UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

CURSO DE ESPECIALIZACIÓN INGENIERÍA DE CALIDAD

ANÁLISIS E IMPLEMENTACIÓN DE PRUEBAS AL SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONVOCATORIAS Y REGISTRO DE PROYECTOS DE INVESTIGACIÓN PARA LA SECRETARÍA DE INVESTIGACIONES CIENTÍFICAS DE LA UNIVERSIDAD DE EL SALVADOR SEGÚN LA NORMA ISO/IEC 25010

PRESENTADO POR:

MARCELO JOSUÉ ALCÁNTARA ALAS

DAVID JOSÉ CASTRO CLÍMACO

OSCAR LEONARDO SERPAS MARTÍNEZ

PARA OPTAR POR AL TÍTULO DE:

INGENIERO DE SISTEMAS INFORMÁTICOS

CIUDAD UNIVERSITARIA, ENERO 2026

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSc. JUAN ROSA QUINTANILLA

SECRETARIO GENERAL:

LCDO. PEDRO ROSALÍO ESCOBAR CASTANEDA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

MSc. LUIS SALVADOR BARRERA MANCÍA

SECRETARIO:

ARQ. RAÚL ALEXANDER FABIÁN ORELLANA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

DIRECTOR:

ING. CÉSAR AUGUSTO GONZÁLEZ RODRIGUEZ

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO DE SISTEMAS INFORMÁTICOS

Título:

ANÁLISIS E IMPLEMENTACIÓN DE PRUEBAS AL SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONVOCATORIAS Y REGISTRO DE PROYECTOS DE INVESTIGACIÓN PARA LA SECRETARÍA DE INVESTIGACIONES CIENTÍFICAS DE LA UNIVERSIDAD DE EL SALVADOR SEGÚN LA NORMA ISO/IEC 25010

Presentado por:

MARCELO JOSUÉ ALCÁNTARA ALAS

DAVID JOSÉ CASTRO CLÍMACO

OSCAR LEONARDO SERPAS MARTÍNEZ

Trabajo de Graduación Aprobado por:

Docente Asesor(a):

INGA. SANDRA GUADALUPE ROMERO HERNANDEZ

SAN SALVADOR, ENERO 2026

Trabajo de Graduación Aprobado por:

Docente Asesor:

INGA. SANDRA GUADALUPE ROMERO HERNANDEZ

Índice de contenidos

I. Agradecimientos

Tengo la oportunidad de brindar el agradecimiento a todas las personas que fueron parte esencial para la culminación de mi carrera universitaria.

Primeramente, doy gracias a Dios por permitirme seguir adelante en mis estudios desde el comienzo hasta este momento con el que puedo dar por finalizada mi carrera universitaria. Este éxito también se los debes a mis padres que han estado apoyándome incondicionalmente desde el primer momento y permitiéndome seguir adelante. Agradezco de corazón el apoyo completo de mi madre Amalia Clímaco por siempre estar conmigo en todos los pasos desde pequeño hasta este momento apoyando y aportando toda la ayuda que necesitará en el hogar. También quiero agradecer a mi padre Juan Castro por estar siempre pendiente por brindarme todos los recursos necesarios para seguir estudiando, dedicando su tiempo y energías en brindarme las mejores oportunidades para poder estudiar y finalizar mis estudios universitarios. Agradezco también a mi hermano César Castro por apoyarme y brindarme su conocimiento y ayuda cuando lo he necesitado, también siendo un ejemplo de superación y dedicación.

Agradezco a mis compañeros de tesis que a pesar de las dificultades que hemos podido encontrar durante todo el año, logramos resolver de la mejor manera posible todos esos inconvenientes y llegar hasta esta etapa con el fin de finalizar nuestros estudios y ser una persona de bien en la sociedad salvadoreña.

También agradezco a la ingeniera Sandra Romero por brindarnos todo el conocimiento y el asesoramiento necesario para poder realizar este trabajo de la mejor manera posible, cumpliendo con los objetivos propuestos como equipo de trabajo durante la realización de la especialización.

De igual manera agradezco al ingeniero Saúl Rodas por brindarnos la oportunidad de trabajar con él y facilitarnos todos los insumos necesarios para la realización de este proyecto. Agradecemos la dedicación y el tiempo que nos ha dedicado para apoyarnos y brindarnos el conocimiento para aclarar dudas y realizar un trabajo de calidad y bienestar para la Universidad de El Salvador.

Gracias a cada una de las personas que fueron parte de mi proceso y formación para la culminación de mis estudios universitarios. Todos fueron una parte importante en mi viaje hasta este momento y probablemente no lo hubiese logrado sin el apoyo de cada uno de ustedes, muchas gracias y estaré siempre agradecido.

David José Castro Clímaco

En estas líneas deseo reconocer a quienes hicieron posible la culminación de mi formación universitaria. Este logro es el resultado del acompañamiento, la confianza y el esfuerzo compartido.

En primer lugar, agradezco a Dios, fuente de fortaleza y esperanza, por guiar mi camino, darme claridad en los momentos decisivos y sostenerme cuando las dificultades parecían superar mis fuerzas.

Mi gratitud más entrañable es para mi abuela, María del Tránsito Serpas. Su apoyo constante a lo largo de los años, sus consejos oportunos y su ejemplo de trabajo y perseverancia han sido el pilar que me animó a continuar. Cada avance en este proceso lleva su huella de cariño y dedicación.

Extiendo un agradecimiento sincero a la ingeniera Sandra Romero por su orientación académica y por la exigencia constructiva que impulsó la calidad de este trabajo. Su disposición para resolver dudas y encauzar ideas fue clave para transformar inquietudes en resultados.

De igual manera, reconozco al ingeniero Saúl Rodas por brindarme tiempo, recursos y retroalimentación precisa durante el desarrollo del proyecto. Su acompañamiento técnico y su experiencia profesional aportaron claridad en las etapas más complejas del proceso.

Finalmente, valoro a todas las personas que, de una u otra forma, ofrecieron palabras de aliento, compartieron conocimientos o facilitaron espacios para avanzar. A cada uno, gracias por contribuir a que esta meta hoy sea una realidad.

Oscar Leonardo Serpas Martínez

Primeramente, agradezco a Dios por haberme brindado sabiduría y perseverancia para culminar mi formación profesional, permitiéndome superar todo tipo de dificultades durante este proceso.

A mi familia, especialmente a mi madre, Lina Aracely de Alcántara, gracias por el amor, cuidado y acompañamiento constante. A mi padre, Marcelino Alcántara, gracias por el esfuerzo, apoyo económico y consejos a lo largo de los años; sin ustedes nada hubiera sido posible. A ustedes dedico especialmente el logro de esta meta. También a mi hermana, Diana Alcántara, por brindarme su apoyo siempre que lo necesité y gracias por escuchar y compartir las dificultades de la vida académica. A mi hijo, I. J. Alcántara, por ser mi motivación para seguir adelante y concluir este proceso.

A mis compañeros de especialización, David y Óscar, con quienes hemos logrado finalizar este proceso pese a los obstáculos. A los docentes de la Escuela de Sistemas Informáticos por compartir sus conocimientos durante toda la carrera. Un agradecimiento especial a la Inga. Sandra Romero, por su guía y orientación académica, y al Ing. Saúl Rodas, por su tiempo y disposición para brindar retroalimentación durante el desarrollo del proyecto.

Marcelo Josué Alcántara Alas.

II. Resumen

El presente proyecto es la evaluación sobre el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador. En la elaboración de este proyecto se desarrollaron varias etapas previas al trabajo final en donde se pondrá a ejecución el plan de pruebas diseñado previamente para evaluar los componentes críticos del sistema, basándose en las directrices con respecto a la calidad del software o un sistema informático encontradas en la norma ISO/IEC 25010.

El primer Capítulo del presente documento tiene como objetivo presentar las generalidades sobre todos los aspectos relacionados con las pruebas de software o de sistemas. Entre los elementos más destacados podemos mencionar el concepto o definición de una prueba y su importancia en el desarrollo de un sistema informático.

Un sistema informático que cumpla con los requerimientos previamente especificados por el usuario final debe además de brindar una experiencia de calidad, brindando a todos los usuarios del sistema una confianza y facilidad de uso en todos los aspectos necesarios para el cumplimiento de sus actividades diarias. Por ello, se expone la importancia de las pruebas y su relevancia para el cumplimiento de los requisitos de un sistema. También se dará a conocer los diferentes procesos para la realización de una prueba, asimismo los niveles y tipos de pruebas que se pueden realizar o ejecutar en un plan de pruebas previamente diseñado.

El segundo capítulo presenta el plan de pruebas para el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación. Para la evaluación de este sistema es esencial elaborar un plan de pruebas que permita identificar las partes críticas del sistema, evaluarlas y documentar todos los defectos encontrados para tomar acciones y así evitar problemas graves de confiabilidad y estabilidad del sistema para el uso dentro de la Secretaría de Investigaciones.

En el plan de pruebas se toma en consideración la norma ISO/IEC 25010 para la elaboración y la ejecución de pruebas, considerando los aspectos o criterios de aprobación para cada caso de prueba. También se documentarán todos los pasos necesarios para realizar las pruebas, así como también se registrarán los resultados obtenidos de la ejecución de pruebas.

Por último, en este capítulo se presentará un plan de riesgos y contingencias en donde se abordarán diferentes aspectos relacionados a los riesgos y se evaluará su impacto dentro del sistema. De igual manera se presentarán diferentes estrategias para la mitigación de estos riesgos y así reducir la probabilidad de encontrar un sistema con fallos graves que causen problemas o pérdidas de información para los usuarios.

En el capítulo final se presentará el caso de estudio del proyecto de la tesina con los resultados obtenidos de la ejecución del plan de pruebas elaborado. Se analizarán los datos obtenidos y se calcularán diferentes métricas y se presentarán todos los elementos planteados para el reporte que se entregará a la institución.

III. Índice De Figuras

IV. Índice De Tablas

V. Introducción

El presente trabajo tiene como finalidad evaluar la calidad del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación, desarrollado para la Secretaría de Investigaciones Científicas de la Universidad de El Salvador (SIC-UES). Dicho software fue creado con el propósito de facilitar la gestión y el registro de proyectos, optimizando los procesos y aumentando la eficiencia en las convocatorias. No obstante, tras una evaluación preliminar, se han identificado diversas deficiencias y vulnerabilidades en su desempeño que podrían comprometer su operatividad y confiabilidad, afectando negativamente la gestión administrativa y académica del personal y usuarios de la SIC-UES.

La evaluación de este proyecto busca identificar errores, vulnerabilidades y áreas de mejora en los componentes más críticos del sistema. Para ello, se diseñará y ejecutará un plan de pruebas basado en el modelo de calidad definido por la norma internacional ISO/IEC 25010. Como parte de esta estrategia, se priorizarán los componentes con mayor impacto en la usabilidad y el funcionamiento del entorno de producción, empleando herramientas de análisis estático y pruebas automatizadas para garantizar la calidad del software.

Este proyecto plantea un diagnóstico detallado del estado actual del sistema, sin ejecutar acciones correctivas o preventivas directas sobre el mismo; más bien, se proporcionarán recomendaciones de mejora y un informe técnico que sirva de base para intervenciones futuras. A través de la identificación y clasificación de defectos conforme a estándares de calidad, el objetivo es entregar a la SIC-UES una base sólida para la optimización y el mantenimiento continuo. De este modo, se espera que los resultados generados cumplan con las expectativas operativas y académicas, apoyando eficientemente los procesos administrativos y facilitando la gestión de proyectos de investigación.

VI. Objetivos

Objetivo General

Evaluar la calidad de los componentes más críticos del Sistema de Gestión de Proyectos de Investigación de la Secretaría de Investigaciones Científicas aquellos con mayor impacto en la usabilidad y operación del sistema mediante análisis y pruebas, con el fin de identificar defectos, mejorar su mantenibilidad y fortalecer su confiabilidad antes del despliegue definitivo.

Objetivos Específicos

- Realizar un análisis del sistema para detectar defectos, vulnerabilidades y malas prácticas en los componentes más críticos a evaluar.
- Evaluar la influencia del diseño de dichos componentes en la mantenibilidad y calidad general del software, con base en los resultados del análisis.
- Diseñar y ejecutar pruebas, utilizando herramientas adecuadas para entornos web, con el fin de validar el comportamiento de los componentes evaluados.
- Elaborar una matriz de pruebas que registre la cobertura funcional y los resultados obtenidos durante la ejecución de pruebas.
- Generar un informe técnico que documente los hallazgos, análisis y recomendaciones para futuras mejoras del sistema, sin intervenir directamente en su implementación.

1. Capítulo 1. Generalidades

1.1. Calidad De Software

En un mundo globalizado donde las organizaciones se ven enfrentadas a competencia de nivel mundial, la calidad se convierte en un importante punto diferenciador, además de aumentar la satisfacción general del cliente, disminuir costos y optimizar los recursos. Los productos o servicios que ostentan certificados de calidad son preferidos por los compradores porque transmiten seguridad y confianza. Esto también constituye un atributo de valor para las estrategias de comercialización en el exterior.

Por este motivo la calidad de un producto o servicio es un aspecto de suma importancia para el crecimiento de una organización. Pero, ¿Qué es la calidad? Entre algunas definiciones encontramos:

a) Según La Real Academia Española (s. f.) define calidad como: "Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor."

b) Según la International Standards Organization (Solorzano, 2022): "La calidad como el grado en el que un conjunto de características inherentes cumple con los requisitos". En otras palabras, la calidad se refiere a la conformidad de un producto o servicio con las expectativas y necesidades de los clientes.

En la industria del software se pueden evidenciar necesidades de satisfacción del cliente de productos o servicios de software, de reducción de recursos invertidos en proyectos de software y de la efectiva asignación de recursos humanos.

La definición de la calidad del software según la IEEE, Std. 610-1990, es "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario". (Azabache Martínez, 2023).

Por lo tanto, podemos definir que la calidad de software es el grado en que un producto digital cumple con las expectativas y requisitos del usuario. Implica el correcto funcionamiento del software, sea confiable y ofrezca una experiencia de usuario satisfactoria. Asegurar una alta calidad de software además de prevenir errores y fallos, también optimiza el rendimiento y la eficiencia del sistema.

1.2. Pruebas De Software

¿Qué es una prueba de software?

La prueba es el proceso de ejecutar un conjunto de elementos software con el fin de encontrar errores. Por tanto, probar no es demostrar que no hay errores en el programa ni únicamente mostrar que el programa funciona correctamente, ambas son definiciones incorrectas y, sin embargo, comúnmente utilizadas (ISTQB, 2021).

El ISTQB (International Software Testing Qualifications Board), una organización sin ánimo de lucro creada en el año 2002 por empresas, instituciones, organizaciones y personas especializadas en el campo de las pruebas y la industria del software (ISTQB, 2024), define las pruebas como:

> El proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas relacionadas con la planificación, preparación y evaluación de productos de software y productos relacionados con el trabajo para determinar que cumplen los requisitos especificados, para demostrar que son aptos para el propósito y para detectar defectos (ISTQB, 2024, p. 15).
>
> Para comprender estas palabras y por ende las pruebas, vamos a ver como las define el ISTQB:
>
> Una persona puede cometer un error que a su vez puede producir un defecto en el código de programa o en un documento. Si se ejecuta un defecto en el código, el sistema puede no hacer lo que debiera (o hacer algo que no debiera), lo que provocaría un fallo. Algunos defectos de software pueden dar lugar a fallos, pero no todos los defectos lo hacen (ISTQB, 2024, p. 17).

Definiendo cada término se tiene:

a. **Error:** se produce por la acción humana, normalmente producido por el desarrollador encargado del método o función y a través de una mala interpretación se producirá un resultado no esperado (ISTQB, 2024).

b. **Defecto:** provocado por un error en la implementación de un método que como consecuencia el sistema no realizará la acción esperada (ISTQB, 2024).

c. **Fallo:** la definición de fallo se puede determinar como la ejecución del sistema con un defecto que podría provocar resultados erróneos o no deseados (ISTQB, 2024).

En definitiva, se puede indicar que las pruebas de software son un proceso crítico en el ciclo de vida del desarrollo de software que implica evaluar la funcionalidad, fiabilidad y rendimiento de una aplicación de software. Mediante la ejecución metódica de una serie de pruebas, los desarrolladores pueden verificar que el software se alinea con los requisitos especificados, descubriendo cualquier defecto o problema que necesite ser abordado.

La importancia de las pruebas

El alcance de las pruebas de software abarca desde la validación de la funcionalidad básica hasta la inspección profunda de seguridad, rendimiento y compatibilidad con otros sistemas. Puede realizarse en diferentes etapas del desarrollo y puede variar desde probar pequeñas unidades de código (pruebas unitarias) hasta examinar el comportamiento de todo el sistema (pruebas del sistema). Cuando se combina con prácticas de desarrollo con métodos de automatización como la integración continua (CI), las evaluaciones se transforman de una fase independiente a una parte continua e integral del proceso de desarrollo, asegurando la calidad y eficiencia continuas.

La importancia de este análisis radica en que sirven como un componente crítico en el proceso de desarrollo de software, ofreciendo beneficios que se extienden mucho más allá de identificar y corregir errores. Cuando se implementan de manera efectiva, la evaluación transforma la calidad general, la usabilidad y la efectividad de la aplicación final y permitir a los equipos agilizar la innovación de forma sostenida.

1.3. Proceso De Pruebas

El proceso de pruebas de software ha ido evolucionando con los años. En el pasado con las metodologías tradicionales de desarrollo las pruebas de software se trataban como una fase distinta realizada después de completar todo el trabajo de desarrollo. Esto significaba que los defectos se descubren tarde en el desarrollo del software, lo que llevaba a esfuerzos de corrección largos, complejos y costosos. También dificultaba que los equipos incorporarán comentarios de los usuarios o se adapten a requisitos cambiantes, ya que cambios importantes al final del ciclo de desarrollo podrían ser prohibitivamente costosos y consumir mucho tiempo de revisión y configuración.

A medida que las pruebas se han entrelazado más profundamente en el proceso de desarrollo, los equipos de desarrollo han implementado un enfoque moderno y sistemático conocido como Ciclo de Vida de Pruebas de Software o Software Testing Life Cycle (STLC, por sus siglas en inglés). Las fases en el STLC incluyen:

1. **Análisis de requisitos:** los probadores colaboran con las partes interesadas para comprender los requisitos desde un punto de vista de pruebas, identificando lo esencial para probar.

2. **Planificación de pruebas:** en esta etapa se crea un plan de pruebas que describa los objetivos, recursos, cronograma y metodologías para los esfuerzos de prueba, estableciendo una dirección clara para las actividades futuras.

3. **Desarrollo de casos de prueba:** basados en los requisitos y el plan de pruebas, se desarrollan casos de prueba, cubriendo todos los aspectos de funcionalidad, rendimiento y seguridad para asegurar pruebas exhaustivas.

4. **Configuración del entorno de prueba:** la fase de configuración del entorno de pruebas define las condiciones bajo las cuales se realizarán las pruebas de software**.** Se debe de configurar el hardware/software necesario para la ejecución de pruebas.

5. **Ejecución de pruebas:** se llevan a cabo los casos de prueba previamente diseñados, se documentan los resultados y se registra cualquier defecto encontrado. Esta fase es crítica en el descubrimiento real y registro de fallos de software.

6. **Cierre de pruebas:** la fase final implica recopilar los resultados de las pruebas en un informe o reporte, proporcionando una visión general completa de las actividades de prueba, resultados y perspectivas para mejoras futuras.

El STLC se ajusta perfectamente al paradigma moderno de integración continua, en el cual los desarrolladores realizan cambios de código pequeños y frecuentes y los fusionan en un repositorio compartido, donde se ejecutan procesos automatizados de construcción y prueba. Este enfoque sistemático asegura que cada integración sea verificada a través de un conjunto completo de pruebas y permite a los equipos identificar y abordar problemas rápidamente, manteniendo un sistema actualizado y comprobando su efectividad (Son, 2025).

1.4. Niveles De Prueba

El ISTQB (International Software Testing Qualifications Board) es el estándar internacional que se ha venido especializando en el campo de la certificación de pruebas, hace diferencia entre niveles y tipos.

Los niveles de prueba se organizan y administran juntos, cada nivel es una instancia en el proceso de pruebas para evaluar el software en distintas fases de desarrollo. Además, cada nivel se aplica a una etapa específica, desde componentes hasta sistemas completos. Puede solaparse con otros niveles en ciertos modelos de desarrollo. En modelos secuenciales, los criterios de salida de un nivel sirven como entrada para el siguiente (ISTQB, 2024).

Los niveles de prueba se dividen en cuatro niveles (ISTQB, 2024):

1. Prueba de componentes.
2. Prueba de integración.
3. Prueba de sistema.
4. Prueba de aceptación.

1.4.1. Pruebas De Componentes

También conocidas como pruebas unitarias son pruebas de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa tu software. En general, las pruebas unitarias son excelentes candidatas a la automatización y hechas de forma separada del resto del sistema, además se pueden ejecutar rápidamente mediante un servidor de integración continua (Atlassian, s. f.).

1.4.2. Pruebas De Integración

Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto y están enfocadas en la interacción entre los componentes o los sistemas, por tanto, se subdividen en:

a. **Integración de componentes:** prueba la interacción e interfaces entre los componentes, son hechas luego de las pruebas de componentes y son perfectas candidatas por automatizar.

b. **Integración de sistemas:** se prueba la interacción e interfaz entre los sistemas, paquetes y microservicios. Además, se hace la comprobación de la interacción del sistema con interfaces que proveen sistemas externos de ser requerido. Estas pruebas se pueden realizar luego de las pruebas de sistemas o en paralelo.

Algunos de los objetos de prueba en este nivel son: subsistemas, base de datos, interfaces, API’s, microservicios. En estas pruebas se debe enfocar en la comunicación entre módulos o sistemas. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y según lo esperado. Debemos tener en cuenta que estas pruebas tienen un costo elevado para su ejecución, ya que requieren que varias partes de la aplicación estén en marcha.

1.4.3. Pruebas De Sistema

Se centran en el funcionamiento y capacidades de un sistema/producto completo, se consideran todas las tareas que el sistema puede realizar y los comportamientos no funcionales que muestra mientras cumple con estas tareas. Además, verifican que diversos flujos de usuario funcionen según lo previsto, y pueden ser tan sencillos como cargar una página web o iniciar sesión, o mucho más complejos, como la verificación de notificaciones de correo electrónico, pagos en línea, entre otros.

Las pruebas de sistemas son muy útiles, pero son costosas de llevar a cabo y pueden resultar difíciles de mantener cuando están automatizadas. Los objetos de prueba son aplicaciones y sistemas enteros.

Entre los principales objetivos de las pruebas de sistema se encuentran (Atlassian, s. f.):

a. Reducir el riesgo.

b. Verificar si el comportamiento funcional y no funcional del sistema cumplen con los requisitos diseñados y especificados.

c. Encontrar defectos.

d. Validar que el sistema está completo y trabaja como se esperaba.

e. Validar la calidad de los datos.

1.4.4. Pruebas De Aceptación

Las pruebas de aceptación son pruebas que verifican si un sistema satisface los requisitos empresariales previamente establecidos con las partes interesadas. Es necesario que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

Por ese motivo al igual que las pruebas de sistema se centran en el comportamiento y capacidades de un sistema/producto completo y entre sus principales objetivos están: establecer confianza en la calidad del sistema completo, validar que el sistema está completo y trabaja como se esperaba. Cabe destacar que las pruebas de aceptación son principalmente responsabilidad de los clientes o product owners. Los objetos de prueba en este nivel son principalmente: sistemas en pruebas, procesos de negocio, reportes, entre otros.

Las formas más comunes de pruebas de aceptación son (ISTQB, 2024):

1. Pruebas de aceptación de usuario (UAT): se enfocan en validar si el sistema se adapta al usuario final en un entorno operativo real o simulado.

2. Pruebas de aceptación operacional: son hechas por los usuarios administradores del sistema en un ambiente simulado de producción. Se enfocan en aspectos operacionales cómo: probar el respaldo y recuperación; instalación, actualización y desinstalación; administración de usuario y tareas de mantenimiento. Pero el principal objetivo es darle

confianza, a los operadores o administradores del sistema, de que este se mantendrá trabajando apropiadamente en un ambiente operacional con condiciones excepcionales e inesperadas.

3. Pruebas de aceptación contractuales y regulatorias: son realizadas contra los criterios de aceptación de un contrato para producir software hecho a medida, los criterios de aceptación deben ser definidos cuándo las partes están de acuerdo con el contrato.

4. Pruebas de aceptación Alfa y Beta: son usadas por programadores de software comercial listo para usarse; que desean obtener comentarios de los usuarios, clientes u operadores potenciales antes de poner el software en el mercado.

1.5. Tipos de Pruebas

Existen diferentes tipos de pruebas de software que se enfocan en evaluar las diferentes características de calidad del software.

Según la ISTQB (2021) en el Syllabus 2018 v3.1.1 define lo siguiente: "Un tipo de prueba es un grupo de actividades de prueba destinadas a probar características específicas de un sistema de software, o una parte de un sistema, en función de objetivos de prueba específicos."

Entre los objetivos específicos que se mencionan de los tipos de pruebas se pueden mencionar (ISTQB, 2021):

- Evaluación de características de calidad funcional, cómo integridad, corrección y oportunidad.
- Evaluación de características de calidad no funcionales, como confiabilidad, eficiencia de desempeño, seguridad, compatibilidad y usabilidad.
- Evaluar si la estructura o arquitectura del componente o sistema es correcta, completa, y como se especifica.
- Evaluar los efectos de los cambios, como confirmar que se han solucionado los defectos (confirmación de pruebas) y buscar cambios no deseados en el comportamiento resultantes del software o cambios en el entorno de desarrollo (pruebas de regresión).

Entre los tipos de pruebas se encuentran: Pruebas Funcionales, Pruebas No Funcionales, Pruebas Estructurales y Pruebas de Regresión (ISTQB, 2021).

1.5.1. Pruebas Funcionales

Son las pruebas más conocidas y mencionadas en el mundo, las pruebas funcionales se centran en los requisitos empresariales de una aplicación y solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

A veces, se confunden las pruebas de integración con las funcionales, ya que ambas requieren que varios componentes interactúen entre sí. La diferencia es que una prueba de integración puede simplemente

verificar que puedes hacer consultas en la base de datos, mientras que una prueba funcional esperaría obtener un valor específico desde la base de datos, según dictan los requisitos del producto.

El diseño y la ejecución de pruebas funcionales pueden implicar habilidades o conocimientos especiales, como tener el conocimiento de la problemática empresarial particular que el sistema en evaluación resuelve.

1.5.2. Pruebas No Funcionales

Las pruebas no funcionales de un sistema evalúan las características de los sistemas y el software, como la usabilidad, eficiencia del desempeño o seguridad, por lo tanto, las pruebas no funcionales son las encargadas de validar el "qué tan bien" el sistema se comporta.

Contrario a la opinión común, las pruebas no funcionales deben hacerse en todos los niveles y realizarse lo antes posible. La detección tardía de defectos no funcionales puede poner en grave riesgo el éxito del proyecto, además de implicar costos elevados para su corrección.

Entre las técnicas más comunes para realizar pruebas no funcionales se encuentra la técnica de Caja Negra (Black-box en inglés) para derivar condiciones de prueba y casos de prueba para no pruebas funcionales.

Las pruebas no funcionales requieren conocimientos especiales para poder diseñarlas y ejecutarlas. Entre el conocimiento requerido está conocer las debilidades inherentes del diseño del sistema o de la tecnología utilizada para el desarrollo del sistema o también la base de usuarios que se esperan utilicen el sistema.

1.5.3. Pruebas Estructurales

Estas pruebas también conocidas como Pruebas de Caja Blanca son las que validan la estructura interna del sistema. Dentro de esta estructura interna se puede incluir el código, arquitectura, flujo de trabajo y flujo de datos dentro del sistema. La cobertura de código está basada en el porcentaje de componentes que han sido probados. Para diseñar y ejecutar las pruebas de caja blanca es necesario tener conocimiento sobre cómo fue construido el código, cómo están guardados los datos, entre otros.

El diseño y la ejecución de pruebas de caja blanca pueden requerir habilidades o conocimientos especiales, como la forma en que se construye el código, cómo se almacenan los datos (por ejemplo, para evaluar posibles consultas a la base de datos) y cómo utilizar las herramientas de cobertura e interpretar correctamente sus resultados (ISTQB, 2021).

1.5.4. Pruebas De Regresión

Cuando se realizan cambios al sistema por cualquier motivo como corregir un defecto o añadir una funcionalidad se deben hacer pruebas para verificar que los cambios han corregido el defecto o han implementado correctamente la funcionalidad y que no han causado consecuencias adversas.

Es posible que un cambio realizado en una parte del código, ya sea una corrección u otro tipo de cambio, afecte accidentalmente al comportamiento de otras partes del código, ya sea dentro del mismo

componente, en otros componentes del mismo sistema o incluso en otros sistemas vinculados. Los cambios pueden incluir modificaciones en el entorno, como una nueva versión de un sistema operativo o un sistema de gestión de bases de datos. Estos efectos secundarios no deseados se denominan regresiones.

Las pruebas de regresión consisten en ejecutar pruebas para detectar esos efectos secundarios no deseados. Estas pruebas siempre se deben de realizar en todos los niveles de pruebas.

Las pruebas de regresión son esenciales especialmente en ciclos de vida de desarrollo iterativos e incrementales como en las metodologías ágiles que constantemente agregan nuevas funcionalidades, cuando se realizan cambios a componentes ya existentes, o se realiza la refactorización de un código son requeridas las pruebas de regresión para comprobar que todo funciona correctamente.

Dado que las suites de regresión se ejecutan repetidamente y tienden a ser estables en el tiempo, constituyen una candidata ideal para la automatización. Por ello, su implementación debería iniciarse desde las etapas tempranas del ciclo de vida del software.

1.6. Testing Outline

Objetivo general

Evaluar la calidad del código fuente del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador con el fin de identificar defectos, vulnerabilidades y malas prácticas que puedan afectar la mantenibilidad, seguridad y fiabilidad del mismo antes de su puesta en producción.

Objetivos específicos

1. Identificar defectos técnicos, vulnerabilidades de seguridad y code smells mediante el uso y configuración de herramientas según el tipo y alcance de las pruebas.
2. Evaluar la calidad del sistema en función de los atributos de la norma ISO/IEC 25010 aplicables a pruebas estáticas: mantenibilidad, fiabilidad, seguridad y eficiencia.
3. Detectar y documentar riesgos potenciales en el código que puedan impactar negativamente en la operación futura del sistema.
4. Generar un informe estructurado de hallazgos que sirva como insumo técnico para la toma de decisiones sobre mejoras en el sistema.

1.6.1. Tipos y Niveles de Pruebas

1.6.1.1. Pruebas Estáticas

Las pruebas estáticas consisten en analizar el software y su documentación sin necesidad de ejecutarlo. Se revisa el código fuente, los documentos de requisitos y los casos de prueba mediante inspección manual y herramientas automáticas como SonarQube. Esto permite identificar defectos de programación, vulnerabilidades de seguridad, inconsistencias y malas prácticas de desarrollo desde etapas tempranas.

Las principales actividades incluyen:

1. Revisión de código fuente buscando errores, duplicidad o problemas de mantenimiento.
2. Inspección de documentación técnica y de casos de prueba para asegurar que los requisitos estén bien detallados y claros.
3. Uso de análisis automatizado para detectar vulnerabilidades, complejidades excesivas y problemas de calidad en el código.

El propósito es detectar problemas y defectos antes de la ejecución del sistema, generando un reporte detallado para que puedan ser atendidos posteriormente, de acuerdo a los planes de mejora y mantenimiento definidos por la institución. De esta forma, se contribuye a una mejor comprensión y preparación del sistema previo a la ejecución de las pruebas contempladas en este documento.

1.6.1.2. *Pruebas Dinámicas*

Las pruebas dinámicas implican la ejecución real del software para verificar su comportamiento y validar que cumple con las funciones y requisitos definidos. Aquí el sistema es sometido a diferentes condiciones y escenarios, tanto esperados como inesperados, para observar las respuestas y registrar resultados.

Las actividades clave son:

1. Ejecución de casos de prueba funcionales y no funcionales que permitan comprobar el correcto funcionamiento en los principales módulos del sistema.
2. Pruebas automatizadas utilizando herramientas como Laravel Dusk y Postman para asegurar repetibilidad y eficiencia en la validación.
3. Evaluación del sistema desde el punto de vista del usuario final (pruebas de usuario y aceptación).

El propósito es asegurar que el sistema responde de forma adecuada a las necesidades reales y detectar defectos que sólo se manifiestan durante su funcionamiento.

Todos los hallazgos se documentan en la matriz de pruebas, aportando evidencia para futuras mejoras y priorizando la solución de los problemas detectados.

1.6.1.3. Niveles

Según el ISTQB se definen los siguientes niveles de prueba tomando en cuenta el enfoque y que es lo que el equipo desea evaluar (ISTQB, 2021).

1. Unitarias (sobre componentes individuales, idealmente por desarrolladores)
2. Integración de componentes (interacción entre partes del sistema)
3. Sistema (evaluación de la solución completa, como usuario final)
4. Integración de sistemas (interacción con otros sistemas, si aplica)

5. Aceptación (validación de cumplimiento de requisitos de negocio)

1.6.2. Priorización y Estrategia de Riesgo

Las pruebas se priorizan según el riesgo e impacto esperado en la operación de la Secretaría. Se ejecutan primero sobre los módulos más críticos y luego sobre los secundarios. Se utilizarán técnicas de pruebas automatizadas y manuales, según la criticidad de los módulos y la disponibilidad de recursos de prueba.

1.6.3 Herramientas y Gestión

Herramientas:

- SonarQube para análisis estático
- Microsoft Project para la realización de las tareas y actividades
- Laravel Dusk para automatización de pruebas funcionales de interfaz
- Microsoft Excel para la matriz de pruebas y seguimiento

Se establecerá una matriz de pruebas con los casos, resultados, cobertura y evidencias, que será actualizada periódicamente y almacenada para trazabilidad y análisis posterior.

1.6.4. Documentación y Entregables

1. Todos los resultados serán documentados: reporte de defectos, cobertura funcional, métricas de calidad, y un informe técnico con los hallazgos y recomendaciones.
2. No se realizarán correcciones directas en el código, sino que las recomendaciones serán transferidas a la Secretaría responsable del sistema.
3. La documentación final incluirá recomendaciones técnicas, priorización de riesgos y métricas de ejecución y cobertura de pruebas, conforme a los estándares internacionales.

