Facultad 2

Sistema para la identificación de vulnerabilidades de Seguridad en Sistemas Gestores de Base de Datos

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

Autor(es):

Luhana Castillo Cribeiro,

Davier Jesús Carballo Mena

Tutor(es):

Ing. Roberto Antonio Infante Milanés

Dr.C. Yunia Reyes González

La Habana, 7 de diciembre de 2023

"Año 65 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los _7_ días del mes _12_ del año __2023__.

_____ _____

Luhana Castillo Cribeiro
(Autor)

Davier Jesús Carballo Mena
(Autor)

_____ _____

Ing. Roberto Antonio Infante Milanés
(Tutor)

Dr.C. Yunia Reyes González
(Tutor)

RESUMEN

Las bases de datos son una herramienta fundamental para almacenar y procesar información. Estas a menudo contienen información sensible de una organización, lo que las expone a una amplia gama de ataques contra su seguridad. De esta manera y más allá de que requieran o no de mayor nivel de confidencialidad en el tratamiento y acceso a la información que contienen, las bases de datos deben estar protegidas de los múltiples riesgos a los que se enfrentan, principalmente los asociados a la pérdida, alteración y robo de información. Para poder darle un tratamiento más efectivo a la mitigación de vulnerabilidades en Sistemas Gestores de Bases de Datos, es importante poder tenerlas identificadas para concentrar los esfuerzos y recursos en los elementos más críticos. El objetivo fundamental de la investigación es desarrollar un sistema informático para la identificación de vulnerabilidades de seguridad en Sistemas Gestores de Bases de Datos encaminado a garantizar la confidencialidad, integridad y disponibilidad de los datos almacenados. La implementación de solución propuesta por la que fue regida para la identificación, modelación y descripción de los procesos del negocio por el uso de la metodología XP, la utilización de los lenguajes HTML5, CSSv3 y Java script. Para la validación del sistema se realizaron pruebas de software evidenciándose la correspondencia entre los objetivos propuestos y los resultados obtenidos.

PALABRAS CLAVES: Seguridad de Base de Datos, vulnerabilidad, sistema informático, incidencia.

ABSTRACT

Databases are a fundamental tool for storing and processing information. These often contain an organization's sensitive information, exposing them to a wide range of security attacks. In this way and regardless of whether or not they require a higher level of confidentiality in the treatment and access to the information they contain, databases must be protected from the multiple risks they face, mainly those associated with loss, alteration and theft of information. In order to be able to give more effective treatment to the mitigation of vulnerabilities in Database Management Systems, it is important to be able to have them identified to concentrate efforts and resources on the most critical elements. The fundamental objective of the research is to develop a computer system for the identification of security vulnerabilities in Database Management Systems aimed at guaranteeing the confidentiality, integrity and availability of the stored data. The implementation of the proposed solution was guided by the identification, modeling and description of business processes through the use of the XP methodology, the use of HTML5, CSSv3 and Java script languages. To validate the system, software tests were carried out, demonstrating the correspondence between the proposed objectives and the results obtained.

KEYWORDS: Database Security, vulnerability, computer system, incidence.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La información es de vital importancia para las organizaciones, ya que les permite tomar decisiones estratégicas, mejorar la eficiencia operativa, identificar oportunidades de crecimiento y mantener una ventaja competitiva en el mercado. La información proporciona datos relevantes y actualizados que permiten a los directivos y empleados tomar decisiones acertadas. Sin información, las decisiones se basarían en suposiciones y conjetura, lo que aumentaría el riesgo de cometer errores costosos. Esta ayuda a los centros a planificar sus actividades y establecer metas realistas. Además, permite monitorear y evaluar el desempeño en los negocios. La misma puede ser utilizada como una ventaja competitiva al permitir identificar oportunidades de mercado, comprender las necesidades, y desarrollar productos y servicios que satisfagan esas necesidades de manera más efectiva que la competencia.

La importancia de la información, es cada día más creciente en la gerencia de las organizaciones, y tiene dentro de sus objetivos básicos, mantenerse y perdurar en el mercado. Durante los últimos años los sistemas de información, constituyen uno de los principales ámbitos de estudio en el área de organización de empresas.

Sin duda alguna, las Tecnologías de la Información y la Comunicación (TIC) han transformado de manera vertiginosa la vida cotidiana y social de los seres humanos. Estas transformaciones han ido permeando los ámbitos profesionales y educativos para facilitar desempeños en varias áreas, una de ellas tiene que ver con el acceso a la información (Arbeláez Gómez, 2014).

El uso tedioso de papel para recoger datos y la lentitud a la hora de encontrar un dato concreto dio lugar al tratamiento automatizado de la información. Así surgen las bases de datos. “La mayoría de los datos mantenidos en una base de datos están ahí para satisfacer alguna necesidad de la organización” (Beynon-Davies, 2018).

En la actualidad existen diferentes tipos de vulnerabilidades que ponen en riesgo los sistemas de información, por ejemplo, las vulnerabilidades de día cero, de aplicaciones web y de gestores de bases de datos entre otros. La seguridad de las bases de datos es una preocupación constante y a menudo se ve comprometida debido a la configuración de los procesos de conexión.

La implementación de un SGBD implica ajustar un conjunto de configuraciones. Algunas de estas configuraciones se utilizan para definir los diversos procesos que el SGBD llevará a cabo, mientras que otras están relacionadas con la asignación de memoria para realizar transacciones y ejecutar consultas

en los modelos de datos. Por último, también se deben considerar los ajustes necesarios para gestionar las conexiones a la base de datos, tanto desde ubicaciones remotas como locales.

La seguridad de la información es de vital importancia en las organizaciones ya que las organizaciones manejan una gran cantidad de información confidencial, como datos de clientes, proveedores, empleados y transacciones financieras. La seguridad de la información garantiza que estos datos estén protegidos de accesos no protegidos de accesos no autorizado y evita su robo, eliminación o alteración. Ayuda con la continuidad del negocio ya que la pérdida o corrupción de datos críticos puede interrumpir las operaciones normales de una organización.

Las organizaciones operan con una red de sistemas de gestión de bases de datos para administrar información sensible, como pueden ser cuentas de clientes, transacciones y registros de inversiones. A pesar de sus esfuerzos por mantener una sólida postura de seguridad, se enfrentan a una serie de desafíos críticos en relación con la integridad y confidencialidad de sus datos. Actualmente se ha percibido un aumento en las amenazas cibernéticas dirigidas a sus bases de datos. Los ataques de inyección SQL y la explotación de vulnerabilidades conocidas en sus SGBD son una preocupación constante. Esto ha llevado a incidentes de seguridad, como la manipulación de datos financieros y el acceso no autorizado a cuentas de clientes.

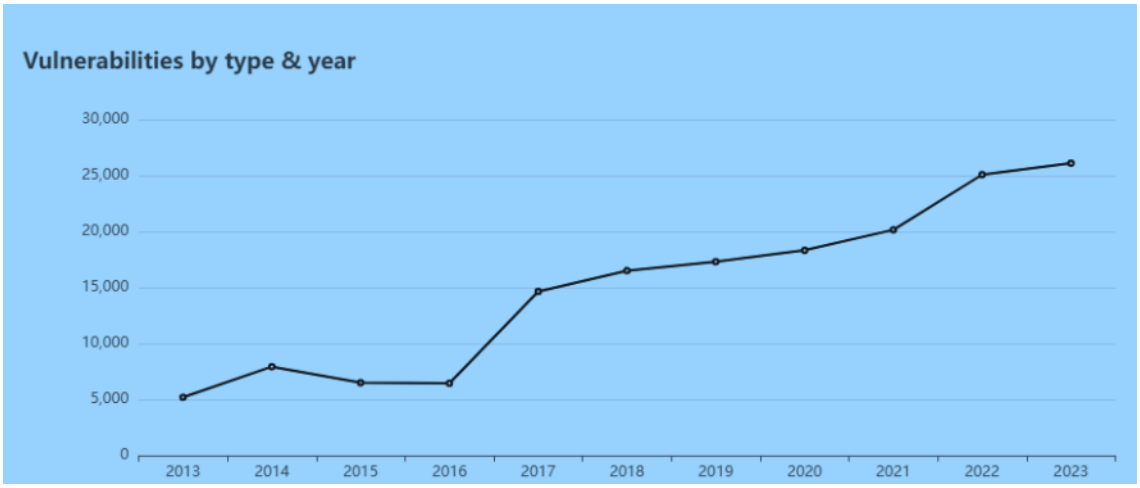


Figura 1. Aumento de las vulnerabilidades por año

Dada la existencia de estos desafíos, surge la necesidad de gestionar los incidentes relacionados con vulnerabilidades en SGBD, con el propósito de informar a los administradores y a otros interesados sobre posibles brechas de seguridad. Este esfuerzo busca destacar posibles debilidades y amenazas que podrían afectar a los sistemas, así como proporcionar estrategias para mitigar y contrarrestar posibles ataques. Esta situación destaca la importancia de contar con mecanismos de seguridad de

bases de datos sólidos y efectivos para proteger la información crítica. La identificación proactiva de vulnerabilidades es esencial para prevenir incidentes de seguridad y mantener la confianza de los clientes. Una de las claves que garantizan el éxito en el reto de lograr bases de datos más seguras, radica en poder identificar las vulnerabilidades que presentan las mismas. Sin embargo, en muchas ocasiones los especialistas encargados de realizar estas tareas, no poseen todos los conocimientos y habilidades que les permitan poder ejecutar a profundidad un análisis y detección de las amenazas. Esto ocurre, porque en la mayoría de las organizaciones y sobre todo en Cuba, no se logra una permanencia de estos especialistas en los principales sectores de las TIC, por lo que no se sostiene en el tiempo, una fuerza laboral calificada y con la experiencia necesaria. De esta manera, la toma de decisiones respecto a cuál o cuáles sistemas de bases de datos, brindarle la mayor atención para solucionar los problemas de seguridad, se torna más engorrosa y puede llevar a tomar decisiones menos efectivas.

Por lo antes mencionado se plantea como **problema de investigación** ¿Cómo contribuir a la prevención de incidentes de seguridad en SGBD? Se definiendo como **objeto de estudio** las vulnerabilidades en Sistemas Gestores de Bases de Datos.

El **campo de acción** lo constituye mitigación de vulnerabilidades en Sistemas Gestores de Bases de Datos. El **objetivo general** del presente trabajo es el desarrollo de un sistema informático para la identificación de vulnerabilidades de seguridad en SGBD encaminado a garantizar la confidencialidad, integridad y disponibilidad de los datos almacenados.

Para darle cumplimiento al objetivo general, se definen las siguientes **tareas de investigación:**

1. Identificación de los elementos asociados al problema de investigación.
2. Realización del diseño teórico metodológico de la investigación.
3. Análisis de los documentos, normativas y regulaciones asociadas a las vulnerabilidades de seguridad en SGBD.
4. Análisis de los referentes actuales, de sistemas para la identificación de vulnerabilidades en SGBD.
5. Selección de las metodologías, lenguajes y tecnologías de desarrollo que serán utilizadas para la implementación de la propuesta de solución.
6. Elaboración del Capítulo I.
7. Levantamiento de las funcionalidades que debe proveer la propuesta de solución.

8. Diseño de los artefactos ingenieriles de la propuesta de solución, de acuerdo a la metodología seleccionada.
9. Elaboración del Capítulo II.
10. Implementación de la propuesta de solución.
11. Diseño de los casos de pruebas.
12. Ejecución de los casos de pruebas.
13. Elaboración del Capítulo III.

Para la realización de este trabajo se utilizan los métodos científicos de la investigación, como es el caso de los **métodos teóricos** y los **métodos empíricos**.

Métodos teóricos

- **Analítico-sintético** se emplea para indagar profundamente en los fundamentos de los fenómenos, identificando sus características distintivas y esenciales. En el contexto de esta investigación, el propósito de aplicar este método es examinar las teorías, documentos y otros recursos relacionados directamente con las estrategias y las vulnerabilidades en sistemas gestores de bases de datos. Esto nos permite extraer de manera efectiva los elementos más relevantes que guardan relación con el objeto de estudio.
- **Histórico-lógico** se emplea para examinar el comportamiento de nuestra situación problemática realizando la recopilación de fuentes primarias y secundarias relacionadas con el objeto de estudio y es aquí donde se realiza un análisis lógico de estos datos para ver las causas, consecuencias o evaluar la validez.
- **Revisión documental** se emplea para recopilar, analizar y sintetizar información relevante sobre algún tema específico, por ejemplo (el concepto de base de datos). Este proporciona una base sólida para el estudio.

Métodos Empíricos

- **Entrevista** se emplea una entrevista **no estructurada** para obtener información detallada y profunda sobre las experiencias, opiniones y perspectivas para generar nuevas ideas.

El documento se encuentra estructurado en: Resumen, introducción, 3 capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos.

Capítulo 1: Fundamentos teóricos de la investigación: En este capítulo se realiza un estudio a fondo de los conceptos fundamentales relacionados con nuestro objeto de estudio, se buscan los sistemas homólogos según los indicadores para cumplir con el problema de investigación, se describe la metodología mejor acorde al desarrollo del software y las herramientas y lenguaje que serán utilizados en el mismo.

Capítulo 2: Análisis y diseño de la propuesta de solución: En este capítulo, se describe la solución propuesta para la identificación de vulnerabilidades de seguridad en los sistemas de gestión de bases de datos. Se ofrece una descripción exhaustiva de este proceso, junto con su modelado correspondiente. Asimismo, se esboza la estructura modular del sistema y se definen los roles necesarios para su implementación y funcionamiento eficaz.

Capítulo 3: Implementación y pruebas: En este capítulo es en donde ya se describe la fase de implementación del proyecto y se desarrolla el sistema, además de la estrategia seguida para la realización de determinadas pruebas para validar y verificar su correcto funcionamiento.

1. CAPÍTULO I: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

1.1. Introducción

En este capítulo se aborda de manera exhaustiva los elementos fundamentales que sustentan la investigación en cuestión. Se exploran en detalle los conceptos claves asociados al tema. También se brinda una concepción profunda del estado del arte y de las tendencias actuales en el ámbito de estudio. Además, se presenta a metodología detallada que se utilizara en el desarrollo del software. Así mismo, se discuten los lenguajes, herramientas y tecnologías relevantes proporcionando una visión integral infraestructura los recursos necesarios para el desarrollo exitoso de este estudio.

1.2. Conceptos asociados al tema

Bases de datos: Las bases de datos son aplicaciones informáticas destinadas al almacenamiento y la gestión de grandes volúmenes de información. Es un fondo común de información interrelacionada para ser accedida mediante consultas. También se puede definir como un sistema computarizado para mantener información de un individuo o de una organización y hacer que esté disponible cuando se solicite (Pérez, 2010).

Sistemas Gestores de Bases de Datos (SGBD): Es una colección de programas que permiten a los usuarios crear y mantener una base de datos. Sistema software de propósito general que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones (Pérez, 2010).

Seguridad Informática: Es la que intenta proteger el almacenamiento, procesamiento y transmisión de información digital (Buendía, 2013). Existe dos tipos de seguridad

- **Seguridad pasiva:** Son todos los mecanismos que, cuando sufrimos un ataque, nos permiten recuperarnos razonablemente bien. Por ejemplo, las baterías ante una caída de tensión o la copia de seguridad cuando se ha estropeado la información de un disco.(Buendía 2013)
- **Seguridad activa:** Intenta protegernos de los ataques mediante la adopción de medidas que protejan los activos de la empresa, como vimos en el epígrafe anterior: equipos, aplicaciones, datos y comunicaciones (Buendía, 2013).

Vulnerabilidad: Es un defecto de una aplicación que puede ser aprovechado por un atacante. Si lo descubre, el atacante programará un software (llamado malware) que utiliza esa vulnerabilidad para tomar el control de la máquina (exploit) o realizar cualquier operación no autorizada (Buendía, 2013).

Existen diferentes tipos de vulnerabilidad, según (Guzmán Quesada, 2019), (COLÓN COLÓN, 2021) y (Marcelo and Ángel, 2023) estas son las vulnerabilidades más frecuentes:

1. **Inyecciones SQL:** Implican a un usuario que se aprovecha de vulnerabilidades en aplicaciones web y procedimientos almacenados, para proceder a enviar consultas de bases de datos no autorizadas, a menudo con privilegios elevados. implican a un usuario que se aprovecha de vulnerabilidades en aplicaciones web y procedimientos almacenados, para proceder a enviar consultas de bases de datos no autorizadas, a menudo con privilegios elevados. Atacan servidores de bases de datos relacionales (RDBMS) que utilizan lenguaje SQL.
2. **Los privilegios excesivos:** Cuando a los usuarios o aplicaciones se conceden privilegios de base de datos, que exceden los requerimientos de su función de trabajo, estos privilegios se pueden utilizar para obtener acceso a información confidencial. Por ejemplo, un administrador de una universidad cuyo trabajo requiere acceso de sólo lectura a los archivos del estudiante, sin embargo, puede beneficiarse de los derechos de actualización para cambiar las calificaciones, explica experto de auditoría de base de datos y seguridad de base de datos.
3. **Auditoría de seguridad débil:** Las políticas débiles de auditoría de base de datos representan riesgos en términos de cumplimiento, la disuasión, detección, análisis forense y recuperación. Por desgracia, el sistema de gestión de base de datos nativa (DBMS), audita las capacidades que dan lugar a una degradación del rendimiento inaceptable y son vulnerables a los ataques relacionados con el privilegio, es decir, los desarrolladores o administradores de bases (DBA) pueden desactivar la auditoría de base de datos.
4. **Denegación del servicio:** La denegación de servicio (DoS) puede ser invocada a través de muchas técnicas, las técnicas más comunes de DOS incluyen desbordamientos de búfer, corrupción de datos, la inundación de la red y el consumo de recursos.
5. **Autenticación débil:** Los esquemas de autenticación débiles, permiten a los atacantes asumir la identidad de los usuarios de bases de datos legítimos. Estrategias de ataque específicas incluyen ataques de fuerza bruta, la ingeniería social, y así sucesivamente.
6. **Existencia de servidores de bases de datos ocultos:** El incumplimiento de las políticas de instalación de software en una organización (o la falta de tales políticas) hace que los usuarios

instalen servidores de bases de datos a su discreción para resolver necesidades particulares. El resultado es que aparecen servidores en la red de la organización, algo que los administradores de seguridad desconocen. Estos servidores exponen datos confidenciales a la organización o exponen vulnerabilidades que los atacantes pueden aprovechar.

