



# Software Architecture

Lab. 08

TDD: Test-driven development

Cobertura de código(SonarCloud)

Integración continua (GitHub Actions)

Herramientas para el análisis estático (SonarCloud)

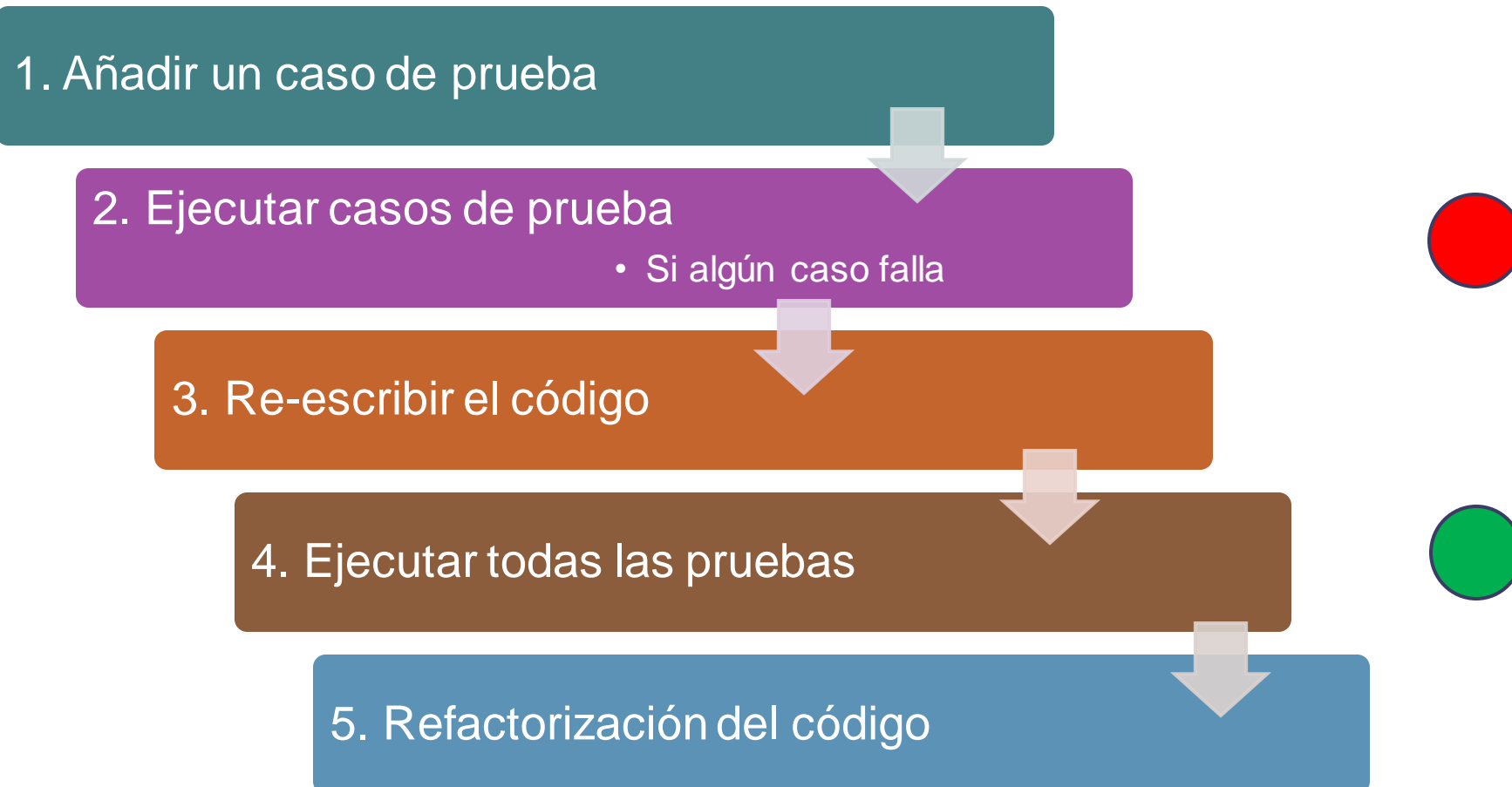
2022-23

Jose Emilio Labra Gayo  
Pablo González  
Irene Cid  
Cristian Augusto

# TDD - Introducción

- Proceso de desarrollo de código donde los requisitos se convierten en casos de test que se pueden probar
- Surge como respuesta al desarrollo de código donde los test se dejaban en la fase final tras el desarrollo.
- Técnica propuesta por Kent Beck

