

# PRACTICA 2

TESTING E2E Y SONAR

AMPLIACIÓN DE INGENIERÍA DEL SOFTWARE  
2023/2024

Ariel Carnés Blasco

# ÍNDICE

Overview inicial.....	2
Corrección de issues.....	6
Overview corregido.....	17

# ÍNDICE DE FIGURAS

Figura 1. Primer preview de sonar.....	2
Figura 2. Código duplicado.....	2
Figura 3. Complejidad ciclomática.....	3
Figura 4. Gráfico de deuda técnica.....	3
Figura 5. Deuda técnica total.....	4
Figura 6. Cobertura de los tests.....	4
Figura 7. Lista de issues.....	5
Figura 8. Issue 1.....	6
Figura 9. Issue 1 resuelto.....	6
Figura 10. Issue 2.....	7
Figura 11 y 12. Issue 2 resuelto.....	7
Figura 13. Issue 3.....	8
Figura 14. Issue 3 resuelto.....	8
Figura 15. Issue 4.....	9
Figura 16. Issue 4 resuelto.....	9
Figura 17. Issue 5.....	10
Figura 18. Issue 5 resuelto.....	10
Figura 19. Issue 6.....	11
Figura 20. Issue 6 resuelto.....	11
Figura 21. Issue 7.....	12
Figura 22. Issue 8.....	13
Figura 23. Issue 8 resuelto.....	13
Figura 24. Issue 9.....	14
Figura 25. Issue 9 resuelto.....	14
Figura 26. Issue 10.....	15
Figura 27. Issue 10 resuelto.....	15
Figura 28. Overview final.....	16
Figura 29. Issues solucionadas con false positive.....	16

# OVERVIEW INICIAL

Nada más entrar en sonar desde localhost:9000 se puede entrar en el proyecto que se desee para ver un resumen de los datos más importantes.

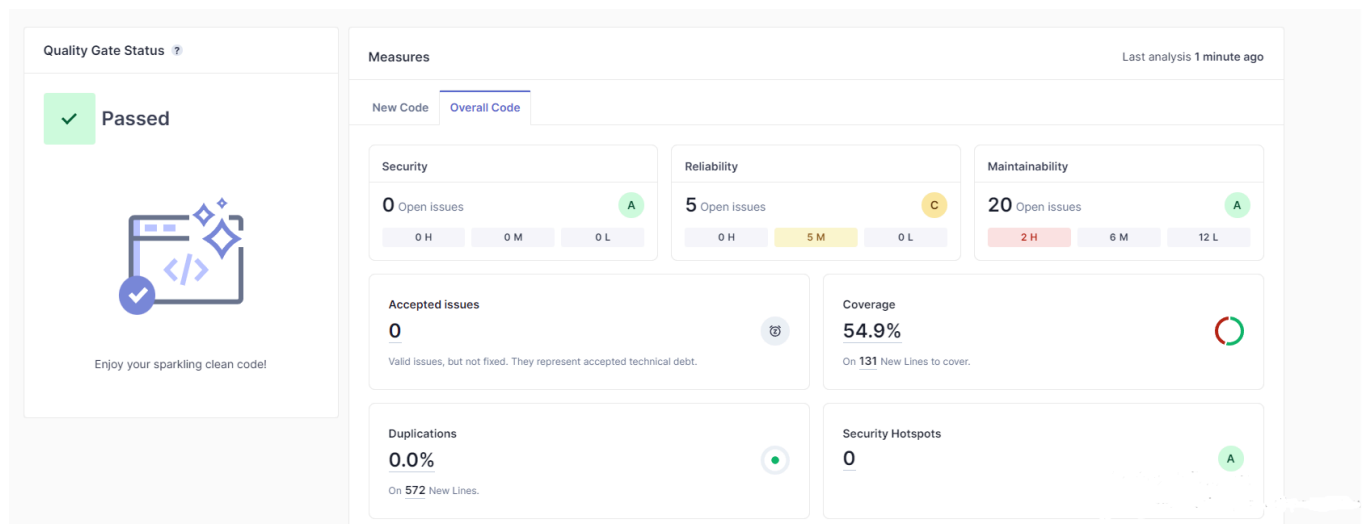
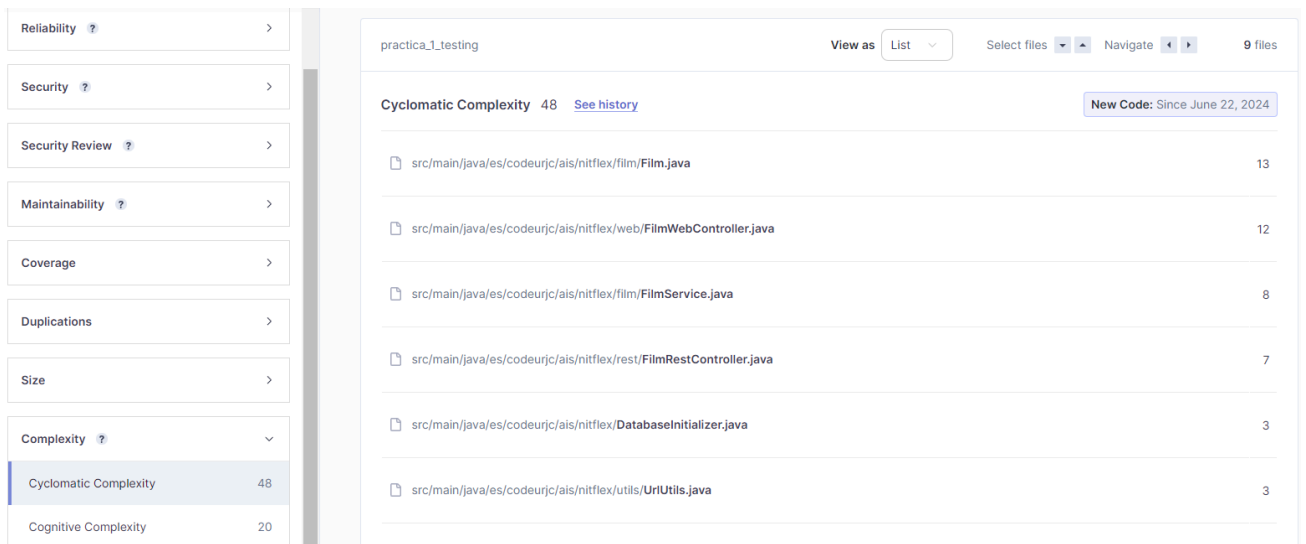


