**Escaneo de vulnerabilidades computacionales**

**Medical AID**

***Integrantes:***

*Raimundo Estévez*

*Marcel Brard*

*Soledad Inostroza*

Índice de Contenidos

Tabla de contenido

[1. **Objetivo General** 3](#_Toc184310892)

[2. **Alcance** 3](#_Toc184310893)

[3. **Definiciones utilizadas** 3](#_Toc184310894)

[4. **Detalle de descarga e instalación de Nessus** 4](#_Toc184310895)

[5. **Resultado de aplicación de Nessus** 11](#_Toc184310896)

[6. **Detalle de descarga e instalación de SonarQube** 14](#_Toc184310897)

[**Resultado de aplicación de SonarQube de Frontend** 16](#_Toc184310898)

[**Resultado de aplicación de SonarQube de Backend** 18](#_Toc184310899)

[**Conclusión** 21](#_Toc184310900)

# **Objetivo General**

Realizar pruebas con las herramientas **SonarQube** y **Nessus** para lograr un análisis detallado de las vulnerabilidades de seguridad tanto en el código como en la infraestructura del sistema. Este enfoque busca garantizar un estándar más alto de calidad y seguridad en el funcionamiento de **Medical Aid**, fortaleciendo su confiabilidad y minimizando riesgos potenciales.

**SonarQube** es una herramienta de análisis de código estático que permite identificar vulnerabilidades, problemas técnicos y problemas de calidad en el desarrollo, ayudando a mantener estándares altos en el ciclo de vida del software. Por su parte, **Nessus** es una plataforma de análisis de vulnerabilidades enfocada en evaluar la seguridad de la infraestructura, identificando configuraciones inseguras, servicios expuestos y riesgos de seguridad.

# **Alcance**

Se generan casos de uso y escenarios más relevantes.

* La aplicación funciona correctamente en los navegadores más populares.
* Se aprueban las pruebas de integración con un 80% de aprobación

# **Definiciones utilizadas**

**Nessus:** Herramienta de escaneo de vulnerabilidades ampliamente reconocida y potente. Desarrollada por la empresa Tenable, se utiliza para evaluar la seguridad de sistemas informáticos, redes y aplicaciones. Nessus es una solución líder en su categoría y se utiliza en el campo de la seguridad informática para identificar y analizar posibles vulnerabilidades que podrían ser explotadas por ciberatacantes.

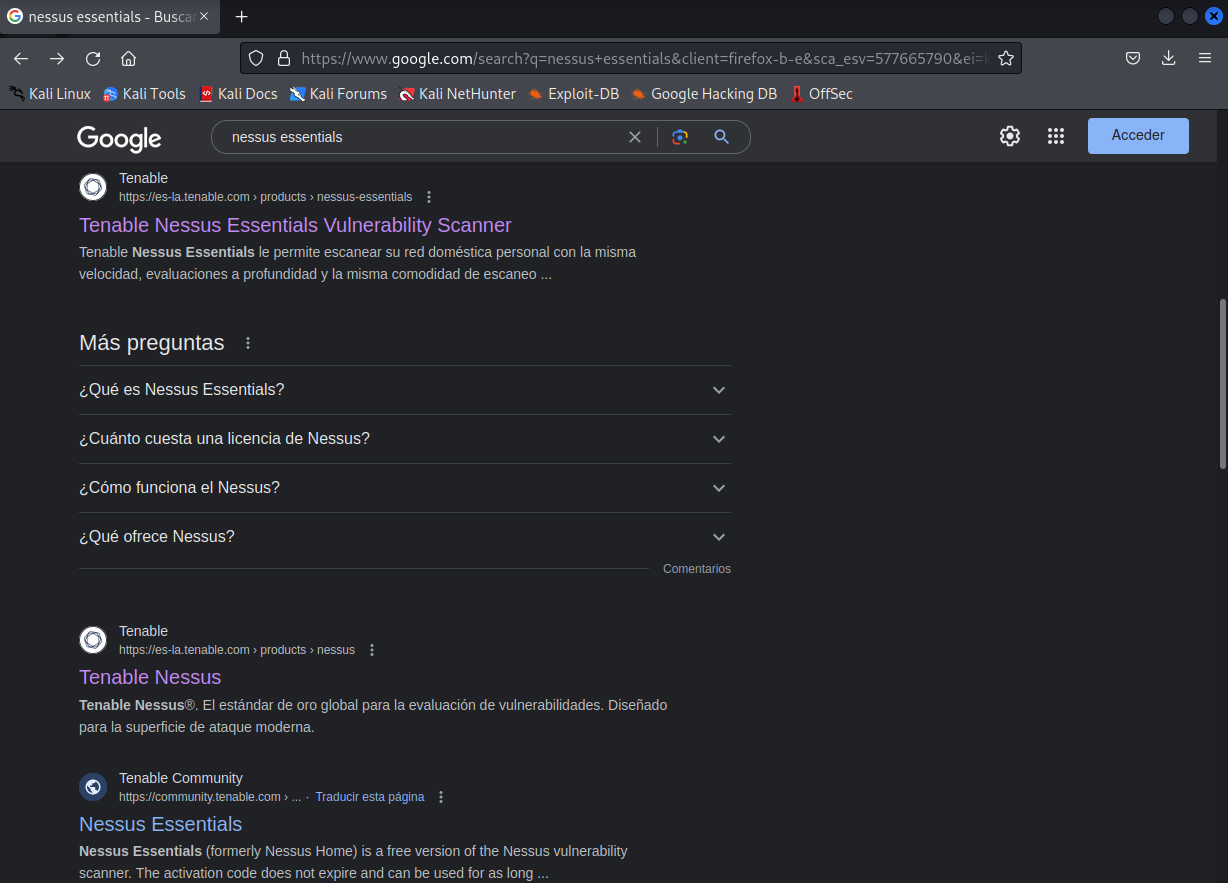
**SonaQube:**

SonarQube es una plataforma de análisis de código estático de código abierto diseñada para inspeccionar la calidad y la seguridad del código en proyectos de software. Proporciona informes detallados sobre posibles vulnerabilidades, errores, problemas de calidad y deuda técnica en el código, ayudando a los equipos de desarrollo a mejorar su calidad y reducir riesgos.