7. **Desbordamiento de búfer:** Se trata de otro de los medios favoritos utilizados por los piratas y que se dan por el exceso de información que se puede llegar a enviar por medio del ingreso de información mediante el uso de formularios, es decir, se recibe mucha más información de lo que la aplicación espera. Por poner un ejemplo, si se espera la entrada de una cuenta bancaria que puede ocupar unos 25 caracteres y se permite la entrada de muchos más caracteres desde ese campo, se podría dar este problema.

Incidencia: Es cualquier evento que puede afectar la integridad, confidencialidad y disponibilidad de la información. En otras palabras, y atendiendo a la norma ISO 27001:2013, un incidente de seguridad es un evento no deseado o no esperado que puede comprometer significativamente las operaciones del negocio y amenazar la seguridad de la información (Tejada, 2023).

Riesgo: El riesgo es la posibilidad de que una amenaza se produzca, dando lugar a un ataque al equipo. Esto no es otra cosa que la probabilidad de que ocurra el ataque por parte de la amenaza. El riesgo permite tomar decisiones para proteger mejor al sistema (Zambrano and Valencia, 2017). Aquí se trata como la posibilidad de que ocurra una incidencia y la probabilidad de que ocurra el ataque por parte de la vulnerabilidad, y se clasificará en alto, medio, o bajo según el riesgo que corra la misma.

Prioridad: Anterioridad o preferencia de algo respecto de otra cosa precisamente en cuanto es causa suya, aunque existan en un mismo instante de tiempo. Se utiliza para saber a qué base de datos darle solución primero y estará dada del 2 al 8, donde 2 será la prioridad más baja y 8 la más alta (Real Academia, 2023).

La prioridad se calculará de la siguiente manera:

$$\text{Prioridad} = \text{Criticidad} + \text{AVG (Riesgo)}$$

$$\text{AVG(Riesgo)} = (\text{SUM(Riesgo)} / \text{Count(Riesgo)}$$

Dónde:

Criticidad tomará los siguientes valores: Muy Crítico (5), Crítico (4), Moderadamente Crítico (3), Menos Crítico (2) y Poco Crítico (1).

Riesgo tomará los siguientes valores: Alto(3), Medio(2) y Bajo(1).

Criticidad: Remite a leer críticamente, comprender críticamente, adoptar un punto de vista crítico. Es un término muy corriente y habitual en los currículums, aunque tiene un significado impreciso, variado (Cassany, 2005). Se trata según el nivel de importancia que tengan las bases de datos y se clasifica en muy crítico, crítico y moderadamente crítico.

1.3. Estudio del Estado de Arte

1.3.1 Sistemas homólogos internacionales

National Vulnerabiliti Database (NVD): Es un repositorio estandarizado del gobierno de los Estados Unidos en el cual se encuentra almacenada información acerca de la gestión de vulnerabilidades. Estos datos permiten la automatización de la gestión de vulnerabilidades y la toma de medidas de seguridad. NVD incluye bases de datos con listas de control de seguridad, fallos de seguridad relacionados con software, errores de configuración, nombres de productos y métricas de impacto. En la actualidad cada día se adiciona a la NVD un promedio de 12 nuevas vulnerabilidades (Franco, Perea et al. 2013).

Common Weakness Enumeration (CWE): Es una lista desarrollada por la comunidad de tipos de debilidades de software y hardware. Sirve como lenguaje común, vara de medir para las herramientas de seguridad y como base para los esfuerzos de identificación, mitigación y prevención de debilidades. En esencia, la enumeración de debilidades comunes (CWE™) es una lista de tipos de debilidades de software y hardware. La creación de la lista es una iniciativa comunitaria destinada a crear definiciones específicas y concisas para cada tipo de debilidad común (CWE, 2023).

AlienVault USM Anywhere (AV): Es una plataforma unificada de gestión de eventos e información de seguridad (SIEM), diseñada para proporcionar y garantizar una defensa completa contra las amenazas de seguridad más recientes a un precio razonable, lo que la hace ideal para pequeñas y medianas empresas. Ofrece un software de detección de intrusos incorporado como parte de una consola de gestión de seguridad unificada todo en uno. Incluye detección de intrusión en host (HIDS), detección de intrusión en la red (NIDS), así como detección de intrusión en la nube para entornos de nubes

públicas, incluyendo AWS y Microsoft Azure, lo que le permite detectar las amenazas a medida que surgen en su nube crítica y en la infraestructura de las instalaciones (USM Anywhere, 2020).

Oracle Database Vault (ODV): Proporciona controles para evitar que usuarios privilegiados no autorizados accedan a datos confidenciales, evitar cambios no autorizados en la base de datos y ayuda a los clientes a cumplir con los estándares de seguridad corporativos, regulatorios o de la industria. Oracle Database Vault crea un entorno de aplicaciones altamente restringido ("Reino") dentro de Oracle Database que impide el acceso a los datos de las aplicaciones desde cuentas privilegiadas y al mismo tiempo continúa permitiendo actividades administrativas autorizadas en la base de datos (prevent unauthorized database, 2020).

1.3.2 Sistemas homólogos cubanos

AUDAT v2.0: La auditoría es el proceso mediante el cual se contrata a una empresa o a un auditor para recopilar información contable. En el Centro de Tecnologías y Gestión de Datos (DATEC) se desarrolló el Sistema Auditoría de Datos (AUDAT), para facilitar a los especialistas de la Contraloría General de la República de Cuba (CGR) detectar incidencias en la manipulación de bases de datos de sistemas que ellos auditan. Actualmente su módulo de Gráficos contiene gráficos de barras, sectores, líneas y área para comparar información a través de la representación de los cambios que se producen en los datos (Robert & del Carmen, 2015).

Notificación de Vulnerabilidades de Tecnologías y Sistemas Informáticos (NVTSI): Es un sistema que informa a todos los interesados, de todas las Vulnerabilidades que puedan tener las diferentes tecnologías y servicios usados por estos y la forma de eliminarlas, lo cual permitirá sin lugar a dudas aumentar el conocimiento sobre este tipo de peligro y a la par aumentar la cultura general (Carmona García & Thompson Martínez, 2010).

1.3.3 Conclusiones del Estado del Arte

Después de completar el análisis de sistemas homólogos al que se pretende desarrollar, se ha observado que ninguno de los sistemas comparables satisface de manera integral todos los requisitos que se establecen para el sistema propuesto.

Tabla 1. Análisis con elementos homólogos

Sistemas Homólogos	**NVD**	**CWE**	**AV**	**ODV**	**AUDAT**	**NVTS**	**SIVSGBD**
Gestión de Base de Datos	x	x	X	✓	✓	✓	✓
Gestión de Vulnerabilidad	✓	✓	✓	x	x	✓	✓
Gestión de Solución	x	✓	✓	x	x	x	✓
Clasificación de riesgo	✓	✓	✓	x	x	x	✓
Gráfico de prioridad de Base de Datos	x	x	X	x	✓	✓	✓
Cálculo de prioridad	x	x	X	x	x	x	✓
Plataforma	Web	Web	Web	Web	Web	Web	Web
Acceso	Internacional	Internacional	Internacional	Internacional	Nacional	Nacional	Nacional

1.4. Metodología de desarrollo de software

Metodología de desarrollo de software: Es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un

producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado (Maida and Pacienzia, 2015).

Metodología ágil: Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas (Ghosh, 2015).

Metodología tradicional: Las metodologías tradicionales de desarrollo de software son orientadas por planeación. Inician el desarrollo de un proyecto con un riguroso proceso de elicitación de requerimientos, previo a etapas de análisis y diseño. Con esto tratan de asegurar resultados con alta calidad circunscritos a un calendario (Khurana and Sohal, 2011).

1.4.1 Metodología ágil

Ventajas de la metodología ágil

- Flexibles.
- Sus proyectos son subdivididos en proyectos más pequeños.
- Incluyen comunicación constante con el cliente.
- Altamente colaborativos.
- Se adaptan mejor a los cambios.
- Revisión de metodologías ágiles

Tabla 2. Comparación entre metodologías ágiles

EC	Metodología XP	AUP	Scrum
Enfoque principal	Software ágil con énfasis en la calidad del código y la retroalimentación constante	Desarrollo iterativo y basado en casos de uso que se adapta a diferentes tipos de proyectos	Trabajo ágil centrado en entregas incrementales y en la flexibilidad para adaptarse a cambios

Ciclos de desarrollo	Iterativo e incremental	Iterativo y secuencial	Iterativo y basado en sprints
Roles claves	Programador, cliente, coach, tester	Analista, diseñador, programador, tester, gerente de proyecto	Scrum Master, Product Owner, Equipo de Desarrollo
Principales artefactos	Historias de usuario, tarjetas CRC, pruebas unitarias	Especificaciones de casos de uso, modelos de diseño, prototipos	Product Backlog, Sprint Backlog, Incremento de producto
Comunicación	Comunicación cara a cara y constante entre el equipo	Énfasis en la documentación y la comunicación formal	Reuniones diarias de Scrum (Daily Standup) y comunicación constante
Flexibilidad	Alta flexibilidad para cambios en requisitos y prioridades	Menor flexibilidad debido a una mayor formalidad en los procesos	Flexibilidad para realizar ajustes en el producto durante los sprints
Adaptabilidad	Se adapta bien a proyectos en constante evolución y con requisitos cambiantes	Adaptación a diferentes tipos de proyectos, incluidos los de mayor escala	Adecuado para proyectos donde se requiere una entrega rápida y frecuente
Énfasis en documentación	Documentación ligera y enfoque en el código	Documentación detallada para cada fase del desarrollo	Enfoque en la transparencia y documentación relevante durante el desarrollo

Según el estudio realizado y las comparaciones anteriores se decide usar la metodología ágil XP para el desarrollo del sistema por las siguientes razones:

- Porque tiene mayor colaboración y comunicación con el cliente lo que ayuda a comprender mejor los requisitos.
- El cliente está dispuesto a mantener una participación activa en el proceso de desarrollo, lo que permite influir en las funcionalidades y prioridades.

- Se tiene una retroalimentación continua lo que facilita la detección temprana de problemas y su resolución.
- Tiene un desarrollo iterativo con entregas frecuentes, para tener una retroalimentación temprana y realizar ajustes rápidamente.

1.4.2 Programación extreme (XP)

Los valores que persigue XP para el desarrollo de software de calidad según (Wells, 2009) son los siguientes:

Simplicidad: Se refiere a dar pequeños pasos a la vez con el objetivo de alcanzar las metas trazadas para alcanzar el éxito de lo que se desea realizar, así como mitigar los posibles fallos que puedan presentarse de la forma más rápida y oportuna.

Coraje: Siempre se dirá la verdad sobre el avance y el estado del proyecto en el que se esté involucrado. No documentar el fracaso o problemas que se haya presentado en su desarrollo, ya que siempre se pretender llegar al éxito del mismo. Las personas involucradas en los proyectos que aplica XP se adaptaran a los cambios suscitados.

Comunicación: Cada persona es parte del equipo y los mismos se relacionarán todos los mismo, por lo siempre que siempre se deberá estar presto a realizar o recibir alguna observación o critica que permita mejorar el desarrollo del proyecto desde planificación, desarrollo y conclusión del mismo.

Retroalimentación: Cada iteración que se pretenda dar, se lo tomara con el mayor compromiso posible. El producto finalizado en cada iteración, será presentado a tiempo con el objetivo de ser revisado, escuchar con atención la opinión que se tiene sobre este y hacer los cambios necesarios en la siguiente iteración.

Respeto: Mantener siempre el respeto ante todo el mundo lo cual es primordial para valorar el trabajo que cada miembro del equipo hace, lo que se transformara en un producto final con valor agregado.

Variante simplificada de XP. Fases de XP según (Pressman, 2010)

Planeación o Planificación del proyecto: Entre las acciones que se ejecutan en esta fase se encuentra definir las historias de usuario con el cliente y crear el plan de iteraciones donde se indica en qué iteración serán implementadas las historias de usuarios previamente definidas.

Diseño: Sugiere utilizar diseños sencillos y simples pues de esta manera costará menos tiempo y esfuerzo desarrollarlo posteriormente.

Codificación: Se definen los estándares de codificación que se utilizarán para facilitar la comprensión del código.

Pruebas: El funcionamiento del sistema se lleva a cabo a través de las historias de usuario definidas en la primera fase, el cual sugiere que se realicen pruebas funcionales que sirven para evaluar las distintas tareas realizadas.

1.5. Lenguajes, herramientas y tecnologías relevantes para el desarrollo del software

En el siguiente acápite, se describen los lenguajes, herramientas y tecnologías utilizadas para la implementación de la propuesta de solución. Para facilitar la identificación de los mismos, según el componente al que corresponden, se agrupan en frontend y backend.

1.5.1 Lenguajes, herramientas y tecnologías del Frontend

HTML5: Es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad (Grupo de colaboradores de MDN Web Docs, 2017) v5.

CSS: Al igual que HTML CSS no es realmente un lenguaje de programación. Es un lenguaje de hojas de estilo, es decir, te permite aplicar estilos de manera selectiva a elementos en documentos HTML. Es una herramienta que permite personalizar aplicaciones web para lograr un diseño de fácil visualización para los usuarios (Grupo de colaboradores de MDN Web Docs, 2018) v3.

JavaScript: También conocido por su abreviación JS, es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js, Apache CouchDB y Adobe Acrobat. Es un lenguaje script multiparadigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa (Grupo de colaboradores de MDN Web Docs, 2018) ECMAScript 6 2021.

1.5.2 Lenguajes, herramientas y tecnologías del Backend

Node.js: Es un entorno de ejecución multiplataforma para el lenguaje de programación JavaScript. Es de código abierto y su licencia es de tipo MIT Licence lo que significa que cualquier persona puede descargarlo o instalarlo en su computadora sin tener que pagar una licencia (Puciarelli, 2020) v18.15. 0.

Express: Express es una infraestructura web rápida, minimalista y flexible para las aplicaciones Node.js. En general, se prefiere Express a Express.js, aunque está también se acepta. ayuda a no comenzar necesariamente desde cero, como todo framework, sino que provee una infraestructura que permite agilizar el desarrollo de la aplicación web. Express es propio de Node, por lo cual se posiciona como el primero en la lista de frameworks a usar, además provee plugins de alto rendimiento nombrados middleware (Fernández and Claudia 2020) v4.18.2.

MongoDB: Es una base de datos de código abierto y líder en bases de datos NoSQL. Es una base de datos distribuida, basada en documentos en formato BSON, que es una representación binaria de los objetos JSON. Además, MongoDb permite una gran escalabilidad y flexibilidad en la gestión de la base de datos, beneficiando al sistema en propuesto (MongoDB, 2023) v4.0.

La elección de MongoDB para el sistema se justifica por su capacidad para gestionar volúmenes masivos de datos **(Big Data).** Dado que la aplicación está sujeta a posibles cambios y evoluciones en el tiempo, **la flexibilidad en los esquemas de datos** proporcionada por MongoDB se presenta como una solución viable. Además, la capacidad de procesar grandes cantidades de datos en tiempo real **(Análisis en tiempo real)** con el fin de detectar vulnerabilidades de manera más eficiente fue un factor determinante en nuestra decisión. Por último, la implementación de una **carga de trabajo distribuida** permite escalar horizontalmente, lo que implica la posibilidad de agregar más servidores a medida que la demanda aumenta, asegurando así una escalabilidad óptima del sistema. Todas estas características son ventajas de NoSQL.

1.5.3 Herramientas

Visual Studio Code: Es un editor de código redefinido y optimizado para construir y depurar modernas aplicaciones web y de la nube. Visual Studio Code es libre y está disponible en su plataforma favorita: Linux, Mac OSX y Windows (De Oliveira, Polat, y Belkheraz, 2018). Algunas ventajas de Visual Studio

Code son que a diferencia de otros editores de Microsoft ofrece un entorno de desarrollo simplificado que permite editar el código sin requerir dificultad continua y la integración de un terminal direccionado que viene por defecto en la misma carpeta donde se ubica el proyecto; la versión utilizada es v1.84.2.

Permite comenzar a editar con una buena cantidad de herramientas de base que evita configurar nada en el editor para disponer de una buena experiencia de desarrollo. Además, se destaca su consola integrada, comandos, atajos y excelentes extensiones (Henry, 2022).

Visual Paradigm: Presenta un amplio conjunto de herramientas Agile y Scrum para la gestión de proyectos. Un amplio conjunto de herramientas de creación de diagramas está disponible para ayudar a crear diagramas de forma rápida y sin problemas. Se pueden crear diagramas profesionales sin necesidad de aprender. La función Catálogo de recursos permite crear diagramas con la sintaxis correcta de forma más rápida y sencilla. No se necesita memorizar la sintaxis del lenguaje de modelado porque filtra automáticamente lo que puede hacer con el elemento del modelo en el que trabaja en el diagrama (Visual Paradigm, 2022); ; la versión utilizada es v16.2.

Diagrams.net: Es una aplicación web gratuita que te permite hacer muchos tipos de diagramas para temas de ingeniería, negocios, software, y generales de forma sencilla. Algunos ejemplos de los tipos de diagramas que permite hacer para temas de ingeniería y/o software son: diagrama de clases, de flujo, de flujo de funciones cruzadas, de entidad relación, de clases, de secuencia, de arquitecturas en la nube, de bases de datos, eléctricos, mapas de redes, árbol de decisiones, flujos Git, varios tipos de diagramas UML, etc (edutools, 2023).

1.6. Conclusiones parciales

En este capítulo se ha explorado la historia del arte proporcionando una base sólida para comprender el contexto en el que se desarrollará este sistema. Además, se ha tomado la decisión de utilizar la metodología XP para guiar el proceso de desarrollo, lo cual permitirá una mayor flexibilidad y capacidad de respuesta a medida que se vallan descubriendo y abordando los desafíos. En cuanto a tecnologías seleccionadas se ha optado por MongoDB como base de datos NoSQL, Express como framework de aplicaciones web y Node.js como entorno de ejecución del lado del servidor, las cuales son tecnologías conocidas por su capacidad de manejar grandes volúmenes de datos y su rendimiento eficiente.

2. CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN

2.1. Introducción

Este capítulo se centra en el análisis y diseño de la propuesta de solución, siguiendo la metodología de desarrollo de software adoptada. Aquí se presenta la solución al problema de investigación, destacando el proceso de identificación de vulnerabilidades. En la fase de exploración, se detallan los artefactos clave, como las Historias de Usuario (HU) y los requisitos no funcionales de la propuesta. La planificación abarca la estimación del esfuerzo, el plan de iteraciones y la estrategia de entrega. En el diseño del sistema, se abordan los patrones arquitectónicos y de diseño, el modelo vista-controlador, las tarjetas CRC y el modelo de datos. Este enfoque sistemático sienta las bases para el desarrollo efectivo y la implementación de la solución propuesta.

2.2. Propuesta de solución

El sistema propuesto por la presente investigación, le permitirá al usuario llevar a cabo un análisis exhaustivo de una base de datos en busca de posibles vulnerabilidades, que permita la toma de decisiones posteriores. En caso de detectar alguna vulnerabilidad, se procede a registrar las incidencias correspondientes, para poder gestionarlo y solucionarlo posteriormente.

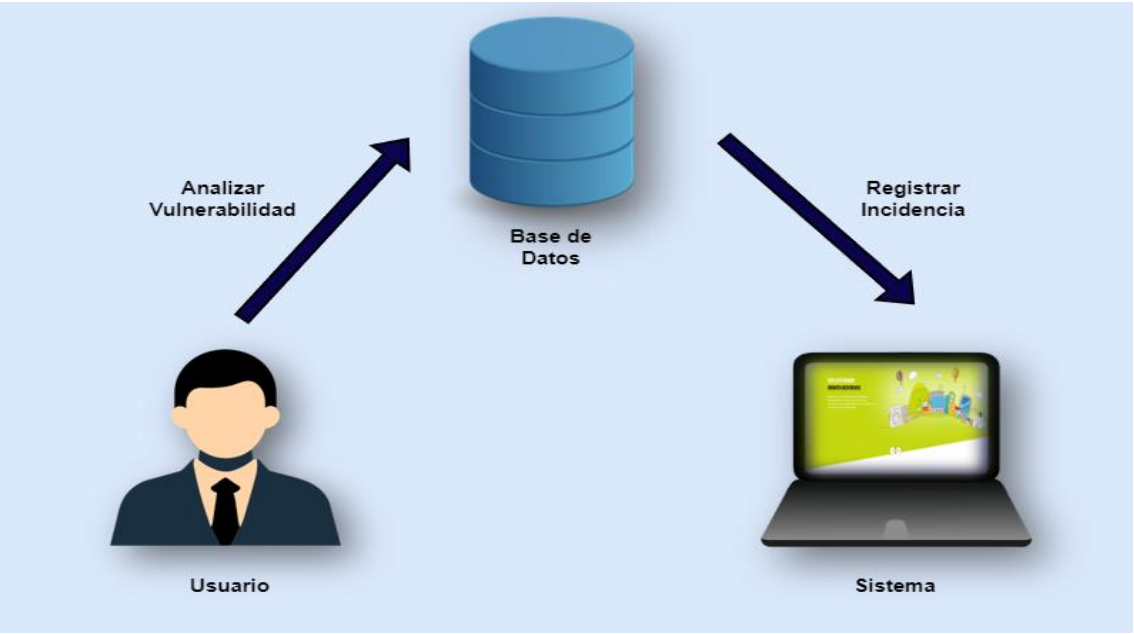


Figura 2. Propuesta de solución

- **Usuario**: Es el especialista o administrador que interactúa con el sistema.
- **Base de datos**: Representa la base de datos, a la cual se le pretende realizar el análisis de vulnerabilidades.
- **Sistema**: Representa el Sistema para la identificación de vulnerabilidades de Seguridad en SGBD.

El proceso se inicia con la identificación de una vulnerabilidad durante la gestión de las bases de datos, momento en el cual el especialista o el administrador, en virtud de su experiencia y conocimientos especializados, toman medidas concretas para gestionar y mitigar el impacto de dichas vulnerabilidades. Una vez que se ha concebido una solución efectiva, esta se integra en la sección de gestión de soluciones, aportando así a una mejora constante y al fortalecimiento continuo de la integridad y seguridad de las bases de datos.

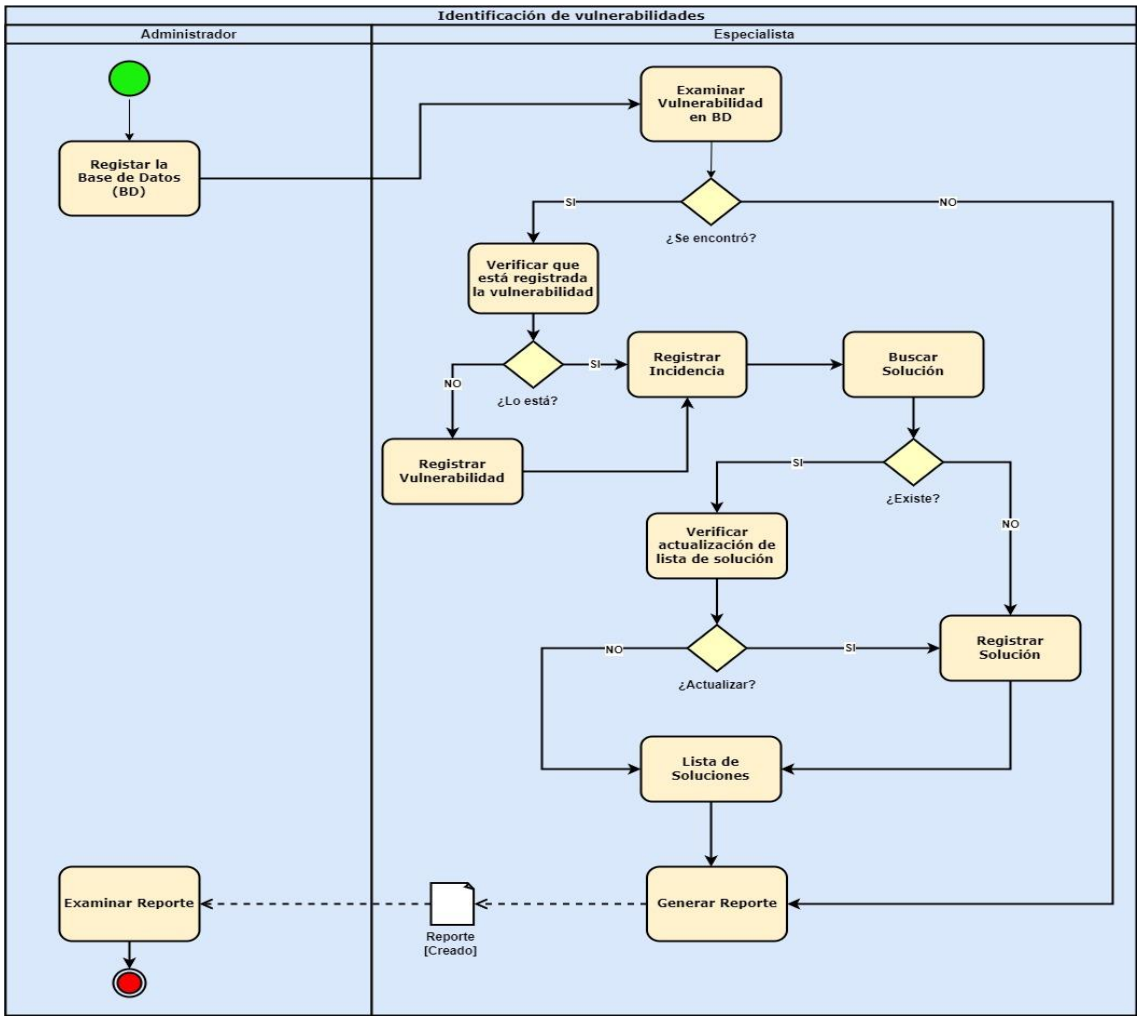


Figura 3. Proceso del negocio

2.2.1 Roles y Funcionalidades

El administrador y el especialista en gestión de bases de datos desempeñan roles clave, colaborando estrechamente para identificar y abordar las vulnerabilidades detectadas en el entorno del sistema.

- **Administrador:** Encargado especialmente para gestionar los usuarios y cambio de contraseña en caso de que el especialista olvide su contraseña, además de también gestionar los demás parámetros como son BD, vulnerabilidad, etc.
- **Especialista:** Encargado de la Gestión de las BD, incidencias y vulnerabilidades, así como su solución

2.3. Planificación

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

El cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la entrega y se determina un cronograma en conjunto con el cliente. Las estimaciones de esfuerzo asociado a la implementación de las HU la establecen los programadores utilizando como medida el punto (Piñeiro Vidal & Roa Parets, 2020).

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

2.3.1 Historia de Usuario

Las **Historias de Usuario (HU)** son una herramienta que agiliza la administración de requisitos, reduciendo la cantidad de documentos formales y tiempo necesarios. Forman parte de la fórmula de captura de funcionalidades (Menzinsky, López et al. 2018).

Con el propósito de determinar cuáles HU son más cruciales para resolver, en función de las demandas del cliente y para llevar a cabo una planificación precisa de la implementación, cada HU es clasificada por el propietario del producto de acuerdo a su relevancia para el negocio, de la siguiente manera:

- **Alta:** HU que son esenciales para el desarrollo del sistema.
- **Media:** HU que representan funcionalidades necesarias, pero no críticas.

- **Baja:** HU que son funcionalidades auxiliares relacionadas con el control de elementos del equipo de desarrollo y la estructura, sin relación directa con el sistema en desarrollo.

Además de esta clasificación, se considera la dificultad y la posible ocurrencia de errores durante la implementación de cada HU. Por lo tanto, el equipo de desarrollo categoriza cada HU en función del riesgo durante su desarrollo en:

- **Alto:** cuando se prevé la posibilidad de errores que podrían inutilizar el código.
- **Medio:** cuando pueden surgir errores que retrasen la entrega de la versión.
- **Bajo:** cuando pueden presentarse errores que se manejarán con relativa facilidad y no afectarán significativamente el progreso del proyecto.

Las HU se representan en tablas divididas en las siguientes secciones:

- **Número:** denota el número, incremental en el tiempo, de la HU que se describe.
- **Nombre de Historia de Usuario:** identifica la HU en la comunicación entre los desarrolladores y el cliente.
- **Usuario:** el rol del usuario que lleva a cabo la funcionalidad.
- **Prioridad de negocio:** se asigna una prioridad (Alta, Media, Baja) a las HU según las necesidades empresariales.
- **Riesgo en desarrollo:** Se asigna una categorización (Alto, Medio, Bajo) a la probabilidad de ocurrencia de errores durante el proceso de desarrollo de la HU.
- **Iteración asignada:** Indica el número de la iteración en la cual se llevará a cabo el desarrollo de la HU.
- **Estimación de puntos:** Representa el tiempo estimado en semanas ideales (equivalentes a 40 horas semanales) necesario para completar el desarrollo de la HU.
- **Descripción:** Brinda una breve descripción de la HU, ofreciendo una visión general de su funcionalidad.
- **Observación:** Proporciona observaciones o advertencias específicas relacionadas con la HU o su implementación.

Gracias al análisis exhaustivo del proceso de gestión e investigación de vulnerabilidades en las bases de datos, se ha logrado una comprensión completa de todos los aspectos relacionados con dicho proceso. Esta claridad ha facilitado la elaboración de todas las historias de usuario en la fase de exploración, sin necesidad de realizar modificaciones o añadir nuevas HU durante las iteraciones posteriores.

A continuación, presentamos algunas de las historias de usuario definidas en colaboración entre el cliente y el equipo de desarrollo. Para observar el resto de las HU, consultar el ANEXO 1. Historias de usuarios.

Tabla 3. Historia de Usuario 1

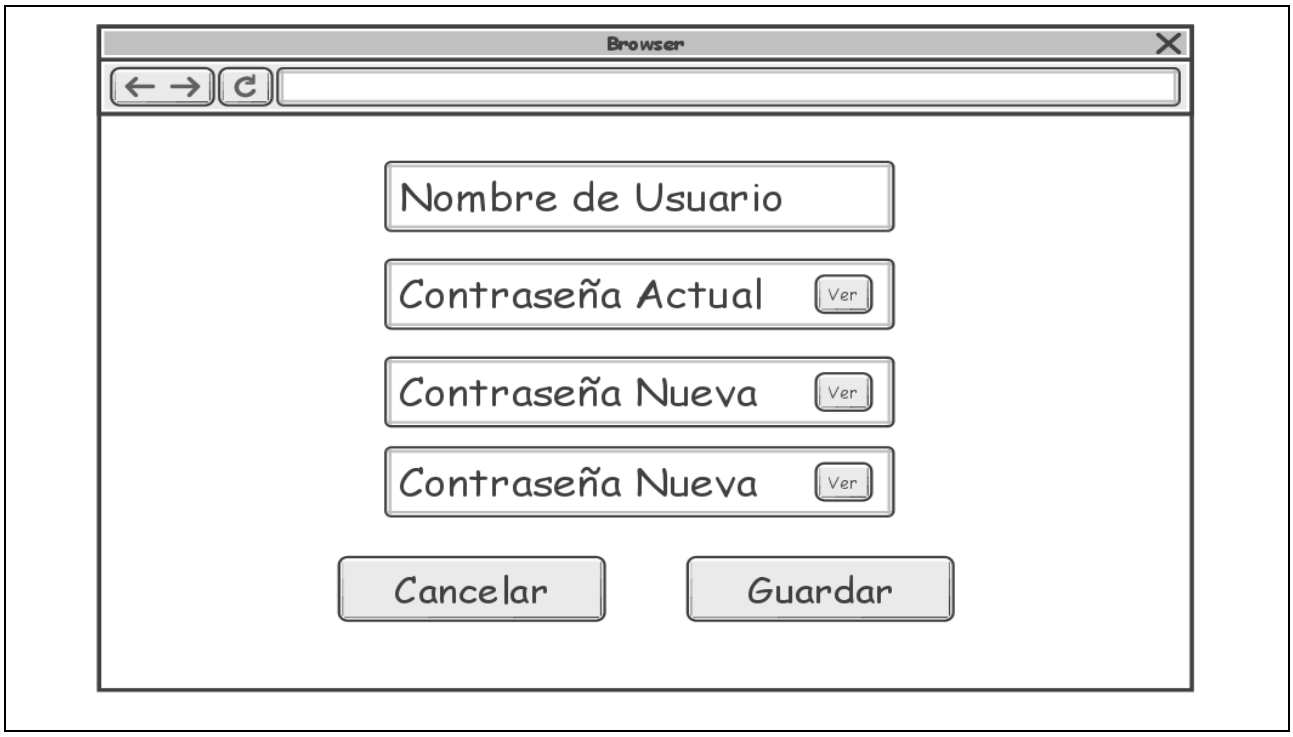
Historia de Usuario	
Número: 1	**Nombre:** HU_Autenticar Usuario
Usuario: Todos los usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 1.0	**Iteración Asignada:** 1
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: Una interfaz de autenticación que requerirá que los usuarios ingresen su nombre de usuario y contraseña, garantizando niveles de seguridad por cada usuario, dando así los permisos de funcionalidad según correspondan.	
Observación: El acceso al sistema requerirá que los usuarios estén debidamente registrados. En el evento de que se introduzcan parámetros de autenticación incorrectos, se generará un mensaje indicando el error correspondiente.	

Tabla 4. Historia de Usuario 2

Historia de Usuario	
Número: 2	**Nombre:** HU_Gestionar Usuario
Usuario: Administrador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 1.6	**Iteración Asignada:** 1
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: El administrador gestionará a los usuarios, que implica crear, modificar, eliminar y mostrar la información de los usuarios, esto es fundamental para mantener la integridad y seguridad del sistema. Y garantiza que el proceso de autenticación del usuario sea efectivo al verificar que la combinación de nombre de usuario y contraseña coincida con la información previamente ingresada y almacenada por el administrador en la tabla correspondiente	
Observación: Solo el administrador tiene acceso a esta página. Datos de usuario: Nombre de usuario, nombre, apellidos y rol.	

Tabla 5. Historia de Usuario 3

Historia de Usuario	
Número: 3	**Nombre:** HU_Cambiar Contraseña
Usuario: Administrador y Usuario	
Prioridad en Negocio: Alto	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 0.6	**Iteración Asignada:** 2
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: El Especialista puede cambiar solo su propia contraseña, mientras que el Administrador puede cambiar tanto su propia contraseña como la de los Especialistas en caso de que estos últimos la pierdan u olviden. Esto proporciona a los usuarios la capacidad de actualizar y restablecer sus contraseñas según sea necesario para garantizar la seguridad y el acceso continuo al sistema.	
Observación: Se solicitará al especialista ingresar su nombre de usuario, contraseña actual y la nueva contraseña, esta última deberá ser ingresada dos veces para su validación. En el caso del administrador, solo se requerirá el nombre de usuario y la nueva contraseña, también ingresada dos veces para confirmarla.	



2.3.2 Propiedades de Software

RNF 1. Requisitos de Usabilidad.

- **RNF 1.1.** El sistema podrá usarse por personas con bajos conocimientos informáticos.
- **RNF 1.2.** Tipo de Software informático: Aplicación WEB.
- **RNF 1.3.** El sistema debe permitir a los usuarios realizar tareas de manera rápida y eficiente.

RNF 2. Requisitos de Eficiencia.

- **RNF 2.1.** La aplicación Web deberá soportar varios usuarios conectados al mismo tiempo.
- **RNF 2.2.** La aplicación web deberá reaccionar lo más rápido posible, a no más de 1 minuto el usuario debe recibir respuesta de su servicio.
- **RN2.3.** La navegación debe ser sin retrasos, la página web no podrá demorarse demasiado en supervisar los servicios pedidos de los usuarios, debe tener la capacidad de responder de forma oportuna a dichos pedidos del usuario sin algún tipo de demora.

