**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

***Escuela Profesional Académica de Ingeniería de Software***

**SISTEMA DE PLANIFICACIÓN DE ESTUDIOS UNIVERSITARIOS**

**FollowClass (FC)**

**PLAN DE PROYECTO**

**Documento de Pruebas del Software**

**GRUPO 6**

**DOCENTE:** Dra. Lenis Rossi Wong Portillo

**CURSO:** Gestión de la Configuración del Software

**COORDINADORA:** Romero Diaz, Bianca Elizabeth

**INTEGRANTES:**

Balandra Camacho, Ivan 20200248

Hernández Bianchi, Stefano Alessandro 20200309

Marcelo Salinas, Moises Enrique 20200310

Ortiz Crisostomo, Edwin Jose 14200224

Quispe Fajardo, Adrián Ismael 20200281

Romero Diaz, Bianca Elizabeth 20200312

Solis Flores, Aldair Jhostin 20200293

**2022-I**

**HISTORIAL DE CAMBIOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Autor(es)** | **Descripción** | **Fecha** |
| ***1.0*** | Solis Flores, Aldair  Romero Diaz, Bianca Elizabeth  Quispe Fajardo, Adrián  Ortiz Crisostomo, Edwin Jose | * Introducción * Herramienta Sonarqube * Resultados del primer análisis | 26.07.22 |
| ***2.0*** | Solis Flores, Aldair  Romero Diaz, Bianca Elizabeth  Quispe Fajardo, Adrián  Ortiz Crisostomo, Edwin Jose | * Resultados del segundo análisis | 04.08.22 |

**ÍNDICE**

[1. Introducción 4](#_Toc110545482)

[1.1. Propósito 4](#_Toc110545483)

[1.2. Objetivo 4](#_Toc110545484)

[2. Herramienta Sonarqube 4](#_Toc110545485)

[3. Resultados del primer análisis 4](#_Toc110545486)

[3.1. Vista general del proyecto 5](#_Toc110545487)

[3.2. Bugs 6](#_Toc110545488)

[3.3. Security Hotspots 6](#_Toc110545489)

[3.3.1. Autenticación 6](#_Toc110545490)

[3.3.2. SQL Injection 7](#_Toc110545491)

[3.4. Code Smell 8](#_Toc110545492)

[3.4.1. Minor 8](#_Toc110545493)

[3.4.2. Major 9](#_Toc110545494)

[3.4.3. Critical 9](#_Toc110545495)

[3.4.4. Bloquer 9](#_Toc110545496)

[4. Resultados del segundo análisis 10](#_Toc110545497)

[4.1. Vista general del proyecto 11](#_Toc110545498)

[4.2. Code Smell 11](#_Toc110545499)

# 

**DOCUMENTACIÓN DE PRUEBAS DE SOFTWARE**

# Introducción

# Propósito

En este documento se detallan los resultados del análisis estático del código fuente, así como los resultados tras la corrección.

# Objetivo

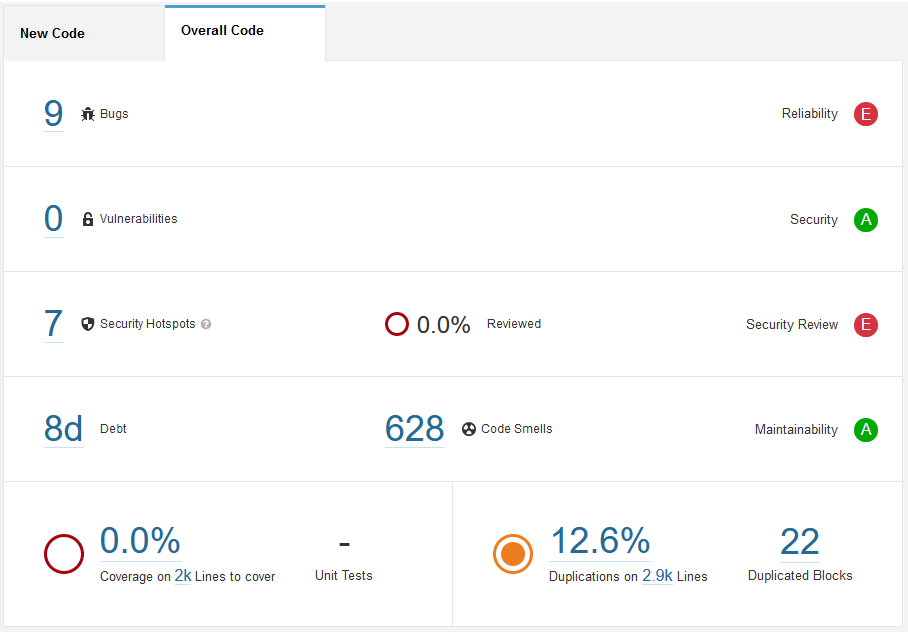
Documentar los resultados del análisis estático para que el equipo de desarrollo realice la mejora del código fuente.