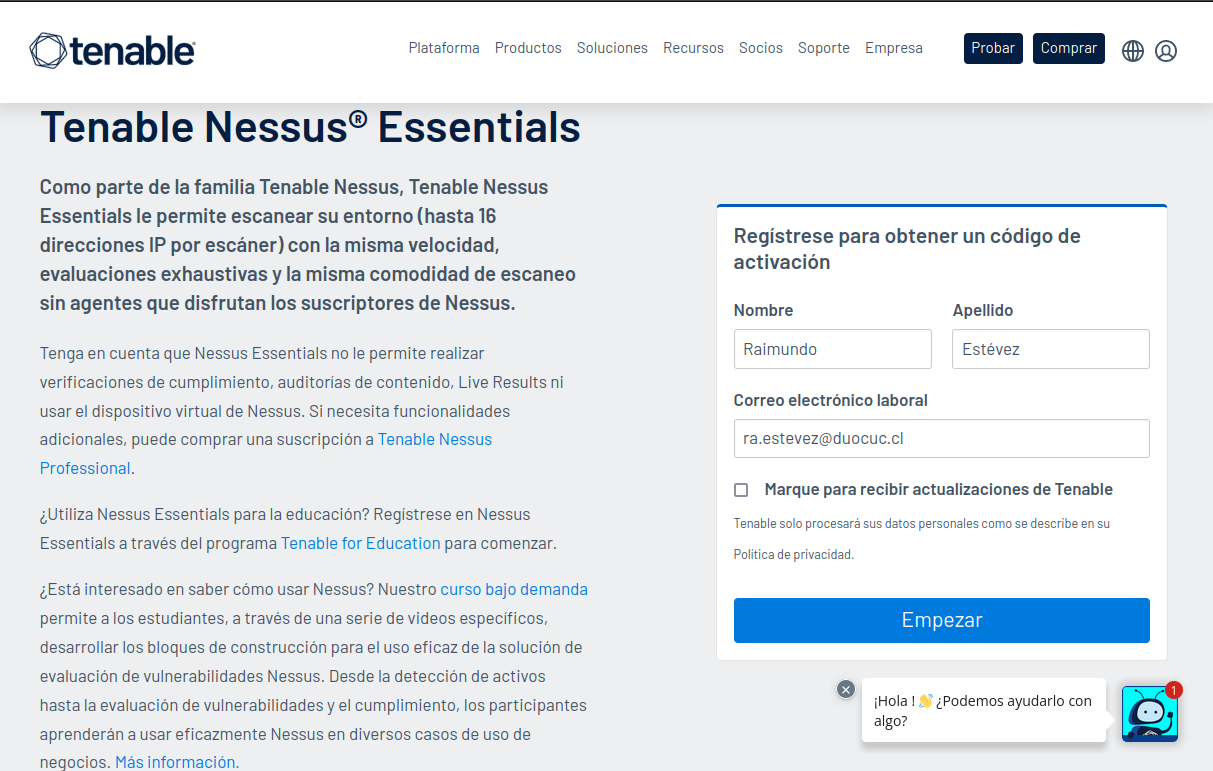
# **Detalle de descarga e instalación de Nessus**

Los pasos realizados para descargar e instalar el software de Nessus fueron:

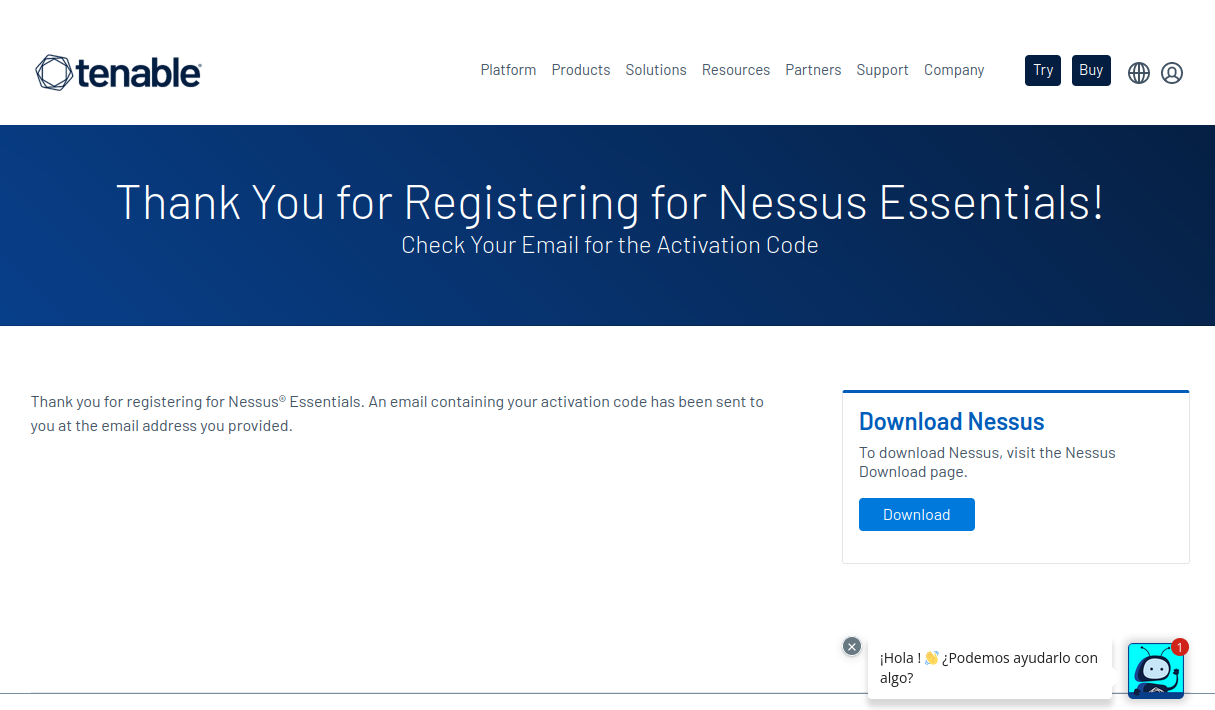
1. Buscar en google la página de descarga de Nessus Essentials e ingresar al link.



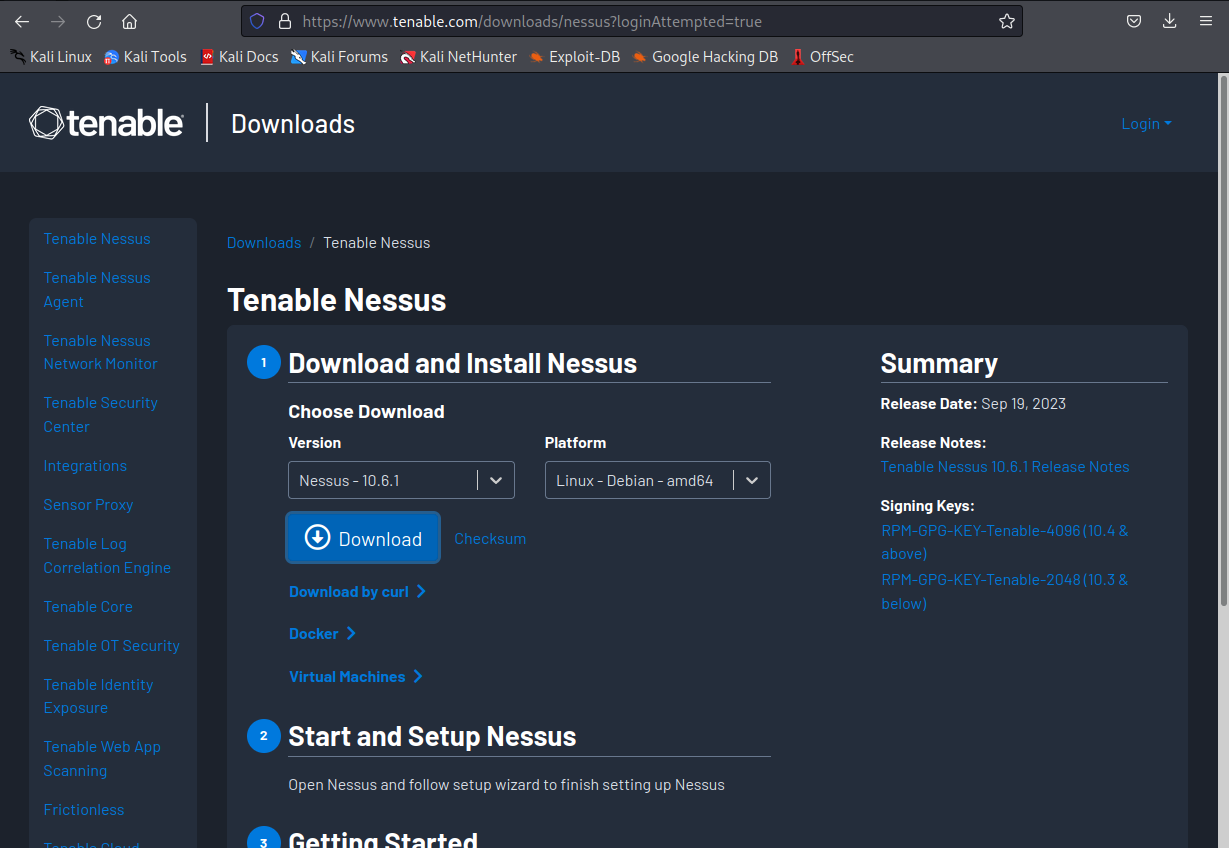
1. Ingresar datos de contacto para recibir un código de activación.