RNF 3. Requisitos de Software.

- **RNF 3.1.** Se podrá acceder mediante cualquier dispositivo informático, empleando cualquier un navegador web

- **RNF 3.2.** El Sistema operativo a utilizar es Windows, Linux o MacOS.
- **RNF 3.3.** El software debe ser capaz de manejar grandes cantidades de datos y usuarios.

RNF 4. Requisitos de Hardware.

- **RNF 4.1.** Memoria RAM mínima de 1GB para el lado del servidor.
- **RNF 4.2.** Mínimo 2 GB de almacenamiento interno para el lado del servidor.
- **RNF 4.3.** Resolución de pantalla compatible con cualquier dispositivo.
- **RNF 4.4.** Conexión a Internet o Intranet.

RNF 5. Requisitos de Seguridad.

- **RNF 5.1.** El acceso al sistema debe ser mediante mecanismos de autenticación y basado en permisos de acuerdo al rol de los usuarios.

RNF 6. Requisitos de Soporte.

- **RNF 6.1.** El sistema será actualizado sistemáticamente para su mejor funcionamiento.
- **RNF 6.2.** Así mismo se podrán incorporar a lo largo del tiempo nuevas funcionalidades según las necesidades del usuario.

RNF 7. Requisitos de Diseño y la Implementación.

- **RNF 7.1.** El software está implementado usando el lenguaje de programación HTML, CCS, JavaScript.
- **RNF 7.2.** Se implementará en la herramienta Visual Studio Code 1.84.2, lanzada el 09/11/23.

RNF 8. Requisitos de Apariencia o Interfaz Externa.

- **RNF 8.1.** El software debe ser rápido y eficiente en su uso de recursos.
- **RNF 8.2.** El software debe ser seguro y proteger los datos y la privacidad de los usuarios.
- **RNF 8.3.** El software debe ser fácil de usar y comprender para los usuarios.
- **RNF 8.4.** Compatibilidad: El software debe ser compatible con diferentes sistemas operativos y dispositivos.

2.3.3 Estimación del esfuerzo

Para proyectar el tiempo necesario para completar todas las historias de usuario y la carga de trabajo correspondiente, se realiza una estimación del esfuerzo para cada HU. Dado que la velocidad del proyecto no se establece definitivamente hasta después de múltiples iteraciones, se derivó un cálculo basado en datos previos: la primera iteración implicó 2.6 puntos, la segunda 2 puntos, la tercera 3.9 puntos y la cuarta 2.1 puntos, agrupando un total de 9 HU en cuatro iteraciones. La determinación del número de iteraciones se fundamentó en dos criterios clave: la experiencia previa del equipo de desarrollo (método analógico) y los ajustes permitidos mediante la planificación orientada a los objetivos del sistema (suma total de puntos / velocidad del proyecto).

Tabla 6. Estimación de esfuerzo por historia de usuario

Iteración	**Historias de Usuarios**		**Puntos estimados (Semanas)**
1	1	Autenticar Usuario	1.0
	2	Gestionar Usuario	1.6
2	3	Cambiar Contraseña	0.6
	4	Gestionar Base de Datos	1.4
3	5	Gestionar Incidencia	1.5
	6	Gestionar Vulnerabilidad	1.5
	7	Gestionar Solución	0.9
4	8	Mostrar Cantidad de Incidencias	1.4
	9	Generar Reporte	0.7
Total			10.6

2.3.4 Plan de Iteraciones

Una vez identificados todas las HU se procede a analizar y general el artefacto de iteraciones. El objetivo de cada iteración es poner en producción algunas historias nuevas que estén probadas y listas para funcionar. El proceso comienza con un plan que se establece las historias que hay que poner en prácticas y se explica cómo lo hará el equipo (Letelier 2006).

Luego se procede con la planificación de la fase de implementación, donde las HU serán desarrolladas y sometidas a pruebas en un ciclo iterativo, traduciéndose en Tecnologías de la Información (TI). Con el fin de mantener una operación organizada, el equipo acuerda llevar a cabo un total de cuatro iteraciones.

Iteración 1:

Durante esta iteración, se enfocará en la implementación de HU de alta prioridad relacionadas con la etapa inicial del proceso de reproducción del sistema. Esto establecerá la base fundamental de la arquitectura. Además, se entregará una primera versión del producto, lo que permitirá al cliente probar algunas de las funcionalidades clave, como la autenticación de usuario y la gestión de usuario.

Iteración 2:

Durante esta iteración, se abordarán las demás HU de alta prioridad para el negocio, y se solucionarán cualquier error o inconformidad reportada por los usuarios en relación a los componentes implementados en la iteración previa. Como resultado, se obtendrá una versión más completa que abarcará diversas funcionalidades del sistema. Al finalizar esta iteración, los usuarios tendrán acceso a características que incluyen la gestión de nomencladores y la gestión de bases de datos.

Iteración 3

Durante esta iteración, se continuará con la implementación de HU de alta prioridad. Se enfocarán específicamente en las HU de gestionar incidencias, gestionar vulnerabilidades y gestionar solución. Además, se abordarán y solucionarán los errores o discrepancias señalados por los usuarios en relación a las funcionalidades desarrolladas en la iteración previa. Como resultado de estos esfuerzos, se logrará una versión del sistema que abarcará hasta la etapa de identificación de vulnerabilidades y sus soluciones.

Iteración 4

Durante esta última iteración, se enfocarán en la implementación de HU de prioridad media, lo que permitirá completar el desarrollo de la aplicación. Se logrará así obtener la versión 1.0 del Sistema para la identificación de vulnerabilidades de Seguridad en Sistemas Gestores de Base de Datos. En esta versión, los usuarios tendrán la capacidad de utilizar todas las funcionalidades y aprovechar al máximo el sistema. Se finalizará con la implementación de las funcionalidades de mostrar criticidad y generar reportes.

Tabla 7. Plan de duración de las iteraciones

Iteración	Historias de Usuarios		Duración (semanas)
1	1	Autenticar Usuario	2.6
	2	Gestionar Usuario	
2	3	Cambiar Contraseña	2.0
	4	Gestionar Base de Datos	
3	5	Gestionar Incidencia	3.9
	6	Gestionar Vulnerabilidad	
	7	Gestionar Solución	
4	8	Mostrar Cantidad de Incidencias	2.1
	9	Generar Reporte	
Total			10.6

2.3.5 Planificación de la Entrega

En esta fase el cliente establece la prioridad de cada HU, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días (Letelier, 2006).

Tras un análisis se procede a ordenar y clasificar las HU, con el objeto de definir prioridades y tiempos de entrega, estableciendo iteraciones iniciales y posteriores incrementos conforme los principios de la metodología XP.

Tabla 8. Plan de Entregas

Iteración	Fecha de inicio	Fecha de fin
1	4 de septiembre del 2023	25 de septiembre del 2023
2	26 de septiembre del 2023	12 de octubre del 2023

3	13 de octubre del 2023	15 de noviembre del 2023
4	16 de noviembre del 2023	2 de diciembre del 2023

2.4. Diseño del Sistema

Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente (PRESSMAN, 2010). La implementación del diseño del sistema Extreme Programming (XP) nos permite adaptarnos rápidamente a los cambios de requisitos, asegurando una respuesta ágil a las demandas en evolución de la gestión de vulnerabilidades en múltiples bases de datos. Además, fomenta una comunicación efectiva entre el equipo, lo que mejora la comprensión de los requisitos y garantiza entregas frecuentes de soluciones eficaces para mantener la seguridad de la información."

La Metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes (PRESSMAN, 2010):

Simplicidad: Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione.

Soluciones "Spike": Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "Spike"), para explorar diferentes soluciones.

Recodificación ("Refactoring"): Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de crearlo más simple, conciso y entendible. Las metodologías de XP sugieren re codificar cada vez que sea necesario.

2.4.1 Patrón arquitectónico

Integra una arquitectura que sea coherente con los valores y principios de XP, como la simplicidad, la adaptabilidad y la entrega continua. Los patrones arquitectónicos para el software definen un enfoque específico para el manejo de algunas características del sistema (PRESSMAN, 2010). La flexibilidad y la entrega continua se alinea con las necesidades de un sistema de gestión de vulnerabilidades en bases de datos en constante evolución. Además, la comunicación efectiva y el énfasis en la calidad del código podrían mejorar la comprensión de los requisitos cambiantes y garantizar la seguridad de la información en tu sistema.

2.4.2 Modelo Vista Controlador

El modelo MVC (Modelo, Vista, Controlador) es un esquema de arquitectura por capas muy utilizado en el desarrollo de software basado en aplicaciones web. El modelo (M) controla todo lo relacionado con los datos, la vista (V) lo relacionado con las interfaces de usuario y el controlador (C) se encarga de la manipulación del M para mostrar información en la vista (Ávila Garzón, 2019).

Modelo (Model): El modelo representa la lógica y los datos del sistema. Aquí se maneja la lógica empresarial y la interacción con la base de datos. El modelo consiste en clases de JavaScript que manejen la lógica principal de la gestión de vulnerabilidades y la interacción con la base de datos.

Vista (View): La vista es la capa de presentación. Se utiliza HTML y CSS para crear la interfaz de usuario y mostrar la información de manera adecuada. Las vistas incluyen plantillas HTML dinámicas que se llenan con datos del controlador.

Controlador (Controller): El controlador actúa como un intermediario entre el modelo y la vista. el controlador en JavaScript maneja la lógica de enrutamiento y la interacción entre la interfaz de usuario y la lógica de la aplicación. Se encarga de gestionar las solicitudes del usuario y de actualizar los datos en el modelo, así como de actualizar la vista en consecuencia.

2.4.3 Patrones de diseño

Los patrones de diseño son estructuras bien definidas que permiten mantener una lógica de organización en el código de un sistema, gracias a esto se puede crear software de calidad, con más facilidad de mantenimiento y con una mejor comprensión del código al buscar modularidad en el sistema (Alvarez et al., 2022).

Patrones GRASP

Los Patrones GRASP (General ResponsibilityAssignment Software Patterns, Patrones Generales de software para asignación de responsabilidades) como su nombre lo indica son los que asignan responsabilidades a objetos software.

A continuación, se reflejan los patrones GRASP empleados en el diseño del sistema:
Bajo acoplamiento: Impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que nos lleve a los resultados negativos que puede producir un acoplamiento alto. Es el grado de interdependencia entre los módulos (Giraldo et al., 2011).

```css
.custom_nav-container.navbar-expand-lg .navbar-nav .nav-link {
  padding: 0px 25px;
  color: #fefeff;
  text-align: center;
  font-family: "Roboto", sans-serif;
}

.custom_menu-btn {
  z-index: 9;
  position: absolute;
  right: 15px;
  top: 14px;
}
```

Figura 4. Evidencia el patrón Bajo Acoplamiento

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación (Giraldo et al., 2011). Se ve en evidencia cuando el administrador puede añadir un usuario (Especialista o Administrador)

```javascript
app.post("/crearperfil", async (req, res) => {
  const newEspecialista= new Especialista({
    nombre:req.body.nombre,
    apellido:req.body.apellido,
    nombreusuario:req.body.nombreusuario,
    contrasena:req.body.contrasena,
    admin:req.body.admin
  });
  if(newEspecialista.admin=="1"){
    newEspecialista.admin=true;
  }else{
    newEspecialista.admin=false;
  }
```

Figura 5. Evidencia el patrón Creador

Experto: Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Expresa la intuición común de que los objetos hacen las cosas relacionadas con la información que tienen (Giraldo et al., 2011). Esto se ve de manifiesto cuando el administrador puede editar los usuarios.

```
app.post("/editarespecialista", async (req, res) => {
  const newEspecialista= new Especialista({
    nombre:req.body.nombre,
    apellido:req.body.apellido,
    nombreusuario:req.body.nombreusuario,
    contrasena:req.body.contrasena,
    admin:req.body.admin
  });
  if(newEspecialista.admin=="1"){
    newEspecialista.admin=true;
  }else{
    newEspecialista.admin=false;
  }
  try{
    if(newEspecialista.nombreusuario==="admin"){
        notifier.notify({
          'title': `Error`,
          'message': 'No se pueden usar estas credenciales',
        });
      res.redirect("/editarespecialista");
```

Figura 6. Evidencia el patrón experto

Patrones GoF

Los patrones GoF describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos y se agrupan en tres grandes categorías: creacionales, estructurales y los de comportamiento (Piñeiro Vidal & Roa Parets, 2020).

A continuación, se reflejan los patrones GoF empleados en el diseño del sistema:

Adaptador (Adapter): Es un patrón de diseño estructural que nos permite reutilizar una o varias clases, integrándolas o embebiéndolas dentro de otra clase que actúa como contenedor (Pérez Delgado, 2020). En nuestro entorno, un especialista puede convertirse en un administrador, pero solo el administrador puede modificar su rol según sea necesario.

```
if(newEspecialista.admin=="1"){
  newEspecialista.admin=true;
}else{
  newEspecialista.admin=false;
}
```

Figura 7. Evidencia el patrón Adaptador

Observador (Observer): es un patrón de comportamiento que permite que uno o varios objetos observadores, sean notificados cuando el objeto que están observando cambie de estado (Pérez Delgado, 2020). Un ejemplo donde este patrón se utiliza es cuando se añade un especialista o administrador y al finalizar, el sistema notifica que se añadió correctamente.

```
await newEspecialista.save();
notifier.notify({
    'title': `Se añadió a ${newEspecialista.nombreusuario}`,
    'message': 'Se ha insertado un especialista nuevo',
});
res.redirect("/admin");
}
```

Figura 8. Evidencia el patrón Observador

Decorador (Decorator): Permite añadir responsabilidades adicionales a un objeto dinámicamente para proporcionar una alternativa flexible a la especialización mediante herencia si se añaden nuevas funcionalidades (Catallops & José, 2019). Se ve ejemplificado cuando la contraseña puede cambiar al dar clic en el icono a su derecha y pasa de oculto (password) a observable (text).

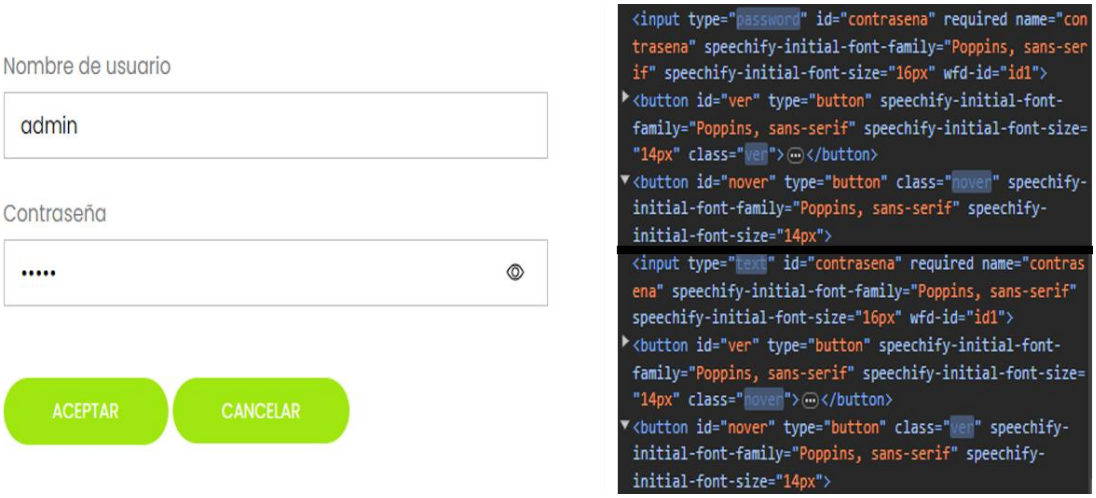


Figura 9. Evidencia el patrón Decorador

2.4.4 Tarjetas CRC

El modelado CRC es una técnica muy efectiva para identificar y validar los requerimientos del usuario, ya que se trabaja con ellos y no en contra. Las tarjetas son la base para un ambiente que ayude al entendimiento del flujo de información en el desarrollo del sistema. Consiste en un conjunto de sesiones cuyo énfasis es moverse gradualmente desde el análisis al diseño (Herrera et al., 2021).

Para el diseño de las aplicaciones utilizando la metodología Extreme Programming (XP), no es necesario recurrir a la representación del sistema mediante diagramas de clases basados en la notación del Lenguaje Unificado de Modelado (UML). En su lugar, se emplean técnicas alternativas como las tarjetas CRC, que son una extensión más informal de UML. Estas tarjetas facilitan la descripción de las clases y responsabilidades, permitiendo un enfoque más práctico y orientado a la interacción en lugar de a la documentación detallada. Con las tarjetas CRC, el énfasis se coloca en la comunicación efectiva entre los miembros del equipo y en la comprensión compartida de la

estructura y las interacciones del sistema. Esta metodología fomenta la colaboración y la flexibilidad, permitiendo una representación más ágil y adaptable del diseño de la aplicación.

Tabla 9. Tarjeta CRC 1

Tarjeta CRC	
Clase: Base_Datos	
Responsabilidad	**Colaboración**
Almacena todos los detalles y características relacionados con las BD	Mongoose Incidencia

Tabla 10. Tarjeta CRC 2

Tarjeta CRC	
Clase: Vulnerabilidades	
Responsabilidad	**Colaboración**
Almacena todos los detalles y características relacionados con las vulnerabilidades	Mongoose Solución

Tabla 11. Tarjeta CRC 3

Tarjeta CRC	
Clase: Incidencia	
Responsabilidad	**Colaboración**
Almacena todos los detalles y características relacionados con las incidencias	Mongoose especialista vulnerabilidad base_de_datos

Tabla 12. Tarjeta CRC 4

Tarjeta CRC

Clase: Soluciones	
Responsabilidad	**Colaboración**
Almacena todos los detalles y características relacionados con las soluciones	Mongoose vulnerabilidad

Tabla 13. Tarjeta CRC 5

Tarjeta CRC	
Clase: Especialistas	
Responsabilidad	**Colaboración**
Almacena todos los detalles y características relacionados con los usuarios	Mongoose

2.4.5 Modelo de Datos

El diagrama entidad-relación (ER) es una representación gráfica de las entidades y sus relaciones en un modelo de datos. El propósito principal de este diagrama es mostrar la estructura lógica de una base de datos de manera visual, con entidades representadas por rectángulos y relaciones entre ellas indicadas por líneas.

