Evaluating and mitigating SQL injections in web applications: developing a prototype

ARTICLE HISTORY
Received 6 may 2025
Accepted 9 june 2025
Published 7 july 2025

Nancy Rodriguez
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
nrodriguez@uteq.edu.ec
ORCID: 0000-0002-0861-4352

Daniel Loor
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
bloorm2@uteq.edu.ec
ORCID: 0009-0002-8110-5375

Lucrecia Llerena
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
lllerena@uteq.edu.ec
ORCID: 0000-0002-4562-6723

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025

Evaluación y mitigación de inyecciones SQL en aplicaciones web: desarrollo de un prototipo

Evaluating and mitigating SQL injections in web applications: developing a prototype

Nancy Rodriguez
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
nrodriguez@uteq.edu.ec

Daniel Loor
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
bloorm2@uteq.edu.ec

Lucrecia Llerena
Quevedo State University
Software Engineering Program
Quevedo, Ecuador
lllerena@uteq.edu.ec

***Resumen*—**En el contexto actual de creciente vulnerabilidad digital, las inyecciones SQL continúan representando una amenaza crítica para la seguridad de aplicaciones web. Ante esta problemática, se desarrolló *SecureSQLTester*, un prototipo orientado a la detección y mitigación de ataques de inyección SQL, diseñado para ser accesible a desarrolladores y pequeñas empresas. La propuesta se fundamentó en una revisión sistemática de técnicas existentes, integrando enfoques clásicos y avanzados de protección. El desarrollo del prototipo se realizó aplicando la metodología ágil Scrum, lo que permitió realizar mejoras progresivas a través de ciclos de trabajo iterativos. Se realizaron pruebas de usabilidad con estudiantes de ingeniería de software, quienes evaluaron la herramienta en escenarios simulados. Los resultados muestran que *SecureSQLTester* identifica con precisión vulnerabilidades SQL en aplicaciones evaluadas. No obstante, se identificaron oportunidades de mejora en la interfaz de usuario, así como la necesidad de ampliar la personalización de parámetros según el contexto de uso. En conjunto, los hallazgos respaldan el potencial del prototipo como herramienta efectiva y de bajo costo para fortalecer la ciberseguridad en entornos de desarrollo de pequeña y mediana escala, y promover la adopción de buenas prácticas en el ciclo de vida del software.

***Palabras clave*—** *inyección SQL, seguridad en aplicaciones web, prototipo, usabilidad, metodología Scrum*

***Abstract*—**In the current context of increasing digital vulnerability, SQL injections continue to pose a critical threat to web application security. To address this issue, SecureSQLTester was developed—a prototype aimed at detecting and mitigating SQL injection attacks, designed to be accessible to developers and small businesses. The proposal was based on a systematic review of existing techniques, integrating both classical and advanced protection approaches. The prototype was developed using the agile Scrum methodology, which enabled progressive improvements through iterative work cycles. Usability tests were conducted with software engineering students, who evaluated the tool in simulated scenarios. The results show that SecureSQLTester accurately identifies SQL vulnerabilities in the evaluated applications. However, opportunities for improvement were identified in the user interface, as well as the need to enhance parameter customization according to the usage context. Overall, the findings support the potential of the prototype as an effective and low-cost tool to strengthen cybersecurity in small- and medium-scale development environments and to promote the adoption of best practices throughout the software lifecycle.

***Keywords*—** *SQL injection, web application security, prototype, usability, Scrum methodology.*

I. Introduccion

En el panorama actual de la transformación digital, las aplicaciones web cumplen una función importante en la organización de actividades cotidianas, desde la comunicación hasta el comercio electrónico [1]. Sin embargo, la creciente dependencia de estos entornos digitales también ha incrementado su exposición a múltiples amenazas de seguridad, entre las cuales las inyecciones SQL destacan como una de las más críticas [2]. Estas inyecciones consisten en la manipulación de consultas SQL mediante la inserción de código malicioso, lo que puede comprometer seriamente la seguridad del sistema, permitir el acceso no autorizado a datos sensibles y generar inconsistencias en las bases de datos [3].

El impacto de este tipo de vulnerabilidades es alarmante. Estudios recientes estiman que más del 76% de los ataques pueden ser detectados mediante técnicas adecuadas, aunque variantes como las "consultas ilegales" siguen siendo difíciles de identificar debido a su sofisticación [3]. Entre los factores que agravan este problema se encuentra la deficiente validación de entradas por parte de los desarrolladores, lo cual facilita la ejecución de estos ataques y representa una amenaza tanto para la estabilidad de los sistemas como para la protección de la información de los usuarios. En este contexto, la implementación de técnicas combinadas mediante lenguajes como JavaScript y PHP ha mostrado ser una estrategia prometedora para separar los datos maliciosos de los normales y fortalecer las medidas de prevención [4], [5].

A pesar del desarrollo de herramientas como SQLMap, QualysGuard y Nessus, su nivel de complejidad, requerimientos técnicos y costos asociados limitan su adopción por parte de pequeñas empresas, instituciones educativas o desarrolladores independientes. Es importante destacar que SQLMap es una herramienta de código abierto, gratuita y ampliamente utilizada en auditorías de seguridad, mientras que soluciones como QualysGuard y Nessus son

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

herramientas propietarias que requieren licencias comerciales y conocimientos técnicos avanzados para su implementación. Esto revela una brecha importante entre las soluciones existentes y las necesidades reales de sectores con recursos limitados, donde el acceso a herramientas de ciberseguridad funcionales, accesibles y fáciles de implementar sigue siendo un desafío no resuelto [6].

Ante esta problemática, surge una pregunta fundamental: ¿Cómo se pueden desarrollar y aplicar de manera efectiva técnicas de seguridad que prevengan y mitiguen los riesgos asociados a ataques de inyección SQL dentro de plataformas web? Esta investigación busca responder a esta interrogante mediante el desarrollo de SecureSQLTester, un prototipo diseñado para detectar y mitigar inyecciones SQL en aplicaciones web. El sistema fue concebido para ser una herramienta funcional, intuitiva y de bajo costo, que pueda ser utilizada por desarrolladores con conocimientos básicos, sin necesidad de infraestructuras complejas.

La propuesta se fundamenta en un análisis sistemático de literatura, mediante el cual se lograron reconocer las técnicas más relevantes de protección frente a inyecciones SQL, y en la aplicación de una metodología ágil basada en Scrum para su desarrollo iterativo. Además, se implementó una evaluación de usabilidad utilizando entrevistas estructuradas con usuarios reales (estudiantes de ingeniería de software), lo cual permitió obtener retroalimentación valiosa sobre la funcionalidad e interacción con el sistema.

En términos de contribución, este estudio propone una solución práctica que no solo incrementa la seguridad en aplicaciones web, sino que también promueve la adopción de métodos adecuados durante las etapas del proceso de construcción de sistemas informáticos. Los resultados obtenidos evidencian el potencial de SecureSQLTester como una alternativa efectiva y de fácil implementación, especialmente en entornos de desarrollo de pequeña y mediana escala. De esta forma, se espera cerrar las brechas actuales en la protección frente a ataques de inyección SQL, y fortalecer la confianza de usuarios y desarrolladores en las plataformas digitales.

La estructura del presente documento es la siguiente: en la Sección II se expone el análisis sistemático de literatura; la Sección III aborda la metodología utilizada para el desarrollo del software; en la Sección IV se detalla la solución propuesta, seguida por la Sección V que presenta la evaluación de la usabilidad; la Sección VI analiza los resultados obtenidos; y finalmente, la Sección VII expone las conclusiones del estudio.