2. Capítulo 2. Plan De Pruebas

2.1. Definición de plan de pruebas

El plan de pruebas tiene como objetivo principal evaluar la calidad del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador. Dado el papel fundamental que desempeña esta herramienta en la administración y gestión de los proyectos de investigación, resulta esencial garantizar su funcionamiento óptimo antes de su despliegue definitivo.

La evaluación se basará en los estándares definidos por la norma internacional ISO/IEC 25010, permitiendo analizar las características clave del sistema, como la funcionalidad, la usabilidad, el rendimiento, la fiabilidad y la seguridad. El enfoque del plan se centrará en los componentes críticos del sistema, particularmente aquellos con mayor impacto en la experiencia del usuario y en la operatividad en entornos reales de producción.

2.2. Norma ISO 25010

La norma ISO/IEC 25010:2011 proporciona un marco de referencia común para evaluar la calidad de productos y sistemas de software. Este estándar define un conjunto de características y subcaracterísticas de calidad que permiten a las organizaciones establecer criterios claros para determinar si un producto cumple con los requisitos de calidad establecidos.

Características Clave De La ISO/IEC 25010

Según la ISO 25010(2011) divide la calidad del software en nueve características principales:

1. Adecuación funcional: Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas de los usuarios cuando el producto se usa en las condiciones especificadas.

2. Eficiencia de desempeño: Esta característica representa el desempeño de un producto en las funciones en unos parámetros de tiempo y rendimientos especificados y con un uso eficiente de recursos (CPU, memoria, almacenamiento, energía) utilizados bajo ciertas condiciones.

3. Compatibilidad: La capacidad de un producto para intercambiar información con otros productos y/o para realizar sus funciones requeridas cuando comparte un entorno y recursos comunes.

4. Capacidad de Interacción: Relacionada a la característica de la Usabilidad en la norma ISO 25010 se define como: capacidad del producto software para que el usuario interactúe mediante su interfaz intercambiando información para completar determinadas tareas.

5. Fiabilidad: Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados sin interrupciones o fallos.

6. Seguridad: La capacidad de un producto para protegerse a sí mismo y a sus usuarios del acceso, uso, divulgación, interrupción o daño no autorizado.

7. Mantenibilidad: Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

8. Flexibilidad: Relacionada con la característica de la Portabilidad según la ISO 25010 es la capacidad del producto para adaptarse a cambios en sus requisitos, contextos de uso o entorno del sistema.

9. Protección: Esta característica representa la capacidad del producto, en condiciones definidas, de evitar un estado en el que se ponga en peligro la vida humana, la salud, la propiedad o el medio ambiente.

Para explicar las características que tienen mayor impacto en el riesgo del producto para este proyecto, es importante detallar sus subcaracterísticas y cómo influyen en la calidad general. Nos enfocaremos en tres características clave: Adecuación Funcional, Seguridad, y Capacidad de Interacción, tal como se describe en la norma ISO 25010.

Costo de implementación

La norma ISO/IEC 25010 es un estándar internacional de calidad para los productos de software o sistemas, que pueden usarse como referencia para definir, medir y evaluar calidad. El costo inicial se puede generar en la adquisición del documento oficial de la norma ISO/IEC 25010 suministrado por la International Standard Organization (ISO). Este documento puede variar dependiendo de la región en que se adquiere y el formato del documento, pero su precio para octubre de 2025 fue de 132.00 CHF (franco suizo) convertido a dólares estadounidenses son $166.42.

La implementación o adaptación dentro de la organización para el cumplimiento de estándares de calidad tiene un costo que depende de los siguientes factores:

a. El tamaño del sistema informático que se va a evaluar o adaptar (referencia a la cantidad de módulos, complejidad, tecnología, número de usuarios, entre otros).
b. El estado de la organización en referencia a procesos de calidad de software o la necesidad de una capacitación en referente al tema de los procesos de calidad.
c. Los costos de capacitación del personal, adaptación de métricas, seguimiento, herramientas de medición y automatización, etc.
d. La necesidad de auditorías o asistencia externa a la organización, teniendo en cuenta la cantidad de recursos necesarios y el tiempo requerido.

Todos estos factores pueden afectar a la adaptación o implementación de la norma ISO/IEC 25010 dentro de la organización. Estos factores se deben de tomar en cuenta para determinar las mejores opciones para el cumplimiento de procesos de calidad ya sea con ayuda externa o con la capacitación de personal dentro de la misma organización (ISO 25010, 2011).

El costo de una certificación realizada a través de una empresa certificadora puede conllevar un coste alto. Por ejemplo, podemos encontrar el costo de certificación que ofrece la empresa Pacific Certifications (ISO/IEC 25010:2023-Systems and Software Engineering Pacific Certifications, 2025) en donde el costo varía según los aspectos que se desean evaluar del software o sistema. Teniendo en cuenta esto podemos visualizar los siguientes costos:

1. Evaluación de brechas y análisis de requisitos: Menos de $5,000 dólares.
2. Desarrollo y capacitación de un marco de calidad: menos de $10,000 dólares.
3. Validación externa o integración de herramientas: Menos de $15,000 dólares, dependiendo de la escala (Pacific Certifications, 2025).

2.3. Identificador del Plan de Pruebas

El plan de pruebas evaluará la calidad de los módulos críticos del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador.

En este aspecto se ha identificado como un plan de prueba específico para los módulos críticos del sistema en base. Su identificación es: **PP-SIGCEST-V1.0-2025** (Plan de Pruebas para el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas - versión 1.0, 2025).

2.4. Elementos de las pruebas

Como elementos de las pruebas a realizar se tiene que considerar el tipo de pruebas a realizar así como el objetivo que estas persiguen al ser ejecutadas sobre el sistema y el alcance que se pretende.

Como objetivo de las pruebas se puede mencionar lo siguiente:

- Evaluar la calidad del código fuente del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador con el fin de identificar defectos, vulnerabilidades y malas prácticas que puedan afectar la mantenibilidad, seguridad y fiabilidad del sistema antes de su puesta en producción.

El alcance las pruebas se define a continuación:

1. Evaluación de funcionalidad: evaluar las funciones clave del sistema y que estos cumplan con los requisitos definidos para los diferentes módulos y roles, evaluando así el comportamiento esperado.
2. Pruebas priorizadas basadas en riesgo: ejecutar las pruebas teniendo como base las áreas más críticas del sistema dando prioridad a aquellas funcionalidades que ante una falla tenga mayor impacto para la Secretaría aplicando estrategias de pruebas automatizadas.

Considerando el objetivo y los alcances del plan de pruebas, así como los requisitos del negocio, se determinó la realización de las siguientes pruebas:

Pruebas funcionales:

Encaminado a validar la característica de Adecuación Funcional se llevarán a cabo pruebas con el objetivo de evaluar que todas las funcionalidades del sistema operen de acuerdo con los requisitos especificados.

Estas pruebas se encargan de comprobar que cada módulo cumpla su propósito correctamente y sin errores, garantizando una experiencia fluida y precisa para los usuarios.

Pruebas no funcionales:

Además de las pruebas funcionales, se implementarán pruebas no funcionales para evaluar aspectos clave del sistema que no están relacionados directamente con las funciones, pero que impactan en su desempeño general:

1. Pruebas de compatibilidad: se verificará la compatibilidad en diferentes entornos de ejecución (navegadores, dispositivos) para asegurar que el sistema funcione correctamente en todas las configuraciones previstas y mantenga una presentación y operación consistentes.

2. Pruebas Rendimiento: se evaluará la Eficiencia de Desempeño con la capacidad del sistema para manejar cargas de trabajo bajo diferentes condiciones, midiendo tiempos de respuesta, estabilidad y el uso eficiente de recursos del sistema.

3. Pruebas de seguridad: se validará la Seguridad de la protección de datos sensibles, como la información de usuarios y transacciones, verificando que el sistema cuente con mecanismos adecuados para evitar vulnerabilidades y prevenir accesos no autorizados. También se revisarán los mecanismos de autenticación y autorización para asegurar que los roles y permisos se apliquen correctamente.

4. Las pruebas de integridad de datos: son aquellas que aseguran la exactitud, consistencia y coherencia de los datos en un sistema. Verifican que los datos se mantengan íntegros y cumplan con reglas de negocio y restricciones de la base de datos durante su procesamiento, garantizando así la confiabilidad de la información.

Estos elementos detallan cuales son los elementos de las pruebas a realizar y también en su conjunto permiten identificar cuáles son los riesgos asociados y las posibles características que se probaran y las que no se probaran.

2.5. Riesgos

Los riesgos de calidad tienen como propósito identificar y evaluar los riesgos que puedan afectar de manera significativa la funcionalidad en las áreas críticas que tengan mayor riesgo potencial para la Secretaría de Investigaciones Científicas de la Universidad de El Salvador.

La identificación de riesgos es importante para la determinación de aquellos módulos o actividades específicas que formarán parte de la matriz de pruebas.

Dentro de los riesgos cabe mencionar están los riesgos de producto y riesgos de proyecto. Estos riesgos pueden ser clasificados por probabilidad, impacto, nivel de riesgo y prioridad.

Metodología para la asignación de valores

Para definir los valores de probabilidad e impacto en la matriz de riesgos, se utilizó el criterio técnico del equipo de pruebas, basándose en el análisis del estado actual del sistema y la revisión de la documentación entregada por la Secretaría de Investigaciones Científicas.

Criterio para la Probabilidad

La probabilidad (del 1 al 5) mide qué tan posible es que un fallo ocurra. Se asignaron valores altos (Alta y Muy Alta probabilidad) a las partes del sistema donde se observó:

1. Falta de revisiones previas: módulos que no fueron probados anteriormente por los desarrolladores.
2. Código complejo: secciones del código que son muy extensas o difíciles de entender, lo que aumenta la posibilidad de errores ocultos.
3. Falta de controles: áreas donde el sistema no verifica correctamente los datos que ingresa el usuario.

Por el contrario, se asignaron valores bajos a los módulos que usan funciones estándar y probadas, donde es difícil que algo falle.

Criterio para el impacto

El impacto mide el daño que causaría el fallo si llegara a ocurrir. Se consideraron de impacto alto o muy alto aquellos problemas que:

1. Detienen las tareas principales: fallos que impiden completar el objetivo del sistema, como no poder registrar una convocatoria o un proyecto.
2. Ponen en riesgo la información: errores que podrían borrar datos, mezclarlos o exponer información privada de los investigadores.
3. Interrumpen el trabajo administrativo: problemas que bloquean los pasos de aprobación, impidiendo que la Secretaría continúe con sus trámites normales.

Esta evaluación permite enfocar las pruebas en las áreas más delicadas e importantes para el funcionamiento de la institución.

Para la priorización de riesgos se utilizaron las siguientes escalas:

Tabla 1
Escala de riesgos.

Valor	Descripción
1	Muy baja probabilidad: la ocurrencia de este tipo de riesgo es sumamente improbable, casi imposible.
2	Baja probabilidad: tiene una ocurrencia poco probable, pero no es imposible.
3	Media probabilidad: este tipo de riesgo puede o no suceder.
4	Alta probabilidad: hay una probabilidad más grande de que ocurra a que no ocurra.
5	Muy alta probabilidad: casi es seguro que el riesgo exista y ocurra.

Nota. Tabla de escala de riesgo de sistema con valores establecidos del 1 al 5.

Evaluación de impacto.

Tabla 2
Escala de impacto.

Valor	Descripción
1	Impacto muy bajo: los efectos que se perciben son mínimos y no afectan el funcionamiento general del sistema, el uso del sistema continua sin interrupciones.
2	Impacto bajo: los efectos son aún pequeños y afectan de forma limitada la operación del sistema, pueden causar molestias menores a los usuarios. No hay impacto a funciones críticas.
3	Impacto medio: en caso de existir estos pueden causar efectos moderados y requieren atención. Pueden causar efectos significativos y afectar funcionalidad críticas pero el sistema continúa siendo utilizable.
4	Impacto alto: pueden ocasionar fallos parciales dentro del sistema, puede existir una interrupción de la operación normal del sistema y afectar funcionalidades importantes.
5	Impacto muy alto: puede provocar que el sistema no funcione de manera correcta. Existe afectación de las funciones principales y sus dependencias hay una interrupción crítica del sistema.

Nota. Tabla de escala de impacto de sistema con valores establecidos del 1 al 5.

Nivel de riesgo.

El nivel de riesgo puede calcularse multiplicando la probabilidad por el impacto. De forma gráfica se puede utilizar la matriz para identificar cual es el nivel de riesgo, obteniendo la siguiente clasificación:

1. Nivel de riesgo bajo: 1 - 6.
2. Nivel de riesgo medio: 7 - 14.
3. Nivel de riesgo alto: 15 - 25.

Figura 1

Matriz de calor de riesgos.

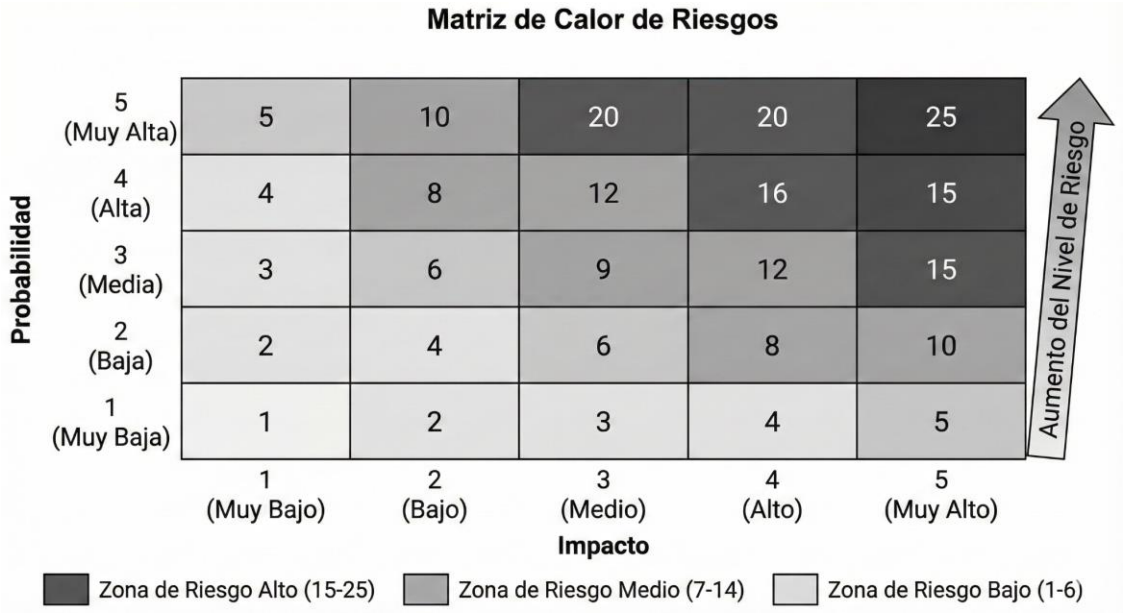


Tabla 3

Riesgos de producto.

ID	Riesgo	Descripción	Proba-bilidad	Impac-to	Nivel de riesgo	Priori-dad
1	Funcionalidades incompletas o con errores críticos	No existe validación de los datos ingresados o se permite información duplicada.	5	5	25	Alta
2	Inconsistencia con las expectativas del cliente.	Inconsistencia con lo que el cliente tiene al momento como sistema y lo que se evalúa.	5	3	15	Alta
3	Errores de ejecución.	El sistema no muestra las funcionalidades de forma correcta o no las muestra.	4	4	16	Alta

ID	Riesgo	Descripción	Proba-bilidad	Impac-to	Nivel de riesgo	Priori-dad
4	Seguridad de la información.	Se puede tener acceso no autorizado al sistema y a la información confidencial que se resguarda.	5	5	25	Alto

Nota. Tabla que describe los riesgos de productos, su probabilidad, impacto, nivel de riesgo y su prioridad.

Tabla 4

Riesgos de proyecto.

ID	Riesgo	Descripción	Proba-bilidad	Impac-to	Nivel de riesgo	Priori-dad
1	Problemas de comunicación	Falta o falla en la comunicación con los clientes usuarios del sistema.	5	3	15	Alta
2	Problemas técnicos.	Documentación incompleta o inadecuada que no permita realizar un correcto proceso de instalación del sistema y por consiguiente existan dificultades para la preparación del ambiente de pruebas.	4	4	16	Alta
3	Retrasos en ejecución	Demoras respecto al cronograma de pruebas y entregables, afectando la priorización y la cobertura planificada.	4	4	16	Alta
4	Deficiencias en documentación	Falta, desactualización o inconsistencia de documentos clave para instalación, operación y pruebas del sistema.	4	3	12	Media

Nota. Tabla donde se describen los riesgos de proyectos y su probabilidad, impacto, nivel de riesgo y prioridad.

2.6. Características que se probarán

Dentro del sistema se enfocará en los módulos críticos del sistema según la funcionalidad de cada módulo (Véase Anexo B para el diagrama completo del sistema) se presentan las características que se probarán:

Tabla 5
Características que se probarán.

Módulo	**Características**
Usuario y permisos	1. Validación de inicio de sesión. 2. Registro de usuario. 3. Seguridad y autenticación. 4. Validación del comportamiento del sistema después de la autenticación.
Convocatorias	1. Creación de convocatorias. 2. Edición de convocatorias. 3. Notificaciones de convocatorias.
Estadísticas	1. Carga y actualización de gráficos por módulo.
Inventario	1. Creación de recurso de inventario y visualización en la lista de inventario. 2. Edición de recursos en el inventario y visualización actualizada. 3. Eliminación de recursos en el inventario. 4. Visualización de lista de recursos en el inventario.
Perfil del investigador	1. Gestión de curriculum del perfil del investigador. 2. Gestión de datos personales en el sistema. 3. Gestión de investigaciones. 4. Gestión de financiamiento de investigación. 5. Gestión de formación académica del perfil investigador. 6. Gestión de proyección social.
Proyectos de investigación antiguos	1. Carga de formulario para visualización de proyectos antiguos.
Proyectos de investigación	1. Gestión de proyectos de investigación. 2. Gestión de recursos en proyectos de investigación. 3. Visualización de proyectos por usuario. 4. Descarga de plantillas/resumen de proyectos. 5. Visualización de proyectos completados. 6. Visualización de dashboard con información sobre el progreso del proyecto de investigación.
Reportes	1. Visualización de reportes. 2. Filtro de información para los reportes de los proyectos. 3. Descarga de reportes en formato PDF y Excel.
Revisión de perfiles	1. Visualización de proyectos enviados para revisión. 2. Paginación en perfiles enviados.
Transversal	1. Renderizado en diferentes navegadores web (Chrome, Firefox y Edge). 2. Responsividad en diferentes sistemas operativos móviles (Android

Módulo	Características
	e IOS). 3. Consistencia en diferentes resoluciones de pantalla. 4. Descarga y subida de archivos en formato PDF y Excel. 5. Rendimiento en dispositivos móviles y escritorio.

Nota. Características que se probarán según cada módulo evaluado del sistema.

2.7. Características que no se probarán

Con base a los requerimientos brindados por los encargados del sistema dentro de la Secretaría de Investigaciones se determina que el siguiente módulo no se probara ya que no representa una funcionalidad crítica o de interés a corto plazo.

Tabla 6
Características que no se probarán.

Módulo	Características
Catálogos	1. Gestión de los tipos de catálogos del área de conocimiento. 2. Gestión de catálogos del área de actividad registrados. 3. Gestión de catálogos por tipo de recurso registrado. 4. Gestión de catálogos del tipo de publicación registrados. 5. Gestión de catálogos del tipo de proyectos registrados.

Nota. Características del sistema que no se evaluarán.

2.8. Alcance (Estrategia)

Evaluación de funcionalidad: evaluar las funciones clave del sistema y que estos cumplan con los requisitos definidos para los diferentes módulos y roles, evaluando así el comportamiento esperado.

Pruebas priorizadas basadas en riesgo: ejecutar las pruebas teniendo como base las áreas más críticas del sistema dando prioridad a aquellas funcionalidades que ante una falla tenga mayor impacto para la Secretaría aplicando estrategias de pruebas automatizadas.

2.9. Criterios de Aprobación/Fallo

Los criterios de aprobación o fallo están descritos para cada caso de prueba registrado en la matriz de pruebas. Para cada caso de prueba se toma en consideración el tipo de prueba que se está ejecutando y el impacto que tiene para el funcionamiento correcto del sistema según lo establecido en los requerimientos del sistema.

Entre los criterios de aprobación se encuentran:

1. El resultado obtenido debe cumplir con los criterios de aceptación para cada caso de prueba.

2. Se deben de cumplir las precondiciones establecidas para los casos de prueba.
3. Se deben de seguir los pasos de ejecución para las pruebas.
4. Debe haber constancia en forma de evidencia de los resultados de las pruebas.

2.10. Criterios de suspensión y requisitos de reanudación

Los criterios de suspensión y de reanudación para las pruebas muestran las condiciones necesarias o validan las razones por el que la ejecución de un caso de prueba planteado en el plan de pruebas se suspendió o en dado caso después de su suspensión, el equipo evaluador reanudó la ejecución del caso de prueba.

Entre los criterios de suspensión se encuentran:

1. El caso de prueba planteado no aporta ningún valor real a mejorar la calidad del sistema en estudio o en dado caso su valor ha sido obtenido por otra prueba.

2. El asesor descalifica el caso de prueba.

3. El sistema sufrió cambios durante la ejecución del plan de pruebas y el caso de prueba ya no es válido.

2.11. Entregables de prueba

1. Matriz de prueba.

2. Plan de pruebas.

3. Reporte de prueba.

4. Informe de defectos encontrados.

5. Resumen de riesgos.

Los reportes e informes se generarán al finalizar la ejecución de todas las pruebas que se definan en el plan y matriz de pruebas. Una vez se hayan completado y cubierto de los casos de pruebas especificados, se hará un consolidado y se analizaran los resultados para brindar recomendaciones finales.

Los entregables se enviarán a las partes interesadas para visualizar el estado de las pruebas y del sistema, así como la calidad que este posee. El objetivo es que obtengan una visión clara de la situación actual y de los riesgos o posibles áreas de mejora antes de tomar decisiones futuras.

2.12. Tareas de pruebas restantes

Las tareas de pruebas restantes que se han identificado durante el desarrollo de la investigación y que no se han tomado en cuenta son:

a. **Pruebas unitarias:** estas pruebas se han omitido principalmente porque estas pruebas no se desarrollan durante la realización de pruebas de calidad, sino que se deben de desarrollar durante

el ciclo de desarrollo del sistema, es decir debe ser realizada por los programadores encargados. Al no existir un antecedente de estas pruebas en la documentación compartida por el negocio se determina que debe existir una revisión más exhaustiva del código en base a los riesgos detectados en este trabajo.

2.13. Ambientes

Los requisitos necesarios para la ejecución de los casos de prueba se dividen en requisitos de hardware y software. A continuación, se describen los elementos mínimos requeridos para el ambiente de pruebas utilizado en este proyecto, el cual es el único ambiente disponible, dado que el sistema ya fue desarrollado y no se cuenta con un ambiente de desarrollo.

Requisitos ambientales de hardware:

- Computadora con acceso estable a Internet.
- Procesador de 4 núcleos con arquitectura de 64 bits.
- Mínimo de 8 GB de memoria RAM.

Requisitos ambientales de software:

- Código fuente del sistema en su última versión disponible.
- Sistema gestor de bases de datos: MySQL 8.0.
- PHP versión 7.4.
- Node.js versión 14.0 o superior.
- Sistema operativo: Windows 10, Windows 11 o Ubuntu 20.04 o superior.
- Navegador web compatible con motor Chromium (por ejemplo, Google Chrome, Microsoft Edge) o Gecko (Mozilla Firefox).
- Microsoft Office (Word, Excel y PowerPoint), usado para análisis, generación de reportes y documentación de evidencias.

El cumplimiento de estos requisitos garantiza que las pruebas se ejecuten en condiciones similares a las del entorno de producción, permitiendo la identificación confiable de defectos y la recolección de resultados representativos. La correcta documentación y preparación del ambiente es fundamental para lograr que los resultados sean reproducibles y útiles para posteriores procesos de validación de calidad.

2.14. Equipo de trabajo y capacitación

Para la ejecución efectiva de las pruebas del sistema, el equipo de trabajo debe estar conformado por roles especializados que aseguren un proceso ordenado, eficiente y preciso. En un entorno de pruebas,

las responsabilidades principales incluyen el diseño, ejecución, análisis de resultados y documentación de las pruebas realizadas.

Roles y responsabilidades principales:

- Analista de Pruebas de Software QA.
- Líder de Pruebas QA.

Tabla 7
Roles del equipo.

Nombre	**Rol**
Oscar Leonardo Serpas Martínez	Líder de Pruebas QA
David José Castro Clímaco	Analista de Pruebas de Software QA
Marcelo Josué Alcántara Alas	Analista de Pruebas de Software QA

Nota. Roles de los diferentes miembros del equipo.

2.14.1. Capacitación Requerida.

Es recomendable que todos los integrantes del equipo de pruebas tengan conocimientos en:

- Técnicas de pruebas de cajas negras y blancas.
- Herramientas de automatización de pruebas compatibles con el sistema (tal como Laravel Dusk).
- Normas internacionales de calidad del software, especialmente la ISO/IEC 25010.
- Uso y configuración de los ambientes de pruebas, incluyendo hardware, software, bases de datos y navegadores compatibles.
- Gestión de defectos, trazabilidad y reportes técnicos.

Capacitarse en estas áreas garantiza la competencia para identificar defectos, evaluar la calidad del sistema y administrar de forma eficiente los ambientes y recursos técnicos.

2.15. Responsabilidades

Tabla 8
Responsabilidades.

Rol	**Descripción**	**Responsabilidades**
Líder de QA	Coordina las actividades del equipo,	1. Planificación de pruebas.

Rol	Descripción	Responsabilidades
	supervisa los cronogramas, asigna tareas específicas y garantiza que cada fase del proceso de pruebas se complete según los estándares y requisitos establecidos.	2. Revisión de resultados. 3. Diseño de pruebas. 4. Monitoreo y control de pruebas. 5. Reporte final del plan de pruebas.
Analista de Pruebas de Software QA	Responsable de definir y ejecutar planes de prueba, crear casos de prueba, automatizar procesos y reportar defectos. Debe contar con conocimientos en metodologías de pruebas, herramientas de automatización y normas de calidad del software.	1. Diseño de pruebas. 2. Implementación de pruebas. 3. Ejecución de pruebas. 4. Reporte de pruebas.

Nota. Tabla donde se describen las responsabilidades de todos los miembros del equipo.

2.16. Calendario

Tabla 9
Calendario.

N°	Actividades	Fecha de inicio	Fecha de finalización
1	Propuesta del tema de tesina	02/06/2025	23/06/2025
2	Diagnóstico y diseño del proyecto	24/06/2025	14/10/2025
3	Defensa del diagnóstico y diseño del proyecto	21/10/2025	24/10/2025
4	Proyecto final	24/10/2025	28/11/2025
5	Predefensa	28/11/2025	28/11/2025
6	Defensa final	01/12/2025	01/12/2025

Nota. Tabla donde se detallan las fechas para el desarrollo de las diferentes actividades.

2.17. Plan de riesgos y contingencias

El plan de riesgos y contingencias tiene como objetivo identificar, evaluar y definir estrategias para mitigar posibles eventos que podrían afectar el desarrollo y ejecución del plan de pruebas y, en consecuencia, la calidad del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación. De esta manera, se garantiza que el proyecto pueda adaptarse y responder adecuadamente frente a riesgos que se presenten durante el proceso.

2.17.1. Identificación y análisis de riesgos

Los riesgos identificados se dividen en dos grandes grupos: riesgos de proyecto y riesgos de producto.

1. **Riesgos de producto:** vinculados a las características y calidad técnica del sistema bajo prueba.

2. **Riesgos de proyecto:** asociados al control, gestión y recursos disponibles para la ejecución de pruebas.

2.17.2. Evaluación del riesgo

Cada riesgo es evaluado considerando:

- **Probabilidad de ocurrencia:** estimación de la frecuencia con que puede suceder el evento.
- **Impacto:** consecuencias que tendría la materialización del riesgo sobre el proyecto o producto.
- **Nivel de riesgo:** resultado de la combinación de probabilidad e impacto, que permite priorizar esfuerzos.

2.17.3. Estrategias de mitigación

Para controlar y reducir los riesgos se implementan:

- Planificación detallada y realista de actividades y recursos.
- Capacitación continua del equipo en metodologías, herramientas y normas pertinentes.
- Uso de herramientas automatizadas para agilizar la ejecución y reducir errores humanos.
- Establecimiento de canales claros de comunicación y seguimiento del progreso.
- Priorización de pruebas en módulos críticos, garantizando cobertura de las funcionalidades esenciales.
- Documentación exhaustiva de hallazgos, facilitando la toma de decisiones basadas en evidencia clara.

2.17.4. Plan de contingencia

En caso de que se materialice un riesgo, se aplicarán las siguientes acciones:

Tabla 10
Plan de contingencia.

Tipo de plan	Riesgo	Acción de contingencia
Riesgo de producto	Funcionalidades incompletas o con errores críticos.	Aislar el módulo afectado, ejecutar pruebas de humo en rutas críticas, comunicar a interesados y activar ventana de hotfix.
Riesgo de proyecto	Deficiencias en	Utilizar pruebas exploratorias y validar con usuarios

Tipo de plan	Riesgo	Acción de contingencia
	documentación.	claves.
Riesgo de proyecto	Ambiente de pruebas limitado	Implementar ambiente UAT para la realización de pruebas.

Nota. Descripción del plan de contingencia para los diferentes riesgos encontrados.

2.17.5. Seguimiento y control

- Se documentará todos los incidentes, su análisis y acciones implementadas.
- El líder del proyecto mantendrá un registro actualizado de la matriz de riesgos.
- Se realizarán reuniones periódicas para revisión del estado y actualización de planes.
- Se buscará la mejora continua del proceso con base en las lecciones aprendidas y resultados obtenidos.

2.18. Aprobaciones

Esta sección define quiénes revisan y aprueban los artefactos del proceso de pruebas, así como los criterios y momentos en los que se emiten dichas aprobaciones. La aprobación no implica corrección de defectos por parte del equipo evaluador; su función es validar que los entregables cumplen con lo planificado y comunicar oficialmente los resultados y recomendaciones a la Secretaría de Investigaciones Científicas.

Entregables

1. Plan de pruebas del sistema (versión base y actualizaciones).
2. Matriz de pruebas con cobertura, resultados y evidencias consolidadas.
3. Reporte de pruebas final, que incluye resumen ejecutivo, métricas, defectos y riesgos identificados, y acciones de mejora recomendadas.
4. Carta de salida (cierre de pruebas) con el estatus de cumplimiento de criterios de entrada/salida y alcance ejecutado.

Roles que aprueban

1. **Asesor(a) de tesis*:*** valida metodología, alineación a ISO/IEC 25010 y consistencia académica de los entregables.
2. **Representante de la Secretaría de Investigaciones Científicas (CIC-UES):** recibe resultados, valida que el contenido es suficiente para la toma de decisiones y que refleja el estado del sistema en su contexto institucional.

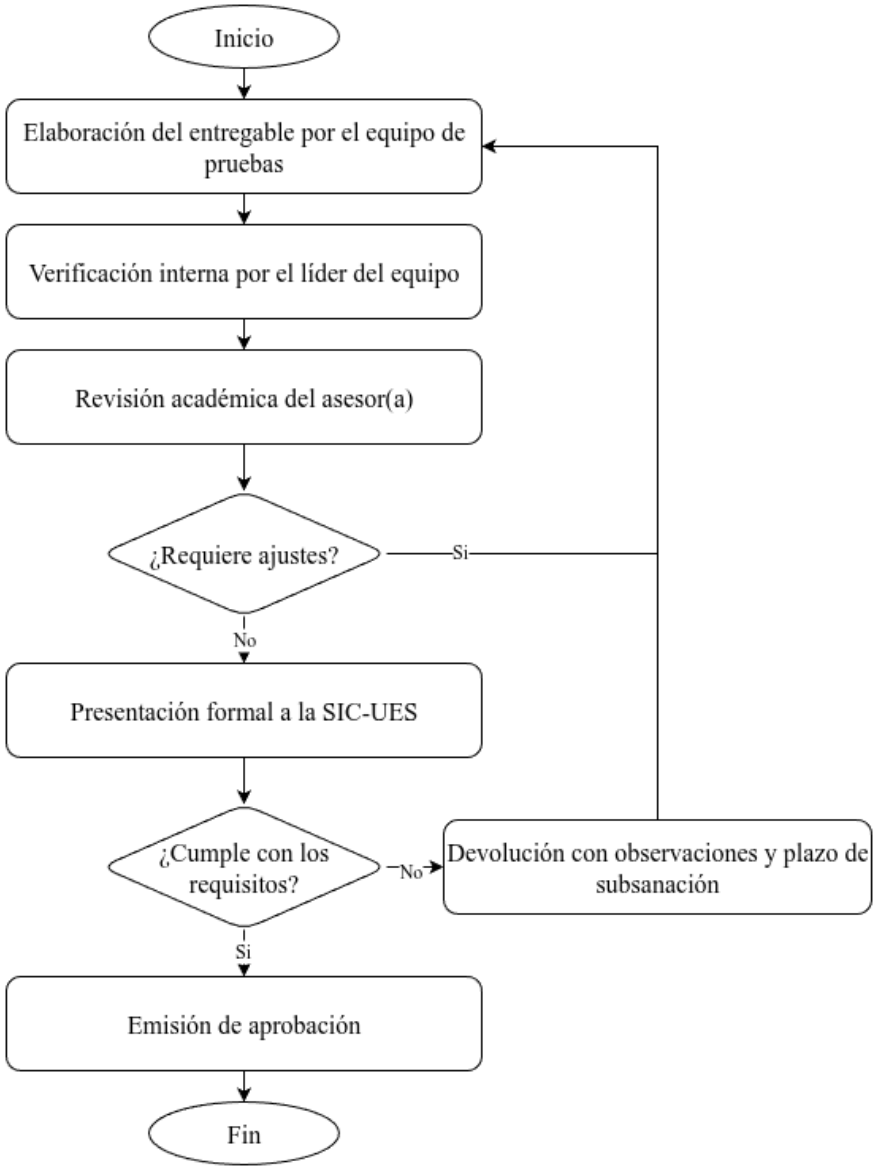
Criterios para aprobar

1. **Cobertura:** los casos ejecutados cubren los módulos y características priorizadas por riesgo y alcance definidos.

