter”.	

Resultado de la prueba: Se mostrarán todas las categorías que cumplan con el criterio de búsqueda.
Evaluación de la prueba: Satisfactoria.

Tabla 51: Caso de prueba de aceptación: HU3_Tarea7

Caso de prueba de aceptación	
Código: HU2_Tarea7.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Listar herramientas.	
Descripción: Muestra un listado con las herramientas existentes en el sistema.	
Condiciones de ejecución: Existencia de herramientas en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta.	
Resultado de la prueba: El sistema muestra un listado de herramientas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles y Buscar herramientas).	
Evaluación de la prueba: Satisfactoria.	

Tabla 52: Caso de prueba de aceptación: HU3_Tarea8

Caso de prueba de aceptación	
Código: HU3_Tarea8.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Insertar nueva herramienta.	
Descripción: Permite insertar una nueva herramienta en la base de datos.	
Condiciones de ejecución: Llenar todos los campos obligatorios de la herramienta.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta. ➢ El usuario presiona el botón “Adicionar herramienta”.	

➢ El usuario llena los campos de la herramienta. ➢ El usuario presiona el botón Aceptar.
Resultado de la prueba: Se insertará la herramienta en la base de datos y se mostrará un mensaje que indica que se ha insertado satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 53: Caso de prueba de aceptación: HU3_Tarea9

Caso de prueba de aceptación	
Código: HU3_Tarea9.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Editar herramienta.	
Descripción: Permite editar una herramienta de la base de datos.	
Condiciones de ejecución: Existencia de la herramienta en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta. ➢ El usuario selecciona la herramienta. ➢ El usuario presiona el botón “Editar herramienta”. ➢ El usuario modifica los datos de la herramienta. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se actualizarán los datos modificados de la herramienta en la base de datos y se mostrará un mensaje que indica que se ha editado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 54: Caso de prueba de aceptación: HU3_Tarea10

Caso de prueba de aceptación	
Código: HU3_Tarea10.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Eliminar herramienta.	
Descripción: Permite eliminar una herramienta de la base de datos.	
Condiciones de ejecución: Existencia de la herramienta en la base de datos.	

Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta. ➢ El usuario selecciona la herramienta. ➢ El usuario presiona el botón “Eliminar herramienta”. ➢ El usuario presiona el botón Aceptar del mensaje de confirmación.
Resultado de la prueba: Se borrará la herramienta de la base de datos y se mostrará un mensaje que indica que se ha eliminado satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 55: Caso de prueba de aceptación: HU3_Tarea11

Caso de prueba de aceptación	
Código: HU3_Tarea11.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Mostrar detalles de herramienta.	
Descripción: Permite mostrar los datos de una herramienta de la base de datos.	
Condiciones de ejecución: Existencia de la herramienta en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta. ➢ El usuario selecciona la herramienta. ➢ El usuario presiona el botón “Ver detalles de herramienta”.	
Resultado de la prueba: Se mostrará toda la información de la herramienta seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla 56: Caso de prueba de aceptación: HU3_Tarea12

Caso de prueba de aceptación	
Código: HU3_Tarea12.	**Historia de Usuario:** Gestionar las herramientas de prueba de seguridad.
Nombre: Buscar herramienta.	

Descripción: Permite buscar las herramientas especificando un criterio de búsqueda.
Condiciones de ejecución: Existencia de herramientas en la base de datos.
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Herramientas. ➢ El usuario presiona la opción Herramienta. ➢ El usuario selecciona el criterio de búsqueda. ➢ El usuario llena el campo de búsqueda y presiona la tecla “Enter”.
Resultado de la prueba: Se mostrarán todas las herramientas que cumplan con el criterio de búsqueda.
Evaluación de la prueba: Satisfactoria.

Tabla 57: Caso de prueba de aceptación: HU1_Tarea13

Caso de prueba de aceptación	
Código: HU1_Tarea13.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Listar vulnerabilidades aprobadas.	
Descripción: Muestra un listado con las vulnerabilidades aprobadas del sistema.	
Condiciones de ejecución: Existencia de vulnerabilidades aprobadas en la base de datos.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas.	
Resultado de la prueba: El sistema muestra un listado de vulnerabilidades aprobadas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles, Buscar y Asignar herramientas).	
Evaluación de la prueba: Satisfactoria.	

Tabla 58: Caso de prueba de aceptación: HU1_Tarea14

Caso de prueba de aceptación	
Código: HU1_Tarea14.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.

Nombre: Listar vulnerabilidades no aprobadas.
Descripción: Muestra un listado con las vulnerabilidades no aprobadas del sistema.
Condiciones de ejecución: Existencia de vulnerabilidades no aprobadas en la base de datos.
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas.
Resultado de la prueba: El sistema muestra un listado de vulnerabilidades no aprobadas y las diferentes acciones posibles a realizar asociadas a las mismas (Insertar, Editar, Eliminar, Mostrar detalles, Buscar y Asignar herramientas).
Evaluación de la prueba: Satisfactoria.