II. Revisión sistemática de literatura

La revisión sistemática constituye una metodología esencial para localizar, seleccionar y evaluar de manera crítica los artículos más relevantes y confiables en el ámbito de estudio. Este proceso permite organizar de forma estructurada la información disponible en la literatura científica, y asegurar que solo los datos pertinentes sean incorporados al análisis. Asimismo, facilita la identificación de tendencias investigativas, brechas de conocimiento y áreas poco exploradas, lo cual proporciona una base sólida que respalda y enriquece el desarrollo de nuevos estudios [7].

El presente estudio enfoca su revisión sistemática en la exploración de la siguiente interrogante investigativa: ¿Cómo se pueden desarrollar y aplicar de manera efectiva técnicas de seguridad que prevengan y mitiguen los riesgos asociados a ataques de inyección SQL dentro de plataformas web? Para ello, el proceso comenzó con la definición de palabras clave relevantes y la construcción de una cadena de búsqueda diseñada para maximizar la cobertura de artículos pertinentes. La cadena de búsqueda aplicada fue la siguiente:

("SQL injection" OR "SQL security" OR "SQL vulnerabilities" OR "SQL injection prevention" OR "web application security") AND ("tools" OR "software" OR "solutions" OR "secure coding practices").

Se llevó a cabo una consulta bibliográfica en repositorios científicos ampliamente reconocidos, tales como: IEEE Xplore, SpringerLink y ACM Digital Library. Además, se establecieron criterios de inclusión y exclusión para determinar qué estudios serían considerados o descartados, los cuales se detallan en la Tabla I. Estos criterios garantizaron la selección rigurosa de los estudios más relevantes para el análisis y el desarrollo posterior del prototipo propuesto.

TABLE I. CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN

Tipo de criterio	Descripción
Criterios de inclusión	Artículos publicados entre los años 2010 y 2024. El artículo debe reportar el uso de herramientas tecnológicas para mitigar vulnerabilidades de inyección SQL en aplicaciones web. El estudio debe enfocarse en técnicas y prácticas de seguridad orientadas a mejorar la protección contra ataques de inyección SQL.
Criterios de exclusión	Artículos que no aborden explícitamente vulnerabilidades de inyección SQL. Estudios que no mencionen herramientas ni prácticas relacionadas con la seguridad de aplicaciones web frente a este tipo de ataques.

En la Tabla II, se sintetiza la cantidad de estudios encontrados, seleccionados y considerados, como resultado de la aplicación del conjunto de términos de búsqueda en diversas fuentes bibliográficas digitales.

TABLE II. RESULTADOS DE LA BÚSQUEDA SISTEMÁTICA POR BASE DE DATOS

Base de Datos	Encontrados	Preseleccionados	Estudios Considerados
IEEE Xplore	20	6	3
SpringerLink	8	3	1
ACM Digital Library	9	5	1
Total	**85**	**41**	**5**

Del examen realizado a los trabajos previamente escogidos, se extrajeron diversas perspectivas relevantes sobre la problemática de las inyecciones SQL y las herramientas desarrolladas para su mitigación. A continuación, se presentan los principales hallazgos que sustentan esta investigación:

Faisal Fadlalla et al. [8] destacan que las inyecciones SQL continúan siendo una de las vulnerabilidades más frecuentes y

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

peligrosas en aplicaciones web, especialmente por la falta de validación adecuada de entradas de usuario. El autor enfatiza que muchas aplicaciones permanecen expuestas por la ausencia de buenas prácticas de codificación segura y la falta de implementación de herramientas específicas destinadas a salvaguardar los entornos donde se almacenan los datos. En este sentido, subraya la necesidad del uso de herramientas como SecureSQLTester para realizar auditorías de seguridad efectivas y fortalecer las defensas de las aplicaciones.

Por su parte, B. Kalaiselvi et al. [9] describen cómo herramientas de escaneo de seguridad, como SecureSQLTester, facilitan la identificación automática de vulnerabilidades de inyección SQL mediante la simulación de ataques controlados y la generación de informes detallados con recomendaciones de mitigación. Esta automatización no solo incrementa la eficiencia de las pruebas de seguridad, sino que también democratiza el acceso a prácticas de protección avanzada para desarrolladores con conocimientos limitados.

De manera complementaria, Saeed et al. [10] analizan los beneficios de integrar herramientas de detección de vulnerabilidades durante todas las fases involucradas en la construcción de sistemas informáticos. Según su investigación, la implementación de pruebas continuas de inyección SQL permite identificar y corregir fallas de seguridad antes de que las aplicaciones sean desplegadas en producción, reduciendo de manera significativa el riesgo de explotación de vulnerabilidades.

En la misma línea, Gupta et al. [11] proponen un enfoque combinado de cifrado mediante AES y criptografía de curva elíptica (ECC) para mitigar los ataques por inyección SQL, una de las amenazas más críticas a la protección de los entornos de almacenamiento de datos dentro de sistemas web. Su solución busca proteger tanto el acceso no autorizado durante el inicio de sesión como la información contenida en los repositorios de datos. En este contexto, y con un enfoque más accesible para usuarios no especializados, surge SecureSQLTester como una herramienta que facilita la detección temprana de vulnerabilidades sin requerir conocimientos avanzados en ciberseguridad.

Damaševičius et al. [12] sostienen que la seguridad en aplicaciones web debe ser abordada como un proceso continuo. En este marco, SecureSQLTester se presenta como una herramienta clave para realizar pruebas de penetración automatizadas, lo que incrementa la cobertura y profundidad de las evaluaciones de seguridad, y permite la detección temprana de puntos débiles explotables.

Durante este proceso de revisión de literatura, se identificaron diferencias importantes entre una diversidad de herramientas empleadas para mitigar vulnerabilidades de inyección SQL, entre las cuales se distinguen soluciones de código abierto **y** herramientas propietarias**.** Por ejemplo, SQLMap es una herramienta de código abierto ampliamente utilizada por su facilidad de acceso, flexibilidad y ausencia de costo de licencia. En contraste, herramientas como QualysGuard **y** Nessus son plataformas comerciales**,** con mayores requisitos técnicos y económicos, lo que restringe su adopción en contextos con limitaciones presupuestarias o técnicos. Esta diferenciación fue considerada al momento de evaluar la aplicabilidad de las soluciones revisadas en función del público objetivo del presente estudio [6] .

En conjunto, los estudios revisados evidencian que SecureSQLTester representa una solución eficaz para identificar y reducir los efectos de ataques por inyección SQL en entornos web. Su uso en etapas tempranas del ciclo de desarrollo permite fortalecer de manera significativa la seguridad general de los sistemas, lo cual reduce la probabilidad de explotación de vulnerabilidades críticas.

Si bien la revisión de literatura se centró en técnicas de detección y mitigación de inyecciones SQL debido a su alta prevalencia y criticidad en aplicaciones web, se reconoce que estas no representan el único vector de ataque relevante. De acuerdo con el estándar OWASP TOP 10, existen otras vulnerabilidades críticas que deben ser consideradas, como los ataques de tipo Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), y la exposición de datos sensibles, entre otros. En consecuencia, se proyecta como línea futura de trabajo la ampliación del alcance del prototipo SecureSQLTester para incorporar pruebas automatizadas que aborden este conjunto más amplio de amenazas.

