

Practica 2 AIS

Autores: Raúl Sánchez Benítez, Andrés Muñoz Muñoz.

Asignatura: AMPLIACION DE INGENIERIA DEL SOFTWARE

Curso: 2024-2025

Índice

| | |
|---|----|
| Introducción | 3 |
| Captura de pantalla del Dashboard | 3 |
| Corrección de errores..... | 4 |
| Errores de Tipo Blocker..... | 4 |
| Errores de Tipo High | 5 |
| Errores de Tipo Medium..... | 8 |
| Vista Previa Final..... | 19 |

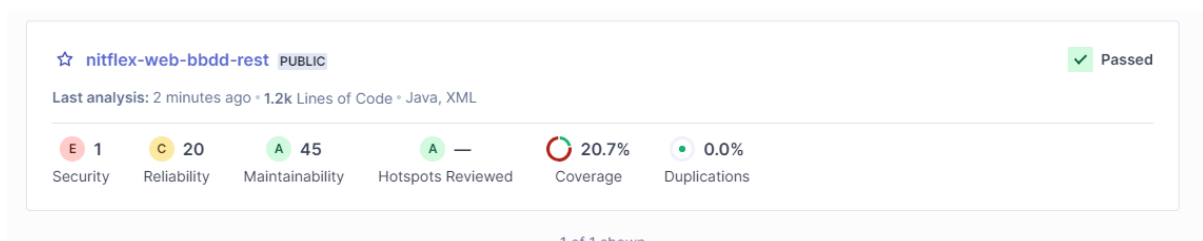
Introducción

En esta parte de la práctica, se ha realizado un análisis del código de la parte anterior, junto con el de la primera práctica, utilizando SonarQube para identificar y corregir errores de calidad y seguridad.

A lo largo del documento, se presentan los errores detectados, sus soluciones y el impacto de las correcciones en el proyecto.

Captura de pantalla del Dashboard

Aquí mostramos la captura de pantalla del dashboard:



Corrección de errores

Vamos a empezar a corregir los issues:

Errores de Tipo Blocker

Empecemos con los errores de tipo blocker:

pom.xml

☐ [Make sure this SonarQube token gets revoked, changed, and removed from the code.](#)

Responsibility

Security

cwe +

☐ Open

☐ Not assigned

L24 • 30min effort • 15 hours ago

1 of 1 shown

Solamente tenemos este error de seguridad, pero no podemos corregirlo debido a que nos pide borrar el token de sonar, pero lo necesitamos para poder ejecutar el proyecto y que sonar lo pueda analizar. Lo voy a marcar como un falso positivo porque es algo necesario para el análisis.

Errores de Tipo High

Pasemos a los errores de tipo High:

En SonarCube nos marca que tenemos 8, vamos a verlos:

src/.../web/nitflex/configuration/RestErrorHandler.java

☐ Remove usage of generic wildcard type. Intentionality

Maintainability

☐ Open ☐ Not assigned ☐ pitfall +

L21 • 20min effort • 15 hours ago

Remove usage of generic wildcard type.

Generic wildcard types should not be used in return types [java:S1452](#)

Line affected: L21 • Effort: 20min • Introduced: 15 hours ago

☐ Open ☐ Not assigned ☐ pitfall +

Software qualities impacted
Maintainability

Clean code attribute
Intentionality | Not clear

| Where is the issue? | Why is this an issue? | How can I fix it? | Activity | More info |
|---------------------|-----------------------|-------------------|----------|-----------|
|---------------------|-----------------------|-------------------|----------|-----------|

```
20 @ExceptionHandler({ FilmNotFoundException.class, IllegalArgumentException.class, BindException.class })
21 public ResponseEntity<?> handleException(Exception ex) {
22
23     if (ex instanceof MethodArgumentNotValidException manvExp) {
24         return ResponseEntity.badRequest().body(manvExp.getFields().get(0).getDefaultMessage());
25     } else if (ex instanceof FilmNotFoundException fnfExp) {
26         return ResponseEntity.notFound().build();
27     } else {
28         return ResponseEntity.badRequest().body(ex.getMessage());
29     }
30 }
```

Remove usage of generic wildcard type.

