"Año del Bicentenario del Perú: 200 años de Independencia"

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

Escuela profesional de Ingeniería de Software

GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE - G1

INTEGRANTES:

Chavez Burgos, Luiz Arnold (19200250)

López Loaiza, Edgar Fernando (19200295)

Huamán Ampuero, Lucero Marysol (19200081)

Magallanes Quiroz, Claudia Carolina (18200331)

Quispe Alarcon, Jorge Luis (19200094)

Romero Angeles, Luis Alfredo Felix (19200317)

Zafra Moran, Rolando Jesus (19200262)

Zarate Villar, Jhennyfer Nayeli (19200248)

Calderon Herrera Miguel Angel (19200071)

Tomasto Solis, Victor Eduardo (18200299)

DOCENTE RESPONSABLE:

Prof. Espinoza Robles, Armando David

Herramienta para Testear: SonarQube

Para el proyecto de nuestra página web “MMP”, usaremos la herramienta SonarQube para una revisión automática de código, para poder detectar bugs, vulnerabilidades y code smells. También integraremos el flujo de trabajo para permitir la inspección continua del código en las ramas del proyecto.

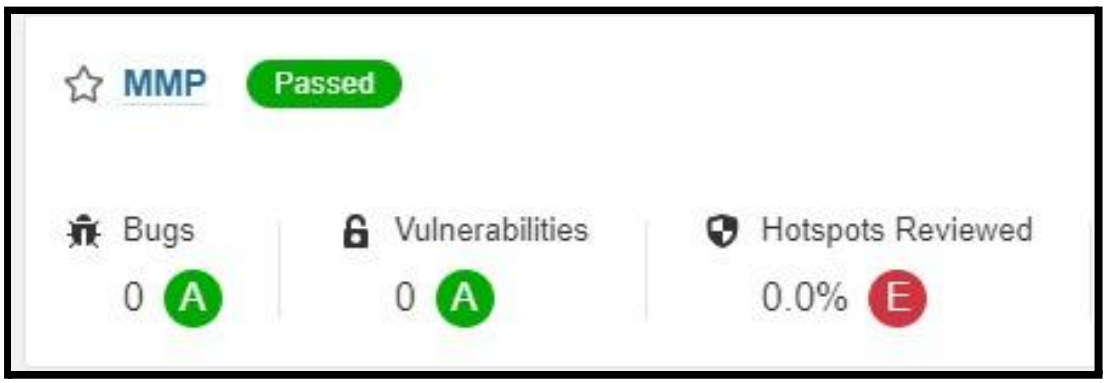
Terminología

- ➔ **Bug :** un punto de fallo real y debe repararse.
- ➔ **Vulnerability:** un agujero de seguridad que puede usarse para atacar nuestro software.
- ➔ **Code Smell :** Problema relacionado con la mantenibilidad en el código y que sea confuso.
- ➔ **Security Hotspot:** detecta un problema de seguridad.
- ➔ **Gravedad del problema**
  Severidad Bloqueador: es un error que afecta el comportamiento de nuestro sistema en producción.
  Severidad Crítico: error que afecta el comportamiento del sistema en una falla de seguridad.
  Severidad Mayor: es un defecto de calidad que afecta a la productividad del desarrollador.
  Severidad Menor: es un defecto de calidad que afecta en la productividad del programador.
  Severidad de Información: no contiene ningún defecto de calidad pero sí un hallazgo.

Características

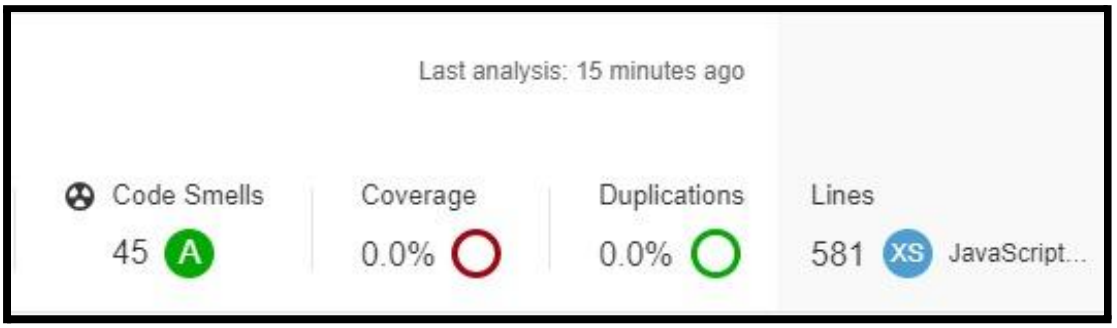
- **Calidad de código**
  Nos permitirá medir la calidad de código de nuestro proyecto, serán evaluados por SonarQube y nos mostraras las métricas.
- **Código duplicado**
  Nos ayudará a detectar si el código fuente se duplica más de una vez.
- **Código muerto**
  Detectará si existe un código muerto en el programa.
- **Estándares de codificación**
  Nos ayudará en las convenciones para escribir código fuente dependiendo del lenguaje de programación utilizado.
- **Complejidad ciclomática**
  Es una métrica de calidad de software, que nos permitirá ver los caminos independientes que tiene nuestro código.
- **Comentarios**
  Se puede agregar comentarios con el propósito de que el código fuente sea más sencillo de entender para su mantenimiento o reutilización.
- **Cobertura de código**
  Es una medida que nos indica el porcentaje de código validado por los tests.