Figura 1. Primer Overview de Sonar.

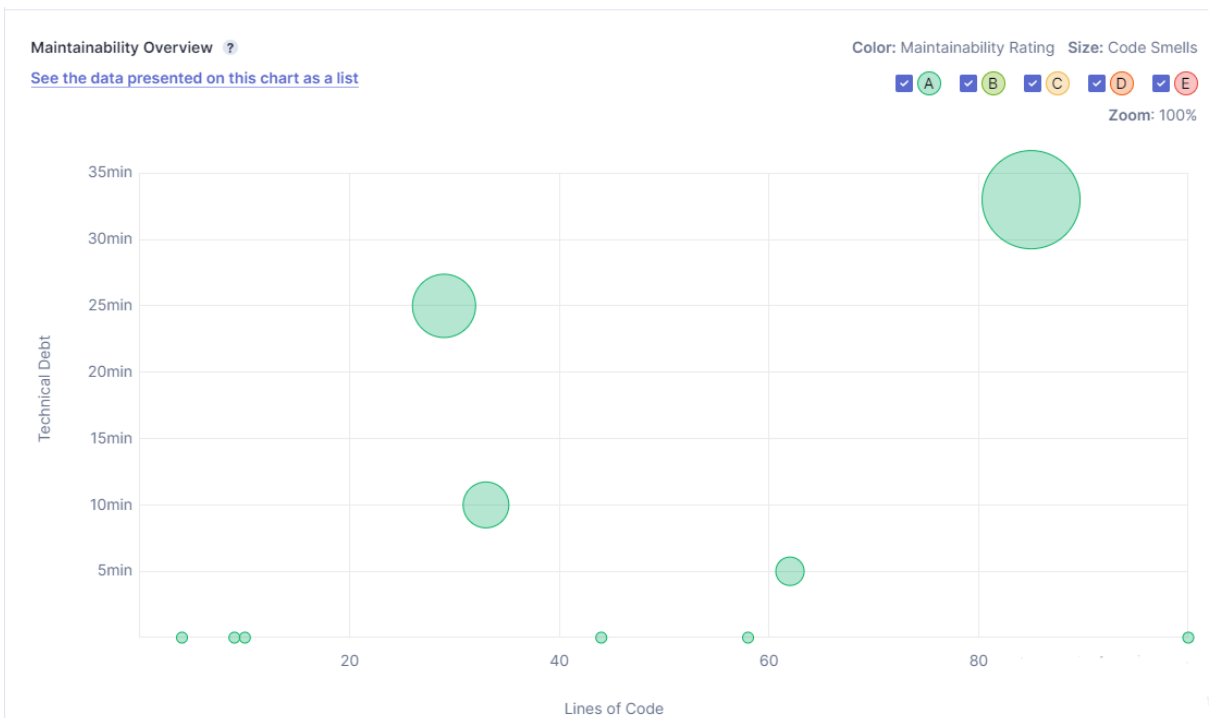
The table shows the duplicated lines for the project 'practica\_1\_testing'. The overall duplicated lines percentage is 0.0%. The table lists four files, all with 0.0% duplicated lines and 0 duplicated lines.

practica_1_testing		View as	Select files	Navigate	13 files
Duplicated Lines (%) 0.0%		List			
					New Code: Since June 22, 2024
			Duplicated Lines (%)	Duplicated Lines	
src/main/java/es/codeurjc/ais/nitflex/	Application.java		0.0%	0	
src/main/java/es/codeurjc/ais/nitflex/	DatabaseInitializer.java		0.0%	0	
src/main/java/es/codeurjc/ais/nitflex/film/	Film.java		0.0%	0	
src/main/java/es/codeurjc/ais/nitflex/film/	FilmRepository.java		0.0%	0	







Figura 2. Código duplicado.



