



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Proyecto Unidad I “Mejoramiento de la aplicación Sustain Partners”

Curso: *Calidad y Pruebas de Software*

Docente: *Ing. Patrick Jose Cuadros Quiroga*

Integrantes:

<i>Agreda Ramirez, Jesús Eduardo</i>	<i>(2021069823)</i>
<i>Cabrera Catari, Camila Fernanda</i>	<i>(2021069824)</i>
<i>Ortiz Fernandez, Ximena Andrea</i>	<i>(2021071080)</i>
<i>Meza Noalcca, Jean Marco</i>	<i>(2021071087)</i>

*Tacna – Perú
2024*

ÍNDICE

Resumen.....	2
Abstract.....	2
1. Antecedentes o introducción.....	3
2. Título.....	3
3. Autores.....	3
4. Planteamiento del problema.....	3
4.1. Problema.....	3
4.2. Justificación.....	4
4.3. Alcance.....	4
5. Objetivos.....	5
5.1. General.....	5
5.2. Específicos.....	5
6. Referentes teóricos.....	6
6.1. Diagramas de Casos de Uso.....	6
6.2. Diagrama de Clases.....	8
6.3. Diagrama de Componentes.....	9
6.4. Arquitectura.....	9
7. Desarrollo de la propuesta.....	10
7.1. Tecnologías de información.....	12
7.2. Metodología, técnicas usadas.....	15
8. Cronograma.....	41
8.1. Recursos.....	42

PROYECTO UNIDAD I

“Análisis Sustain Partners”

Resumen

El proyecto "Sustain Partners", busca implementar una plataforma de crowdsourcing para fomentar la colaboración en proyectos de sostenibilidad en Perú. Esta iniciativa tiene como objetivo conectar a individuos, organizaciones y comunidades interesadas en los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas, facilitando la propuesta, búsqueda y participación en proyectos sostenibles. La plataforma ofrece una interfaz de usuario intuitiva que permite a usuarios de diversos niveles técnicos interactuar fácilmente y contribuir a proyectos relevantes. Además, integra herramientas avanzadas para la gestión de proyectos, el seguimiento del impacto mediante métricas específicas y el apoyo a contribuciones financieras seguras. Sustain Partners se propone como solución a la fragmentación de esfuerzos en sostenibilidad, centralizando recursos y maximizando el impacto de las iniciativas. Al promover una cultura de colaboración y sostenibilidad, la plataforma busca empoderar a la comunidad peruana para enfrentar desafíos locales y globales de forma más efectiva y sostenida. En definitiva, Sustain Partners no sólo pretende ser un facilitador tecnológico, sino también un motor de cambio hacia un futuro más sostenible en Perú.

Abstract

Sustain Partners is an innovative crowdsourcing platform, aimed at enhancing collaboration in sustainability projects across Peru. The platform is designed to connect individuals, organizations, and communities interested in the United Nations' Sustainable Development Goals (SDGs), facilitating the proposal, search, and engagement in sustainable initiatives. Featuring an intuitive user interface, Sustain Partners enables users of varying technical abilities to actively participate in and contribute to relevant projects. It integrates advanced tools for project management, impact tracking through specific metrics, and secure financial contributions. By addressing the fragmentation of sustainability efforts and centralizing resources, the platform significantly enhances the efficiency and impact of these initiatives. Sustain Partners not only serves as a technological facilitator but also as a catalyst for fostering a culture of collaboration and sustainability, empowering the Peruvian community to tackle local and global challenges more effectively and sustainably. This initiative represents a strategic approach to promoting sustainable development and environmental stewardship in Peru, positioning itself as a model for similar endeavors worldwide.

1. Antecedentes o introducción

En la actualidad, la sociedad del Perú enfrenta desafíos urgentes relacionados con el desarrollo sostenible, la conservación del medio ambiente y el logro de los Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas. A pesar de los esfuerzos en curso, existen diversas problemáticas que obstaculizan el progreso hacia un futuro más sostenible y equitativo. Estos son algunos de los antecedentes y la situación actual que resaltan la necesidad de un proyecto como Sustain Partners:

- Falta de Colaboración Efectiva: A menudo, las iniciativas sostenibles y los proyectos relacionados con los ODS enfrentan desafíos para atraer la colaboración activa de individuos, organizaciones y comunidades. La falta de un mecanismo centralizado para conectar a las partes interesadas con proyectos relevantes dificulta la colaboración efectiva.
- Dificultad para Identificar Oportunidades de Contribución: Muchas personas y organizaciones desean contribuir a la consecución de los ODS, pero les resulta difícil identificar proyectos o iniciativas alineados con sus intereses y habilidades. La falta de visibilidad de proyectos sostenibles relevantes es un obstáculo importante.
- Fragmentación de los Esfuerzos: A menudo, los esfuerzos sostenibles se encuentran fragmentados y carecen de coordinación. Esto puede resultar en la duplicación de esfuerzos y la falta de eficiencia en la consecución de los ODS.

2. Título

Implementación de plataforma Crowdsourcing - Sustain Partners

3. Autores

- Agreda Ramirez, Jesús Eduardo
- Cabrera Catari, Camila Fernanda
- Ortiz Fernandez, Ximena Andrea
- Meza Noalcca, Jean Marco

4. Planteamiento del problema

4.1. Problema

- *Fragmentación de Esfuerzos:* Actualmente, en Perú, existen numerosas iniciativas y esfuerzos aislados relacionados con la sostenibilidad. La falta de una plataforma centralizada conduce a la fragmentación de recursos y conocimientos valiosos.
- *Dificultad en la Gestión de Proyectos Sostenibles:* La planificación, ejecución y seguimiento de proyectos de sostenibilidad son procesos complejos y requieren una gestión eficiente. La falta de herramientas especializadas dificulta la efectividad de estos proyectos.

- *Desafío de la Medición del Impacto:* Evaluar el impacto real de los proyectos de sostenibilidad es un desafío. Sustain Partners se posiciona como una solución que facilitará la medición y el seguimiento de los resultados de los proyectos sostenibles.
- *Falta de Conectividad entre Actores de Sostenibilidad:* La falta de un espacio de colaboración e intercambio de ideas en línea dificulta la conectividad y el trabajo conjunto entre los interesados en la sostenibilidad en Perú. Sustain Partners aborda este problema conectando a los actores clave en un solo ecosistema.
- *Complejidad de los ODS:* Los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas son un marco ambicioso. Sustain Partners se presenta como una plataforma que desglosa estos ODS en proyectos y acciones concretas, lo que facilita su implementación a nivel local y nacional.

4.2. Justificación

La justificación en el proyecto Sustain Partners se sustenta en su capacidad para abordar desafíos nacionales apremiantes, especialmente en lo que respecta a la consecución de los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas. La plataforma servirá como un motor para la colaboración en proyectos con un impacto directo en áreas críticas como la erradicación de la pobreza, la igualdad de género, la acción climática y la protección del medio ambiente. Además de su impacto social y ambiental, Sustain Partners presenta oportunidades estratégicas para colaboraciones con diversas organizaciones interesadas en promover la sostenibilidad. Esto se traduce en un valor a largo plazo, ya que la inversión no solo tiene el potencial de generar ingresos, sino que también respalda la construcción de un mundo más sostenible y equitativo.

4.3. Alcance

Inclusiones:

- Interfaz de usuario intuitiva: La plataforma tendrá un diseño fácil de utilizar y con opciones necesarias y entendibles para su correcto uso.
- Medidas de seguridad para el acceso a la página: La plataforma tendrá un Captcha para poder iniciar sesión en la página, esta medida de seguridad se implementará con el fin de evitar ataques de bots.
- Escalabilidad: Diseño del sistema que permita la escalabilidad para manejar un crecimiento en el número de proyectos y usuarios a lo largo del tiempo.

Exclusiones:

- Desarrollo de Aplicaciones Móviles Personalizadas: El alcance del proyecto no incluirá el desarrollo de aplicaciones móviles personalizadas para plataformas específicas, como iOS o Android. En lugar de ello, Sustain Partners se diseñará

como una plataforma web accesible desde dispositivos móviles y equipos de escritorio, garantizando una experiencia uniforme en diferentes dispositivos.

- Contenido de Proyectos de Sostenibilidad: Si bien Sustain Partners facilitará la creación y administración de proyectos de sostenibilidad, el proyecto no asumirá la responsabilidad de definir el contenido específico de estos proyectos. Los usuarios y organizaciones serán responsables de definir los detalles y objetivos de sus propios proyectos.
- Infraestructura Tecnológica Existente: Sustain Partners se diseñará para integrarse con la infraestructura tecnológica disponible, sin requerir una actualización o modificación sustancial.

5. Objetivos

5.1. General

El objetivo general del proyecto Sustain Partners es crear una plataforma en línea que promueva la colaboración activa entre individuos y organizaciones con el fin de contribuir de manera significativa a la consecución de los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas. Esta plataforma facilitará la identificación, creación, gestión y seguimiento de proyectos sostenibles en diversas áreas, impulsando el impacto positivo en la sociedad y el medio ambiente.

5.2. Específicos

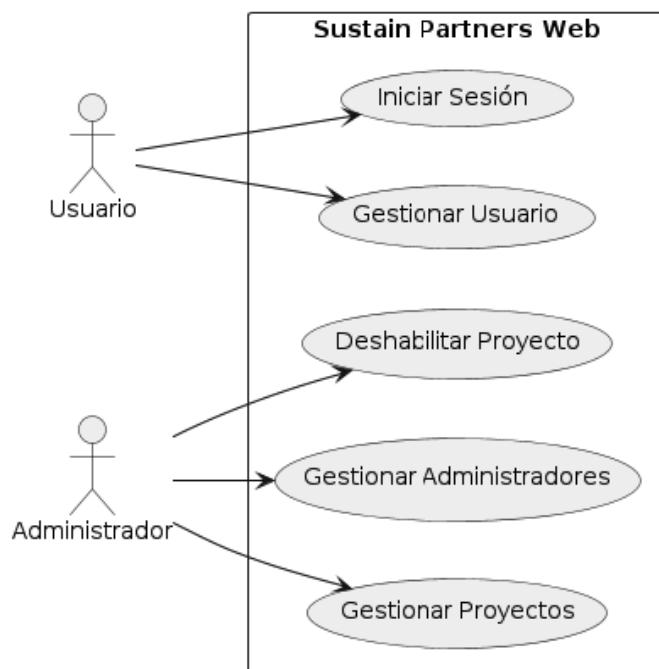
- Facilitar la Creación de Proyectos Sostenibles:
 - Permitir a los usuarios proponer proyectos relacionados con los ODS, especificando objetivos, recursos y ubicación.
 - Posibilitar la colaboración y co-creación de proyectos por parte de usuarios con intereses afines.
- Promover la Colaboración Activa:
 - Facilitar la búsqueda y participación de usuarios en proyectos existentes que se alineen con sus intereses y habilidades.
 - Fomentar la comunicación efectiva entre los miembros del proyecto para impulsar la colaboración.
- Medir y Evaluar el Impacto en los ODS:
 - Permitir a los usuarios realizar un seguimiento del progreso de los proyectos y registrar hitos significativos.
 - Recopilar datos y métricas para evaluar el impacto real de los proyectos en la consecución de los ODS.
- Sensibilizar sobre los Objetivos de Desarrollo Sostenible:
 - Hay que destacar proyectos que se alineen con los ODS más apremiantes, generando conciencia sobre los desafíos nacionales y locales.
 - Proporcionar información educativa sobre los ODS y su importancia.

- Garantizar la Seguridad y Privacidad de los Usuarios:
 - Implementar medidas sólidas de seguridad de datos para proteger la información personal y los detalles de los proyectos.
 - Respetar la privacidad de los usuarios y cumplir con las regulaciones de privacidad.
- Promover la Participación Inclusiva:
 - Hacer que la plataforma sea accesible y amigable para una amplia gama de usuarios, independientemente de su origen geográfico o experiencia en sostenibilidad.
 - Fomentar la diversidad y la inclusión en la comunidad de Sustain Partners.

6. Referentes teóricos

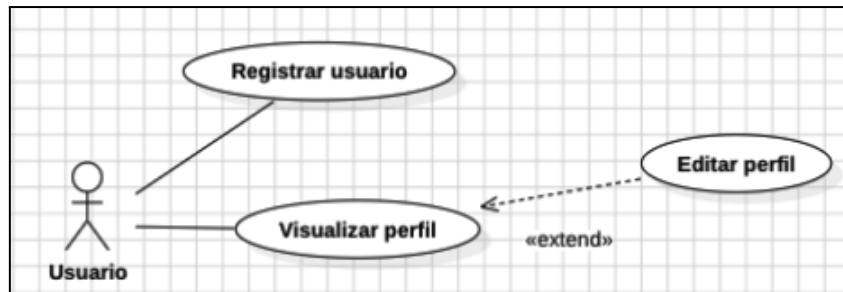
6.1. Diagramas de Casos de Uso

Aplicación web - Sustain Partners



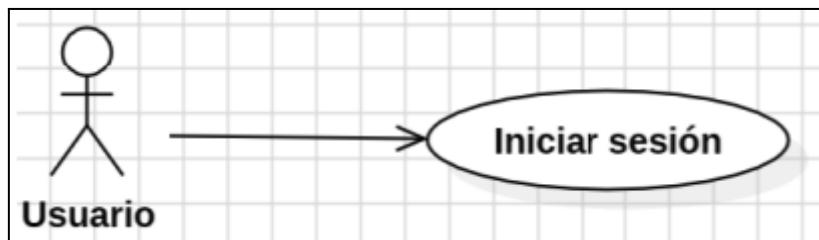
Este diagrama representa las interacciones de dos tipos de usuarios, "Usuario" y "Administrador". El "Usuario" tiene permisos para "Iniciar Sesión", y "Gestionar Usuario". Por otro lado, el "Administrador" comparte algunas de estas interacciones, como "Gestionar Usuario", pero también tiene acceso exclusivo a funciones adicionales como "Gestionar Proyectos", "Gestionar Administradores", y "Deshabilitar Proyecto". Esto indica una jerarquía de permisos donde los administradores pueden realizar una gestión más amplia dentro del sistema.

Gestionar Usuario



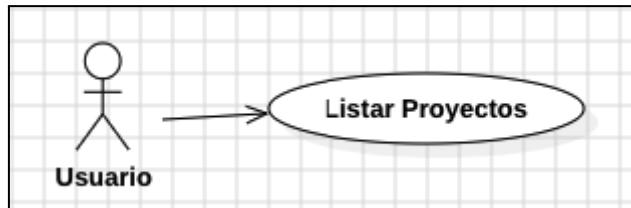
El “Usuario” puede registrar su información en el sistema. Además, hay una extensión que permite al usuario editar su perfil, lo cual es una acción opcional que se deriva del caso de uso principal de visualizar el perfil.

Iniciar Sesión



El “Usuario” puede iniciar sesión en el sistema.

Gestionar Proyectos



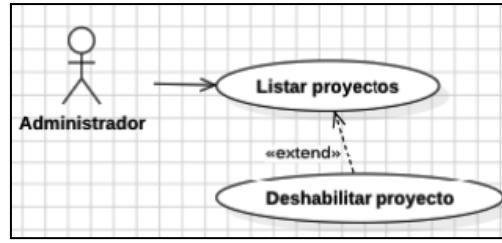
El “Usuario” tiene la capacidad de listar los proyectos disponibles o en los que está involucrado.

Gestionar Administradores



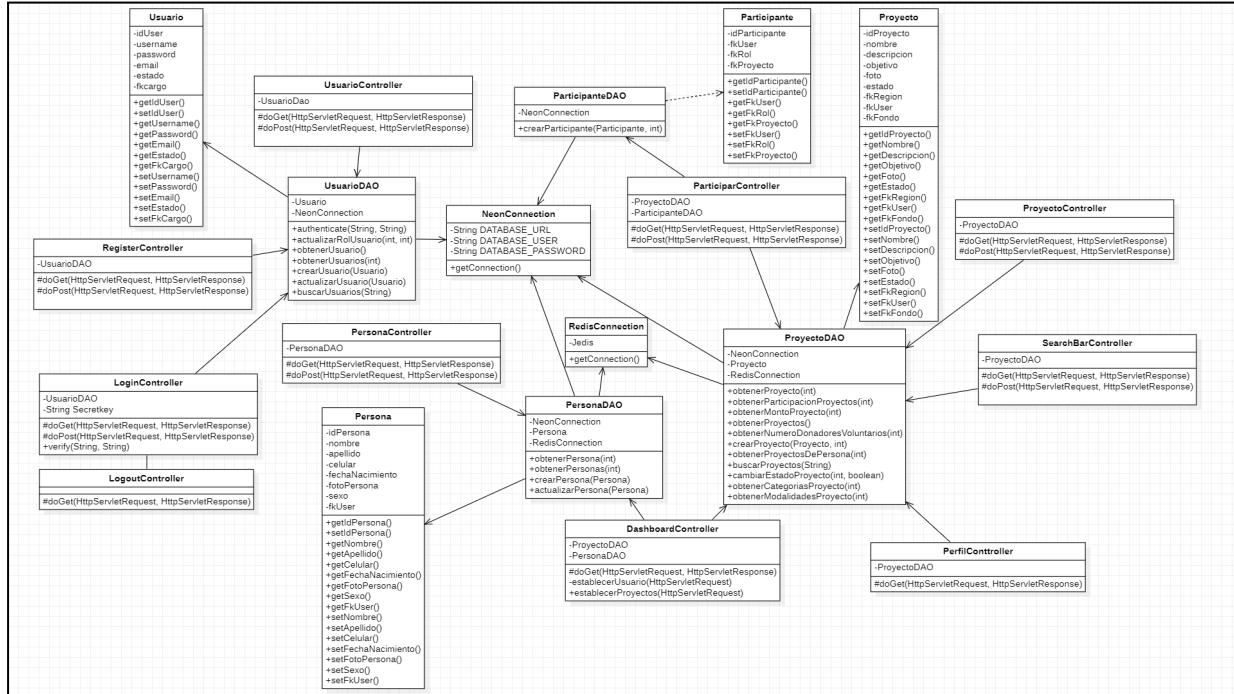
El “Administrador” puede asignar roles de administrador a otros usuarios.