Figura N°1: Resultados de Análisis N°1



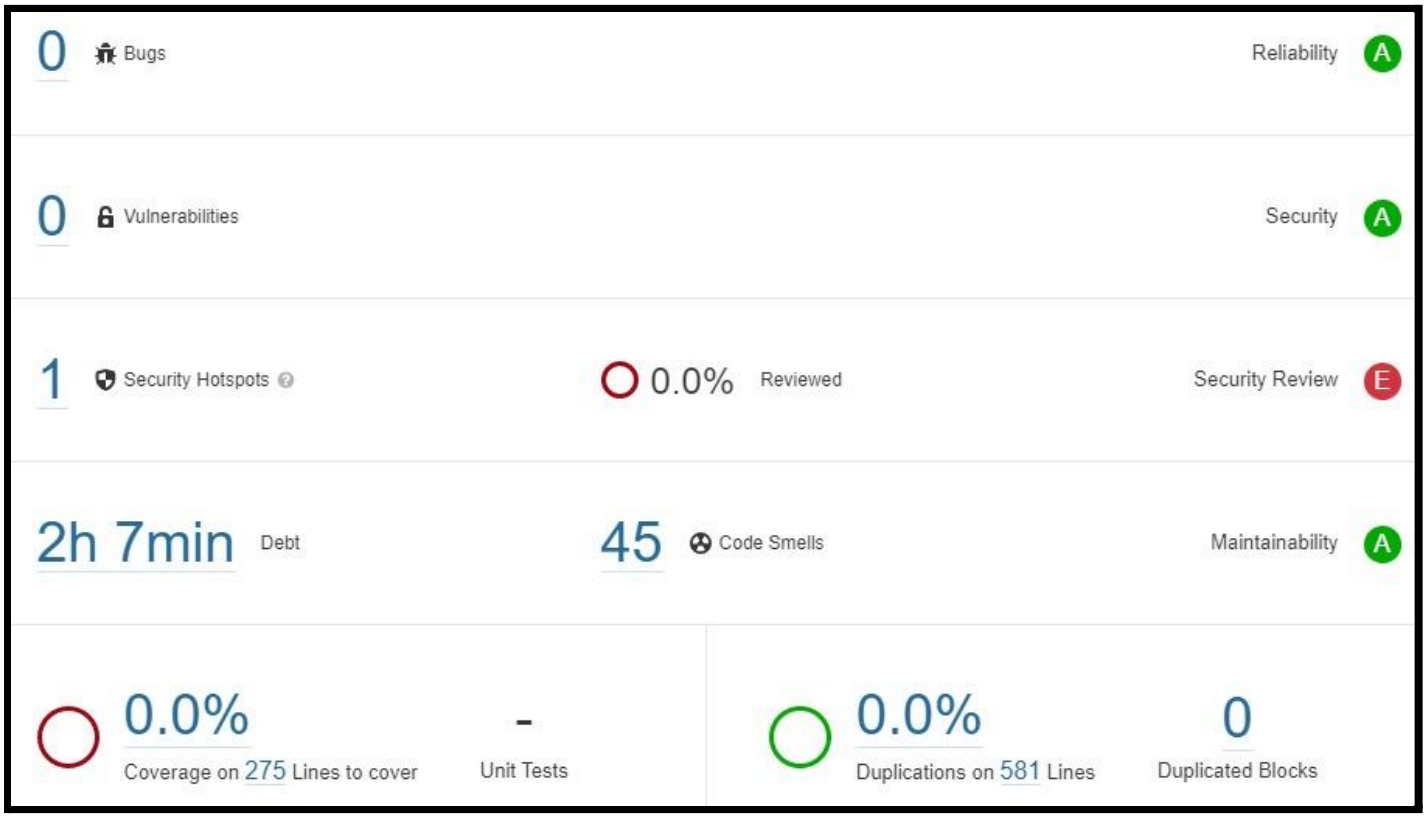
Nota : Después de realizar el análisis el programa dio como resultado los siguientes datos, cero bugs, cero vulnerabilidades, y un fuerte error de seguridad, más adelante se explorará a fondo estas.