1. Dirigirse al enlace de descargas al clickear el botón “Download”.



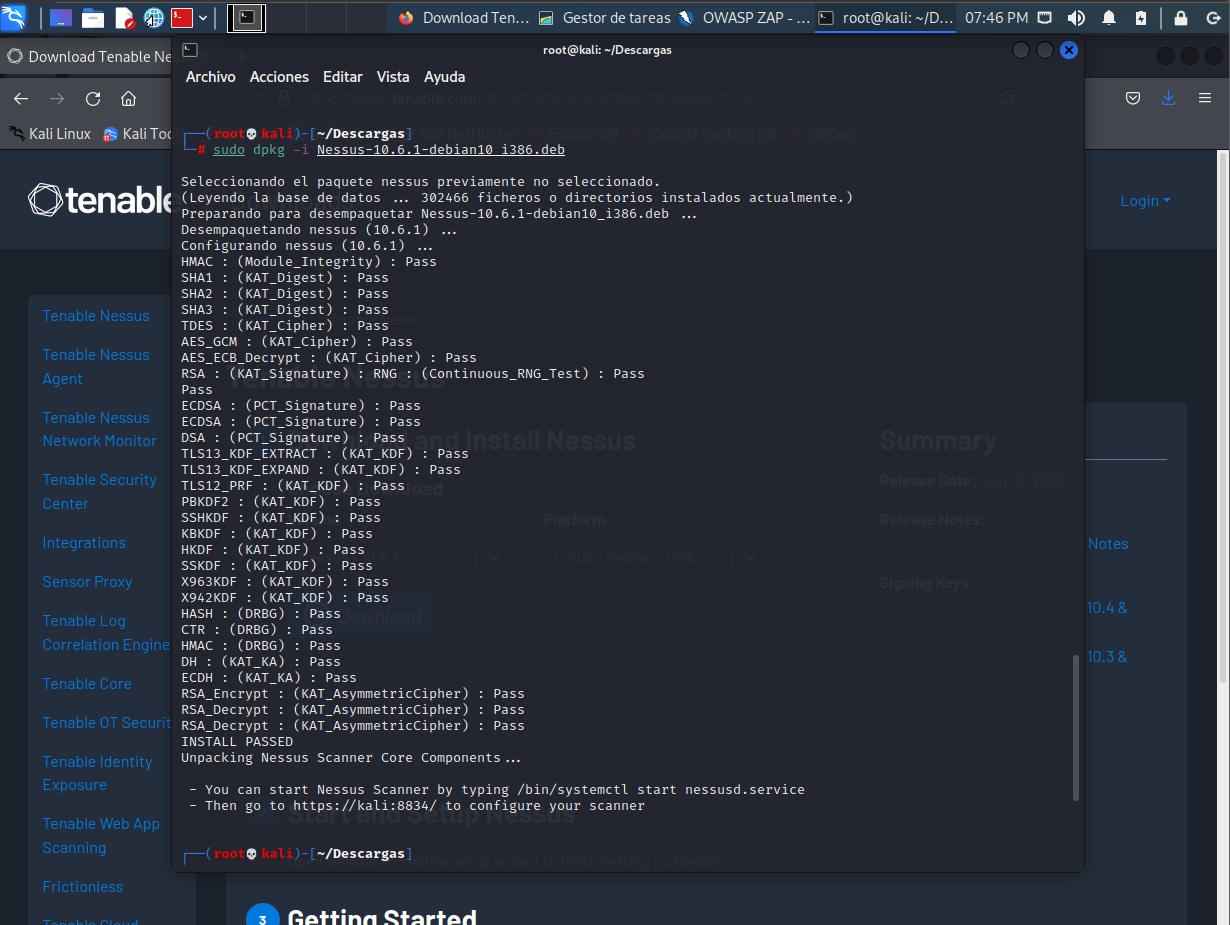
1. Seleccionar el instalador Linux Debian amd64 y clickear el botón “Download”.



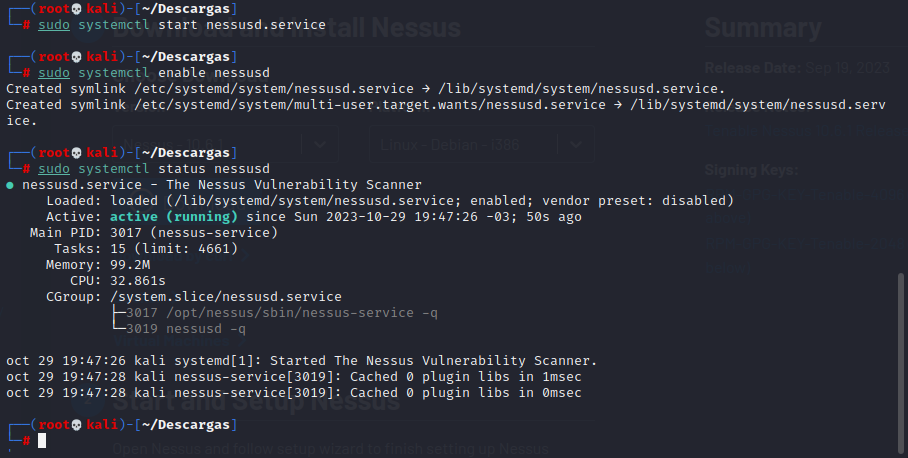
1. Aceptar los términos y condiciones clickeando el botón “Agree”.