Nos da error porque devuelve un tipo de datos genérico, y esto se soluciona cambiándolo por String:

```
@ExceptionHandler({ FilmNotFoundException.class, IllegalArgumentException.class, BindException.class })
public ResponseEntity<String> handleException(Exception ex) {
    if (ex instanceof MethodArgumentNotValidException manvExp) {
        return ResponseEntity.badRequest().body(manvExp.getFields().get(0).getDefaultMessage());
    } else if (ex instanceof FilmNotFoundException fnfExp) {
        return ResponseEntity.notFound().build();
    } else {
        return ResponseEntity.badRequest().body(ex.getMessage());
    }
}
```

src/.../web/nitflex/configuration/WebErrorHandler.java

☐ Define a constant instead of duplicating this literal "message" 3 times. Adaptability

Maintainability design +

☐ Open ☐ Not assigned L22 • 8min effort • 15 hours ago

Define a constant instead of duplicating this literal "message" 3 times. 🔗

String literals should not be duplicated [java:S1192](#) 🔗

Line affected: L22 • Effort: 8min • Introduced: 15 hours ago

☐ Open ☐ Not assigned design +

Software qualities impacted
Maintainability 🔴

Clean code attribute
Adaptability | Not distinct

| Where is the issue? | Why is this an issue? | How can I fix it? | Activity |
|---------------------|-----------------------|-------------------|----------|
|---------------------|-----------------------|-------------------|----------|

```
20 public ModelAndView handleException(Exception ex){
21     ModelAndView modelAndView = new ModelAndView();
22     modelAndView.setViewName("message");
23
24     modelAndView.addObject("error", true);
25
26     if(ex instanceof MethodArgumentNotValidException manvExp){
27         modelAndView.addObject("message", manvExp.getFieldError().getDefaultMessage());
28     }else{
29         modelAndView.addObject("message", ex.getMessage());
30     }
```

Define a constant instead of duplicating this literal "message" 3 times.

Este error nos dice que, en vez de duplicarlo, creamos una variable.

Para ello creamos una variable que evite los duplicados.

```
@ExceptionHandler({FilmNotFoundException.class, IllegalArgumentException.class, BindException.class})
public ModelAndView handleException(Exception ex){
    ModelAndView modelAndView = new ModelAndView();
    String a = "message";
    modelAndView.setViewName(a);
    modelAndView.addObject("error", true);

    if(ex instanceof MethodArgumentNotValidException manvExp){
        modelAndView.addObject(a, manvExp.getFieldError().getDefaultMessage());
    }else{
        modelAndView.addObject(a, ex.getMessage());
    }

    return modelAndView;
}
```

☐ Define a constant instead of duplicating this literal "Film not found" 3 times. Adaptability

Maintainability design +

☐ Open ☐ Not assigned L64 • 8min effort • 16 hours ago

Define a constant instead of duplicating this literal "Film not found" 3 times. [🔗](#)

String literals should not be duplicated [java:S1192](#) [🔗](#)

Line affected: L64 • Effort: 8min • Introduced: 16 hours ago

☐ Open ☐ Not assigned ☐ design [+](#)

Where is the issue? Why is this an issue? How can I fix it? Activity

```

59         FilmDTO film = op.get();
60         model.addAttribute("film", film);
61         model.addAttribute("isInFavorites", favoriteFilmService.isFavorite(film));
62         return "film";
63     }else {
64         throw new RuntimeException(HttpStatus.NOT_FOUND, "Film not found");
65     }
66
67

```

Define a constant instead of duplicating this literal "Film not found" 3 times.