# Herramienta Sonarqube

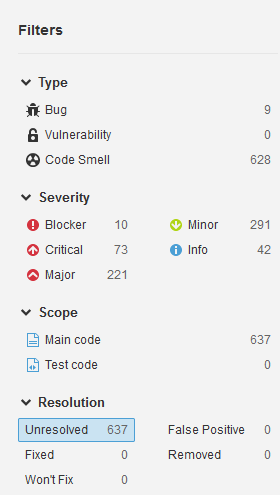
Sonarqube es una plataforma que evalúa la calidad del código fuente a través de un análisis estático sobre el código (sin ejecutarlo). El objetivo es reconocer los aspectos que requieren de mejora que podrían causar complicaciones en el desarrollo, además de obtener otras métricas. De esta manera, es posible mejorar la calidad del código.

# Resultados del primer análisis

A continuación, se muestra el resultado del primer análisis estático del código fuente. La calificación en fiabilidad es de E con 9 bugs en total, la calificación de seguridad es de A, la revisión de seguridad tiene una calificación de E debido a 7 security hotspots, en mantenibilidad tiene una calificación de A con 628 code smells.

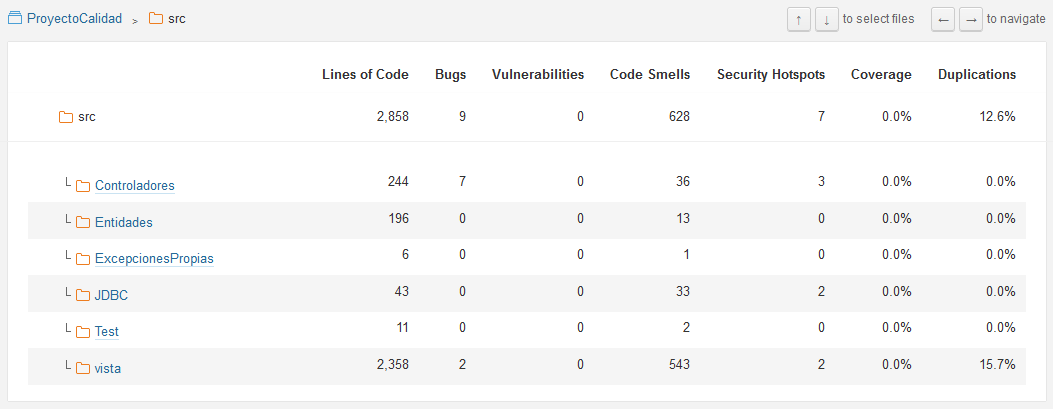


Además, se muestra la severidad de los issues identificados por sonarqube. De estos, la mayoría son calificados como minor, solo 10 son calificados como bloquer.



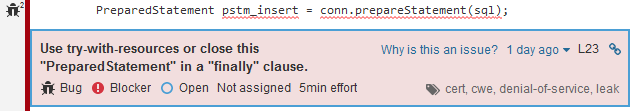
## Vista general del proyecto

La siguiente imagen muestra la evaluación del código fuente por paquete



## Bugs

|  |  |
| --- | --- |
| Issue | Use try-with-resources or close this “Prepared Statement” in a “finally” clause. |
| Severidad | Blocker |
| Cantidad | 9 |
| Items | paquete Controladores y Vista |
| Riesgo | Si no se cierran correctamente los recursos, se producirá una pérdida de recursos que podría hacer caer primero a la aplicación y luego, quizás, a la consola en la que se encuentra la aplicación. |