III. METODOLOGÍA PARA LA CONSTRUCCIÓN DEL SOFTWARE

Con el objetivo de llevar a cabo esta propuesta, se adoptó la metodología ágil Scrum, considerada una estrategia idónea para gestionar proyectos complejos y dinámicos mediante ciclos iterativos e incrementales (ver figura 1). Scrum facilita la colaboración continua entre los miembros del equipo, permite una adaptación rápida a los cambios, y promueve la entrega constante de valor, y garantiza que el producto final cumpla con los requisitos planteados de manera eficiente [13].

En cada iteración, se consideró el principio de escalabilidad como parte integral del enfoque incremental, con el fin de garantizar que las funcionalidades desarrolladas puedan adaptarse progresivamente a diferentes entornos y cargas de trabajo. Esta orientación permite que el prototipo evolucione no solo en funcionalidad, sino también en robustez y capacidad de adaptación a proyectos de mayor escala.

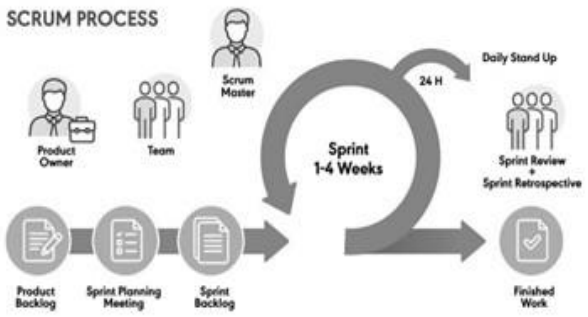

Fig. 1. Procesos Scrum

Fuente: Adaptada de [14]

A. Definición de Requisitos

La definición de requisitos constituyó una etapa fundamental, orientada a identificar los aspectos clave necesarios para mitigar ataques de inyección SQL mediante el desarrollo de un prototipo funcional y seguro. Este proceso incluyó una revisión exhaustiva de literatura especializada y la evaluación crítica de herramientas tecnológicas existentes, lo que permitió precisar las funcionalidades esenciales y las características deseadas del sistema.

a) Recolección de Requisitos: Los requisitos fueron clasificados en dos categorías principales: requisitos

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

funcionales y requisitos no funcionales. Esta estructuración permitió asegurar que el prototipo cumpliera con las necesidades operativas de detección y prevención de vulnerabilidades SQL, además de garantizar su desempeño, escalabilidad, facilidad de uso y seguridad.

b) Documentación de Requisitos: En la Tabla III, se presentan los requisitos funcionales, los cuales describen las capacidades específicas que debe ofrecer el sistema para cumplir con su propósito de detección y mitigación de vulnerabilidades de inyección SQL. Por su parte, la Tabla IV muestra los requisitos no funcionales, orientados a definir las características de calidad que aseguren el desempeño, la seguridad y la usabilidad del prototipo.

TABLE III. REQUISITOS FUNCIONALES DEL SISTEMA

Requerimiento	Descripción
Pruebas de Inyección SQL	Permitir la ejecución de pruebas para detectar vulnerabilidades de inyección SQL en aplicaciones que interactúan con bases de datos SQL.
Pruebas de Autenticación	Verificar que los métodos de autenticación y credenciales de acceso estén protegidos frente a ataques de fuerza bruta o de diccionario.
Generación de Reportes de Seguridad	Proporcionar informes detallados sobre las pruebas realizadas, indicando vulnerabilidades detectadas, su gravedad y sugerencias de mitigación.
Simulación de Ataques	Simular distintos tipos de ataques, como inyección SQL y Cross-Site Scripting (XSS), para evaluar la resistencia del sistema.
Pruebas de Encriptación de Datos	Verificar la correcta implementación de la encriptación de información confidencial mientras se transfiere y cuando se encuentra almacenada.
Validación de Permisos y Roles	Comprobar que los permisos de acceso y roles de usuario estén correctamente configurados para evitar accesos no autorizados.
Análisis de Estructuras de Bases de Datos	Ejecutar un análisis sobre bases de datos SQL para detectar configuraciones vulnerables, tales como claves poco robustas o accesos sin restricciones.
Pruebas de Vulnerabilidad de Contraseñas	Verificar la fortaleza de las contraseñas en bases de datos, asegurando la adhesión a estándares definidos para contraseñas robustas.
Acceso al almacenamiento de información de Forma Segura	Proveer métodos seguros de conexión a las bases de datos durante las pruebas, evitando la exposición de credenciales y datos sensibles.

B. Planificación del Proyecto

El desarrollo del proyecto se organizó siguiendo los principios de la metodología ágil Scrum, que permite gestionar proyectos complejos mediante entregas iterativas y la priorización continua de tareas. Inicialmente, se elaboró un Product Backlog que recopiló todas las funcionalidades y actividades necesarias para la construcción del prototipo SecureSQLTester. Cada ítem del backlog fue priorizado de acuerdo con su importancia, con el fin de alcanzar las metas definidas en la propuesta. Posteriormente, se estableció una planificación basada en sprints de dos semanas de duración. Al inicio de cada sprint, se seleccionaron las tareas de mayor prioridad, y se establecieron metas claras y alcanzables para garantizar avances consistentes y enfocados. Esta estrategia permitió una adaptación ágil a los requerimientos emergentes y facilitó la entrega incremental de valor a lo largo del proyecto En la Tabla V, se incluye un detalle completo de los sprints llevados a cabo, junto con las principales tareas realizadas y los objetivos definidos en cada etapa.

TABLE IV. REQUISITOS NO FUNCIONALES DEL SISTEMA

Requerimiento	Descripción
Interfaz de Usuario Intuitiva	El sistema debe contar con una interfaz amigable que facilite la configuración y ejecución de pruebas a usuarios sin conocimientos avanzados en seguridad.
Diseño Responsivo	La interfaz debe adaptarse automáticamente a diferentes dispositivos (computadoras, tabletas, teléfonos móviles).
Alta Velocidad en la Ejecución de Pruebas	Las pruebas de seguridad deben ejecutarse de forma rápida y eficiente, reduciendo al mínimo las afectaciones sobre el desempeño de la base de datos analizada.
Compatibilidad Multibase de Datos	El sistema debe soportar bases de datos SQL populares como MySQL, PostgreSQL y SQL Server.
Escalabilidad	El prototipo debe manejar bases de datos de diversos tamaños sin afectar la precisión ni el rendimiento de las pruebas
Confiabilidad	Las pruebas deben ofrecer resultados consistentes, minimizando falsos positivos y falsos negativos.
Seguridad de la Aplicación	El sistema debe asegurar la confidencialidad de los datos utilizados durante las pruebas y preservar la fidelidad de los resultados generados.
Soporte Multilingüe	Debe ofrecer soporte para múltiples idiomas, facilitando su uso en diferentes contextos geográficos.
Integración con Herramientas de CI/CD	Debe ser compatible con plataformas de integración y entrega continua (CI/CD), permitiendo la automatización de las pruebas de seguridad.