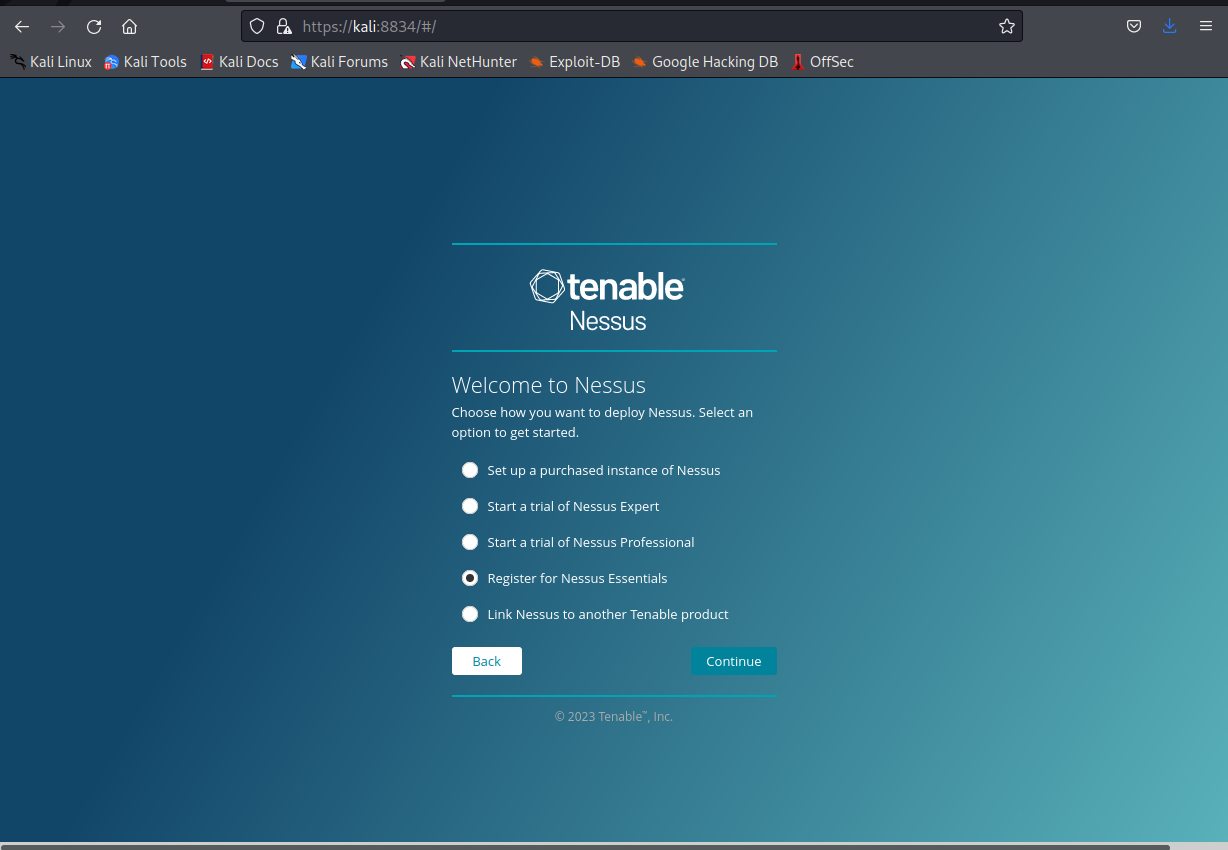
1. Instalar el archivo descargado a través de la terminal de líneas de comandos.



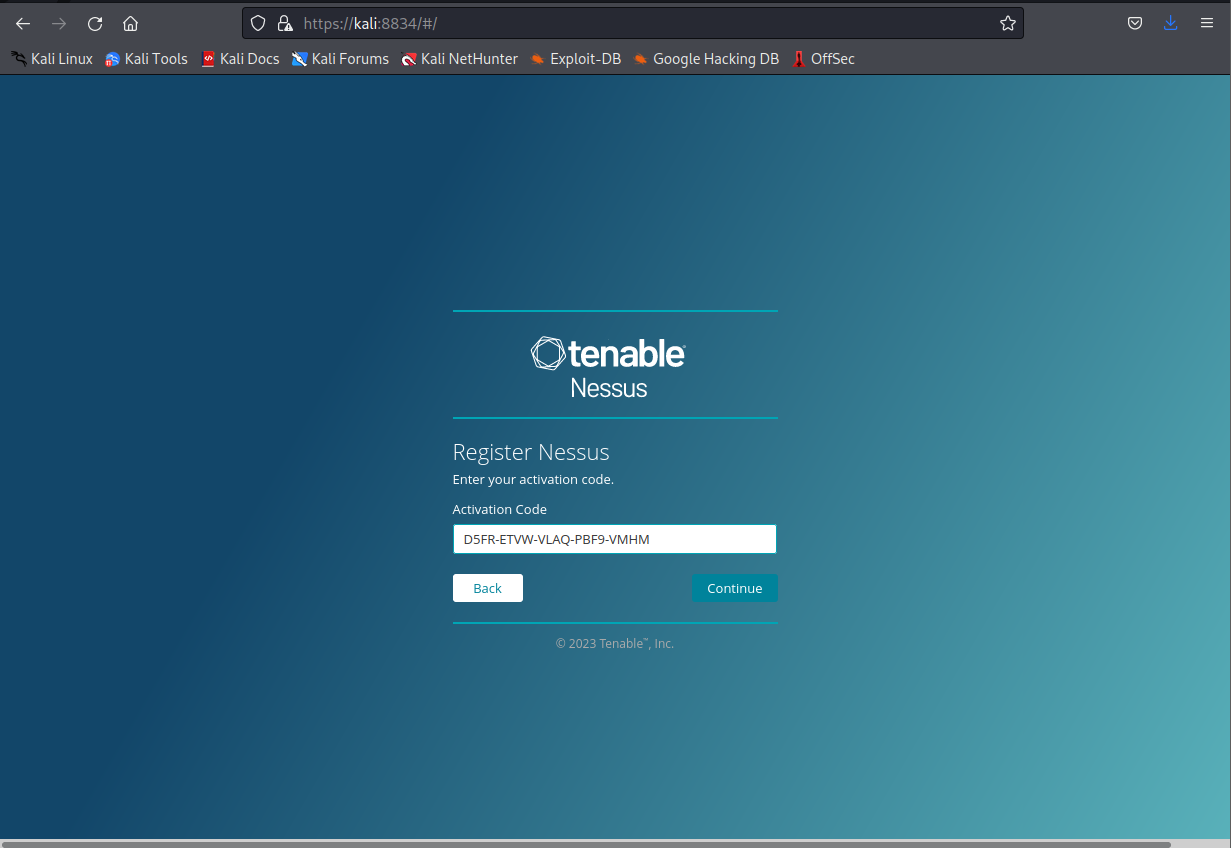
1. Iniciar los servicios de Nessus, como extra también se habilita el inicio automático al iniciar sesión.