# TDD - Fases



# TDD - Características

- Código sencillo que satisface las necesidades del cliente
- Obtenemos código sencillo
- ....Y nuestra batería de pruebas
- Nos ayuda a centrarnos en lo que queremos desarrollar

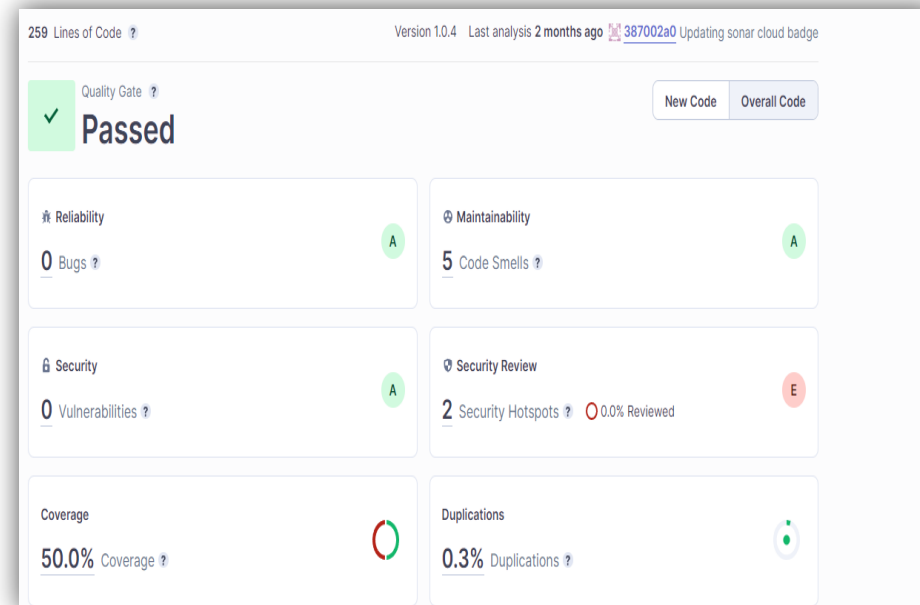
# SonarCloud - Coverage

- Herramienta que incluye la cobertura de código como una métrica más en la evaluación del código
- Cobertura de código: Medida que nos indica la proporción de líneas de código que son probadas en alguno de nuestros test
- Sonar Cloud recoge los datos de los resultados del lanzamiento de los test, recogiendo los siguientes valores:
  - $LC = \text{Líneas cubiertas ( lines\_to\_cover - uncovered\_lines )}$
  - $EL = \text{Número total de líneas ejecutables ( lines\_to\_cover )}$

# SonarCloud

- La ratio de cobertura es calculado con la siguiente fórmula:
  - $LC/EL$
- Tras la ejecución de test, nos genera un fichero para su posterior análisis

[https://sonarcloud.io/summary/overall?id=Arquisoft\\_lomap\\_???](https://sonarcloud.io/summary/overall?id=Arquisoft_lomap_???)



# TDD - Test de ejemplo

- Comprobación del funcionamiento del componente UserList:

- Creamos una lista de usuarios
- Se la añadimos al componente

Comprobamos que el nombre es renderizado en el componente

```
1  import React from 'react'
2  import { render } from "@testing-library/react";
3  import UserList from "../UserList";
4  import {User} from "../shared/sharedddtypes";
5
6  test('check that the list of users renders properly', async () => {
7      const userList:User[] = [{name: 'Pablo', email: 'gonzalezgpablo@uniovi.es' }];
8      const {getByText} = render(<UserList users={userList}/>);
9      expect(getByText(userList[0].name)).toBeInTheDocument();
10     expect(getByText(userList[0].email)).toBeInTheDocument();
11 });
```