TABLE V. PLANIFICACIÓN DE SPRINTS

Sprint	Actividad Principal	Tiempo Semana	Tareas	Objetivos
1	Diseño de la Interfaz de Usuario	1-2	Dibujar bocetos y maquetas de la interfaz, incluyendo la pantalla principal, pruebas y área de resultados.	Desarrollar un diseño intuitivo y amigable que facilite la interacción y navegación de los usuarios en el sistema.
2	Gestión de Usuarios y Autenticación	3-4	Crear el registro de usuarios. Implementar autenticación segura mediante contraseñas.	Restringir el acceso a configuraciones avanzadas exclusivamente a los administradores
3	Implementación de Pruebas de Seguridad Básicas	5-6	Desarrollar pruebas de inyección SQL y validar vulnerabilidades en bases de datos.	Validar que el sistema identifique correctamente vulnerabilidades de inyección SQL.
4	Implementación de Reportes y Notificaciones	7-8	Desarrollar funcionalidades como reportes detallados de pruebas de seguridad.	Proporcionar al usuario reportes claros y útiles para la revisión de los resultados de las pruebas.
5	Integración Final y Pruebas de Usabilidad	9-10	Integrar todos los módulos del sistema y asegurar el correcto funcionamiento de sus funcionalidades	
Final	Evaluación y feedback	11-12	Entregar el prototipo final. Presentar el sistema funcional a los stakeholders para su evaluación.	Presentar la versión final del sistema y obtener feedback para comprobar si se alcanzaron las metas

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

IV. Evaluación de Usabilidad del Prototipo SecureSQLTester

Con el objetivo de validar la experiencia del usuario y detectar oportunidades de mejora en el prototipo desarrollado con el fin de identificar fallas de seguridad en sistemas web, se propuso la realización de una evaluación de usabilidad basada en la técnica entrevista estructurada.

La aplicación de esta técnica permitió recopilar información sistemática y consistente sobre la interacción de los usuarios con el prototipo, así como identificar necesidades, problemas y percepciones relevantes que orientaron las recomendaciones de mejora.

A. Descripción de la Técnica de Usabilidad: Entrevista Estructurada

La entrevista estructurada es un método cualitativo en el que se plantean preguntas predeterminadas a un grupo de participantes, con el objetivo de recolectar información detallada sobre su interacción y experiencia con un sistema [15]. Su principal fortaleza reside en el grado de dirección que mantiene el investigador durante el desarrollo de la conversación, lo que permite asegurar la obtención de datos organizados y comparables.

En este enfoque, las preguntas deben ser diseñadas con anticipación, de forma clara, precisa y alineadas con los aspectos de usabilidad que se desean evaluar [15]. De esta manera, se garantiza que las respuestas obtenidas estén directamente relacionadas con los objetivos de la evaluación.

La técnica aplicada en este estudio siguió el procedimiento clásico propuesto por Hix y Hartson [16], el cual contempla cinco pasos esenciales, que se detallan en la Tabla VI.

TABLE VI. Pasos y tareas de la técnica entrevista estructurada según Hix y Hartson [16]

N°	Nombre del Paso	Tareas
1	Definición de los objetivos	Identificar el objetivo principal de aplicar la técnica, orientado a mejorar la usabilidad del proyecto.
2	Identificación del público objetivo	Seleccionar a los participantes adecuados para la evaluación de la herramienta.
3	Diseño de las preguntas	Formular preguntas claras, concisas y directamente relacionadas con los objetivos de evaluación.
4	Validación de las preguntas	Solicitar la revisión de las preguntas por parte de otros usuarios o expertos para garantizar su claridad y relevancia.
5	Análisis de los datos	Emplear métodos adecuados de procesamiento de información con el propósito de comprender los hallazgos y derivar interpretaciones relevantes.

B. Adaptaciones de la Técnica de Usabilidad: Entrevista Estructurada

Si bien la técnica entrevista estructurada es ampliamente utilizada en evaluaciones de usabilidad y no representa mayores complicaciones operativas [17], su correcta aplicación tradicional requiere la participación de un experto en usabilidad que posea conocimientos técnicos y habilidades blandas para conducir entrevistas efectivas [18]. Dado el contexto particular del proyecto, fue necesario realizar adaptaciones metodológicas para facilitar su implementación.

En la Tabla VII, se presentan las principales condiciones adversas identificadas y las adaptaciones propuestas para asegurar la ejecución efectiva de la técnica.

TABLE VII. Condiciones adversas y adaptaciones propuestas de la técnica entrevista estructurada

N°	Nombre del Paso	Condiciones Adversas	Adaptaciones Propuestas
1	Definición de los objetivos	Se requiere de un experto en usabilidad.	Sustituido por estudiantes de la UTEQ bajo la supervisión de un mentor.
2	Identificación del público objetivo	Se requiere la participación presencial de usuarios.	Participación remota a través de foros, correo electrónico y comentarios en blogs.
3	Diseño de las preguntas	Se requiere de un experto para formular las preguntas.	Diseño de preguntas por estudiantes supervisados por un mentor académico.
4	Validación de las preguntas	Se requiere de un experto para validar el instrumento.	Validación interna entre estudiantes y mentores.
5	Análisis de los datos	Se requiere de un experto en análisis de usabilidad.	Análisis efectuado por estudiantes bajo supervisión académica, utilizando agrupación temática de respuestas.

Debido a las restricciones operativas mencionadas, se definieron y aplicaron una serie de pasos adicionales para facilitar la ejecución de la técnica entrevista estructurada adaptada:

a) Paso 1: Realización de una prueba piloto. Se llevó a cabo una prueba preliminar con la finalidad de examinar la funcionalidad general del proceso de entrevista estructurada y validar los insumos iniciales diseñados.

b) Paso 2: Uso herramientas de comunicación y colaboración. Considerando la imposibilidad de encuentros presenciales, se establecieron canales de comunicación remota mediante Google Meet, Gmail y WhatsApp, lo que garantizó la participación activa de los usuarios durante todo el proceso de evaluación.

c) Paso 3: Adaptación del formato de las entrevistas. Se diseñó un conjunto de preguntas orientadas a los factores específicos de usabilidad que se deseaba evaluar, para asegurar su claridad y pertinencia mediante revisión y validación preliminar por parte del mentor académico.

d) Paso 4: Elaboración de plantillas y documentos de apoyo. Las preguntas validadas fueron organizadas en un formulario de Word y complementadas con materiales de apoyo, tales como la guía de instalación del sistema, documentos de consentimiento informado, tareas específicas para los usuarios y plantillas de recolección de respuestas.

e) Paso 5: Ejecución de la evaluación de usabilidad. La evaluación fue realizada en modalidad remota, siguiendo un cronograma predefinido que incluyó comunicaciones

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

formales a los participantes, asignación de tareas, envío de insumos y sesiones de asistencia técnica virtual.

f) Paso 6: Análisis de datos y consolidación de resultados. Las respuestas obtenidas fueron sistematizadas y agrupadas temáticamente para eliminar redundancias. Posteriormente, se documentaron en un informe de entrevista estructurada que sirvió como base para identificar áreas de mejora del prototipo.

Estos pasos de adaptación se resumen en la Tabla VIII.

TABLE VIII. PASOS Y TAREAS DE LA TÉCNICA ADAPTADA ENTREVISTA ESTRUCTURADA

N°	Nombre del Paso	Tarea
1	Realizar una prueba piloto	Validar previamente los insumos y procedimientos de la evaluación.
2	Utilizar herramientas de comunicación y colaboración	Definir el perfil de usuario, enviar correos de invitación y reclutar participantes mediante redes sociales.
3	Adaptar el formato de las entrevistas	Diseñar preguntas claras y específicas relacionadas con los aspectos de usabilidad a evaluar.
4	Diseñar plantillas y herramientas	Crear formularios en Google Forms o documentos de Word para la recopilación de respuestas.
5	Realizar la evaluación de usabilidad	Ejecutar la evaluación mediante reuniones remotas, asegurando el acceso a todos los insumos necesarios.
6	Analizar los datos y agrupar resultados	Sistematizar y consolidar los resultados obtenidos, socializar el análisis con los participantes para retroalimentación.