El diagrama entidad-relación describe las entidades (tablas), los atributos de estas entidades (columnas) y las relaciones entre las entidades. También puede representar restricciones de integridad, como claves primarias y foráneas. El modelo entidad-relación es una herramienta importante en el diseño de bases de datos, ya que proporciona una comprensión clara de la estructura de la base de datos y sirve como punto de partida para la implementación del sistema.

Los modelos de datos constituyen una representación estructurada y conceptual de la información que se va a almacenar, procesar y manipular en una base de datos. Como el sistema de bases de datos utilizados es MongoDB, el cual es no relacional, el modelo de datos para describir como se estructura la base de datos, es un modelo de datos de documentos. Cada documento representa una entidad independiente y contiene pares de clave-valor, almacenándose en formatos como JSON (JavaScript Object Notation) o BSON (Binary JSON). Los documentos no tienen un esquema fijo, lo que permite flexibilidad en la representación de los datos.

2.5. Conclusiones parciales

Este capítulo revela varias conclusiones importantes sobre el desarrollo del proyecto en cuestión. En primer lugar, la identificación de 9 HU durante la fase de planificación ha sido fundamental para determinar las partes del software que deben implementarse con prioridad. Además, la estimación del tiempo realizada ha confirmado que el proyecto puede ser entregado en el plazo adecuado según las prácticas de Extreme Programming (XP) y el tiempo asignado para este tipo de trabajo. Por último, la elaboración de 5 tarjetas CRC ha demostrado ser crucial para detectar errores tempranos en el diseño del sistema, lo que ha facilitado la corrección oportuna de posibles problemas. El uso de los patrones MVC, GRASP y GOF proporcionó un criterio objetivo para asignar responsabilidades a las clases y definir las relaciones que se establecen entre ellas, permitiendo un diseño flexible y fácil de mantener.

3. CAPÍTULO III: IMPLIMENTACIÓN Y PRUEBAS

3.1. Introducción

Después de finalizar la fase de diseño, que establece la base estructural del sistema, se da inicio a la fase de ejecución y pruebas. En esta etapa se tiene en cuenta la implementación en la cual se representa el diagrama de despliegue y se observan los estándares de codificación utilizados en la implementación del sistema. Aquí las Historias de Usuario (HU) se desglosan en Tareas de Programación (TP) para simplificar su entendimiento y desarrollo. Se lleva a cabo la programación de pruebas unitarias y se realizan pruebas de aceptación. Para ellos se tiene en cuenta la estrategia de pruebas, niveles de pruebas y los métodos de caja blanca y caja negra. Este período destaca por presentar los avances más concretos y significativos, ya que culmina con un sistema completo validado por el cliente. En este capítulo se abordan temas como los lineamientos de codificación aplicados al código desarrollado, las iteraciones llevadas a cabo para la implementación de las HU, así como las pruebas realizadas y sus respectivos resultados.

3.2. Codificación

Durante esta etapa, se lleva a cabo el desarrollo de las funcionalidades, lo que resulta en entregables funcionales al final de cada iteración que implementan las Historias de Usuario asignadas. Dado que las Historias de Usuario pueden carecer de detalles suficientes para el análisis y desarrollo, al inicio de cada iteración se realizan TP con el cliente para recopilar todos los datos necesarios.

Es fundamental que el cliente participe de manera activa en esta fase del ciclo. Además, las iteraciones se utilizan para medir el progreso del proyecto. El logro de una iteración sin errores representa un indicador claro de avance en el desarrollo del proyecto.

3.2.1 Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado (UML) que se utiliza para modelar la disposición física de los artefactos software en nodos. Muestra la arquitectura del sistema como el despliegue de los artefactos de software a los objetivos de despliegue (Carrillo Peña, 2020).

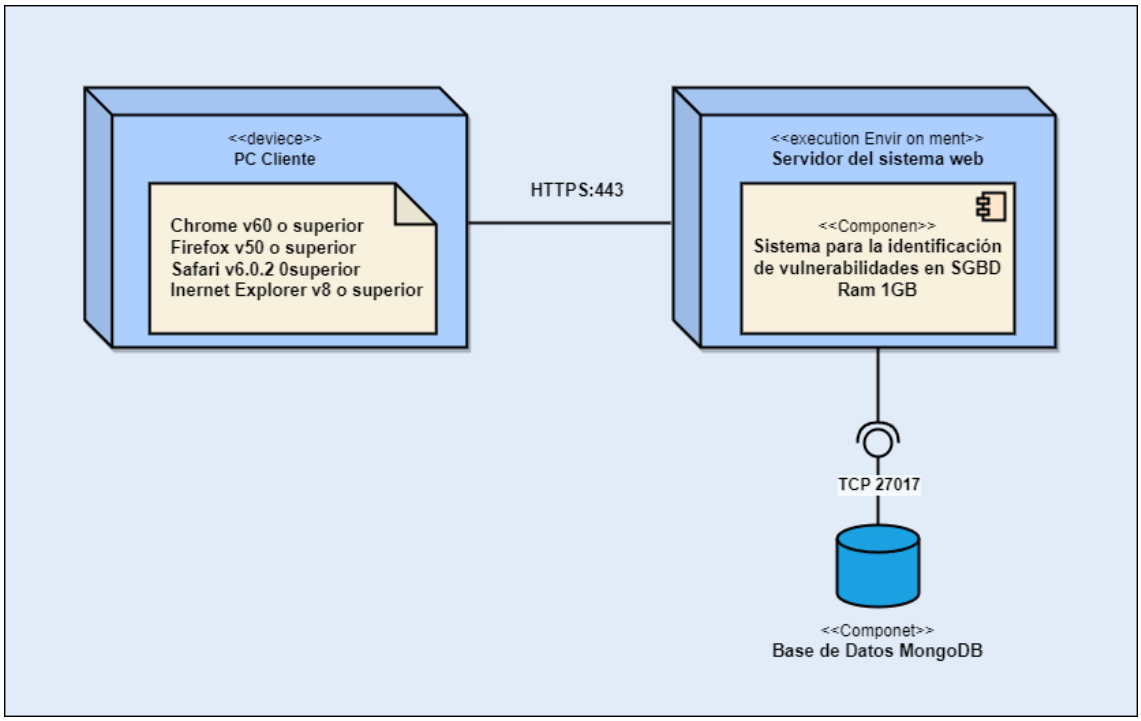


Figura 10. Diagrama de despliegue

PC cliente: Representa el hardware y el software necesario donde se desplegará la aplicación Web. Este dispositivo requiere un navegador web ya sea Chrome, Firefox, Safari, etc.

Servidor: Sistema un sistema de alta capacidad y rendimiento que provee los datos mediante el servicio web. Requiere una capacidad de 1 gigabyte (GB) de RAM

Base de datos MongoDB: Es la base de datos que utiliza el Sistema para la identificación de vulnerabilidades en SGBD para guardar información referente a los usuarios, base de datos, vulnerabilidades, incidencias y soluciones.

HTTPS: Protocolo de transferencia de hipertexto seguro (Hypertext Transfer Protocol Secure) es un protocolo de comunicación seguro que se utiliza para enviar datos de manera segura entre un servidor web y un navegador web en este caso por el puerto 443.

TCP: Protocolo de la capa de transporte (Transmission Control Protocol) es un protocolo de comunicación utilizado en redes de computadoras para garantizar la transmisión confiable de paquetes de datos en este caso usando TCP 27017.

3.2.2 Estándares de codificación

Los estándares de codificación facilitan la lectura y entendimiento del código y también garantiza que las revisiones al código, el mantenimiento, el reúso y el proceso de depuración sea más fácil (Ampuero & Trujillo, 2016). Alguno de los estándares utilizados es:

- **Snake_case** es un estilo de escritura en el que las palabras se escriben en minúsculas y se separan con guiones bajos.

```
console.log(bases_de_datos);
try {
  const existingDocument = await Bases_de_datos.findOne({ nombre: bases_de_datos.nombre }).exec();
  if(existingDocument){
    notifier.notify({
      'title': `Error`,
      'message': `Ya la base de datos ${bases_de_datos.nombre} está en nuestro sitio web ya`,
```

Figura 11. Uso de snake_case. Fuente: (Elaboración propia)

- **UpperCamelCase** es un estilo de escritura donde la primera letra de cada palabra se escribe en mayúscula, sin espacios ni caracteres especiales entre las palabras.

```
const Especialista = require('./models/especialista');
const Nomenclador =require('./models/nomenclador');
const Bases_de_datos=require('./models/bases_de_datos');
const Vulnerabilidad=require('./models/vulnerabilidad');
const Incidencia=require('./models/incidencia');
const Solucion=require('./models/solucion');
```

Figura 12. Uso de UpperCamelCase. Fuente: (Elaboración propia)

- **UTF-8 (Unicode Transformation Format-8)** es un esquema de codificación de caracteres que representa cada símbolo en la norma Unicode.

```
<!-- Basic -->
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Figura 13. Uso de UTF-8. Fuente: (Elaboración propia)

- **Indentación y formato** se utiliza sangrías consistentes (normalmente de 2 o 4 espacios).

```
try{
  if(newEspecialista.nombreusuario==="admin"){
      notifier.notify({
        'title': `Error`,
        'message': 'No se pueden usar estas credenciales',
      });
    res.redirect("/editarespecialista");
```

Figura 14. Uso de Cuatro espacios por indentación: (Elaboración propia)

- **Comentarios en códigos complejo** proporciona explicaciones detalladas y claras sobre la lógica, el propósito y el funcionamiento de secciones complejas del código.

```
        suma+=riesgo[i]; //Aqui se suma los riesgo por BD según si es alto=3, medio=2, bajo=1
    }
    let prom=suma/riesgo.length; //Se calcula promedio de riesgo (suma de riesgo entre la cantidad)
    prom = prom|0; //Se obtiene solo la parte entera del promedio de riesgo
  return criticidad+prom; //Se obtiene la prioridad po cda BD con la suma entre criticidad y el promedio de riesgo
```

Figura 15. Uso de Comentarios en códigos complejos: (Elaboración propia)

- **Comillas consistentes** mantiene un estilo coherente al usar comillas simples o dobles para cadenas de texto.

```
if (err) {
  notifier.notify({
    'title': 'Error',
    'subtitle': 'Mantenimiento diario',
    'message': 'Hubo algún error',
    'icon': './public/images/icon.png',
    'contentImage': './public/images/icon.png',
    'sound': 'ding.mp3',
    'wait': true
  });
```

Figura 16. Uso de Comillas consistentes: (Elaboración propia)

- **Uso de === en lugar de ==** al comparar valores, se recomienda utilizar el operador estricto de igualdad (===) en lugar del operador de igualdad (==) para evitar errores inesperados relacionados con la conversión de tipos de datos.