# TDD - Test de ejemplo

- Comprobamos que el componente EmailForm funciona bien:
  - Algunas veces tenemos que moquear parte de la prueba
  - Si no mockeamos en este caso, dependemos de los resultados de la restapi
  - Como se trata de test unitarios debemos eliminar esta dependencia



```
6  jest.mock('../api/api');
7
8  test('check register fail', async () => {
9    jest.spyOn(api, 'addUser').mockImplementation((user:User):Promise<boolean> => Promise.resolve(false))
10   await act(async () => {
11     const {container, getByText} = render(<EmailForm OnUserListChange={()=>{}}/>)
12     const inputName = container.querySelector('input[name="username"]')!;
13     const inputEmail = container.querySelector('input[name="email"]')!;
14     fireEvent.change(inputName, { target: { value: "Pablo" } });
15     fireEvent.change(inputEmail, { target: { value: "gonzalezgpablo@uniovi.es" } });
16     const button = getByText("Accept");
17     fireEvent.click(button);
18   });
19 })
```



# Integración Continua - Definición

- Práctica de desarrollo que promueve la integración del código varias veces al día.
- El lanzamiento del proceso de integración continua es ejecutado cuando se cumple alguna condición
  - Cada vez que se genera una instancia, un push o un pull en el repositorio

# Integración Continua - Mejoras

- Detecta y resuelve problemas de una manera continua
- Siempre una versión disponible
- Ejecución automática de los casos de test
- Monitorización de la calidad de código.
- Despliegue automático
- Monitorización de la calidad de código

# Integración Continua - ejemplos

- Jenkins
- Pipeline
- Hudson
- Apache Continuum
- Travis
- GitHub Actions

# Integración Continua - Usos

- Mantenimiento del código en el repositorio.
- Construcción automática
- Despliegue
- Ejecutar los test en un entorno clonado en los entornos de producción
- Mantener el histórico de las construcciones.

# GitHub Actions

- Permite gestionar la integración continua sobre los proyectos de los repositorios en GitHub
- Gratis para proyectos gratuitos
- La configuración se mantiene en uno o varios ficheros yaml dentro del directorio **.github/workflows** , que podemos localizar en la raíz del directorio

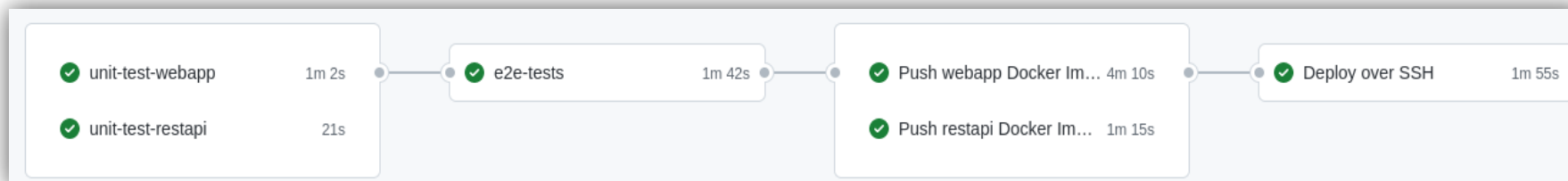
# GitHub Actions

- Contenido .yml :
  - Condiciones que lanzan el proceso (On )
  - Lista de tareas (Jobs)
    - Cada tarea ejecutada en su propio entorno
  - Una especificacion para cada tarea (checkout, install dependencies, build and test)

```
name: CI for LOMAP_0

on:
  release:
    types: [published]

jobs:
  unit-test-webapp:
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: webapp
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm ci
      - run: npm test --coverage --watchAll
      - name: Analyze with SonarCloud
        uses: sonarsource/sonarcloud-github-action@master
        env:
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
          SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```



# GitHub Actions

- Cada tarea debe tener un propósito específico
  - Probar una parte de la app, desplegar, etc.
- Se puede usar para automatizar otras partes del repositorio.
  - Ejemplo: responder automáticamente cuando un nuevo issue es creado.

# GitHub Actions

- *uses: actions/checkout@v2.*  
Uso de una acción ya creada por la comunidad.  
In this case, realiza un checkout de la rama especificada y se la pasa al Runner
- *uses: actions/setup-node@v1*  
*with:*  
    *node-version: 12.14.1*  
    Instala node en el Runner
- *run: npm ci*  
Ejecuta un commando, en este caso instalamos las dependencias del Proyecto vía npm
- *run: npm test*  
Ejecuta las pruebas unitarias. Si alguna falla, la integración continua fallará



# GitHub Actions

- También tenemos jobs para crear imágenes de docker y publicarlás
- Comprueba la [documentation](#) para más configuraciones