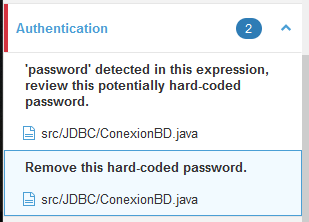


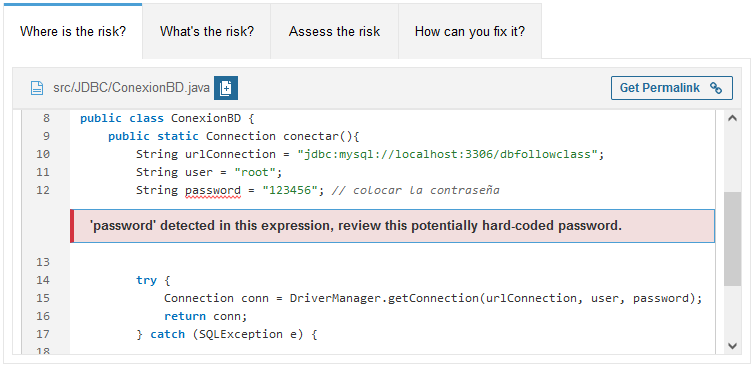
## Security Hotspots

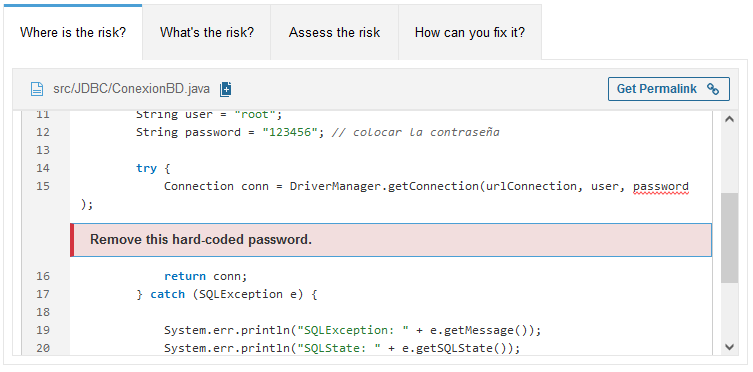
Se detectaron 7 problemas de este tipo: 2 de autenticación y 5 de SQL injection

### Autenticación

|  |  |
| --- | --- |
| Issue | Hard-coded password |
| Cantidad | 9 |
| Items | ConexiónBD |
| Riesgo | Dado que es fácil de extraer del código fuente o del binario de una aplicación, las contraseñas no deberían estar en código duro. |

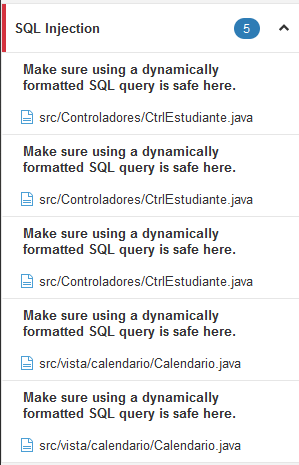


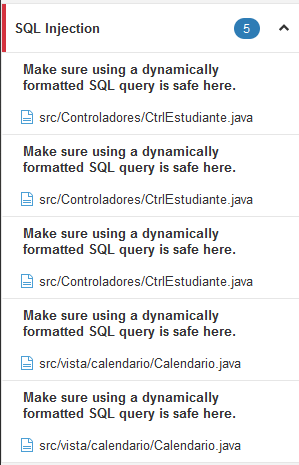


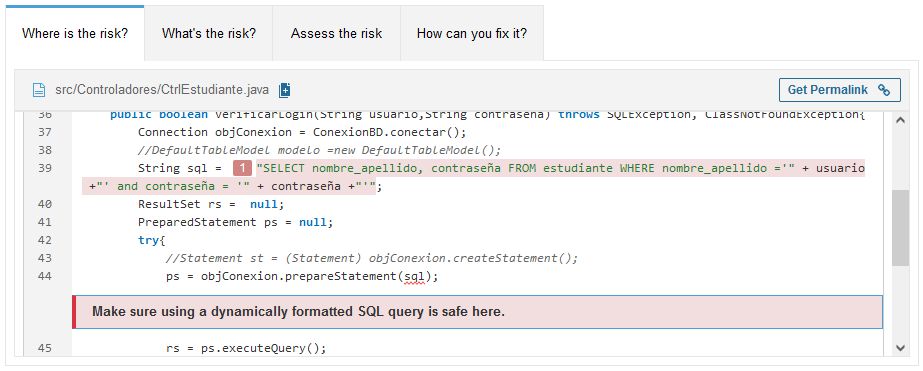


### SQL Injection

|  |  |
| --- | --- |
| Issue | dynamic format SQL query safe. |
| Cantidad | 5 |
| Items | paquete Controladores y el item Calendario |
| Riesgo | Las consultas SQL con formato pueden ser difíciles de mantener, depurar y pueden aumentar el riesgo de inyección SQL al concatenar valores no contrastados en la consulta. |







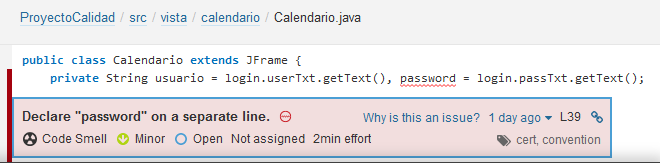
## Code Smell

En los siguientes apartados se mostrarán issues code smell que sí puedes ser mejorados.