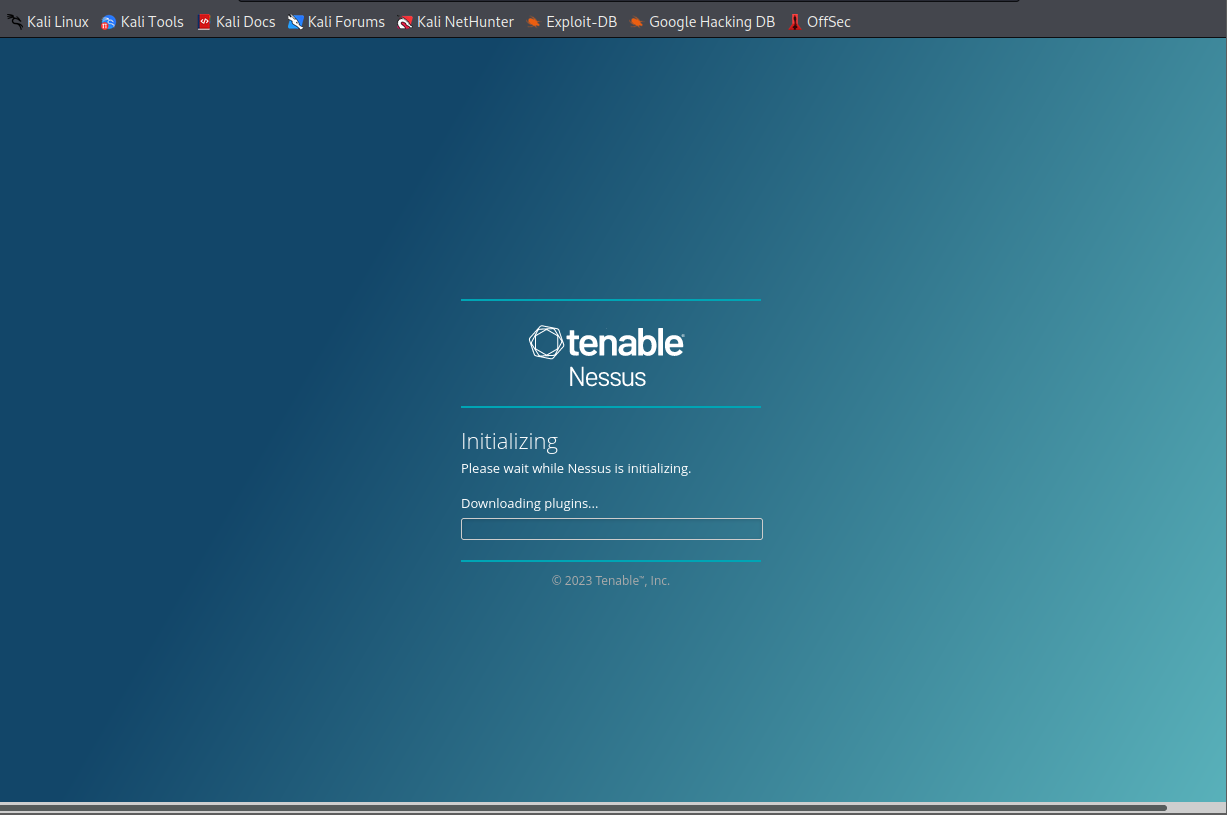
1. Ejecutar nessus ingresando al link <https://kali:8834/> y seleccionar el registro de Nessus Essentials



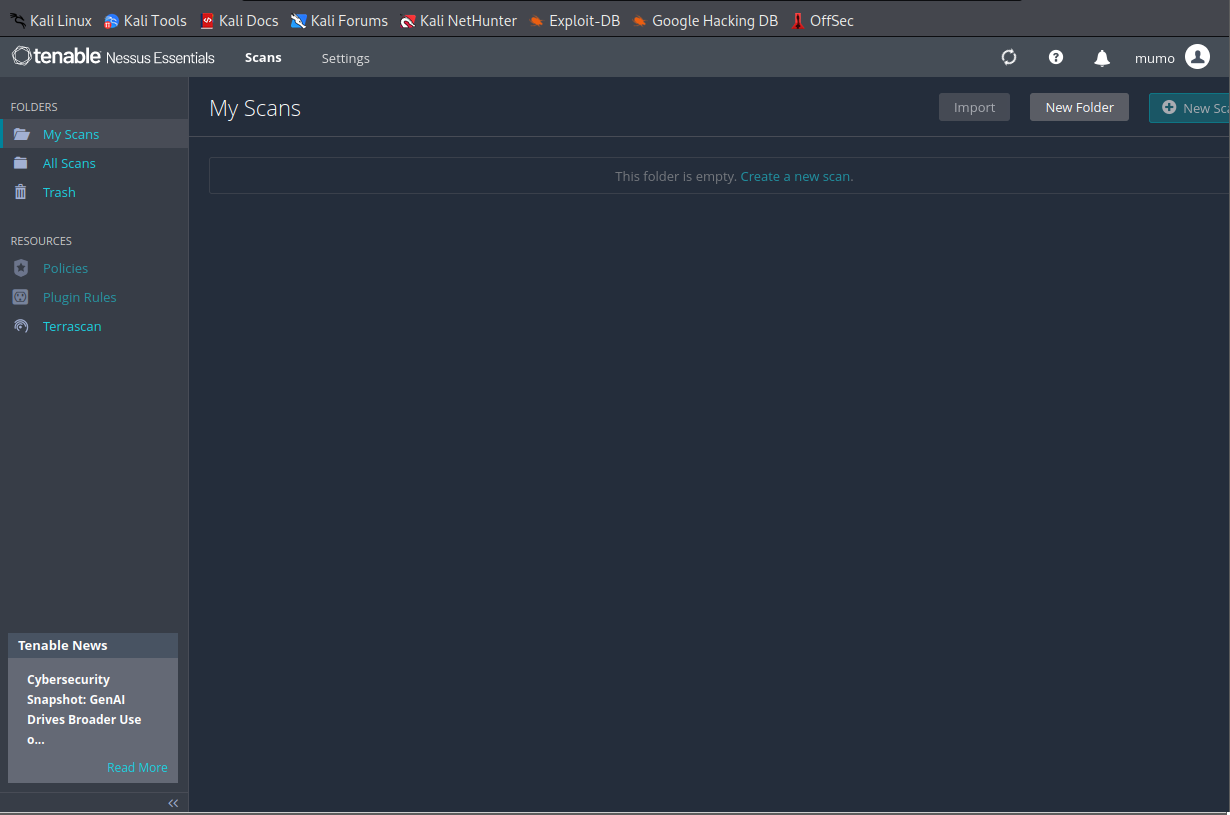
1. Ingresar el código de activación previamente solicitado.



1. Iniciar sesión y esperar la descarga de plugins.



1. La instalación está lista y se puede ejecutar Nessus.



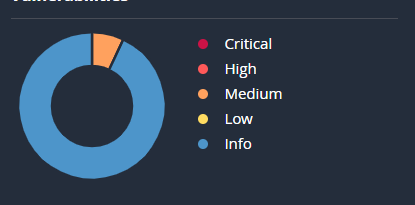
# **Resultado de aplicación de Nessus**

Se realizó el escaneo de la ip que entregaba el servidor que realiza el hosting de Medical AID (190.21.57.14)

Tras el término del escaneo se encontraron 20 vulnerabilidades, de los cuáles se encontraron:

* 1 vulnerabilidades de nivel medio.
* 19 vulnerabilidades informativas.





**Vulnerabilidades con severidad MEDIUM**

1. **OpenSSL (Multiple Issues)**

**Hallazgos:**

Dentro de esta vulnerabilidad se encuentran un total de 2 vulnerabilidades de nuestro servidor y su certificado SSL. De estas vulnerabilidades, tienen un puntaje CVSS de medio.

**Descripción:**   
La versión de OpenSSL instalada en el host remoto es anterior a la 3.1.8. Está afectada por una vulnerabilidad referenciada en el aviso 3.1.8.

**Resumen del problema:** El uso de las APIs de curvas elípticas de bajo nivel GF(2^m) con valores explícitos no confiables para el polinomio del campo puede provocar lecturas o escrituras fuera de los límites de la memoria.

**Resumen del impacto:** Las escrituras fuera de los límites pueden llevar a un fallo de la aplicación o incluso a la ejecución remota de código. Sin embargo, en los protocolos que involucran criptografía de curvas elípticas, los casos problemáticos son poco frecuentes, ya que las curvas nombradas son generalmente las soportadas, o las curvas explícitas siguen una codificación X9.62 de curvas binarias que no pueden representar valores problemáticos.