```
docker-push-webapp:  
  name: Push webapp Docker Image to GitHub Packages  
  runs-on: ubuntu-latest  
  needs: [e2e-tests]  
  steps:  
    - uses: actions/checkout@v2  
    - name: Publish to Registry  
      uses: elgohr/Publish-Docker-Github-Action@3.04  
      env:  
        API_URI: http://${{ secrets.DEPLOY_HOST }}:5000/api  
      with:  
        name: pglez82/asw2122_0/webapp  
        username: ${{{ github.actor }}}  
        password: ${{{ secrets.DOCKER_PUSH_TOKEN }}}  
        registry: ghcr.io  
        workdir: webapp  
        buildargs: API_URI
```

# Análisis estático del código

- Analiza el código sin compilarlo
- Detecta bugs, code smells, vulnerabilidades del sistema, etc
- Util para medir la calidad del código.
- Se puede bloquear la subida de código que no cumpla con ciertas características de calidad

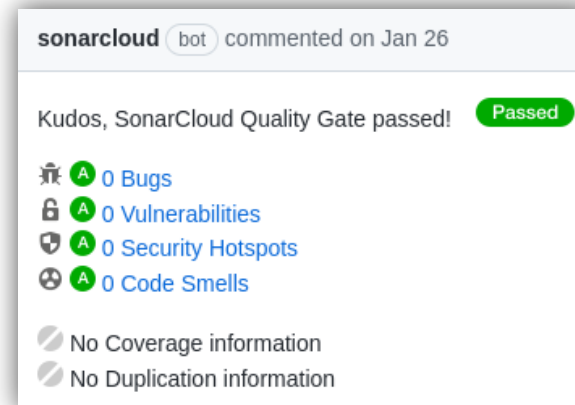
# SonarCloud



- Herramienta para el análisis estático del código
- Necesita:
  - Git server como GitHub
  - Acceso al repositorio
  - Un lenguaje aceptado
- Dos clases de configuración de los análisis:
  - **Automated Analysis** (Default). Cobertura de código no disponible. Scanner del código en servidor sonar.
  - **CI-based analysis**. Sonar scanner ejecutado externamente. Los report son enviados a SonarCloud.

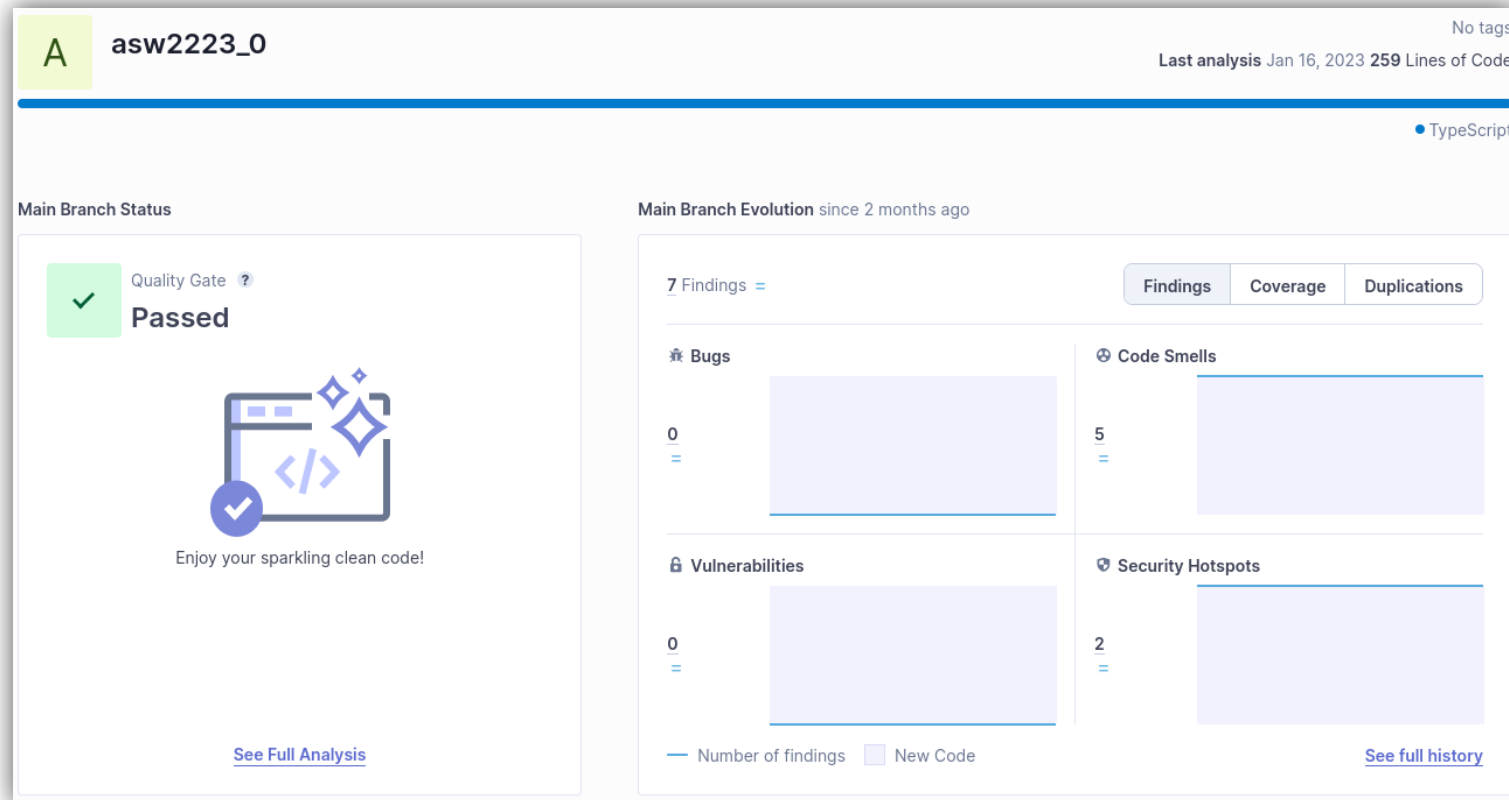
# SonarCloud - lomap\_0 configuration

- Cuando los cambios son enviados al repositorio (example, a new pull request)
- Recuperamos información del análisis de código

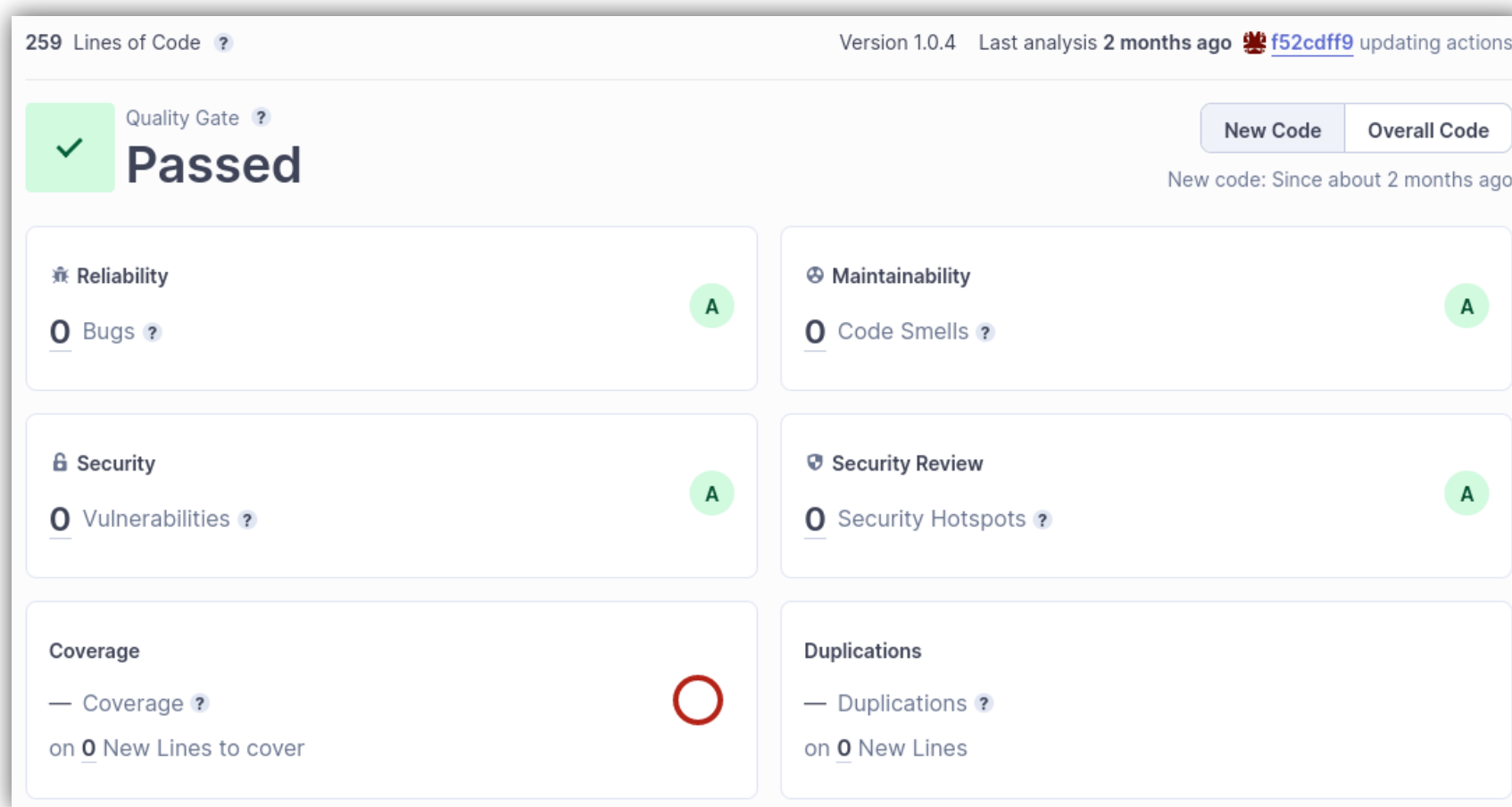


# SonarCloud

- En: Project Dashboard podemos chequear el ultimo análisis de la rama principal, las pull-requests y las ramas específicas