V. RESULTADOS

Esta sección presenta los principales resultados obtenidos durante el desarrollo y la evaluación del prototipo SecureSQLTester, los cuales abarcaron las actividades de análisis, investigación, comunicación con usuarios, aplicación de técnicas de usabilidad, observaciones recogidas y el diseño final de las interfaces.

A. Evaluación de Usabilidad del Prototipo SecureSQLTester

La evaluación de usabilidad se llevó a cabo mediante la técnica entrevista estructurada, dirigida a estudiantes de nivel inicial en el programa académico de Ingeniería de Software, quienes representan un perfil cercano al público objetivo del prototipo. La planificación de la evaluación incluyó la elaboración de un comunicado formal invitando a los participantes, enviado mediante correo electrónico.

Una vez recibida la invitación, los usuarios pudieron iniciar sesión en el prototipo, e ingresaron su nombre, personalizaron así su experiencia de uso (ver Figura 2).

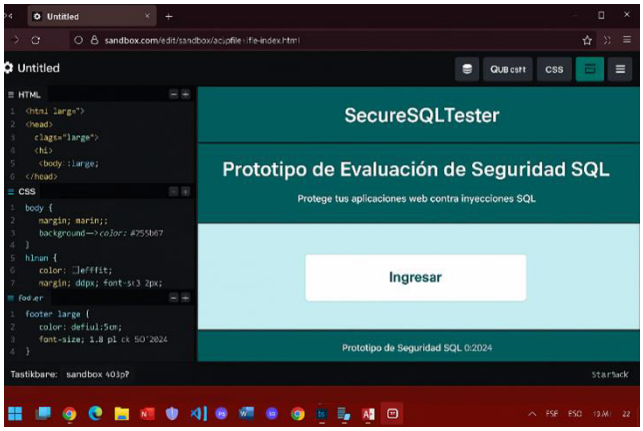

Fig. 2. Pantalla de inicio de sesión del prototipo

Posteriormente, se presentó un módulo explicativo al que los participantes pudieron acceder para consultar recomendaciones generales, advertencias sobre inyecciones SQL y pautas sobre el uso correcto del sistema (ver Figura 3).

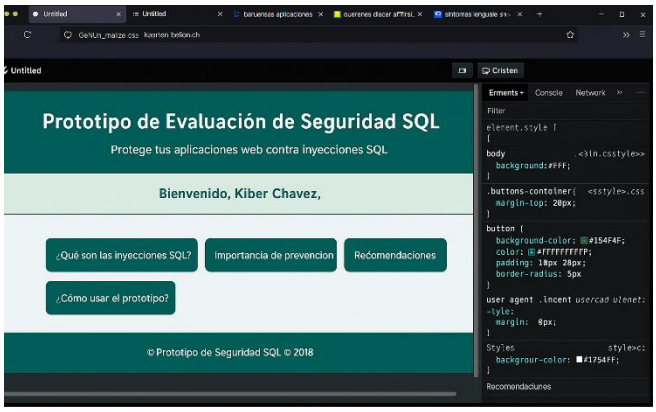

Fig. 3. Parámetros de información mostrados a los usuarios

En el módulo principal, los usuarios ingresaron diversas URLs para verificar la presencia de posibles inyecciones SQL. En caso de detección de amenazas, el sistema mostraba una advertencia; en caso contrario, confirmaba la seguridad de la URL analizada. Los resultados obtenidos eran almacenados en un historial accesible para consulta posterior (ver Figura 4).

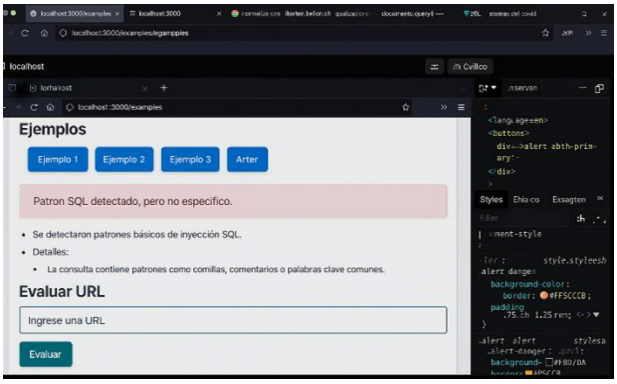

Fig. 4. Módulo de ingreso de URLs y visualización de resultados

Durante la evaluación, se asignaron diversas tareas diseñadas para simular escenarios prácticos de uso, lo que

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

permitió evaluar la funcionalidad y facilidad de interacción del prototipo (ver Figuras 5 y 6).

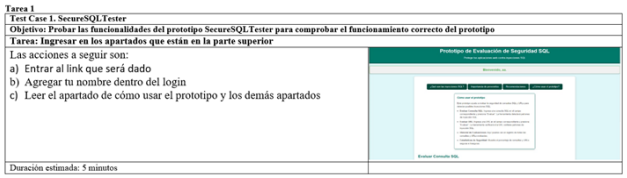

Fig. 5. Formato para la tarea 1 de interacción con el sistema

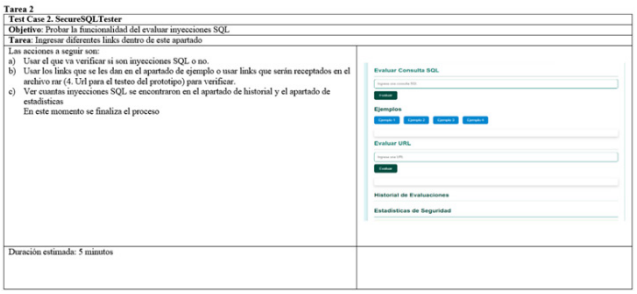

Fig. 6. Formato para la tarea 2 de interacción con el sistema

Además, se utilizó un documento específico para la recolección sistemática de las respuestas de los usuarios tras la ejecución de las tareas (ver Figura 7).

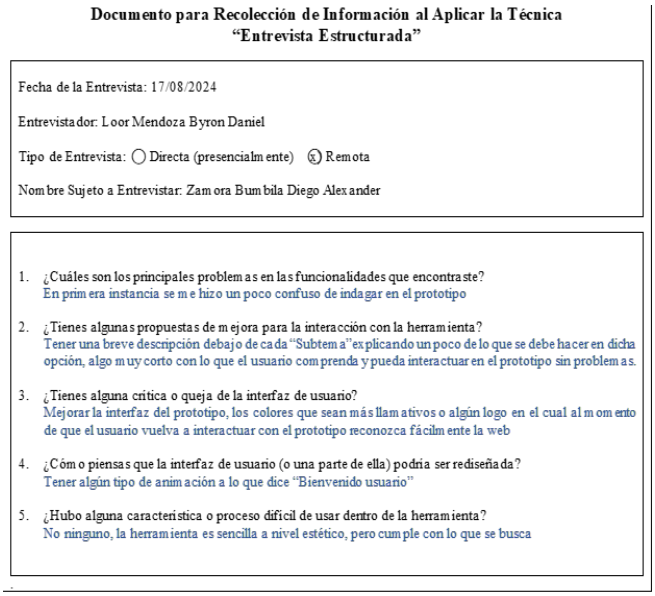

Fig. 7. Plantilla de recolección de datos de la evaluación de usabilidad

Posterior a la aplicación de la técnica, se recopilaron y analizaron las respuestas de los cuatro participantes que completaron la prueba, identificándose problemas, observaciones críticas y propuestas de mejora. Los resultados fueron organizados y sistematizados para facilitar su interpretación y orientar futuros ajustes al prototipo.