Figura N°2: Resultados de Análisis N°3



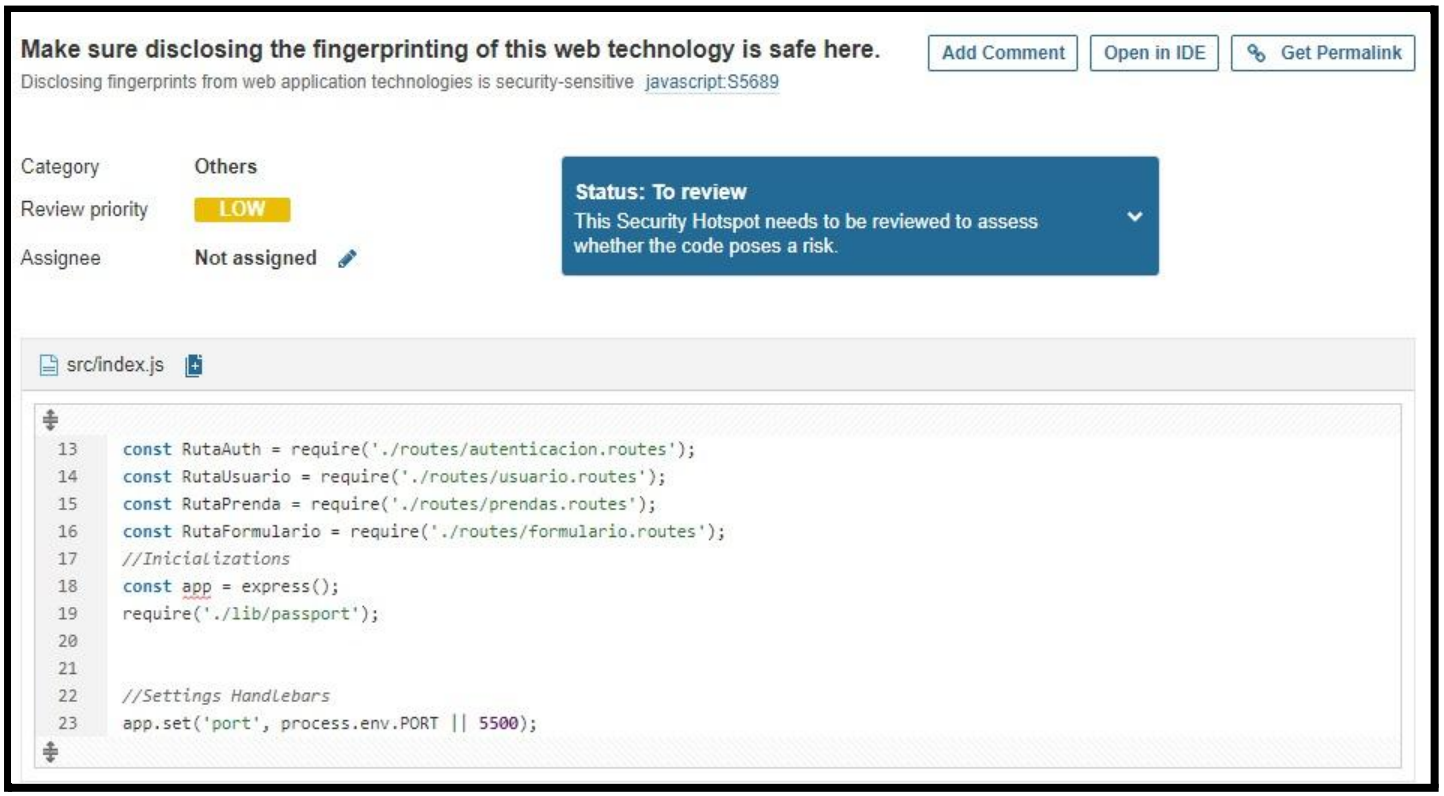
Nota: Después de realizar el análisis el programa dio como resultado los siguientes datos, 45 code smells (errores mínimos), cero por ciento de coverage y cero por ciento de duplicaciones de código, más adelante se explorará a fondo estas

Figura N°3: Resultado de Análisis N°3



Nota: Después de realizar el análisis los resultados generales fueron, cero bugs, cero vulnerabilidades, un error de seguridad, dos horas con 7 minutos de deuda de programación y 45 errores pequeños. Esto da un resultado de A, se pasó el test

Figura N°4: Security Hotspot



Nota: Para poder asegurarnos que las huellas digitales estén correctas, se revisa el Security Hotspot deben evaluar el código si es que este presenta un riesgo. La prioridad de revisión es bajo por lo cual el peligro es mínimo.