# SonarCloud: Evolución de la calidad del proyecto



# SonarCloud: Umbral de calidad

- En el nivel de la organización definimos distintos umbrales de calidad para asignarlos a los proyectos.

The screenshot shows the SonarCloud interface for configuring quality gates. The main section is titled 'aws-quality-gates' and includes buttons for 'Rename', 'Copy', 'Set as Default', and 'Delete'. Below this, there's a section for 'Conditions' with a table of metrics and their thresholds.

Metric	Operator	Value	Edit	Delete
Coverage	is less than	80.0%		
Duplicated Lines (%)	is greater than	15.0%		
Maintainability Rating	is worse than	A		
Reliability Rating	is worse than	A		
Security Hotspots Reviewed	is less than	100%		
Security Rating	is worse than	A		

The 'Add Condition' dialog box shows options for where the quality gate fails. It has two radio buttons: 'On New Code' (selected) and 'On Overall Code'. Below this, there's a section 'Quality Gate fails when' with a search bar and a list of metrics.

**Add Condition**

☒ On New Code ☐ On Overall Code

**Quality Gate fails when**

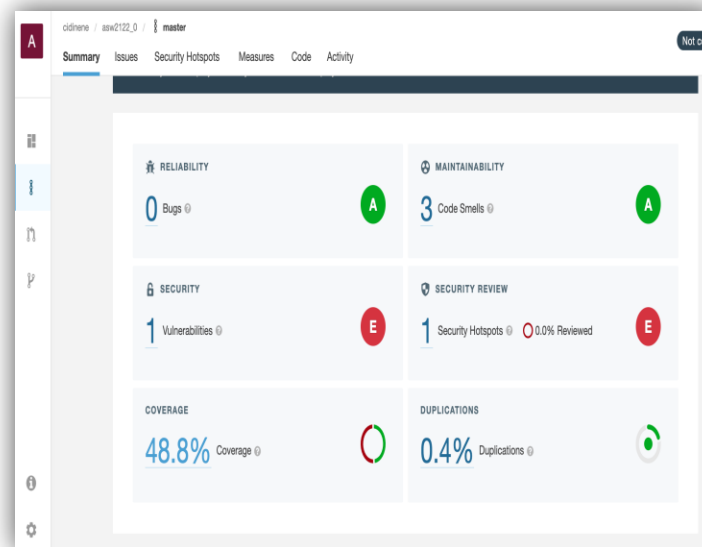
Search for metrics...

- Coverage**
  - Condition Coverage
  - Conditions to Cover
  - Line Coverage
  - Lines to Cover
  - Uncovered Conditions
  - Uncovered Lines
- Duplications**
  - Duplicated Blocks
  - Duplicated Lines

Ejemplo AWS-Quality-Gates , se puede incrementar el porcentaje de líneas duplicadas que se pueden encontrar antes de lanzar una excepción