2. **Evidencia:** existen capturas, registros de ejecución y trazabilidad entre requisitos, casos de prueba y resultados en la matriz.
3. **Métricas y defectos:** se presentan métricas consolidadas y defectos clasificados con su severidad e impacto, junto a la evaluación de riesgos.
4. **Conclusiones y recomendaciones:** el reporte incluye conclusiones técnicas y acciones de mejora claras, sin intervenir el sistema, para decisión posterior de la SIC-UES.

Flujo de revisión y firma

Figura 2
Flujo de revisión y firma.



Nota: Diagrama donde se describe la ruta crítica desde la creación del entregable por el equipo técnico hasta la emisión de la resolución oficial por parte de la instancia universitaria.

Aprobación de la Carta de Salida

La Carta de Salida se emite cuando:

1. Se cumplieron los criterios de entrada/salida definidos y el alcance ejecutado corresponde al plan ajustado por atrasos en la planificación, riesgos y restricciones.
2. Los defectos críticos encontrados fueron documentados y comunicados; la corrección queda bajo responsabilidad de quienes determine la SIC-UES.
3. La matriz de pruebas y el reporte final quedan archivados y disponibles para auditoría y futuras mejoras.

Registro y trazabilidad

Todas las aprobaciones, observaciones y versiones de documentos quedarán registradas en el documento final del proyecto de tesina, garantizando trazabilidad entre versiones del plan, ejecuciones y resultados.

3. Capítulo 3. Caso de estudio

3.1. Antecedentes

La Secretaría de Investigaciones Científicas (SIC-UES) de la Universidad de El Salvador (UES) es la unidad orgánica encargada de coordinar los esfuerzos de la investigación científica que realiza la Universidad de El Salvador (SIC-UES, 2019). Su objetivo principal es la generación de nuevo conocimiento que no solo responda a las necesidades de la sociedad salvadoreña, sino que también facilite la obtención de recursos financieros para la investigación básica y aplicada. De esta manera, se busca posicionar a la UES como la institución de educación superior líder en investigación en el país. En su portal oficial, la SIC-UES establece su misión: "promover la cultura de investigación y la divulgación de la actividad científica desarrollada en la Universidad de El Salvador, cuyo resultado será la generación de nuevo conocimiento que facilite la obtención de recursos financieros de apoyo a la investigación básica y aplicada" (SIC-UES, 2022, párr. 1).

Para cumplir con esta misión, la SIC-UES interactúa con diversos actores clave. Según el análisis del sistema previo, estos incluyen: los usuarios directos del sistema que gestionan los proyectos, la Dirección de Investigación que coordina las políticas, las autoridades universitarias que toman decisiones estratégicas, los usuarios externos interesados en la ciencia UES, y los referentes de investigación de cada facultad, quienes son miembros del Consejo de Investigaciones Científicas (CIC-UES) y gestionan la información para la generación de indicadores de ciencia y tecnología (CIC-UES, 2016).

El camino hacia la digitalización de estos procesos no ha estado exento de desafíos. En el año 2021, se coordinó un esfuerzo para desarrollar dos sistemas informáticos, uno para el registro de proyectos y otro para indicadores de ciencia y tecnología, los cuales no pudieron ser finalizados. La necesidad de una herramienta centralizada era evidente, ya que permitiría concentrar "el flujo de autorización, los avances e informes técnicos, en un solo punto acorde a las normativas vigentes de la SIC-UES, de manera que se tendrá una fuente de información oficial confiable para la obtención de los reportes requeridos" (NTA, 2017). Es así como nace el actual "Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación", una aplicación web diseñada para optimizar la gestión de convocatorias, administrar los perfiles de investigadores y manejar integralmente los proyectos de investigación.

La necesidad de implementar sistemas de información para la gestión de la investigación es una tendencia recurrente en las principales casas de estudios de Centroamérica. En la Universidad de San Carlos de Guatemala, por ejemplo, Alvarado Maldonado (2022) desarrolló un sistema web para la gestión de proyectos de investigación, destacando que " a implementación de la plataforma web, se mejora de manera significativa el desempeño de los procesos de trabajos de investigación" (p. XI). De manera similar, en la Universidad de Costa Rica propuso un Sistema de Información y gestión de proyectos, programas y actividades, denominado SIGPRO; es una “aplicación en línea permite a los usuarios consultar de manera práctica la información relacionada con sus proyectos: revisar los datos de sus proyectos sobre presupuesto e información personal como becas, estudios, distinciones y capacitaciones y capacitaciones recibidas” (UCR, 2010). A nivel nacional, el sistema de información de la producción científica de la Universidad de El Salvador SIPC-UES puesto en producción en 2016 y retirado por obsolescencia tecnológica en 2019, sentó las bases metodológicas y técnicas para el sistema actual de la SIC-UES, demostrando la viabilidad y la necesidad de una solución robusta y adaptada a la realidad de la Universidad de El Salvador.

Aunque estos trabajos previos demuestran la importancia y el esfuerzo por digitalizar la gestión de la investigación, el presente trabajo se distingue al no centrarse en la creación de un nuevo sistema, sino en el análisis e implementación de pruebas de calidad sobre uno ya existente. La calidad del software es fundamental para garantizar su fiabilidad, usabilidad y eficiencia. Por ello, esta tesina propone aplicar la norma ISO/IEC 25010, que define un modelo de calidad estandarizado a nivel internacional, para evaluar sistemáticamente el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la SIC-UES, asegurando que esta herramienta cumpla con los estándares necesarios para respaldar eficazmente la labor científica de la Universidad de El Salvador.

3.2. Contexto del problema

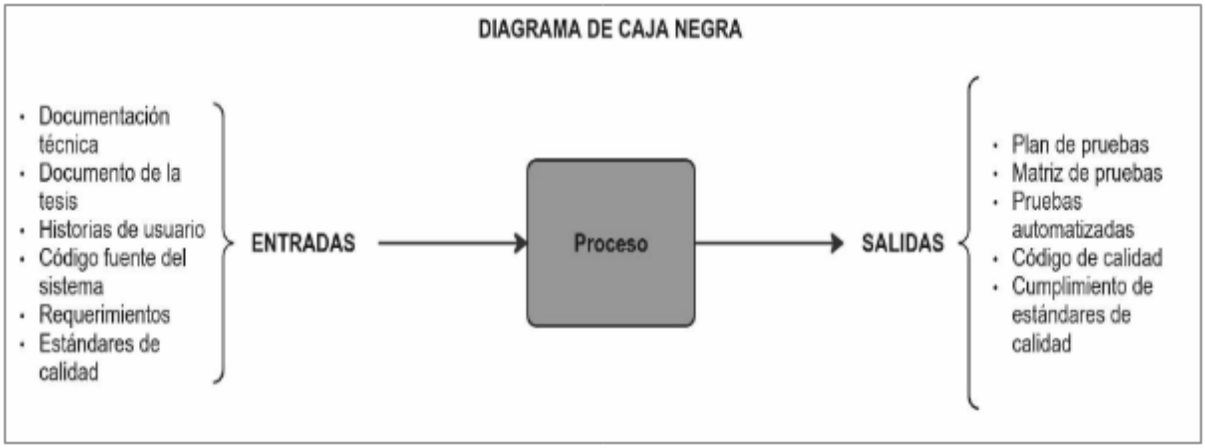
El Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador fue desarrollado con el objetivo de mejorar la eficiencia y transparencia en los procesos administrativos vinculados a las convocatorias de investigación, el registro de proyectos y la gestión de recursos asociados, así como generar una adecuada administración de los perfiles de los investigadores.

Como parte del proceso de análisis y evaluación de calidad del sistema de registro de la Secretaría de Investigaciones Científicas, se elaboró una matriz FODA y un diagrama de Ishikawa (disponibles en el Anexo A) con el propósito de identificar factores internos y externos que inciden en el desarrollo del análisis técnico. Esta herramienta permitió estructurar estratégicamente las condiciones del entorno y del equipo evaluador.

Sin embargo, el sistema al no contar con un proceso de pruebas de calidad desarrollado y aplicado puede verse afectada directamente su operatividad y la satisfacción de los usuarios. Estos problemas, si no se detectan antes de la implementación formal y definitiva del sistema, pueden incluir errores en módulos críticos, como el registro de proyectos, la validación de datos y la generación de reportes, generando inconsistencias en la información y la toma de decisiones. Aunque el sistema está en fase previa a la implementación definitiva, existe preocupación por la aparición de fallos frecuentes, lentitud en la carga de datos y dificultades para realizar operaciones. Como resultado, puede existir desconfianza en el uso del sistema.

Figura 3

Diagrama de caja negra



Este problema se ha generado debido a la falta de una evaluación formal y detallada del desempeño de la calidad del sistema desde su análisis y desarrollo. Aunque se inició en una fase parcial de implementación de los módulos del sistema, no se ha validado un proceso que permita identificar y corregir los defectos existentes. La relevancia de esta situación radica en que el sistema gestiona información clave sobre los proyectos de investigación de la universidad y a su vez los componentes económicos y detalles específicos de investigaciones a desarrollar, lo cual impacta de manera directa en la eficiencia administrativa, la transparencia y la calidad de la gestión académica. Si el sistema se implementa con defectos no detectados podría generar problemas en la toma de decisiones importantes, afectando la confianza de los usuarios y obstaculizando el funcionamiento adecuado de la Secretaría. Estos problemas pueden estar relacionados con una planificación insuficiente en las etapas iniciales del proyecto, falta de pruebas exhaustivas de calidad y deficiencias en el diseño del sistema. Si estos problemas no se identifican para posteriormente resolverlos, el sistema podría seguir siendo ineficiente, lo que llevaría a su bajo nivel de adopción y, eventualmente, a la necesidad de rediseñarlo o abandonarlo, lo que resultaría en una pérdida considerable de recursos. Para evitar que la situación empeore, se requiere evaluar la calidad del sistema con un plan de pruebas basado en la norma ISO/IEC 25010, que permitirá identificar áreas de mejora y ofrecer recomendaciones para optimizar el sistema, garantizando que cumpla con los estándares de calidad necesarios.

3.3. Modelado de negocio

La Secretaría de Investigaciones Científicas (SIC-UES) es el motor que impulsa y coordina la investigación en la Universidad de El Salvador. Su rol no es solo administrativo, sino estratégico: promueve la cultura de investigación, da seguimiento a los proyectos en las seis áreas del conocimiento definidas en el manual de Frascati (OECD, 2015) y asegura que los resultados tengan un impacto real. Para lograrlo, la SIC-UES gestiona un ciclo de vida completo para los proyectos, que va desde la idea inicial hasta la difusión de los resultados.

Este ciclo de vida o proceso de negocio central se puede desglosar en las siguientes etapas clave:

1. La SIC-UES define las prioridades de investigación, diseña las convocatorias y las publica para que la comunidad universitaria presente sus propuestas.
2. Los investigadores preparan individual o en grupo para presentar sus protocolos o propuestas de investigación a través de los canales oficiales.
3. Un comité de expertos, en este caso el Consejo Ejecutivo de Investigaciones (CEI), revisa y evalúa las propuestas basándose en criterios científicos, técnicos y de relevancia.
4. Una vez aprobado un proyecto, la SIC-UES tramita los apoyos financieros y las adquisiciones necesarias. Además, da seguimiento al avance del proyecto, asegurando que se cumplan los hitos y la entrega de informes.
5. Al finalizar, se verifica la correcta ejecución del proyecto y se gestionan los mecanismos para difundir los resultados, como publicaciones o ponencias, manteniendo la trazabilidad institucional.

Para que todas estas etapas funcionen de manera eficiente, transparente y sin errores, la SIC-UES implementó el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación. Este sistema informático es la herramienta tecnológica diseñada para apoyar el flujo de negocio, su propósito es optimizar cada una de las etapas mencionadas: la gestión de las convocatorias, el registro de los protocolos,

la evaluación por parte del CEI, la emisión de resoluciones, la administración de los apoyos, el seguimiento del progreso y la generación de reportes institucionales para el cierre.

Por ello, la esencia e importancia de esta tesina: la calidad del sistema informático es directamente proporcional a la calidad de la gestión de la investigación. Un error en el registro de una propuesta, un fallo en el módulo de evaluación o un problema en la generación de un reporte pueden tener consecuencias serias, como la pérdida de una investigación relevante o una mala toma de decisiones.

Es por eso que no basta con tener un sistema; hay que asegurarse de que funcione de manera confiable, segura y usable, de acuerdo a estándares internacionales reconocidos. Por lo tanto, el presente trabajo se enfoca en analizar este sistema a través de la lente de la norma ISO/IEC 25010, evaluando sistemáticamente si la herramienta tecnológica cumple con los requisitos de calidad necesarios para soportar de manera efectiva el crítico proceso de negocio de la Secretaría de Investigaciones Científicas de la Universidad de El Salvador.

Actores principales:

1. Investigadores y equipos de proyecto (postulan, responden observaciones y ejecutan).
2. Secretaría/Gestora SIC (verifica requisitos, canaliza a CEI y notifica).
3. CEI/CIC-UES (genera observaciones, dictamina y aprueba/rechaza; define política y presupuesto).
4. Unidades administrativas (adquisiciones y logística para proyectos aprobados).

Procesos clave:

1. Gestión de convocatorias (bases, calendario, documentos, recepción).
2. Evaluación y aprobación (observaciones CEI, expertos, resolución).
3. Gestión de apoyos y adquisiciones (sólo si hay presupuesto aprobado).
4. Seguimiento y cierre (monitoreo, evidencia de resultados y difusión).

Reglas esenciales:

1. Sin subsanar observaciones, el protocolo no pasa a aprobación final.
2. La ejecución de apoyos ocurre únicamente tras aprobación CEI y trámite formal de la SIC.
3. Los proyectos se alinean a programas/líneas institucionales vigentes.

3.3.1. Definición De Conceptos Clave

1. **Convocatoria:** Conjunto de bases, requisitos y calendario para postular a financiamiento de proyectos institucionales en períodos específicos.
2. **Protocolo:** Documento de solicitud estructurado con objetivos, metodología, presupuesto y cronograma de actividades exigidos por la SIC-UES.

3. **Observaciones preliminares CEI:** Devolución inicial con requisitos formales/técnicos a corregir antes del dictamen de expertos.
4. **Dictamen de expertos:** Evaluación especializada que sustenta la decisión de aprobación/rechazo por el CEI/CIC-UES.
5. **Resolución CEI:** Decisión formal de aprobar/rechazar y, en su caso, el presupuesto autorizado.
6. **Adquisiciones:** Gestión administrativa de bienes/servicios aprobados para la ejecución del proyecto.
7. **Política de investigación y agendas de investigación:** Marcos institucionales para clasificar y priorizar proyectos por áreas estratégicas.
8. **Seguimiento:** Actividades de control de avance, cumplimiento de objetivos y uso de recursos, con registro de evidencias y reportes.
9. **Archivos de proyectos:** Conjunto de proyectos vigentes/aprobados con su trazabilidad y estado para gobierno y difusión.

3.4. Necesidades Del Negocio

1. **Eficiencia en convocatorias:** Publicar bases y documentos oficiales, administrar calendarios y recibir postulaciones con validaciones formales para reducir retrabajos.
2. **Trazabilidad integral:** Se garantiza la trazabilidad entre la convocatoria, el protocolo versionado, las observaciones, los dictámenes, la resolución, los apoyos gestionados, el seguimiento del proyecto y el cierre con difusión de resultados.
3. **Transparencia y gobernanza:** Métricas e informes para CEI/SIC/CIC que respalden decisiones y política institucional (tiempos de evaluación, tasas de aprobación, ejecución de apoyos).
4. **Control de cumplimiento:** Flujos que impidan avanzar sin requisitos cumplidos y registren quién/qué/ cuándo en cada hito.
5. **Gestión de apoyos:** Módulos y registros que faciliten el trámite de adquisiciones/servicios conforme a presupuesto aprobado por el CEI.
6. **Seguridad y confidencialidad:** Manejo de datos de protocolos y dictámenes con controles de acceso y resguardo institucional.
7. **Experiencia del investigador:** Formularios claros, validaciones amigables, notificaciones oportunas y visibilidad del estado de su postulación.
8. **Evidencia y difusión:** Registro de productos/resultados para comunicación institucional y reportes públicos cuando aplique.

3.5. Matriz de pruebas del trabajo Análisis e implementación de pruebas al Sistema Informático para la Gestión de Convocatorias y Registro de Proyectos De Investigación para la Secretaría de Investigaciones Científicas de la Universidad de El Salvador según la Norma ISO/IEC 25010

Para obtener una cobertura correcta bajo la norma ISO/IEC 25010, se diseñaron un total de 78 casos de prueba. Debido a la naturaleza del proyecto se aplicó una metodología de evaluación para dividir estos casos en dos estrategias de ejecución: Automatizada y Manual.

Es importante mencionar que el criterio de selección o en otras palabras la decisión de cuáles casos automatizar no fue arbitraria. Se realizó un análisis de viabilidad técnica y de riesgo detallado (Anexo C).

En resumen, podemos mencionar los siguientes puntos que también son desarrollados en los siguientes párrafos.

1. **Automatización (Laravel Dusk):** se priorizaron los flujos críticos de negocio (Smoke Tests) y las pruebas de regresión que deben ejecutarse repetidamente para asegurar la estabilidad del sistema.
2. **Manual:** se mantuvieron manuales aquellos casos que requieren validación visual humana, interacción compleja con hardware, o flujos de un solo uso (registro inicial complejo) donde el retorno de inversión de la automatización era bajo.

3.5.1. Casos automatizar

La automatización de pruebas permite una mayor eficiencia y eficacia en la gestión y ejecución de casos de pruebas dentro del sistema. Para la automatización se utiliza la herramienta de Laravel Dusk que nos permite definir los casos de prueba, configurarlos y ejecutarlos para su automatización.

Los casos de pruebas que van a automatizar son los siguientes:

Tabla 11
Casos a automatizar.

ID Caso de Prueba	Caso de prueba
CP-01	Validar el Inicio de Sesión en el sistema, teniendo en consideración los siguientes componentes: 1. Login usuario válido. 2. Login con contraseña incorrecta. 3. Login campos vacíos. 4. Login con usuario no registrado. 5. Usuario ya autenticado. 6. Acceso al login cuando el usuario ya está autenticado.
CP-03	Validar la Seguridad y Autenticación en el sistema, teniendo en consideración los siguientes componentes: 1. Límite de intentos fallidos. 2. Inicio de sesión simultáneo del mismo usuario en múltiples dispositivos. 3. Verificar que las cookies de sesión tengan los atributos de seguridad HttpOnly y Secure correctamente configurados. 4. Validar que las sesiones expiren automáticamente tras cierre de navegador, inactividad (timeout) o acción de cierre de sesión (logout).
CP-05	Validar la Seguridad en el acceso y navegación del sistema, teniendo en consideración los siguientes componentes: 1. Acceso a rutas web sin autenticación debe redirigir automáticamente a la página de login. 2. Acceso a rutas web con sesión autenticada debe permitir el ingreso sin restricciones. 3. Acceso a rutas protegidas con sesión expirada debe redirigir al login o mostrar mensaje de expiración. 4. La estructura del sistema debe forzar la navegación segura mediante

ID Caso de Prueba	Caso de prueba
	HTTPS en todas las rutas.
CP-06	Validar la Seguridad en el acceso y navegación del sistema, teniendo en consideración los siguientes componentes: 1. Acceso a rutas API sin autenticación debe devolver el código de estado HTTP 401 (Unauthorized) 2. Middleware signed debe validar que los enlaces firmados sólo sean válidos para la ruta "verify"
CP-08	Cargar currículum muestra datasets requeridos.
CP-10	Editar currículum carga vista con datos relacionados.
CP-12	Actualizar datos personales en perfil del investigador: sin imagen válida y con imagen inválida.
CP-14	Eliminar con éxito los datos correspondientes al currículum del perfil Investigador. - Nota: Los Datos Personales con imagen no pueden ser borrados una vez se han ingresado por primera vez. Los registros que se pueden borrar pertenecen a: - Docencias. - Investigaciones. - Financiamiento de investigación. - Proyección social. - Reconocimiento.
CP-16	Registrar pasos candidatos como completados en Datos personales vista Curriculum, Finalizar secciones correspondientes a Datos personales, Docencia, Investigaciones, Financiamiento de investigación, Proyección social, Reconocimiento y completar proceso al 100%
CP-18	Editar formación académica en perfil del investigador (marcando o no "nivel máximo").
CP-20	Abrir sección de experiencia científica.
CP-22	Ver datos precargados al abrir edición con datos precargados en publicaciones y patentes en experiencia científica.
CP-24	Eliminar publicación científica y patente desde la vista experiencia científica.
CP-26	Registrar red de investigación (éxito).
CP-28	Eliminar red de investigación.
CP-30	Registrar competencia informática o de idioma desde el formulario u opción "Otras competencias".

ID Caso de Prueba	Caso de prueba
CP-32	Eliminar habilidad informática o idioma desde otras competencias.
CP-34	Eliminar proyecto sin financiamientos asociados.
CP-36	Abrir “Mis proyectos de investigación”, crear proyecto éxito y visualizarlo en lista.
CP-38	Editar título de proyecto e ingresar objetivos, actividades y colaboradores en proyecto de investigación.
CP-40	Eliminar objetivos, actividades y colaboradores en proyecto de investigación.
CP-42	Registrar presupuesto: fuente de financiamiento, recurso, contratación, viáticos nacionales e internacionales y publicaciones.
CP-44	Eliminar registros de Presupuesto: fuente, recurso, contratación, viático nacional, viático internacional y publicación.
CP-46	Bloqueo por dependencias al eliminar proyecto muestra mensaje específico.
CP-48	Actualizar proyecto y marcar pasó “título” completado.
CP-50	Ver resumen de progreso con pasos existentes.
CP-52	Crear pasos por defecto al abrir resumen sin registros.
CP-54	Descargar plantilla/resumen de Protocolo desde el proyecto.
CP-56	Paginación en Perfiles enviados: ver segunda/última página.
CP-58	Registrar archivado guarda datos y relaciones.
CP-60	Reportes de Colaboradores: filtrar, paginar y descargar PDF/Excel.
CP-62	Estadísticas: cargar y actualizar gráficos por módulo (facultad e investigadores).
CP-64	Crear recurso de inventario y verificar redirección y visualización en la lista.
CP–66	Eliminar recurso de inventario desde la lista.
CP-68	"Validar la creación y notificación de convocatorias en el sistema, teniendo en consideración los siguientes componentes: 1. Crear convocatoria inactiva. 2. Crear convocatoria activa cuando ya existe una activa. 3. Notificar convocatoria activa".
CP-70	"Validar la eliminación de convocatorias en el sistema, teniendo en consideración los siguientes componentes: 1. Eliminar convocatoria sin proyectos asociados.

ID Caso de Prueba	Caso de prueba
	2. Intentar eliminar convocatoria con proyectos asociados".

Nota. Tabla donde se describen los casos de prueba que se automatizarán.

3.5.2. Casos que no se van a automatizar.

Los casos de pruebas que no se van a automatizar son el resultado del no cumplimiento de los criterios necesarios según la evaluación realizada para la selección de casos de prueba que se automatizarán (Véase Anexo B). Sin embargo, las pruebas se realizarán de forma manual en donde se documentará todo el proceso correspondiente al proceso de ejecución de pruebas.

Los casos de prueba que no se van a automatizar son los siguientes:

Tabla 12

Casos que no se van a automatizar.

ID Caso de Prueba	Caso de Prueba
CP-02	Validar el Registro de Usuario en el sistema, teniendo en consideración los siguientes componentes: 1. Registro de usuario válido. 2. Registro con email existente. 3. Registro con nombre vacío. 4. Registro con contraseña corta. 5. Registro con confirmación de contraseña diferente. 6. Acceso al formulario de registro cuando el usuario ya está autenticado.
CP-04	Validar la Seguridad en el proceso de Registro de Usuario, teniendo en consideración los siguientes componentes: 1. Registro con campos múltiples vacíos. 2. Registro con intento de inyección de código malicioso (XSS). 3. Registro con intento de inyección SQL. 4. Registro duplicado en ráfaga o múltiples intentos consecutivos con los mismos datos.
CP-07	Validar el comportamiento del sistema tras la confirmación de contraseña, teniendo en consideración los siguientes componentes: 1. Confirmación de contraseña correcta debe mantener la sesión activa del usuario sin requerir reauteticación adicional. 2. Redirección automática a la página originalmente solicitada (intended) tras confirmar la contraseña correctamente.
CP-09	Ingresar los datos correspondientes a los formularios que componen el currículum del perfil Investigador, este se compone de - Datos Personales con imagen - Docencias

ID Caso de Prueba	Caso de Prueba
	- Investigaciones - Financiamiento de investigación - Proyección social - Reconocimiento
CP-11	Editar con éxito los datos correspondientes a los formularios que componen el currículum del perfil Investigador, este se compone de - Datos Personales con imagen. - Docencias. - Investigaciones. - Financiamiento de investigación. - Proyección social. - Reconocimiento.
CP-13	Validar la gestión completa de datos personales en el sistema, teniendo en consideración los siguientes componentes: 1. Guardar datos personales correctamente. 2. Validar que el campo obligatorio “Cargo actual” impida guardar si está vacío. 3. Subir foto de perfil válida (formato permitido, tamaño adecuado). 4. Subir foto de perfil inválida (formato no permitido o tamaño excedido).
CP-15	Acceso y carga correcta al formulario de “Formación académica” (acceso normal y por enlace directo).
CP-17	Crear formación académica exitosamente de 3 diferentes formas con sus datos válidos: 1. Crear formación académica marcada como nivel máximo (sí). 2. Crear formación académica con nivel máximo en “no”. 3. Crear formación académica sin marcar nivel máximo.
CP-19	Eliminar formación académica existente.
CP-21	Registrar satisfactoriamente una publicación científica y una patente en la experiencia científica.
CP-23	Editar publicación científica y patente desde experiencia científica.
CP-25	Abrir Red de investigadores con listas vacías.
CP-27	Edición exitosa de red de investigación mostrando datos precargados y proyectos disponibles.
CP-29	Abrir “Otras competencias” con datos disponibles.
CP-31	Editar habilidad informática o idioma: cargar, modificar y actualizar.
CP-33	Gestión de curriculum por administrador: visualización, observación y

ID Caso de Prueba	Caso de Prueba
	aprobación de perfiles.
CP-35	Eliminar financiamiento asociado a un proyecto.
CP-37	Crear proyecto Error Controlado Intentar crear un proyecto con datos válidos sin convocatoria activa y validar campos obligatorios.
CP-39	Editar objetivos, actividades y colaboradores en proyecto de investigación.
CP-41	Intento de crear proyecto con datos inválidos muestra error.
CP-43	Editar presupuesto: fuente de financiamiento, recurso, contratación, viáticos (nac/intl) y publicaciones.
CP-45	Eliminar proyecto desde Mis proyectos con confirmación visible.
CP-47	Abrir confirmación de eliminación de proyecto.
CP-49	Intento de abrir detalle inexistente maneja el error.
CP-51	Avance de progreso con mensajes de éxito al finalizar secciones.
CP-53	Ver resumen con el registro completado al 100%.
CP-55	Ver lista de enviados mostrando solo estado “enviado”.
CP-57	Crear archivado: carga de formulario con selects visibles y comportamiento con catálogos poblados o vacíos.
CP-59	Reportes de Proyectos: filtrar, paginar y descargar PDF/Excel.
CP-61	Reporte de colaboradores sin filtros.
CP-63	Estadísticas: cargar y actualizar gráficos por módulo (financiamientos y convocatorias).
CP-65	Editar recurso de inventario y reflejar cambios en la lista.
CP-67	Ver detalle del recurso de inventario.
CP-69	Validar la edición de convocatorias en el sistema, teniendo en consideración los siguientes componentes: 1. Editar convocatoria y cambiar a activa sin existir otra activa 2. Editar convocatoria y cambiar a activa cuando ya existe una activa.
CP-71	Render y navegación en Chrome, Firefox y Edge.
CP-72	Responsive en iOS (Safari) y Android (Chrome).
CP-73	Consistencia en resoluciones 1366x768, 1920x1080 y 2560x1440.

ID Caso de Prueba	Caso de Prueba
CP-74	Descarga de Excel y PDF en navegadores principales.
CP-75	Subida de protocolo y documentos en distintos navegadores.
CP-76	Inputs, selects y validaciones consistentes entre navegadores.
CP-77	Rendimiento percibido en escritorio y móvil.
CP-78	Accesibilidad básica compatible entre navegadores.

Nota. Tabla donde se describen los casos de prueba que no se automatizarán.

3.6. Id y objetivos de las pruebas

En un Plan de Pruebas se deben de identificar todos los casos de prueba planteados para su ejecución y se debe de definir el objetivo del caso de prueba para determinar si las pruebas son aprobadas exitosamente o han fallado y se deben de hacer las correcciones correspondientes.

Para la identificación de los casos de prueba se ha utilizado el siguiente formato: **CP-XX.** El significado de cada letra se da a continuación:

- C: Caso.
- P: Prueba.
- XX: Es un número correlativo al caso de prueba.

Los objetivos de los casos de prueba registrados en la matriz de prueba se dividen según el tipo principal y el módulo del sistema en donde se realizará la prueba. Para los tipos de prueba realizadas se hace la siguiente división:

1. **Pruebas de seguridad:** las pruebas de seguridad se enfocan en verificar la capacidad del sistema para cumplir con los requerimientos de protección del propio sistema y a sus usuarios del acceso, uso, divulgación, interrupción o acceso no autorizado, en cumplimiento con los criterios de calidad de la Norma ISO/IEC 25010.
2. **Pruebas del sistema:** el objetivo de estas pruebas es comprobar el funcionamiento correcto de los componentes del sistema en evaluación, verificando que los resultados obtenidos sean los previstos en los requerimientos y comprobando la capacidad del sistema para proporcionar satisfacer las necesidades declaradas e implícitas de los usuarios cuando el producto se usa en las condiciones especificadas.
3. **Pruebas de compatibilidad:** tienen como objetivo verificar la compatibilidad del sistema para el uso en diferentes navegadores web, resoluciones de pantallas y en diferentes sistemas operativos. También se comprueba la capacidad del sistema para intercambiar información con los usuarios para realizar sus funciones requeridas cuando sean requeridas.

3.7. Alcance de pruebas

Los casos de pruebas que se han diseñado para el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación están basados en los componentes o módulos principales del sistema, la parte más crítica del sistema principalmente debido a la limitación de recursos y de personal para la realización y registro de las pruebas.

Las pruebas realizadas están orientadas a la Norma ISO/IEC 25010 y sus criterios de calidad (principalmente en las características de Adecuación Funcional, Seguridad y Capacidad de Interacción). Estás pruebas están enfocadas en dos aspectos principales:

1. Evaluación de la funcionalidad clave del sistema y el cumplimiento de los requisitos definidos para los módulos y funcionalidades críticas del sistema.
2. La ejecución de las pruebas se realiza en base a la importancia y su impacto en el funcionamiento correcto del sistema, teniendo en cuenta el riesgo que representa para la integridad del sistema.

Los módulos más importantes del sistema son los siguientes:

- Módulo de perfil del investigador.
- Módulo de proyectos de investigación.
- Módulo de usuarios y permisos.
- Módulo de convocatorias.
- Módulo de reportes.

En estos módulos se han realizado la mayoría de pruebas para comprobar su funcionalidad y el cumplimiento de los requisitos del sistema y los criterios de calidad de la Norma ISO/IEC 25010.

3.8. Estrategia de pruebas

Las estrategias para las pruebas implican el uso de diferentes herramientas para el registro y gestión de las pruebas, además del seguimiento de un procedimiento para la ejecución de pruebas. La elección de las herramientas y metodología a utilizar es esencial para lograr los objetivos planteados en la evaluación del sistema. Debido a su importancia para la selección de las herramientas se tomaron en consideración varias alternativas para las diferentes pruebas que se van a realizar y se seleccionaron las mejores alternativas según los requerimientos para el plan de pruebas.

Las herramientas seleccionadas para la realización de las pruebas fueron las siguientes:

Tabla 13
Herramientas utilizadas para las pruebas.

Herramienta	**Descripción**	**Función**
Laravel Dusk	Es una herramienta de automatización de pruebas en el navegador para aplicaciones Laravel. Permite realizar pruebas de integración end-to-end (E2E)	Pruebas funcionales

Herramienta	Descripción	Función
SonarQube	Es una plataforma de código abierto para la inspección continua de la calidad del código a través de diferentes herramientas de análisis estático de código fuente.	Análisis estático
Microsoft Excel	Es un programa de hoja de cálculo que se utiliza para organizar, analizar y visualizar datos mediante el uso de fórmulas, funciones, tablas y gráficos. Es una herramienta de Microsoft, parte de la suite Office.	Gestión de pruebas Dashboard de pruebas.
Microsoft Project	Es un software de administración de proyectos y programas de proyectos desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes.	Gestión de plan de trabajo

Nota. Descripción de las herramientas seleccionadas y su funcionalidad.

Las herramientas seleccionadas serán utilizadas para la gestión y ejecución de las pruebas de la siguiente forma:

1. El control de las actividades se realizará en Microsoft Project con tiempos específicos y la asignación de los recursos requeridos para la realización de las tareas y actividades.
2. Se utilizará SonarQube para la inspección del código fuente del sistema para encontrar posibles vulnerabilidades, código que no cumple con los estándares establecidos y posibles complejidades de módulos que se pueden reducir.
3. Laravel Dusk será utilizado para la realización de pruebas de UI y verificar el funcionamiento correcto de los componentes como usuario final.
4. La matriz de pruebas se registrará en Microsoft Excel en donde se colocarán los casos de pruebas y los resultados y evidencias obtenidas en la ejecución de pruebas.
5. En Microsoft Excel se creará una hoja específica para la visualización general de las pruebas en un dashboard, donde se encontrará un resumen de las pruebas con gráficos y la posibilidad de filtrar las pruebas por tipo y módulo, además también se encontrarán las métricas obtenidas durante la ejecución de las pruebas.

Diagrama de flujo del procedimiento para la ejecución de pruebas.

Para la ejecución de las pruebas se seguirá el procedimiento que se muestra a continuación a través de un diagrama para mejor visualización. Todas las pruebas seguirán este procedimiento según los pasos correspondientes que estén descritos en la matriz de pruebas para los casos de prueba:

Figura 4

Procedimiento para la ejecución de pruebas.