Deshabilitar Proyecto



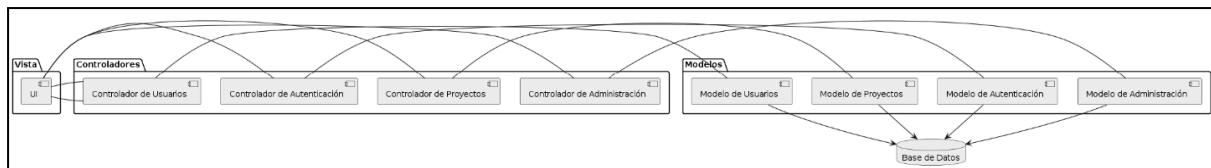
El “Administrador” puede listar proyectos y, como extensión a este caso de uso, tiene la opción de deshabilitar un proyecto si lo considera necesario.

6.2. Diagrama de Clases



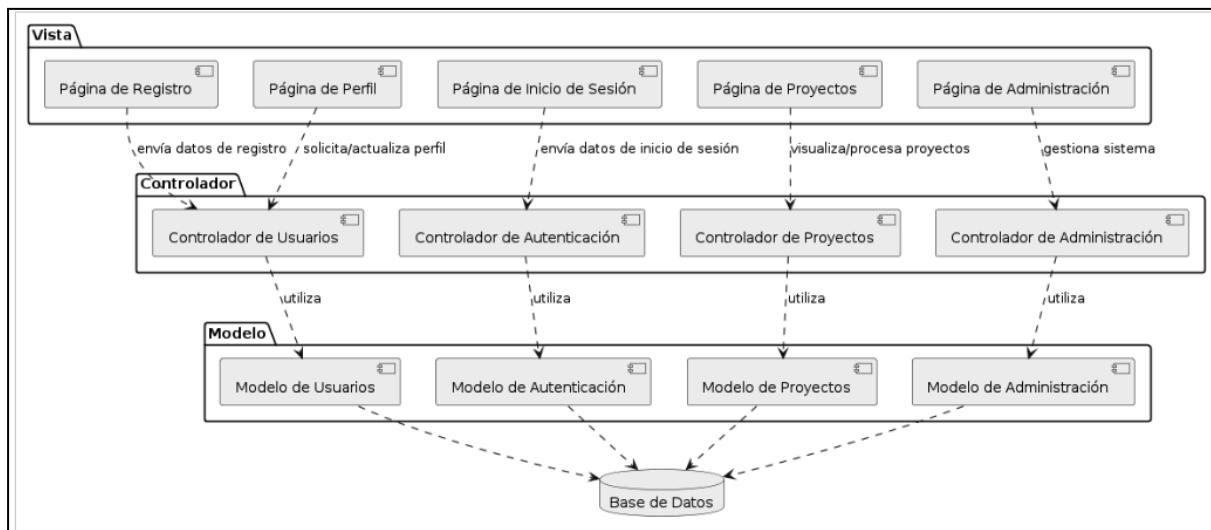
Este diagrama de clases representa la arquitectura de un sistema web, mostrando cómo los controladores gestionan las interacciones del usuario y las comunicaciones HTTP, los DAOs proporcionan una interfaz para las operaciones de la base de datos, y las clases de modelo representan las entidades del negocio. Los controladores, como “LoginController” o “UsuarioController”, manejan las solicitudes y respuestas específicas, mientras que los DAOs, como “UsuarioDAO” o “ProyectoDAO”, encapsulan la lógica de acceso a los datos. Las conexiones a bases de datos y al sistema de almacenamiento en memoria Redis están también representadas, lo que indica un enfoque en la persistencia y la recuperación eficiente de datos.

6.3. Diagrama de Componentes



Este diagrama de arquitectura representa un patrón MVC (Modelo-Vista-Controlador) con tres capas distintas. La capa de Vista, compuesta por interfaces de usuario (UI), permite al usuario interactuar con la aplicación. La capa de Controladores procesa las solicitudes del usuario, dirigiéndose a los Modelos adecuados, que gestionan la lógica de negocio y los datos. Finalmente, todos los Modelos interactúan con una base de datos central para realizar operaciones de persistencia.

6.4. Arquitectura



El diagrama representa una arquitectura de software MVC, dividiendo la aplicación en tres capas interconectadas: la capa de Vista maneja la interfaz de usuario, la capa de Controlador procesa las interacciones del usuario, y la capa de Modelo gestiona la lógica de negocio y la comunicación con la base de datos.

7. Desarrollo de la propuesta

Análisis con SonarQube

El análisis de SonarQube de la plataforma "Sustain Partners" ha revelado varios aspectos que requieren atención para mejorar la calidad y seguridad del código. A continuación, se desglosan los hallazgos y se propone un plan de mejora:

Hallazgos Clave:

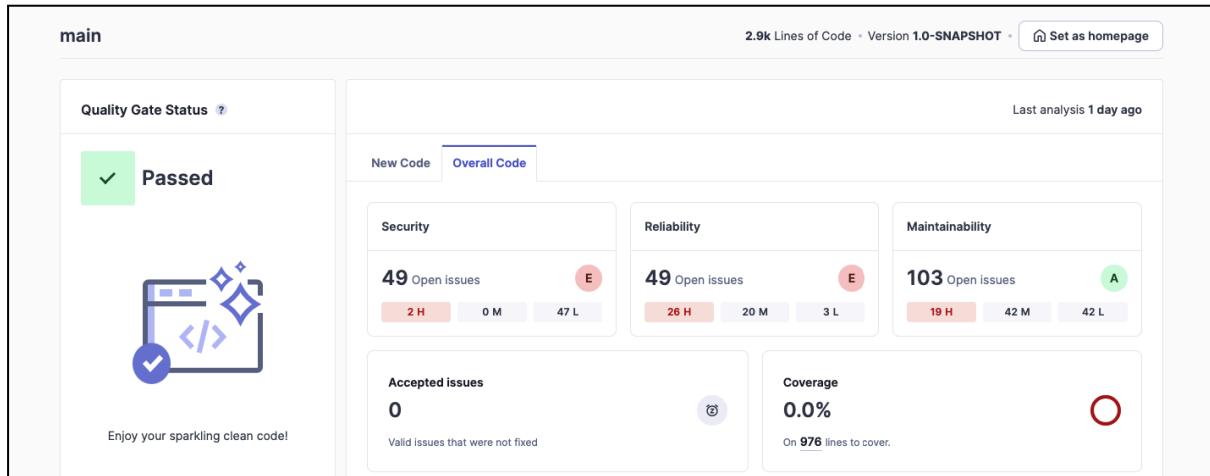
- *Seguridad:* Se han identificado 49 problemas de seguridad, de los cuales 2 son críticos (marcados como 'H') y 47 son significativos (marcados como 'L'). Es esencial abordar estos problemas para proteger la plataforma y los datos de los usuarios frente a posibles vulnerabilidades.
- *Confiabilidad:* Existen 47 cuestiones de confiabilidad pendientes. Los 26 problemas marcados como 'H' y 19 como 'M' podrían resultar en comportamientos imprevistos o fallas en la aplicación.
- *Mantenibilidad:* Se han detectado 91 problemas de mantenibilidad, con 19 clasificados como 'H' y 42 como 'M'. Estos problemas, si no se tratan, pueden aumentar la deuda técnica y afectar la escalabilidad y la facilidad de mantenimiento a largo plazo.
- *Cobertura de Código:* Actualmente se registra una cobertura de código del 0.0%, lo que indica que no hay pruebas automatizadas que verifiquen la correctitud del código existente.
- *Hotspots de Seguridad:* SonarQube ha encontrado 41 hotspots de seguridad que necesitan una revisión manual para evaluar y mitigar riesgos adicionales.

Plan de Mejora Propuesto:

- Priorización de Vulnerabilidades Críticas: Dedicar los recursos inmediatamente para resolver las vulnerabilidades de seguridad de nivel 'H'.
- Refactorización para Confiabilidad: Planificar un ciclo de refactorización para tratar las cuestiones de confiabilidad y prevenir fallos potenciales en la plataforma.
- Reducción de la Deuda Técnica: Abordar los problemas de mantenibilidad para mejorar la claridad del código y reducir la complejidad.
- Implementación de Pruebas Automatizadas: Desarrollar y aplicar un conjunto de pruebas unitarias y de integración para aumentar la cobertura de código y garantizar la funcionalidad esperada.
- Revisión de Hotspots de Seguridad: Realizar una revisión manual detallada de los hotspots de seguridad para descartar falsos positivos y resolver problemas genuinos.

Impacto Esperado:

La implementación de este plan mejorará significativamente la seguridad y la calidad de la plataforma "Sustain Partners". Además de prevenir riesgos de seguridad, estas mejoras proporcionarán una base sólida para la expansión futura y la confiabilidad a largo plazo de la aplicación. Con un código más limpio y una plataforma segura, Sustain Partners puede continuar su misión de facilitar proyectos sostenibles con una mayor confianza de sus usuarios y colaboradores.



Análisis con Snyk

El análisis de Snyk de la plataforma "Sustain Partners" ha revelado varios aspectos que requieren atención para mejorar la calidad y seguridad del código. A continuación, se desglosan los hallazgos y se propone un plan de mejora:

Hallazgos Clave:

- *Seguridad:* Se han identificado un total de 11 problemas de seguridad en el código, compuestos por 3 problemas de alta severidad y 6 de severidad media y 2 de severidad baja. Estos incluyen vulnerabilidades como Cross-Site Scripting (XSS) y Path Traversal, que podrían permitir a un atacante comprometer la aplicación y acceder a datos sensibles.
- *Confiabilidad:* Las incidencias encontradas sugieren áreas donde la entrada no saneada podría llevar a la manipulación de archivos o datos críticos, afectando la confiabilidad del sistema.
- *Mantenibilidad:* Es crucial dirigir la atención hacia prácticas de codificación seguras, especialmente al manejar credenciales y constantes de seguridad, para evitar que la información sensible se codifique de forma rígida en la base del código .
- *Cobertura de Código:* El reporte indica que se han escaneado 21 archivos Java, 12 archivos Java Server Pages y 2 archivos XML, lo que refleja el alcance del análisis realizado.

Plan de Mejora Propuesto:

- *Priorización de Vulnerabilidades Críticas:* Es imprescindible corregir de inmediato las vulnerabilidades de alta severidad, comenzando con las que tienen el potencial de ser explotadas de manera más fácil o que tienen un alto impacto.
- *Refactorización para Confiabilidad:* Refactorizar el código donde se ha identificado la entrada no saneada para evitar vulnerabilidades de Path Traversal y asegurar la confiabilidad en la manipulación de archivos .
- *Reducción de la deuda técnica:* Cambiar las prácticas de codificación para evitar credenciales codificadas de forma rígida y adoptar un enfoque de almacenamiento externo y manejo seguro de las mismas.
- *Implementación de Pruebas Automatizadas:* Si bien el reporte no especifica una métrica de cobertura de pruebas, es recomendable implementar pruebas automáticas para validar todas las rutas críticas y mejorar la confiabilidad del sistema.
- Revisión de Hotspots de Seguridad: Aunque no se especifican hotspots de seguridad directamente, se deben revisar las vulnerabilidades marcadas como de alta severidad para realizar una evaluación de riesgos y mitigar posibles puntos de explotación.

Impacto Esperado:

La implementación de este plan mejorará significativamente la seguridad y la calidad de la plataforma "Sustain Partners". Además de prevenir riesgos de seguridad, estas mejoras proporcionarán una base sólida para la expansión futura y la confiabilidad a largo plazo de la aplicación. Con un código más limpio y una plataforma segura, Sustain Partners puede continuar su misión de facilitar proyectos sostenibles con una mayor confianza de sus usuarios y colaboradores.

7.1. Tecnologías de información

Tecnología de información con SonarQube

La implementación tecnológica detrás de "Sustain Partners" se fundamenta en un ecosistema de software robusto y seguro, diseñado para facilitar la colaboración en proyectos de sostenibilidad. Con el análisis de SonarQube ya proporcionado, se evidencian áreas clave que requieren la adopción de tecnologías y prácticas de información mejoradas. A continuación se detallan las recomendaciones y acciones en la esfera de la Tecnología de la Información (TI):

Gestión del Código y Calidad:

- Uso de repositorios de código fuente versionado como Git para un manejo eficiente del código y colaboración.
- Integración de herramientas de revisión de código y pair programming para mejorar la calidad del software.

Cobertura de Código y Pruebas:

- Incorporación de un marco de pruebas automatizadas para evaluar cada función y módulo de la aplicación.
- Implementación de pruebas regulares para identificar y solucionar vulnerabilidades de seguridad.

Educación y Conciencia de TI:

- Programas de formación para el equipo de desarrollo en prácticas de codificación segura y uso efectivo de las herramientas de TI.

El compromiso con la tecnología de información no solo mejorará el funcionamiento interno de "Sustain Partners", sino que también potenciará la experiencia de usuario y facilitará la consecución de los Objetivos de Desarrollo Sostenible de manera colaborativa y eficiente. Estas inversiones tecnológicas están alineadas con los valores fundamentales del proyecto y reforzarán la posición de "Sustain Partners" como líder en la innovación para la sostenibilidad en Perú.

Tecnología de información con Snyk

La implementación tecnológica detrás de "Sustain Partners" se fundamenta en un ecosistema de software robusto y seguro, diseñado para facilitar la colaboración en proyectos de sostenibilidad. Con el análisis de Snyk proporcionado, se evidencian áreas clave que requieren la adopción de tecnologías y prácticas de información mejoradas. A continuación se detallan las recomendaciones y acciones en la esfera de la Tecnología de la Información (TI):

Gestión del Código y Calidad:

A. Seguimiento de Cambios de Seguridad:

- Se documentaron todas las modificaciones de código en un repositorio de código fuente versionado, asegurando que cada cambio en respuesta a las vulnerabilidades identificadas sea rastreable.
- Se habilitaron las revisiones de código obligatorias mediante solicitudes de extracción (pull requests), permitiendo que cambios críticos como la eliminación de credenciales codificadas y correcciones de XSS sean revisados y aprobados por pares.

B. Revisiones de Código Dedicadas:

- Se introdujeron sesiones de pair programming enfocadas en la seguridad, lo cual mejoró el entendimiento del equipo sobre prácticas seguras y contribuyó a la detección temprana de potenciales problemas de seguridad y mantenibilidad.

Cobertura de Código y Pruebas:

A. Pruebas Basadas en Análisis de Seguridad:

- Se estableció un proceso para desarrollar y ejecutar pruebas automatizadas basadas en los escenarios de vulnerabilidades identificados, aumentando así la cobertura de código y la confiabilidad de la aplicación.
- Se agregaron pruebas de seguridad, como pruebas de penetración, al ciclo de desarrollo para identificar y remediar proactivamente las vulnerabilidades antes del despliegue.

Educación y Conciencia de TI:

A. Capacitación en Gestión de Configuraciones Sensibles:

- Se proporcionó capacitación específica sobre la gestión segura de credenciales y configuraciones, enseñando al equipo cómo utilizar archivos .env y otras herramientas para manejar la información sensible.
- Se subrayó la importancia de la seguridad en el proceso de desarrollo, asegurando que cada miembro del equipo comprenda los riesgos asociados con prácticas inseguras y cómo mitigarlas.

B. Conciencia sobre Vulnerabilidades Específicas:

- Se organizaron sesiones de aprendizaje centradas en entender y prevenir ataques XSS y otras vulnerabilidades comunes, empleando los casos reales identificados en el análisis de Snyk como estudios de caso.
- Se enfatizó en el proceso de revisión manual de los hotspots de seguridad identificados para asegurar que los miembros del equipo estén capacitados para reconocer y actuar sobre estos riesgos.

El compromiso con la tecnología de información no solo mejorará el funcionamiento interno de "Sustain Partners", sino que también potenciará la experiencia de usuario y facilitará la consecución de los Objetivos de Desarrollo Sostenible de manera colaborativa y eficiente. Estas inversiones tecnológicas están alineadas con los valores fundamentales del proyecto y reforzarán la posición de "Sustain Partners" como líder en la innovación para la sostenibilidad en Perú.