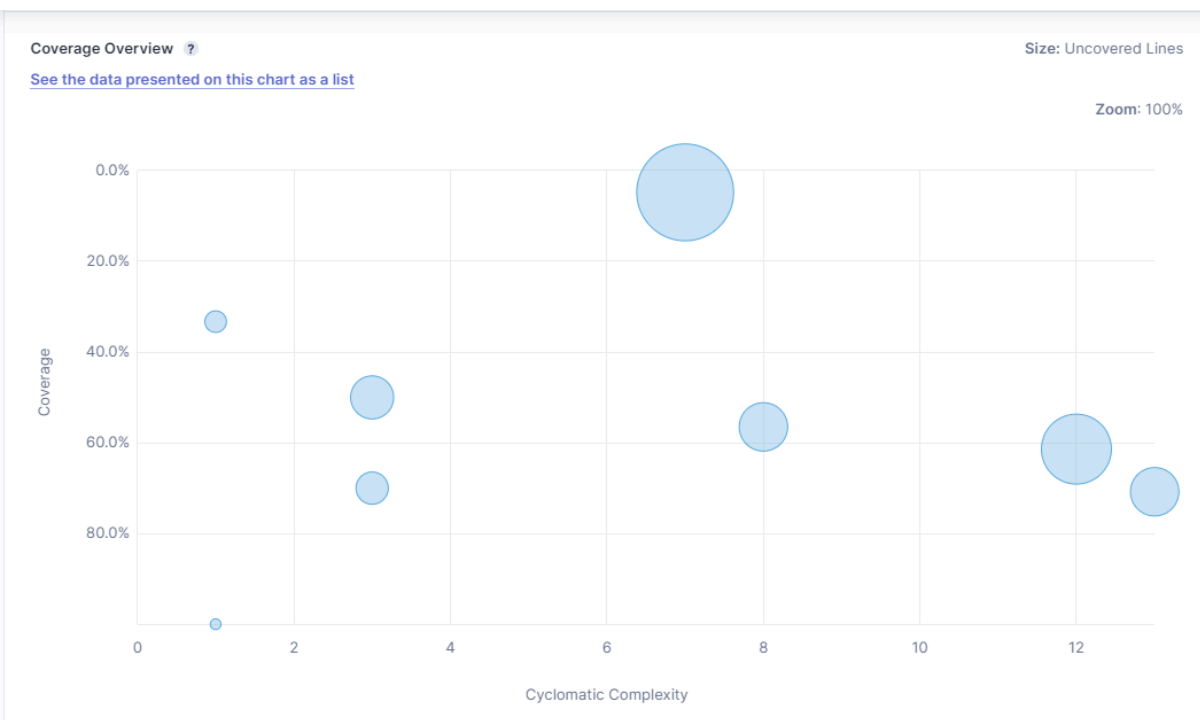
**Figura 3.** Complejidad ciclomática.



**Figura 4.** Gráfico de deuda técnica.

Technical Debt 1h 31min <a href="#">See history</a>	
 src/main/java/es/codeurjc/ais/nitflex/web/FilmWebController.java	33min
 src/main/java/es/codeurjc/ais/nitflex/DatabaseInitializer.java	25min
 src/main/java/es/codeurjc/ais/nitflex/urls/UrlUtils.java	10min
 src/test/java/es/codeurjc/ais/nitflex/film/GuardarYBorrarTest.java	6min
 src/test/java/es/codeurjc/ais/nitflex/E2E/ModificacionPelículasTest.java	6min
 src/test/java/es/codeurjc/ais/nitflex/urls/URLTest.java	6min

**Figura 5.** Deuda técnica total.



**Figura 6.** Cobertura de los tests.

My Issues

All

Filters

Issues in new code

Clean Code Attribute

Consistency

3

Intentionality

15

Adaptability

4

Responsibility

0

Software Quality

Security

0

Reliability

5

Maintainability

20

Severity

?

Bulk Change

Select issues

Navigate to issue

22 issues

1h 46min effort

src/.../java/es/codeurjc/ais/nitflex/DatabaseInitializer.java

Remove this field injection and use constructor injection instead.

Consistency

Maintainability

Reliability

No tags

Open

Not assigned

L13

5min effort

46 minutes ago

Code Smell

Major

src/.../es/codeurjc/ais/nitflex/film/FilmService.java

Replace this use of System.out by a logger.

Adaptability

Maintainability

bad-practice

cert

Open

Not assigned

L32

10min effort

46 minutes ago

Code Smell

Major

Replace this use of System.out by a logger.

Adaptability

Maintainability

bad-practice

cert

Open

Not assigned

L35

10min effort

46 minutes ago

Code Smell

Major

src/.../es/codeurjc/ais/nitflex/film/FilmService.java

Strings and Boxed types should be compared using "equals()".

Intentionality

Figura 7. Lista de issues.

5

# CORRECCIÓN DE ISSUES

## Issue 1

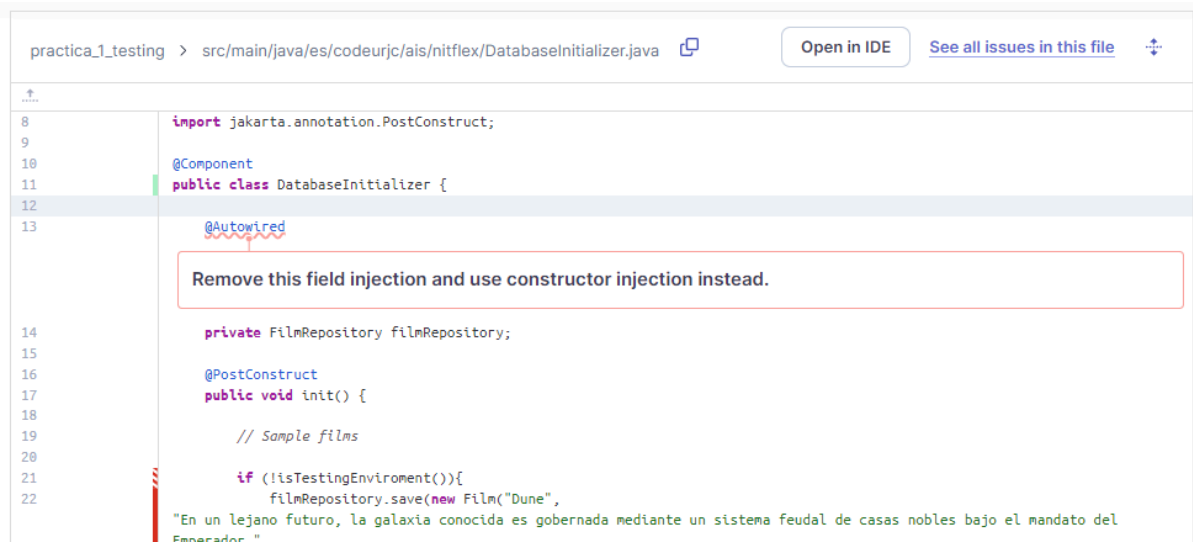


Figura 8. Issue 1.