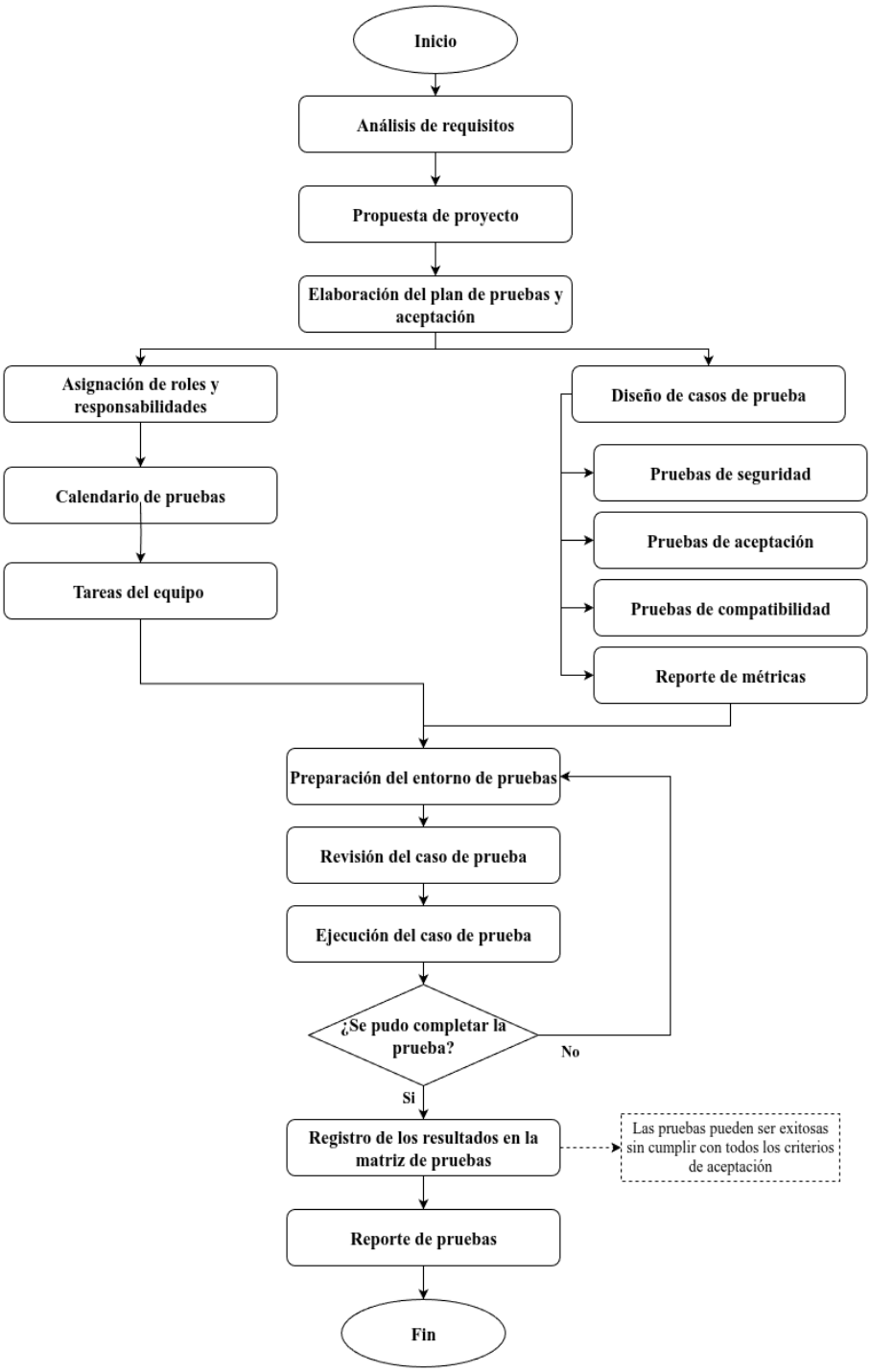


Tabla 14

Procedimiento para la ejecución de pruebas.

Actividad	**Descripción**	**Responsable**
Análisis de requisitos	Determinación de los requisitos necesarios para la elaboración de la propuesta de proyecto.	Oscar Serpas Marcelo Alcántara David Castro
Propuesta de proyecto	Presentación de la propuesta de proyecto para aceptación.	Oscar Serpas Marcelo Alcántara David Castro
Elaboración del plan de pruebas y aceptación	Elaboración del plan de pruebas para evaluar la calidad del sistema.	Oscar Serpas Marcelo Alcántara David Castro
Asignación de roles y responsabilidades	Determinación de roles y responsabilidades para los miembros del equipo.	Oscar Serpas
Diseño de casos de prueba	Análisis y diseño de pruebas para los componentes críticos del sistema.	Oscar Serpas Marcelo Alcántara David Castro
Calendario de pruebas	Determinación de las fechas para la ejecución de todas las pruebas diseñadas.	Oscar Serpas David Castro
Tareas del equipo	Actividades a realizar por cada miembro del equipo para el cumplimiento del proyecto.	Oscar Serpas Marcelo Alcántara
Preparación del entorno de pruebas	Configuración y preparación de un entorno adecuado para realizar las pruebas.	Oscar Serpas Marcelo Alćantara David Clímaco
Revisión del caso de prueba	Verificación del caso de pruebas y los pasos para realizar la prueba	Oscar Serpas Marcelo Alcántara
Ejecución del caso de pruebas	Se realizan las pruebas siguiendo los pasos detallados del caso de prueba.	Oscar Serpas Marcelo Alcántara David Castro
Registro de los resultados en la matriz de pruebas	Se colocará la evidencia y observaciones de los resultados obtenidos de la prueba.	Oscar Serpas David Clímaco

Nota. Tabla donde se detallan las actividades y responsables para la ejecución de pruebas.

3.9. Niveles de pruebas y fases

El plan se organiza en niveles progresivos y fases operativas para evaluar el Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación, priorizando módulos críticos como usuarios y permisos, convocatorias, proyectos de investigación, catálogos, inventario, reportes, estadísticas y revisión de perfiles. Los niveles son: pruebas de componentes, de integración de componentes, de sistema, de integración de sistemas y de aceptación, ejecutados en fases de planificación, pruebas estáticas, pruebas dinámicas por nivel y consolidación y reporte.

Objetivo por nivel:

1. **Componentes:** validar unidades y servicios internos con pruebas funcionales y, cuando sea viable, estructurales sobre lógica crítica de formularios y cálculos (p. ej., presupuesto y financiamiento del proyecto).

2. **Integración de componentes:** verificar flujos entre módulos como autenticación, gestión de proyectos, catálogos, notificaciones y reportes, incluyendo manejo de errores y consistencia de datos.

3. **Sistema:** evaluar extremo a extremo escenarios de negocio prioritarios (registro de proyectos, avance por pasos, envío, gestión de convocatorias, reportes e inventario), junto con aspectos de usabilidad, seguridad y rendimiento básico.

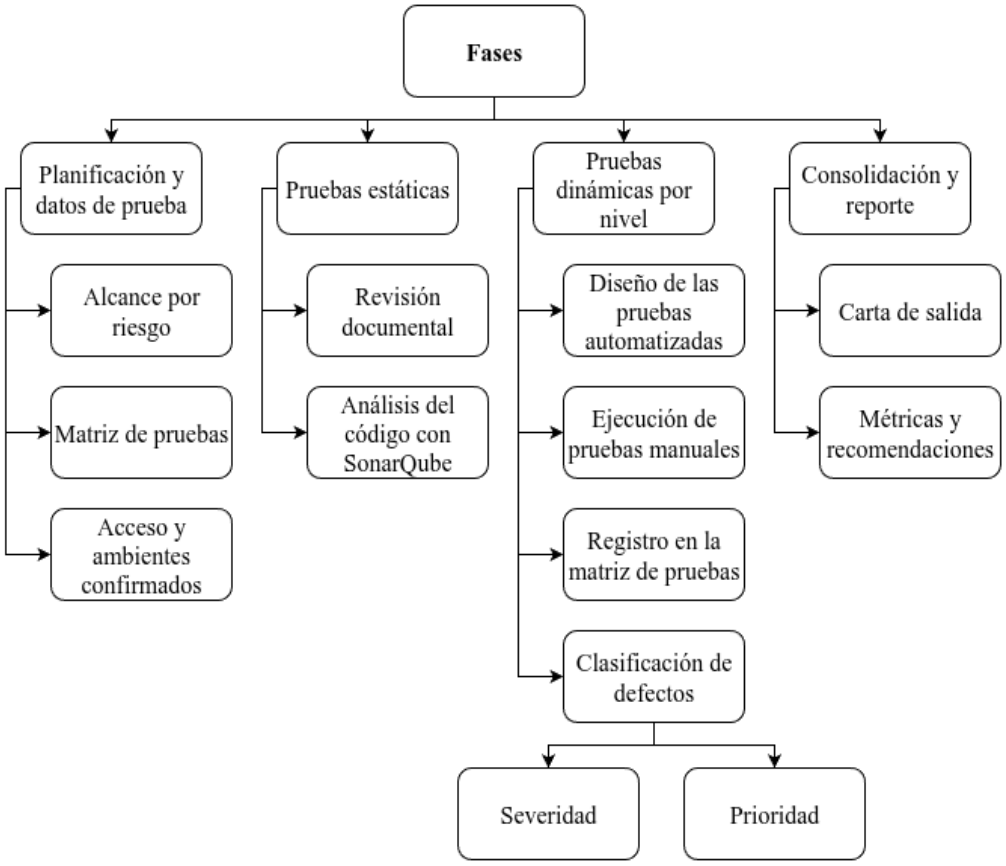
4. **Integración de sistemas:** validar, cuando aplique, interacciones con servicios externos (p. ej., correo y exportaciones) mediante datos controlados y trazabilidad de resultados.

5. **Aceptación:** demostrar, con representantes institucionales, que el sistema satisface necesidades de la SIC-UES en la gestión de convocatorias y proyectos, considerando criterios de negocio y evidencia documentada.

Fases:

Las fases de las pruebas se detallan a través de un diagrama en donde se visualizan todas las actividades o tareas que se desarrollarán:

Figura 5
Fases de pruebas.



Nota. Figura que representa las diferentes fases con todas las actividades que se realizan en cada una de ellas.

3.9.1. Proceso / Logística De Las Pruebas De Integración

Las pruebas de integración validan la interacción entre módulos principales del sistema: autenticación y permisos, gestión de convocatorias, proyectos de investigación (registro, pasos, presupuesto, financiamiento, colaboradores), catálogos, inventario, reportes/estadísticas y notificaciones, conforme a los procesos institucionales de la SIC-UES. Se preparan datos de prueba representativos y, cuando existan dependencias no disponibles, se utilizan sustitutos controlados; se mantiene trazabilidad con requisitos, historias de usuario y resultados en la matriz.

Preparación:

1. Inventario de interfaces y contratos entre módulos; definición de escenarios nominales y de falla; construcción de datos sintéticos consistentes y verificación del ambiente de pruebas.
2. Procedimientos y guías por interfaz (formularios web, exportaciones a PDF/Excel, envío de notificaciones) con entradas, salidas esperadas y validaciones de negocio.

Ejecución:

1. Orden incremental por riesgo: primero usuarios/roles, luego proyectos (alta, edición, pasos, envío), posteriormente convocatorias, reportes, inventario y estadísticas.

2. Casos de compatibilidad y manejo de errores (datos incompletos, duplicados, permisos insuficientes, fallos de exportación), registrando logs, capturas y códigos de caso en la matriz.

Criterios de salida:

1. Cobertura de interacciones planificadas, sin defectos críticos abiertos que bloqueen flujos de negocio y con evidencia completa y trazable en la matriz.

3.9.2. Proceso / Logística De Las Pruebas De Aceptación De Usuario

Las pruebas de aceptación comprueban, con perfiles representativos (gestión SIC-UES y usuarios clave), que el sistema cumple necesidades del negocio para gestionar convocatorias y proyectos de investigación, desde la postulación hasta el seguimiento y generación de reportes institucionales. Los escenarios se basan en historias críticas: registro y avance de proyectos, validaciones y finalización, gestión de convocatorias y notificaciones, generación de reportes e inventario, y consultas de estadísticas para apoyo a decisiones.

Preparación:

1. Selección de participantes (p. ej., investigador, administrador) y calendario; definición de guiones por caso de negocio con criterios de aceptación claros y datos seguros en ambiente de pruebas.

2. Formularios de evaluación de usabilidad y de cumplimiento de reglas de negocio para cada sesión, con instrucciones no técnicas y tiempos estimados por escenario.

Ejecución:

1. Sesiones por escenario con miembros del equipo de pruebas; registro de resultados, incidencias, tiempos, percepciones de claridad de interfaz y mensajes del sistema; soporte a dudas de procedimiento sin alterar resultados.

2. Evidencias asociadas en la matriz y actas por escenario (aprobado/aprobado con observaciones/no aprobado), preservando controles de acceso y confidencialidad.

Criterios de salida:

Cumplimiento de escenarios críticos del negocio, ausencia de defectos bloqueantes y documentación de observaciones priorizadas, con evaluación de riesgo residual para la carta de salida.

3.10. Áreas de enfoque de prueba

Basado en el Análisis de Riesgos desarrollado en el capítulo 2 y el contexto de negocio, el esfuerzo de pruebas no será uniforme pero se priorizará el esfuerzo en las siguientes áreas de enfoque para maximizar el valor y mitigar los riesgos más altos para la SIC-UES.

1. **Seguridad y Control de Acceso (Enfoque ISO: Seguridad):** Siendo un sistema que maneja datos de investigadores y proyectos, el control de acceso es la prioridad más alta. El enfoque validará que los roles, permisos y la autenticación sean robustos y que un usuario no autenticado no pueda acceder a ninguna ruta protegida.

2. **Flujo Crítico de Negocio (Enfoque ISO: Adecuación Funcional):** Se centrará en el "camino feliz" (happy path) que según Corinne Bernstein (2024) “son un tipo de prueba de software que utiliza entradas conocidas y produce una salida esperada” que define el propósito del sistema. Esto incluye:

    - La creación de Convocatorias.
    - El registro y gestión de Proyectos de Investigación.
    - La gestión del perfil del investigador.

3. **Integridad de Datos (Enfoque ISO: Fiabilidad):** Las pruebas asegurarán que los datos guardados (ej. en el perfil del investigador) sean los mismos que se muestran posteriormente en los Reportes y Estadísticas, garantizando la coherencia de la información.

4. **Calidad Estructural del Código (Enfoque ISO: Mantenibilidad):** Un enfoque clave no será solo qué hace el sistema, sino cómo está construido. Se utilizará análisis estáticos para asegurar que el sistema sea fácil de mantener, reduciendo el costo futuro de propiedad para la UES.

5. **Compatibilidad (Enfoque ISO: Compatibilidad):** Se validará que la experiencia de usuario sea consistente en los navegadores web más utilizados por la comunidad universitaria (Chrome, Firefox, Edge).

3.10.1. Pruebas Funcionales

Las pruebas funcionales (tipo "Caja Negra") se centran en validar la característica de Adecuación Funcional de la norma ISO 25010. Verifican que el sistema cumpla con los requisitos de negocio especificados, independientemente de la tecnología interna. Para este proyecto, se dividieron en dos estrategias complementarias:

1. **Pruebas Funcionales Automatizadas (Regresión):** Se automatizan 35 flujos críticos utilizando Laravel Dusk. El objetivo principal es crear un set de pruebas de regresión robusto. Este set valida las funcionalidades más importantes (Login, Creación de Proyectos, Gestión de Perfil) y se puede ejecutar rápidamente para certificar que nuevas correcciones o cambios no han roto funcionalidades existentes.

2. **Pruebas Funcionales Manuales (Exploratorias y de Cobertura):** Se definieron 43 casos de prueba manuales para cubrir áreas donde la automatización es ineficiente o requiere validación humana. Esto incluye:

    - Pruebas de seguridad complejas (ej. intentos de inyección SQL).
    - Validación de usabilidad y experiencia de usuario (UX).

- Comprobación visual de archivos exportados (PDF, Excel) de los reportes.
- Pruebas exploratorias para descubrir flujos no documentados.

3.10.2. Pruebas Estructurales

Las pruebas estructurales (tipo "Caja Blanca") evalúan la calidad interna de la arquitectura y el código fuente del sistema. Este enfoque es vital para validar la característica de Mantenibilidad de la ISO 25010.

Tal como se definió en la Estrategia de Pruebas, la principal herramienta para este enfoque es SonarQube. El código fuente completo del sistema se sometió a un análisis estático para identificar y reportar:

- **Bugs:** errores lógicos en el código (ej. variables nulas) que podrían causar fallos en producción.
- **Vulnerabilidades:** puntos débiles de seguridad (Security Hotspots) que el código expone (ej. riesgo de XSS, credenciales hardcodeadas), complementando las pruebas funcionales de seguridad.
- **Code Smells:** indicadores de un diseño deficiente, como código duplicado, métodos excesivamente largos o complejidad ciclomática alta. Corregir esto es clave para la mantenibilidad a largo plazo.
- **Deuda Técnica:** una estimación del tiempo que le tomaría al equipo de desarrollo corregir todos los problemas estructurales encontrados.

Como se mencionó en el Capítulo 2, las Pruebas Unitarias no forman parte del alcance de esta tesina, ya que son responsabilidad del equipo de desarrollo. El análisis estructural se enfoca, por tanto, en la calidad del producto ya construido.

3.11. Criterios de entrada/salida

Para gestionar formalmente el ciclo de ejecución de pruebas de este caso de estudio, se definen los siguientes criterios:

3.11.1. Criterios de Entrada

Condiciones que deben cumplirse para iniciar la ejecución de pruebas.

1. **Ambiente Listo:** el ambiente de pruebas local está configurado y es funcional, incluyendo el servidor web, PHP 7.4, MySQL 8.0.
2. **Código Desplegado:** la versión release candidate (candidata a lanzamiento) del código fuente ha sido desplegada en el ambiente de pruebas.
3. **Documentación Aprobada:** la Matriz de Pruebas está completa, revisada y aprobada por el Líder de QA.
4. **Herramientas Listas:** los scripts de automatización (Laravel Dusk) están desarrollados y SonarQube está configurado y conectado al repositorio.

3.11.2. Criterios de Salida

Condiciones que deben cumplirse para finalizar y aprobar el ciclo de pruebas.

1. **Ejecución Completa:** el 100 % de los casos de prueba definidos (78/78) han sido ejecutados al menos una vez y sus resultados están documentados en la matriz.
2. **Automatización Estable:** el 100 % de los scripts de regresión automatizados (35/35) se ejecutan exitosamente (o los fallos son reportados como defectos).
3. **Sin Defectos Críticos:** no existen defectos abiertos con severidad "Crítica" o "Alta" en los módulos priorizados.
4. **Reportes Generados:** el reporte de análisis de SonarQube ha sido generado.
5. **Entregables Listos:** el Reporte Final de Pruebas y el Resumen de Riesgos están compilados y listos para ser entregados a la SIC-UES.

3.12. Definición de defectos y tiempos de respuesta

Para gestionar los hallazgos de manera objetiva, se establece una clasificación estándar. Un defecto se define como cualquier discrepancia entre el comportamiento real del sistema y el comportamiento esperado (definido en los requisitos o casos de prueba).

3.12.1. Definición de prioridades en los defectos

Cada defecto se clasificará usando dos ejes: Severidad (impacto técnico) y Prioridad (impacto de negocio para la SIC-UES).

Tabla de severidad (impacto técnico).

Tabla 15
Severidad (impacto técnico).

Severidad	Descripción	Ejemplo en el Sistema SIC-UES
Crítica (Bloqueante)	El sistema se cae, hay pérdida de datos, o una función crítica principal es inutilizable. No existe solución alterna.	El botón 'Guardar Proyecto' genera un error 500 y borra los datos.
Alta	Una funcionalidad principal no opera correctamente, pero existe una solución alterna (workaround).	El filtro de Reportes no funciona, pero se puede exportar a Excel y filtrar manualmente.
Media	Comportamiento inesperado que no impide el flujo principal. Defecto estético en un módulo crítico.	La validación del campo 'Cargo actual' no permite tildes.

Severidad	Descripción	Ejemplo en el Sistema SIC-UES
Baja	Defecto menor, estético o de usabilidad que no afecta la funcionalidad. Error ortográfico.	El logo de la UES está desalineado en el navegador Firefox.

Nota. Se muestra la severidad o impacto técnico que puede tener un defecto encontrado en la evaluación.

Tabla 16
Prioridad (impacto del negocio).

Prioridad	Descripción
Urgente (P1)	Debe ser corregido inmediatamente. Usualmente asignado a defectos Críticos/Altos en el flujo principal.
Alta (P2)	Debe ser corregido para el lanzamiento (release) o en el siguiente ciclo.
Media (P3)	Arreglar si el tiempo lo permite.
Baja (P4)	Se registra en el backlog para futuras mejoras.

Nota. El impacto que puede tener un defecto y también la importancia de realizar una corrección.

3.13. Análisis de riesgos

Si bien la metodología de cálculo y el plan de contingencia teórico se definieron en el Capítulo 2, durante la fase de ejecución de este Caso de Estudio fue necesario activar estrategias reales de mitigación. A continuación, se detalla en la tabla cómo se gestionan los riesgos materializados durante el trabajo de campo:

Tabla 17
Análisis de riesgos.

Tipo de Riesgo	Riesgo Identificado	Gestión Realizada durante la Ejecución
Riesgo de Producto	Funcionalidades incompletas o con errores críticos.	Este riesgo fue mitigado al enfocar el 70% del esfuerzo de pruebas (automatizadas y manuales) exclusivamente en los módulos críticos para el negocio de la SIC-UES: Perfil de Investigador y Proyectos de Investigación. Esto aseguró que la funcionalidad principal del sistema estuviera operativa.
	Seguridad de la información.	Se mitigó mediante una estrategia de defensa en profundidad, ejecutando casos de prueba específicos de control de acceso y utilizando SonarQube para realizar un análisis estructural. Esto permitió detectar vulnerabilidades de código antes de la puesta en producción.

Tipo de Riesgo	Riesgo Identificado	Gestión Realizada durante la Ejecución
Riesgo de Proyecto	Retrasos en ejecución.	Se mitigó proactivamente mediante la automatización con Laravel Dusk de los 35 casos de prueba más repetitivos. Esta decisión técnica redujo el tiempo de las pruebas de regresión de horas a minutos, permitiendo cumplir con el calendario estipulado.
	Deficiencias en documentación.	Se mitigó al complementar los casos de prueba formales con sesiones de Pruebas Exploratorias. Esto permitió descubrir y validar flujos de usuario reales que no estaban reflejados en la documentación técnica original entregada.

Nota. Tabla donde se realiza un análisis de los riesgos identificados y la gestión que se realizó durante la ejecución del plan de pruebas.

3.14. Supuestos

Dentro del caso de estudio y el proyecto realizado se tienen diferentes supuestos o condiciones que establecemos como ya establecidas. También conocidas como precondiciones, dentro de estos supuestos tenemos:

a. El Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación se encuentra desarrollado y el plan de pruebas realizadas evaluará el sistema antes de entrar a producción para detectar errores o defectos que afecten la adopción del sistema.
b. Los usuarios del Sistema de Gestión de Convocatorias y Registros de Proyectos de Investigación serán miembros de la Universidad de El Salvador que tengan acceso autorizado por la Secretaría de Investigaciones.
c. Los manuales de usuario, manual técnico y toda la documentación proporcionada por la Secretaría de Investigaciones sobre el sistema fue verificada y corregida previamente por el equipo de desarrollo y aprobada por la propia Secretaría de Investigaciones.
d. La Secretaría de Investigaciones revisará el resultado obtenido por el equipo de pruebas a través de los reportes de pruebas que el equipo proporcionará y la Secretaría de Investigaciones tendrá la decisión de implementar las acciones de mejora o no según sea el motivo.

3.15. Tareas del equipo de pruebas

Cada miembro del equipo realizará diferentes tareas para la evaluación del sistema y proporcionar una evaluación de calidad con el fin de mejorar la integridad y fiabilidad del sistema para el uso dentro de la Secretaría de Investigaciones.

Para este objetivo cada miembro del equipo realizará las siguientes tareas:

Tabla 18
Tareas del equipo de pruebas.

Integrante	**Tarea**
Oscar Leonardo Serpas Martínez	● Analizar y diseñar pruebas para detectar defectos o errores en el sistema. ● Planificar el calendario para la ejecución de los casos de pruebas. ● Asignar a los miembros del equipo responsables de la ejecución de cada caso de prueba. ● Ejecutar los casos de pruebas y registrar con evidencia los resultados en la matriz de pruebas. ● Utilizar los criterios de calidad de la Norma ISO/IEC 25010 para la validación de los resultados de las pruebas obtenidas. ● Documentar todos los defectos encontrados durante la ejecución de pruebas y registrar los cambios en la matriz de pruebas. ● Revisar el cumplimiento correcto de los pasos para cada caso de prueba ejecutado y registrado en la matriz de pruebas. ● Verificar los recursos utilizados para la realización de las pruebas y proporcionar ayuda en dado caso falten recursos. ● Estimar el esfuerzo requerido para el cumplimiento del plan de pruebas.
Marcelo Josué Alcántara Alas	● Analizar y diseñar pruebas para detectar defectos o errores en el sistema. ● Ejecutar los casos de pruebas y registrar con evidencia los resultados en la matriz de pruebas. ● Utilizar los criterios de calidad de la Norma ISO/IEC 25010 para la validación de los resultados de las pruebas obtenidas. ● Documentar todos los defectos encontrados durante la ejecución de pruebas y registrar los cambios en la matriz de pruebas. ● Indicar acciones para mejorar la calidad del sistema según las pruebas realizadas. ● Estimar el esfuerzo requerido para el cumplimiento del plan de pruebas.
David José Castro Clímaco	● Analizar y diseñar pruebas para detectar defectos o errores en el sistema. ● Ejecutar los casos de pruebas y registrar con evidencia los resultados en la matriz de pruebas. ● Utilizar los criterios de calidad de la Norma ISO/IEC 25010 para la validación de los resultados de las pruebas obtenidas. ● Documentar todos los defectos encontrados durante la ejecución de pruebas y registrar los cambios en la matriz de pruebas. ● Indicar acciones para mejorar la calidad del sistema según las pruebas realizadas. ● Estimar el esfuerzo requerido para el cumplimiento del plan de pruebas.

Nota. Descripción de las diferentes tareas que tienen que realizar cada miembro del equipo de pruebas.

3.16. Roles Y Responsabilidades

Dentro de los roles principales para este proyecto tenemos:

- Analista de Pruebas de Software QA.
- Líder de Pruebas QA.

Con los roles seleccionados se determinaron para cada miembro del equipo en base a su conocimiento y capacidades y las responsabilidades que tiene cada miembro se presentan a continuación:

Tabla 19
Roles y responsabilidades.

Integrante	**Rol**	**Responsabilidades**
Oscar Leonardo Serpas Martínez.	Líder de Pruebas QA.	1. Planificación de pruebas. 2. Revisión de resultados. 3. Diseño de pruebas. 4. Monitoreo y control de pruebas. 5. Reporte final del plan de pruebas.
Marcelo Josué Alcántara Alas	Analista de Pruebas de Software QA.	1. Diseño de pruebas. 2. Implementación de pruebas. 3. Ejecución de pruebas. 4. Reporte de pruebas.
David José Castro Clímaco	Analista de Pruebas de Software QA.	1. Documentación del plan de pruebas. 2. Implementación de pruebas. 3. Ejecución de pruebas. 4. Reporte de pruebas.

Nota. Tabla que contiene los diferentes roles y responsabilidades por cada miembro del equipo de pruebas.

3.17. Calendario de Pruebas

Para la ejecución de los casos de pruebas se ha elaborado un calendario en donde se detallan las fechas previstas para realizar las pruebas:

Tabla 20
Calendario de pruebas.

Casos de pruebas	**Fecha de inicio**	**Fecha de finalización**	**Responsable**
CP-01 - CP-07	03/11/2025	04/11/2025	● Oscar Serpas. ● Marcelo Alcántara.
CP-08 - CP-35	05/11/2025	11/11/2025	● Oscar Serpas. ● Marcelo Alcántara.

Casos de pruebas	Fecha de inicio	Fecha de finalización	Responsable
CP-36 - CP-54	12/11/2025	15/11/2025	● Oscar Serpas. ● Marcelo Alcántara. ● David Castro.
CP-55 - CP-70	16/11/2025	19/11/2025	● Oscar Serpas. ● Marcelo Alcántara. ● David Castro.
CP-71 - CP-78	20/11/2025	23/11/2025	● David Castro.

3.18. Principales Hitos

Los eventos claves del proceso de pruebas se definen a continuación:

Tabla 21
Principales hitos.

Hito	Descripción
Planeación de pruebas	Definición de los casos de pruebas a realizar en el plan de pruebas.
Inicio de la ejecución de pruebas	Este hito marca el inicio de la ejecución de pruebas definidas en la matriz de pruebas.
Ejecución de las pruebas de seguridad	Se comienzan a ejecutar las pruebas de seguridad sobre el sistema.
Ejecución de las pruebas del sistema o funcionales	Se ejecutan las pruebas funcionales definidas en la matriz de pruebas.
Ejecución de las pruebas de compatibilidad	Ejecución de las pruebas no funcionales de compatibilidad definidas en la matriz de pruebas.
Finalización de la ejecución de pruebas	Se finaliza el proceso de ejecución de pruebas y documentación de los resultados obtenidos.
Revisión de defectos	Se validan y verifican los defectos encontrados durante la ejecución de los casos de pruebas.
Análisis de los resultados	El equipo analiza los resultados obtenidos y determina las acciones o recomendaciones necesarias para corregir los defectos encontrados.
Entrega del reporte final	Después de analizar y proporcionar recomendaciones o acciones a mejora para el sistema se entrega el reporte final a la Secretaría de

Hito	Descripción
	Investigaciones Científicas.

Nota. Tabla con los principales hitos durante la ejecución del plan de pruebas.

3.19. Recursos Necesarios

Los recursos necesarios para el cumplimiento del plan de pruebas se dividen en tres tipos:

- **Recursos de hardware.** Recursos necesarios para poder simular un ambiente de pruebas necesario para ejecutar las pruebas definidas en el plan de pruebas.
- **Recursos de software.** Aplicaciones y herramientas para elaborar, registrar y automatizar las pruebas de software.
- **Recursos humanos.** Es el personal necesario para la elaboración y ejecución de pruebas.

Para cada uno de los tipos de recursos se determinan los recursos indispensables para poder continuar con el plan de pruebas y su ejecución, estos recursos son los siguientes:

Tabla 22
Recursos necesarios.

Tipo de recursos	Recurso
Hardware	● 3 computadoras con acceso a Internet. ● Procesador de 4 núcleos con arquitectura de 64 bits. ● Memoria RAM mínima de 8 GB.
Software	● Código fuente del sistema. ● Dependencias de Laravel (PHP, NodeJS y MySQL). ● Sistema operativo Windows 10/11 o Ubuntu 20.04 o superior. ● Navegador web Google Chrome, Mozilla Firefox o Microsoft Edge. ● Microsoft Office. ● Laravel Dusk. ● SonarQube.
Humano	● 2 analistas de QA. ● 1 líder de QA.

Nota. Recursos requeridos para el cumplimiento del plan de pruebas.

3.19.1. Ambientes De Pruebas

Los requisitos necesarios para la ejecución de los casos de prueba se dividen en requisitos de hardware y software. A continuación, se describen los elementos mínimos requeridos para el ambiente de pruebas utilizado en este proyecto, el cual es el único ambiente disponible, dado que el sistema ya fue desarrollado y no se cuenta con un ambiente de desarrollo.

Requisitos ambientales de hardware

- Computadora con acceso estable a Internet.
- Procesador de 4 núcleos con arquitectura de 64 bits.
- Mínimo de 8 GB de memoria RAM.

Requisitos ambientales de software

1. Código fuente del sistema en su última versión disponible.
2. Sistema gestor de bases de datos: MySQL 8.0.
3. PHP versión 7.4.
4. Node.js versión 14.0 o superior.
5. Sistema operativo: Windows 10, Windows 11 o Ubuntu 20.04 o superior.
6. Navegador web compatible con motor Chromium (por ejemplo, Google Chrome, Microsoft Edge) o Gecko (Mozilla Firefox).
7. Microsoft Office (Word, Excel y PowerPoint), usado para análisis, generación de reportes y documentación de evidencias.

Para la ejecución de las pruebas se utilizará una base de datos con datos de pruebas o ficticios y todas las pruebas se realizarán de manera local para no generar costos o problemas en un servidor de producción.

El cumplimiento de estos requisitos garantiza que las pruebas se ejecuten en condiciones similares a las del entorno de producción, permitiendo la identificación confiable de defectos y la recolección de resultados representativos. La correcta documentación y preparación del ambiente es fundamental para lograr que los resultados sean reproducibles y útiles para posteriores procesos de validación de calidad.

3.19.2. Planeación de recursos

La planeación de recursos consiste en la asignación de todos los recursos disponibles para las diferentes actividades que se realizarán durante el proyecto y de esta manera distribuir de manera correcta para completar todas las actividades. Para el plan de pruebas se tiene la siguiente asignación de recursos:

Tabla 23
Planeación de recursos.

Actividad	**Encargados**	**Herramienta/recursos**	**Duración**
Planificación de las pruebas	Líder de QA. 2 analistas de QA.	● Computadora con Internet. ● Microsoft Excel.	2 semanas
Ejecución de pruebas de seguridad	Líder de QA 2 analistas de QA.	● Computadora con Internet. ● Laravel Dusk. ● SonarQube ● Navegador web.	1 semana
Ejecución de pruebas	Líder de QA.	● Computadora con Internet.	2 semana

Actividad	Encargados	Herramienta/recursos	Duración
funcionales o de sistema	2 analistas de QA.	● Laravel Dusk. ● SonarQube ● Navegador web.	
Ejecución de pruebas de compatibilidad	Analista de QA.	● Computadora con Internet. ● Laravel Dusk. ● Navegador web.	3 días
Registro de defectos	2 analistas de QA.	● Computadora con Internet. Microsoft Excel.	5 días
Análisis de los resultados	Líder de QA. 2 analistas de QA.	● Computadora con Internet. Microsoft Excel.	3 días
Reporte final de pruebas	Líder de QA. 2 analistas de QA.	● Computadora con Internet. Microsoft Word. ● Microsoft Excel.	2 días

Nota. Asignación de los diferentes recursos y responsables para las diferentes actividades.