B. Análisis de Respuestas por Pregunta

Antes de presentar el análisis individual, se señala que algunas respuestas no se alinearon estrictamente con las preguntas planteadas; por ello, fueron reubicadas de acuerdo con su mayor pertinencia temática.

La Tabla IX resume las respuestas obtenidas de los usuarios durante la aplicación de la técnica entrevista estructurada.

TABLE IX. ANÁLISIS DE RESPUESTAS A LAS PREGUNTAS DE LA EVALUACIÓN DE USABILIDAD

Pregunta	Resumen de respuestas de los usuarios
¿Cuáles son los principales problemas en las funcionalidades que encontraste?	Se identificó falta de claridad en el funcionamiento de algunos parámetros, lo cual dificultaba su comprensión. Se sugiere una mejor especificación de cada función.
¿Tienes algunas propuestas de mejora para la interacción con la herramienta?	Los usuarios propusieron hacer más detallada la información en cada apartado, para facilitar la lectura y entendimiento de las funcionalidades.
¿Tienes alguna crítica o queja de la interfaz de usuario?	Se criticó que la interfaz era muy simple y no resultaba visualmente atractiva, recomendándose hacerla más llamativa para mejorar la experiencia del usuario.
¿Cómo piensas que la interfaz de usuario (o una parte de ella) podría ser rediseñada?	Se sugirió agregar más colores, incorporar animaciones en el saludo inicial, y eliminar elementos de código visible que podrían representar vulnerabilidades.
¿Hubo alguna característica o proceso difícil de usar dentro de la herramienta?	En general, los usuarios no reportaron dificultades significativas; consideraron que el prototipo era fácil de usar gracias a las instrucciones previas proporcionadas.

C. Análisis de Observaciones Dadas por el Usuario

El análisis de las observaciones y sugerencias emitidas por los usuarios entrevistados proporciona una visión integral de las áreas del prototipo SecureSQLTester que requieren atención y mejora. Esta retroalimentación es fundamental para orientar las futuras iteraciones de desarrollo, lo cual permite priorizar cambios basados en las necesidades reales y percepciones de los usuarios finales. La Tabla X incluye una síntesis de los hallazgos más relevantes, clasificados por categoría, tipo de observación y frecuencia de aparición entre los participantes.

TABLE X. OBSERVACIONES RECOPILADAS MEDIANTE ENTREVISTAS, CLASIFICADAS POR CATEGORÍA, HALLAZGO Y FRECUENCIA

Categoría	Hallazgo	Frecuencia
Problemas /Errores	Dificultades para autenticar al usuario mediante credenciales válidas	3
	Lentitud al realizar pruebas de inyección SQL en bases de datos grandes	2
Mejoras de Interfaz	Mejorar la visualización de los resultados de pruebas	2
	Añadir opciones de personalización en el diseño de la interfaz	1
	Incluir más detalles sobre los errores encontrados en las pruebas	2
	Mejorar la organización de las opciones del menú para facilitar el acceso	1
Nuevas Funcionalidades	Agregar opción de informes automáticos al finalizar las pruebas	1
	Implementar función para probar vulnerabilidades en bbdd no relacionales	2
	Añadir la capacidad de realizar pruebas de seguridad programadas	1
Usabilidad	Mejorar la documentación interna para usuarios novatos	2
	Simplificar los pasos para iniciar las pruebas de seguridad	1
	Agregar opción para guardar configuraciones de pruebas	1

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

La información obtenida en esta etapa resulta crucial para priorizar las mejoras, optimizar la experiencia de usuario, y ampliar las funcionalidades de seguridad en futuras versiones del sistema.

D. Diseño de Interfaces Resultantes

Como parte de la mejora continua del prototipo, se diseñaron y ajustaron interfaces que optimizan la interacción de los usuarios con el sistema, basadas en las observaciones recopiladas durante la fase de evaluación de usabilidad.

La Figura 8 muestra la interfaz renovada de autenticación de usuario, donde se solicita al usuario ingresar su nombre para acceder al sistema de manera personalizada y amigable.

Fig. 8. Interfaz de inicio de sesión del prototipo SecureSQLTester

Posteriormente, la Figura 9 presenta la interfaz principal del prototipo, diseñada para facilitar la evaluación de vulnerabilidades por inyección SQL en enlaces ingresados por los usuarios. Esta pantalla permite gestionar de manera intuitiva las pruebas realizadas y consultar los resultados obtenidos.

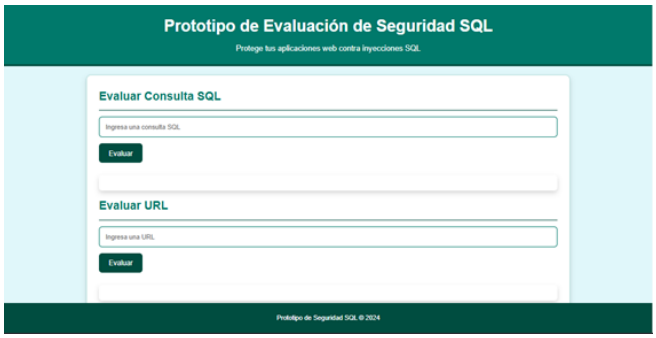

Fig. 9. Interfaz principal del prototipo SecureSQLTester

VI. Discusión de Resultados

Los hallazgos obtenidos durante el proceso de evaluación indican que el prototipo SecureSQLTester ha cumplido de manera satisfactoria con los objetivos planteados, destacándose su relevancia en el fortalecimiento de la ciberseguridad para aplicaciones web. La integración de técnicas efectivas para la detección y mitigación de vulnerabilidades de inyección SQL permitió obtener resultados confiables en la identificación de amenazas críticas, con un desempeño adecuado en la mayoría de los escenarios evaluados.

Una de las principales fortalezas identificadas fue la capacidad del prototipo para analizar y procesar solicitudes en tiempo real, lo que demuestra su aplicabilidad en contextos que requieren respuestas inmediatas, tales como auditorías de bases de datos y pruebas de penetración. Estos resultados son coherentes con investigaciones previas que subrayan la importancia de contar con herramientas específicas y rápidas para mitigar amenazas de seguridad en aplicaciones web.

Los usuarios participantes valoraron de forma favorable la usabilidad del sistema y la comprensión de los reportes generados sobre las vulnerabilidades detectadas, y destacaron su utilidad como una herramienta accesible y práctica para desarrolladores y pequeñas empresas, especialmente en proyectos de mediana o baja escala presupuestaria. Este aspecto resalta la contribución de SecureSQLTester en el ámbito de soluciones de ciberseguridad accesibles para públicos con experiencia limitada en seguridad informática.

En comparación con herramientas comerciales más robustas, como QualysGuard Web Application Scanning y Nessus Vulnerability Scanner, SecureSQLTester destacó principalmente por su simplicidad de instalación y configuración [6]. Mientras que las herramientas mencionadas ofrecen una cobertura más exhaustiva de vulnerabilidades, su implementación suele requerir mayores recursos y conocimientos técnicos especializados. En contraste, SecureSQLTester fue valorado por su enfoque intuitivo y accesible, lo que resulta especialmente útil para usuarios con conocimientos básicos en ciberseguridad y manejo de consultas SQL.