### Resolución:

Sonarqube nos indica que la inyección de dependencias, que realizamos `@Autowired` o `@Injection`, es poco clara y difícil de mantener. En su lugar usaremos constructores de inyección.

Este issue aparece 2 veces más con `FilmWebController` y `FilmRestController`. Se soluciona de la misma forma.

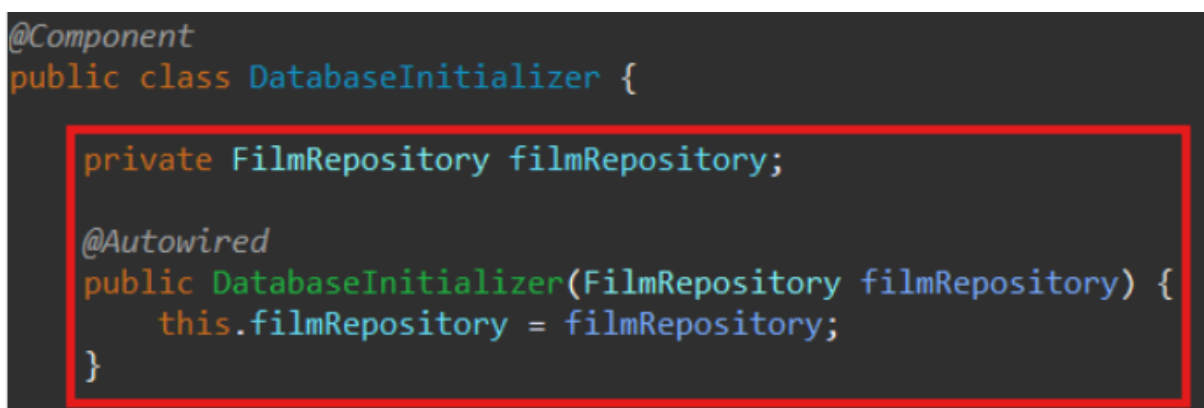


Figura 9. Issue 1 resuelto.

## Issue 2

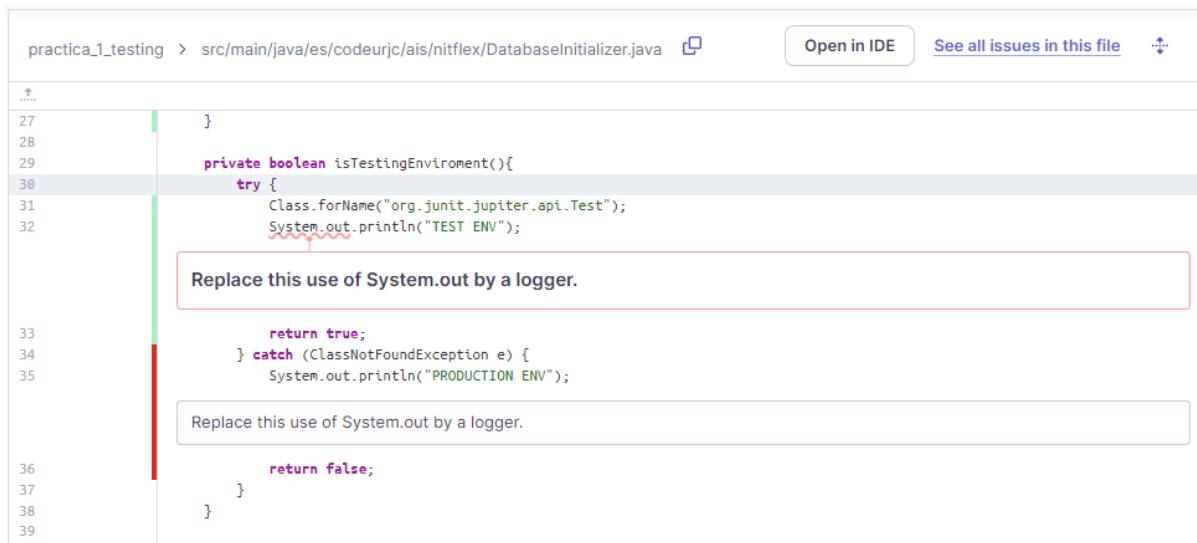


Figura 10. Issue 2.

### Resolución:

Sonarqube nos está recordando la importancia de los logs a la hora de crear software para mantener un registro de todo lo que ocurre durante la ejecución de código. Por este motivo cambiaremos los System.out por loggers.

Sonarqube nos indica 2 issues por este mismo motivo, como podemos ver en la captura. Los resolveremos de la misma forma.

```
@Component
public class DatabaseInitializer {

    Logger logger = Logger.getLogger(getClass().getName());

    private FilmRepository filmRepository;

    @Autowired
    public DatabaseInitializer(FilmRepository filmRepository) {
        this.filmRepository = filmRepository;
    }
}
```

```
37     private boolean isTestingEnviroment(){
38         try {
39             Class.forName("org.junit.jupiter.api.Test");
40             logger.info("TEST ENV");
41             return true;
42         } catch (ClassNotFoundException e) {
43             logger.info("PRODUCTION ENV");
44             return false;
45         }
46     }
47 }
```

Figuras 11 y 12. Issue 2 resuelto.