Aquí se nos marca el mismo error que el anterior. Para corregirlo, hemos creado una variable que guarde el mensaje y lo hemos sustituido.

src/.../web/nitflex/controller/web/FilmWebController.java

☐ Define a constant instead of duplicating this literal "error" 3 times. [Adaptability](#)
Maintainability [design](#) [+](#)
☐ Open ☐ Not assigned L78 • 8min effort • 16 hours ago

☐ Define a constant instead of duplicating this literal "action" 4 times. [Adaptability](#)
Maintainability [design](#) [+](#)
☐ Open ☐ Not assigned L89 • 10min effort • 16 hours ago

☐ Define a constant instead of duplicating this literal "filmForm" 4 times. [Adaptability](#)
Maintainability [design](#) [+](#)
☐ Open ☐ Not assigned L92 • 10min effort • 16 hours ago

☐ Define a constant instead of duplicating this literal "redirect:/films/" 6 times. [Adaptability](#)
Maintainability [design](#) [+](#)
☐ Open ☐ Not assigned L110 • 14min effort • 16 hours ago

Aquí se nos muestran otros 4 errores más de este mismo tipo.

```

private String filmNotFound = "Film not found";
private String error = "error";
private String action = "action";
private String filmForm = "filmForm";
private String redirectFilms = "redirect:/films/";

```

Con esto finalizaríamos la corrección de errores de tipo High y vamos a pasar a los errores de tipo Medium.

Errores de Tipo Medium

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

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability Maintainability

☐ Open ☐ Not assigned

L21 • 5min effort • 17 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability Maintainability

☐ Open ☐ Not assigned

L24 • 5min effort • 17 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability Maintainability

☐ Open ☐ Not assigned

L27 • 5min effort • 17 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability Maintainability

☐ Open ☐ Not assigned

L30 • 5min effort • 17 hours ago

Vamos a empezar por estos que nos piden que borremos la anotación `@Autowired` y usemos una inyección por constructor:


```

private final FilmService filmService;
private final ReviewService reviewService;
private final UserRepository userRepository;
private final ImageUtils imageUtils;

public DatabaseInitializer(FilmService filmService,
ReviewService reviewService,
UserRepository userRepository,
ImageUtils imageUtils) {
    this.filmService = filmService;
    this.reviewService = reviewService;
    this.userRepository = userRepository;
    this.imageUtils = imageUtils;
}

```

Aquí hemos borrado la anotación `@Autowired` y lo hemos sustituido por este fragmento de código, porque SonarQube pide cambiarlo en variables por inyección en el constructor para que el código sea más seguro, fácil de probar y claro.

src/.../web/nitflex/configuration/RestErrorHandler.java

☐ [Replace the chain of if/else with a switch expression.](#)
Intentionality

Maintainability

Open

Not assigned

L22 • 10min effort • 18 hours ago

☐ [A "NullPointerException" could be thrown; "getFieldError\(\)" can return null.](#)
Intentionality

Reliability

Open

Not assigned

cwe symbolic-execution ... +

L23 • 10min effort • 18 hours ago

Estos dos errores me piden que use un switch y que arregle una `NullPointerException`.

Con este fragmento de código hemos arreglado ambos problemas, con el switch todo queda más claro y estructurado, y con el `@SuppressWarnings` evitamos la excepción de puntero nulo.

```

@SuppressWarnings("null")
@ExceptionHandler({ FilmNotFoundException.class, IllegalArgumentException.class, BindException.class })
public ResponseEntity<String> handleException(Exception ex) {
    switch (ex){
        case MethodArgumentNotValidException manvExp:
            return ResponseEntity.badRequest().body(manvExp.getFieldError().getDefaultMessage());
        case FilmNotFoundException fnfExp:
            return ResponseEntity.notFound().build();
        default:
            return ResponseEntity.badRequest().body(ex.getMessage());
    }
}

```

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

☐ Remove the declaration of thrown exception 'java.io.IOException', as it cannot be thrown from method's body.
 Intentionality