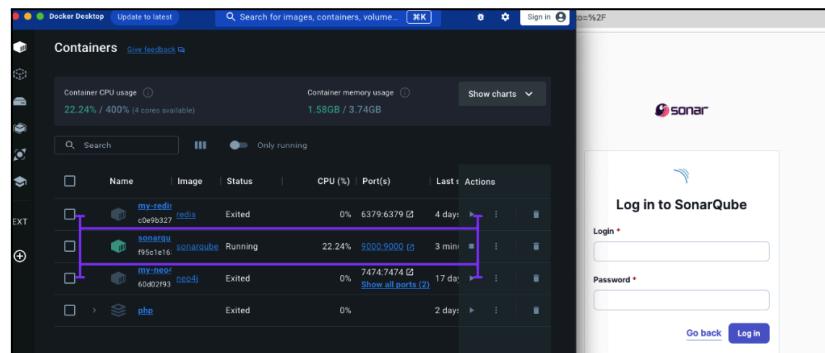
7.2. Metodología, técnicas usadas

SonarQube

- Integración de SonarQube: Configuramos SonarQube en tu entorno de Docker. Esto implica instalar el servicio de Docker y ejecutar los comandos de inicialización de SonarQube.

```
~/Desktop
docker run -d --name sonarqube -p 9000:9000 sonarqube
```

Visualizamos el contenedor creado con la imagen de SonarQube y accedemos al localhost en el puerto 9000



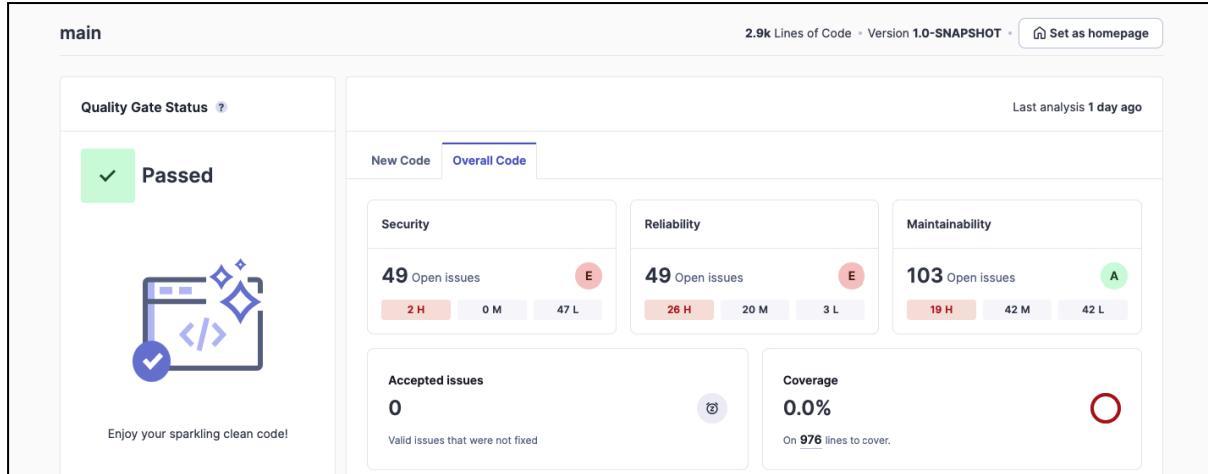
Debemos de ejecutar el código que nos da Sonarqube al momento de crear el proyecto a analizar

```
mvn clean verify sonar:sonar \
-Dsonar.projectKey=sustain-partners \
-Dsonar.projectName='sustain-partners' \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.token=sqf_fdbf4b1665f6750ff66eb656fdfc6399ac383023
```

Ejecutamos el comando dentro del proyecto que se va a analizar

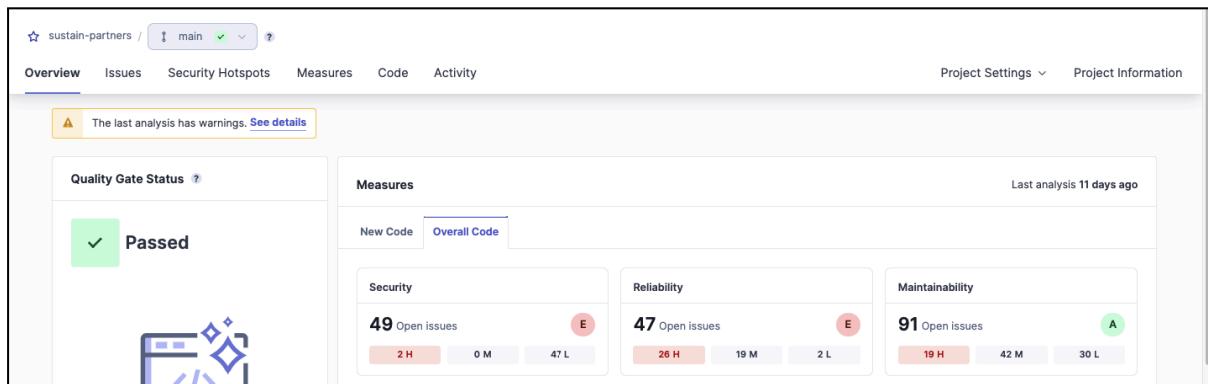
```
[ximenaortiz@MacBook-Air-de-Ximena Desktop % cd SustainPartners2
ximenaortiz@MacBook-Air-de-Ximena SustainPartners2 % mvn clean verify sonar:sonar
\
-Dsonar.projectKey=sustain-partners \
-Dsonar.projectName='sustain-partners' \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.token=sqf_fdbf4b1665f6750ff66eb656fdfc6399ac383023
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-dependency-plugin/3.6.1/maven-dependency-plugin-3.6.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-dependency-plugin/3.6.1/maven-dependency-plugin-3.6.1.pom (18 kB at 15 kB/s)
```

- Análisis Inicial: Realizamos un análisis inicial para establecer una línea base de la calidad del código actual. Este análisis identificará problemas de código, vulnerabilidades de seguridad, bugs y otros aspectos relacionados con la deuda técnica. Este análisis nos devuelve el siguiente resultado en Sonar.



- Revisión de Resultados: Tras examinar los resultados del análisis en el dashboard de SonarQube identificamos los problemas críticos que requieren atención inmediata y planificamos la refactorización.

Identificamos problemas de alto impacto (high impact) en los campos de seguridad, fiabilidad y mantenibilidad).



- **Planificación de la Refactorización:** Basado en la gravedad y el impacto de los resultados, priorizamos las correcciones en el área de Reliability, Security, Maintainability, Hotspots y por último Duplications.

Se comenzó con fiabilidad debido a la alta cantidad de problemas de alto impacto (26)

The screenshot shows the SonarQube interface for the project 'sustain-partners'. The 'Issues' tab is selected. On the left, there's a sidebar with filters and a list of issues categorized by attribute: Clean Code Attribute (Consistency: 0, Intentionality: 26, Adaptability: 0, Responsibility: 0) and Software Quality (Security: 2, Reliability: 26, Maintainability: 19). The main panel displays two code snippets with issues. The first snippet is from 'src/.../java/com/loscuchurrumines/DAO/ParticipanteDAO.java' and the second from 'src/.../java/com/loscuchurrumines/DAO/PersonaDAO.java'. Both snippets have a single issue related to resource handling: 'Use try-with-resources or close this "PreparedStatement" in a "finally" clause.' This issue is marked as 'Reliability' and 'Blocker'. The total count of 26 issues is displayed at the top right.

Continuamos con seguridad debido a que hay datos críticos involucrados como contraseñas en texto plano y API keys públicas

This screenshot shows the SonarQube interface for the same project. The 'Issues' tab is selected. The sidebar shows the same filter categories and issue counts as the previous screenshot. The main panel displays two code snippets with issues. The first snippet is from 'src/.../com/loscuchurrumines/Config/NeonConnection.java' and the second from 'src/.../loscuchurrumines/Controller/LoginController.java'. The first issue in the first snippet is 'Revoke and change this password, as it is compromised.' marked as 'Security' and 'Blocker'. The second issue in the second snippet is 'Make sure this reCaptcha key gets revoked, changed, and removed from the code.' marked as 'Security' and 'Blocker'. The total count of 2 issues is displayed at the top right.

Continuamos con mantenibilidad ya que con esto podemos reducir la duplicación de código y manejo de literales (texto plano) con constantes

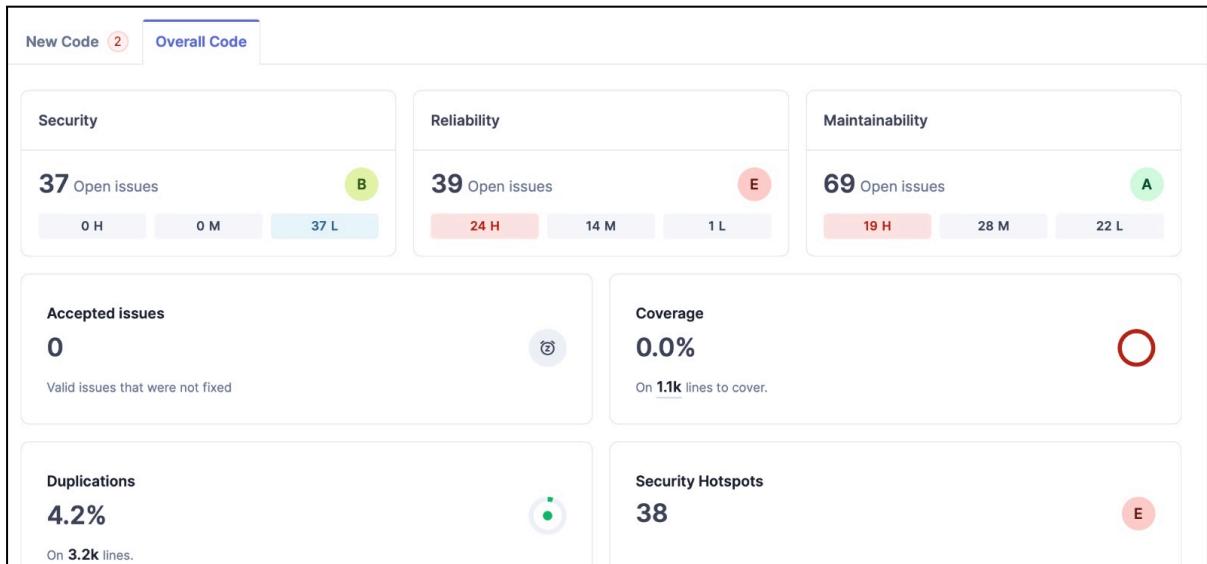
The screenshot shows a software interface for managing code quality and security. On the left, there's a sidebar with filters for 'My Issues' and 'All'. It includes sections for 'Issues in new code' and 'Clean Code Attribute' (Consistency, Intentionality, Adaptability, Responsibility), and 'Software Quality' (Security, Reliability, Maintainability). The main panel displays three specific issues under 'src.../ioscuchurruenes/Controller/LoginController.java':

- Define a constant instead of duplicating this literal "Views/Login/login.jsp" 4 times.** (Adaptability, design +) Status: Open
- Define a constant instead of duplicating this literal "error" 3 times.** (Adaptability, design +) Status: Open
- Define a constant instead of duplicating this literal "UTF-8" 3 times.** (Adaptability, design +) Status: Open

- **Implementación de Mejoras:** Se empezó a realizar las mejoras en el código basándose en las prioridades establecidas. Esto se realizó de manera iterativa:

Primera iteración

Reducimos los problemas más importantes como contraseñas expuestas, remover literales duplicadas y usos de try catch. De esta manera podemos priorizar Reliability como se planificó inicialmente.



Esta fue la manera en la que se corrigió el error de seguridad sobre contraseñas expuestas, se eliminó la URL y el password del archivo NeonConnection.java y se trabajó con variables de entorno.

The image shows two code editors side-by-side, both displaying the same Java file, `NeonConnection.java`, from different perspectives or stages of development.

SustainPartners2 main: This editor shows the original code. It includes imports for `java.sql.Connection`, `java.sql.DriverManager`, and `java.sql.SQLException`. The class `NeonConnection` contains a static block that attempts to load the `org.postgresql.Driver` class. A `getConnection()` method returns a database connection using `DriverManager.getConnection`.

```
src/main/java/com/loscuchurrumines/Config/NeonConnection.java

1 package com.loscuchurrumines.Config;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5
6 public class NeonConnection {
7
8     private static final String DATABASE_URL = "jdbc:postgresql://ep-small-haze-88987949.us-west-2.rds.amazonaws.com:5432/loscuchurrumines";
9     private static final String DATABASE_USER = "ximena-ortiz";
10    private static final String DATABASE_PASSWORD = "neq9Gm5awICO";
11
12    static {
13        try {
14            Class.forName("org.postgresql.Driver");
15        } catch (ClassNotFoundException e) {
16            e.printStackTrace();
17        }
18    }
19
20    public static Connection getConnection() {
21        try {
22            return DriverManager.getConnection(DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD);
23        } catch (SQLException ex) {
24            ex.printStackTrace();
25            return null;
26        }
27    }
28
29 }
30
```

projetoCalidad main: This editor shows the same code after refactoring. It uses a `Dotenv` library to manage environment variables. The imports now include `io.github.cdimascio.dotenv.Dotenv`. The class `NeonConnection` is now a utility class that throws an `IllegalStateException` if called directly. The `getConnection()` method delegates to a static inner class `ConnectionHolder` which handles the database connection logic using `DriverManager.getConnection`.

```
src/main/java/com/loscuchurrumines/config/NeonConnection.java > public class

1 package com.loscuchurrumines.config;
2
3 import io.github.cdimascio.dotenv.Dotenv;
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.sql.SQLException;
7
8 public class NeonConnection {
9
10    private static final Dotenv dotenv = Dotenv.load();
11    private static final String DATABASE_URL = dotenv.get("DATABASE_URL");
12    private static final String DATABASE_USER = dotenv.get("DATABASE_USER");
13    private static final String DATABASE_PASSWORD = dotenv.get("DATABASE_PASSWORD");
14
15    private NeonConnection() {
16        throw new IllegalStateException("Utility class");
17    }
18
19    public static Connection getConnection() {
20        try {
21            return DriverManager.getConnection(
22                DATABASE_URL,
23                DATABASE_USER,
24                DATABASE_PASSWORD
25            );
26        } catch (SQLException ex) {
27            return null;
28        }
29    }
30
31 }
32
33 }
```

El segundo error critico de seguridad estaba relacionado con la APIkey de Recaptcha, se trabajó de igual forma que las contraseñas, con la librería dotenv, la cual lee las variables de entorno

```
src/main/java/com/loscuchurrumines/Controller/LoginController.java public class LoginController extends HttpServlet {    private UsuarioDAO usuarioDAO = new UsuarioDAO();    private static final String SECRET_KEY = "6LFXgspAAAAAF0im5PqixZqbtybfKzGNDIUOCx";    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {        response.setContentType("text/html");        response.setCharacterEncoding("UTF-8");        PrintWriter out = response.getWriter();        out.println("Hello, World!");    } }
```

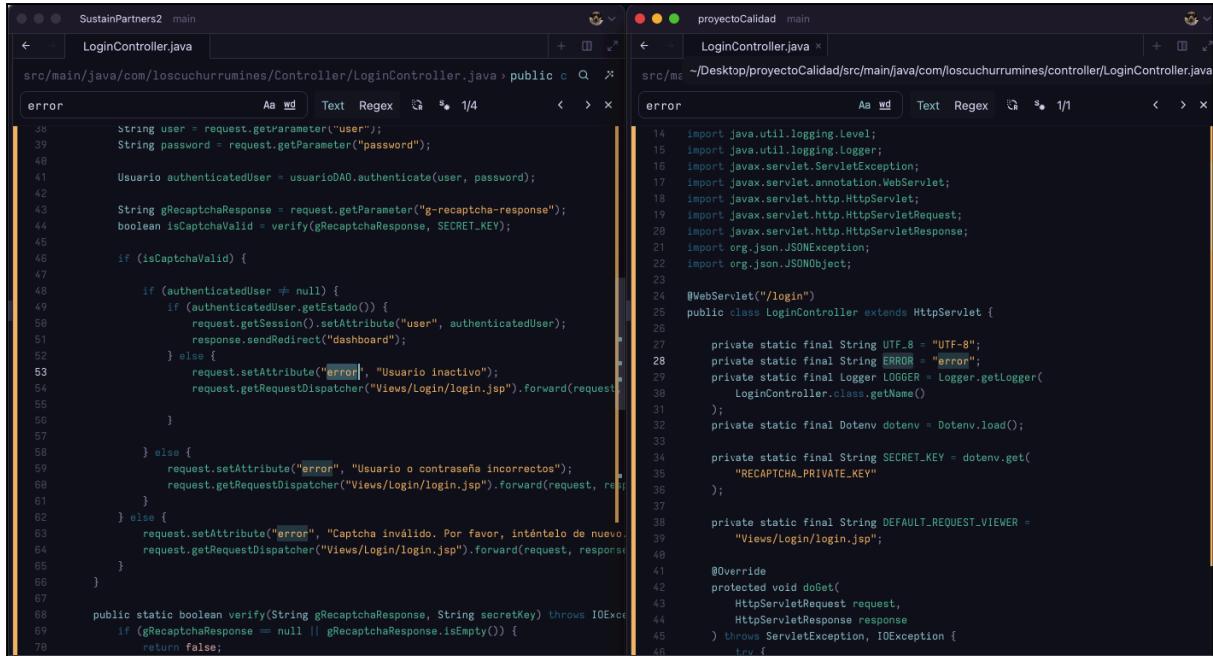
```
src/main/java/com/loscuchurrumines/controller/LoginController.java private static final Logger LOGGER = Logger.getLogger(LoginController.class.getName()); private static final Dotenv dotenv = Dotenv.load(); private static final String SECRET_KEY = dotenv.get("RECAPTCHA_PRIVATE_KEY");
```