## Issue 3

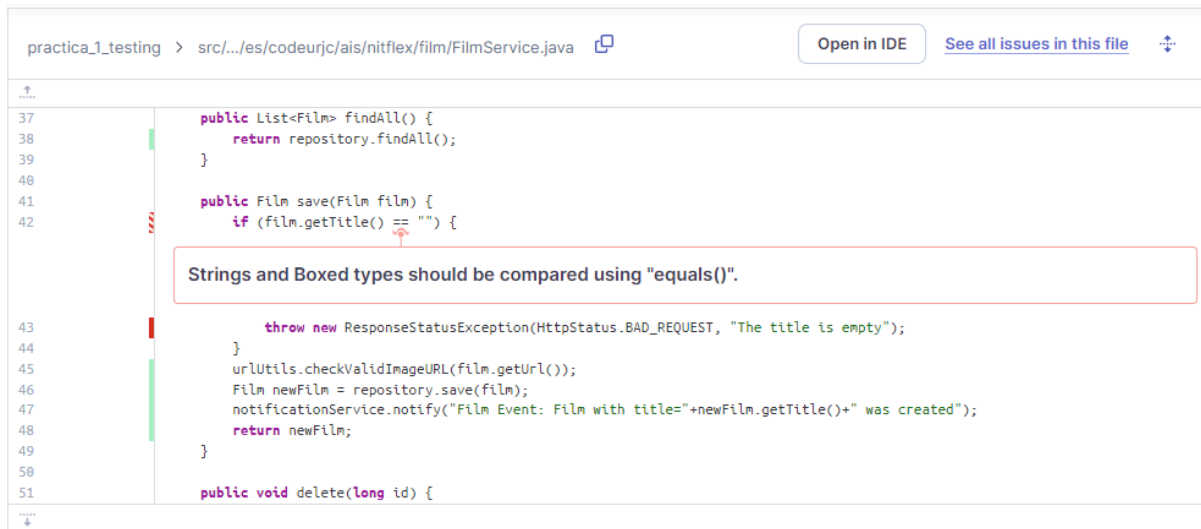


Figura 13. Issue 3.

### Resolución:

Es un error común comparar Strings Con “==” como hacemos con otros tipos de datos. Pero en String no comprueba que se trate de dos hilos iguales sino que compara la ubicación en memoria, por lo que deberíamos usar .equals() para evitar errores.

```
41 public Film save(Film film) {
42     if (film.getTitle().equals("")) {
43         throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "The title is empty");
44     }
45     urlUtils.checkValidImageUrl(film.getUrl());
46     Film newFilm = repository.save(film);
47     notificationService.notify("Film Event: Film with title="+newFilm.getTitle()+" was created");
48     return newFilm;
49 }
```

Figura 14. Issue 3 resuelto.

## Issue 4

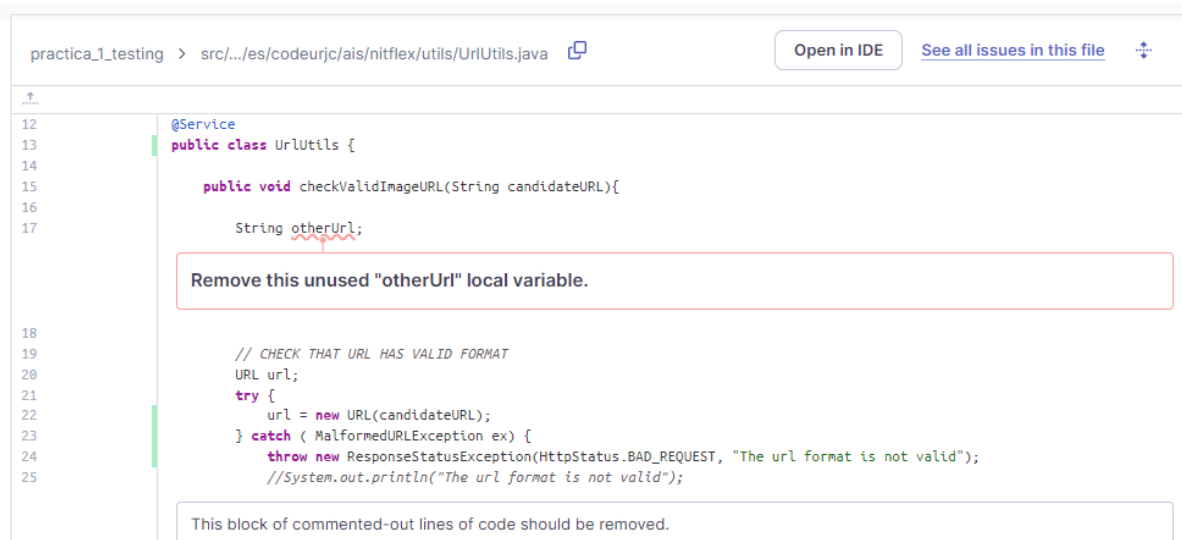


Figura 15. Issue 4.

### Resolución:

Sonarqube nos indica una variable inutilizada. Esto puede suponer un problema para entender el código, lo que dificulta el mantenimiento y la modificación. También hay algunos compiladores que reservan memoria para la variable aunque no se use. Por lo que eliminaremos la variable.

```
12 @Service
13 public class UrlUtils {
14
15     public void checkValidImageUrl(String candidateURL){
16         // CHECK THAT URL HAS VALID FORMAT
17         URL url;
18         try {
19             url = new URL(candidateURL);
20         } catch ( MalformedURLException ex) {
21             throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "The url format is not valid");
22         }
```

Figura 16. Issue 4 resuelto.

## Issue 5



Figura 17. Issue 5.