3.20. Estrategia de datos de prueba y herramientas

Para la ejecución de las pruebas se debe realizar una estrategia con los procedimientos con los que se definirán el tipo de datos utilizados para la ejecución de pruebas y las herramientas utilizadas para la automatización y pruebas manuales. De igual manera se deberá especificar y detallar el procedimiento para el registro de los resultados obtenidos de las pruebas ejecutadas.

3.20.1. Estrategia de datos

Los datos utilizados para las pruebas se dividen según el tipo de prueba de la siguiente manera:

Pruebas de seguridad

Para las pruebas de seguridad se hará uso de los datos de usuarios de ejemplos en el Manual del Usuario del sistema para verificar los diferentes roles y permisos de los usuarios. En base a esto se hará uso de los siguientes roles:

Tabla 24
Roles de usuarios del sistema.

Rol	Usuario	Contraseña
Administrador	a@a	12345678
Gestor de seguimiento	gespic@gespic	12345678
Candidato	candidato@candidato	12345678

Rol	Usuario	Contraseña
Investigador	i@i	12345678

Nota. Diferentes usuarios y contraseñas que se utilizarán en las pruebas para la evaluación del sistema.

Para los casos exitosos se utilizarán estos valores de usuarios donde se comprobará los permisos y opciones disponibles para cada rol del usuario. También se utilizarán datos erróneos con el objetivo de verificar la validación de los campos y la seguridad del sistema.

Pruebas del sistema

En las pruebas funcionales del sistema se utilizarán datos específicos para pruebas que cumplan con los requisitos específicos de cada formulario o campo dentro del sistema tanto para las pruebas manuales como las pruebas automatizadas.

Los datos personales (como nombres, apellidos, fechas de nacimiento, identificación, entre otros) serán datos ficticios y no corresponden a ninguna identificación personal de personas relacionadas a la Secretaría de Investigaciones.

Algunos ejemplos de estos datos personales son:

- Nombres: Ejemplo Nombre.
- Apellidos: Ejemplo Apellidos.
- Fecha de nacimiento: 1990-01-01.
- Teléfono: 7777 – 8594.

Pruebas de compatibilidad

Para las pruebas de compatibilidad se hará uso de las direcciones correspondientes a los distintos formularios y pantallas de visualización para verificar la compatibilidad del sistema en diferentes entornos.

Para las pruebas relacionadas a la subida o descarga de archivos en formato PDF o de hoja de cálculo de Microsoft Excel (extensión .xlsx) se trabajará con datos ficticios almacenados en la base de datos de uso único y específico para pruebas y no tienen relación a los datos que se almacenarán en el sistema para el uso dentro de la Secretaría de Investigaciones Científicas. De igual manera con respecto a la subida de archivos se comprobará solo con la compatibilidad del formato del archivo (PDF o Microsoft Excel) en diferentes navegadores y resoluciones para comprobar el correcto funcionamiento del sistema.

3.20.2. Herramientas de prueba

Se utilizaron diferentes herramientas para la gestión, ejecución y registro de las pruebas descritas en la matriz de pruebas. Estas herramientas se describen a continuación:

Tabla 25
Herramientas para las pruebas.

Herramienta	Tipo de herramienta	Descripción de uso
Laravel Dusk	Automatización de pruebas	Ejecución automática de pruebas funcionales.
Microsoft Excel	Matriz de pruebas	Registro y descripción de los casos de pruebas y sus resultados.

Nota. Herramientas y descripción del uso de las mismas para las diferentes pruebas.

3.20.2.1. Pruebas de código

Para las pruebas de código se ha utilizado la herramienta de SonarQube. Esta es una herramienta diseñada para analizar y encontrar defectos en el código fuente de una aplicación con el análisis estático.

SonarQube permite detectar posibles errores de diseño y de desarrollo dentro de un sistema antes de su ejecución e implementación, por lo tanto, es una herramienta diseñada para minimizar los defectos estructurales encontrados en el desarrollo de un sistema, proporcionando estándares para reducir la complejidad de un sistema y aumentar su facilidad de mantenimiento y portabilidad.

Dentro de las características con las que cuenta SonarQube se encuentra la capacidad para realizar un análisis estático a un componente, módulo o método del código fuente del sistema o a todo el código fuente. Este análisis se realiza para encontrar posibles problemas de complejidad o fiabilidad dentro del sistema y proporciona una clasificación de la severidad del defecto encontrado para dar prioridad a los defectos más críticos y así solucionar los problemas estructurales dentro del sistema en desarrollo. También permite generar reportes en diferentes formatos para uso en auditorías o en presentaciones.

El uso de SonarQube se realiza dentro de los componentes críticos del sistema dentro de los módulos principales para el funcionamiento esperado del sistema. Entre estos módulos encontramos:

- Módulo de perfil del investigador.
- Módulo de proyectos de investigación.
- Módulo de usuarios y permisos.
- Módulo de convocatorias.
- Módulo de reportes.

Con el resultado del análisis estático realizado con SonarQube se determinarán los aspectos a mejorar y dar cumplimiento con los requisitos del sistema y se reportarán las acciones recomendadas para solucionar los problemas encontrados.

3.21. Entregables de pruebas automatizada y no automatizadas

Los entregables que se realizarán después de la ejecución de los casos de pruebas se dividen en pruebas automatizadas y no automatizadas. Se describen los entregables a continuación:

Tabla 26
Entregables de pruebas automatizadas y no automatizadas.

Tipo	Descripción	Entregables
Pruebas automatizadas	Son las pruebas que se determinaron para su automatización con Laravel Dusk.	● Código fuente con scripts donde están implementados los casos de prueba automatizados. ● Reporte de ejecución de las pruebas automatizadas (éxito o fallo). ● Documentación técnica de pruebas automatizadas. ● Evidencia visual y observaciones obtenidas durante la ejecución de las pruebas automatizadas.
Pruebas no automatizadas	Estás son las pruebas que no fueron contempladas para su automatización y se realizarán manualmente para registrar y comprobar el funcionamiento correcto.	● Reporte de evidencia y observaciones obtenidas durante la ejecución de las pruebas no automatizadas. ● Casos de pruebas con descripción de los pasos a seguir para su ejecución y los criterios de aceptación. ● Reporte de defectos y recomendaciones de mejora para asegurar una calidad óptima del sistema.

Estos entregables se presentarán a la Secretaría de Investigaciones Científicas para corroborar la evaluación realizada al Sistema Informático para la Gestión de Convocatorias y Registro de Proyectos de Investigación.

3.21.1. Matriz de pruebas

La matriz de pruebas constituye un instrumento esencial dentro del proceso de evaluación de calidad, ya que permite dar trazabilidad entre los requisitos funcionales y no funcionales y los casos de prueba diseñados. Según la ISO/IEC 25010, la adecuación funcional y la confiabilidad se evalúan mediante la verificación de que cada requisito es cubierto por pruebas específicas y medibles.

Además, permite gestionar y brindar un dashboard informativo para el usuario en donde pueda obtener información útil para la toma de decisiones, para gestionar los recursos necesarios para brindar acciones correctivas al sistema.

La matriz de prueba tiene 15 campos que brindan información relevante sobre cada caso de prueba, además de su identificación y los resultados esperados. Su descripción se presenta a continuación:

Tabla 27
Campos de la matriz de pruebas.

N°	Campo	Descripción
1	ID Caso de prueba	Identificador único del caso de prueba
2	Prueba automatizada	Indica si se automatizará la prueba.
3	Tipo prueba	Identificación del tipo de prueba.
4	Módulo	Módulo del sistema donde se ejecuta la prueba.
5	Caso de prueba	Nombre del caso de prueba.
6	Descripción	Descripción breve del caso de prueba.
7	Precondiciones	Condiciones previas para la ejecución de la prueba.
8	Datos de prueba	Datos que se utilizarán para ejecutar el caso de prueba.
9	Pasos de ejecución	Pasos ordenados para ejecutar el caso de prueba.
10	Criterios de aceptación	Breve descripción del resultado esperado al realizar la prueba
11	Resultado obtenido	Breve resumen del resultado obtenido
12	Estado	Resultado del caso de prueba
13	Prioridad	Nivel de prioridad del caso de prueba.
14	Porcentaje de avance	Porcentaje de avance del caso de prueba.
15	Evidencia	Evidencia de la ejecución de la prueba.

Para la visualización de la matriz de pruebas se entregará un documento de Microsoft Excel en donde se encontrará toda la información correspondiente a los casos de prueba registrados en la matriz de pruebas junto con el dashboard informativo y gráficas correspondientes al estado de las pruebas.

3.21.2. Test Readiness Review (TRR)

El Test Readline Review o en español Revisión de Preparación para las Pruebas es una revisión multidisciplinaria diseñada para asegurar que el sistema que se está probando (hardware o software) esté listo para proceder a las pruebas formales (DAU, s. f.). Entre otros aspectos, la TRR evalúa los objetivos, métodos y procedimientos de las pruebas, los recursos de apoyo, el alcance, la identificación de riesgos y las medidas de mitigación, así como la disponibilidad de las autorizaciones pertinentes. Esta evaluación debe confirmar que todos los recursos necesarios se han identificado correctamente y están disponibles para respaldar las actividades de prueba planificadas.

En resumen, su propósito es confirmar o asegurar que todos los requerimientos para comenzar las pruebas estén completos y el equipo de pruebas esté listo para comenzar a realizar las pruebas.

Para el caso de este proyecto se tienen los siguientes requisitos para poder iniciar la evaluación del sistema:

Tabla 28
Requisitos para iniciar las pruebas.

Requisito	**Estado**
Ambiente de pruebas estable y preparado para las pruebas.	Listo
Código fuente del sistema disponible.	Listo
Documentación técnica y manuales de usuario del sistema.	Listo
Casos de pruebas definidos y aceptados.	Listo
Roles y responsabilidades establecidas para los miembros del equipo.	Listo
Datos de prueba disponibles para los casos de prueba.	Listo
Herramientas requeridas para la realización de pruebas configuradas y listas para su uso.	Listo

3.22. Estimaciones

Para dimensionar el esfuerzo requerido en la fase de ejecución y control de calidad, se realizaron estimaciones basadas en la complejidad de los casos de prueba diseñados y la experiencia técnica del equipo de trabajo. Estas estimaciones permiten gestionar el calendario y los recursos humanos de manera eficiente.

1. **Esfuerzo de Pruebas Manuales:** se estimó un promedio de 15 minutos por caso de prueba manual. Considerando los 43 casos manuales (enfocados en UX, validaciones complejas de formularios y compatibilidad entre navegadores), se proyectó un esfuerzo aproximado de 11 horas de ejecución. Este tiempo incluye la preparación del dato de prueba, la ejecución de los pasos y la documentación de la evidencia.
2. **Esfuerzo de Pruebas Automatizadas:** para la automatización con Laravel Dusk, el esfuerzo se dividió en dos etapas diferenciadas: Desarrollo de Scripts (Scripting): Se estimó un promedio de 2 horas de desarrollo por script complejo. Para los 35 casos automatizados, esto representó la mayor inversión de tiempo técnico previo a la ejecución. Tiempo de Ejecución (Run-time): Una vez automatizados, la ejecución completa de la suite de pruebas de regresión toma aproximadamente entre 5 a 8 minutos en total. Esto demuestra la alta eficiencia de la automatización, permitiendo ejecutar la batería completa múltiples veces al día sin costo adicional de tiempo humano.
3. **Esfuerzo de Análisis Estático:** la configuración, ejecución e interpretación de los reportes de SonarQube se estimó en 16 horas hombre, dedicadas a la revisión de cada "Code Smell" y vulnerabilidad crítica reportada para generar las recomendaciones de mejora pertinentes.

3.23. Defectos

Un defecto en el contexto de este proyecto se define como cualquier desviación entre el resultado esperado definidos en los requisitos de la SIC-UES, la norma ISO 25010 y el resultado obtenido durante la ejecución de las pruebas.

Todos los defectos encontrados se documentan utilizando un formato de control estandarizado identificado como Gestión de defectos. Este formato asegura la uniformidad en el reporte y facilita la corrección por parte del equipo de desarrollo. La estructura de reporte utilizada incluye los siguientes campos obligatorios:

Tabla 29
Estructura del reporte de defectos.

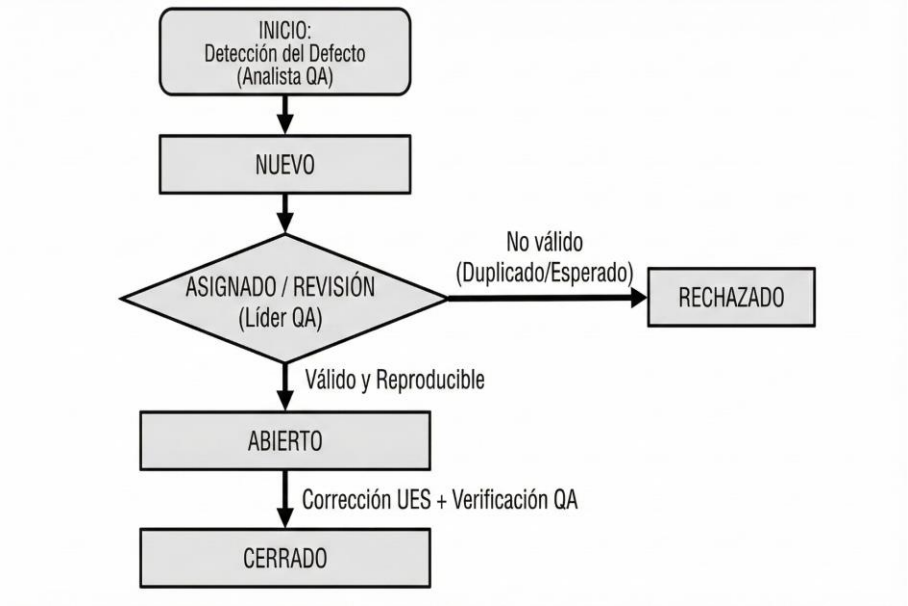
Campo	**Descripción**
ID del defecto	Identificador único generado para el seguimiento del hallazgo (ej. DEF-001).
ID Caso de Prueba	Referencia al código del caso de prueba (ej. CP-01) donde se detectó el fallo, garantizando la trazabilidad entre la prueba y el error.
Descripción	Explicación breve y concisa del problema encontrado, indicando qué está fallando en el sistema.
Severidad	Clasificación del impacto técnico del defecto (Crítica, Alta, Media, Baja).
QA asignado	Nombre del analista de calidad responsable de haber detectado y reportado el incidente.
Módulo	Módulo del sistema en donde se encontró el defecto.
Recomendación	Acciones de mejora o prevención para corregir el defecto encontrado y proporcionar una mayor fiabilidad al sistema en busca de una calidad óptima para el uso de los usuarios finales.

3.23.1. Proceso de seguimiento de defectos

Para asegurar que cada hallazgo registrado en el formato planteado sea gestionado correctamente, se estableció un ciclo de vida del defecto. Dado que el alcance de esta tesina es de análisis e implementación de pruebas, el flujo se adapta a la comunicación con la Secretaría de Investigaciones Científicas.

A continuación, en la figura, se presenta el diagrama de flujo que ilustra los estados por los que transita un defecto desde su detección hasta su cierre.

Figura 6
Estados de un defecto.



Los estados detallados en el flujo se describen a continuación:

1. **Nuevo:** el defecto es detectado por el analista QA y registrado en la matriz con todos sus campos completos.
2. **Asignado/Revisión:** el Líder de QA revisa los campos "Resultados observados" y "Evidencia adjunta" para asegurar que el defecto sea válido y reproducible.
3. **Abierto:** el defecto ha sido validado y comunicado oficialmente en el reporte para la SIC-UES, quedando pendiente de acción correctiva.
4. **Rechazado:** estado asignado si se determina que el "Resultado observado" es realmente una funcionalidad esperada o si el reporte está duplicado.
5. **Cerrado:** estado final que se asignará en el futuro, una vez que el equipo de desarrollo de la UES corrija el fallo y el equipo de QA verifique que los "Resultados observados" ahora coinciden con los "Resultados esperados".

3.23.2. Revisión de defectos

Antes de incluir cualquier hallazgo en el Informe Final, se realiza un proceso de depuración de defectos. Este proceso valida la calidad de la información ingresada en el formato de gestión de defectos.

Este proceso es responsabilidad del Líder de QA y tiene como objetivos:

- **Validar la Trazabilidad:** asegurar que el campo "ID Caso de Prueba" corresponda correctamente al flujo que se estaba probando.
- **Claridad en la Reproducción:** verificar que los "Pasos para reproducir" sean lo suficientemente claros para que cualquier desarrollador pueda entender el error sin necesidad de contactar al QA asignado.

- **Consistencia de Severidad:** confirmar que la clasificación de severidad sea objetiva y coherente con el impacto real en el negocio, evitando alarmas innecesarias o subestimación de riesgos críticos.

3.24. Proceso para reporte de avance

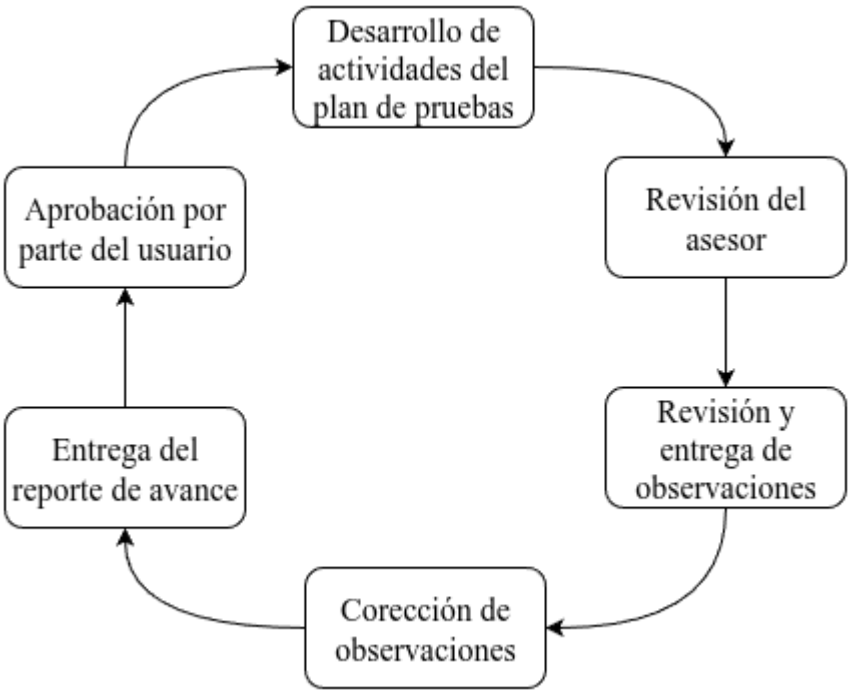
El Reporte de Avance es utilizado para poder comunicar el estado de las actividades de prueba al usuario o stakeholders involucrados en el sistema. Este reporte tiene que determinar cada cuánto será presentado para informar sobre los avances en las actividades realizadas y debe contener diferentes datos y métricas de suma importancia.

Para el Reporte de Avance se ha seguido un proceso en donde se indicará diferentes métricas entre las cuales tenemos:

- Cobertura de pruebas.
- Tasa de éxito de las pruebas.
- Gráfica representativa del estado de las pruebas.

El Reporte de Avance se entregará al finalizar cada capítulo del correspondiente documento para la visualización del usuario y aceptación. Este reporte seguirá el siguiente flujo representado en un diagrama:

Figura 7
Diagrama del proceso de reporte de avance.



3.25. Documentación técnica de pruebas automatizadas.

La automatización de las pruebas funcionales se realizó utilizando el framework Laravel Dusk, el cual permite la simulación de interacciones de usuario en un navegador real (Google Chrome). Para garantizar la mantenibilidad y la trazabilidad entre el Plan de Pruebas y el código fuente, se estableció una estructura técnica estandarizada para todos los scripts desarrollados.

A continuación, se detalla la arquitectura de los tests, la ubicación de los archivos y la correspondencia con los Casos de Prueba (CP) definidos.

Estructura del repositorio de pruebas: los scripts de automatización se encuentran en el directorio que es creado por defecto dentro del proyecto. Se organizó cada archivo de pruebas agrupando funcionalidades por módulo, facilitando su localización y ejecución.

- **Directorio base:** /tests/Browser
- **Archivos de prueba generados:**
  - LoginTest.php
  - SeguridadAccesoTest.php
  - SeguridadAutenticacionTest.php
  - SeguridadTest.php
  - ConvocatoriaDeleteTest.php
  - ConvocatoriaTest.php

Ejecución y Generación de Evidencias

La ejecución de las pruebas automatizadas se realiza mediante la consola de comandos del servidor de desarrollo. El framework genera dos tipos de evidencia técnica que respaldan los resultados presentados en la matriz de pruebas:

1. **Reporte de Consola:** al finalizar la ejecución, Laravel Dusk emite un resumen con el tiempo total de ejecución (ej. 34 segundos para el CP-01), la cantidad de pruebas pasadas y las fallidas.
2. **Capturas de Fallo:** en caso de que una aserción falle (por ejemplo, si el sistema no muestra el error esperado al ingresar una contraseña incorrecta), Dusk genera automáticamente una captura de pantalla en la carpeta /tests/Browser/screenshots/ con el nombre del fallo, permitiendo al equipo de QA realizar un análisis posterior del error.

El código fuente completo de los scripts desarrollados se encuentran detallados en el Anexo D: Scripts de Automatización.

3.26. Carta De Salida

La ejecución de las pruebas planteadas en la matriz de pruebas se ha realizado a través de pruebas manuales y automatizadas definidas según los criterios planteados en los temas correspondientes. Para la realización de las pruebas se utilizaron las siguientes herramientas:

Tabla 30
Herramientas utilizadas.

Herramienta utilizada	Función
Laravel Dusk	Pruebas automatizadas
SonarQube	Análisis estático
Microsoft Excel	Matriz de pruebas Dashboard
Navegadores (Google Chrome, Microsoft Edge y Mozilla Firefox)	Pruebas manuales
Microsoft Word	Documentación de evidencias

Nota. Herramientas utilizadas y su función principal dentro del proyecto.

Resultados obtenidos

Durante la ejecución de la evaluación del sistema a través de un análisis estático con la herramienta SonarQube y las pruebas manuales y automatizadas con Laravel Dusk se obtuvieron diferentes resultados que reflejan el estado actual del sistema.

Los resultados obtenidos del análisis estático se muestran a continuación en donde se indican la cantidad de anomalías obtenidas y su impacto dentro del sistema y también el tipo de defecto encontrado.

Análisis estático

Dentro de los defectos encontrados en el análisis estático realizado al código del sistema encontramos un total de 9 bugs y 550 Code Smell según el reporte de SonarQube, estos tipos de defectos tienen su severidad dentro del sistema y se clasifican en cuatro tipos y se muestran desglosados en la siguiente tabla:

Tabla 31
Resultados análisis estático.

Tipo	Severidad	Cantidad de defectos
Bug	Bloqueadora	2
	Crítica	0
	Media	5
	Baja	2
Code Smell	Bloqueadora	0
	Crítica	93
	Media	135

Tipo	Severidad	Cantidad de defectos
	Baja	322

Con estos resultados visualizamos que existen problemas críticos y bloqueantes dentro del código fuente del sistema y se debe de resolver los problemas encontrados para no perjudicar más adelante al funcionamiento correcto de otros componentes a la hora de integrar y comprobar el funcionamiento correcto de los métodos y funciones del sistema.

Para visualizar de manera detallada cada defecto encontrado durante el análisis estático con SonarQube se genera un reporte detallado en formato compatible con Microsoft Excel o cualquier herramienta compatible con archivos con extensión .xlsx en donde se visualiza cada uno de los defectos encontrados.

De igual manera si se quiere utilizar SonarQube para realizar un análisis estático se debe configurar correctamente para poder realizar el análisis deseado. Para esto se encuentra en el Anexo E un manual de instalación y uso para ejecutarlo en el sistema operativo Windows.

Con la ejecución de pruebas finalizada obtenemos los siguientes resultados:

Tabla 32
Resultados obtenidos.

Resumen de pruebas	Éxito	Fallo	En curso	No ejecutado	Suspendido	Total de pruebas
Pruebas	66	8	0	0	4	78
Porcentaje	84.62%	10.26%	0,00%	0,00%	5.13%	100.00%

Esto indica el estado de las pruebas realizadas y el porcentaje que representa cada uno de los estados. Con estos resultados podemos determinar las métricas necesarias para el total de pruebas y el porcentaje de éxito/fallo obtenido en la ejecución de pruebas

Las métricas obtenidas según los resultados de la ejecución de pruebas se muestran a continuación:

Tabla 33
Métricas de las pruebas.

Métrica	Fórmula	Resultado
Cobertura de pruebas	(Casos ejecutados/Total de casos planteados) X 100	100%
Densidad de defectos	Total defectos / (Total líneas de código/1000)	2,32%
Tasa de éxito de pruebas	(Pruebas exitosas / Total de pruebas) X 100	84.62%

Nota. Métricas de relevancia obtenidas con los resultados de las pruebas realizadas.

Defectos y recomendaciones

Los defectos encontrados representan una amenaza al correcto funcionamiento del sistema y puede perjudicar su adopción por los usuarios o generar datos e información erróneos causando posibles problemas y daños a la Secretaría de Investigaciones Científicas.

Debido a esto se presentarán todos los defectos encontrados, indicando su prioridad e impacto sobre el sistema para tomar acciones correctivas o preventivas según sea la necesidad y la capacidad según los recursos con los que cuente la Secretaría de Investigaciones Científicas.

Se presentan a continuación los defectos encontrados después de la ejecución de pruebas y se detallan acciones recomendadas para implementar y solventar los problemas detectados:

Tabla 34
Defectos y recomendaciones.

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
DEF-01	CP-01	No hay información de ayuda en forma de texto o emergente cuando el usuario omite alguna de las credenciales o son incorrectas.	Media	Marcelo Alcántara	Usuarios y permisos	Personalizar y mostrar mensajes que indiquen al usuario la omisión de datos o datos incorrectos de credenciales.
DEF-02	CP-02	En el formulario de registro no existen mensajes de ayuda indicando errores o falta de datos en el formulario de registro.	Media	Marcelo Alcántara	Usuarios y permisos	Personalizar y mostrar mensajes que indiquen al usuario la omisión de datos de registro o datos incorrectos.
DEF-03	CP-03	No es posible validar que la cuenta se bloquea solo se valida que no se permite el inicio de sesión, tampoco existe un mensaje visual que indique al usuario que se ha bloqueado el usuario o cuenta tras múltiples intentos fallidos de sesión.	Alta	Marcelo Alcántara	Usuarios y permisos	Validar el bloqueo de inicio de sesión de forma consecutiva con credenciales incorrectas o mostrar al usuario una ayuda que indique que está ingresando credenciales incorrectas.
DEF-04	CP-04	Se observa la falta de mensajes indicativos en forma de texto o emergente que muestran alguna omisión o error con los campos del formulario de registro. Además, no existe validación de caracteres especiales en el ingreso de datos. Aunque el sistema ignora las sentencias utilizadas para la ejecución de SQLi o Cross-	Alta	Marcelo Alcántara	Usuarios y permisos	Personalizar y mostrar mensajes de error que indiquen al usuario si existe omisión o error en los datos de registro. Validación en frontend y backend de los datos de entrada evitando que se pueda registrar con nombres con caracteres especiales.

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
		Site Script (XSS) sin afectar al sistema ni comprometer su seguridad si permite el almacenar las sentencias utilizadas lo que afecta a la base de datos referente a la sanidad de la información que se está almacenando.				
DEF-05	CP-05	No es posible validar que la estructura del sistema fuerce la navegación segura mediante HTTPS en todas las rutas	Alta	Marcelo Alcántara	Usuarios y permisos	Se recomienda la ejecución de la prueba posterior a la instalación del sistema en servidor para validar la navegación HTTPS.
DEF-06	CP-06	El sistema permitió el acceso a una URL no firmada en lugar de bloquearla con un error HTTP 403 como se esperaba.	Alta	Marcelo Alcántara	Usuarios y permisos	Se debe verificar las rutas e implementación del middleware aumentando la seguridad del sistema al permitir la validación de las rutas al sistema.
DEF-07	CP-07	El sistema no implementa la validación de contraseña.	Alta	Marcelo Alcantara	Usuarios y Permisos	Se debe implementar la validación de contraseña como método de autenticación al acceder a funciones críticas tales como la modificación de roles o permisos.
DEF-08	CP-09	La Prueba automatizada ingresó los datos correctamente pero no muestra el mensaje de éxito ni queda cargada la sección correspondiente una vez ingresada la nueva información en los campos para las secciones del Curriculum Datos personales, Docencia,	Media	Oscar Leonardo Serpas	Perfil del investigador	Cuando se ingresan los datos correspondientes curriculum en las secciones Datos personales, Docencia, Investigaciones, Proyección Social y Reconocimiento, el sistema debería mostrar un mensaje claro de éxito de inserción para la sección correspondiente y dicha sección

ID Defecto	ID Caso de prueba	Descripción	Severi-dad	QA Asignado	Módulo	Recomendación
		Investigaciones, Proyección Social y Reconocimiento.				debería mostrar la tabla con el nuevo registro ingresado, actualmente carga la vista Datos personales general.
DEF-09	CP-11	La Edición de los datos se efectúa correctamente con datos válidos, pero no muestra mensaje de confirmación ni permanece en la vista relacionada tras la modificación para las secciones del Curriculum: Datos personales, Docencia, Investigaciones, Proyección Social y Reconocimiento.	Media	Oscar Leonardo Serpas	Perfil del investigador	Cuando se modifican los datos correspondientes curriculum en las secciones Datos personales, Docencia, Investigaciones, Proyección Social y Reconocimiento, el sistema debería mostrar un mensaje claro de éxito de edición para la sección correspondiente y dicha sección debería mostrar la tabla con el nuevo registro ingresado, actualmente carga la vista Datos personales general.
DEF-10	CP-14	La eliminación de los datos se efectúa correctamente, pero no muestra mensaje de confirmación ni permanece en la vista relacionada tras la modificación para las secciones del Curriculum: Datos personales, Docencia, Investigaciones, Proyección Social y Reconocimiento.	Media	Oscar Leonardo Serpas	Perfil del investigador	Cuando se Eliminan los registros correspondientes al currículum en las secciones Datos personales, Docencia, Investigaciones, Proyección Social y Reconocimiento, el sistema debería mostrar un mensaje claro de éxito de eliminación para la sección correspondiente y dicha sección debería mostrar la tabla con el registro eliminado, actualmente carga la vista Datos personales general.

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
DEF-11	CP-20	La tabla publicacion no se adapta a una resolución baja, y pierde consistencia, se sale del margen de la página.	Media	Oscar Leonardo Serpas	Perfil del investigador	Modificar el CSS o el contenedor de la tabla para que se acople a la página, tal como ya lo hace la tabla Patente, en la misma sección
DEF-12	CP-22	En la sección “Editar publicación” no carga correctamente todos los campos, esto puede confundir en saber cuál era el valor anterior del campo “tipo”	Alta	Oscar Leonardo Serpas	Perfil del investigador	Se debe modificar la vista para que tome el valor que contiene en la base de datos antes de editar, para el registro Publicación ya que puede causar confusión en el usuario.
DEF-13	CP-31	Al intentar editar “Habilidad Informática” el segundo campo no cuenta con título ni carga automáticamente el valor que tiene en la base de datos.	Alta	Oscar Leonardo Serpas	Perfil del investigador	Se debe modificar la vista para que en el campo sin nombre tome el valor que contiene en la base de datos antes de editar, además se le debe agregar título al campo ya que puede causar confusión al usuario
DEF-14	CP-31	Al intentar editar “Idioma” los campos “Compresión auditiva y oral” y “Comprensión lectura y escritura” no cargan automáticamente el valor que tiene en la base de datos, además que para el segundo campo está mal escrito la palabra correcta es “Comprensión”	Alta	Oscar Leonardo Serpas	Perfil del investigador	Se debe modificar la vista para que en los campos tomen el valor que contienen en la base de datos antes de editar, además se le debe arreglar el título al segundo campo ya que está mal escrito todo esto puede causar confusión al usuario.
DEF-15	CP-37	Al Intentar crear un proyecto sin convocatoria activa para el año actual da error y saca del sistema con error crítico de Laravel	Alta	Oscar Leonardo Serpas	Proyectos de investigación	Se debe poner un error controlado que diga que el Proyecto no se puede ingresar ya que no hay convocatoria activa. El usuario no puede

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
						saber que error se está generando si el sistema no especifica.
DEF-16	CP-38	En la pantalla de edición del Título del proyecto, no hay botón de "Editar" solo hay botón de "Finalizar"	Alta	Oscar Leonardo Serpas	Proyectos de investigación	Se debe agregar un botón para "Editar" la sección "título" del proyecto ya que actualmente solo se puede actualizar presionando el botón "Finalizar" pero eso hace que el proyecto avance en su finalización y podría ser que no se quiera hacer eso, si no tan solo editar la información.
DEF-17	CP-40	En la pantalla de confirmación para la eliminación de un objetivo no aparece el tipo si no que aparece el id asociado a ese tipo	Baja	Oscar Leonardo Serpas	Proyectos de investigación	Se debe poner el nombre asociado al ID del campo tipo, para el ID 1 es General y el ID 2 es Específico.
DEF-18	CP-42	En la tabla de contrataciones de personal de investigación, no se visualizan los campos "Fuente" y "Monto Fuente" tanto cuando se ingresan por primera vez como cuando se editan	Media	Oscar Leonardo Serpas	Proyectos de investigación	La tabla contrataciones de personal de investigación, debe arreglarse para que muestre los datos correspondientes a "Fuente" y "Monto Fuente".
DEF-19	CP-41	Poca información de ayuda para la navegación del usuario.	Baja	David Castro	Proyectos de Investigación	Personalizar mensajes de ayuda para el usuario.
DEF-20	CP-43	• Las fuentes de financiamiento externas a la UES no aparecen	Media	David Castro	Proyectos de Investigación	Detallar correctamente la fuente de financiamiento requerida para cada sección de

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
		correctamente a la hora de editar campos. • Las fuentes de financiamiento para cada paso se deben de explicar cuáles son las necesarias.				un proyecto y mostrar mensajes de ayuda para los usuarios.
DEF-21	CP-44	En viáticos nacionales al finalizar y mostrar el mensaje de éxito, aunque el mensaje si se muestra, pero se ve en la pantalla de viáticos internacionales, y debería verse en la gestión de viáticos nacionale	Baja	Oscar Leonardo Serpas	Proyectos de investigación	Al terminar de eliminar viáticos Nacionales el mensaje debe mostrarse en su pantalla de gestión correspondiente, no cambiar a Viáticos Internacionales.
DEF-22	CP-45	La eliminación no funciona y no muestra ninguna información sobre el error.	Alta	David Castro	Proyectos de Investigación	Corregir el error para la eliminación de un proyecto y mostrar un mensaje de éxito o error más detallado.
DEF-23	CP-49	El redireccionamiento devuelve a la página principal y no al listado de proyectos.	Baja	David Castro	Proyectos de Investigación	Cambiar el redireccionamiento al listado de proyectos.
DEF-24	CP-51	El proyecto automáticamente pasa a proyectos antiguos después de enviarlo a revisión.	Baja	David Castro	Proyectos de Investigación	Mostrar un mensaje informativo sobre el cambio del estado del proyecto.
DEF-25	CP-55	La visualización dentro del listado de proyectos enviados a revisión no redirige correctamente para mostrar la información del proyecto.	Media	David Castro	Revisión de perfiles	Corregir el funcionamiento del botón Ver para que muestre los detalles del proyecto seleccionado.