Maintainability
error-handling
unused
...
+

☐ Open
 ☐ Not assigned

L88 • 5min effort • 18 hours ago

Nos vuelve a pedir que quitemos @Autowired, así que vamos a usar la misma metodología para solucionarlo.

```

private final UserComponent userComponent;

public UserModelAttributes(UserComponent userComponent) {
    this.userComponent = userComponent;
}

```

src/.../web/nitflex/configuration/WebErrorHandler.java

☐ A "NullPointerException" could be thrown; "getFieldError()" can return null.
 Intentionality

Reliability
cwe
symbolic-execution
...
+

☐ Open
 ☐ Not assigned

L28 • 10min effort • 2 hours ago

Aquí se nos vuelve a presentar el problema con NullPointerException, así que vamos a volver a aplicar la misma solución de antes, es decir, añadir @SuppressWarnings, sabemos que no es la más óptima porque realmente no soluciona el problema, pero así SonarCube deja de marcar el error.

src/.../web/nitflex/controller/rest/FilmRestController.java

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability ⬇

Maintainability ⬇

No tags +

☐ Open ▾

Not assigned ▾

L31 • 5min effort • 18 hours ago

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability ⬇

Maintainability ⬇

No tags +

☐ Open ▾

Not assigned ▾

L34 • 5min effort • 18 hours ago

Aquí se nos vuelve a pedir quitar la anotación `@Autowired`, así que vamos a volver a aplicar la misma metodología.

```
private final FilmService filmService;
private final ReviewService reviewService;

public FilmRestController(FilmService filmService,
    ReviewService reviewService) {
    this.filmService = filmService;
    this.reviewService = reviewService;
}
```

src/.../web/nitflex/controller/rest/UserRestController.java

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability ⬇

Maintainability ⬇

No tags +

☐ Open ▾

Not assigned ▾

L19 • 5min effort • 18 hours ago

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability ⬇

Maintainability ⬇

No tags +

☐ Open ▾

Not assigned ▾

L22 • 5min effort • 18 hours ago

Otra vez lo mismo de quitar `@Autowired`.

```
private final UserService userService;
private final UserMapper userMapper;

public UserRestController(UserService userService,
    UserMapper userMapper) {
    this.userService = userService;
    this.userMapper = userMapper;
}
```

src/.../web/nitflex/controller/web/FilmWebController.java

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability

Maintainability

No tags

☐ Open ☐ Not assigned

L37 • 5min effort • 18 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability

Maintainability

No tags

☐ Open ☐ Not assigned

L40 • 5min effort • 18 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability

Maintainability

No tags

☐ Open ☐ Not assigned

L43 • 5min effort • 18 hours ago

Se vuelve a pedir quitar @Autowired otra vez.

```
private final FilmService filmService;
private final FavoriteFilmService favoriteFilmService;
private final ReviewService reviewService;

public FilmWebController(FilmService filmService,
FavoriteFilmService favoriteFilmService,
ReviewService reviewService) {
    this.filmService = filmService;
    this.favoriteFilmService = favoriteFilmService;
    this.reviewService = reviewService;
}
```

src/.../web/nitflex/controller/web/UserWebController.java

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability

Maintainability

No tags

☐ Open ☐ Not assigned

L17 • 5min effort • 18 hours ago

☐ [Remove this field injection and use constructor injection instead.](#) Consistency

Reliability

Maintainability

No tags

☐ Open ☐ Not assigned

L20 • 5min effort • 18 hours ago

Se vuelve a pedir quitar @Autowired otra vez.

```
private final UserComponent userComponent;
private final UserService userService;

public UserWebController(UserComponent userComponent,
UserService userService) {
    this.userComponent = userComponent;
    this.userService = userService;
}
```

src/.../es/codeurjc/web/nitflex/model/Review.java

☐ A "NullPointerException" could be thrown; "film" is nullable here.