### Resolución:

Mantener código comentado no afecta al rendimiento ni la ejecución del programa. El problema que supone es que es innecesario y lo hace más complicado de leer lo que puede dificultar el mantenimiento y modificación. Eliminaremos el comentario.

```
16
17 // CHECK THAT URL HAS VALID FORMAT
18 URL url;
19 try {
20     url = new URL(candidateURL);
21 } catch ( MalformedURLException ex) {
22     throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "The url format is not valid");
23 }
24
25 // CHECK THAT THE URL IS AN IMAGE
26 if(!candidateURL.matches(".*\\. (jpg|jpeg|gif|png)")){
27     throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "The url is not an image resource");
28 }
```

Figura 18. Issue 5 resuelto.

## Issue 6

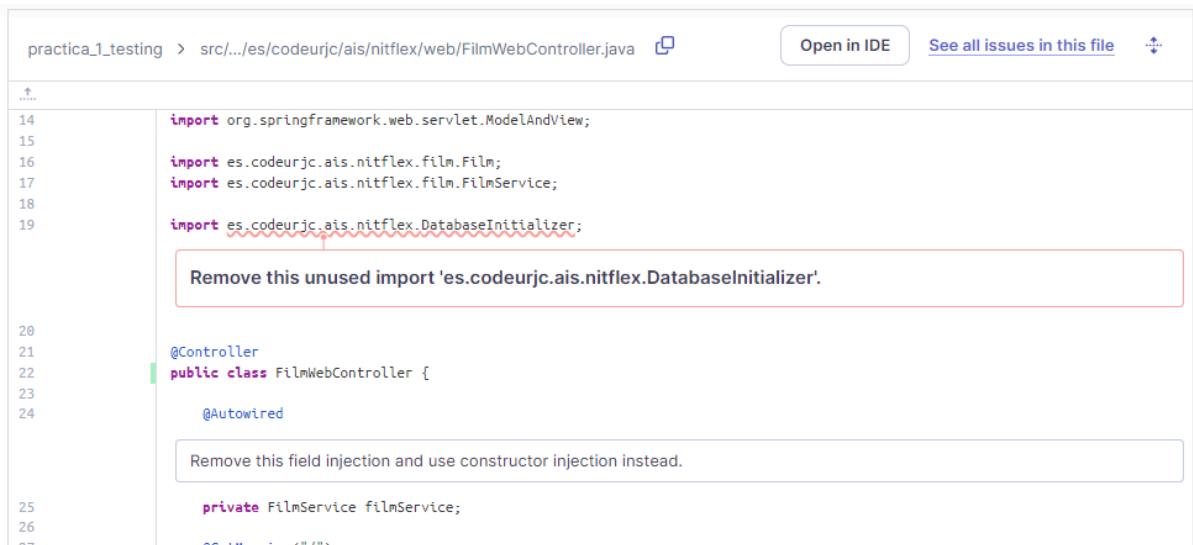


Figura 19. Issue 6.

### Resolución:

De nuevo es un issue que indica código que no se utiliza. La solución es eliminarlo para hacer el código más legible y evitar problemas de este tipo.

```
3 import java.util.Optional;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.crossstore.ChangeSetPersister.NotFoundException;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.validation.BindException;
10 import org.springframework.web.bind.annotation.ExceptionHandler;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.PathVariable;
13 import org.springframework.web.bind.annotation.PostMapping;
14 import org.springframework.web.server.ResponseStatusException;
15 import org.springframework.web.servlet.ModelAndView;
16
17 import es.codeurjc.ais.nitflex.film.Film;
18 import es.codeurjc.ais.nitflex.film.FilmService;
19
20 @Controller
21 public class FilmWebController {
22
```

Figura 20. Issue 6 resuelto.

## Issue 7

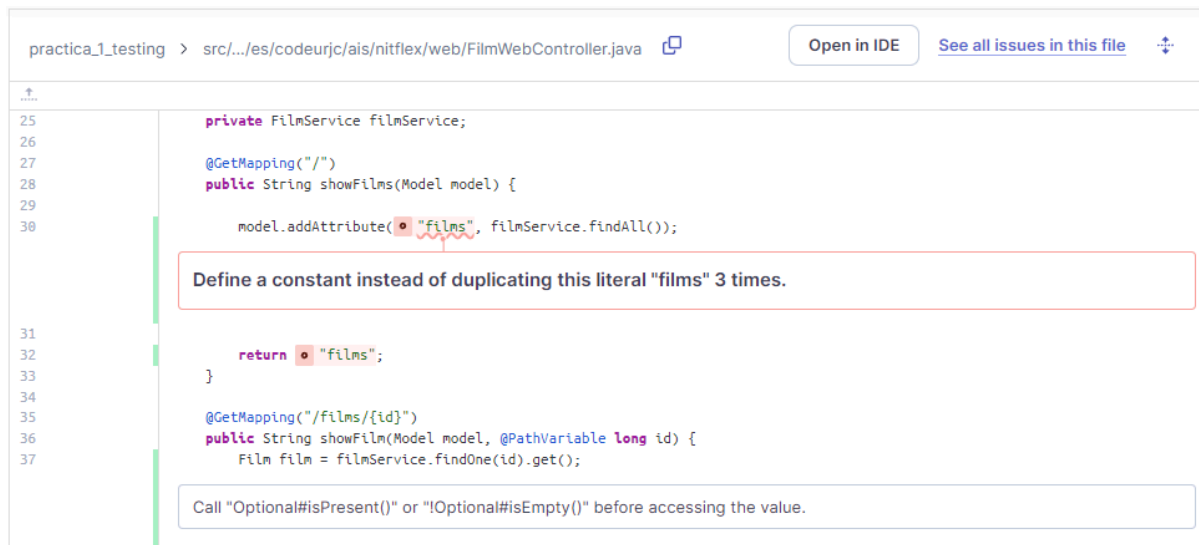


Figura 21. Issue 7.