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
DEF-26	CP-56	En la pantalla “Proyectos a revisar” no se puede ver el detalle del proyecto recibido para su revisión, Al darle al botón “Ver” no realiza la acción que debería realizar.	Alta	Oscar Leonardo Serpas	Revisión de perfiles	Hablar con el cliente y averiguar cuál es la funcionalidad que le hace falta al presionar el botón “Ver” ya que en este momento no hace nada
DEF-27	CP-59	La filtración de datos solo se puede realizar al comienzo de la consulta.	Baja	David Castro	Reportes	Debe permitir cambiar los filtros después de la primera consulta.
DEF-28	CP-61	El archivo Excel descargado no presenta una estructura correcta o similar a la mostrada por el reporte PDF lo que dificulta su lectura.	Media	David Castro	Reportes	Configurar el archivo Excel con la misma estructura que el mostrado en el sistema y en el reporte PDF.
DEF-29	CP-65	Los cambios no se reflejan en el listado después de actualizar los datos.	Media	David Castro	Inventario	En el listado de recursos del Inventario deben de reflejarse los cambios al actualizar un recurso y debe ser un requisito para mostrar el mensaje de éxito.
DEF-30	CP-67	Interfaz de botones poco intuitivos.	Baja	David Castro	Inventario	Colocar un mensaje emergente al colocar el cursor sobre el botón en acciones para brindar mayor información al usuario o una leyenda a cada botón.
DEF-31	CP-71	• En la consola del navegador aparecen algunos errores que no afectan al funcionamiento pero es indicativo de un posible defecto.	Baja	David Castro	Transversal	• Corregir los errores mostrados en la consola del navegador.

ID Defecto	ID Caso de prueba	Descripción	Severidad	QA Asignado	Módulo	Recomendación
		• No hay mensaje de éxito, error o bienvenida cuando se inicia sesión. • El financiamiento externo no se visualiza correctamente.				• Mostrar mensajes de éxito y error al iniciar sesión (en caso de error indicar que campo es incorrecto). • Corregir valores mostrados en el campo de financiamiento externo cuando su valor es 0.
DEF-32	CP-72	En un dispositivo móvil (Android o iOS) se presentan inconsistencias y problemas visuales en la interfaz (botones mal colocados, elementos encima de otros, elementos no legibles, entre otros).	Media	David Castro	Transversal	Corregir interfaz en dispositivos móviles y agregar responsividad e imágenes de diferentes resoluciones según el tamaño del dispositivo.
DEF-33	CP-74	El financiamiento externo no se muestra correctamente en el navegador.	Baja	David Castro	Transversal	Corregir el campo financiamiento externo cuando su valor es 0 en el navegador.
DEF-34	CP-75	No se indica los requisitos de una convocatoria activa	Baja	David Castro	Transversal	Mostrar mensaje de la convocatoria actual que está activa.
DEF-35	CP-77	El tiempo de respuesta en dispositivos móviles muestra un rendimiento muy inferior al escritorio, debe optimizarse para mejorar los tiempos de respuesta.	Baja	David Castro	Transversal	Optimizar el consumo de datos y recursos del sistema en dispositivos móviles y almacenar en caché componentes de la interfaz.
DEF-36	CP-78	El menú lateral solo permite navegar entre las opciones principales.	Baja	David Castro	Transversal	Permitir la navegación con el teclado dentro de las opciones secundarias del menú lateral.

VII. Conclusiones

La evaluación del Sistema Informático para la Gestión de Convocatorias y Registro de Proyectos de Investigación permitió determinar que la plataforma posee un nivel de madurez funcional apto para las operaciones de la Secretaría de Investigaciones Científicas. Al aplicar las métricas de la norma ISO/IEC 25010, se validó que en una fase previa a la puesta en un ambiente productivo el sistema cumple con los objetivos de funcionalidad y fiabilidad propuestos originalmente, garantizando una administración centralizada que agiliza el flujo de convocatorias académicas dentro de la Universidad de El Salvador.

En relación con el problema de investigación, la ejecución de este proyecto resolvió la carencia de documentación técnica sobre el comportamiento del software. La identificación sistemática de defectos críticos mediante matrices de prueba transformó la incertidumbre operativa en un diagnóstico técnico objetivo. Este proceso no solo permitió corregir fallos en la interacción de componentes, sino que también aseguró que la adopción del sistema por parte de la Secretaría esté respaldada por evidencias de calidad y seguridad.

El aporte fundamental de esta investigación reside en la entrega de un marco de trabajo de Aseguramiento de Calidad (QA) adaptado a las necesidades institucionales. La documentación técnica y las estrategias de verificación generadas constituyen una línea base esencial para el mantenimiento evolutivo del sistema. Finalmente, se concluye que, para garantizar la escalabilidad y eficiencia operativa a largo plazo, es imperativo transitar hacia la automatización de pruebas de regresión, mitigando así los riesgos asociados a futuras actualizaciones de la plataforma.

VIII. Glosario

Acuerdos de Nivel de Servicio (SLA): Es un contrato o compromiso formal entre un proveedor de servicios y un cliente, donde se definen los requerimientos del servicio que deben cumplirse y las consecuencias del incumplimiento de los requisitos.
Ambiente de pruebas: Ambiente controlado que simula las condiciones del entorno de producción, utilizado para realizar verificaciones y validaciones del sistema.
Análisis estático: Consiste en la evaluación del código fuente, documentación o artefactos de un sistema sin ejecutarlo, con el fin de detectar defectos, vulnerabilidades, incumplimientos de estándares o posibles mejoras.
Auditoría: Es el proceso sistemático y documentado mediante el cual se examinan y evalúan los procedimientos, productos y actividades de un sistema o proyecto para verificar su conformidad con normas, políticas o requisitos establecidos.
Automatización: Se refiere al uso de herramientas o software para ejecutar tareas repetitivas de manera automática, reduciendo la intervención humana.
Backlog: Lista priorizada de tareas, requisitos o funcionalidades pendientes de realizar en un proyecto.
Calidad: Es el grado en que un producto o servicio cumple con los requisitos especificados y las expectativas del usuario.
Causa raíz: Es el motivo fundamental que origina un problema o fallo.
Código abierto: Se trata de un modelo de desarrollo y distribución de software en el que el código fuente es accesible, modificable y redistribuible por cualquier persona bajo licencias abiertas.
Consejo Ejecutivo de Investigaciones (CEI): Es el organismo responsable de promover, coordinar y ejecutar la política de desarrollo científico y tecnológico de la Universidad de El Salvador.
Control de calidad: Conjunto de actividades y procesos destinados a verificar que un producto o servicio cumple los estándares y requisitos establecidos.
Defecto: Comportamiento incorrecto o inesperado encontrado en un componente de un sistema.
Diagrama de caja negra: Es una representación gráfica de un sistema que muestra únicamente sus entradas, salidas y la relación funcional entre ellas, sin describir el funcionamiento interno o proceso.
Entorno de producción: Ambiente real en el que el sistema está disponible para los usuarios finales y donde se ejecutan las transacciones u operaciones diarias del sistema.
Fallo: Se trata de un evento o comportamiento incorrecto que ocurre cuando un sistema o algún componente no funciona como se espera.
Gestor de seguimiento: Persona encargada de monitorear y verificar los avances de un proyecto de investigación.
Hardware: Conjunto de componentes físicos y tangibles que forman parte de un sistema informático
Historia de usuario: Descripción breve y simple de una necesidad o funcionalidad desde la perspectiva del usuario o cliente
ISO/IEC: Conjunto de normas internacionales desarrolladas por la International Organization for Standardization (ISO) y la International Electrotechnical Commission (IEC), que establecen estándares para la gestión de calidad y desarrollo de software.
ISTQB: Sus siglas se refieren a la International Software Testing Qualifications Board Organización, una organización internacional que define estándares y certificaciones globales en pruebas de software.
Matriz de pruebas: Es una herramienta que relaciona los requisitos del sistema con los casos de prueba diseñados para validarlos.
Métricas: Son medidas cuantitativas utilizadas para evaluar aspectos del proceso o producto como calidad, rendimiento, defectos, esfuerzo o cumplimiento de requisitos.
Módulo: Es un componente independiente dentro de un sistema de software que realiza una tarea específica y puede interactuar con otros módulos.

Plan de Pruebas: Es un documento que describe el enfoque, alcance, recursos, cronograma y actividades de prueba a realizar en un proyecto.
Proyectos de investigación: Es un trabajo sistemático y planificado que busca generar nuevo conocimiento o resolver un problema específico.
Prueba: Proceso mediante el cual se ejecuta un sistema o componente con el fin de identificar defectos.
Reporte: Es un documento donde se presentan los resultados obtenidos de una actividad, como pruebas o auditorías, incluyendo observaciones, defectos detectados y conclusiones.
Requisitos: Son las necesidades o condiciones establecidas que debe cumplir un sistema o componente de software.
Resumen ejecutivo: Síntesis breve y clara de un informe, proyecto o propuesta que presenta los puntos clave, conclusiones y recomendaciones,
Secretaría de Investigaciones Científicas: La Secretaría de Investigaciones Científicas de la Universidad de El Salvador (SIC-UES) es la unidad orgánica encargada de coordinar los esfuerzos de la investigación científica que realiza la Universidad de El Salvador.
Sistema: Conjunto de componentes interrelacionados, tanto de hardware como de software, que trabajan conjuntamente para cumplir un objetivo o resolver un problema específico.
Software: Conjunto de programas, aplicaciones y datos que permiten que un sistema informático realice tareas específicas.
Stakeholder: Es la persona o grupo que tiene interés, influencia o se ve afectado por el desarrollo o resultados de un proyecto.
Tesina: Trabajo de investigación académica que presenta los resultados de un estudio o proyecto desarrollado.
Trazabilidad: Es la capacidad de seguir y relacionar los elementos de un proyecto como requisitos, cambios, pruebas y la coherencia entre ellos.
UI: Son los elementos visuales y de interacción mediante los cuales una persona se comunica con un sistema o aplicación.
Usuario: Persona o entidad que utiliza o interactúa con un sistema informático.
UX: Es el grado de satisfacción que tiene un usuario al interactuar con un producto o servicio, considerando aspectos como utilidad, facilidad de uso y accesibilidad.
Workaround: Solución temporal que permite sortear un problema o fallo cuando no es posible aplicar una corrección definitiva de inmediato.

IX. Referencias

Alvarado Maldonado, K. C. J., Guzmán, A. R. I., Ordoñez Morales, B. O., & Melgar Morales, J. A. H. (2022). *Planificación y diseño de una plataforma WEB para la gestión de procesos de investigación de un sistema integrado de información para la Dirección General de Investigación de la Universidad de San Carlos de Guatemala*. Guatemala: Universidad de San Carlos de Guatemala. https://catalogosiidca.csuca.org/Record/USAC.646482

Atlassian. (s. f.). *Los distintos tipos de pruebas en software Atlassian*. https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

Azabache Martínez, G. (2023). *La calidad en el software: Definiciones*. Brave Developer. https://bravedeveloper.com/2023/05/30/la-calidad-en-el-software-definiciones/

Bernstein, C. (2024, 17 octubre). *¿Qué es la prueba de ruta feliz?* Search Software Quality. https://www.techtarget.com/searchsoftwarequality/definition/happy-path-testing

CIC-UES. (2016). *Reglamento del Consejo de Investigaciones Científicas (CIC-UES)*. Universidad de El Salvador. https://humanidades.ues.edu.sv/wp-content/uploads/sites/7/2024/12/Reglamento-del-Consejo-de-Investigaciones-Cientificas.pdf

CircleCI. (2025). *¿Qué es la prueba de software?* https://circleci.com/es/topics/software-testing/

DAU. (s. f.). *Test Readiness Review (TRR)*. Defense Acquisition University (DAU). https://www.dau.edu/acquipedia-article/test-readiness-review-trr

Departamento de Informática. Facultad de Ciencias Exactas y Naturales y Agrimensura. (2012). *Calidad de software e Ingeniería de Usabilidad*. Universidad Nacional del Nordeste https://sedici.unlp.edu.ar/bitstream/handle/10915/19202/Documento_completo.pdf?sequence=1&isAllowed=y

ECCMA. (2025). *Principios: Definiendo datos de Calidad. ISO 9001:2005*. https://eccmatraining.com/MDQM_Certification/3_Principles1_Defining_Quality/ES/?page=4

Hernández, C. H., & Menjívar, W. M. (2025). *SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONVOCATORIAS Y REGISTRO DE PROYECTOS DE INVESTIGACIÓN PARA LA SECRETARÍA DE INVESTIGACIONES CIENTÍFICAS DE LA UNIVERSIDAD DE EL SALVADOR*. Tesis de Pregrado, Universidad de El Salvador.

International Software Testing Qualifications Board. (2021). *ISTQB - Certified Tester Foundation Level. Syllabus* (versión 3.1.1). https://istqb.org/wp-content/uploads/2024/11/ISTQB-CTFL_Syllabus_2018_v3.1.1.pdf

International Software Testing Qualifications Board. (2024). *ISTQB - Certified Tester Foundation Level. Syllabus* (versión 4.0.1). https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf

ISO/IEC 25010:2011. (2011). *Ingeniería de sistemas de software – Requisitos de calidad y evaluación de sistemas y software*. https://es.scribd.com/document/775921734/iso-25010-en-es

NTA. (2017). *Normas Técnicas Y Administrativas del Consejo de Investigaciones Científicas de la Universidad De El Salvador CIC-UES.* Universidad de El Salvador. https://sic.ues.edu.sv/storage/app/media/normastecnicas.pdf

OECD. (2015). *Directrices para la recopilación y presentación de informes de datos sobre investigación y desarrollo experimental*. Manual de Frascati. https://www.oecd.org/content/dam/oecd/es/publications/reports/2015/10/frascati-manual-2015_g1g57dcb/9789264310681-es.pdf

Pacific Certifications. (2025). *ISO/IEC 25010:2023 Software Quality Model explained*. Pacific Certifications. https://pacificcert.com/iso-iec-25010-systems-and-software-engineering/

Portal ISO 25000. (s. f.). *ISO 25010*. https://iso25000.com/index.php/normas-iso-25000/iso-25010

Real Academia Española. (s.f.). *Calidad*. En *Diccionario de la lengua española* (23.ª ed.). https://dle.rae.es/calidad

Sánchez Peño, J. S. (2015). *Pruebas de Software. Fundamentos y Técnicas* [Universidad Politécnica de Madrid]. https://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf

SIC-UES. (2022). *Secretaría de Investigaciones Científicas SIC-UES*. Universidad de El Salvador. https://sic.ues.edu.sv/sicues

Solorzano, J. S. (2022). *Tipos y niveles de pruebas*. Tipos y Niveles de Pruebas. https://medium.com/nicasource/tipos-y-niveles-de-prueba-1a276651a60d

Son, H. (2025, 22 noviembre). Software Testing Life Cycle (STLC): Best Practices for Optimizing Testing - TestRail. *TestRail | The Quality OS for QA Teams*. https://www.testrail.com/blog/software-testing-life-cycle-stlc/

UCR. (2010). *Investigación lanza nuevo sistema de información de proyectos. Universidad de Costa Rica.* https://www.ucr.ac.cr/noticias/2010/11/29/investigacion-lanza-nuevo-sistema-de-informacion-de-proyectos.html

X. Anexos

Anexo A: Técnicas de análisis para el planteamiento del problema.

Matriz FODA

Esta herramienta permitió estructurar estratégicamente las condiciones del entorno y del equipo evaluador. La matriz fue construida mediante revisión documental, sesiones de trabajo colaborativo y análisis preliminar del sistema.

Figura 8
Matriz FODA.

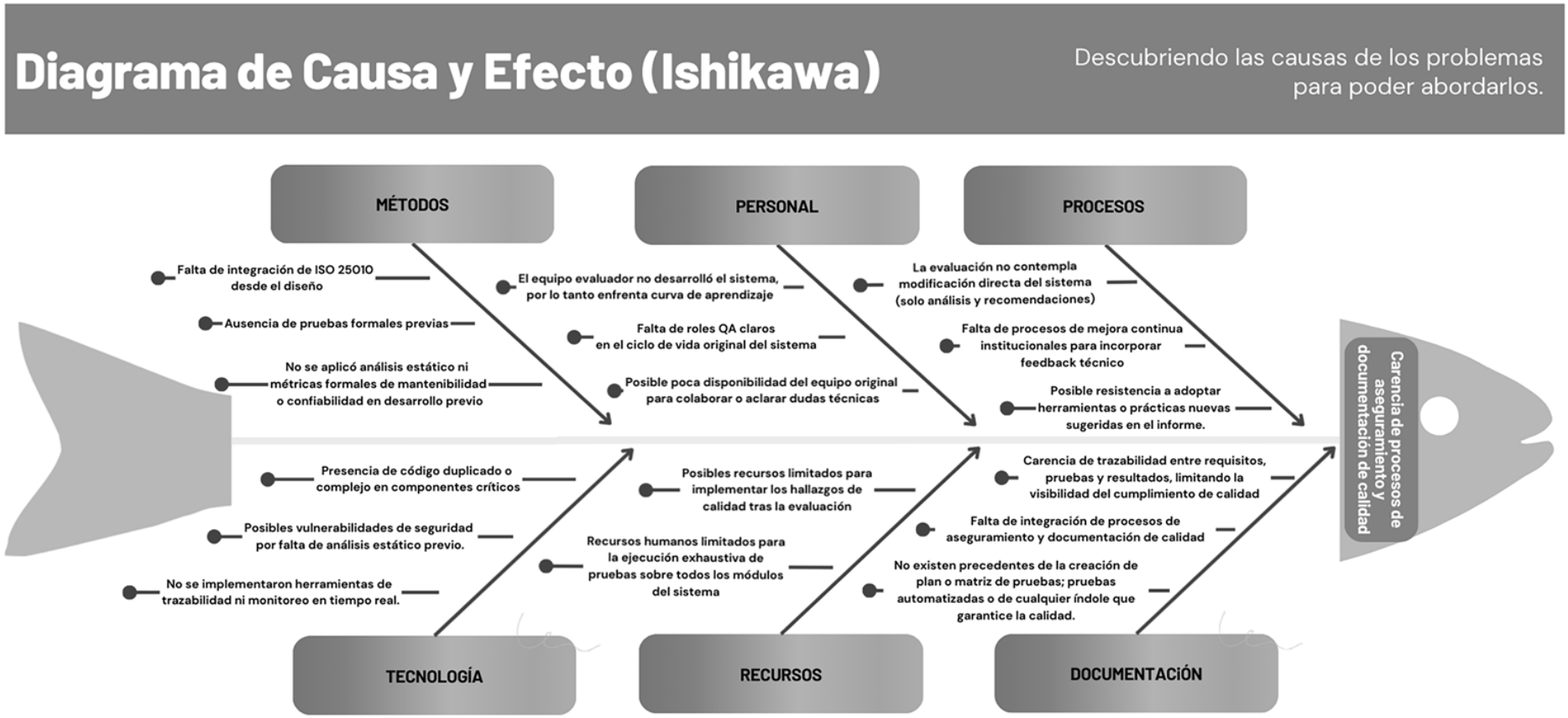
Fortalezas	**Oportunidades**
Se pretende usar una metodología formal, estándar ISO/IEC 25010 aplicada a pruebas de calidad para una evaluación estructurada y reconocida	Fomentar cultura de calidad dentro de la Secretaría de Investigaciones Científicas.
Aplicación de análisis estático y pruebas automatizadas con herramientas modernas	Posibilidad de encontrar fallos no detectados previamente y proponer mejoras técnicas.
Equipo académico acceso al código fuente y documentación	Generar insumos valiosos para decisiones institucionales y futuras auditorías.
Gestión de la matriz de pruebas mediante herramientas.	Aporte académico aplicable a otros sistemas institucionales similares.
Debilidades	**Amenazas**
No se aplicarán cambios directos al sistema tras los hallazgos	Dificultad para obtener acceso completo o colaboración directa del equipo original que creó el sistema a analizar.
Acceso limitado solo al subsistema de registro y componentes críticos del mismo.	Desactualización de tecnologías y partes del sistema frente a estándares de calidad actuales.
Dependencia de documentación y comprensión de un sistema que no fue desarrollado por el grupo actual de trabajo.	La Secretaría de Investigaciones Científicas podría no implementar recomendaciones por falta de recursos o debido al costo de la implementación de las mejoras basadas en la norma ISO/IEC 25010
Recursos humanos limitados para automatización y pruebas profundas en todo el sistema. Se deberán priorizar las partes más críticas.	Cambios en políticas o infraestructura tecnológica que afecten la validez del análisis.

Diagrama de Causa y Efecto (Ishikawa)

El diagrama de causa y efecto (también conocido como Diagrama de Ishikawa o diagrama de pescado) es una herramienta eficaz para la resolución de problemas. En lugar de centrarse en una solución rápida, este diagrama permite a los equipos realizar una lluvia de ideas e identificar la causa raíz de un problema para encontrar una solución a largo plazo. En la elaboración de este diagrama identificamos el problema principal en la cabeza del pescado y en base a ese problema se va desarrollando las posibles causas y efectos de manera detallada por toda la estructura en forma de espinas de pescado.

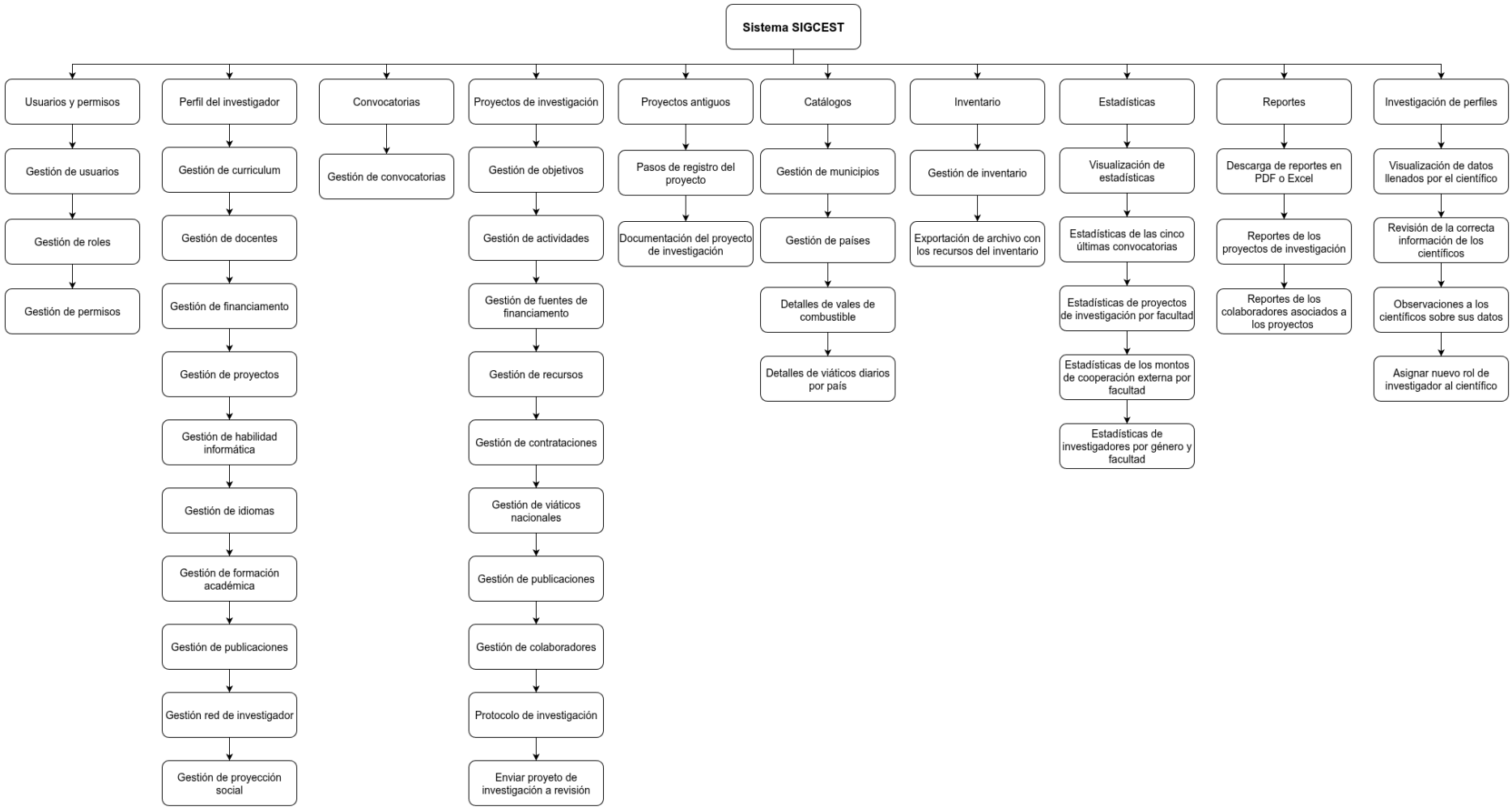
Figura 9

Diagrama de causa y efecto (Ishikawa).



Anexo B: Diagrama de módulos del sistema.

Figura 10
Diagrama de módulos del sistema.



Anexo C: Evaluación para automatizar pruebas.

1. Objetivo

Establecer un criterio técnico y objetivo para seleccionar qué casos de prueba del Sistema de Gestión de Convocatorias y Registro de Proyectos de Investigación son candidatos idóneos para la automatización con la herramienta Laravel Dusk, y cuáles deben ejecutarse manualmente, optimizando así los recursos del proyecto y asegurando la cobertura de riesgos críticos.

2. Metodología de Evaluación

En este paso cada caso de prueba fue evaluado bajo tres criterios ponderados, asignando un valor del 1 al 3 (donde 3 es alto y 1 es bajo):

- **Frecuencia de Ejecución / Regresión (Peso: 40%):** ¿Cuántas veces se necesita ejecutar esta prueba? (3: En cada despliegue; 1: Solo una vez).
- **Criticidad del Negocio (Peso: 40%):** ¿Qué impacto tiene si esta función falla? (3: Bloqueante/Crítico; 1: Cosmético).
- **Complejidad de Automatización (Peso: 20%):** ¿Qué tan difícil es crear el script en Laravel Dusk? (1: Muy complejo/Requiere intervención humana; 3: Estándar/Fácil).

Fórmula de Prioridad: Puntuación = (Frecuencia + Criticidad) - Complejidad Inversa

Para efectos de este trabajo, se utilizó un enfoque cualitativo basado en la siguiente matriz de decisión:

3. Criterios de Selección

Tabla 35
Matriz de decisión.

Categoría	Criterio	Decisión
Candidato Por Automatizar	Alta repetitividad (regresión) + alta criticidad + flujo estable.	Se automatiza (Laravel Dusk)
Prueba Manual	Baja frecuencia + alta complejidad técnica (ej. validación visual, correos externos) + valoración humana requerida.	Se ejecuta manualmente

1. Resumen de la Evaluación Aplicada

A continuación, se presenta la justificación técnica de los grupos de casos seleccionados:

Tabla 36
Justificación técnica.

Grupo de Pruebas	Evaluación de Viabilidad	Decisión Tomada
Login y Sesión (CP-01, CP-03, CP-05)	Alta Viabilidad. Son las pruebas más repetitivas. Si fallan, nadie entra al sistema. Fácil en complejidad de hacer el código con Laravel Dusk.	Automatizado
Registro de Usuario (CP-02, CP-04)	Baja Viabilidad. Requiere validación de correos electrónicos y posible manejo de tokens únicos que complican el script. Se prioriza la validación manual.	Manual
Gestión de Proyectos "Happy Path" (CP-36, CP-38, CP-48)	Alta Viabilidad. Es el corazón del sistema SIC-UES. Se requiere asegurar que el flujo principal nunca se rompa tras cambios en el código.	Automatizado
Cargas de Archivos y UI Específica (CP-09, CP-13)	Media/Baja Viabilidad. La interacción con el sistema de archivos del sistema operativo y la validación visual de fotos/PDFs es propensa a falsos positivos en automatización.	Manual
Compatibilidad (CP-71 a CP-78)	Nula Viabilidad (para Dusk). Laravel Dusk usa ChromeDriver por defecto. Validar renderizado en Safari/Firefox y móviles requiere inspección visual humana.	Manual

Anexo D: Scripts de Automatización.

Script CP-01: LoginTest.php

```
<?php
namespace Tests\Browser;

use Laravel\Dusk\Browser;
use Tests\DuskTestCase;
use App\Models\User;
use Illuminate\Support\Facades\Artisan;

class LoginTest extends DuskTestCase
{
   /**
    * Base URL para todas las pruebas.
    *
    * @var string
    */
   protected $baseUrl = 'http://127.0.0.1:8000';

   private const PATH_LOGIN = '/login';
   private const PATH_LOGIN_CORRECTO='/datosPersonales';
   private const EMAIL = 'input[name="email"]';

   protected function setUp(): void
   {
      parent::setUp();
      Artisan::call('cache:clear');
   }

   public function test_usuario_valido_puede_iniciar_sesion()
   {
      $this->browse(function (Browser $browser) {
         $browser->driver->manage()->deleteAllCookies();
         $browser->visit($this->baseUrl . self::PATH_LOGIN)
               ->waitFor(self::EMAIL, 10)
```

```
            ->type('email', 'a@a')
            ->type('password', '12345678')
            ->press('Entrar')
            ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(self::PATH_LOGIN_CORRECTO);
    });
}

public function test_contraseña_incorrecta_no_permite_acceso()
{
    $this->browse(function (Browser $browser) {
        $browser->driver->manage()->deleteAllCookies();
        $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type('email', 'a@a')
            ->type('password', 'contraseña_incorrecta')
            ->press('Entrar')
            ->pause(1000)
            ->assertPathIs(self::PATH_LOGIN)
            ->assertSee('Estas credenciales no coinciden con nuestros registros');
    });
}
public function test_envio_campos_vacios_muestra_errores()
{
    $this->browse(function (Browser $browser) {
        $browser->driver->manage()->deleteAllCookies();
        $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->press('Entrar')
            ->pause(1000)
            ->assertPathIs(self::PATH_LOGIN)
            ->assertSee('El campo correo electrónico es obligatorio')
            ->assertSee('El campo contraseña es obligatorio');
    });
}
```

```
    public function test_usuario_ya_autenticado_accede_home_directamente()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();
            $browser->visit($this->baseUrl . self::PATH_LOGIN)
                    ->waitFor(self::EMAIL, 10)
                    ->type(self::EMAIL, 'a@a')
                    ->type('input[name="password"]', '12345678')
                    ->press('Entrar')
                    ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
                    ->assertPathIs(self::PATH_LOGIN_CORRECTO);

            $browser->visit($this->baseUrl . self::PATH_LOGIN)
                    ->pause(1000)
                    ->assertPathIs(self::PATH_LOGIN_CORRECTO);
        });
    }
    public function test_usuario_no_registrado_no_permite_acceso()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();
            $browser->visit($this->baseUrl . self::PATH_LOGIN)
                    ->waitFor(self::EMAIL, 10)
                    ->type('email', 'noexiste@dominio.com')
                    ->type('password', 'cualquier')
                    ->press('Entrar')
                    ->pause(1000)
                    ->assertPathIs(self::PATH_LOGIN)
                    ->assertSee('Estas credenciales no coinciden con nuestros registros');
        });
    }
}
```

Script CP-03: SeguridadAutenticacionTest.php

```
<?php
```

```
namespace Tests\Browser;

use Laravel\Dusk\Browser;
use Tests\DuskTestCase;

class SeguridadAutenticacionTest extends DuskTestCase
{
    protected $baseUrl = 'http://127.0.0.1:8000';

    private const EMAIL = 'input[name="email"]';
    private const PATH_LOGIN = '/login';
    private const CONTRASENA= 'input[name="password"]';
    private const PATH_LOGIN_CORRECTO='/datosPersonales';

    public function test_bloqueo_por_intentos_fallidos()
    {
        $this->browse(function (Browser $browser) {
            for ($i = 0; $i < 5; $i++) {
                $browser->visit($this->baseUrl . self::PATH_LOGIN)
                    ->waitFor(self::EMAIL, 10)
                    ->type(self::EMAIL, 'a@a')
                    ->type(self::CONTRASENA, 'incorrecta')
                    ->press('Entrar')
                    ->pause(1000)
                    ->assertPathIs(self::PATH_LOGIN);
            }

            $browser->visit($this->baseUrl . self::PATH_LOGIN)
                ->type(self::EMAIL, 'a@a')
                ->type(self::CONTRASENA, 'incorrecta')
                ->press('Entrar')
                ->pause(1500)
                ->waitForText('demasiados intentos', 5)
                ->assertSee('demasiados intentos');
        });
```

```
}

public function test_sesiones_simultaneas_en_dos_navegadores()
{
    $this->browse(function (Browser $first, Browser $second) {
        $first->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'user@correo.com')
            ->type(self::CONTRASENA, 'User123!')
            ->press('Entrar')
            ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(self::PATH_LOGIN_CORRECTO);

        $second->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'user@correo.com')
            ->type(self::CONTRASENA, 'User123!')
            ->press('Entrar')
            ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(self::PATH_LOGIN_CORRECTO);

        $first->assertPathIs(self::PATH_LOGIN_CORRECTO);
        $second->assertPathIs(self::PATH_LOGIN_CORRECTO);

        $first->driver->manage()->deleteAllCookies();
        $second->driver->manage()->deleteAllCookies();
    });
}

public function test_expiracion_de_sesion()
{
    $this->browse(function (Browser $browser) {
        $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'tiempo@correo.com')
```

```
            ->type(self::CONTRASENA, 'Timeout456!')
            ->press('Entrar')
            ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(self::PATH_LOGIN_CORRECTO);

        $browser->waitFor('#userDropdown', 10)
            ->click('#userDropdown')
            ->pause(500);

        $browser->click('.dropdown-item[data-target="#logoutModal"]');

        $browser->whenAvailable('#logoutModal', function (Browser $modal) {
            $modal->waitFor('a.btn.btn-danger', 10)
                ->click('a.btn.btn-danger');
        });

        $browser->waitForLocation('/', 10)
            ->assertPathIs('/');

        $browser->driver->manage()->deleteAllCookies();

        $browser->visit($this->baseUrl . self::PATH_LOGIN_CORRECTO)
            ->pause(1000)
            ->assertPathIs(self::PATH_LOGIN);
    });
}

public function test_cookies_tienen_atributos_de_seguridad()
{
    $this->browse(function (Browser $browser) {
        $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'seguridad@correo.com')
            ->type(self::CONTRASENA, 'Segura2025!')
            ->press('Entrar')
```

```
            ->waitForLocation(self::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(self::PATH_LOGIN_CORRECTO);

        $secure  = config('session.secure');
        $httpOnly = config('session.http_only');

        $this->assertTrue(
            $httpOnly,
            'La cookie de sesión no tiene el atributo HttpOnly habilitado en config/session.php.'
        );

        if (app()->environment('production')) {
            $this->assertTrue(
                $secure,
                'La cookie de sesión no tiene el atributo Secure habilitado en producción.'
            );
        }

        $cookies = $browser->driver->manage()->getCookies();
        $sessionCookie = collect($cookies)->firstWhere('name', config('session.cookie'));
        $this->assertNotNull($sessionCookie, 'No se encontró la cookie de sesión.');
    });
  }
}
```