Tabla 59: Caso de prueba de aceptación: HU1_Tarea15

Caso de prueba de aceptación	
Código: HU1_Tarea15.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Insertar nueva vulnerabilidad.	
Descripción: Permite insertar una nueva vulnerabilidad en la base de datos.	
Condiciones de ejecución: Llenar todos los campos obligatorios de la vulnerabilidad.	
Entrada/Pasos de ejecución 1: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas. ➢ El usuario presiona el botón “Adicionar vulnerabilidad”. ➢ El usuario llena los campos de la vulnerabilidad. ➢ El usuario presiona el botón Aceptar. **Entrada/Pasos de ejecución 2:** ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas. ➢ El usuario presiona el botón “Adicionar vulnerabilidad”. ➢ El usuario llena los campos de la vulnerabilidad.	

➢ El usuario presiona el botón Aceptar.
Resultado de la prueba: Se insertará la vulnerabilidad en la base de datos y se mostrará un mensaje que indica que se ha insertado satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

Tabla 60: Caso de prueba de aceptación: HU1_Tarea16

Caso de prueba de aceptación	
Código: HU1_Tarea16.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Editar vulnerabilidad.	
Descripción: Permite editar una vulnerabilidad de la base de datos.	
Condiciones de ejecución: Existencia de la vulnerabilidad en la base de datos.	
Entrada/Pasos de ejecución 1: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Editar vulnerabilidad”. ➢ El usuario modifica los datos de la vulnerabilidad. ➢ El usuario presiona el botón Aceptar. **Entrada/Pasos de ejecución 2:** ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Editar vulnerabilidad”. ➢ El usuario modifica los datos de la vulnerabilidad. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se actualizarán los datos modificados de la vulnerabilidad en la base de datos y se mostrará un mensaje que indica que se ha editado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 61: Caso de prueba de aceptación: HU1_Tarea17

Caso de prueba de aceptación	
Código: HU1_Tarea17.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Eliminar vulnerabilidad.	
Descripción: Permite eliminar una vulnerabilidad de la base de datos.	
Condiciones de ejecución: Existencia de la vulnerabilidad en la base de datos.	
Entrada/Pasos de ejecución 1: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Eliminar vulnerabilidad”. ➢ El usuario presiona el botón Aceptar del mensaje de confirmación. **Entrada/Pasos de ejecución 2:** ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Eliminar vulnerabilidad”. ➢ El usuario presiona el botón Aceptar del mensaje de confirmación.	
Resultado de la prueba: Se borrará la vulnerabilidad de la base de datos y se mostrará un mensaje que indica que se ha eliminado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 62: Caso de prueba de aceptación: HU1_Tarea18

Caso de prueba de aceptación	
Código: HU1_Tarea18.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Mostrar detalles de vulnerabilidad.	
Descripción: Permite mostrar los datos de una vulnerabilidad de la base de datos.	
Condiciones de ejecución: Existencia de la vulnerabilidad en la base de datos.	
Entrada/Pasos de ejecución 1:	

- El usuario selecciona del menú principal la opción Vulnerabilidades.
- El usuario presiona el submenú Gestionar Vulnerabilidades.
- El usuario presiona la opción Vulnerabilidades Aprobadas.
- El usuario selecciona la vulnerabilidad.
- El usuario presiona el botón “Ver detalles de vulnerabilidad”.

Entrada/Pasos de ejecución 2:

- El usuario selecciona del menú principal la opción Vulnerabilidades.
- El usuario presiona el submenú Gestionar Vulnerabilidades.
- El usuario presiona la opción Vulnerabilidades No Aprobadas.
- El usuario selecciona la vulnerabilidad.
- El usuario presiona el botón “Ver detalles de vulnerabilidad”.

Resultado de la prueba: Se mostrará toda la información de la vulnerabilidad seleccionada.

Evaluación de la prueba: Satisfactoria.

Tabla 63: Caso de prueba de aceptación: HU1_Tarea19

Caso de prueba de aceptación	
Código: HU1_Tarea19.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Buscar vulnerabilidad.	
Descripción: Permite buscar las vulnerabilidades especificando un criterio de búsqueda.	
Condiciones de ejecución: Existencia de vulnerabilidades en la base de datos.	

Entrada/Pasos de ejecución 1:

- El usuario selecciona del menú principal la opción Vulnerabilidades.
- El usuario presiona el submenú Gestionar Vulnerabilidades.
- El usuario presiona la opción Vulnerabilidades Aprobadas.
- El usuario selecciona el criterio de búsqueda.
- El usuario llena el campo de búsqueda y presiona la tecla “Enter”.

Entrada/Pasos de ejecución 2:

- El usuario selecciona del menú principal la opción Vulnerabilidades.
- El usuario presiona el submenú Gestionar Vulnerabilidades.
- El usuario presiona la opción Vulnerabilidades No Aprobadas.
- El usuario selecciona el criterio de búsqueda.