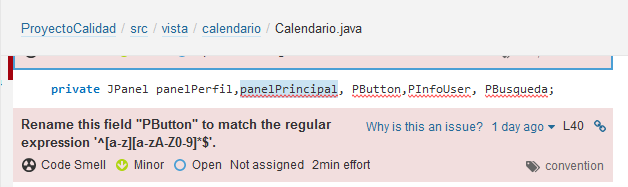
### Minor

A continuación, se mostrarán algunos de los issues encontrados clasificados como code smell que fueron encontrados en diversos items tipo fuente.

|  |  |
| --- | --- |
| Issue | Declarar variable en una línea separada. |
| Severidad | Minor |
| Riesgo | Declarar múltiples variables en una línea es difícil de leer. |

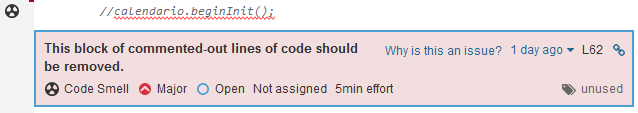


|  |  |
| --- | --- |
| Issue | Renombrar un campo para que coincida con [a-z][a-zA-Z0-9]\*$ |
| Severidad | Minor |
| Riesgo | Se debe compartir algunas convenciones de nomenclatura es un punto clave para que un equipo pueda colaborar eficazmente. |



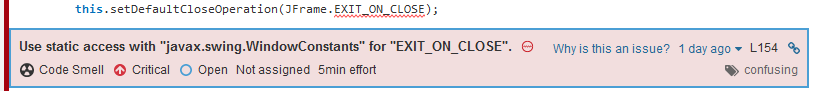
### Major

|  |  |
| --- | --- |
| Issue | Este bloque de líneas de código comentadas debe ser eliminado. |
| Severidad | Major |
| Riesgo | El código que no se utilice debe eliminarse pues reduce la legibilidad del código. |



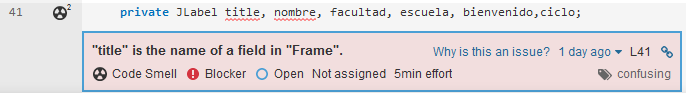
### Critical

|  |  |
| --- | --- |
| Issue | Usa accesos estáticos con “javax.swing.WindowConstants” for “EXIT\_ON\_CLOSE” |
| Severidad | Critical |
| Riesgo | Nunca se debe acceder a los miembros estáticos de una clase base utilizando el nombre de un tipo derivado. Hacerlo es confuso y podría crear la ilusión de que existen dos elementos estáticos diferentes. |



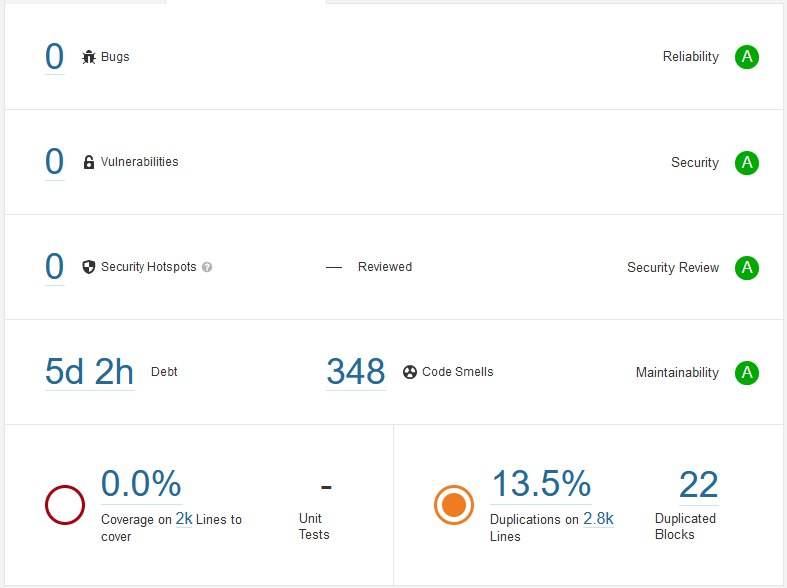
### Bloquer

|  |  |
| --- | --- |
| Issue | “\_\_” es el nombre de un campo en “\_\_” |
| Severidad | Bloquer |
| Riesgo | Tener una variable con el mismo nombre en dos clases no relacionadas está bien, pero hacerlo dentro de una jerarquía de clases obtendrá confusión en el mejor de los casos. |