### Resolución:

Se trata de un falso positivo. En este caso los literales no son varias instancias de un mismo dato, sino que se utiliza por Spring para otras funciones.

El issue aparece otra vez con el string "message" en FilmWebController.

## Issue 8

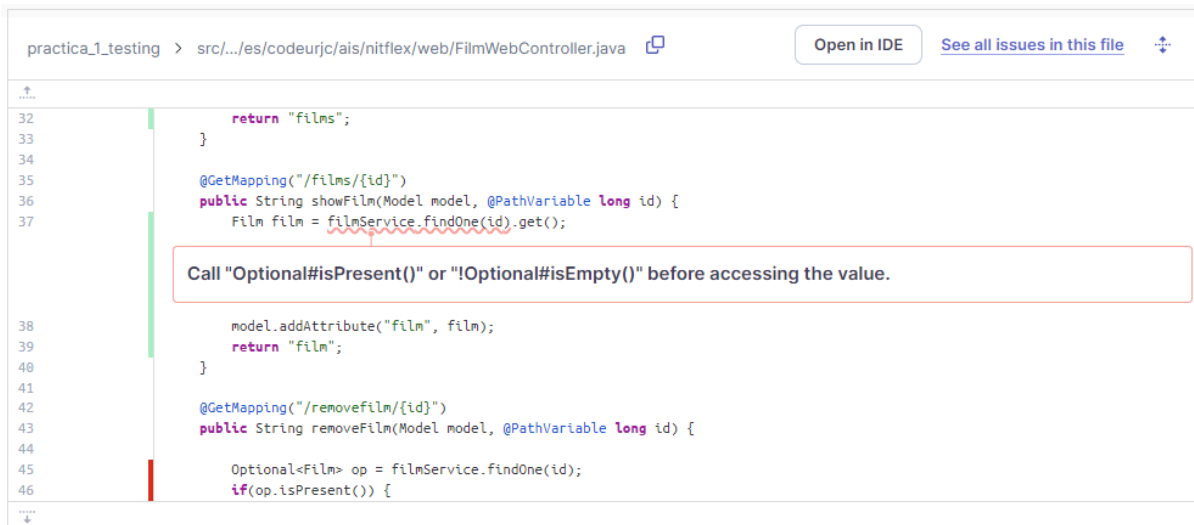


Figura 22. Issue 8.

### Resolución:

En este caso la posición en id puede no contener ningún valor. Sonarqube nos indica que antes de usar `get()` se debe comprobar que no esté vacío. Podemos provocar que se cause un error si está vacía la variable.

```
37
38 @GetMapping("/films/{id}")
39 public String showFilm(Model model, @PathVariable long id) throws NotFoundException {
40     Optional<Film> optionalFilm = filmService.findOne(id);
41
42     if (optionalFilm.isEmpty()) {
43         throw new NotFoundException();
44     }
45
46     Film film = optionalFilm.get();
47
48     model.addAttribute("film", film);
49     return "film";
50 }
51
```

Figura 23. Issue 8 resuelto.

## Issue 9

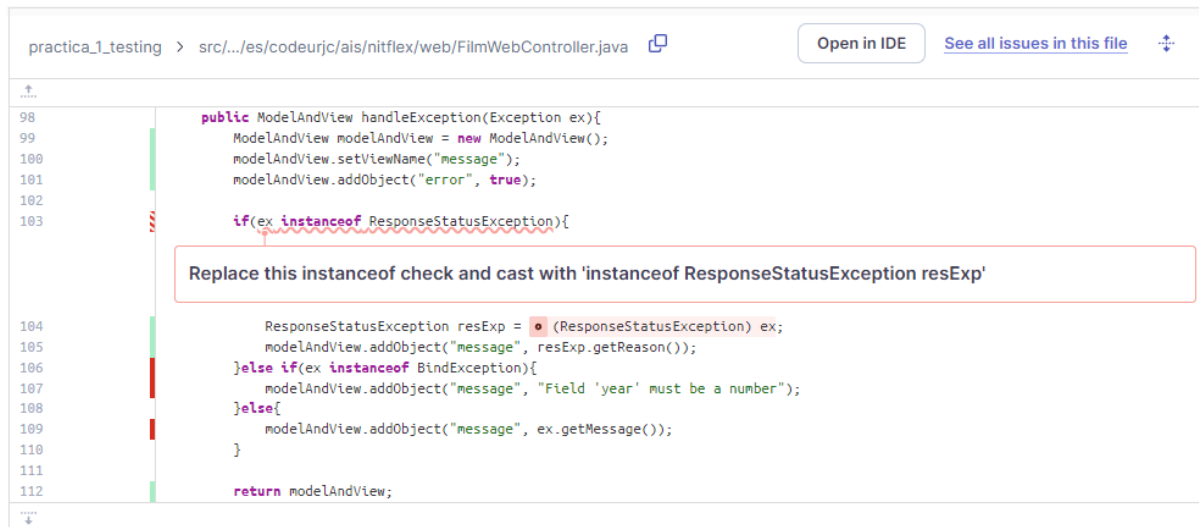


Figura 24. Issue 9.

### Resolución:

Sonarqube indica que deberíamos actualizar el código ya que desde java 16 se añade la función de “Pattern Matching for instanceof”. Esta función reduciría el código y lo haría más legible.

Podría ser un falso positivo, ya que el código es funcional y legible como lo tenemos. Además genera una dependencia de java 16. En caso de no disponer de esta versión de java o una superior sería un falso positivo ya que el código es correcto y no habría posibilidad de actualizarlo.