**Soluciones:**

* Upgrade to OpenSSL version 3.1.8 or later.

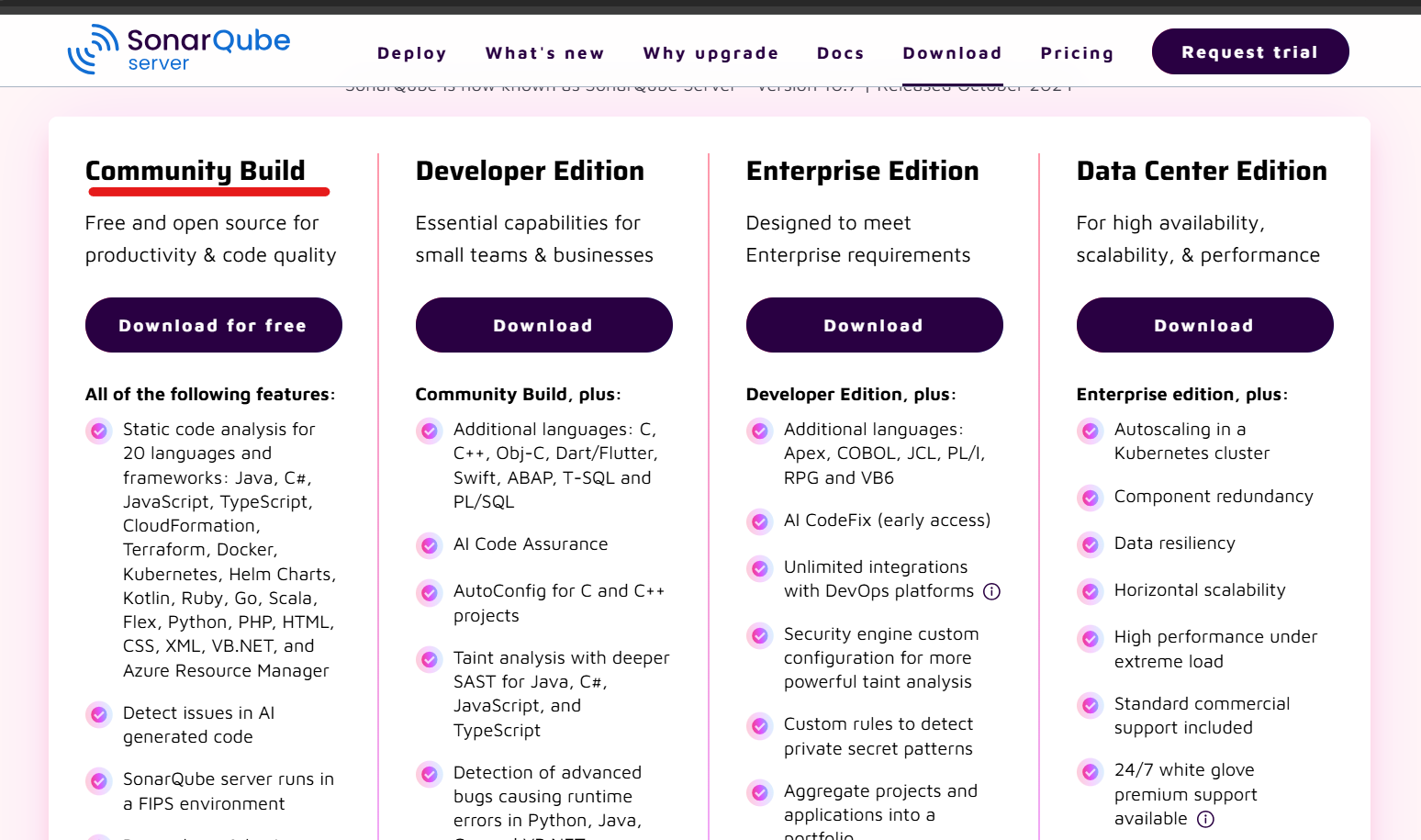
**Vulnerabilidades con severidad INFORMATIVE**

1. **HTTP (Múltiples problemas)**
   * Descripción: Vulnerabilidades detectadas en la implementación HTTP del servidor web. Pueden incluir errores de configuración o problemas en la gestión de peticiones.
   * Soluciones: Verifica y actualiza la configuración del servidor HTTP. Asegúrate de que esté ejecutando la última versión y aplica parches de seguridad necesarios.
2. **Servidor Web (Múltiples problemas)**
   * Descripción: Se detectaron varias vulnerabilidades relacionadas con el servidor web, como exposiciones innecesarias o configuraciones inseguras.
   * Soluciones: Reduce la exposición del servidor revisando las configuraciones y desactivando módulos innecesarios. Aplica actualizaciones regulares.
3. **HTTP (CGI Abuses)**
   * Descripción: Riesgos asociados a scripts CGI inseguros que podrían ser explotados para comprometer el sistema.
   * Soluciones: Asegúrate de que todos los scripts CGI sean confiables y actualizados. Considera deshabilitar la funcionalidad CGI si no es necesaria.
4. **Nessus SYN Scanner**
   * Descripción: Herramienta utilizada para escanear puertos, detectando puertos abiertos y posibles servicios expuestos.
   * Soluciones: Bloquea puertos innecesarios y utiliza listas de control de acceso (ACL) para restringir el tráfico.
5. **Versión de Apache HTTP Server**
   * Descripción: Detectada la versión del servidor Apache. Si es una versión desactualizada, puede tener vulnerabilidades conocidas.
   * Soluciones: Actualiza Apache HTTP Server a la versión más reciente y aplica parches de seguridad.
6. **Información del escaneo de Nessus**
   * Descripción: Datos informativos sobre la ejecución del escaneo Nessus en la máquina objetivo.
   * Soluciones: No se requiere acción, pero revisa el informe para identificar áreas **de mejora en la configuración del servidor.**
7. **Informe de parches**
   * Descripción: Muestra parches aplicados y pendientes para mejorar la seguridad del sistema.
   * Soluciones: Aplica los parches recomendados para mitigar vulnerabilidades detectadas.
8. **Mapa del sitio de la aplicación web**
   * Descripción: Información sobre las rutas y páginas expuestas por la aplicación web.
   * Soluciones: Minimiza las rutas expuestas y verifica que no haya información confidencial accesible públicamente.