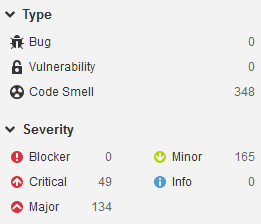


# Resultados del segundo análisis

A continuación, se muestra el resultado del segundo análisis estático del código fuente. La calificación en fiabilidad es de A con 0 bugs en total, la calificación de seguridad es de A, la revisión de seguridad tiene una calificación de A, en mantenibilidad tiene una calificación de A con 348 code smells.

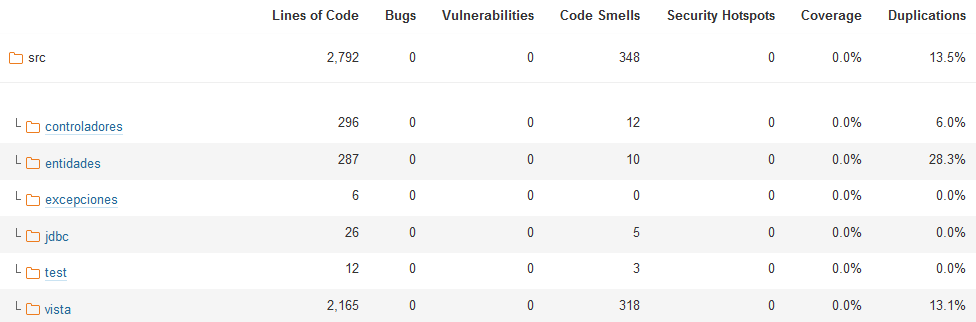
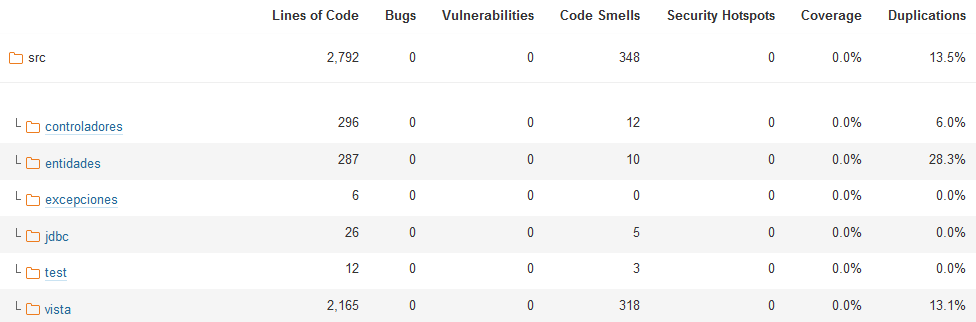


Además, se muestra la severidad de los issues identificados por sonarqube. De estos, la mayoría son calificados como minor.



## Vista general del proyecto

La siguiente imagen muestra la evaluación del código fuente por paquete



## Code Smell

Muchos de los resultados clasificados como code smell provienen de partes del código generado por la librería swing el cual no puede ser editado. Por lo tanto, está fuera del control del equipo de desarrollo a pesar de la severidad con la que ha sido clasificada. A continuación, se muestran algunos de los casos:

|  |  |
| --- | --- |
| Issue | Definir como constante en lugar de duplicar x veces |
| Items | paquete vista |
| Severidad | Crítico |
| Cantidad | 42 |

Imagen que contiene Texto

Descripción generada automáticamente

|  |  |
| --- | --- |
| Issue | Eliminar el parámetro de método no utilizado |
| Items | paquete vista |
| Severidad | Major |
| Cantidad | 25 |

Escala de tiempo

Descripción generada automáticamente con confianza media

|  |  |
| --- | --- |
| Issue | Eliminar el campo y declararlo como variable local en los métodos correspondientes. |
| Items | paquete vista |
| Severidad | Minor |
| Cantidad | 101 |