En mi equipo tengo una versión de java compatible por lo que realizaré el cambio para mejorar el código.

```
104
105 // Cuando se produce una excepción 'ResponseStatusException' se ejecuta este método
106 // -> Devuelve una vista con un mensaje
107 @ExceptionHandler({ResponseStatusException.class, BindException.class})
108 public ModelAndView handleException(Exception ex){
109     ModelAndView modelAndView = new ModelAndView();
110     modelAndView.setViewName("message");
111     modelAndView.addObject("error", true);
112
113     if(ex instanceof ResponseStatusException resExp){
114         modelAndView.addObject("message", resExp.getReason());
115     }else if(ex instanceof BindException){
116         modelAndView.addObject("message", "Field 'year' must be a number");
117     }else{
118         modelAndView.addObject("message", ex.getMessage());
119     }
120
121     return modelAndView;
122 }
123
```

Figura 25. Issue 9 resuelto.

## Issue 10



Figura 26. Issue 10.

### Resolución:

Desde JUnit5 no hace falta que las clases de test ni las funciones sean públicas, por lo que en el manual de usuario se recomienda dejarlos en default (No hace falta indicar la visibilidad de la clase o función).

Desde el archivo pom.xml se puede ver que tengo la versión 5 de JUnit por lo que realizaré el cambio. De esta forma el código será convencional, lo que mejorará la legibilidad.

Si sonarqube indicara que hay que realizar este cambio en un proyecto que usa una versión anterior de JUnit sería un falso positivo, ya que no es posible realizar ese cambio.

Este issue se ve en las tres clases de test y los dos tests que contiene cada una.

```
18 class ModificacionPeliculasTest{
19
20     @LocalServerPort
21     int port;
22
23     private WebDriver driver;
24     private String title, url, fecha, sinopsis;
25
26     @BeforeEach
27     void setup() {
28         driver = new ChromeDriver();
29         driver.get("http://localhost:"+this.port+"/");
30
31         title = "Pulp Fiction";
32         fecha = "1995";
33         sinopsis = "Película de Quentin Tarantino";
34     }
35 }
```

Figura 27. Issue 10 resuelto.



# OVERVIEW CORREGIDO

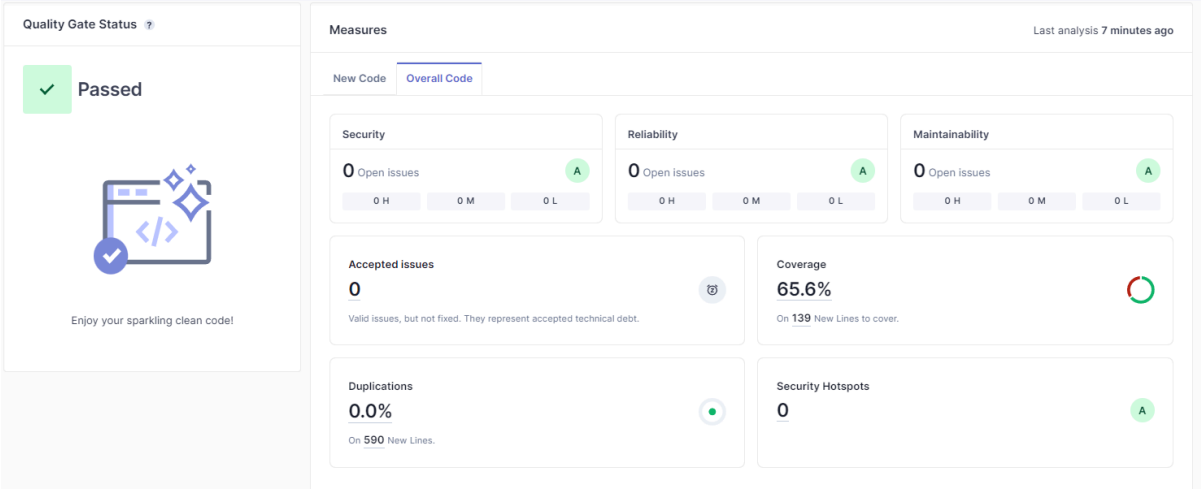


Figura 28. Overview Final

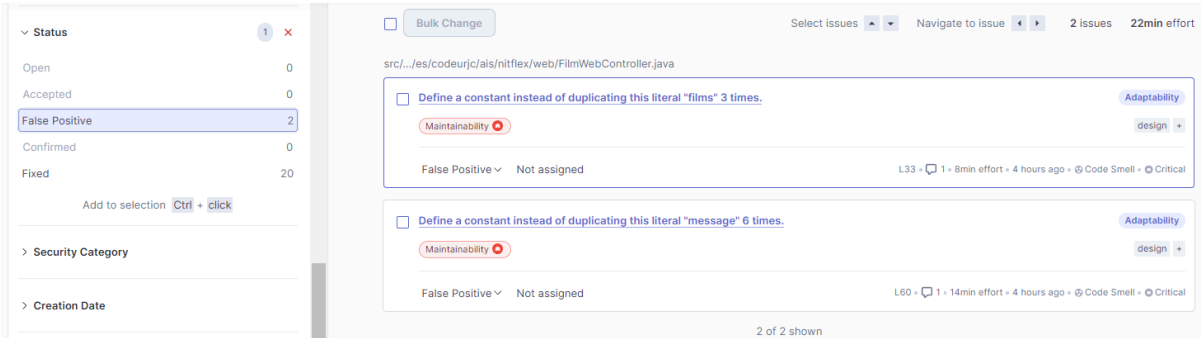


Figura 29. Issues solucionadas con false positive.