# SonarCloud: Umbral de calidad

- Lo que se conoce como **Quality Gate** es la definición de condiciones que nuestro proyecto debe alcanzar.
  - Estas condiciones requieren distintos aspectos: cobertura de código, análisis estático del código, líneas duplicadas, ..
- **lomap\_o** tiene configurada la calidad de código con sonarcloud.





# SonarCloud: Perfiles y reglas

- Las reglas están definidas en los perfiles
- Podemos añadir, desactivar y actualizar reglas creando un nuevo perfil :
  - Copiar un perfil padre – Cambiarlo – asociarlo al proyecto

The image consists of three screenshots from the SonarCloud web interface, illustrating the process of creating and configuring a new quality profile.

- Left Screenshot:** Shows the 'Quality Profiles' page. A list of profiles is displayed, including 'Sonar way', 'Text', 'TypeScript', and 'VB.NET'. The 'TypeScript' profile is selected, and a context menu is open, showing options like 'Compare', 'Copy', 'Extend', and 'Set as Default'. A red box highlights this menu.
- Middle Screenshot:** Shows the 'Rules' configuration page for the 'Sonar new Way' profile. A table lists rules with columns for 'Active' and 'Inactive' counts. A red box highlights the 'Rules' table.
- Right Screenshot:** Shows the 'Projects' configuration page for the 'Sonar new Way' profile. It indicates that no projects are explicitly associated to the profile. A red box highlights the 'Projects' section.

Create a new profile

Set the profile rules

Associate the profile to the project

# Configuración de reglas

← → ↺

sonarcloud.io/organizations/arquisoft/rules?qprofile=AX-mgR2YnzNFv0H6nzDH&activation=true

🔍 📄 ☆ ⚙️ 👤 Paused ⋮

sonarcloud