No obstante, también se identificaron áreas de mejora, particularmente en lo que respecta a la interfaz de usuario. Aunque el sistema demostró una funcionalidad eficiente, algunos usuarios señalaron que el diseño resultaba excesivamente simple y carecía de elementos visuales atractivos. Esta observación coincide con estudios previos que enfatizan la importancia de la usabilidad y el diseño intuitivo en el éxito de herramientas tecnológicas. Se sugirió mejorar la organización visual de los componentes en la interfaz y aumentar las opciones de personalización de los parámetros de entrada, de manera que el prototipo pueda adaptarse mejor a diferentes configuraciones de bases de datos y perfiles de usuarios.

Además, es importante señalar una limitación metodológica relevante identificada durante la evaluación de usabilidad es la homogeneidad del grupo de participantes, compuesto exclusivamente por estudiantes de Ingeniería de Software. Si bien este perfil resulta adecuado para una primera aproximación técnica, no representa de forma amplia a la población objetivo del prototipo, que puede incluir desarrolladores con distintos niveles de experiencia, profesionales de áreas no técnicas y personal administrativo. Esta limitación reduce la generalización de los hallazgos obtenidos y condiciona la interpretación de la percepción de usabilidad.

En general, el prototipo ha demostrado ser una solución viable y práctica para mejorar la seguridad de aplicaciones web, facilitando su adopción temprana y contribuyendo a la reducción de riesgos asociados a vulnerabilidades de inyección SQL. Se estima que, con futuras optimizaciones en la personalización de configuraciones y en el diseño de la interfaz, SecureSQLTester podrá ampliar su competitividad en el mercado de herramientas de ciberseguridad, e incrementar su capacidad de respuesta a una gama más diversa de necesidades de especialistas del campo de la ciberseguridad.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025
DOI: 10.5281/zenodo.15740582

VII. Conclusiones

El desarrollo del prototipo SecureSQLTester ha demostrado ser una solución eficaz orientada a identificar y reducir los efectos de ataques por inyección SQL en entornos web, e impulsar notablemente la mejora de la protección y consistencia de los sistemas informáticos. La implementación de medidas automatizadas para la identificación de inyecciones SQL ha permitido minimizar los riesgos asociados a ataques maliciosos, favorecer la protección de datos sensibles y asegurar la estabilidad de las aplicaciones evaluadas.

Además, la facilidad de uso y accesibilidad del prototipo posicionan a SecureSQLTester como una alternativa viable para desarrolladores independientes y pequeñas empresas que buscan adoptar buenas prácticas de ciberseguridad sin incurrir en grandes inversiones tecnológicas. Durante la evaluación de usabilidad, los usuarios valoraron positivamente la claridad de las funcionalidades y la efectividad en la detección de vulnerabilidades. No obstante, se identificaron áreas de mejora relacionadas con la estructura visual del entorno interactivo, que fue percibida como demasiado simple y carente de elementos visuales atractivos.

A partir de las observaciones recopiladas, se plantea como trabajo futuro el rediseño de la interfaz del prototipo, con la incorporación de elementos visuales más modernos, como paletas de colores, animaciones y mejoras en la organización de los elementos de navegación. Asimismo, se propone la inclusión de tutoriales interactivos y guías prácticas que faciliten la comprensión de las funcionalidades por parte de usuarios con conocimientos limitados en ciberseguridad.

Por otra parte, se prevé extender las capacidades del prototipo más allá de las inyecciones SQL, incorporando pruebas para detectar otros tipos de vulnerabilidades web descritas en el marco de referencia OWASP TOP 10**.** Esta ampliación permitirá realizar una evaluación de seguridad más integral, y potenciará la utilidad del sistema en entornos reales donde múltiples vectores de ataque pueden coexistir.

Igualmente, se reconoce que la evaluación de usabilidad realizada en esta fase se centró en un grupo restringido de usuarios técnicos. Como línea futura de trabajo, se plantea la realización de nuevas pruebas con una muestra más heterogénea de usuarios, entre los que incluyan desarrolladores profesionales, administradores de sistemas y usuarios sin formación técnica, lo cual permitirá validar la aplicabilidad del prototipo en contextos operativos reales y enriquecer el proceso de mejora continua de la herramienta.

Estas optimizaciones no solo mejorarán la experiencia de usuario, sino que también incrementarán la adopción y aceptación de la herramienta en una audiencia más amplia. Con estas mejoras, SecureSQLTester podrá consolidarse como una solución clave dentro del campo de la protección de plataformas web, lo que permitirá ampliar su aplicabilidad y aportar de manera efectiva a la reducción de riesgos de seguridad en entornos digitales diversos. Asimismo, como parte de las futuras líneas de investigación, se contempla la validación del rendimiento del prototipo en entornos reales, tales como sistemas en producción o laboratorios que simulen condiciones operativas complejas. Esta evaluación permitirá comprobar su eficacia frente a desafíos como la concurrencia de usuarios, variabilidad de la carga de trabajo y la presencia de condiciones de red inestables, y contribuir a una mejora sustancial en su confiabilidad y aplicabilidad.

En paralelo, se continuará fortaleciendo el diseño del prototipo para mejorar su escalabilidad, lo que permitirá que la herramienta sea utilizada eficientemente en proyectos de diferente complejidad y en empresas de distintas dimensiones, sin comprometer el rendimiento o la precisión de las pruebas de seguridad.

También, se plantea como mejora la integración del prototipo con CMS y frameworks web populares mediante el desarrollo de módulos, APIs o scripts embebidos, lo cual permitirá la automatización de pruebas de seguridad directamente desde el entorno de desarrollo del usuario. Esta funcionalidad facilitará el uso continuo de SecureSQLTester durante el ciclo de vida del software y su incorporación a procesos de CI/CD en entornos reales.

Finalmente, como parte de las mejoras funcionales planificadas, se contempla el desarrollo de un módulo de grabación de sesiones de usuario, inspirado en herramientas como Selenium. Esta funcionalidad permitirá capturar flujos reales de interacción, que incluyen la introducción de credenciales y navegación dentro de la aplicación, lo que hará posible identificar inyecciones de datos que solo se manifiestan en contextos específicos o secuencias dinámicas de uso.

Referencias

[1] E. A. Medellín Cabrera, “Un modelo de gestión de la transformación digital para la innovación,” *360: Revista de Ciencias de la Gestión*, Nov. 2024, doi: 10.18800/360gestion.202409.009.

[2] A. Hernández and J. Mejía, “Guía de ataques, vulnerabilidades, técnicas y herramientas para aplicaciones web,” *ReCIBE. Revista electrónica de Computación, Informática Biomédica y Electrónica*, vol. 4, no. 1, Feb. 2015.

[3] C. Añasco Loor, K. Morocho, and M. Hallo, “Using Data Mining Techniques for the Detection of SQL Injection Attacks on Database Systems,” *Revista Politécnica*, vol. 51, no. 2, pp. 19–28, May 2023, doi: 10.33333/rp.vol51n2.02.

[4] J. S. Monar Monar, D. M. Pastor Ramirez, G. de L. Arcos Medina, and M. A. Oñate Andino, “Técnicas de programación segura para mitigar vulnerabilidades en aplicaciones web,” *Congreso de Ciencia y Tecnología ESPE*, vol. 13, no. 1, Jun. 2018, doi: 10.24133/cctespe.v13i1.753.

[5] J. R. Tadhani, V. Vekariya, V. Sorathiya, S. Alshathri, and W. El-Shafai, “Securing web applications against XSS and SQLi attacks using a novel deep learning approach,” *Sci Rep*, vol. 14, no. 1, p. 1803, Jan. 2024, doi: 10.1038/s41598-023-48845-4.