Para trabajar de forma local se utilizó un archivo .env para declarar las contraseñas y APIkeys



```
proyectoCalidadV2 main
. env
.env
1 DATABASE_URL=jdbc:postgresql://ep-small-haze-88087949.us-east-2.awsn.eon.tech/dbSustainPartners
2 DATABASE_USER=ximena-ortiz
3 DATABASE_PASSWORD=neq9Gm5awICO
4
5 RECAPTCH_PRIVATE_KEY=6LfVXgspAAAAAF0Im5PqiXZqbtyibfKzGND1UOCx
6
```

- Remover uso de literales: Cadenas de texto repetidas que al momento de querer realizar un cambio este tenga que hacerse en todos los literales, para esto utilizamos una constante que declara la cadena una vez y utilizamos la constante en el resto del código



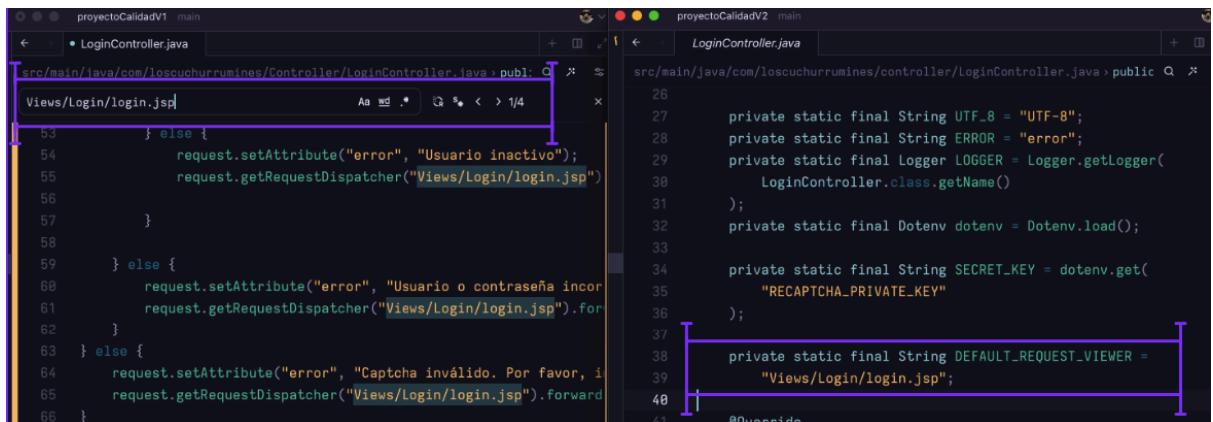
```

SustainPartners2 main
src/main/java/com/loscuchurrumines/Controller/LoginController.java public c
error Aa wd Text Regex 1/4 < > x
  88     String user = request.getParameter("user");
  89     String password = request.getParameter("password");
  90
  91     Usuario authenticatedUser = usuarioDAO.authenticate(user, password);
  92
  93     String gRecaptchaResponse = request.getParameter("g-recaptcha-response");
  94     boolean isCaptchaValid = verify(gRecaptchaResponse, SECRET_KEY);
  95
  96     if (isCaptchaValid) {
  97
  98         if (authenticatedUser != null) {
  99             if (authenticatedUser.getEstado()) {
100                 request.getSession().setAttribute("user", authenticatedUser);
101                 response.sendRedirect("dashboard");
102             } else {
103                 request.setAttribute("error", "Usuario inactivo");
104                 request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
105             }
106         } else {
107             request.setAttribute("error", "Usuario o contraseña incorrectos");
108             request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
109         }
110     }
111
112     public static boolean verify(String gRecaptchaResponse, String secretKey) throws IOException {
113         if (gRecaptchaResponse == null || gRecaptchaResponse.isEmpty())
114             return false;
115     }
116 }

proyectoCalidad main
src/main/java/com/loscuchurrumines/controller/LoginController.java public c
error Aa wd Text Regex 1/1 < > x
  14 import java.util.logging.Level;
  15 import java.util.logging.Logger;
  16 import javax.servlet.ServletException;
  17 import javax.servlet.annotation.WebServlet;
  18 import javax.servlet.http.HttpServlet;
  19 import javax.servlet.http.HttpServletRequest;
  20 import javax.servlet.http.HttpServletResponse;
  21 import org.json.JSONException;
  22 import org.json.JSONObject;
  23
  24 @WebServlet("/login")
  25 public class LoginController extends HttpServlet {
  26
  27     private static final String UTF_8 = "UTF-8";
  28     private static final String ERROR = "error";
  29     private static final Logger LOGGER = Logger.getLogger(
  30         LoginController.class.getName()
  31     );
  32     private static final Dotenv dotenv = Dotenv.load();
  33
  34     private static final String SECRET_KEY = dotenv.get(
  35         "RECAPTCHA_PRIVATE_KEY"
  36     );
  37
  38     private static final String DEFAULT_REQUEST_VIEWER =
  39         "Views/Login/login.jsp";
  40
  41     @Override
  42     protected void doGet(
  43         HttpServletRequest request,
  44         HttpServletResponse response
  45     ) throws ServletException, IOException {
  46         try {
  47
  48             if (request.getAttribute("error") != null)
  49                 response.sendRedirect("Views/Login/login.jsp");
  50             else if (request.getAttribute("user") != null)
  51                 response.sendRedirect("dashboard");
  52             else
  53                 request.setAttribute("error", "Usuario inactivo");
  54                 request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  55             }
  56         }
  57
  58         } else {
  59             request.setAttribute("error", "Usuario o contraseña incorrectos");
  60             request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  61         }
  62     }
  63
  64     } else {
  65         request.setAttribute("error", "Captcha inválido. Por favor, inténtelo de nuevo.");
  66         request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  67     }
  68
  69     public static boolean verify(String gRecaptchaResponse, String secretKey) throws IOException {
  70         if (gRecaptchaResponse == null || gRecaptchaResponse.isEmpty())
  71             return false;
  72     }
  73 }

```

De igual manera se siguieron reduciendo literales en el archivo LoginController.java, como se muestra en la imagen para Views/Login/login.jsp



```

projetoCalidadv1 main
src/main/java/com/loscuchurrumines/Controller/LoginController.java public c
Views/Login/login.jsp Aa wd Text Regex 1/4 < > x
  53     } else {
  54         request.setAttribute("error", "Usuario inactivo");
  55         request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  56     }
  57
  58     } else {
  59         request.setAttribute("error", "Usuario o contraseña incorrectos");
  60         request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  61     }
  62
  63 } else {
  64     request.setAttribute("error", "Captcha inválido. Por favor, inténtelo de nuevo.");
  65     request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  66 }
projetoCalidadv2 main
src/main/java/com/loscuchurrumines/controller/LoginController.java public c
error Aa wd Text Regex 1/1 < > x
  26
  27     private static final String UTF_8 = "UTF-8";
  28     private static final String ERROR = "error";
  29     private static final Logger LOGGER = Logger.getLogger(
  30         LoginController.class.getName()
  31     );
  32     private static final Dotenv dotenv = Dotenv.load();
  33
  34     private static final String SECRET_KEY = dotenv.get(
  35         "RECAPTCHA_PRIVATE_KEY"
  36     );
  37
  38     private static final String DEFAULT_REQUEST_VIEWER =
  39         "Views/Login/login.jsp";
  40
  41     @Override
  42     protected void doGet(
  43         HttpServletRequest request,
  44         HttpServletResponse response
  45     ) throws ServletException, IOException {
  46         try {
  47
  48             if (request.getAttribute("error") != null)
  49                 response.sendRedirect("Views/Login/login.jsp");
  50             else if (request.getAttribute("user") != null)
  51                 response.sendRedirect("dashboard");
  52             else
  53                 request.setAttribute("error", "Usuario inactivo");
  54                 request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  55             }
  56         }
  57
  58         } else {
  59             request.setAttribute("error", "Usuario o contraseña incorrectos");
  60             request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  61         }
  62     }
  63
  64     } else {
  65         request.setAttribute("error", "Captcha inválido. Por favor, inténtelo de nuevo.");
  66         request.getRequestDispatcher("Views/Login/login.jsp").forward(request, response);
  67     }
  68
  69     public static boolean verify(String gRecaptchaResponse, String secretKey) throws IOException {
  70         if (gRecaptchaResponse == null || gRecaptchaResponse.isEmpty())
  71             return false;
  72     }
  73 }

```

- Try y catch: Se corrigieron los recursos sin contemplar errores como Connection y PreparedStatement, para arreglar esto se pusieron como parámetros en el try y se manejaron sus excepciones en el catch

```

src/main/java/com/loscuchurrumines/dao/ParticipanteDAO.java
public class ParticipanteDAO {
    public boolean crearParticipante(Participante participante, int monto) {
        Connection connection = NeonConnection.getConnection();
        PreparedStatement statement;
        if (participante.getFkRol() == 1) {
            String query = "call crearDonante(?, ?, ?, ?)";
            try {
                statement = connection.prepareStatement(query);
                statement.setInt(1, participante.getFkUser());
                statement.setInt(2, participante.getFkRol());
                statement.setInt(3, participante.getFkProyecto());
                statement.setInt(4, monto);
                statement.executeUpdate();
                return true;
            } catch (Exception e) {

```

```

src/main/java/com/loscuchurrumines/dao/ParticipanteDAO.java
public class ParticipanteDAO {
    private static final Logger LOGGER = Logger.getLogger(
        ParticipanteDAO.class.getName()
    );
    public boolean crearParticipante(Participante participante, int monto) {
        String query = "call crearDonante(?, ?, ?, ?)";
        try {
            Connection connection = NeonConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(query);
            if (participante.getFkRol() == 1) {
                statement.setInt(1, participante.getFkUser());
                statement.setInt(2, participante.getFkRol());
                statement.setInt(3, participante.getFkProyecto());
                statement.setInt(4, monto);
                statement.executeUpdate();
                return true;

```

Las líneas 13, 14, 19 y 36 del proyecto de la primera versión se corrigieron en la otra versión como se observa en las líneas 19 y 20 del lado derecho

```

src/main/java/com/loscuchurrumines/dao/ParticipanteDAO.java
else {
    String query = "INSERT INTO tbparticipante (fkuser,fkrol,fkproyecto) ";
    try {
        statement = connection.prepareStatement(query);
        statement.setInt(1, participante.getFkUser());
        statement.setInt(2, participante.getFkRol());
        statement.setInt(3, participante.getFkProyecto());
        statement.executeUpdate();
    } catch (Exception e) {

```

```

src/main/java/com/loscuchurrumines/dao/ParticipanteDAO.java
public boolean crearParticipante(Participante participante, int monto) {
    String query = "call crearDonante(?, ?, ?, ?)";
    try {
        Connection connection = NeonConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement(query);
        if (participante.getFkRol() == 1) {
            statement.setInt(1, participante.getFkUser());
            statement.setInt(2, participante.getFkRol());

```

Logrando en la segunda iteración reducir más de la mitad de los issues relacionados con Reliability.



Se corrigieron los recursos en los diferentes archivos sin contemplar errores como Connection y PreparedStatement, los cuales son errores de alta prioridad, para arreglar esto se pusieron como parámetros en el try y se manejaron sus excepciones en el catch, como se observa en el archivo PersonaDAO.java, donde las líneas 51, 52 y 56 fueron corregidas en el lado derecho en las líneas 58 y 59.



```
src/main/java/com/loscuchurrumines/dao/PersonaDAO.java public class PersonaDAO {  
  
    public Persona obtenerPersona(int idPersona) {  
        jedis jedis = RedisConnection.getConnection();  
        String key = "persona:" + idPersona;  
        if (jedis.exists(key)) {  
            String cachedPersona = jedis.get(key);  
            return deserializePersona(cachedPersona);  
        }  
        Persona persona = new Persona();  
        Connection connection = NeonConnection.getConnection();  
        PreparedStatement statement;  
        ResultSet resultSet;  
        String query = "SELECT * FROM tbpersona WHERE idpersona = ?";  
        try {  
            statement = connection.prepareStatement(query);  
            statement.setInt(1, idPersona);  
            resultSet = statement.executeQuery();  
            if (resultSet.next()) {  
                persona.setIdPersona(resultSet.getInt("idpersona"));  
                persona.setNombre(resultSet.getString("nombre"));  
                persona.setApellido(resultSet.getString("apellido"));  
                persona.setCelular(resultSet.getString("celular"));  
                persona.setFotoPersona(resultSet.getString("fotopersona"))  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        } finally {  
            if (connection != null) {  
                try {  
                    connection.close();  
                } catch (SQLException e) {  
                    e.printStackTrace();  
                }  
            }  
        }  
    }  
}
```

```
src/main/java/com/loscuchurrumines/dao/PersonaDAO.java public class PersonaDAO {  
  
    public Persona obtenerPersona(int idPersona) {  
        jedis jedis = RedisConnection.getConnection();  
        String key = "persona:" + idPersona;  
        if (jedis.exists(key)) {  
            String cachedPersona = jedis.get(key);  
            return deserializePersona(cachedPersona);  
        }  
        Persona persona = new Persona();  
        ResultSet resultSet;  
        String query = "SELECT * FROM tbpersona WHERE idpersona = ?";  
        try {  
            Connection connection = NeonConnection.getConnection();  
            PreparedStatement statement = connection.prepareStatement(query);  
            statement.setInt(1, idPersona);  
            resultSet = statement.executeQuery();  
            if (resultSet.next()) {  
                persona.setIdPersona(resultSet.getInt("idpersona"));  
                persona.setNombre(resultSet.getString("nombre"));  
                persona.setApellido(resultSet.getString("apellido"));  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Luego continuamos con los errores de prioridad media, en este caso el error indicaba que las declaraciones de las clases del controlador deben ir dentro de cada método como se observa en el lado derecho en las líneas 45 y 62

```
src/main/java/com/loschuchurumines/Controller/DashboardController.java
```

```
private ProyectoDAO proyectoDAO = new ProyectoDAO();
private PersonaDAO personaDAO = new PersonaDAO();

protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    if (!establecerUsuario(request)) {
        response.sendRedirect("persona");
    } else {
        establecerProyectos(request);
        request.getRequestDispatcher("Views/Dashboard/dashboard.jsp").forward(request, response);
    }
}

private boolean establecerUsuario(HttpServletRequest request) {
    Usuario authenticatedUser = (Usuario) request.getSession().getAttribute("authenticatedUser");
    int idUser = (int) authenticatedUser.getIdUser();

    Persona persona = personaDAO.obtenerPersona(idUser);
    if (persona != null) {
        request.getSession().setAttribute("persona", persona);
    } else {
        return false;
    }
    return true;
}

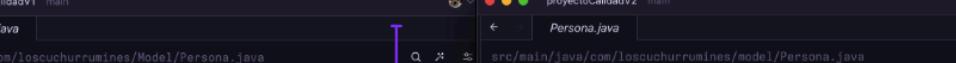
private void establecerProyectos(HttpServletRequest request) {
    ProyectoDAO proyectoDAO = new ProyectoDAO();
    List<Proyecto> proyectos = proyectoDAO.obtenerProyectos();
    request.setAttribute("proyectos", proyectos);
}
```

```
src/main/java/com/loschuchurumines/controller/DashboardController.java
```

```
private boolean establecerUsuario(HttpServletRequest request) {
    PersonaDAO personaDAO = new PersonaDAO();
    Usuario authenticatedUser = (Usuario) request.getSession()
            .getAttribute("user");
    int idUser = authenticatedUser.getIdUser();

    Persona persona = personaDAO.obtenerPersona(idUser);
    if (persona != null) {
        request.getSession().setAttribute("persona", persona);
    } else {
        return false;
    }
    return true;
}
```

El siguiente error estaba relacionado con el paso de la clase persona a través de una sesión sin ser serializado, esto se resolvió serializando la clase persona como se observa en la clase persona



```
src/main/java/com/loscuchurrumines/Model/Persona.java
```

```
1 package com.loscuchurrumines.Model;
2
3 public class Persona {
4     private int idPersona;
5     private String nombre;
6     private String apellido;
7     private String celular;
8     private String fechaNacimiento;
9     private String fotoPersona;
10    private String sexo;
11    private int fkUser;
12    public Persona(){}
```

```
src/main/java/com/loscuchurrumines/model/Persona.java
```

```
1 package com.loscuchurrumines.model;
2
3 import java.io.Serializable;
4
5 public class Persona implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     private int idPersona;
9     private String nombre;
10    private String apellido;
11    private String celular;
```

Debido a que es igual que el error anterior declaramos las clases del controlador dentro de cada método, en la imagen se observa como se soluciono declarando en la línea 29
ProyectoDAO

