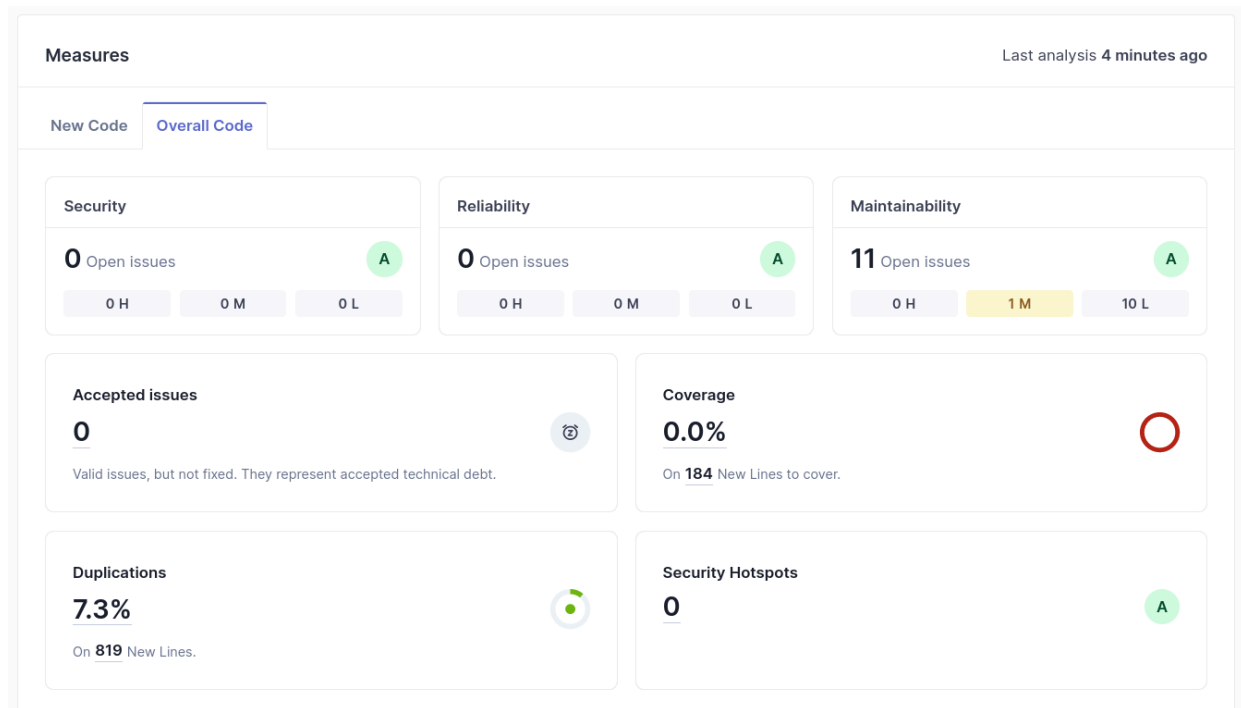


Informe reporte análisis estático

Repositorio seleccionado:

<https://github.com/training-practice-sofkau/ddd-demo>



Utilizando la herramienta SonarQube para el análisis estático del código, se encontraron los siguientes resultados:

- El proyecto es aprobado por las Quality Gate standard, lo que significa que el código no presenta mayores riesgos para ser desplegado al entorno de producción.
- Se encontraron **11** issues de mantenibilidad, de las cuales solo 1 tiene un impacto medio y las restantes con impacto bajo.
- El proyecto cuenta con una duplicidad de código de un 7.3% de las 819 líneas, lo cual sugiere realizar una refactorización de la lógica para reducir el riesgo de errores, tener un código más conciso y mantenible.
- No se encontraron riesgos de seguridad en este análisis.

Análisis de las Issues encontradas

La issue de impacto medio se debe a realizar una aserción de que se lance una excepción, pero dentro de la lambda se usan varios métodos que pueden lanzar excepciones de tipo Runtime.

The screenshot shows a code editor with a tab labeled 'alzate...'. The code is in Kotlin and includes a lambda function. A red box highlights a suggestion to refactor the lambda code to have only one invocation possibly throwing a runtime exception.

```
69  Mockito.when(service.enviarMensajeAConductor(Mockito.any(ConductorId.class), Mockito.anyString())).thenReturn(false);
70
71  usecase.addRepository(repository);
72  usecase.addServiceBuilder(new ServiceBuilder().addService(service));
73
74  var mensaje = Assertions.assertThrows(BusinessException.class, () -> {
75
76      UseCaseHandler.getInstance()
77          .setIdentifyExecutor("xxxxx")
78          .syncExecutor(usecase, new TriggeredEvent<>(event));
79  }).getMessage();
```

Refactor the code of the lambda to have only one invocation possibly throwing a runtime exception.

Las demás issues son de bajo impacto como comentarios con TODOs, clases vacías, o refactorización de estilo de programación.