Intentionality

Reliability

cwe symbolic-execution ... +

☐ Open ☐ Not assigned

L111 • 10min effort • 18 hours ago

Aquí se nos muestra que el método puede devolver un NullPointerException porque film puede ser null, para solucionar esto vamos a usar este fragmento de código:

```
return ((film == null) && (review.film == null)) || ((film != null) && (film.equals(review.film)));
```

Hemos añadido (film != null) para evitar el NullPointerException.

src/.../codeurjc/web/nitflex/service/FilmService.java

☐ Define and throw a dedicated exception instead of using a generic one.

Intentionality

Maintainability

cwe error-handling ... +

☐ Open ☐ Not assigned

L53 • 20min effort • 19 hours ago

Aquí se nos pide usar una excepción dedicada en vez de una tan genérica. Lo hemos arreglado usando una SQLException en lugar de la RuntimeException que se estaba usando.

```

public InputStream getPosterFile(long id) throws SQLException {
    Film film = filmRepository.findById(id)
        .orElseThrow(() -> new FilmNotFoundException(id));
    Blob blob = film.getPosterFile();
    try {
        return blob.getBinaryStream();
    } catch (SQLException e) {
        throw new SQLException(reason:"Error getting image from database", e);
    }
}

```

src/.../codeurjc/web/nitflex/service/UserComponent.java

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability

Maintainability

No tags +

☐ Open ☐ Not assigned

L12 • 5min effort • 19 hours ago

Otra vez nos pide que quitemos @Autowired.

```

private final UserRepository userRepository;

public UserComponent(UserRepository userRepository) {
    this.userRepository = userRepository;
}

```

src/.../codeurjc/web/nitflex/service/UserService.java

☐ [Remove this field injection and use constructor injection instead.](#)

Consistency

Reliability

Maintainability

No tags +

☐ Open ☐ Not assigned

L16 • 5min effort • 19 hours ago

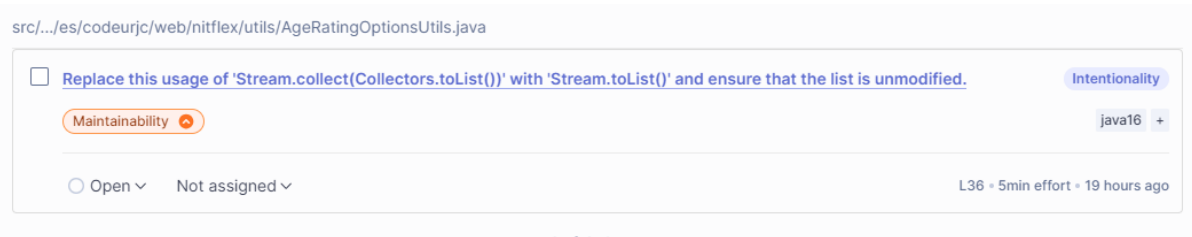
Se vuelve a pedir quitar @Autowired otra vez.

```

private final UserRepository userRepository;

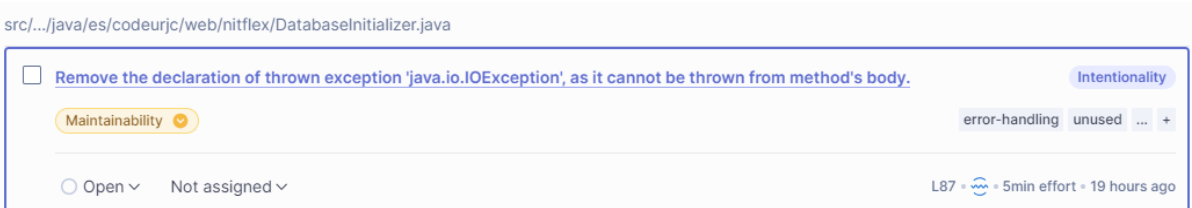
public UserService(UserRepository userRepository) {
    this.userRepository = userRepository;
}