```

    public class ParticiparController extends HttpServlet{
        private ProyectoDAO proyectoDAO = new ProyectoDAO();
        private ParticipanteDAO participanteDAO = new ParticipanteDAO();
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            List<Integer> modalidades = proyectoDAO.obtenerModalidades();
            request.getSession().setAttribute("modalidades", modalidades);
            request.getRequestDispatcher("/Views/Proyecto/formularioPa
        }
    }

```

```

    @Override
    protected void doGet(
        HttpServletRequest request,
        HttpServletResponse response
    ) throws ServletException, IOException {
        try {
            ProyectoDAO proyectoDAO = new ProyectoDAO();
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            List<Integer> modalidades = proyectoDAO.obtenerModalidades();
            idProyecto
        );
        request.getSession().setAttribute("modalidades", modalidades);
    }
}

```

En la línea 62 se observa como se declaró participante dentro del método doPost

```

    public class ParticiparController extends HttpServlet{
        private ProyectoDAO proyectoDAO = new ProyectoDAO();
        private ParticipanteDAO participanteDAO = new ParticipanteDAO();
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            List<Integer> modalidades = proyectoDAO.obtenerModalidades();
            request.getSession().setAttribute("modalidades", modalidades);
            request.getRequestDispatcher("/Views/Proyecto/formularioPa
        }
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            int fkUser = ((Persona) request.getSession().getAttribute("persona")).getPkUser();
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            int fkRol = Integer.parseInt(request.getParameter("modalidad"));
            int monto = 0;

```

```

    @Override
    protected void doPost(
        HttpServletRequest request,
        HttpServletResponse response
    ) throws ServletException, IOException {
        try {
            int fkUser =
                ((Persona) request
                    .getSession()
                    .getAttribute("persona")).getPkUser();
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            int fkRol = Integer.parseInt(request.getParameter("modalidad"));

            Participante participante = new Participante();
            participante.setPkUser(fkUser);
            participante.setFkRol(fkRol);
            participante.setPkProyecto(idProyecto);

```

El siguiente error decia que debiamos de eliminar la condicional debido a que el contenido era el mismo, se quitando la condicional

```

    public class ParticiparController extends HttpServlet{
        private ProyectoDAO proyectoDAO = new ProyectoDAO();
        private ParticipanteDAO participanteDAO = new ParticipanteDAO();
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            List<Integer> modalidades = proyectoDAO.obtenerModalidades();
            request.getSession().setAttribute("modalidades", modalidades);
            request.getRequestDispatcher("/Views/Proyecto/formularioPa
        }
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            int fkUser = ((Persona) request.getSession().getAttribute("persona")).getPkUser();
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            int fkRol = Integer.parseInt(request.getParameter("modalidad"));

            Participante participante = new Participante();
            participante.setPkUser(fkUser);
            participante.setFkRol(fkRol);
            participante.setPkProyecto(idProyecto);

            boolean participacion =
                participanteDAO.crearParticipante(participante, monto);
            if (participacion) {
                response.sendRedirect("proyecto?vista=detalleProyecto&id="
                    + idProyecto);
            } else {
                response.sendRedirect("proyecto?vista=detalleProyecto&id="
                    + idProyecto);
            }
        }
    }

```

```

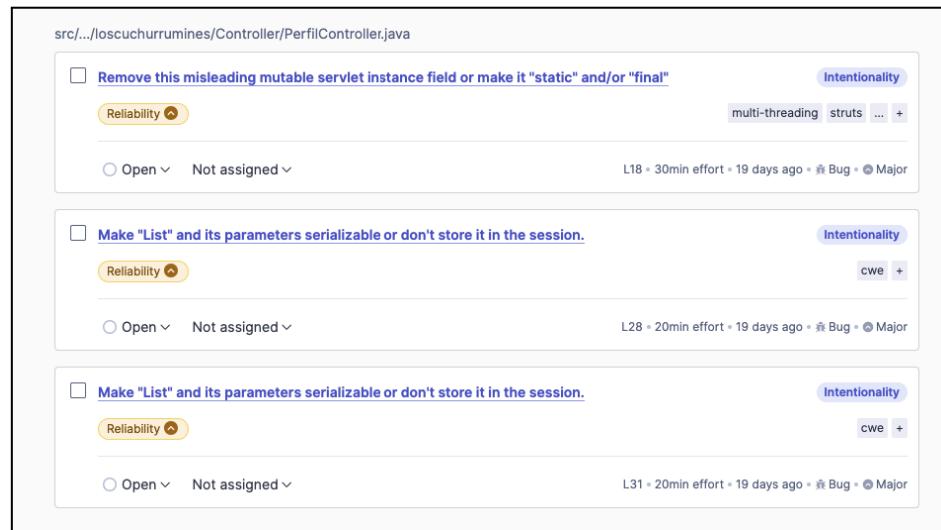
    @Override
    protected void doPost(
        HttpServletRequest request,
        HttpServletResponse response
    ) throws ServletException, IOException {
        try {
            int fkUser =
                ((Persona) request
                    .getSession()
                    .getAttribute("persona")).getPkUser();
            int idProyecto = Integer.parseInt(request.getParameter("id"));
            int fkRol = Integer.parseInt(request.getParameter("modalidad"));

            Participante participante = new Participante();
            participante.setPkUser(fkUser);
            participante.setFkRol(fkRol);
            participante.setPkProyecto(idProyecto);

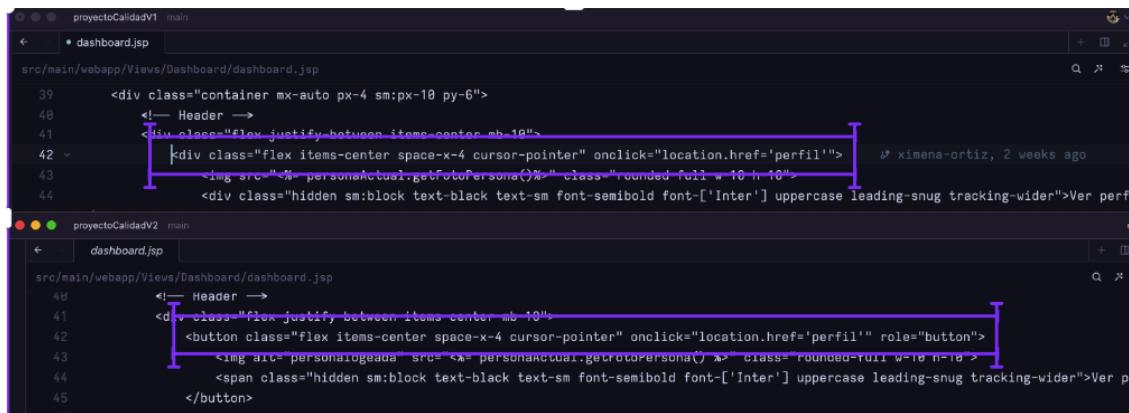
            response.sendRedirect(
                "proyecto?vista=detalleProyecto&id=" + idProyecto
            );
        } catch (IOException | NumberFormatException e) {
            LOGGER.log(Level.SEVERE, e.getMessage());
        }
    }
}

```

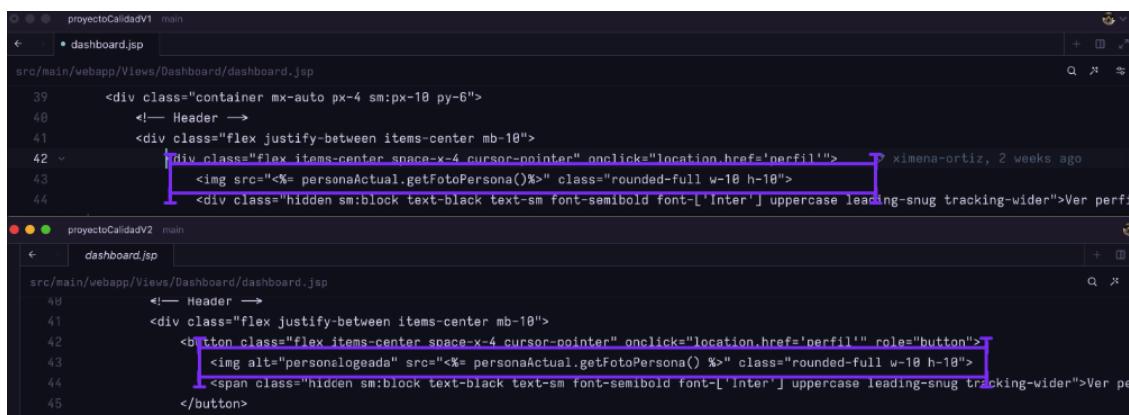
Los siguientes errores se encontraban en el archivo PerfilController.java y para solucionarlos comenzamos declarando las clases del controlador dentro de cada método y luego serializamos la clase proyecto



El siguiente error indica el uso incorrecto de un atributo div para hacer la función de un botón



El error indicaba que teníamos que agregar el atributo “Alt”, el cual se vera solo si no se carga la imagen



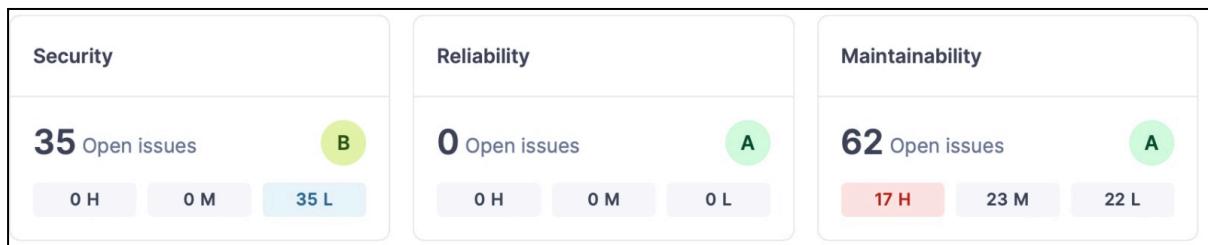
El error indicaba que la palabra “imagen” era redundante por lo que fue modificado el Alt para solucionarlo

```

187 <div>
188   
189   <%-- %>
190   <p class="mb-4"><%= proyecto.getDescripcion() %></p>
191
192   <div class="grid grid-cols-2 gap-4 mb-4">

```

El objetivo a seguir era eliminar reliability para continuar con security lo cual se logró terminar en la tercera iteración.



Continuamos reduciendo los errores de seguridad, este error nos indica que no estamos manejando la excepción de sendRedirect, la cual es la IOException

```

21 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
22     if (!establecerUsuario(request)) {
23         response.sendRedirect("persona");
24     } else {
25         establecerProyectos(request);
26         request.getRequestDispatcher("Views/Dashboard/dashboard.jsp").forward(request, response);
27     }
28 }
29
30
31 private boolean establecerUsuario(HttpServletRequest request) {
32     Usuario authenticatedUser = (Usuario) request.getSession().getAttribute("authenticatedUser");
33     int idUser = (int) authenticatedUser.getIdUser();
34
35     Persona persona = personaDAO.obtenerPersona(idUser);
36     if (persona != null) {
37         request.getSession().setAttribute("persona", persona);
38     }
39 }

```

```

24
25     @Override
26     protected void doGet(
27         HttpServletRequest request,
28         HttpServletResponse response
29     ) throws ServletException, IOException {
30         try {
31             if (!establecerUsuario(request)) {
32                 response.sendRedirect("persona");
33             } else {
34                 establecerProyectos(request);
35                 request.getRequestDispatcher("Views/Dashboard/dashboard.jsp").forward(request, response);
36             }
37         } catch (IOException | ServletException e) {
38             LOGGER.log(Level.SEVERE, "Exception caught in doGet method");
39         }
40     }
41 }
42

```

El error tambien se encontraba dentro de DashboardController.java nos indicaba que no estabamos manejando la excepción de ServletException y IOException

```

src/main/java/com/loscuchurrumines/controller/DashboardController.java
21
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         if (!establecerUsuario(request)) {
24             response.sendRedirect("persona");
25         } else {
26             establecerProyectos(request);
27             request.getRequestDispatcher("Views/Dashboard/dashboard.jsp")
28                 .forward(request, response);
29         }
30     }
31

```

```

src/main/java/com/loscuchurrumines/controller/DashboardController.java
32         response.sendRedirect("persona");
33     } else {
34         establecerProyectos(request);
35         request
36             .getRequestDispatcher("Views/Dashboard/dashboard.jsp")
37             .forward(request, response);
38     }
39 } catch (IOException | ServletException e) {
40     LOGGER.log(Level.SEVERE, "Exception caught in doGet method");
41 }
42

```

El error dentro del archivo ParticiparController.java nos indicaba que no estábamos manejando correctamente la excepción NumberFormatException se corrigió en el catch de la línea 40

```

src/main/java/com/loscuchurrumines/controller/ParticiparController.java
17     @WebServlet("/participar")
18     public class ParticiparController extends HttpServlet {
19
20         private ProyectoDAO proyectoDAO = new ProyectoDAO();
21         private ParticipanteDAO participanteDAO = new ParticipanteDAO();
22         protected void doGet(HttpServletRequest request, HttpServletResponse response)
23             throws ServletException, IOException {
24             int idProyecto = Integer.parseInt(request.getParameter("id"));
25             List<Integer> modalidades = proyectoDAO.obtenerModalidadesProyecto(idProyecto);
26             request.getSession().setAttribute("modalidades", modalidades);
27             request.getRequestDispatcher("/Views/Proyecto/formularioParticipar.jsp").forward(request, response);
28         }
29
30         protected void doPost(HttpServletRequest request, HttpServletResponse response)
31             throws ServletException, IOException {
32             int fkUser = ((Persona) request.getSession().getAttribute("persona")).getId();
33             int idProyecto = Integer.parseInt(request.getParameter("id"));
34             int fkRol = Integer.parseInt(request.getParameter("modalidad"));
35             int monto = 0;
36             if (request.getParameter("monto") == null) {
37                 monto = 0;
38             } else {

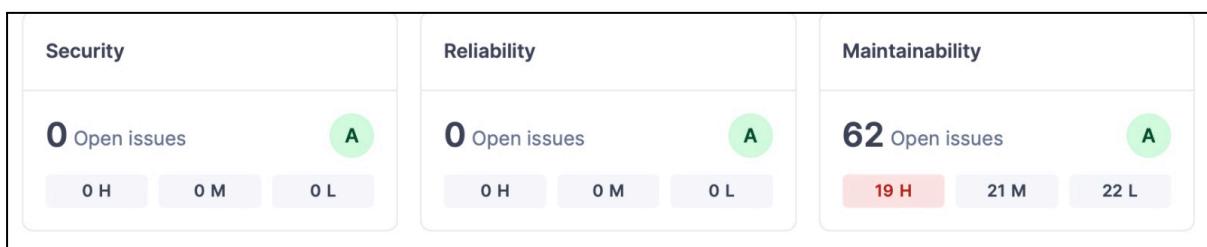
```

```

src/main/java/com/loscuchurrumines/controller/ParticiparController.java
24     protected void doGet(
25         HttpServletRequest request,
26         HttpServletResponse response
27     ) throws ServletException, IOException {
28         try {
29             ProyectoDAO proyectoDAO = new ProyectoDAO();
30             int idProyecto = Integer.parseInt(request.getParameter("id"));
31             List<Integer> modalidades = proyectoDAO.obtenerModalidadesProyecto(idProyecto);
32         };
33         request.getSession().setAttribute("modalidades", modalidades);
34         request
35             .getRequestDispatcher(
36                 "/Views/Proyecto/formularioParticipar.jsp"
37             )
38             .forward(request, response);
39     } catch (IOException | ServletException | NumberFormatException e) {
40         try {
41             response.sendRedirect("proyecto");
42         } catch (IOException ex) {
43             LOGGER.log(Level.SEVERE, ex.getMessage());
44         }
45     }
46 }

```

La siguiente iteración se realizó con la culminación de Security llevado a cero y se continuó reduciendo los errores de medio y de alto impacto en mantenibilidad



Se agregó manejo de excepciones con loggers en los controladores para solucionar el problema de impacto medio de trabajar los errores con system.out