Script CP-05: SeguridadTest.php

```
<?php

namespace Tests\Browser;

use Laravel\Dusk\Browser;
use Tests\DuskTestCase;
use App\Models\User;
use Spatie\Permission\Models\Role;
use Facebook\WebDriver\WebDriverBy;
```

```
class SeguridadTest extends DuskTestCase
{
    /**
     * Base URL para todas las pruebas.
     *
     * @var string
     */
    protected $baseUrl = 'http://127.0.0.1:8000';

    private const PATH_LOGIN = '/login';
    private const PATH_PROTEGIDA='/gestionCurriculum';
    private const PATH_HOME = '/datosPersonales';
    private const USER_EMAIL = 'i@i';
    private const USER_PASSWORD = '12345678';
    private const PATH_ADMIN_ROLES = '/roles';

    protected function setUp(): void
    {
        parent::setUp();
        static::closeAll();

        if (!Role::where('name', 'investigador')->exists()) {
            Role::create(['name' => 'investigador']);
        }
        if (!Role::where('name', 'administrador')->exists()) {
            Role::create(['name' => 'administrador']);
        }

        $user = User::firstOrCreate(
            ['email' => self::USER_EMAIL],
            [
                'name' => 'Usuario Prueba Roles',
                'password' => bcrypt(self::USER_PASSWORD)
            ]
        );
```

```
        if (method_exists($user, 'syncRoles')) {
            $user->syncRoles(['investigador']);
        }
    }

    public function test_redirigir_al_login_sin_sesion()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();

            $browser->visit($this->baseUrl . self::PATH_PROTEGIDA)
                    ->waitForLocation($this->baseUrl . self::PATH_LOGIN, 10)
                    ->assertPathIs(self::PATH_LOGIN);

            $browser->driver->manage()->deleteAllCookies();
        });
    }

    public function test_acceso_permitido_con_sesion_iniciada()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();

            $browser->visit($this->baseUrl . self::PATH_LOGIN)
                    ->waitFor('input[name="email"]', 10)
                    ->type('email', 'a@a')

                    ->type('password', self::USER_PASSWORD)
                    ->press('Entrar')
                    ->visit($this->baseUrl.self::PATH_PROTEGIDA)
                    ->waitForLocation(self::PATH_PROTEGIDA, 10)
                    ->assertPathIs(self::PATH_PROTEGIDA); // Asegura que el usuario accede a la ruta
protegida
        });
```

```
}

public function test_redirigir_a_login_con_sesion_expirada()
{
   $this->browse(function (Browser $browser) {
      $browser->driver->manage()->deleteAllCookies();
      $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor('input[name="email"]', 10)
            ->type('email', 'a@a')
            ->type('password', '12345678')
            ->press('Entrar')
            ->waitForLocation('/datosPersonales', 10);

      $browser->driver->manage()->deleteAllCookies();

      $browser->visit($this->baseUrl . self::PATH_PROTEGIDA)
            ->waitForLocation($this->baseUrl . self::PATH_LOGIN, 10)
            ->assertPathIs(self::PATH_LOGIN);
   });
}

public function test_usuario_sin_permisos_no_puede_acceder_a_gestion_roles()
{
   $this->browse(function (Browser $browser) {
      $browser->driver->manage()->deleteAllCookies();

      $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->type('email', self::USER_EMAIL)
            ->type('password', self::USER_PASSWORD)
            ->press('Entrar')
            ->waitForLocation(self::PATH_HOME, 10);

      $browser->visit($this->baseUrl . self::PATH_ADMIN_ROLES);

      $browser->assertDontSee('Crear Rol')
```

```
            ->assertDontSee('Lista de Roles');

        $textoError = $browser->driver->findElement(WebDriverBy::tagName('body'))->getText();

        $esBloqueado = str_contains($textoError, '403') ||
                str_contains($textoError, 'Unauthorized') ||
                str_contains($textoError, 'does not have the right roles') ||
                str_contains($textoError, 'no autorizado') ||
                str_contains($textoError, 'THIS ACTION IS UNAUTHORIZED');

        $this->assertTrue($esBloqueado, 'FALLO DE SEGURIDAD: El usuario pudo acceder o no se
mostró un error 403 claro.');
    });
  }

  public function test_redirigir_http_a_https()
  {
    $this->browse(function (Browser $browser) {
      $browser->visit('http://127.0.0.1:8000/login')
            ->waitForLocation('https://127.0.0.1:8000/login', 10)
            ->assertUrlIs('https://127.0.0.1:8000/login');
    });
  }
}
```

Script CP-06: SeguridadAccesoDuskTest.php

```
<?php

namespace Tests\Browser;

use Laravel\Dusk\Browser;
use Tests\DuskTestCase;
use Illuminate\Support\Facades\Artisan;

class SeguridadAccesoDuskTest extends DuskTestCase
```

```
{

    protected $baseUrl = 'http://127.0.0.1:8000';

    private const PATH_LOGIN = '/login';
    private const PATH_PROTEGIDO = '/gestionCurriculum';
    private const TEXT_LOGIN_WELCOME = 'Bienvenido';
    private const TEXT_403_ERROR = '403';

    protected function setUp(): void
    {
        parent::setUp();
        Artisan::call('optimize:clear');
    }


    public function test_acceso_a_ruta_protegida_sin_autenticacion()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();

            $browser->visit($this->baseUrl . self::PATH_PROTEGIDO);

            $browser->waitForLocation(self::PATH_LOGIN, 10);
            $browser->assertPathIs(self::PATH_LOGIN);

            $browser->assertSee(self::TEXT_LOGIN_WELCOME);
        });
    }


    public function test_middleware_signed_bloquea_acceso_invalido()
    {
        $this->browse(function (Browser $browser) {
            $browser->driver->manage()->deleteAllCookies();
```

```
        try {
            $urlSinFirma = route('projects.confirm', ['cod' => '123']);
        } catch (\Exception $e) {
            $this->fail('La ruta "projects.confirm" no se encontró en web.php. Revisa tus rutas.');
            return;
        }

        $browser->visit($urlSinFirma);

        $browser->waitForText(self::TEXT_403_ERROR, 10)
                ->assertSee(self::TEXT_403_ERROR)
                ->assertSee('signature');
    });
  }
}
```

Script CP-68: ConvocatoriaTest.php

```
<?php

namespace Tests\Browser;

use Laravel\Dusk\Browser;
use App\Models\User;
use App\Models\UsuarioPrueba; // Asegúrate que la ruta a tu modelo es correcta
use Illuminate\Support\Facades\Storage;
use Tests\DuskTestCase;
use Illuminate\Support\Facades\DB;

class ConvocatoriaTest extends DuskTestCase
{
  /**
   * URL base para las pruebas.
   *
   * @var string
```

```
 */
protected $baseUrl = 'http://127.0.0.1:8000';

private const EMAIL = 'input[name="email"]';
private const PATH_LOGIN = '/login';
private const CONTRASENA_V='12345678';
private const CONTRASENA= 'input[name="password"]';
private const PATH_LOGIN_CORRECTO='/datosPersonales';
private const CODIGO_CONV= 'input[name="codigo"]';

public function test_crear_convocatoria_inactiva()
{
   $newCode = time();

   $this->browse(function (Browser $browser) use ($newCode) {
      $browser->driver->manage()->deleteAllCookies();

      $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'a@a')
            ->type(self::CONTRASENA, self::CONTRASENA_V)
            ->press('Entrar')
            ->waitForLocation(SELF::PATH_LOGIN_CORRECTO, 10)
            ->assertPathIs(SELF::PATH_LOGIN_CORRECTO);

      $browser->visit($this->baseUrl . '/convocatoria')
            ->waitFor(self::CODIGO_CONV, 10)
            ->type(self::CODIGO_CONV, $newCode)
            ->type('input[name="fechainicio"]', '12/11/2025')
            ->type('input[name="fechafin"]', '16/11/2025')
            ->type('input[name="presupuesto"]', '10000')
            ->select('select[name="estado"]', 'Inactiva')
            ->type('textarea[name="observacion"]', 'Observación de prueba Dusk')
            ->press('Crear')
            ->waitForText('Se ha iniciado una nueva convocatoria: '.$newCode)
```

```
            ->assertSee('Se ha iniciado una nueva convocatoria: '.$newCode);
    });
}

public function test_bloquear_creacion_convocatoria_activa_si_existe_otra()
{
    $convocatoriaActiva = DB::table('convocatoria')->where('estadodescr', 'Activa')->first();

    if (!$convocatoriaActiva) {
        $this->markTestSkipped('No existe una convocatoria "Activa" en la BD para probar el bloqueo.');
    }

    $newCode = time() + 1;

    $this->browse(function (Browser $browser) use ($newCode, $convocatoriaActiva) {
        $browser->driver->manage()->deleteAllCookies();

        $browser->visit($this->baseUrl . self::PATH_LOGIN)
                ->waitFor(self::EMAIL, 10)
                ->type(self::EMAIL, 'a@a')
                ->type(self::CONTRASENA, self::CONTRASENA_V)
                ->press('Entrar')
                ->waitForLocation(SELF::PATH_LOGIN_CORRECTO, 10);

        $browser->visit($this->baseUrl . '/convocatoria')
                ->waitFor(self::CODIGO_CONV, 10)
                ->value(self::CODIGO_CONV, $newCode)
                ->value('input[name="fechainicio"]', '2025-10-01')
                ->value('input[name="fechafin"]', '2025-12-01')
                ->value('input[name="presupuesto"]', '5000')
                ->select('select[name="estado"]', 'Activa') // Intentamos crearla como "Activa"
                ->type('textarea[name="observacion"]', 'Observación de prueba Dusk (debe fallar)')
                ->press('Crear')
```

```
            ->waitForText("La siguiente convocatoria está activa: {$convocatoriaActiva-
>numeroconvocatoria}", 10)
            ->assertSee("La siguiente convocatoria está activa: {$convocatoriaActiva-
>numeroconvocatoria}. Debe deshabilitarla primero")
            ->assertPathIs('/convocatoria/show');
    });
  }

  public function test_notificar_convocatoria_activa()
  {
    $convocatoriaActiva = DB::table('convocatoria')
                    ->where('estadodescr', 'Activa')
                    ->first();
    $usuariosPruebaCount = DB::table('users')->count();

    if (!$convocatoriaActiva) {
      $this->markTestSkipped('No existe una convocatoria "Activa" en la BD (columna estadodescr)
para notificar.');
    }
    if ($usuariosPruebaCount === 0) {
      $this->markTestSkipped('No hay usuarios en "usuarios_prueba" para notificar.');
    }

    Storage::disk('public')->put('convocatoria_prueba.pdf', 'Este es un PDF de prueba de Dusk');
    $filePath = storage_path('app/public/convocatoria_prueba.pdf');

    try {
      $this->browse(function (Browser $browser) use ($convocatoriaActiva, $filePath) {
        $browser->driver->manage()->deleteAllCookies();

        $browser->visit($this->baseUrl . self::PATH_LOGIN)
            ->waitFor(self::EMAIL, 10)
            ->type(self::EMAIL, 'a@a')
            ->type(self::CONTRASENA, self::CONTRASENA_V)
            ->press('Entrar')
```

```
                ->waitForLocation(SELF::PATH_LOGIN_CORRECTO, 10);

            $browser->visit($this->baseUrl . '/convocatoria/enviar')
                ->waitFor('select[name="idconvocatoria"]', 10)
                ->select('select[name="idconvocatoria"]', $convocatoriaActiva->idconvocatoria)
                ->type('textarea[name="comentarios"]', 'Prueba de notificación Dusk')
                ->attach('input[name="attachment"]', $filePath)
                ->press('Enviar')
                ->waitForText('Se ha enviado la notificación a todos los usuarios exitosamente', 15)
                ->assertSee('Se ha enviado la notificación a todos los usuarios exitosamente');
        });
    } finally {
        Storage::disk('public')->delete('convocatoria_prueba.pdf');
    }
  }
}
```

Script CP-70: ConvocatoriaDeleteTest.php

```
<?php

namespace Tests\Browser;

use Laravel\Dusk\Browser;
use Tests\DuskTestCase;
use App\Models\User;
use Illuminate\Support\Facades\DB;

class ConvocatoriaDeleteTest extends DuskTestCase
{
  /**
   * URL base para las pruebas.
   *
   * @var string
   */
  protected $baseUrl = 'http://127.0.0.1:8000';
```

```
private const EMAIL = 'input[name="email"]';
private const PATH_CONVOCATORIA_SHOW = "/convocatoria/show";

public function test_permite_eliminar_convocatoria_sin_proyectos_asociados()
{
    $convocatoriaData = [
        'numeroconvocatoria' => time(),
        'estadodescr' => 'Inactiva',
        'fechainicio' => '2025-01-01',
        'fechafin' => '2025-02-01',
        'anoconvocatoria' => 2025
    ];

    $idParaEliminar = DB::table('convocatoria')->insertGetId($convocatoriaData);

    $deleteHref = "/convocatoria/confirm/{$idParaEliminar}";

    try {
        $this->browse(function (Browser $browser) use ($deleteHref) {

            $browser->driver->manage()->deleteAllCookies();
            $browser->visit($this->baseUrl . '/login')
                    ->waitFor(self::EMAIL, 10)
                    ->type(self::EMAIL, 'a@a')
                    ->type('input[name="password"]', '12345678')
                    ->press('Entrar')
                    ->waitForLocation('/datosPersonales', 10);

            $browser->visit($this->baseUrl . self::PATH_CONVOCATORIA_SHOW)
                    ->waitFor("a[href*='{$deleteHref}']", 10)
                    ->click("a[href*='{$deleteHref}']")

                    ->waitForText('Confirme la eliminación de convocatoria', 10)
```

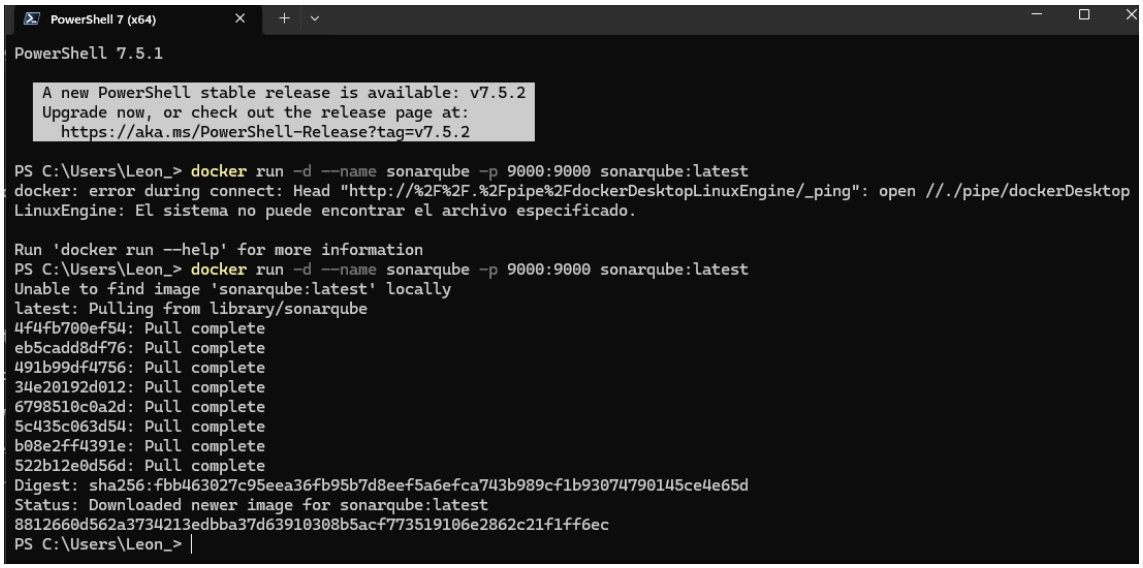
```
                ->press('Borrar')

                ->waitForLocation($this->baseUrl . self::PATH_CONVOCATORIA_SHOW, 10)
                ->assertPathIs(self::PATH_CONVOCATORIA_SHOW)

                ->waitForText('Convocatoria eliminada exitosamente.', 10)
                ->assertSee('Convocatoria eliminada exitosamente.')

                ->assertDontSee("a[href*='{$deleteHref}']");
        });
    } finally {
        DB::table('convocatoria')->where('idconvocatoria', $idParaEliminar)->delete();
    }
}

public function test_bloquea_eliminar_convocatoria_con_proyectos_asociados()
{
    $idConvocatoria = DB::table('convocatoria')->insertGetId([
        'numeroconvocatoria' => time(),
        'estadodescr' => 'Activa',
        'fechainicio' => '2024-01-01',
        'fechafin' => '2024-02-01',
        'anoconvocatoria' => 2024
    ]);

    $idProyecto = DB::table('proyecto')->insertGetId([
        'codproyecto' => 'PROY-DUSK-999',
        'idconvocatoria' => $idConvocatoria,
        'idareaconocimiento' => 1,
        'idestadoproyecto' => 1,
        'idtipoproyecto' => 1,

        'tituloproyecto' => 'Proyecto de Prueba Dusk para Borrado',
        'acronimo' => 'PPD-BORRADO',
        'usuario' => 'a@a',
```

```
        'idfacultad' => 1,
        'tiempo' => 10
    ]);

    $deleteHref = "/convocatoria/confirm/{$idConvocatoria}";

    try {
        $this->browse(function (Browser $browser) use ($deleteHref) {

            $browser->driver->manage()->deleteAllCookies();
            $browser->visit($this->baseUrl . '/login')
                ->waitFor(self::EMAIL, 10)
                ->type(self::EMAIL, 'a@a')
                ->type('input[name="password"]', '12345678')
                ->press('Entrar')
                ->waitForLocation('/datosPersonales', 10);

            $browser->visit($this->baseUrl . self::PATH_CONVOCATORIA_SHOW)
                ->waitFor("a[href*='{$deleteHref}']", 10)
                ->click("a[href*='{$deleteHref}']")

                ->waitForText('Confirme la eliminación de convocatoria', 10)
                ->press('Borrar')

                ->waitForLocation($this->baseUrl . self::PATH_CONVOCATORIA_SHOW, 10)
                ->assertPathIs(self::PATH_CONVOCATORIA_SHOW)

                ->waitForText('No se puede eliminar la convocatoria debido a que está en uso.', 10)
                ->assertSee('No se puede eliminar la convocatoria debido a que está en uso.');
        });
    } finally {
        DB::table('proyecto')->where('idproyecto', $idProyecto)->delete();
        DB::table('convocatoria')->where('idconvocatoria', $idConvocatoria)->delete();
    }
}
```

}

Anexo E: Instalación y uso de SonarQube.

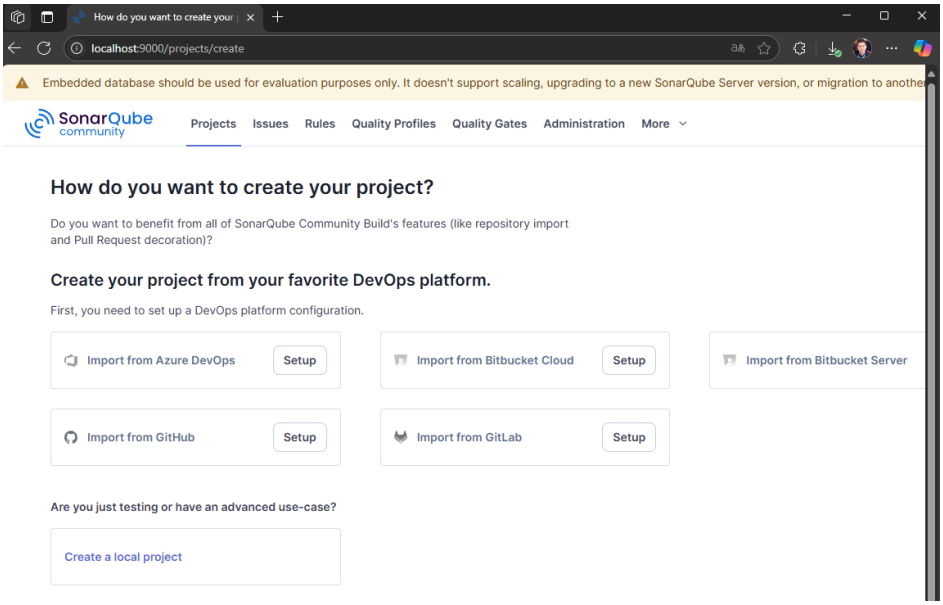
En Consola PowerShell en Windows ejecutar *docker run -d --name sonarqube -p 9000:9000 sonarqube:latest.* Se puede apreciar en la imagen:



Requisitos:

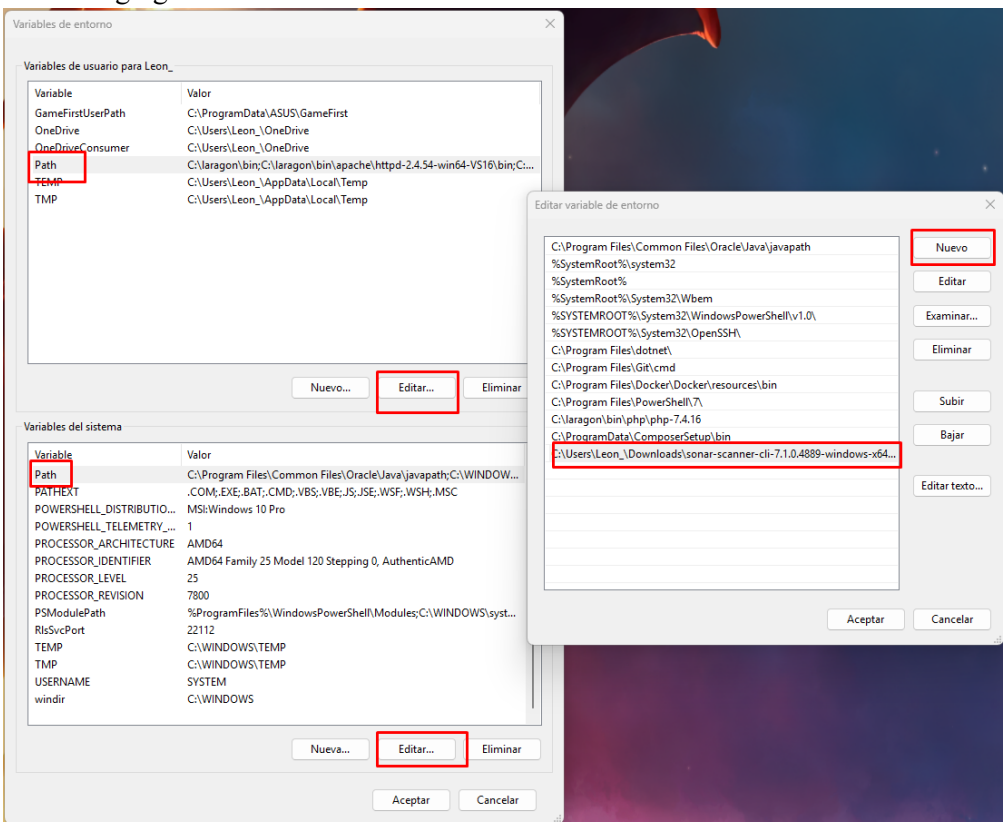
1. Tener Docker Desktop instalado y ejecutándose.
2. Abrir PowerShell (como administrador si es necesario).

- Después de ejecutarlo correctamente, SonarQube queda accesible en el navegador desde: http://localhost:9000
  Usuario: admin
  Contraseña: admin (pedirá cambiarla la primera vez)

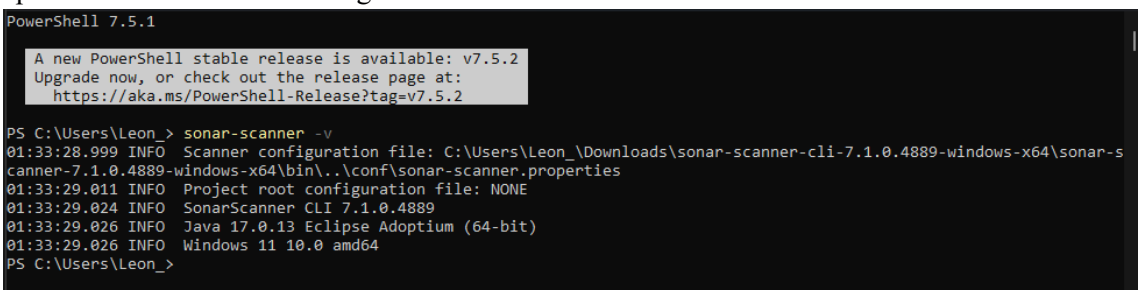


- **Instalar SonarScanner**
  Se Descarga desde: https://docs.sonarcloud.io/advanced-setup/ci-based-analysis/sonarscanner-cli/

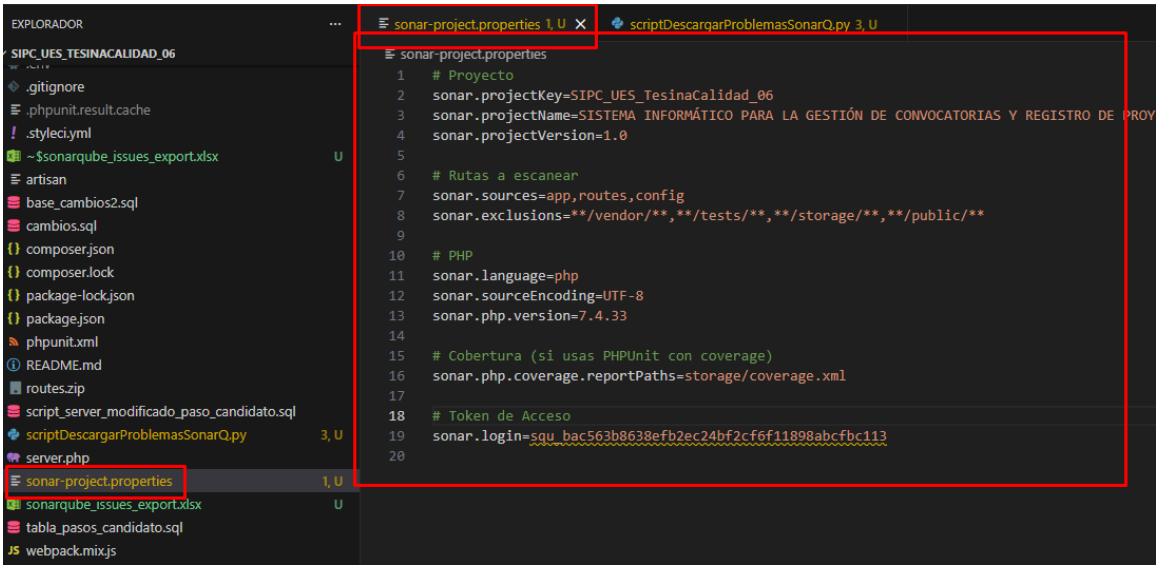
  Se debe agrega la ruta del Binario al PATH así:



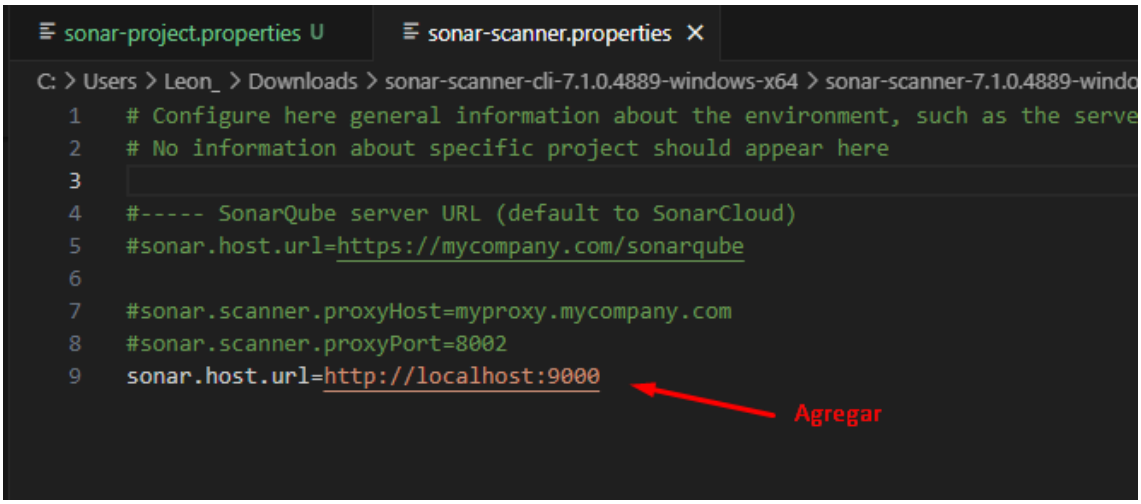
- Ahora verificamos que si está bien instalado escribiendo en consola *sonar-scaner -v,* debe aparecer como se ve en la imagen:



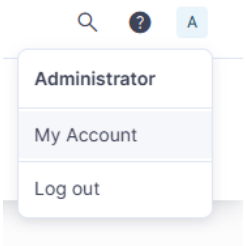
- Luego en la raíz del proyecto se debe crear el archivo llamado: *sonar-project.properties* y configurarlo como se aprecia en la imagen.