➢ El usuario llena el campo de búsqueda y presiona la tecla “Enter”.
Resultado de la prueba: Se mostrarán todas las vulnerabilidades que cumplan con el criterio de búsqueda.
Evaluación de la prueba: Satisfactoria.

Tabla 64: Caso de prueba de aceptación: HU1_Tarea20

Caso de prueba de aceptación	
Código: HU1_Tarea20.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Asignar herramientas a una vulnerabilidad.	
Descripción: Permite quitar o asignar herramientas a una vulnerabilidad.	
Condiciones de ejecución: Existencia de vulnerabilidades en la base de datos.	
Entrada/Pasos de ejecución 1: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Asignar herramientas a vulnerabilidad”. ➢ El usuario escoge las herramientas de la vulnerabilidad. ➢ El usuario presiona el botón Aceptar. **Entrada/Pasos de ejecución 2:** ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Asignar herramientas a vulnerabilidad”. ➢ El usuario escoge las herramientas de la vulnerabilidad. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se asignarán las herramientas a la vulnerabilidad seleccionada y se mostrará un mensaje que indica que se ha asignado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 65: Caso de prueba de aceptación: HU1_Tarea21

Caso de prueba de aceptación	
Código: HU1_Tarea21.	**Historia de Usuario:** Gestionar vulnerabilidad de seguridad informática.
Nombre: Aprobar o desaprobar vulnerabilidad.	
Descripción: Permite cambiar el estado en que se encuentra una determinada vulnerabilidad.	
Condiciones de ejecución: Existencia de vulnerabilidades en la base de datos.	
Entrada/Pasos de ejecución 1: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Desaprobar vulnerabilidad”. ➢ El usuario modifica el estado. ➢ El usuario presiona el botón Aceptar. **Entrada/Pasos de ejecución 2:** ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Gestionar Vulnerabilidades. ➢ El usuario presiona la opción Vulnerabilidades No Aprobadas. ➢ El usuario selecciona la vulnerabilidad. ➢ El usuario presiona el botón “Aprobar vulnerabilidad”. ➢ El usuario modifica el estado. ➢ El usuario presiona el botón Aceptar.	
Resultado de la prueba: Se modificará el estado de la vulnerabilidad y se mostrará un mensaje que indica que se ha modificado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 66: Caso de prueba de aceptación: HU4_Tarea22

Caso de prueba de aceptación	
Código: HU4_Tarea22.	**Historia de Usuario:** Exportar la Base de Datos de vulnerabilidades de seguridad informática.
Nombre: Exportar la Base de Datos de vulnerabilidades de seguridad informática.	

Descripción: Permite almacenar en un fichero JSON toda la información de la base de datos.
Condiciones de ejecución: Ninguna.
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Exportar Base de Datos. ➢ El usuario presiona la opción Exportar Base de Datos. ➢ El usuario selecciona “Guardar archivo” ➢ El usuario presiona el botón Aceptar. ➢ El usuario especifica el nombre y la ubicación en la que se va a guardar la información. ➢ El usuario presiona el botón Aceptar.
Resultado de la prueba: Se almacenará en un fichero JSON la información contenida en la base de datos.
Evaluación de la prueba: Satisfactoria.

Tabla 67: Caso de prueba de aceptación: HU5_Tarea23

Caso de prueba de aceptación	
Código: HU5_Tarea23.	**Historia de Usuario:** Importar la Base de Datos de vulnerabilidades de seguridad informática.
Nombre: Importar la Base de Datos de vulnerabilidades de seguridad informática.	
Descripción: Permite cargar hacia la base de datos, la información del sistema almacenada en un fichero con formato JSON.	
Condiciones de ejecución: Ninguna.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Importar Base de Datos. ➢ El usuario presiona la opción Importar Base de Datos del Sistema. ➢ El usuario escoge la ubicación en la que se encuentra el fichero. ➢ El usuario presiona el botón Importar Archivo JSON.	
Resultado de la prueba: Se actualizará la información contenida en el sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 68: Caso de prueba de aceptación: HU6_Tarea24

Caso de prueba de aceptación	
Código: HU6_Tarea24.	**Historia de Usuario:** Importar la Base de Datos de CVE.
Nombre: Importar la Base de Datos de CVE.	
Descripción: Permite cargar hacia la base de datos, la información de la vulnerabilidades de CVE almacenadas en un fichero con formato XML.	
Condiciones de ejecución: Ninguna.	
Entrada/Pasos de ejecución: ➢ El usuario selecciona del menú principal la opción Vulnerabilidades. ➢ El usuario presiona el submenú Importar Base de Datos. ➢ El usuario presiona la opción Importar Base de Datos de CVE. ➢ El usuario escoge la ubicación en la que se encuentra el fichero. ➢ El usuario presiona el botón Importar CVE.	
Resultado de la prueba: Se agregarán las vulnerabilidades que no estén contenidas en la base de datos.	
Evaluación de la prueba: Satisfactoria.	