[6] Geeta Sandeep Nadella, Hari Gonaygunta, Deepak Kumar, and Priyanka Pramod Pawar, “Exploring the impact of AI-driven solutions on cybersecurity adoption in small and medium enterprises,” *World Journal of Advanced Research and Reviews*, vol. 22, no. 1, pp. 1199–1197, Apr. 2024, doi: 10.30574/wjarr.2024.22.1.1185.

[7] J. P. Z. Proano and V. C. Párraga Villamar, “Systematic mapping study of literature on educational data mining to determine factors that affect school performance,” in *Proceedings - 3rd International Conference on Information Systems and Computer Science, INCISCOS 2018*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 239–245. doi: 10.1109/INCISCOS.2018.00042.

[8] F. Faisal Fadlalla and H. T. Elshoush, “Input Validation Vulnerabilities in Web Applications: Systematic Review, Classification, and Analysis of the Current State-of-the-Art,” *IEEE Access*, vol. 11, pp. 40128–40161, 2023, doi: 10.1109/ACCESS.2023.3266385.

[9] B. K. B. Kalaiselvi, M. S. Chandu, M. Narendra, and M. Deekshith Kumar, “SQL-Injection Vulnerability Scanning Tool for Automatic Creation of SQL-Injection Attacks,” *International Journal of Advances in Engineering and Management*, vol. 7, no. 1, pp. 577–587, Jan. 2025, doi: 10.35629/5252-0701577587.

[10] H. Saeed, I. Shafi, J. Ahmad, A. A. Khan, T. Khurshaid, and I. Ashraf, “Review of Techniques for Integrating Security in

Software Development Lifecycle," *Computers, Materials & Continua*, vol. 82, no. 1, pp. 139–172, 2025, doi: 10.32604/cmc.2024.057587.

[11] H. Gupta, S. Mondal, S. Ray, B. Giri, R. Majumdar, and V. P. Mishra, "Impact of SQL Injection in Database Security," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, IEEE, Dec. 2019, pp. 296–299. doi: 10.1109/ICCIKE47802.2019.9004430.

[12] R. Damaševičius and L. Zailskaitė-Jakštė, "Usability and Security Testing of Online Links: A Framework for Click-Through Rate Prediction Using Deep Learning," *Electronics (Basel)*, vol. 11, no. 3, p. 400, Jan. 2022, doi: 10.3390/electronics11030400.

[13] S. S. M. M. Rahman *et al.*, "OSCRUM: A Modified Scrum for Open Source Software Development," *International journal of simulation: systems, science & technology*, Jan. 2019, doi: 10.5013/IJSSST.a.19.03.20.

[14] G. Mendoza, "Qué Ejemplo Práctico Ilustra la Metodología Scrum en Proyectos," Pasaporte Mexicano. Accessed: Jun. 11, 2025. [Online]. Available: https://pasaporte-mexicano.com/que-ejemplo-practico-ilustra-la-metodologia-scrum-en-proyectos/?expand_article=1

[15] K. Dunwoodie, L. Macaulay, and A. Newman, "Qualitative interviewing in the field of work and organisational psychology: Benefits, challenges and guidelines for researchers and reviewers," *Applied Psychology*, vol. 72, no. 2, pp. 863–889, Apr. 2023, doi: 10.1111/apps.12414.

[16] D. Hix and R. Hartson, *Developing User Interfaces: Ensuring Usability Through Product and Process*, vol. 4, no. 1. Wiley, 1994. doi: 10.1002/STVR.4370040109.

[17] C. Lopezosa, "Entrevistas semiestructuradas con NVivo: pasos para un análisis cualitativo eficaz," in *Methodos Anuario de Métodos de Investigación en Comunicación Social, 1*, Universitat Pompeu Fabra, 2020, pp. 88–97. doi: 10.31009/methodos.2020.i01.08.

[18] J. W. Castro, "Incorporación de la Usabilidad en el Proceso de Desarrollo Opencas Source Software," Tesis Doctoral, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, España, 2014.

AUTHORS

Nancy Rodriguez

Nancy Rodríguez obtuvo su título de Máster en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones en la Universidad Autónoma de Madrid (España), donde actualmente cursa un Doctorado en Ingeniería Informática y de Telecomunicaciones. Cuenta con más de diez años de experiencia profesional en desarrollo de software y actualmente se desempeña como profesora en la Facultad de Ciencias de la Computación y Diseño Digital de la Universidad Técnica Estatal de Quevedo (UTEQ) en Ecuador. Ha impartido una variedad de asignaturas a nivel de pregrado y posgrado, particularmente en las áreas de programación, ingeniería de software, bases de datos y tecnologías web. Su trabajo académico incluye la participación en proyectos de investigación FOCICYT-UTEQ, enfocados en sistemas inteligentes, educación digital y tecnologías para el envejecimiento activo, orientadas a mejorar el bienestar de los adultos mayores. También ha sido ponente en conferencias nacionales e internacionales en el campo de la informática educativa y el aprendizaje mediado por tecnologías. Sus principales áreas de investigación incluyen los procesos de desarrollo de software, la usabilidad en sistemas de código abierto, los entornos de aprendizaje en línea, y los cursos en línea masivos y abiertos (MOOC).

Daniel Loor

Byron Daniel Loor Mendoza es un estudiante de Ingeniería de Software en la Universidad Técnica Estatal de Quevedo (UTEQ). Cuenta con la experiencia en el desarrollo de software, habiendo participado en diversos proyectos que abarcan distintos sectores. Su trabajo incluye contribuciones significativas en el área de físico-deportivos, sistemas de inventarios, sistemas contables y la seguridad en bases de datos.
A lo largo de su trayectoria académica y profesional, ha trabajado con metodologías de prototipado, lo que le permite visualizar y refinar rápidamente las soluciones, asegurando que se adapten a las necesidades del usuario. Ha desarrollado aplicaciones de escritorio, utilizando herramientas de vanguardia como Visual Studio 2022, lo que demuestra su dominio en entornos de desarrollo .NET. Su pasión se inclina fuertemente hacia el software de código abierto, creyendo en la colaboración y la transparencia para crear soluciones innovadoras y accesibles. Su principal enfoque es el desarrollo de aplicaciones de escritorio, siempre bajo la implementación de metodologías ágiles para asegurar entregas continuas y una adaptación eficiente a los cambios.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025

AUTHORS

Lucrecia Llerena

Lucrecia Llerena finalizó su Doctorado en Informática y Telecomunicaciones con mención CUM LAUDE, y obtuvo también el Máster Universitario en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones (I2TIC), ambos en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid (UAM). Además, cursó una Maestría en Educación a Distancia y Abierta, así como su título de Ingeniera en Sistemas, en la Universidad Autónoma de Los Andes (Ecuador). Actualmente se desempeña como profesora titular en la Facultad de Ciencias de la Computación y Diseños Digitales de la Universidad Técnica Estatal de Quevedo (UTEQ), donde labora desde el año 2001. Ha dirigido varios proyectos FOCICYT y tesis de pregrado y posgrado en las universidades UTEQ y UPSE. Sus líneas de investigación se centran en la ingeniería de software, los procesos de desarrollo, la integración de la usabilidad, los sistemas inteligentes y la educación en entornos e-learning.

N. Rodríguez, D. Loor, and L. Llerena,
"Evaluating and mitigating SQL injections in web applications: developing a prototype",
Latin-American Journal of Computing (LAJC), vol. 12, no. 2, 2025.

ISSN:1390-9266 e-ISSN:1390-9134 LAJC 2025