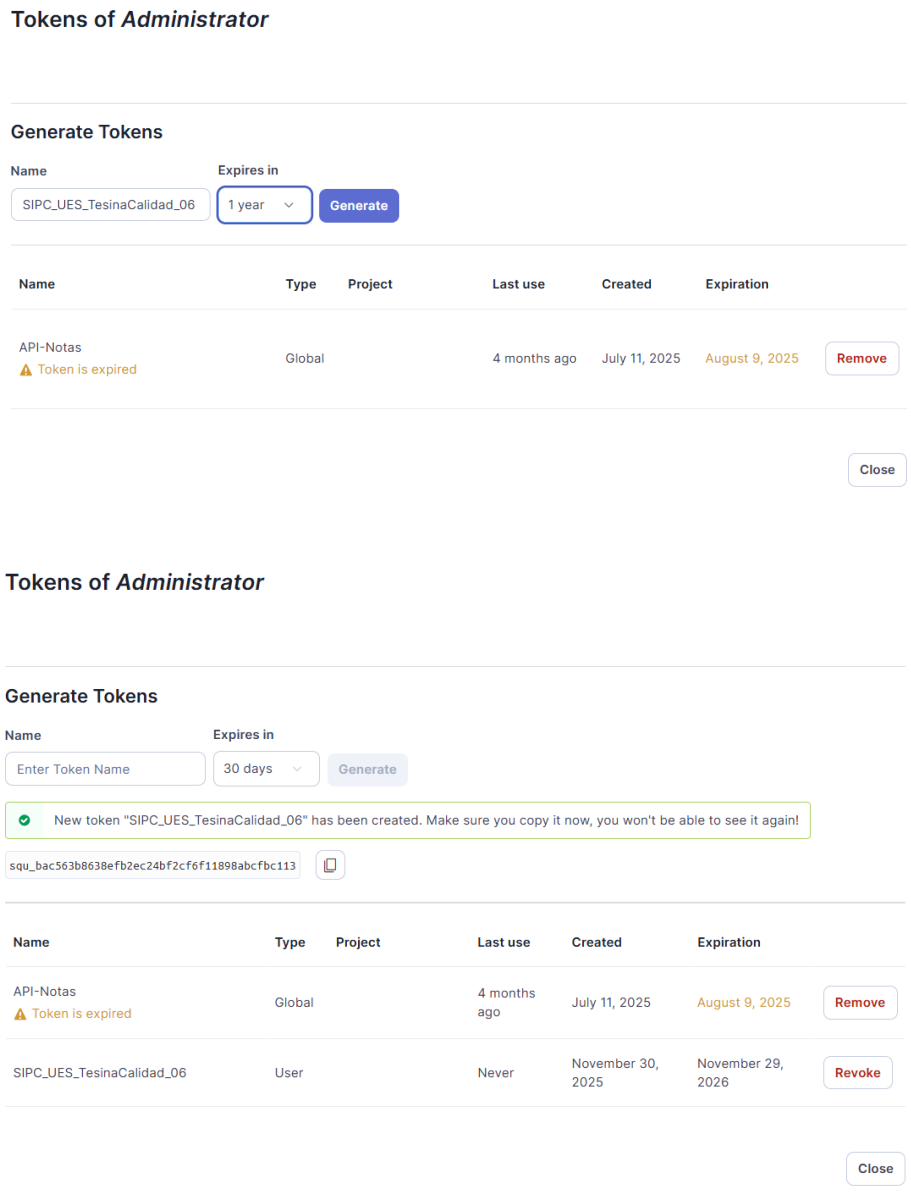
- En la ruta donde se descargó el Binario de Sonar Qube buscar la ruta y archivo .*sonar-scanner-cli-7.1.0.4889-windows-x64\sonar-scanner-7.1.0.4889-windows-x64\conf\sonar-scanner.properties* luego abrir dicho archivo y agregar la *línea sonar.host.url=http://localhost:9000* como se aprecia en la imagen:



En la página del SonarQube *Local* *http://localhost:9000* se debe buscar la opción de My Account luego buscar **Security**



En la pantalla de Security -> Tokens de Administrador se crea un Nuevo Token de acceso como se aprecia en la imagen, en este caso se llama *SIPC_UES_TesinaCalidad_06*:



Se modifica el archivo de propiedades y se agrega el Token de acceso:

```
sonar-project.properties 1, U
sonar-project.properties
1   # Proyecto
2   sonar.projectKey=SIPC_UES_TesinaCalidad_06
3   sonar.projectName=SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONVOCATORIAS Y REGISTRO DE PROYECTOS DE INVESTIGA
4   sonar.projectVersion=1.0
5
6   # Rutas a escanear
7   sonar.sources=app,routes,config
8   sonar.exclusions=**/vendor/**,**/tests/**,**/storage/**,**/public/**
9
10  # PHP
11  sonar.language=php
12  sonar.sourceEncoding=UTF-8
13  sonar.php.version=7.4.33
14
15  # Cobertura (si usas PHPUnit con coverage)
16  sonar.php.coverage.reportPaths=storage/coverage.xml
17
18  # Token de Acceso
19  sonar.login=squ_bac563b8638efb2ec24bf2cf6f11898abcfbc113
20
```

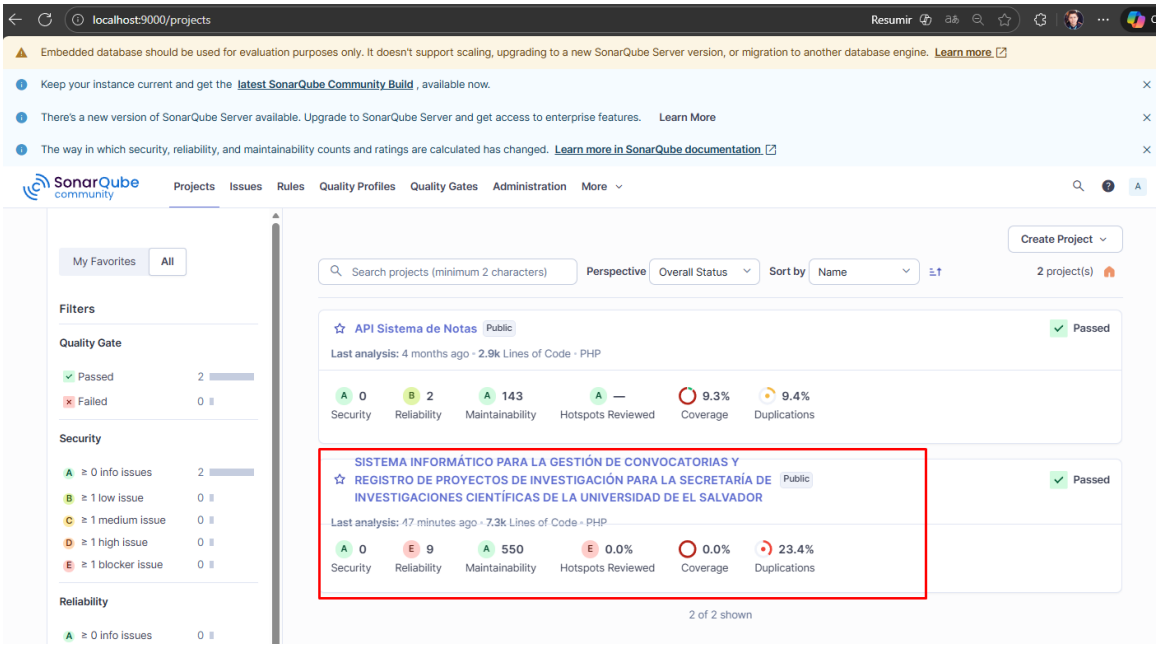
- Ahora en el PowerShell ejecutar *docker ps* para verificar que se encuentra ejecutándose:

```
PS C:\Users\Leon_> docker ps
>>
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                    NAM
ES
8812660d562a   sonarqube:latest   "/opt/sonarqube/dock…"   42 minutes ago   Up 42 minutes   0.0.0.0:9000->9000/tcp   son
arqube
PS C:\Users\Leon_>
```

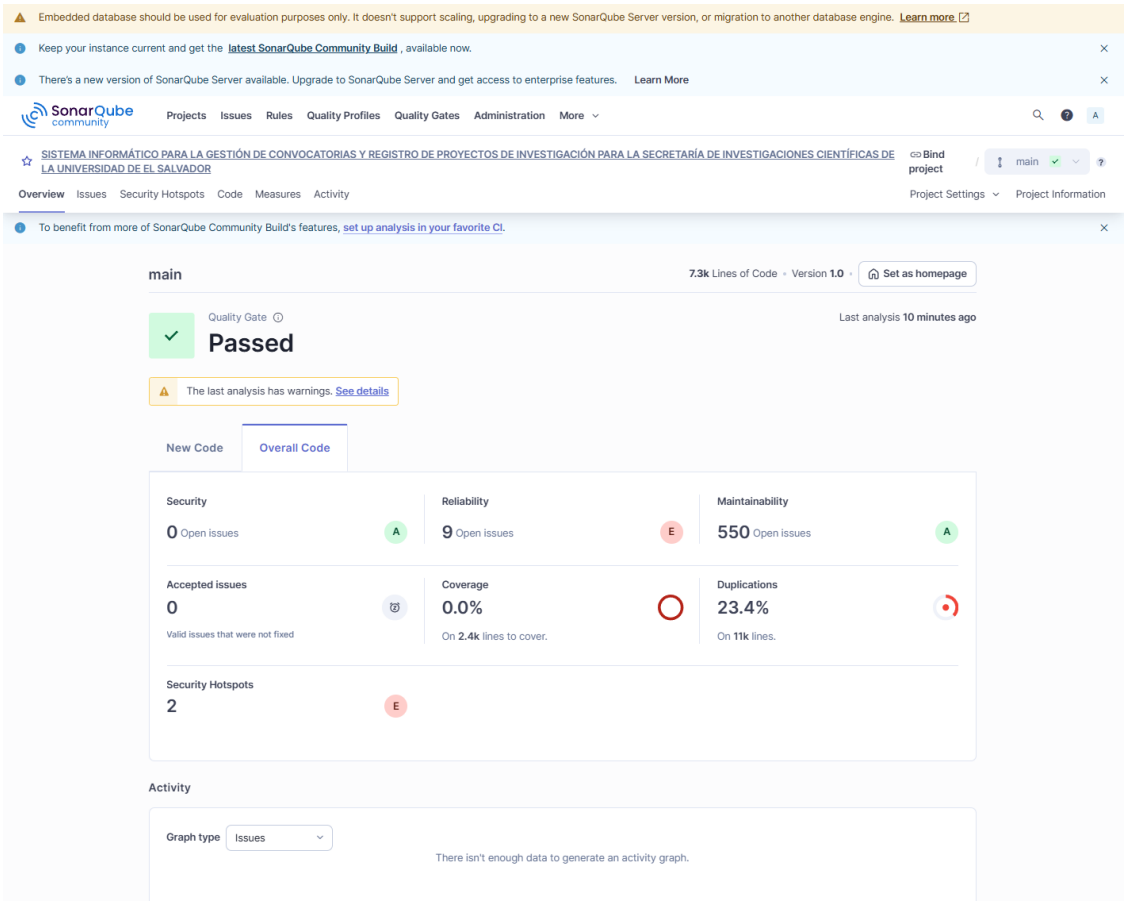
Ya se puede ejecutar el análisis de código, en la raíz del proyecto ejecutar el *comando sonar-scanner* y debe aparecer como se aprecia en la imagen:

```
C:\Users\Leon_\OneDrive\Documentos\GitHub\SIPC_UES_TesinaCalidad_06(tesina_g06_2aEntrega -> origin)
λ sonar-scanner
17:11:22.976 INFO  Scanner configuration file: C:\Users\Leon_\Downloads\sonar-scanner-cli-7.1.0.4889-windows-x64\sonar-scanner-7.1.0.4889
indows-x64\bin\..\conf\sonar-scanner.properties
17:11:22.991 INFO  Project root configuration file: C:\Users\Leon_\OneDrive\Documentos\GitHub\SIPC_UES_TesinaCalidad_06\sonar-project.pro
rties
17:11:23.011 INFO  SonarScanner CLI 7.1.0.4889
17:11:23.013 INFO  Java 17.0.13 Eclipse Adoptium (64-bit)
17:11:23.014 INFO  Windows 11 10.0 amd64
17:11:23.041 INFO  User cache: C:\Users\Leon_\.sonar\cache
17:11:23.945 INFO  Communicating with SonarQube Community Build 25.7.0.110598
17:11:23.945 WARN  Use of 'sonar.login' property has been deprecated in favor of 'sonar.token' (or the env variable alternative 'SONAR_TO
N'). Please use the latter when passing a token.
17:11:23.947 INFO  JRE provisioning: os[windows], arch[amd64]
17:11:24.712 INFO  Starting SonarScanner Engine...
17:11:24.713 INFO  Java 17.0.13 Eclipse Adoptium (64-bit)
```

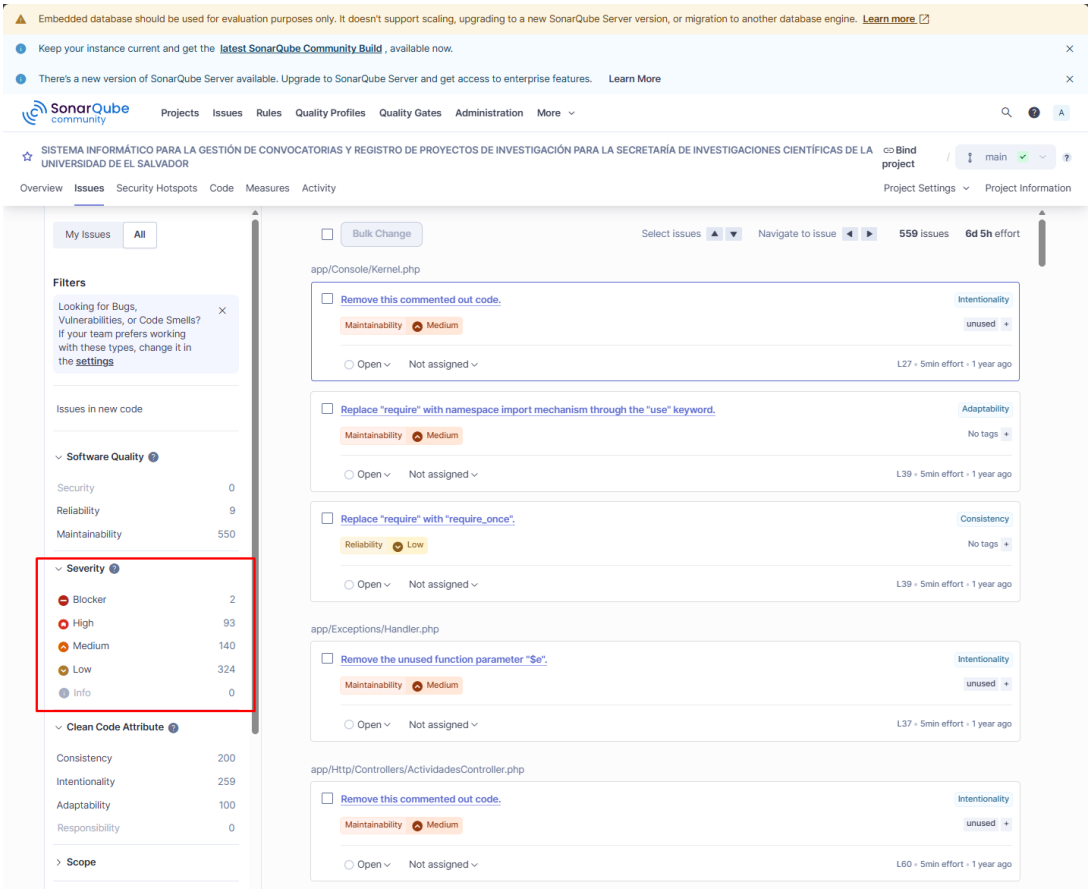
Para poder ver el análisis nos dirigimos a *http://localhost:9000* Y seleccionamos el proyecto a observar



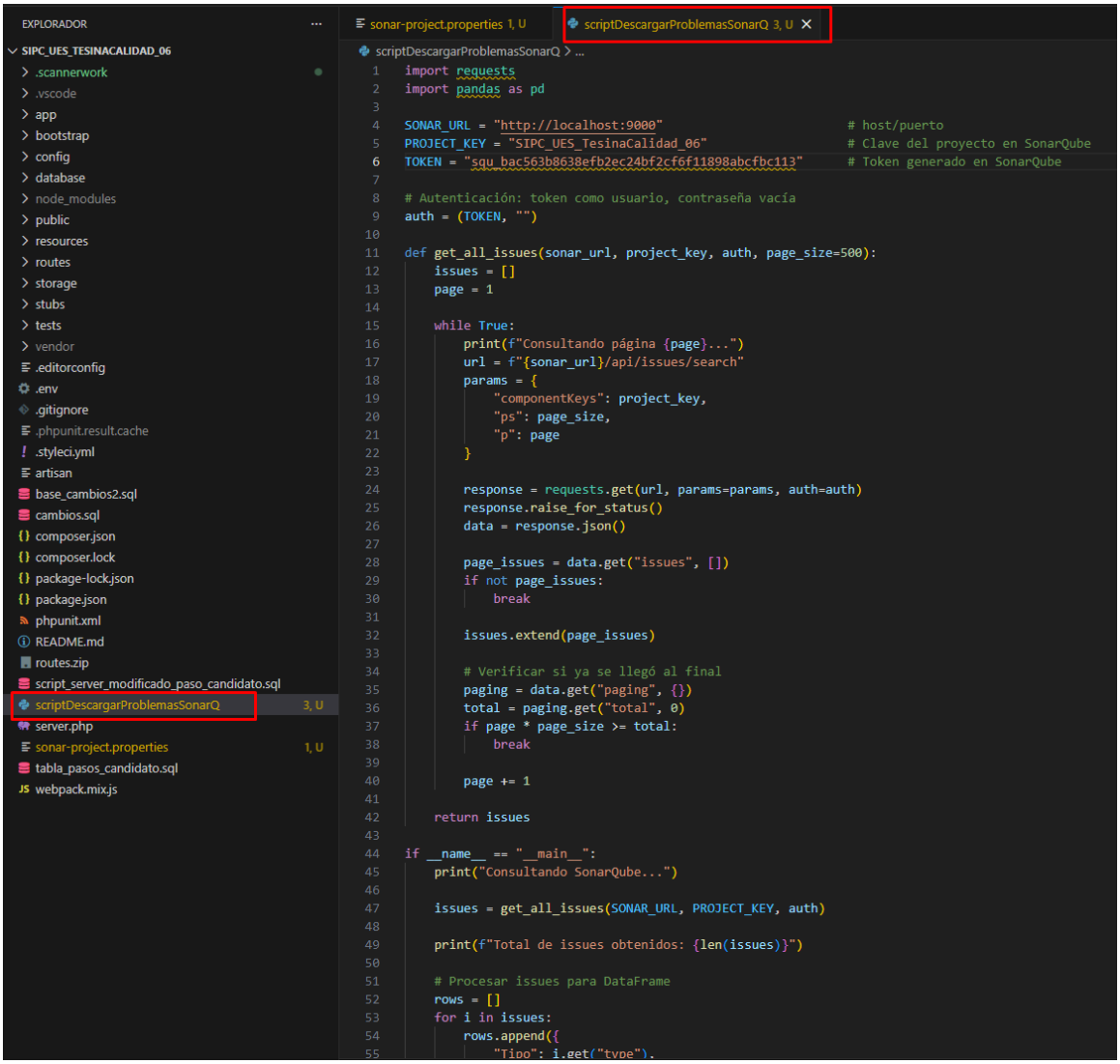
Se aprecia un anaisis complet de todo el sistema, como no cuenta con test unitarios observando un Coverage del 0%



Adicionalmente se pueden ver todas las Issues o problemas que tiene el Código, las que deberían solucionarse son las High



- Se creó un Script en Phyton para generar un Exel de todos los problemas que muestra SonarQube, el Script es el Siguiente:

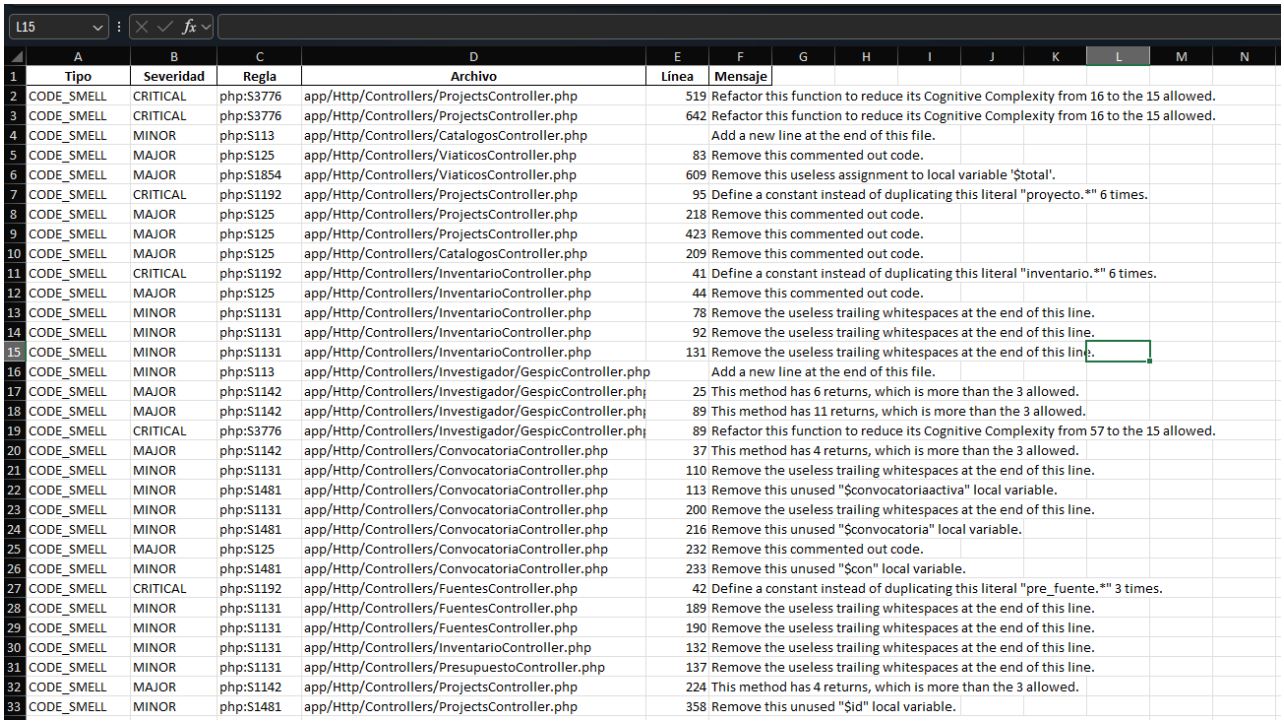


Cabe aclarar que se debe tener instalado Phyton y tener instalado pandas openpyxl, se instala en consola ejecutando el comando *pip install requests pandas openpyxl*

- Se corre el Script y debe generar el Excel con todo lo relacionado al análisis de SonarQube

```
C:\Users\Leon_\OneDrive\Documentos\GitHub\SIPC_UES_TesinaCalidad_06(tesina_g06_2aEntrega -> origin)
λ python scriptDescargarProblemasSonarQ.py
Consultando SonarQube...
Consultando página 1...
Consultando página 2...
Total de issues obtenidos: 559
Reporte generado exitosamente: sonarqube_issues_export.xlsx
```

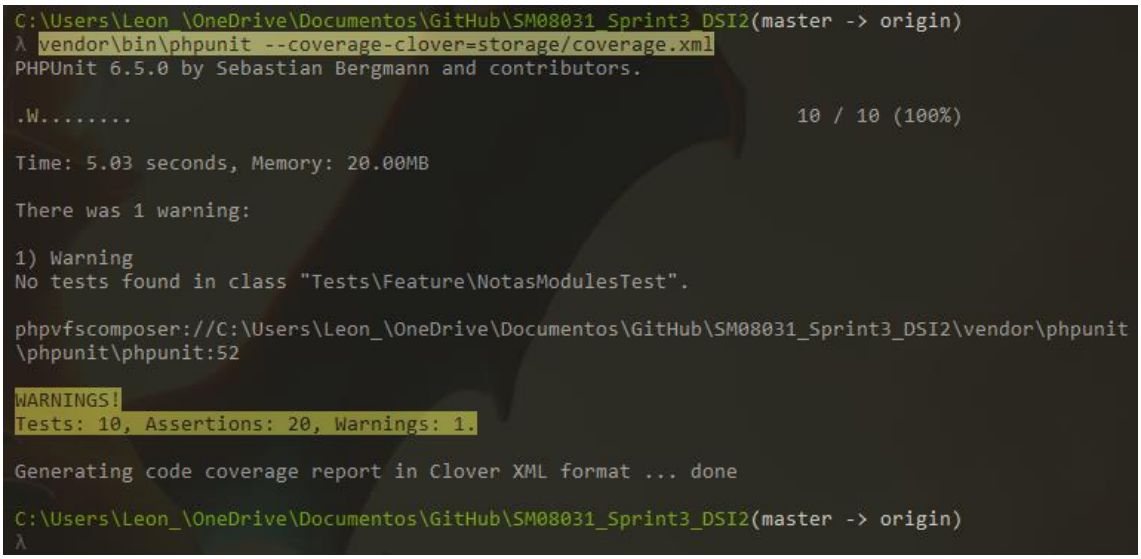
Se debería crear un archivo Excel en la raíz del proyecto llamado *sonarqube_issues_export.xlsx* en el cual se pueden observar todos los problemas encontrados por el análisis estático:



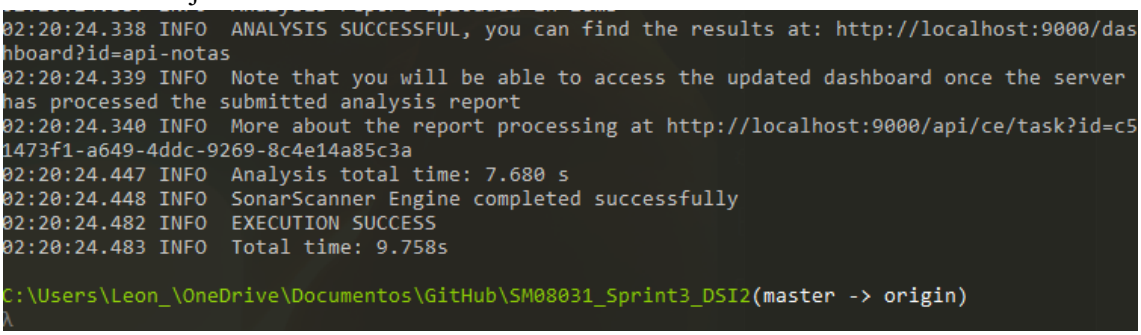
Tipo	Severidad	Regla	Archivo	Línea	Mensaje
CODE_SMELL	CRITICAL	php:S3776	app/Http/Controllers/ProjectsController.php	519	Refactor this function to reduce its Cognitive Complexity from 16 to the 15 allowed.
CODE_SMELL	CRITICAL	php:S3776	app/Http/Controllers/ProjectsController.php	642	Refactor this function to reduce its Cognitive Complexity from 16 to the 15 allowed.
CODE_SMELL	MINOR	php:S113	app/Http/Controllers/CatalogosController.php		Add a new line at the end of this file.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/ViaticosController.php	83	Remove this commented out code.
CODE_SMELL	MAJOR	php:S1854	app/Http/Controllers/ViaticosController.php	609	Remove this useless assignment to local variable '$total'.
CODE_SMELL	CRITICAL	php:S1192	app/Http/Controllers/ProjectsController.php	95	Define a constant instead of duplicating this literal "proyecto.*" 6 times.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/ProjectsController.php	218	Remove this commented out code.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/ProjectsController.php	423	Remove this commented out code.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/CatalogosController.php	209	Remove this commented out code.
CODE_SMELL	CRITICAL	php:S1192	app/Http/Controllers/InventarioController.php	41	Define a constant instead of duplicating this literal "inventario.*" 6 times.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/InventarioController.php	44	Remove this commented out code.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/InventarioController.php	78	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/InventarioController.php	92	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/InventarioController.php	131	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S113	app/Http/Controllers/Investigador/GespicController.php		Add a new line at the end of this file.
CODE_SMELL	MAJOR	php:S1142	app/Http/Controllers/Investigador/GespicController.php	25	This method has 6 returns, which is more than the 3 allowed.
CODE_SMELL	MAJOR	php:S1142	app/Http/Controllers/Investigador/GespicController.php	89	This method has 11 returns, which is more than the 3 allowed.
CODE_SMELL	CRITICAL	php:S3776	app/Http/Controllers/Investigador/GespicController.php	89	Refactor this function to reduce its Cognitive Complexity from 57 to the 15 allowed.
CODE_SMELL	MAJOR	php:S1142	app/Http/Controllers/ConvocatoriaController.php	37	This method has 4 returns, which is more than the 3 allowed.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/ConvocatoriaController.php	110	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1481	app/Http/Controllers/ConvocatoriaController.php	113	Remove this unused "$convocatoriaactiva" local variable.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/ConvocatoriaController.php	200	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1481	app/Http/Controllers/ConvocatoriaController.php	216	Remove this unused "$convocatoria" local variable.
CODE_SMELL	MAJOR	php:S125	app/Http/Controllers/ConvocatoriaController.php	232	Remove this commented out code.
CODE_SMELL	MINOR	php:S1481	app/Http/Controllers/ConvocatoriaController.php	233	Remove this unused "$con" local variable.
CODE_SMELL	CRITICAL	php:S1192	app/Http/Controllers/FuentesController.php	42	Define a constant instead of duplicating this literal "pre_fuente.*" 3 times.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/FuentesController.php	189	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/FuentesController.php	190	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/InventarioController.php	132	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MINOR	php:S1131	app/Http/Controllers/PresupuestoController.php	137	Remove the useless trailing whitespaces at the end of this line.
CODE_SMELL	MAJOR	php:S1142	app/Http/Controllers/ProjectsController.php	224	This method has 4 returns, which is more than the 3 allowed.
CODE_SMELL	MINOR	php:S1481	app/Http/Controllers/ProjectsController.php	358	Remove this unused "$id" local variable.

Sección Opcional Ejecutado con otro proyecto que si cuenta con test Unitarios

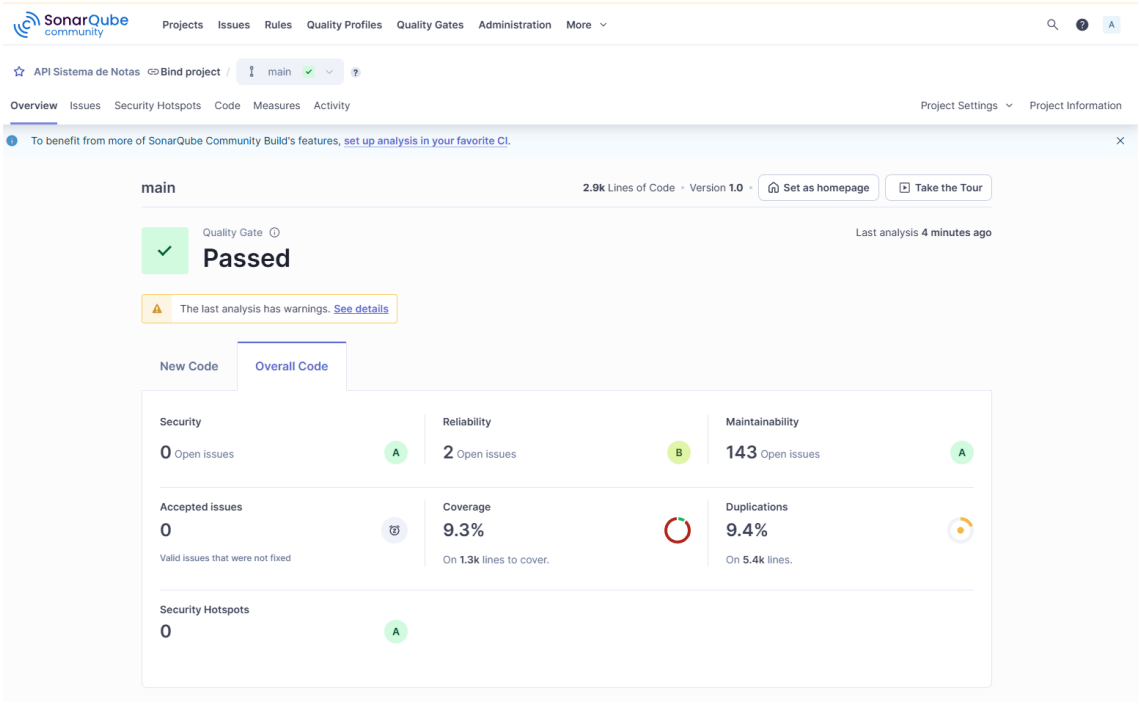
- Ya que para los test unitarios en PHP se usa PHPUnit, debemos ejecutar en consola *vendor/bin/phpunit --coverage-clover=storage/coverage.xml* Para que SonarQube tome el Coverage de PHPUnit



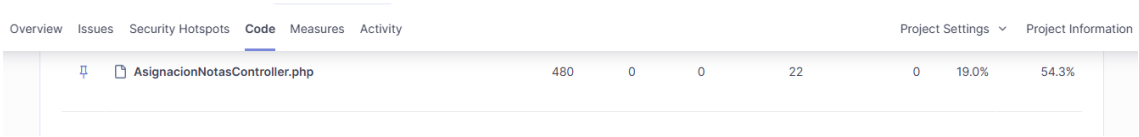
- Ahora se debe ejecutar de nuevo *sonar-scanner*



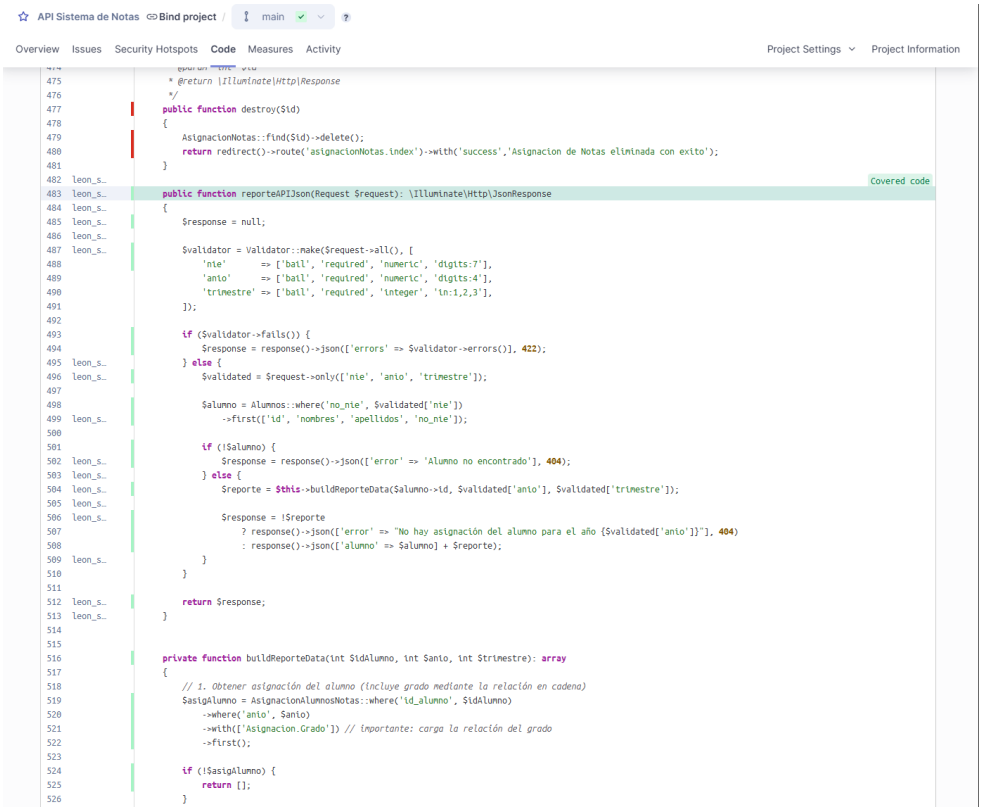
- Luego de esto ya se puede ver el Código con cobertura en SonarQube tal como salía en PHPUnit



Puede verse el porcentaje del archivo donde se encuentra el código del API cubierto por las pruebas:



Y también el código marcado en Verde, lo cual indica que las pruebas fueron exitosas y que cubre todo el código del API


API Sistema de Notas
Bind project
main
Overview
Issues
Security Hotspots
Code
Measures
Activity
Project Settings
Project Information
Covered code
* @return \Illuminate\Http\Response
*/
public function destroy($id)
{
AsignacionNotas::find($id)->delete();
return redirect()->route('asignacionNotas.index')->with('success','Asignacion de Notas eliminada con exito');
}
public function reporteAPIJson(Request $request): \Illuminate\Http\JsonResponse
{
$response = null;
$validator = Validator::make($request->all(), [
'nie' => ['ball', 'required', 'numeric', 'digits:7'],
'anio' => ['ball', 'required', 'numeric', 'digits:4'],
'trimestre' => ['ball', 'required', 'integer', 'in:1,2,3'],
]);
if ($validator->fails()) {
$response = response()->json(['errors' => $validator->errors()], 422);
} else {
$validated = $request->only(['nie', 'anio', 'trimestre']);
$alumno = Alumnos::where('no_nie', $validated['nie'])
->first(['id', 'nombres', 'apellidos', 'no_nie']);
if (!$alumno) {
$response = response()->json(['error' => 'Alumno no encontrado'], 404);
} else {
$reporte = $this->buildReporteData($alumno->id, $validated['anio'], $validated['trimestre']);
$response = !$reporte
? response()->json(['error' => "No hay asignación del alumno para el año {$validated['anio']}"], 404)
: response()->json(['alumno' => $alumno] + $reporte);
}
}
return $response;
}
private function buildReporteData(int $idAlumno, int $anio, int $trimestre): array
{
// 1. Obtener asignación del alumno (incluye grado mediante la relación en cadena)
$asigAlumno = AsignacionAlumnosNotas::where('id_alumno', $idAlumno)
->where('anio', $anio)
->with(['Asignacion.Grado']) // importante: carga la relación del grado
->first();
if (!$asigAlumno) {
return [];
}