```
if(newEspecialista.nombreusuario==="admin"){
    notifier.notify({
      'title': `Error`,
      'message': 'No se pueden usar estas credenciales',
    });
  res.redirect("/editarespecialista");
```

Figura 17. Uso de === en lugar de ==: (Elaboración propia)

3.2.3 Tareas de programación

En XP cada historia de usuario se divide en tareas de programación. Estas se crean para obtener una mejor planificación de la historia; estas pretenden cumplir con las funcionalidades básicas que luego conformaran las funcionalidades generales de cada historia (Cortina Perdomo, 2018).

Iteración 1

En esta etapa, se enfocará en las HU de alta prioridad, estableciendo así los cimientos arquitectónicos esenciales del sistema. El objetivo era construir una versión inicial del producto que incluyera funcionalidades críticas para presentar al cliente y obtener retroalimentación rápidamente. Las HU abordadas fueron: Autenticar usuario y Gestionar usuario, y para su desarrollo, las dividimos

en cinco TP. Este enfoque modular permitió una implementación eficiente y una evaluación temprana por parte del cliente, véase en el ANEXO 2. Iteración 1. Tareas de programación.

HU1: Autenticar Usuario

Tabla 14. Tarea de programación 1 Autenticar usuario

Tarea de programación	
Número de tarea: 1	**Historia de Usuario: 1.** HU_Autenticar Usuario
Nombre de la tarea: Autenticar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 4 de septiembre de 2023	**Fecha de fin:** 11 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para autenticar usuarios, empleando la función hash md5 para cifrar las contraseñas. Gestionar mensajes que informen sobre el éxito o el fallo de la autenticación.	

HU2: Gestionar Usuario

Tabla 15. Tarea de programación 2 Mostrar Usuario

Tarea de programación	
Número de tarea: 2	**Historia de Usuario:** 2. HU_Gestionar Usuario
Nombre de la tarea: Mostrar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 12 de septiembre de 2023	**Fecha de fin:** 14 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para mostrar usuario datatable que permita mostrar los campos de todos los usuarios listadas, manejar mensaje en caso de que no exista ningún usuario registrado.	

Iteración 2

Se cumplió con la modificación solicitada por el cliente durante esta iteración, con base en la entrega previa, se implementaron las siguientes HU: . Cambiar Contraseña y Gestionar Base de Datos. Con ese propósito, se establecieron cinco TP véase en el ANEXO 3. Iteración 2. Tareas de programación.

HU3: Cambiar Contraseña

Tabla 16. Tarea de programación Cambiar contraseña

Tarea de programación	
Número de tarea: 6	**Historia de Usuario:** 3. HU_Cambiar Contraseña
Nombre de la tarea: Modificar Contraseña	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha de inicio: 26 de septiembre de 2023	**Fecha de fin:** 30 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para cambiar contraseña de un usuario ya registrado, en caso de manejarlo el administrador, mostrar los campos de nombre de usuario y pedir contraseña nueva de usuario a actualizar, mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de usuario. En caso de especialista manejar, mostrar los campos de nombre de usuario, pedir contraseña actual y pedir contraseña nueva de usuario a actualizar, mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de BD.	

HU4: Gestionar Base de Datos

Tabla 17. Tarea de programación Mostrar BD

Tarea de programación	
Número de tarea: 7	**Historia de Usuario:** 4. HU_Gestionar Base de Datos
Nombre de la tarea: Mostrar Base de Datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 2 de octubre de 2023	**Fecha de fin:** 3 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	

Descripción: Crear plantilla y vista para mostrar BD datatable que permita mostrar los campos de todas las bases de datos listadas, manejar mensaje en caso de que no exista ninguna BD registrada.

Iteración 3

En esta iteración, se abordó la no conformidad identificada en la entrega previa y se llevaron a cabo las modificaciones solicitadas por el cliente en esa misma entrega. Posteriormente, se procedió a implementar las HU: Gestionar Incidencia, Gestionar Vulnerabilidad y Gestionar Solución. Para ello se establecieron doce TP, véase en el ANEXO 4. Iteración 3. Tareas de programación.

HU5: Gestionar Incidencia

Tabla 18. Tarea de programación Mostrar incidencia

Tarea de programación	
Número de tarea: 11	**Historia de Usuario:** 5. HU_Gestionar Incidencia
Nombre de la tarea: Mostrar Incidencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 13 de octubre de 2023	**Fecha de fin:** 14 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para mostrar incidencia datatable que permita mostrar los campos de todas las incidencias, manejar mensaje en caso de que no exista ninguna incidencia registrada.	

HU6: Gestionar Vulnerabilidad

Tabla 19. Tarea de programación Mostrar vulnerabilidad

Tarea de programación	
Número de tarea: 15	**Historia de Usuario:** 6. HU_Gestionar Vulnerabilidad
Nombre de la tarea: Mostrar Vulnerabilidad	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 26 de octubre de 2023	**Fecha de fin:** 27 de octubre de 2023

Programador Responsable: Davier J. Carballo, Luhana Castillo.
Descripción: Crear plantilla y vista para mostrar vulnerabilidad datatable que permita mostrar los campos de todas las vulnerabilidades listadas, manejar mensaje en caso de que no exista ninguna vulnerabilidad registrada.

HU7: Gestionar Solución

Tabla 20. Tarea de programación Mostrar solución

Tarea de programación	
Número de tarea: 19	**Historia de Usuario:** 7. HU_Gestionar Solución
Nombre de la tarea: Mostrar Solución	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 8 de noviembre de 2023	**Fecha de fin:** 8 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para mostrar solución datatable que permita mostrar los campos de todas las soluciones por cada vulnerabilidad listadas, manejar mensaje en caso de que no exista ninguna solución registrada.	

Iteración 4

En el transcurso de esta iteración, se efectuaron las modificaciones requeridas por el cliente en la entrega anterior. Posteriormente, se llevaron a cabo la implementación de las HU: Mostrar Cantidad de Incidencias y Generar Reporte. Se establecieron dos TP.

HU8: Mostrar Cantidad de Incidencias

Tabla 21. Tarea de programación Mostrar cantidad de incidencia

Tarea de programación	
Número de tarea: 23	**Historia de Usuario:** 8. HU_Mostrar Cantidad de Incidencias
Nombre de la tarea: Mostrar Cantidad de Incidencias	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.4

Fecha de inicio: 16 de noviembre de 2023	**Fecha de fin:** 27 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista gráfica para mostrar la cantidad de incidencia por BD, mostrar filtro por fecha, manejar mensaje en que el grafico esté vacío.	

HU9: Generar Reporte

Tabla 22. Tarea de programación Generar reporte

Tarea de programación	
Número de tarea: 24	**Historia de Usuario:** 9. HU_Generar Reporte
Nombre de la tarea: Generar Reporte	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.4
Fecha de inicio: 28 de noviembre de 2023	**Fecha de fin:** 2 de diciembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear vista de barra de búsqueda para las tablas: base de datos, incidencias, vulnerabilidad y soluciones.	

3.3. Pruebas

3.3.1 Estrategia de prueba

Las pruebas de software son procesos que permiten verificar y revelar la calidad de un producto. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa (PRESSMAN, 2010). Las estrategias de pruebas están diseñadas para garantizar que el software cumpla con los requisitos del cliente y mantenga un alto nivel de calidad. XP promueve prácticas ágiles y flexibles en términos de pruebas.

3.3.2 Niveles de pruebas

Los niveles de pruebas proporcionan una estructura para asegurar la calidad del software a medida que avanza a través de las etapas del ciclo de vida del desarrollo. Aquí se describen los niveles de pruebas según Pressman:

1. Pruebas de Unidad.
2. Pruebas de Integración.
3. Pruebas de Sistema.
4. Pruebas de Aceptación del Usuario (UAT).
5. Pruebas de Regresión.
6. Pruebas de Humo (Smoke Testing).

3.3.3 Métodos caja blanca y caja negra

Las pruebas de caja negra también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa (PRESSMAN, 2010).

Las pruebas de caja negra intentan encontrar errores en las categorías siguientes:

1. Funciones incorrectas o faltantes.
2. Errores de interfaz.
3. Errores en las estructuras de datos o en el acceso a bases de datos externas.
4. Errores de comportamiento o rendimiento.
5. Errores de inicialización y terminación.

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba (PRESSMAN, 2010).

Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

1. Garantizar que todas las rutas independientes dentro de un módulo se revisaron al menos una vez
2. Revisar todas las decisiones lógicas en sus lados verdadero y falso
3. Ejecutar todos los bucles en sus fronteras y dentro de sus fronteras operativas

4. Revisar estructuras de datos internas para garantizar su validez.

3.3.4 Pruebas unitarias:

La creación de pruebas unitarias antes de que comience la codificación es un elemento clave del enfoque de XP. Las pruebas unitarias que se crean deben implementarse con el uso de una estructura que permita automatizarlas (de modo que puedan ejecutarse en repetidas veces y con facilidad) (Martínez González, 2019). El requisito mínimo que se establece para culminar una iteración es que todas las funcionalidades implementadas superen cada uno de sus test unitarios.

En la Figuras 17 se muestran los resultados de la ejecución de los test unitarios para las HU donde todas superaron el test unitario.

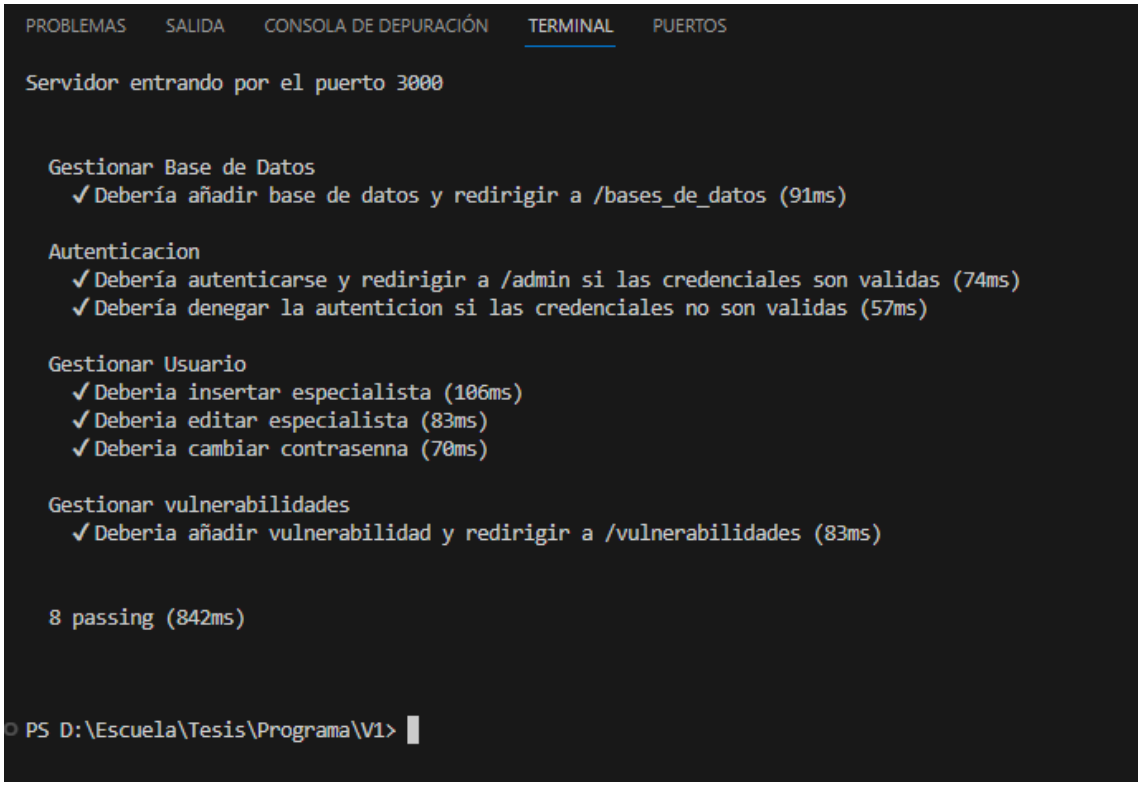


Figura 18. Test Unitario por HU

3.3.5 Prueba de aceptación:

Son aquellas en las que los usuarios o expertos en el negocio validan los requisitos funcionales. Una propuesta temprana que se definen directamente desde los requisitos. Esto implica que el equipo de desarrollo, una vez definidos los requerimientos y validados éstos con el usuario, no son los encargados de definir o diseñar dichas pruebas, sino que estas se generan de manera sistemática, desde los requisitos del usuario (González et al., 2014).

Iteración 1.

Tabla 23. Prueba de aceptación HU1

Caso de Prueba de aceptación	
Código: CP1	**Historia de Usuario:** 1
Nombre: HU_Autenticar Usuario	
Descripción: La prueba se centra verifica el proceso de autenticar a un usuario en el sistema.	
Condición de ejecución: El sistema está en un estado donde el usuario ya esté registrado y se haya establecido la funcionalidad de autenticación.	
Pasos de ejecución: 1. Ingresar al formulario de inicio de sesión. 2. Ingresar el nombre de usuario y la contraseña. 3. Hacer clic en el botón de "Iniciar Sesión".	
Resultados esperados: Éxito de Autenticación: 1. Redirigir al usuario a la página principal. 2. Mostrar un mensaje de bienvenida. 3. El estado de autenticación del usuario cambia a "autenticado". Fallo de Autenticación: 1. Si las credenciales son incorrectas, el sistema muestra un mensaje de error. 2. El estado de autenticación del usuario no cambia.	

Tabla 24. Prueba de aceptación HU2

Caso de Prueba de aceptación	
Código: CP2	**Historia de Usuario:** 2
Nombre: HU_Gestionar Usuario	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta gestión de usuarios en el sistema, abordando la creación, actualización y eliminación de cuentas de usuario.	

Condición de ejecución: La funcionalidad de gestión de usuarios debe estar habilitada y el sistema debe encontrarse en un estado adecuado para la ejecución de las pruebas.
Pasos de ejecución: Añadir Nuevo Usuario: 1. Hacer clic en el botón de "Añadir". 2. Completar los campos del usuario. 3. Hacer clic en el botón de "Guardar". Modificar Usuario: 1. Seleccionar un usuario existente desde la lista de usuarios. 2. Hacer clic en el botón de "Añadir". 3. Actualizar los campos del usuario. 4. Hacer clic en el botón de "Guardar". Eliminar Usuario: 1. Seleccionar un usuario existente desde la lista de usuarios. 2. Hacer clic en el botón de "Eliminar".
Resultados esperados: Inserción Exitosa: Confirmar que el sistema muestre un mensaje de confirmación y que el nuevo usuario aparezca en la lista de usuarios. Actualización Exitosa: Confirmar que el sistema muestre un mensaje de confirmación y que los cambios realizados se reflejen correctamente en los detalles del usuario. Eliminación Exitosa: Confirmar que el sistema muestre un mensaje de confirmación y que el usuario eliminado no aparezca en la lista de usuarios.

Iteración 2.

Tabla 25. Prueba de aceptación HU3

Caso de Prueba de aceptación

Código: CP3	**Historia de Usuario:** 3
Nombre: HU_Cambiar Contraseña	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta funcionalidad de cambio de contraseña para un usuario en el sistema.	
Condición de ejecución: El usuario debe estar autenticado en el sistema.	
Pasos de ejecución: 1. Buscar y acceder a la sección de cambio de contraseña. 2. Ingresar la contraseña actual del usuario. 3. Ingresar la nueva contraseña. 4. Confirmar la nueva contraseña. 5. Hacer clic en el botón de "Guardar".	
Resultados esperados: Cambio de Contraseña Exitoso: Verificar que el sistema muestre un mensaje de confirmación y que el usuario pueda iniciar sesión con la nueva contraseña. Fallo en el Cambio de Contraseña: Verificar que el sistema muestre un mensaje de error si la contraseña actual no es correcta o si la confirmación de la nueva contraseña no coincide.	

Tabla 26. Prueba de aceptación HU4

Caso de Prueba de aceptación	
Código: CP4	**Historia de Usuario:** 4
Nombre: HU_Gestionar Base de Datos	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta gestión de incidencias en el sistema, incluyendo la creación, actualización y cierre de incidencias.	
Condición de ejecución: El sistema debe estar en un estado adecuado para la ejecución de pruebas, y el usuario debe tener los permisos necesarios para gestionar la base de datos.	
Pasos de ejecución:	

Añadir Nueva BD: 1. Hacer clic en el botón de "Añadir". 2. Completar los campos de la BD. 3. Hacer clic en el botón de "Guardar". Modificar BD: 1. Seleccionar una BD existente desde la lista. 2. Hacer clic en el botón de "Añadir". 3. Actualizar los campos de la BD. 4. Hacer clic en el botón de "Guardar". Eliminar BD: 1. Seleccionar una BD existente desde la lista de BD. 2. Hacer clic en el botón de "Eliminar".
Resultados esperados: Inserción Exitosa de Registro: Confirmar que el sistema muestre un mensaje de confirmación y que el nuevo registro esté presente en la base de datos. Actualización Exitosa de Registro: Confirmar que el sistema muestre un mensaje de confirmación y que los cambios realizados se reflejen correctamente en la base de datos. Eliminación Exitosa de Registro: Confirmar que el sistema muestre un mensaje de confirmación y que el registro eliminado no esté presente en la base de datos.

Iteración 3.

Tabla 27. Prueba de aceptación HU5

Caso de Prueba de aceptación	
Código: CP5	**Historia de Usuario:** 5
Nombre: HU_Gestionar Incidencia	

Descripción: Este caso de prueba tiene como objetivo verificar la correcta gestión de incidencias en el sistema, incluyendo la creación, actualización y cierre de incidencias.
Condición de ejecución: El sistema debe estar en un estado adecuado para la ejecución de pruebas, y el usuario debe tener los permisos necesarios para gestionar incidencias.