# **Detalle de descarga e instalación de SonarQube**

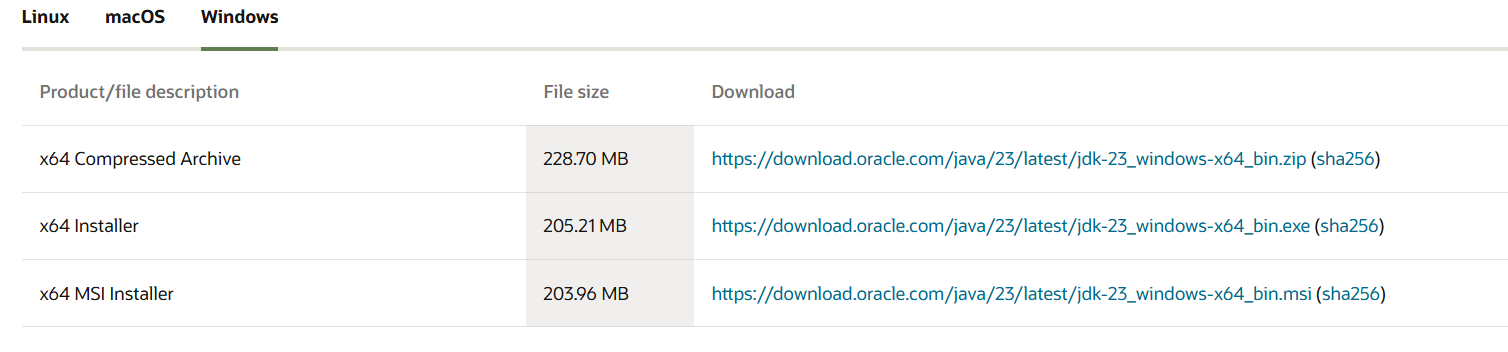
**1. Descarga SonarQube**

1. Ve al sitio oficial de SonarQube.
2. Descarga la versión más reciente de **SonarQube Community Edition** (o cualquier otra versión según tus necesidades).



**2. Instala Java JDK**

1. Asegúrate de que tengas instalado **Java JDK 11 o superior**. Si no lo tienes:
   * Descárgalo desde [Oracle](https://www.oracle.com/java/technologies/javase-downloads.html) o [OpenJDK](https://openjdk.org/).
   * Instala y configura la variable de entorno JAVA\_HOME apuntando al directorio de instalación del JDK.

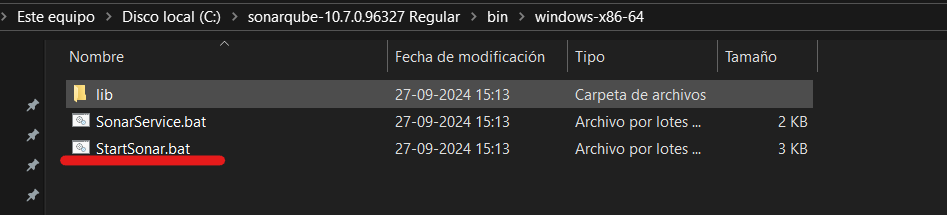


**3. Extrae SonarQube**

1. Extrae el archivo zip de **SonarQube** descargado.
2. Coloca la carpeta extraída en un directorio, por ejemplo, C:\SonarQube.

**5. Inicia SonarQube**

1. Abre una terminal (PowerShell o CMD) como administrador.
2. Ve al directorio C:\SonarQube\bin\windows-x86-64.
3. Ejecuta el archivo StartSonar.bat:

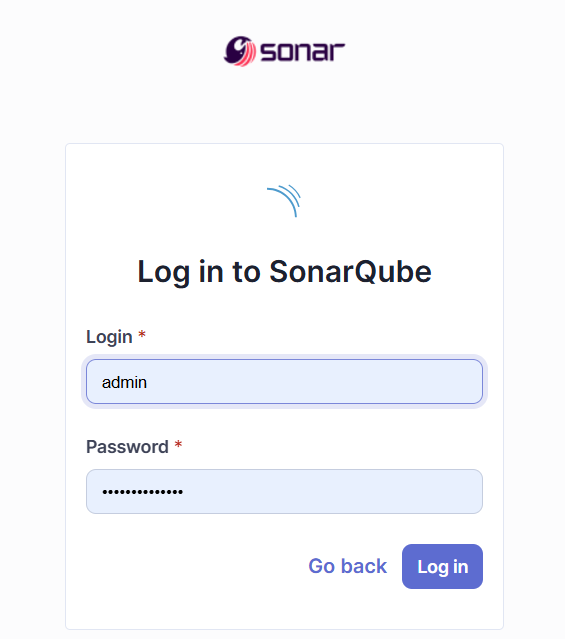


**6. Accede a la Interfaz Web**

1. Abre tu navegador y ve a:

* <http://localhost:9000>

1. Ingresa con las credenciales predeterminadas:
   * **Usuario:** admin
   * **Contraseña:** admin



# **Resultado de aplicación de SonarQube de Frontend**

Durante el análisis realizado con la herramienta SonarQube, se identificaron áreas clave de mejora en el código del Frontend, especialmente en las dimensiones de fiabilidad y mantenibilidad. A continuación, se presenta un resumen de los resultados obtenidos:

* **Seguridad (Security):** Sin problemas detectados (0).
* **Fiabilidad (Reliability):** Se encontraron 117 problemas que requieren atención para garantizar un correcto funcionamiento del sistema.
* **Mantenibilidad (Maintainability):** Se identificaron 145 puntos de mejora, evidenciando un considerable potencial de optimización en el formato y estructura del código.
* **Cobertura (Coverage):** Sin análisis aplicable en esta revisión (0.0%).
* **Hotspots Revisados**: Ninguno (0.0%).
* **Duplicaciones (Duplications):** Se detectó un 7.6% de código duplicado, que puede simplificarse y modularizarse para mejorar la eficiencia y el mantenimiento.



**Hallazgos Principales:**

El análisis destaca una alta densidad de recomendaciones relacionadas con **mantenibilidad**. Por ejemplo, uno de los problemas críticos identificados sugiere refactorizar el código para evitar la anidación de funciones con más de cuatro niveles de profundidad, lo que complica la legibilidad y dificulta su gestión. Este tipo de problemas cae bajo la categoría de "Code Smell" (malos olores en el código) y puede ser considerado como una prioridad alta para la corrección.

**Detalle de uno de los problemas detectados:**

* **Descripción:** "Refactor this code to not nest functions more than 4 levels deep."
* **Impacto:** Problema crítico de mantenibilidad.
* **Ubicación:** Línea 219.
* **Esfuerzo estimado de corrección:** 20 minutos.
* **Categoría:** Code Smell - Crítico.
* **Comentario:** Este problema puede derivar en dificultades futuras para el equipo de desarrollo al intentar escalar o depurar el código.

**Descripción General del Informe**

Las sugerencias identificadas por SonarQube pertenecen, en su mayoría, a la categoría de **mantenibilidad**, destacándose por proponer optimizaciones en las funcionalidades del código y en su formato general. Este enfoque busca asegurar que el código sea más entendible, escalable y menos propenso a errores durante futuros desarrollos o mantenimientos.

**Solución Propuesta**

Para abordar las recomendaciones proporcionadas por SonarQube, se deben implementar los cambios sugeridos, priorizando los siguientes aspectos:

1. **Refactorización de funciones:** Reducir la complejidad estructural del código, eliminando anidaciones profundas y promoviendo una modularización efectiva.
2. **Eliminación de duplicados:** Identificar y consolidar fragmentos de código redundante en funciones reutilizables o componentes comunes.
3. **Mejora de prácticas de desarrollo:** Incorporar estándares y guías claras para el equipo, incluyendo revisión de código (Code Reviews) y pruebas automatizadas para validar los cambios.
4. **Seguimiento de métricas:** Continuar utilizando herramientas como SonarQube para monitorear y mantener la calidad del código a lo largo del tiempo.