```

src/main/java/com/loscuchurrumines/controller/DashboardController.java
22 protected void doGet(HttpServletRequest request, HttpServletResponse response)
23 throws ServletException, IOException {
24 if (establecerUsuario(request)) {
25 response.sendRedirect("persona");
26 } else {
27 establecerProyectos(request);
28 request.getRequestDispatcher("Views/Dashboard/dashboard.jsp").forward(request, response);
29 }
30 }
31 }

private boolean establecerUsuario(HttpServletRequest request) {
32 Usuario authenticatedUser = (Usuario) request.getSession().getAttribute("authenticatedUser");
33 int idUser = (int) authenticatedUser.getIdUser();
34 Persona persona = personaDAO.obtenerPersona(idUser);
35 if (persona != null) {
36 request.getSession().setAttribute("persona", persona);
37 } else {
38 }
39 }
40 }

src/main/java/com/loscuchurrumines/controller/DashboardController.java
25 @Override
26 protected void doGet(
27 HttpServletRequest request,
28 HttpServletResponse response
29 ) throws ServletException, IOException {
30 try {
31 if (establecerUsuario(request)) {
32 response.sendRedirect("persona");
33 } else {
34 establecerProyectos(request);
35 request.getRequestDispatcher("Views/Dashboard/dashboard.jsp").forward(request, response);
36 }
37 } catch (IOException | ServletException e) {
38 LOGGER.log(Level.SEVERE, "Exception caught in doGet method", e);
39 }
40 }
41 }
42 }
43 }

```

El siguiente error es sobre literales repetidos, se corrigió declarando las constantes en representación de los literales como se observa desde la línea 23 hasta la 31

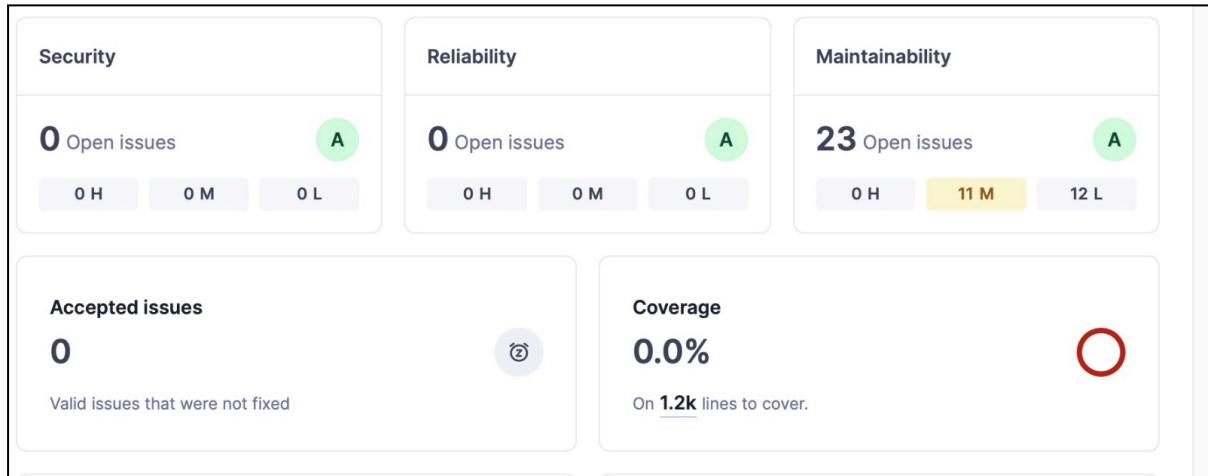
```

src/main/java/com/loscuchurrumines/dao/ProyectoDAO.java
21 public class ProyectoDAO {
22 public Proyecto obtenerProyecto(int idProyecto) {
23 Jedis jedis = RedisConnection.getConnection();
24 String key = "proyecto:" + idProyecto;
25
26 if (jedis.exists(key)) {
27 String cachedProyecto = jedis.get(key);
28 return deserializeProyecto(cachedProyecto);
29 }
30
31 Proyecto proyecto = new Proyecto();
32 Connection connection = NeonConnection.getConnection();
33 PreparedStatement statement; // ximena-ortiz, 2 weeks ago
34 ResultSet resultSet;
35 String query = "SELECT * FROM tbiproyecto WHERE idprojeto = ?";
36
37 try {
38 statement = connection.prepareStatement(query);
39 statement.setInt(1, idProyecto);
40 resultSet = statement.executeQuery();
41 if (resultSet.next()) {
42 proyecto.setIdProyecto(resultSet.getInt("idprojeto"));
43 proyecto.setNombre(resultSet.getString("nombre"));
44 proyecto.setDescripcion(resultSet.getString("descripcion"));
45 proyecto.setObjetivo(resultSet.getString("objetivo"));
46 proyecto.setFoto(resultSet.getString("foto"));
47 proyecto.setEstado(resultSet.getBoolean("estado"));
48 proyecto.setFkRegion(resultSet.getInt("fkregion"));
49 }
50 }
51 catch (SQLException e) {
52 System.out.println("Error al obtener el proyecto: " + e.getMessage());
53 }
54 finally {
55 try {
56 if (resultSet != null) resultSet.close();
57 if (statement != null) statement.close();
58 if (connection != null) connection.close();
59 }
60 catch (SQLException e) {
61 System.out.println("Error al cerrar las conexiones: " + e.getMessage());
62 }
63 }
64 }
65 }

src/main/java/com/loscuchurrumines/dao/ProyectoDAO.java
20
21 public class ProyectoDAO {
22
23 private static final String IDPROYECTO = "idprojeto";
24 private static final String NOMBRE = "nombre";
25 private static final String DESCRIPCION = "descripcion";
26 private static final String OBJETIVO = "objetivo";
27 private static final String FOTO = "foto";
28 private static final String ESTADO = "estado";
29 private static final String FKREGION = "fkregion";
30 private static final String FKUSER = "fkuser";
31 private static final String FKFONDO = "fkfondo";
32
33 private static final Logger LOGGER = Logger.getLogger(
34 ProyectoDAO.class.getName()
35 );
36
37 public Proyecto obtenerProyecto(int idProyecto) {
38 Jedis jedis = RedisConnection.getConnection();
39 String key = "proyecto:" + idProyecto;
40
41 if (jedis.exists(key)) {
42 String cachedProyecto = jedis.get(key);
43 return deserializeProyecto(cachedProyecto);
44 }
45
46 Proyecto proyecto = new Proyecto();
47
48 }
49
50 }
51
52 }
53
54 }
55
56 }
57
58 }
59
60 }
61
62 }
63
64 }
65
66 }
67
68 }
69
70 }
71
72 }
73
74 }
75
76 }
77
78 }
79
79 }
80
81 }
82
83 }
84
85 }
86
87 }
88
89 }
89 }
90
91 }
92
93 }
94
95 }
96
97 }
98
99 }
99 }

```

Con la seguridad y fiabilidad al cero, nos centramos en corregir los errores de medio y de bajo impacto de mantenibilidad



El error decía que debíamos agregar el “decorator” @override según los estándares de java mvc

```

SustainPartners2 main
LoginController.java
src/main/java/com/loscuchurrumines/Controller/LoginController.java
31 }
32
33 protected void doPost(HttpServletRequest request, HttpServletResponse response
throws ServletException, IOException {
34     response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");
35     response.setHeader("Pragma", "no-cache");
36     response.setDateHeader("Expires", 0);
37     String user = request.getParameter("user");
38     String password = request.getParameter("password");
39
40     Usuario authenticatedUser = usuarioDAO.authenticate(user, password);
41
42     String gRecaptchaResponse = request.getParameter("g-recaptcha-response");
43     boolean isCaptchaValid = verify(gRecaptchaResponse, SECRET_KEY);
44
45     if (isCaptchaValid) {
46
47         ...
48     }
49 }
50 }

proyectoCalidad main
LoginController.java
src/main/java/com/loscuchurrumines/controller/LoginController.java
61 @Override
62 protected void doPost(
63     HttpServletRequest request,
64     HttpServletResponse response
65 ) throws ServletException, IOException {
66     UsuarioDAO usuarioDAO = new UsuarioDAO();
67     try {
68         response.setHeader(
69             "Cache-Control",
70             "no-cache, no-store, must-revalidate"
71         );
72         response.setHeader("Pragma", "no-cache");
73         response.setDateHeader("Expires", 0);
74         String user = request.getParameter("user");
75         String password = request.getParameter("password");
76     }
77 }
78 }
```

El siguiente error indicaba un problema con el estándar de variable de java, para el cual se puso la variable con la convención de java de nombrado de variables(camelcase)

```

projetoCalidadV1 main
ParticiparController.java ProjektDAO.java
src/main/java/com/loscuchurrumines/DAO/ProjektDAO.java
410
411     public int getFondo(int idProyecto) {
412         Connection connection = NeonConnection.getConnection();
413         PreparedStatement statement;
414         String Sql = "SELECT monto FROM tbproyecto INNER JOIN tbfondo on fkfondo = ";
415         try {
416             statement = connection.prepareStatement(Sql);
417             statement.setInt(1, idProyecto);
418             ResultSet resultSet = statement.executeQuery();
419             if (resultSet.next()) {
420                 return resultSet.getInt("monto");
421             }
422             return 0;
423         } catch (Exception e) {
424             e.printStackTrace();
425             return 0;
426         }
427     }
428 }

projetoCalidadV2 main
ProjektDAO.java
src/main/java/com/loscuchurrumines/deu/ProjektDAO.java
415
416     public int getFondo(int idProyecto) {
417         String sql =
418             "SELECT monto FROM tbproyecto INNER JOIN tbfondo on fkfondo = ";
419         try {
420             Connection connection = NeonConnection.getConnection();
421             PreparedStatement statement = connection.prepareStatement(sql);
422         } {
423             statement.setInt(1, idProyecto);
424             ResultSet resultSet = statement.executeQuery();
425             if (resultSet.next()) {
426                 return resultSet.getInt("monto");
427             }
428             return 0;
429         } catch (Exception e) {
430             LOGGER.log(Level.SEVERE, null, e);
431             return 0;
432         }
433     }
434 }
```

El siguiente error indica que debían de borrarse las líneas comentadas

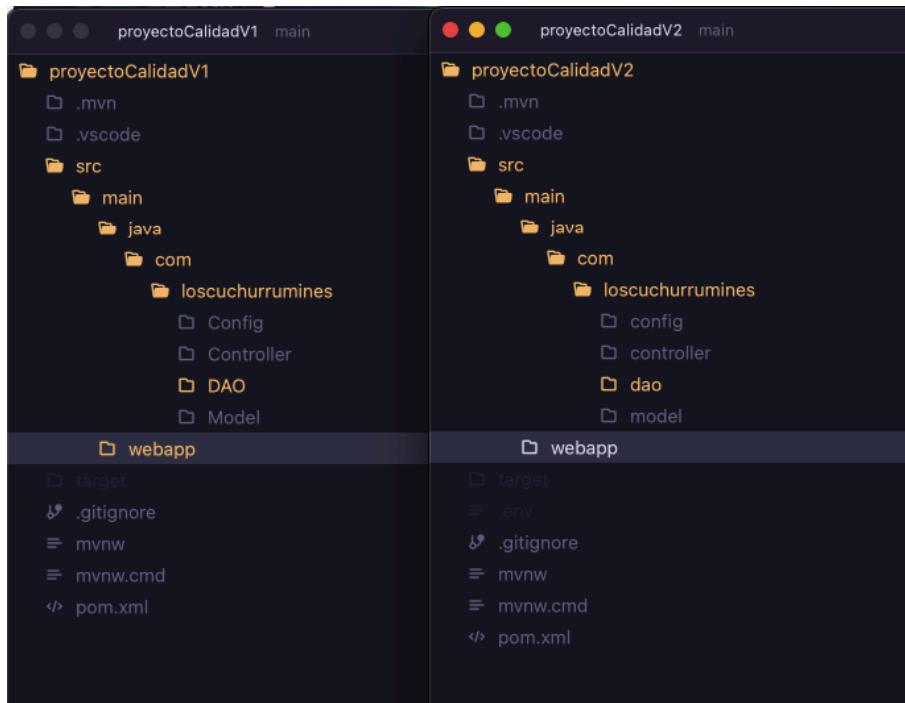
```

src/main/java/com/loscuchurrumines/dao/ProyectoDAO.java
public class ProyectoDAO {
    ...
    public int obtenerMontoProyecto(int idProyecto) {
        // jedis jedis = RedisConnection.getConnection();
        // String key = "montoProyecto:" + idProyecto;
        // if (jedis.exists(key)) {
        //     return Integer.parseInt(jedis.get(key));
        // }
        Connection connection = NeonConnection.getConnection();
        PreparedStatement statement;
        try {
            String query = "SELECT SUM(monto) AS total FROM tbdonacion WHERE fkproyecto = ?";
            statement = connection.prepareStatement(query);
            statement.setInt(1, idProyecto);
            ResultSet resultSet = statement.executeQuery();
            int total = 0;
            if (resultSet.next()) {
                total = resultSet.getInt("total");
                // jedis.set(key, String.valueOf(total));
            }
            return total;
        } catch (Exception e) {
            ...
        }
    }
}

src/main/java/com/loscuchurrumines/dao/ProyectoDAO.java
public class ProyectoDAO {
    ...
    public int obtenerMontoProyecto(int idProyecto) {
        String query =
        "SELECT SUM(monto) AS total FROM tbdonacion WHERE fkproyecto = ?";
        try {
            Connection connection = NeonConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setInt(1, idProyecto);
            ResultSet resultSet = statement.executeQuery();
            int total = 0;
            if (resultSet.next()) {
                total = resultSet.getInt("total");
            }
            return total;
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, null, e);
            return 0;
        }
    }
    public List<Proyecto> seteoProyectos(PreparedStatement statement)
}

```

El siguiente error indicó que los nombres de los paquetes deben modificarse debido a la convención de java del nombrado de paquete



El error indicaba que no hicieramos las consultas utilizando “*”, se corrigió llamando a cada columna independientemente

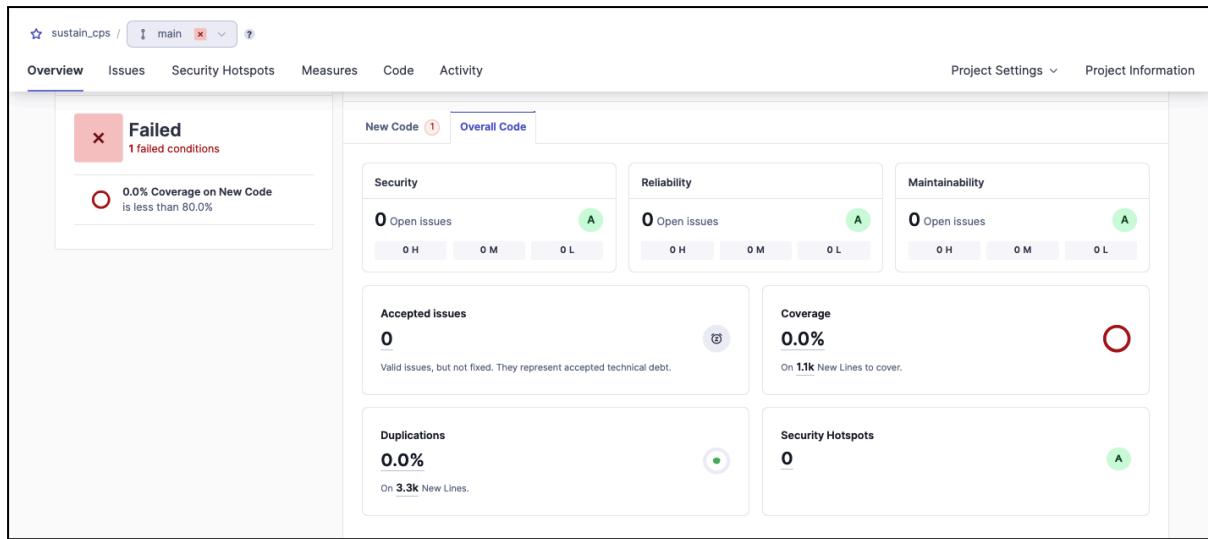
```

src/main/java/com/loscuchurrumines/dao/PersonaDAO.java
public class PersonaDAO {
    ...
    public Persona obtenerPersona(int idpersona) {
        String query = "SELECT * FROM tbpersona WHERE idpersona = ?";
        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);
            if (resultSet.next()) {
                persona.setIdPersona(resultSet.getInt("idpersona"));
                ...
            }
        } catch (SQLException e) {
            ...
        }
    }
}

src/main/java/com/loscuchurrumines/dao/PersonaDAO.java
public class PersonaDAO {
    ...
    public Persona obtenerPersona(int idpersona) {
        Persona persona = new Persona();
        ResultSet resultSet;
        String query = "SELECT idpersona,nombre,apellido,celular,fotopersona,fechanacimiento,sexo,fkuser FROM tbpersona WHERE idpersona = ?";
        try {
            Connection connection = NeonConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(query);
            ...
        } catch (SQLException e) {
            ...
        }
    }
}

```

Por último se redujo toda la deuda técnica a cero como se evidencia en la imagen



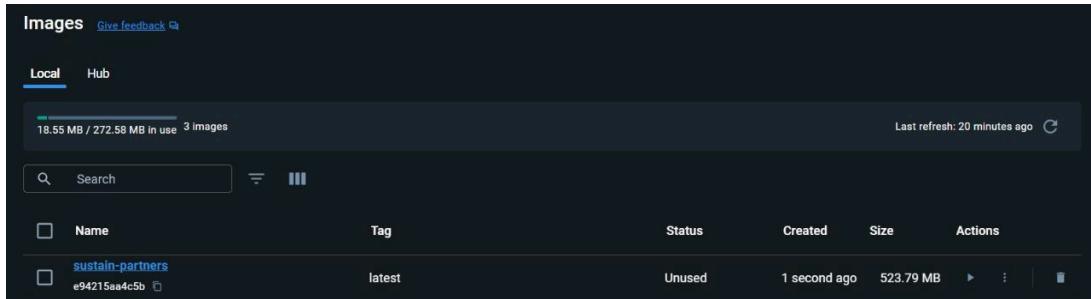
- **Documentación de Cambios y Aprendizajes:** Crear la documentación de los cambios realizados es crucial para futuras auditorías y mantenimientos del código.

Snyk

- Integración de Snyk: El Dockerfile muestra un proceso de construcción de una imagen Docker en dos etapas. En la primera etapa, se utiliza una imagen de Maven con OpenJDK 17 para construir la aplicación. En la segunda etapa, se parte de la imagen de Tomcat más reciente, se copia el archivo .war resultante de la construcción y se expone el puerto 8080. La integración de Snyk busca monitorear y corregir vulnerabilidades de seguridad durante la construcción de la imagen Docker.