My Projects My Issues + 🔍

type 1/1 ^ v x 🔔 ? 👁 🗺

Arquitectura del Software

http://campusvirtual.uniovi.es Key: arquisoft

Projects Quality Profiles Rules Quality Gates Members Administration

Filters

Clear All Filters

Search for rules...

Language

Type

- Bug 36
- Vulnerability 24
- Code Smell 108
- Security Hotspot 32

Tag

Repository

Default Severity

Status

Security Category

Available Since

Quality Profile SONAR N... Clear

Inheritance

Bulk Change

↑ ↓ to select rules ← → to navigate ↺ 1 / 200 rules

⬆	"===" and "!== " should be used instead of "==" and "!="	TypeScript	Code Smell	suspicious	⌵	Deactivate
⬆	"arguments.caller" and "arguments.callee" should not be used	TypeScript	Code Smell	obsolete	⌵	Deactivate
⬆	"await" should not be used redundantly	TypeScript	Code Smell	redundant	⌵	Deactivate
⬆	"await" should only be used with promises	TypeScript	Code Smell	confusing	⌵	Deactivate
⬆	"catch" clauses should do more than rethrow	TypeScript	Code Smell	clumsy, error-ha...	⌵	Deactivate
⬆	"default" clauses should be last	TypeScript	Code Smell		⌵	Deactivate
⬆	"delete" should be used only with object properties	TypeScript	Bug		⌵	Deactivate
⬆	"delete" should not be used on arrays	TypeScript	Code Smell		⌵	Deactivate
⬆	"for in" should not be used with iterables	TypeScript	Code Smell		⌵	Deactivate
⬆	"for of" should be used with Iterables	TypeScript	Code Smell	clumsy	⌵	Deactivate
⬆	"for" loop increment clauses should modify the loops' counters	TypeScript	Code Smell	confusing	⌵	Deactivate

# Ver las alertas mientras programamos

- <https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarlint-vscode>