```



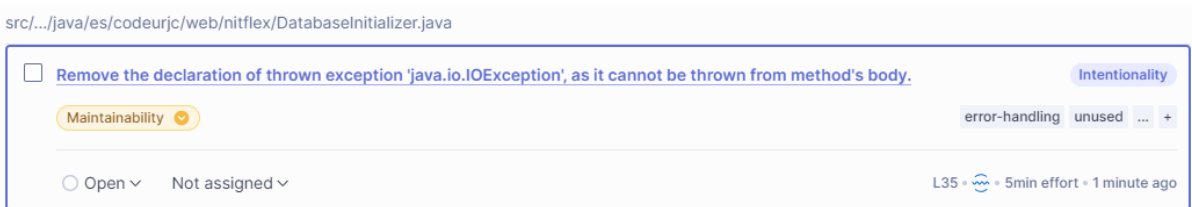
Este error nos dice que cambiemos nuestro código para evitar el uso de `.collect(Collectors.toList())`. Esto lo hemos solucionado cambiando el código para añadir `.toList()`

```
public static Collection<AgeRatingOption> getAgeRatingOptions(String selectedAgeRating) {  
    return Stream.of(AgeRating.values())  
        .map(  
            ageRating -> new AgeRatingOption(  
                ageRating.getDescription(),  
                ageRating.getDescription().equals(selectedAgeRating)  
            ))  
        .toList();  
}
```



Este issue nos decía que quitásemos el `throws` porque el método no era capaz de lanzarlo, así que hemos borrado el `throws` porque no servía de nada porque en el cuerpo de la función nunca se lanzaba una excepción.

```
private FilmDTO saveFilmWithURLImage(CreateFilmRequest film, String localpath){  
    return filmService.save(film, imageUtils.localImageToBlob(localpath));  
}
```



Hemos quitado el `throws IOException` por el mismo motivo que en el issue anterior.

```
@PostConstruct  
public void init(){  
}
```

src/.../web/nitflex/controller/web/FilmWebController.java

☐ [Set a HttpStatus code reflective of the operation.](#)

Adaptability

Reliability

spring best-practice +

☐ Open ☐ Not assigned

L165 • 5min effort • 19 hours ago

```
@GetMapping("/films/{id}/poster")
public ResponseEntity<Object> getMethodName(@PathVariable long id) throws IOException {
    Resource poster;
    try {
        poster = new InputStreamResource(filmService.getPosterFile(id));
    } catch (Exception e) {
        ClassPathResource resource = new ClassPathResource(path:"static/images/no-image.png");
        byte[] imageBytes = resource.getInputStream().readAllBytes();
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .header(HttpHeaders.CONTENT_TYPE, ...headerValues:"image/jpeg").body(imageBytes);
    }
    return ResponseEntity.ok().header(HttpHeaders.CONTENT_TYPE, ...headerValues:"image/jpeg").body(poster);
}
```

Para este error se nos pedía que el mensaje de httpStatus fuese el adecuado, y hemos cambiado de .ok() (200) a NOT_FOUND (404) para que refleje correctamente que no se encontró la imagen q buscaba.

src/.../es/codeurjc/web/nitflex/model/Film.java

☐ [Replace this instanceof check and cast with 'instanceof Film f'](#)

Intentionality

Maintainability

java16 +

☐ Open ☐ Not assigned

L147 • 5min effort • 19 hours ago

```
@Override
public boolean equals(Object o){
    if(o == null) return false;
    if(o == this) return true;
    if(o instanceof Film){
        Film f = (Film) o;
        return f.getId().equals(this.id);
    }
    return false;
}
```

Esta sería la solución:


```
@Override
public boolean equals(Object o){
    if(o == null) return false;
    if(o == this) return true;
    if(o instanceof Film f){
        return f.getId().equals(this.id);
    }
    return false;
}
```

src/.../es/codeurjc/web/nitflex/model/Review.java

☐ Rename this field "created_at" to match the regular expression `^[a-z][a-zA-Z0-9]*$`.