```
dockerfile
FROM maven:3.8.4-openjdk-17 AS build
COPY pom.xml /usr/src/app/
COPY src /usr/src/app/src/
WORKDIR /usr/src/app
RUN mvn clean package
FROM tomcat:latest AS base
WORKDIR /usr/local/tomcat/webapps
EXPOSE 8080
COPY --from=build /usr/src/app/target/sustainpartners.war .
CMD ["catalina.sh", "run"]
```

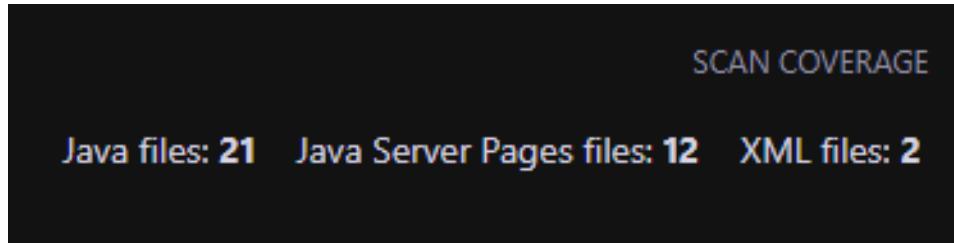
Se creó con éxito una imagen Docker local denominada 'sustain-partners'. El tamaño de la imagen es de aproximadamente 523.79 MB, lo cual es relevante para consideraciones de almacenamiento y despliegue.



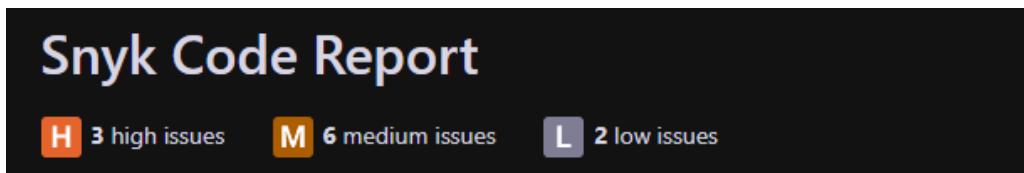
The screenshot shows the Docker Images interface. At the top, it says 'Images' and 'Give feedback'. Below that, there are tabs for 'Local' and 'Hub', with 'Local' selected. It displays 3 images with a total size of 18.55 MB / 272.58 MB in use. A search bar is present. The table below lists the images:

Name	Tag	Status	Created	Size	Actions
sustain-partners e94215aa4c5b	latest	Unused	1 second ago	523.79 MB	[More]

- Análisis Inicial: El informe de Snyk proporciona una visión detallada de las rutas del sistema de archivos escaneadas en el contexto del proyecto "sustain-partners". Estas rutas incluyen áreas como la base de datos, la carpeta webapps de Tomcat y sus bibliotecas asociadas, así como las bibliotecas de OpenJDK.



- Revisión de Resultados: Podemos observar la descripción de cada vulnerabilidad encontrada en el análisis. Se dividen en high issues (problemas altos), medium issues (problemas medios) y low issues (problemas bajos). De los altos se encontraron 3, de los medios 6 y los bajos 2.



- Planificación de la Refactorización: En este problema (high issues) se detectó que existe una vulnerabilidad de XSS en el archivo formularioParticipar.jsp en la línea 23. Esta vulnerabilidad podría permitir a un atacante injectar código JavaScript malicioso en el atributo "action" del formulario. Cuando un usuario legítimo visite esta página y envíe el formulario, el código JavaScript malicioso se ejecutaría en el contexto del navegador del usuario

The screenshot shows the Snyk security tool interface. At the top, it displays a red warning icon followed by the text "Cross-site Scripting (XSS)". Below this, it says "SNYK-CODE | CWE-79 | XSS". On the right, there are two buttons: "Data Flow" (highlighted in orange) and "Fix Analysis". The main content area contains a message: "Unsanitized input from an HTTP parameter flows into print, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS)." Below this message, it says "Found in: src/main/webapp/Views/Proyecto/formularioParticipar.jsp (line : 23)". Under the heading "Data Flow", there is a code snippet from the JSP file:

```
src/main/webapp/Views/Proyecto/formularioParticipar.jsp
23:41 <form action="/participar?id=<%= request.getParameter("id")%>" method="post" class="bg-white p-6 rounded-lg shadow-lg space-y-4">
23:41   <input type="text" name="id" value="<%= request.getParameter("id")%>"/>
23:41 </form>
```

Annotations show the flow of data from the "request.getParameter("id")" source (labeled 0) through the "value" attribute (labeled 1) to the "action" attribute (labeled 2).

Para mitigar este problema, es necesario sanitizar y/o escapar adecuadamente cualquier entrada del usuario antes de incluirla en la salida HTML. Además, es importante validar y limitar los datos de entrada a solo aquellos que sean estrictamente necesarios para evitar la inserción de datos maliciosos.

El problema identificado es una vulnerabilidad de Cross-site Scripting (XSS). Esta vulnerabilidad ocurre cuando datos no sanitizados provenientes de un parámetro HTTP se incorporan directamente en el código HTML de una página web y luego se envían al navegador del usuario. Un atacante puede aprovechar esta vulnerabilidad para inyectar y ejecutar scripts maliciosos en el contexto del navegador del usuario, lo que podría conducir a ataques como robo de cookies, redirección de usuarios a sitios web maliciosos o modificación del contenido de la página. En el archivo searchBar.jsp en la línea 29, el valor del parámetro "query" se utiliza directamente dentro del atributo value de un elemento HTML, lo que sugiere que no se está realizando ningún tipo de sanitización o escape de caracteres especiales. Esto significa que si un usuario malintencionado proporciona datos maliciosos a través del parámetro "query", estos datos se renderizarán directamente en la página web sin ninguna restricción, lo que podría permitir la ejecución de scripts no deseados en el navegador del usuario.

The screenshot shows the Snyk security tool interface. At the top, it displays a red warning icon followed by the text "Cross-site Scripting (XSS)". Below this, it says "SNYK-CODE | CWE-79 | XSS". On the right, there are two buttons: "Data Flow" (highlighted in orange) and "Fix Analysis". The main content area contains a message: "Unsanitized input from an HTTP parameter flows into print, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS)." Below this message, it says "Found in: src/main/webapp/Views/Proyecto/searchBar.jsp (line : 29)". Under the heading "Data Flow", there is a code snippet from the JSP file:

```
src/main/webapp/Views/Proyecto/searchBar.jsp
29:31 <input type="text" value="<%= request.getParameter("query")%>">
29:31 <input type="text" value="<%= request.getParameter("query")%>">
29:31 <input type="text" value="<%= request.getParameter("query")%>">
```

Annotations show the flow of data from the "request.getParameter("query")" source (labeled 0) through the "value" attribute (labeled 1) to the "value" attribute (labeled 2).

El problema identificado es el uso de una constante codificada de forma rígida y relevante para la seguridad en el código fuente. En particular, se ha encontrado una clave secreta codificada directamente en el archivo LoginController.java en la línea 23. Este tipo de práctica es peligrosa ya que expone secretos de seguridad directamente en el código fuente, lo que facilita su acceso no autorizado si el código se compromete. La solución para este problema es evitar el uso de valores codificados rígidamente para secretos en el código fuente. En lugar de eso, se deben utilizar mecanismos seguros para gestionar y almacenar secretos, como variables de entorno, servicios de gestión de secretos, o archivos de configuración externos protegidos. Estos métodos permiten mantener los secretos separados del código fuente, lo que reduce el riesgo de exposición en caso de una brecha de seguridad.

H Use of Hardcoded, Security-relevant Constants

SNYK-CODE | CWE-547 | NonCryptoHardcodedSecret

Avoid hardcoding values that are meant to be secret. Found hardcoded secret.
Found in: [src/main/java/com/loscuchurrumines/Controller/LoginController.java](#) (line : 23)

Data Flow

```
src/main/java/com/loscuchurrumines/Controller/LoginController.java
```

```
23:46 private static final String SECRET_KEY = "6LfVXgspAAAAFOTm5Pq1XZqbTYibfKzGND1UOCx";
```

SOURCE SINK 0

El problema identificado es el uso de credenciales codificadas de forma rígida en el código fuente. En particular, se ha encontrado una contraseña codificada directamente en el archivo NeonConnection.java en la línea 10. Este tipo de práctica es peligrosa ya que expone credenciales de acceso directamente en el código fuente, lo que facilita su acceso no autorizado si el código se compromete. La solución para este problema es evitar el uso de contraseñas codificadas rígidamente en el código fuente. En lugar de eso, se deben utilizar mecanismos seguros para gestionar y almacenar credenciales, como variables de entorno, servicios de gestión de secretos, o archivos de configuración externos protegidos. Estos métodos permiten mantener las credenciales separadas del código fuente, lo que reduce el riesgo de exposición en caso de una brecha de seguridad.

M Use of Hardcoded Credentials

SNYK-CODE | CWE-798,CWE-259 | HardcodedPassword

Do not hardcode passwords in code. Found hardcoded password used in [here](#).
Found in: [src/main/java/com/loscuchurrumines/Config/NeonConnection.java](#) (line : 10)

Data Flow

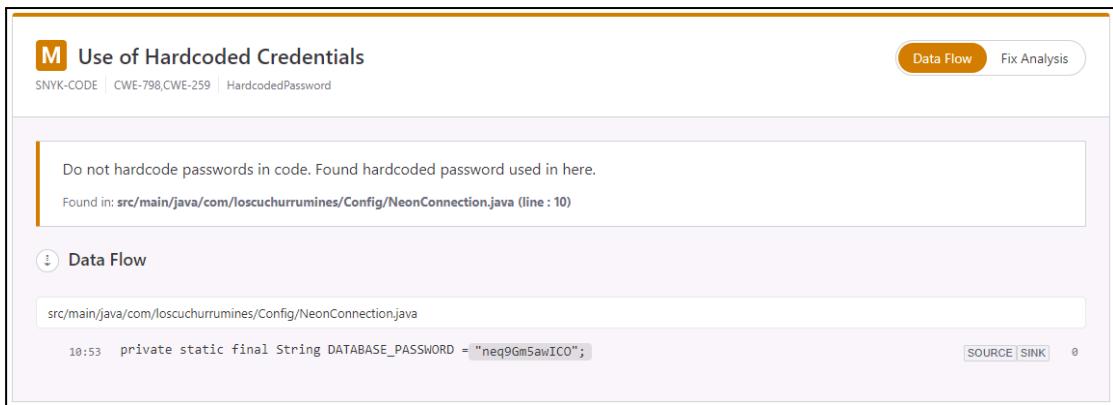
```
src/main/java/com/loscuchurrumines/Config/NeonConnection.java
```

```
10:53 private static final String DATABASE_PASSWORD = "neq9Gm5awICO";
```

SOURCE SINK 0

- Implementación de Mejoras:

El informe de Snyk resalta un problema crítico en el desarrollo de software: el almacenamiento de contraseñas en el código fuente, lo que representa un riesgo de seguridad significativo. Se encuentra una contraseña hardcoded en un archivo Java, lo que podría permitir el acceso no autorizado a la base de datos.



El archivo NeonConnection.java revela información sensible en su código fuente, incluyendo la URL de la base de datos, el usuario y la contraseña. Aunque la URL y el usuario no son intrínsecamente inseguros, la contraseña se encuentra hardcoded en texto plano, lo que representa una vulnerabilidad crítica.

The screenshot shows a Java code editor with the file "NeonConnection.java" open. The code defines a class "NeonConnection" with a static block that sets the database URL, user, and password. The password "neq9Gm5awICO" is highlighted in red, indicating it is a hardcoded value. The code also includes a try-catch block for the JDBC driver loading.

```

    package com.loscuchurrumines.config;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.SQLException;

    public class NeonConnection {
        private static final String DATABASE_URL = "jdbc:postgresql://ep-small-haze-880087949.us-east-2.aws.neon.tech/dbSustainPartners";
        private static final String DATABASE_USER = "ximena-ortiz";
        private static final String DATABASE_PASSWORD = "neq9Gm5awICO";

        static {
            try {
                Class.forName("org.postgresql.Driver");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }

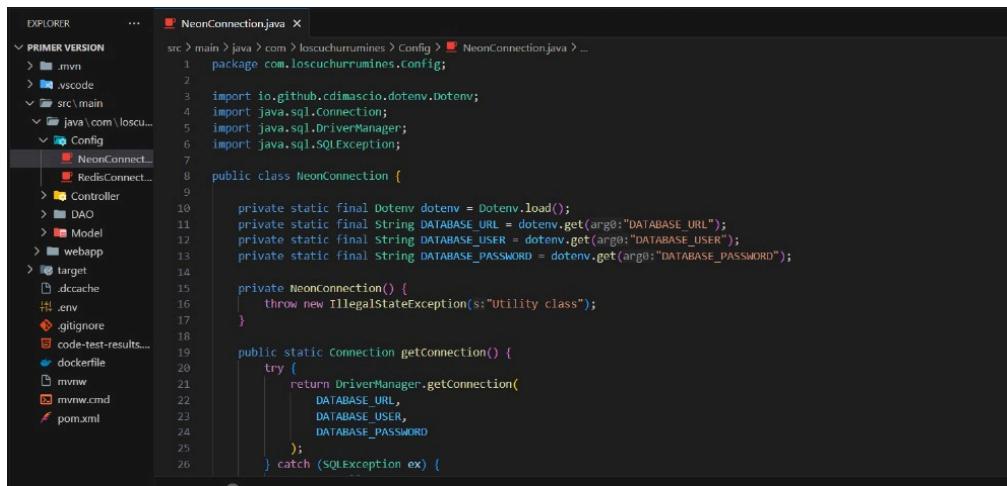
        public static Connection getConnection() {
            try {
                return DriverManager.getConnection(DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            return null;
        }
    }
  
```

La clase ha sido modificada para utilizar una biblioteca externa, io.github.cdimascio.dotenv, que es un gestor de variables de entorno. Esto permite que la aplicación cargue configuraciones sensibles como URL de base de datos, usuario y contraseña desde un archivo .env que no se debería incluir en el control de versiones, en lugar de tenerlas codificadas en el código.

Las líneas clave son:

Línea 11: Se crea una instancia de Dotenv para manejar la carga de las variables de entorno.

Línea 13-15: Las constantes DATABASE_URL, DATABASE_USER, y DATABASE_PASSWORD ya no tienen valores codificados, sino que se asignan dinámicamente en tiempo de ejecución utilizando el método dotenv.get("KEY"), donde KEY correspondería a las claves definidas en el archivo .env.



```
NeonConnection.java
...
src > main > java > com > loscuchurumines > Config > NeonConnection.java > ...
1 package com.loscuchurumines.Config;
2
3 import io.github.cdimascio.dotenv.Dotenv;
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.sql.SQLException;
7
8 public class NeonConnection {
9
10     private static final Dotenv dotenv = Dotenv.load();
11     private static final String DATABASE_URL = dotenv.get("DATABASE_URL");
12     private static final String DATABASE_USER = dotenv.get("DATABASE_USER");
13     private static final String DATABASE_PASSWORD = dotenv.get("DATABASE_PASSWORD");
14
15     private NeonConnection() {
16         throw new IllegalStateException("Utility class");
17     }
18
19     public static Connection getConnection() {
20         try {
21             return DriverManager.getConnection(
22                 DATABASE_URL,
23                 DATABASE_USER,
24                 DATABASE_PASSWORD
25             );
26         } catch (SQLException ex) {
27             throw new RuntimeException(ex);
28         }
29     }
30 }
```

El archivo .env, es utilizado para definir variables de entorno con configuraciones sensibles y secretos necesarios para la aplicación.

Estas variables se referencian en el código de la aplicación mediante la biblioteca io.github.cdimascio.dotenv, permitiendo que el código acceda a la información de manera segura sin estar codificada directamente en el código fuente.

El uso de un archivo .env facilita la gestión de configuraciones para diferentes entornos y mejora la seguridad al no incluir en el repositorio de código. Además, solo está presente en el entorno de ejecución con los permisos adecuados para prevenir accesos no autorizados.