# **Resultado de aplicación de SonarQube de Backend**

El análisis realizado con SonarQube al código del Backend identifica áreas importantes de mejora, especialmente en las categorías de mantenibilidad y duplicación de código. A continuación, se detallan los resultados principales:

**Resumen de Resultados:**

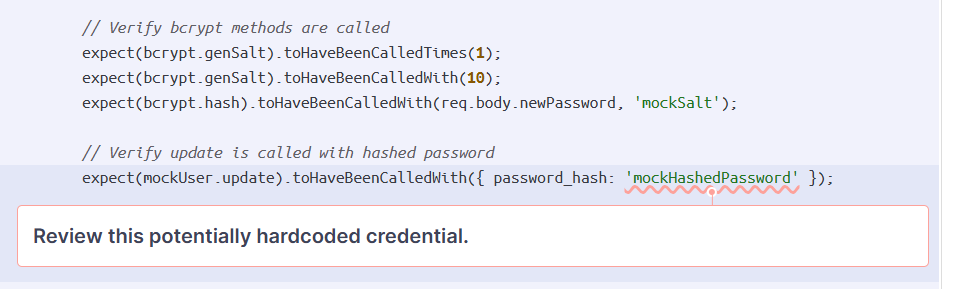
* **Seguridad (Security):** No se detectaron problemas abiertos (0). Calificación: **A**.
* **Fiabilidad (Reliability):** Se encontraron **2 problemas abiertos**, lo que indica un buen nivel general de robustez, pero con oportunidades de mejora. Calificación:
* **Mantenibilidad (Maintainability):** Se identificaron **61 problemas abiertos**, lo que resalta la necesidad de refactorizar el código para optimizar su mantenimiento y escalabilidad. Calificación: **C**.
* **Cobertura (Coverage):** La cobertura de código es **0.0%** sobre un total de 1.9k líneas analizadas. Esto implica que no hay pruebas automatizadas que respalden el código revisado.
* **Duplicaciones (Duplications):** Un **16.9% de duplicación** en 5.8k líneas analizadas representa un punto crítico a resolver para mejorar la eficiencia del código.
* **Hotspots de Seguridad:** Se detectaron **27 hotspots**, los cuales requieren un análisis adicional para determinar su relevancia y priorización.

**Hallazgos Principales**

1. **Mantenibilidad:**
   * Los 61 problemas abiertos evidencian una necesidad significativa de optimización del código, especialmente para mejorar la estructura y reducir la complejidad,

Este factor es clave para facilitar futuras modificaciones y garantizar un desarrollo eficiente.

1. **Duplicación de Código:**
   * El alto porcentaje de duplicación (16.9%) es un área crítica que afecta tanto la legibilidad como la mantenibilidad del proyecto.
   * La refactorización de fragmentos redundantes en componentes reutilizables reducirá significativamente este porcentaje.
2. **Cobertura de Pruebas:**
   * La falta de cobertura (0.0%) indica que no se han implementado pruebas automatizadas en el proyecto. Esto aumenta el riesgo de introducir errores durante futuras implementaciones.
3. **Hotspots de Seguridad:**
   * Los 27 hotspots de seguridad requieren análisis detallado para descartar vulnerabilidades críticas. Aunque no se clasifican como problemas abiertos, deben abordarse de manera preventiva, actualmente solo son advertencias generadas por pruebas de “Mockeo”



# **Descripción General del Informe**

Las observaciones resaltan que el código del Backend presenta fortalezas en términos de seguridad y fiabilidad general. Sin embargo, la mantenibilidad y las duplicaciones de código requieren atención inmediata para garantizar un desarrollo sostenible. La falta de pruebas automatizadas es un riesgo que debe abordarse para mejorar la estabilidad y calidad del proyecto.

**Soluciones Propuestas**

1. **Mantenibilidad:**
   * Refactorizar las secciones del código con problemas abiertos para reducir la complejidad, siguiendo principios de diseño limpio (Clean Code).
   * Implementar patrones de diseño para modularizar el código, facilitando su reutilización.
2. **Duplicaciones:**
   * Identificar las secciones duplicadas y consolidarlas en funciones o clases reutilizables.
   * Establecer revisiones de código (Code Reviews) más rigurosas para prevenir nuevas duplicaciones.
3. **Cobertura de Pruebas:**
   * Diseñar e implementar pruebas unitarias para alcanzar un nivel mínimo aceptable de cobertura.
   * Incorporar pruebas de integración para garantizar la funcionalidad de módulos interconectados.
4. **Hotspots de Seguridad:**
   * Realizar un análisis detallado de cada hotspot y priorizar los que puedan representar riesgos críticos.
   * Implementar herramientas de pruebas de seguridad (e.g., SAST y DAST) como parte del pipeline de integración continua.

# **Conclusión**

El análisis realizado con las herramientas Nessus y SonarQube permitió identificar áreas críticas de mejora en los sistemas evaluados, abarcando tanto la seguridad informática como la calidad del código. A continuación, se resumen los principales hallazgos y recomendaciones:

1. **Seguridad (Nessus):**
   * Nessus identificó vulnerabilidades clasificadas como de mediana severidad en los sistemas analizados, particularmente relacionadas con versiones desactualizadas de software y configuraciones inseguras. Estas vulnerabilidades representan riesgos potenciales, como ejecuciones remotas de código o fallos en la aplicación, que deben ser atendidos con prioridad.
   * Las recomendaciones incluyen la actualización de componentes como OpenSSL y Apache, además de la revisión de configuraciones para minimizar la superficie de ataque.
2. **Calidad del Código (SonarQube):**
   * SonarQube destacó problemas en la mantenibilidad del código, con un alto número de problemas abiertos y una significativa duplicación de código. Estos factores aumentan la deuda técnica y dificultan el mantenimiento y escalabilidad del sistema.
   * La cobertura de pruebas automatizadas fue inexistente, lo que eleva el riesgo de errores y afecta la estabilidad general del software.
   * Se detectaron hotspots de seguridad que, aunque no se clasifican como vulnerabilidades abiertas, requieren análisis detallado para prevenir futuros problemas.

**Recomendaciones Prioritarias:**

1. **Desde la perspectiva de seguridad:**
   * Actualizar todos los componentes desactualizados identificados por Nessus.
   * Fortalecer las configuraciones de seguridad y restringir accesos innecesarios.
   * Realizar evaluaciones periódicas para asegurar la mitigación de futuras vulnerabilidades.
2. **Desde la perspectiva de calidad del código:**
   * Refactorizar las secciones de código con problemas de mantenibilidad, eliminando redundancias y simplificando su estructura.
   * Diseñar e implementar pruebas unitarias y de integración para alcanzar un nivel aceptable de cobertura.
   * Establecer procesos de revisión de código y prácticas de desarrollo limpio para prevenir problemas futuros.

**Notas finales**

El uso de Nessus y SonarQube ha permitido validar y reforzar la calidad actual del sistema Medical Aid, demostrando que se encuentra en una sólida posición para enfrentar desafíos de seguridad y calidad de desarrollo. Los análisis realizados confirmaron la estabilidad del código y su capacidad para ser optimizado de manera eficiente. Las áreas de mejora identificadas representan oportunidades para alcanzar un nivel aún más alto de desempeño y robustez.

La implementación de las recomendaciones propuestas fortalecerá la confiabilidad del sistema, asegurando que Medical Aid continúe operando con altos estándares de seguridad y calidad. Este proceso evidencia el compromiso de mantener un enfoque preventivo y de mejora continua, posicionando a la herramienta como un referente confiable en su ámbito.