Pasos de ejecución: Añadir Incidencia: 1. Hacer clic en el botón de "Añadir". 2. Completar los campos de la incidencia a registrar. 3. Hacer clic en el botón de "Guardar". Modificar incidencia: 1. Seleccionar una incidencia existente desde la lista. 2. Hacer clic en el botón de "Añadir". 3. Actualizar los campos de la incidencia. 4. Hacer clic en el botón de "Guardar". Eliminar incidencia: Seleccionar una incidencia existente en la lista de incidencia. Hacer clic en el botón de "Eliminar".
Resultados esperados: Inserción Exitosa de Incidencia: Confirmar que el sistema muestre un mensaje de confirmación y que la nueva incidencia esté registrada correctamente en el sistema. Actualización Exitosa de Estado de Incidencia: Confirmar que el sistema muestre un mensaje de confirmación y que el estado de la incidencia se actualice correctamente. Eliminación Exitoso de Incidencia: Confirmar que el sistema muestre un mensaje de confirmación y que el registro eliminado no esté presente.

Tabla 28. Prueba de aceptación HU6

Caso de Prueba de aceptación

Código: CP6	**Historia de Usuario:** 6
Nombre: HU_Gestionar Vulnerabilidad	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta gestión de vulnerabilidades en el sistema, abordando las operaciones de insertar, modificar y eliminación.	
Condición de ejecución: El sistema debe estar en un estado adecuado para la ejecución de pruebas, y el usuario debe tener los permisos necesarios para gestionar vulnerabilidades.	
Pasos de ejecución: Añadir vulnerabilidad: 1. Hacer clic en el botón de "Añadir". 2. Completar los campos de vulnerabilidad a registrar. 3. Hacer clic en el botón de "Guardar". Modificar vulnerabilidad: 1. Seleccionar una vulnerabilidad existente desde la lista. 2. Hacer clic en el botón de "Añadir". 3. Actualizar los campos de vulnerabilidad. 4. Hacer clic en el botón de "Guardar". Eliminar vulnerabilidad: 1. Seleccionar una vulnerabilidad existente en la lista de vulnerabilidades. 2. Hacer clic en el botón de "Eliminar".	
Resultados esperados: Inserción Exitosa de Vulnerabilidad: Confirmar que el sistema muestre un mensaje de confirmación y que la nueva vulnerabilidad esté registrada correctamente en el sistema. Modificación Exitosa de Datos de Vulnerabilidad: Confirmar que el sistema muestre un mensaje de confirmación y que los cambios realizados se reflejen correctamente en la vulnerabilidad. Eliminación Exitosa de Vulnerabilidad: Confirmar que el sistema muestre un mensaje de confirmación y que la vulnerabilidad eliminada no esté presente en la lista.	

Tabla 29. Prueba de aceptación HU7

Caso de Prueba de aceptación	
Código: CP7	**Historia de Usuario:** 7
Nombre: HU_Gestionar Solución	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta gestión de soluciones para vulnerabilidades de las bases de datos en el sistema, abordando las operaciones de creación, modificación y eliminación de las soluciones propuestas.	
Condición de ejecución: El sistema debe estar en un estado adecuado para la ejecución de pruebas, y el usuario debe tener los permisos necesarios para gestionar soluciones para vulnerabilidades de bases de datos.	
Pasos de ejecución: Añadir solución: 1. Hacer clic en el botón de "Añadir". 2. Completar los campos de la solución a registrar. 3. Hacer clic en el botón de "Guardar". Modificar solución: 1. Seleccionar una solución existente desde la lista. 2. Hacer clic en el botón de "Añadir". 3. Actualizar los campos de la solución. 4. Hacer clic en el botón de "Guardar". Eliminar solución: 1. Seleccionar una solución existente en la lista de soluciones. 2. Hacer clic en el botón de "Eliminar".	
Resultados esperados: Inserción Exitosa de Solución: Confirmar que el sistema muestre un mensaje de confirmación y que la nueva solución esté registrada correctamente en el sistema. Modificación Exitosa de Datos de Solución:	

Confirmar que el sistema muestre un mensaje de confirmación y que los cambios realizados se reflejen correctamente en la solución. Eliminación Exitosa de Solución: Confirmar que el sistema muestre un mensaje de confirmación y que la solución eliminada no esté presente en la lista.

Iteración 4.

Tabla 30. Prueba de aceptación HU8

Caso de Prueba de aceptación	
Código: CP8	**Historia de Usuario:** 8
Nombre: HU_Mostrar Cantidad de Incidencias	
Descripción: Este caso de prueba tiene como objetivo verificar la correcta funcionalidad de mostrar la cantidad de incidencias por bases de datos en un gráfico, así como la cantidad que fue dada por parte del administrador y el especialista.	
Condición de ejecución: El sistema debe estar en un estado adecuado para la ejecución de pruebas, y los datos de incidencias, bases de datos y roles de administrador y especialista deben estar disponibles.	
Pasos de ejecución: Acceder a la Vista de Gráfico: 1. Iniciar sesión como un usuario con permisos para visualizar el gráfico de incidencias. 2. Navegar a la sección que presenta el gráfico de incidencias. Observar el Gráfico por Bases de Datos: 1. Hacer clic en el botón “Fecha de inicio" y selecciona la fecha. 2. Hacer clic en el botón “Fecha final" y selecciona la fecha. 3. Revisar el gráfico para cada base de datos filtrada por fecha	
Resultados esperados: Visualización Exitosa del Gráfico por Bases de Datos: Confirmar que el gráfico muestre la cantidad de incidencias y así como las solucionadas asociadas a cada base de datos.	

Tabla 31. Prueba de aceptación HU9

Caso de Prueba de aceptación	
Código: CP9	**Historia de Usuario:** 9
Nombre: HU_Generar Reporte	
Descripción: Este caso de prueba verifica la funcionalidad de generar un informe con búsqueda filtrada por id, nombre, fecha, y tipo. El informe debe presentar ordenadamente la información relevante que cumple con los criterios de búsqueda, facilitando el análisis detallado.	
Condición de ejecución: El sistema debe estar preparado para las pruebas, y se deben disponer de datos de prueba que se ajusten a los criterios de búsqueda.	
Pasos de ejecución: Realizar Búsqueda Filtrada: 1. Hacer clic en la barra de búsqueda. 2. Ingresar valores específicos para cada criterio. 3. Ejecutar la búsqueda.	
Resultados esperados: Generación Exitosa del Informe con Búsqueda Filtrada: Confirmar que el informe incluya solo registros que cumplen con los criterios de búsqueda.	

3.3.6 Resumen de los resultados de la prueba de aceptación

Estas pruebas, realizadas típicamente al final del ciclo de desarrollo, tienen como objetivo asegurar que la aplicación o sistema entregado satisface las expectativas del cliente y se comporta conforme a las especificaciones establecidas.

Las iteraciones sucesivas en el desarrollo de software se llevan a la identificación y corrección de no conformidades o problemas detectados en las pruebas de aceptación. Estas no conformidades abarcan errores de ortografía y validación y funcionalidad del sistema. A continuación, los resultados:

- **En la primera iteración,** se identificaron un total de 7 no conformidades (NC): 2 relacionados con ortografía, 3 de validación y 2 de índole funcional. Se llevaron a cabo cinco pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

- **En la segunda iteración,** se encontraron un total de 3 no conformidades (NC): 2 de validación y 1 de naturaleza funcional. Se llevaron a cabo tres pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

- **Durante la tercera iteración,** se detectó dos hallazgos de no conformidades (NC), una de validación y otra de ortografía. Se llevó a cabo una pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

- **En la última iteración,** se registraron cuatro errores no conformidades (NC): 1 de validación y 3 de naturaleza funcional. Se llevaron a cabo cuatro pruebas de aceptación con el fin de abordar y corregir todas las observaciones de no conformidades.

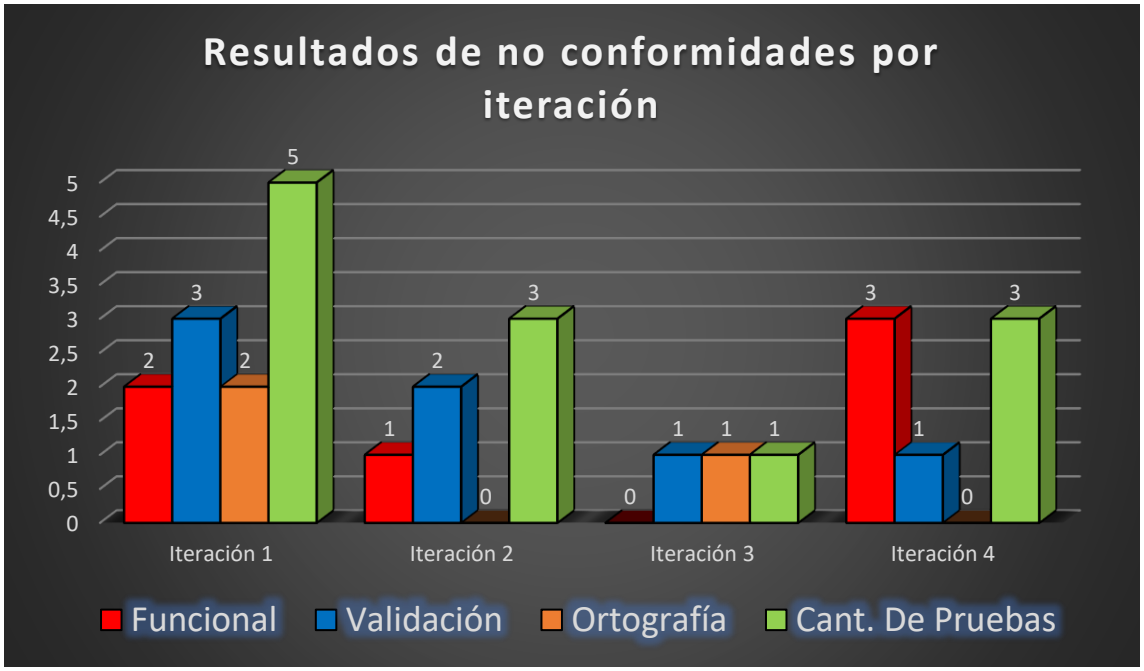


Figura 19. Resultados de no conformidades

A medida que se avanzaba en el desarrollo del sistema, disminuyó la cantidad de no conformidades detectadas en las pruebas de aceptación. Al finalizar todas las pruebas de aceptación no se identificaron errores de ningún tipo, esto refleja la madurez y estabilidad alcanzada en el producto, indicando que se ha logrado una alineación exitosa con los requisitos y expectativas del cliente.

3.4. Conclusiones parciales

Este capítulo refleja la ejecución exitosa de la fase de implementación y pruebas, cumpliendo con los objetivos específicos tercero y cuarto de la investigación. Las cuatro iteraciones planificadas se llevaron a cabo, implementando un total de 9 Historias de Usuario (HU) desglosadas en 24 Tareas de Programación (TP). La elaboración del diagrama de despliegue proporcionó una comprensión clara de la implementación del sistema en el mundo real y facilitó la anticipación de posibles

problemas durante la etapa de despliegue. La aplicación de estándares de codificación contribuyó a la legibilidad y mantenimiento eficiente del código. La ejecución de las TP permitió cumplir con el plan de entrega de las HU, y las pruebas unitarias y de aceptación aseguraron la correcta implementación de las funcionalidades, validando así que el sistema cumple con las necesidades del cliente. Este capítulo representa un hito sustancial, donde la teoría se convierte en avances tangibles, marcando el camino hacia la conclusión exitosa de la investigación.

CONCLUSIONES

Con la culminación de este Trabajo de Diploma, se derivan las siguientes conclusiones significativas:

- La elaboración del marco teórico estableció las bases conceptuales esenciales para el sistema informático propuesto, delineando claramente la naturaleza y el comportamiento de sus funcionalidades.
- Las tareas de investigación se llevaron a cabo de manera completa, resultando en el desarrollo exitoso del Sistema para la identificación de vulnerabilidades de seguridad en SGBD como el principal logro del estudio.
- El análisis de sistemas similares reveló que ninguno abordaba integralmente la problemática identificada, motivando así la investigación y el diseño de la solución propuesta.
- La elección de herramientas y tecnologías, como Node JS, Mongo DB, Express, Visual Studio Code, respaldada por la metodología XP, fundamentó la implementación de la solución y contribuyó a su eficiencia.
- La especificación detallada de las historias de usuario sirvió como punto de partida sólido para la concepción y diseño de la propuesta de solución, asegurando una comprensión completa del software a desarrollar.
- La aplicación del patrón arquitectónico MVC resultó fundamental, al separar los datos, la interfaz de usuario y la lógica de negocio en componentes distintos, facilitando así la implementación y el mantenimiento continuo del sistema.
- La adherencia a estándares de codificación garantizó una implementación organizada y responsable, proporcionando un sistema robusto y permitiendo futuros mantenimientos con eficacia.
- La validación exhaustiva del sistema mediante pruebas unitarias y de aceptación aseguró su estabilidad y corrección, proporcionando una solución confiable que exhibe robustez y flexibilidad para adaptarse a cambios posteriores. Estas conclusiones reflejan el éxito y la calidad alcanzada en el desarrollo.

RECOMENDACIONES

- Agregarle al sistema la funcionalidad de predecir un patrón de comportamiento en cuanto a las vulnerabilidades, es decir predecir un comportamiento de anomalías sobre una base de datos específica dadas sus vulnerabilidades.
- Incorporal al sistema herramientas de inteligencia artificial para que detecte vulnerabilidades, y se registren directamente de manera automática.
- Sistema incluya funcionalidades que permitan el escaneo de vulnerabilidades a partir de una minería de log (historial de eventos) de las bases de datos.

REFERENCIAS BIBLIOGRÁFICAS

- Arbeláez Gómez, M. C. (2014). "Las tecnologías de la información y la comunicación (TIC) un instrumento para la investigación." Investigaciones Andina **16**(29): 997-1000.
- Beck, K. (1999). Extreme Programming Explained, First Edit, Addison-Wesley Professional.
- Beynon-Davies, P. (2018). Sistemas de bases de datos, Reverté.
- Buendía, J. F. R. (2013). Seguridad informática, McGraw-Hill España.
- Cassany, D. (2005). Investigaciones y propuestas sobre literacidad actual: multiliteracidad, internet y criticidad. Conferencia presentada en Congreso Nacional Cátedra UNESCO para la lectura y la escritura, Universidad de Concepción.
- COLÓN COLÓN, C. M. (2021). "DAVID KENT, FUNDADOR DEL WEBSITE DE REDES DE PETRÓLEO Y GAS SE DECLARA CULPABLE DE UN CARGO DE FRAUDE INFORMÁTICO."
- Fernández, A. and C. Claudia (2020). "Aplicación web para fortalecer la gestión del transporte en la recolección de residuos sólidos urbanos."
- Franco, D. A., et al. (2013). "Herramienta para la Detección de Vulnerabilidades basada en la Identificación de Servicios." Información tecnológica **24**: 13-22.
- Ghosh, S. (2015). "Systemic comparison of the application of EVM in traditional and agile software project." Integration **5**(3).
- Guzmán Quesada, P. A. (2019). Análisis de las vulnerabilidades en sistemas gestores de bases de datos, Ecuador-PUCESE-Escuela de Sistemas y Computación.
- Khurana, H. and J. Sohal (2011). "Agile: The necessitate of contemporary software developers." International Journal of Engineering Science and Technology **3**(2).
- Letelier, P. (2006). "Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)."
- Maida, E. G. and J. Pacienzia (2015). "Metodologías de desarrollo de software."
- Marcelo, V. and G. Ángel (2023). "Análisis de la protección de la información digital de las Fuerzas Armadas en el marco de la política de seguridad y defensa nacional en la región Lima, 2018."
- Menzinsky, A., et al. (2018). "Historias de usuario." Ingeniería de requisitos ágiPérez, M. T. G. (2010). "Sistemas Gestores de Bases de Datos." Granada: sn.
- Puciarelli, L. (2020). Node JS-Vol. 1: Instalación-Arquitectura-node y npm, RedUsers.
- Ronald, J. (2012). "What is extreme programming [Internet]." Disponible desde http://xprogramming. com/what-is-extreme-programming/[Acceso Junio 1, 2013].
- Sato, D., et al. (2006). "Experiences tracking agile projects: an empirical study." Journal of the Brazilian Computer Society **12**: 45-64.
- Schwaber, K. and M. Beedle (2001). Agile software development with Scrum, Prentice Hall PTR.
- Tejada, E. C. (2023). Gestión de incidentes de seguridad informática. IFCT0109, IC Editorial.
- Wells, D. (2009). "The Values of Extreme Programming." Retrieved from Extreme Programming: http://www. extremeprogramming. org/values. html.

- Zambrano, S. M. Q. and D. G. M. Valencia (2017). "Seguridad en informática: consideraciones." Dominio de las Ciencias **3**(3): 676-688.
- Alvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. *Polo del Conocimiento: Revista científico-profesional*, *7*(7), 2146-2165.
- Ampuero, M. A., & Trujillo, Y. L. (2016). Creando un profesional con disciplina en el proceso de desarrollo de software. *Ingeniería Industrial*, *27*(1), 27-30.
- Ávila Garzón, C. (2019). Modelo vista controlador. *Ingeniería de Software*.
- Carmona García, E. A., & Thompson Martínez, R. (2010). *Notificacion de Vulnerabilidades de Tecnologias y Sistemas Informaticos* [B.S. thesis].
- Carrillo Peña, A. L. (2020). *Prácticas académicas en Electrosoftware S.A.S*. https://repository.unab.edu.co/handle/20.500.12749/14272
- Catallops, S., & José, L. (2019). *Sistema de gestión de información tecnológica para la Facultad 1 de la Universidad de las Ciencias Informáticas*. Universidad de las Ciencias Informáticas. Facultad 1.