Consistency

Maintainability

convention +

☐ Open ☐ Not assigned

L39 • 2min effort • 19 hours ago

Aquí se nos pide que cambiemos el nombre de una variable que usa '_'. Entonces pasa de ser `created_at` a `createdAt`.

```
@CreationTimestamp
private Date createdAt;
```

☐ Rename this method name to match the regular expression `^[a-z][a-zA-Z0-9]*$`.

Consistency

Maintainability

convention +

☐ Open ☐ Not assigned

L75 • 5min effort • 19 hours ago

Este es el mismo error anterior, el nombre del método usaba '_' y lo hemos cambiado.

```
public Date getcreatedAt() {
    return createdAt;
}
```

src/.../es/codeurjc/web/nitflex/e2e/e2eTests.java

☐ Rename class "e2eTests" to match the regular expression: `^((Test|IT)[a-zA-Z0-9_]+|[A-Z][a-zA-Z0-9_]*(Test|Tests|TestCase|IT|ITCase))$`

Consistency

Maintainability

convention tests +

☐ Open ☐ Not assigned

L20 • 5min effort • 19 hours ago

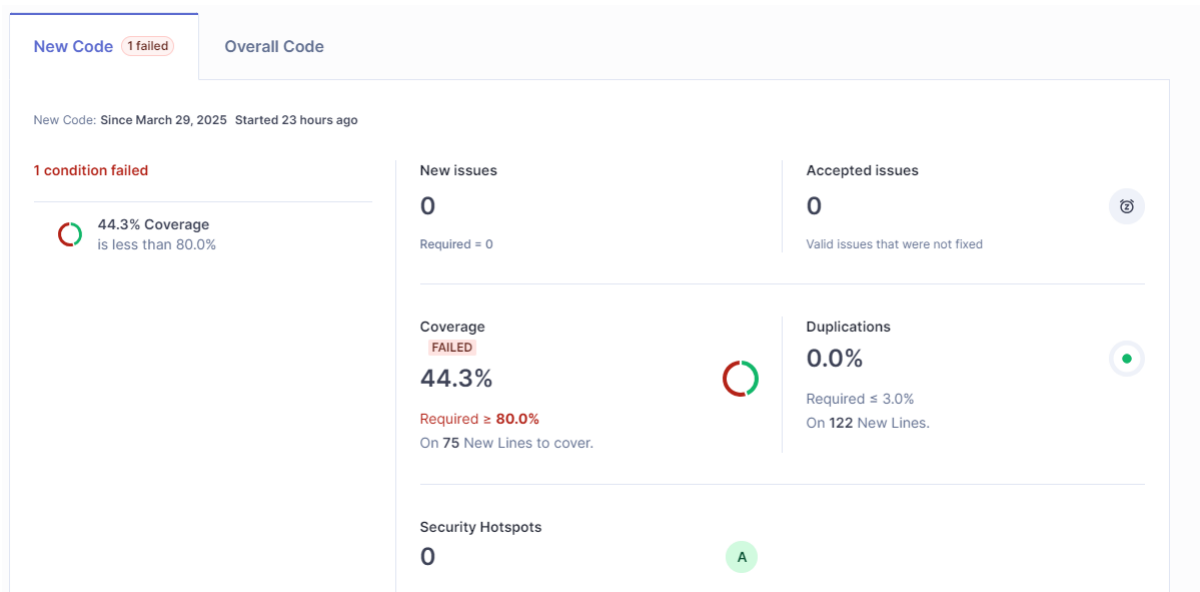
6 of 6 shown

Este error pide que renombramos la clase `e2eTests`, pondremos `EndToEndTests`.

```
@SpringBootTest
public class EndToEndTests {
    ⬤
```

Vista Previa Final

Tras haber arreglado todos los errores, este es el dashboard en la pestaña New Code



Y en la vista previa de los proyectos se ve así