```
.env
...
DATABASE_URL=jdbc:postgresql://ep-small-haze-88087949.us-east-2.awsn.neon.tech/dbSustainPartners
DATABASE_USER=ximena-ortiz
DATABASE_PASSWORD=neq9Gm5awICO
RECAPTCH_PRIVATE_KEY=6LfVXgspAAAAFOIm5PqiXZqbtYibfKzGNdIuOCx
```

Snyk identifica un problema alto que describe cómo la entrada no desinfectada de un parámetro HTTP fluye hacia la impresión, advirtiéndonos sobre un posible ataque de secuencias de comandos entre sitios (XSS).

The screenshot shows the Snyk interface with the following details:

- Title:** H Cross-site Scripting (XSS)
- Category:** SNYK-CODE | CWE-79 | XSS
- Description:** Unsanitized input from an HTTP parameter flows into print, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).
- Location:** Found in: src/main/webapp/Views/Proyecto/formularioParticipar.jsp (line : 23)
- Data Flow:**
 - Line 23:41 shows the source code: <form action="/participar?id=<%= request.getParameter("id")%>" method="post" class="bg-white p-6 rounded-lg shadow-lg space-y-4">
 - Line 23:41 shows the sink code: <form action="/participar?id=<%= request.getParameter("id")%>" method="post" class="bg-white p-6 rounded-lg shadow-lg space-y-4">
 - Line 23:41 shows the flow from the source to the sink.

En la línea 23 observamos el error donde la acción del formulario está configurada para enviar los datos a la URL "/participar", con el valor del parámetro "id" obtenido dinámicamente del request actual.

The screenshot shows the Java code editor with the following details:

- File:** formularioParticipar.jsp
- Code:**

```

5   <head>
6     <script>
7       function toggleDonationInput(selectedValue) {
8         if (donationInput.disabled) {
9           donationInput.value = selectedValue;
10        } else {
11          donationInput.value = '';
12        }
13      }
14    </script>
15  </head>
16  <body class="bg-gray-100 font-[\'Newsreader\']">
17    <div class="container mx-auto px-10 py-6">
18      <button class="bg-green-500 text-white px-8 py-2 rounded-full" onclick="location.href='dashboard'">Volver al Inicio</button><br>
19      <form action="/participar?id=<%= request.getParameter("id")%>" method="post" class="bg-white p-6 rounded-lg shadow-lg space-y-4">
20        <div class="form-group">
21          <span class="block text-gray-700 font-semibold">¿Cómo desea contribuir?</span> <br>
22          <%>
23          List<Integer> modalidades = (List<Integer>) request.getSession().getAttribute("modalidades");
24          for(Integer modalidad : modalidades) {
25            if(modalidad == 1) {
26              <%>
27              <label class="inline-flex items-center mt-3">
28                <input type="radio" id="modalidad1" name="modalidad" value="1" class="form-radio h-5 w-5 text-green-600" onchange="toggleDonatio
29              </label>
30            <%>
31            } else if(modalidad == 2) {
32              <%>
33              <label class="inline-flex items-center mt-3">
34                <input type="radio" id="modalidad2" name="modalidad" value="2" class="form-radio h-5 w-5 text-green-600" onchange="toggleDonatio
35              </label>
36            <%>
37          </div>

```

Para eliminar este error cambiamos el código, en la línea 23 eliminamos y agregamos "<form action="/participar" method="post" class="bg-white p-6 rounded-1g shadow-lg-space-y-4">" cambia ya que la acción del formulario está establecida estáticamente en "/participar". Esto significa que cada vez que se envíe el formulario, independientemente de los parámetros adicionales o valores dinámicos, siempre se enviará a la misma URL, "/participar". En cambio, antes de modificar la línea incluye un parámetro dinámico, que es el valor de "id". Esto se logra utilizando una expresión JSP <%= request.getParameter("id")%>, que obtiene el valor del parámetro "id" del

request actual. Por lo tanto, la acción del formulario será diferente según el valor actual del parámetro "id" en la solicitud.

```

10   <script>
11     function toggleDonationInput(selectedValue) {
12       var donationInput = document.getElementById('monto');
13       donationInput.disabled = selectedValue != '1';
14       if (donationInput.disabled) {
15         donationInput.value = '';
16       }
17     }
18   </script>
19 </head>
20 <body class="bg-gray-100 font-[Newsreader]">
21   <div class="container mx-auto px-10 py-6">
22     <button class="bg-green-500 text-white px-8 py-2 rounded-full" onclick="location.href='dashboard'">Volver al Inicio</button><br>
23     <form action="/participar" method="post" class="bg-white p-6 rounded-lg shadow-lg space-y-4" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">
24       <div class="form-group">
25         <span class="block text-gray-700 font-semibold">¿Cómo desea contribuir?</span> <br>
26         <%>
27         <list<Integer> modalidades = (List<Integer>) request.getSession().getAttribute("modalidades");
28         for(Integer modalidad : modalidades) {
29           if(modalidad == 1) {
30             <%>
31             <label class="inline-flex items-center mt-3">
32               <input type="radio" id="modalidad1" name="modalidad" value="1" class="form-radio h-5 w-5 text-green-600" onchange="toggleDonationInput(this.value)">
33             </label>
34           <%>

```

Snyk identifica un problema alto de vulnerabilidad que describe cómo la entrada no desinfectada de un parámetro HTTP fluye hacia la impresión, advirtiéndonos sobre un posible ataque de secuencias de comandos entre sitios (XSS).

H Cross-site Scripting (XSS)

SNYK-CODE | CWE-79 | XSS

Unsanitized input from an HTTP parameter flows into print, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

Found in: [src/main/webapp/Views/Proyecto/searchBar.jsp](#) (line : 29)

[Data Flow](#)

En la línea 29 es donde se encuentra el problema “value=”request.getParameter(“query”)%>”. Esta vulnerabilidad ocurre cuando la entrada del usuario no se valida o se filtra adecuadamente y se inserta directamente en el HTML sin escapar caracteres especiales.

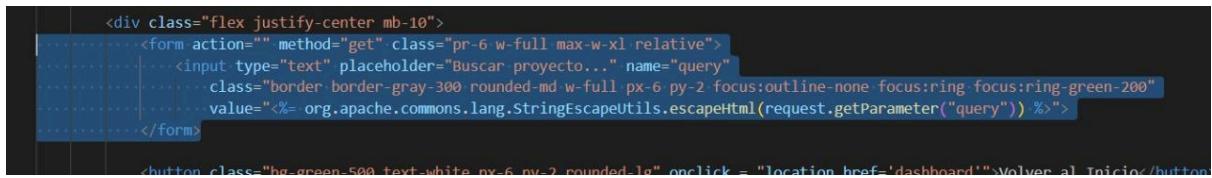
">. This line contains an input element with a placeholder attribute and a value attribute that includes a script tag."/>

```

10 <head>
11   <meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no" />
12   <style>
13     .newsreader-font {
14       font-family: 'Newsreader', sans-serif;
15     }
16   </style>
17 <body class="bg-gray-100 font-[Newsreader]">
18   <div class="text-center mb-10 relative pt-6" style="position: relative; height: 40px; width: 100%;">
19     <!--<div class="text-lime-800 font-medium newsreader-font leading-none text-5xl">Sustain Partners</div>-->
20
21     <!--<div class="h-1 bg-gray-900 w-1/3 mx-auto my-2" style="position: absolute; top: -10px; left: 50%; transform: rotate(-45deg); width: 0; height: 0; border-left: 15px solid transparent; border-right: 15px solid transparent; border-bottom: 30px solid black; z-index: 1;"></div>-->
22
23     <div class="container mx-auto px-10 py-6" style="position: relative; height: 40px; width: 100%;">
24       <!-- Search Bar -->
25       <div class="flex justify-center mb-10" style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-color: white; border-radius: 10px; padding: 10px; z-index: 2;">
26         <form action="" method="get" class="pr-6 w-full max-w-xl relative">
27           <input type="text" placeholder="Buscar proyecto..." name="query" value="<%request.getParameter("query")%>" style="border: 1px solid #ccc; border-radius: 10px; width: 100%; height: 100%; font-size: inherit; font-weight: inherit; color: inherit; background-color: inherit; outline: none; focus: none; transition: all 0.3s ease; z-index: 1;">
28         </form>
29       </div>
30     </div>

```

Esta modificación de la línea de código escapa los caracteres especiales en el valor del parámetro "query" antes de usarlo en el atributo value del elemento HTML. De esta manera, se previene la ejecución de scripts maliciosos en el navegador del usuario.

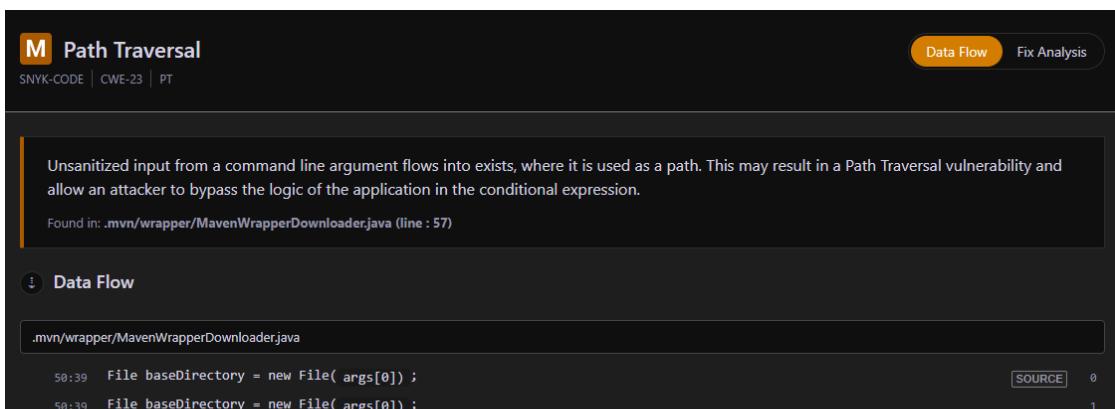


```
<div class="flex justify-center mb-10">
    <form action="" method="get" class="pr-6 w-full max-w-xl relative">
        <input type="text" placeholder="Buscar proyecto..." name="query"
            class="border border-gray-300 rounded-md w-full px-6 py-2 focus:outline-none focus:ring focus:ring-green-200"
            value="<% org.apache.commons.lang.StringEscapeUtils.escapeHtml(request.getParameter("query")) %>">
    </form>
<button class="bg-green-500 text-white px-6 py-2 rounded-lg" onclick="location.href='dashboard'>Volver al Inicio</button>
```

Luego de modificar el código, observamos que los problemas han disminuido.



Otro de los problemas de nivel medio que reporta snyk, es cuando esta vulnerabilidad ocurre cuando la aplicación utiliza una entrada no filtrada o no validada de un argumento de línea de comandos para construir una ruta de archivo. En este caso específico, el argumento de la línea de comandos args[0] se utiliza directamente para crear un objeto File, que luego se utiliza para comprobar la existencia de un archivo en esa ruta. El problema radica en que si un atacante proporciona una entrada maliciosa que contiene secuencias de caracteres especiales como .. en el argumento de la línea de comandos, podría navegar más allá del directorio previsto y acceder a archivos sensibles en el sistema de archivos del servidor.



A screenshot of the Snyk Path Traversal report for a file named `MavenWrapperDownloader.java`. The report highlights an issue where unsanitized input from a command line argument flows into `exists()`, which may result in a Path Traversal vulnerability. The report includes a 'Data Flow' section showing the flow of the `args[0]` argument into the `File` object creation, and a 'Fix Analysis' button.

El problema relacionado con el código anterior sobre el que hablamos antes es la vulnerabilidad de Path Traversal. Esta vulnerabilidad ocurre cuando la entrada del usuario no se valida adecuadamente y se utiliza para construir una ruta de archivo sin restricciones. Si un atacante puede controlar la entrada y proporcionar una ruta de archivo maliciosa que contiene secuencias de caracteres especiales como `..`, podría permitir que el atacante acceda a archivos sensibles fuera del directorio previsto. En este caso, el argumento `args[0]` se utiliza directamente para crear un objeto `File`, lo que podría ser vulnerable a un ataque, si el usuario proporciona una ruta maliciosa.

```

48     public static void main(String args[]) {
49         System.out.println("- Downloader started");
50         File baseDirectory = new File(args[0]);
51         System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());
52
53         // If the maven-wrapper.properties exists, read it and check if it contains a custom
54         // wrapperUrl parameter.
55         File mavenWrapperPropertyFile = new File(baseDirectory, MAVEN_WRAPPER_PROPERTIES_PATH);
56         String url = DEFAULT_DOWNLOAD_URL;
57         if(mavenWrapperPropertyFile.exists()) {
58             FileInputStream mavenWrapperPropertyFileInputStream = null;
59             try {
60                 mavenWrapperPropertyFileInputStream = new FileInputStream(mavenWrapperPropertyFile);

```

Al normalizar el directorio base recibido del usuario utilizando la función `FilenameUtils.normalize()` de Apache Commons IO. Esta normalización elimina cualquier secuencia de escape o caracteres de traversal de ruta potencialmente peligrosos. Luego, se construye la ruta del archivo `maven-wrapper.properties` utilizando este directorio base normalizado y se crea un objeto `File`. Posteriormente, se carga el archivo de propiedades y se obtiene la URL del wrapper de Maven. Si el archivo de propiedades no existe, se utiliza una URL predeterminada. Con estas modificaciones, se garantiza que la ruta del archivo se maneje de manera segura, mitigando así la posibilidad de un traversal de ruta y mejorando la seguridad del programa.

```

19     import java.util.Properties;
20     import org.apache.commons.io.FilenameUtils;
21

```

```

57     File mavenWrapperPropertyFile = new File(FilenameUtils.normalize(baseDirectory), MAVEN_WRAPPER_PROPERTIES_PATH);
58
59     String url = DEFAULT_DOWNLOAD_URL;
60     if(mavenWrapperPropertyFile.exists()) {
61         FileInputStream mavenWrapperPropertyFileInputStream = null;
62         try {
63             mavenWrapperPropertyFileInputStream = new FileInputStream(mavenWrapperPropertyFile);
64             Properties mavenWrapperProperties = new Properties();
65             mavenWrapperProperties.load(mavenWrapperPropertyFileInputStream);
66             url = mavenWrapperProperties.getProperty(PROPERTY_NAME_WRAPPER_URL, url);
67         } catch (IOException e) {
68             System.out.println("- ERROR loading '" + MAVEN_WRAPPER_PROPERTIES_PATH + "'");
69         } finally {
70             try {
71                 if(mavenWrapperPropertyFileInputStream != null) {
72                     mavenWrapperPropertyFileInputStream.close();
73                 }
74             } catch (IOException e) {
75                 // Ignore ...
76             }
77         }
78     }
79     System.out.println("- Downloading from: " + url);
80

```

Finalmente, luego de las modificaciones respectivas del código, realizamos el análisis de snyk nuevamente y observamos que ya no encuentran problemas en el código fuente.



- Documentación de Cambios y Aprendizajes:

Documentación de Cambios:

A. Refactorización de Código para Manejar Credenciales:

- Se eliminaron las credenciales codificadas de forma rígida de los archivos Java, reemplazándolas por variables de entorno, mejorando así la seguridad y cumpliendo con las mejores prácticas de gestión de configuraciones sensibles.
- Se implementó el uso del archivo .env para almacenar información sensible, que no se incluye en el control de versiones para evitar su exposición.

B. Corrección de Vulnerabilidades XSS:

- Se identificaron y corrigieron instancias de Cross-Site Scripting (XSS) en archivos JSP, sanitizando las entradas de usuario para prevenir ataques.
- Se proporcionaron ejemplos concretos de cómo se corrigieron estas vulnerabilidades en formularioParticipar.jsp y searchBar.jsp.

C. Resultados del Escaneo de Seguridad:

- Se documentaron los resultados del análisis de seguridad, identificando y priorizando la corrección de vulnerabilidades de alta severidad.

Aprendizajes Clave:

A. Importancia de la Seguridad en Credenciales:

- Se aprendió la importancia de no hardcodear credenciales y cómo la exposición de estas puede comprometer toda la aplicación.

B. Comprensión de Ataques XSS:

- Se adquirió un entendimiento más profundo sobre cómo los ataques XSS pueden comprometer la seguridad de una aplicación web y se aprendió a implementar métodos de saneamiento de entradas para mitigar estos riesgos.

C. Implementación de Archivos .env:

- Se destacó la efectividad de utilizar archivos .env para gestionar configuraciones y se aprendió la mejor manera de integrarlos sin comprometer la seguridad.

D. Estrategias de Prevención de XSS:

- Se identificaron las estrategias de prevención contra XSS como el escape de caracteres especiales y la implementación de políticas de seguridad de contenido (CSP).

E. Reflexión sobre el Manejo de Vulnerabilidades:

- Se reflexionó sobre la necesidad de una revisión y manejo continuos de vulnerabilidades, especialmente en proyectos con múltiples dependencias y componentes interactivos.

F. Cultura de Seguridad de Código:

- Se reconoció la importancia de desarrollar una cultura de seguridad de código dentro del equipo de desarrollo, fomentando la educación continua y la actualización de habilidades relacionadas con la seguridad informática.

8. Cronograma

Se trabajó con un cronograma en github.



Link: <https://github.com/orgs/UPT-FAING-EPIS/projects/37/views/4>



8.1. Recursos

- **Humano:** Los integrantes del equipo que se encargarán de realizar los análisis tanto en Sonarqube como Snyk y refactorizar el código según los resultados de las herramientas.
- **Tecnológicos:** El servicio de Sonarqube dockerizado y Snyk para el análisis del proyecto, sistema de control de versiones a utilizarse será github para versionar las refactorizaciones en los diversos módulos en base a los análisis que realicen el sonarqube.