- Cortina Perdomo, R. (2018). *Sistema de Gestión de Informes de Ensayos* [PhD Thesis, Departamento de Informática]. http://ninive.ismm.edu.cu/handle/123456789/2652
- Giraldo, G. L., Acevedo, J. F., & Moreno, D. A. (2011). Una ontología para la representación de conceptos de diseño de software. *Revista Avances en Sistemas e Informática*, *8*(3), 103-110.
- Herrera, C. M., Bruchmann, E. C. S., Barrera, M. A., & Korzeniewski, M. I. (2021). *Modelado CRC Aplicado al Sistema de Gestión de Recursos Humanos.*
- *MongoDB: La Plataforma De Datos Para Aplicaciones*. (2023). MongoDB. https://www.mongodb.com/es
- Pérez Delgado, B. (2020). *Mantenimiento perfectivo de los directorios del portal web Intranet*. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS. FACULTAD 3.
- Piñeiro Vidal, E., & Roa Parets, R. (2020). *Módulo para el control del personal civil del Sistema de gestión de la información y control estadístico del Ministerio del Interior*. Universidad de las Ciencias Informáticas. Facultad 3.
- PRESSMAN, R. S. (2010). Ingeniería de software enfoque practico. Pressman. PDF. *Ingeniería del software, un enfoque práctico*.
- Robert, E., & del Carmen, A. (2015). *Sistema Auditoría de Datos: Módulo de Gráficos v2. 0* [B.S. thesis, Universidad de las Ciencias Informáticas. Facultad 6]. https://repositorio.uci.cu/handle/123456789/7410
- NVD - hogar . (s/f). Nist.gov. Recuperado el 27 de noviembre de 2023, de https://nvd.nist.gov/
- prevent unauthorized database changes, & Industry, H. C. M. (s/f). Oracle Database Vault. Oracle.com. Recuperado el 28 de noviembre de 2023, de https://www.oracle.com/a/tech/docs/wp-dv-20c.pdf

- USM Anywhere. (2020, febrero 12). GB Advisors. https://www.gb-advisors.com/es/siem-gestion-de-eventos/alienvault-siem-seguridad-digital/
- (S/f-a). Recuperado el 27 de noviembre de 2023, de https://cybersecurity-att-com
- (S/f-b). Rae.es. Recuperado el 29 de noviembre de 2023, de https://dle.rae.es/prioridad
- (S/f-c). Tec.mx. Recuperado el 30 de noviembre de 2023, de https://edutools.tec.mx/

ANEXOS

ANEXO 1. Historias de usuarios

Tabla 32. Historia de Usuario 4

Historia de Usuario	
Número: 4	**Nombre:** HU_Gestionar Base de Datos
Usuario: Administrador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 1.4	**Iteración Asignada:** 3
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: El administrador gestionará las Bases de Datos, incluye crear, modificar, eliminar y mostrar la información de las Bases de Datos. Este proceso asegura un control efectivo sobre la estructura y el contenido de las bases de datos, lo que permite una administración eficiente y precisa de la información almacenada.	
Observación: Se mostrará id base de datos, nombre, Criticidad (Muy Crítico, Crítico, Moderadamente Crítico, Menos Crítico y Poco Crítico), Prioridad (2-8 se calculará, se mostrará “No hay riesgo” en caso de que no existan incidencia), SGBD, tipo (SQL o NOSQL), versión y fecha de actualización.	

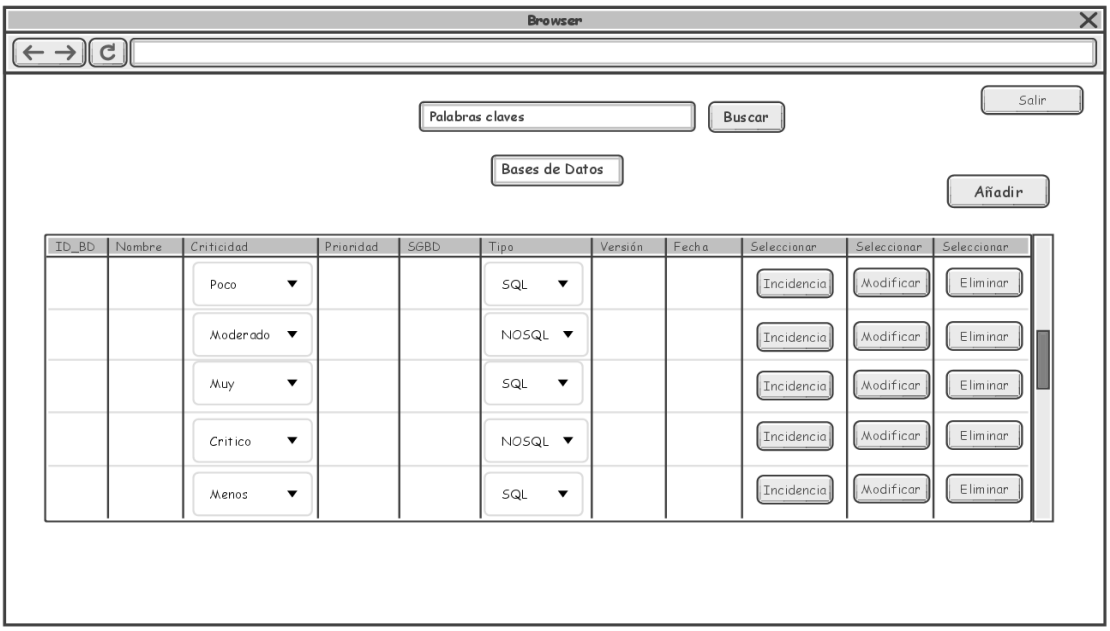


Tabla 33. Historia de Usuario 5

Historia de Usuario	
Número: 5	**Nombre:** HU_Gestionar Incidencia
Usuario: Administrador, Especialista	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 1.5	**Iteración Asignada:** 2
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: El administrador y el especialista gestionarán las incidencias, incluye crear, modificar, eliminar y mostrar la información de las incidencias; esto proporciona un marco estructurado para abordar y solucionar eficazmente cualquier contratiempo, asegurando así la continuidad y la seguridad de las operaciones de la base de datos. Esto garantiza un manejo efectivo de cualquier problema o anomalía que pueda surgir en el contexto de las bases de datos, lo que a su vez contribuye a mantener la integridad y la disponibilidad de los datos.	
Observación: Se mostrará identificador, tipo (Vulnerabilidad), riesgo (alto, medio, bajo), responsable de detectarlo (responsable en detectar la incidencia), estado (Solucionado, En proceso y No solucionado) y fecha de detención.	

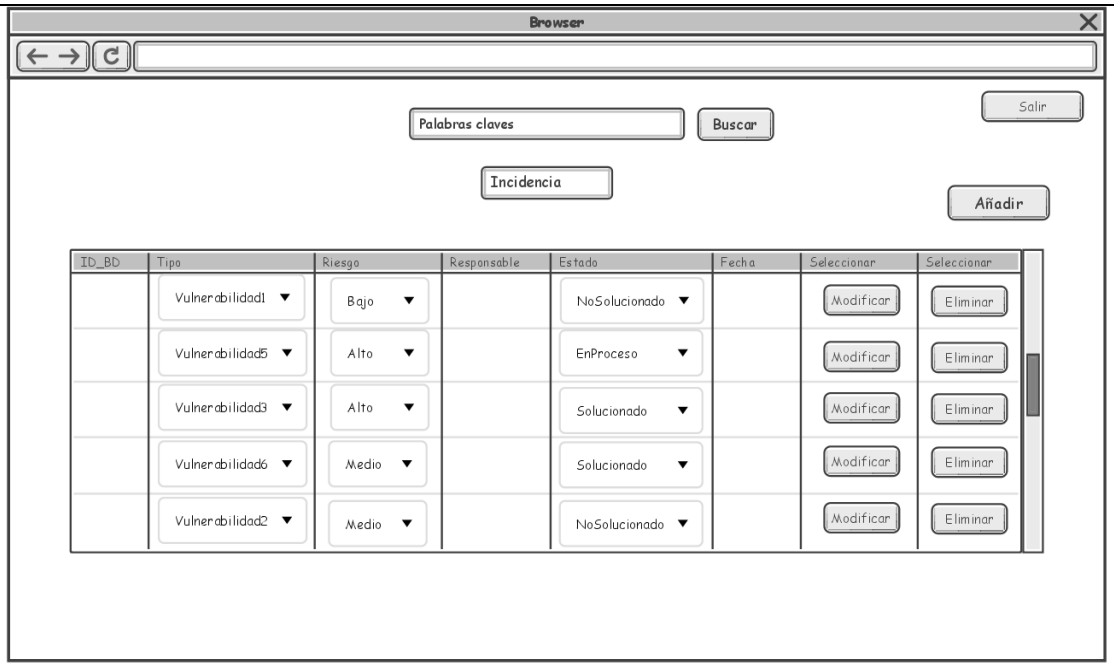


Tabla 34. Historia de Usuario 6

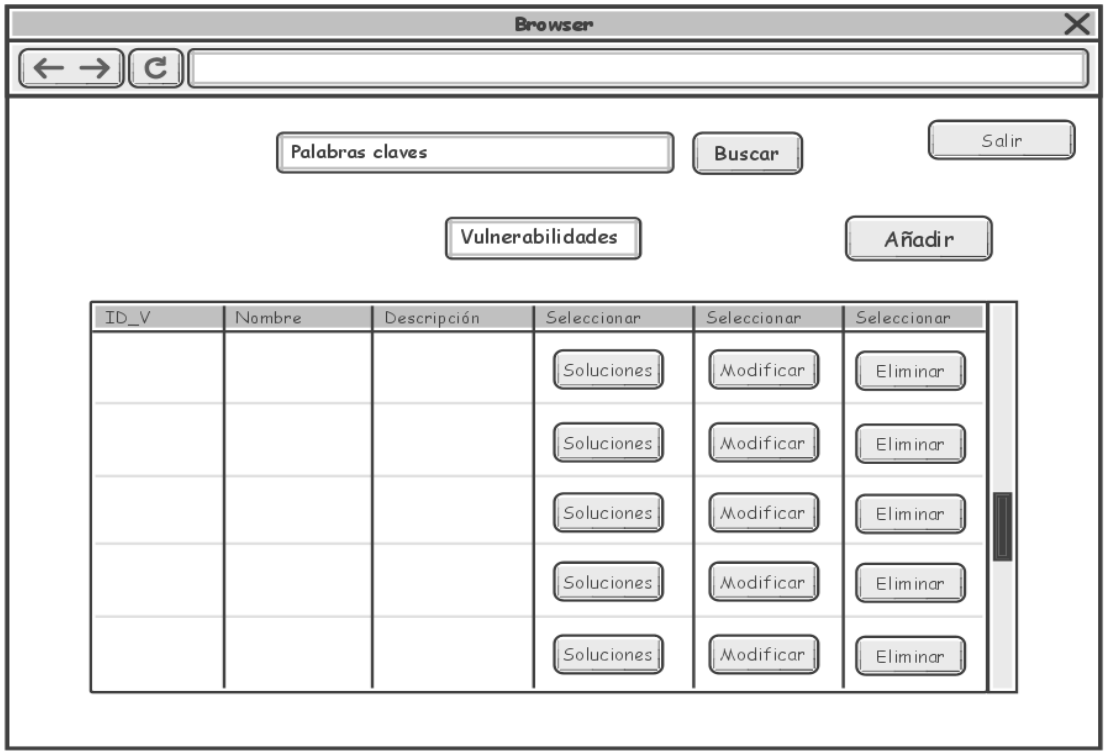
Historia de Usuario	
Número: 6	**Nombre:** HU_Gestionar Vulnerabilidad
Usuario: Administrador, Especialista	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Medio
Puntos Estimados: 1.5	**Iteración Asignada:** 3
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: El Administrador y Usuario gestionaran las vulnerabilidades, incluye crear, modificar, eliminar y ver la información de las vulnerabilidades esto implica una serie de tareas esenciales, que van desde la identificación y registro de posibles vulnerabilidades hasta su posterior análisis y mitigación. Esta función garantiza un manejo proactivo de posibles riesgos de seguridad que podrían afectar la integridad de los datos.	
Observación: Se mostrará id de vulnerabilidad, nombre y descripción.	
	

Tabla 35. Historia de Usuario 7

Historia de Usuario

Número: 7	**Nombre:** HU_Gestionar Solución
Usuario: Administrador, Especialista	
Prioridad en Negocio: Alta	**Riesgo en Desarrollador:** Bajo
Puntos Estimados: 0.9	**Iteración Asignada:** 4
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: El Administrador y Usuario gestionarán las soluciones, incluye crear, modificar, eliminar y ver las soluciones propuestas. Esta función asegura una respuesta oportuna y eficaz ante posibles riesgos y contratiempos que puedan afectar la integridad y seguridad de la base de datos. Proporciona un marco estructurado para abordar y resolver eficientemente cualquier problema, lo que contribuye significativamente a mantener la continuidad y la fiabilidad del sistema de gestión de bases de datos.	
Observación: Se mostrará id y descripción de las soluciones por cada vulnerabilidad.	

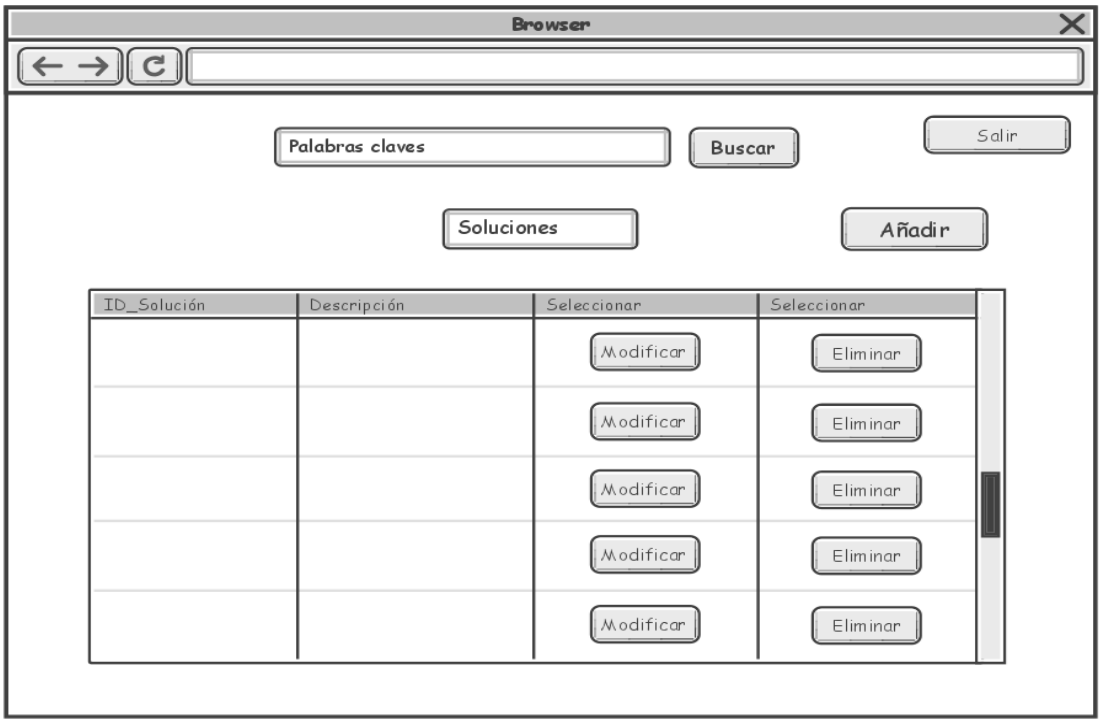


Tabla 36. Historia de Usuario 8

Historia de Usuario	
Número: 8	**Nombre:** HU_Mostrar Cantidad de Incidencias
Usuario: Administrador, Especialista	

Prioridad en Negocio: Media	**Riesgo en Desarrollador:** Bajo
Puntos Estimados: 1.4	**Iteración Asignada**: 4
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: La capacidad de mostrar la cantidad de incidencias por bases de datos en un gráfico, así como la cantidad que fueron solucionadas por parte del administrador y el especialista.	
Observación: Esta función ofrece una representación gráfica y comprensible de la importancia relativa de las diferentes bases de datos, lo que facilita la toma de decisiones informadas en cuanto a la asignación de recursos y prioridades para la protección y seguridad de los datos.	

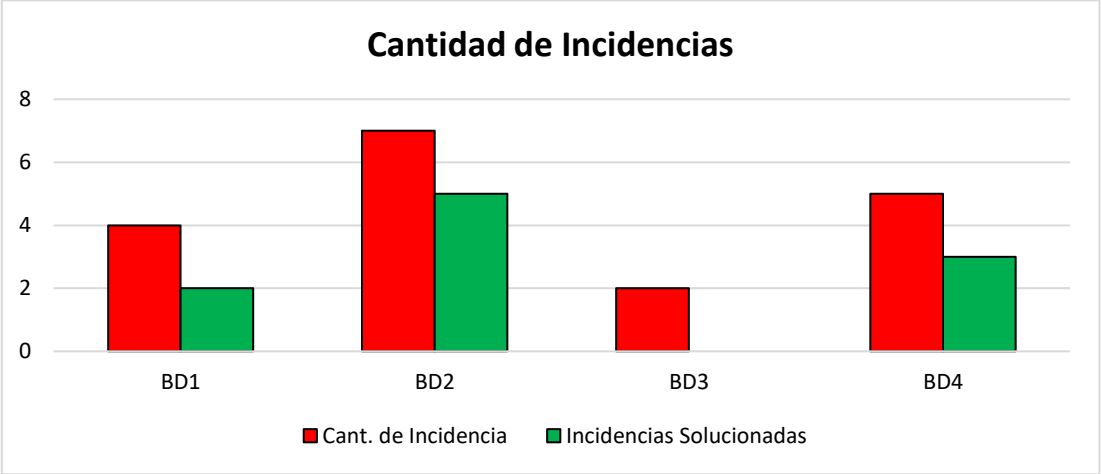


Tabla 37. Historia de Usuario 9

Historia de Usuario	
Número: 9	**Nombre:** HU_Generar Reporte
Usuario: Administrador, Especialista	
Prioridad en Negocio: Media	**Riesgo en Desarrollador:** Bajo
Puntos Estimados: 0.7	**Iteración Asignada:** 4
Programador Responsable: Davier J. Carballo, Luhana Castillo	
Descripción: Una búsqueda filtrada por id, nombre, fecha (incluye un rango), tipo, permite recopilar y presentar de manera ordenada la información relevante que se ajuste a los criterios de búsqueda establecidos, lo que facilita la obtención de datos específicos para su análisis y revisión detallada	

Observación: Se mostrará los resultados de la búsqueda, incluye todos los datos de las BD, incidencia o vulnerabilidad.

ANEXO 2. Iteración 1. Tareas de programación

HU2: Gestionar Usuario

Tabla 38. Tarea de programación Insertar usuario

Tarea de programación	
Número de tarea: 3	**Historia de Usuario:** 2. HU_Gestionar Usuario
Nombre de la tarea: Insertar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 16 de septiembre de 2023	**Fecha de fin:** 18 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para adicionar usuario, mostrar los campos de usuario a registrar, manejar mensaje de registrado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de usuario.	

Tabla 39. Tarea de programación Modificar usuario

Tarea de programación	
Número de tarea: 4	**Historia de Usuario:** 2. HU_Gestionar Usuario
Nombre de la tarea: Modificar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 20 de septiembre de 2023	**Fecha de fin:** 21 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	

Descripción: Crear plantilla y vista para editar un usuario ya registrado, mostrar los campos de usuario a actualizar, manejar mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de usuario.

Tabla 40. Tarea de programación Eliminar usuario

Tarea de programación	
Número de tarea: 5	**Historia de Usuario:** 2. HU_Gestionar Usuario
Nombre de la tarea: Eliminar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 22 de septiembre de 2023	**Fecha de fin:** 25 de septiembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para eliminar un usuario ya registrado, mostrar confirmación de aceptar o cancelar, manejar mensaje de eliminado correctamente, redireccionar a la url donde se encuentran la lista de usuario.	

ANEXO 3. Iteración 2. Tareas de programación

HU4: Gestionar Base de Datos

Tabla 41. Tarea de programación Insertar BD

Tarea de programación	
Número de tarea: 8	**Historia de Usuario:** 4. HU_Gestionar Base de Datos
Nombre de la tarea: Insertar Base de Datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 4 de octubre de 2023	**Fecha de fin:** 6 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para adicionar una BD, mostrar los campos de la BD a registrar, manejar mensaje de registrado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de BD.	

Tabla 42. Tarea de programación Modificar BD

Tarea de programación	
Número de tarea: 9	**Historia de Usuario:** 4. HU_Gestionar Base de Datos
Nombre de la tarea: Modificar Base de Datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 7 de octubre de 2023	**Fecha de fin:** 10 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para editar una BD ya registrado, mostrar los campos de la BD a actualizar, manejar mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de BD.	

Tabla 43. Tarea de programación Eliminar BD

Tarea de programación	
Número de tarea: 10	**Historia de Usuario:** 4. HU_Gestionar Base de Datos
Nombre de la tarea: Eliminar Base de Datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 11 de octubre de 2023	**Fecha de fin:** 12 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para eliminar BD ya registrada, mostrar confirmación de aceptar o cancelar, manejar mensaje de eliminado correctamente, redireccionar a la url donde se encuentran la lista de BD.	

ANEXO 4. Iteración 3. Tareas de programación

HU5: Gestionar Incidencia

Tabla 44. Tarea de programación Insertar incidencia

Tarea de programación

Número de tarea: 12	**Historia de Usuario:** 5. HU_Gestionar Incidencia
Nombre de la tarea: Insertar Incidencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 16 de octubre de 2023	**Fecha de fin:** 18 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para adicionar incidencia, mostrar los campos de la incidencia a registrar, manejar mensaje de registrado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de incidencia.	

Tabla 45. Tarea de programación Modificar incidencia

Tarea de programación	
Número de tarea: 13	**Historia de Usuario:** 5. HU_Gestionar Incidencia
Nombre de la tarea: Modificar Incidencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 19 de octubre de 2023	**Fecha de fin:** 21 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para editar una incidencia ya registrada, mostrar los campos de la incidencia a actualizar, manejar mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de incidencia.	

Tabla 46. Tarea de programación Eliminar incidencia

Tarea de programación	
Número de tarea: 14	**Historia de Usuario:** 5. HU_Gestionar Incidencia
Nombre de la tarea: Eliminar Incidencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 23 de octubre de 2023	**Fecha de fin:** 25 de octubre de 2023

Programador Responsable: Davier J. Carballo, Luhana Castillo.
Descripción: Crear plantilla y vista para eliminar una incidencia ya registrada, mostrar confirmación de aceptar o cancelar, manejar mensaje de eliminado correctamente, redireccionar a la url donde se encuentran la lista de incidencia.

HU6: Gestionar Vulnerabilidad

Tabla 47. Tarea de programación Insertar vulnerabilidad

Tarea de programación	
Número de tarea: 16	**Historia de Usuario:** 6. HU_Gestionar Vulnerabilidad
Nombre de la tarea: Insertar Vulnerabilidad	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 28 de octubre de 2023	**Fecha de fin:** 31 de octubre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para adicionar vulnerabilidad, mostrar los campos de la vulnerabilidad a registrar, manejar mensaje de registrado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de vulnerabilidad.	

Tabla 48. Tarea de programación Modificar vulnerabilidad

Tarea de programación	
Número de tarea: 17	**Historia de Usuario:** 6. HU_Gestionar Vulnerabilidad
Nombre de la tarea: Modificar Vulnerabilidad	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 1 de noviembre de 2023	**Fecha de fin:** 3 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para editar una vulnerabilidad ya registrada, mostrar los campos de la vulnerabilidad a actualizar, manejar mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de vulnerabilidad.	

Tabla 49. Tarea de programación Eliminar vulnerabilidad

Tarea de programación	
Número de tarea: 18	**Historia de Usuario:** 6. HU_Gestionar Vulnerabilidad
Nombre de la tarea: Eliminar Vulnerabilidad	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 4 de noviembre de 2023	**Fecha de fin:** 7 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para eliminar una vulnerabilidad ya registrada, mostrar confirmación de aceptar o cancelar, manejar mensaje de eliminado correctamente, redireccionar a la url donde se encuentran la lista de vulnerabilidad.	

HU7: Gestionar Solución

Tabla 50. Tarea de programación Insertar solución

Tarea de programación	
Número de tarea: 20	**Historia de Usuario:** 7. HU_Gestionar Solución
Nombre de la tarea: Insertar Solución	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 9 de noviembre de 2023	**Fecha de fin:** 10 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para adicionar solución, mostrar los campos de solución a registrar, manejar mensaje de registrado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de soluciones.	

Tabla 51. Tarea de programación Modificar solución

Tarea de programación	
Número de tarea: 21	**Historia de Usuario:** 7. HU_Gestionar Solución
Nombre de la tarea: Modificar Solución	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 11 de noviembre de 2023	**Fecha de fin:** 13 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para editar una solución ya registrada, mostrar los campos de la solución a actualizar, manejar mensaje de modificado correctamente o de formulario incorrecto, redireccionar a la url donde se encuentran la lista de soluciones.	

Tabla 52. Tarea de programación Eliminar solución

Tarea de programación	
Número de tarea: 22	**Historia de Usuario:** 7. HU_Gestionar Solución
Nombre de la tarea: Eliminar Solución	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 14 de noviembre de 2023	**Fecha de fin:** 15 de noviembre de 2023
Programador Responsable: Davier J. Carballo, Luhana Castillo.	
Descripción: Crear plantilla y vista para eliminar una solución ya registrada, mostrar confirmación de aceptar o cancelar, manejar mensaje de eliminado correctamente, redireccionar a la url donde se encuentran la lista de soluciones.	