



ARQUITECTURA DEL SOFTWARE

2024-25

Jose Emilio Labra Gayo

Pablo González

Irene Cid

Diego Martín



Escuela de
Ingeniería
Informática



Universidad de Oviedo

Laboratorio 6

TDD: Test-driven development

Cobertura de código(SonarCloud)

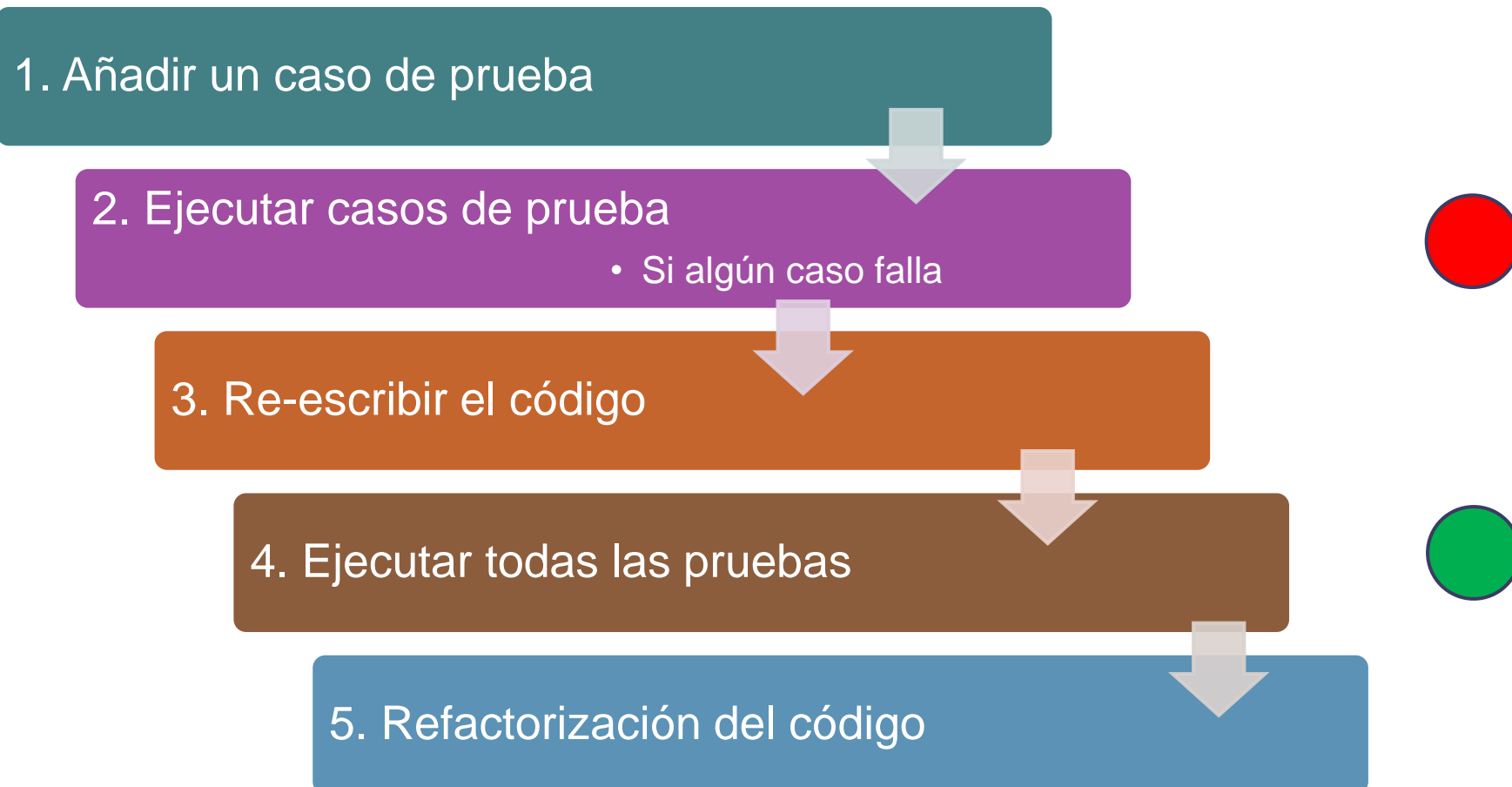
Integración continua (GitHub Actions)

Herramientas para el análisis estático (SonarCloud)

TDD - Introducción

- Proceso de desarrollo de código donde los requisitos se convierten en casos de prueba específicos.
- 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.

Code Coverage

Cobertura de código: Medida que nos indicala proporción de líneas de c código que son probadas en alguno de nuestros test.

SonarCloud: Herramienta que incluye la cobertura de código como una métrica más en la evaluación del código.

Sonar Cloud recoge los datos de los resultados del lanzamiento de los test, recogiendo los siguientes valores:

LC = Líneas cubiertas (`lines_to_cover` - `uncovered_lines`)

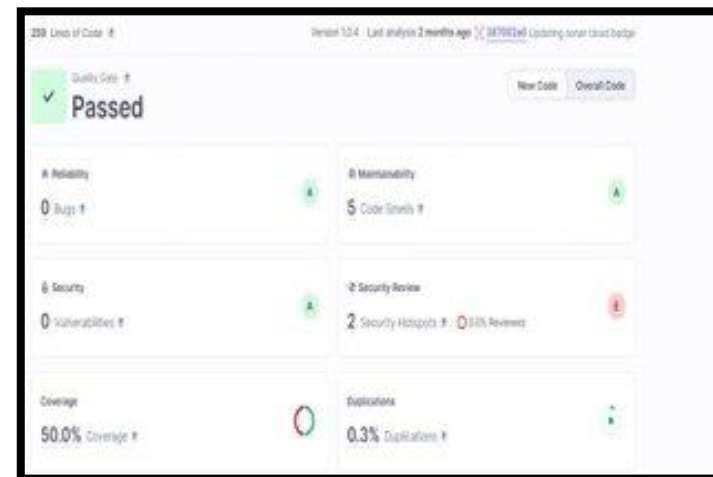
EL = Número total de líneas ejecutables (`lines_to_cover`)

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_wiq_???



TDD - Test de ejemplo

- Probar un componente básico en React.js

```
webapp > src > JS App.test.js > ...
```

```
1  import { render, screen } from '@testing-library/react';
2  import App from './App';
3
4  test('renders welcome message', () => {
5    render(<App />);
6    const welcomeMessage = screen.getByText(/Welcome to the 2024 edition of the Software Architecture course/i);
7    expect(welcomeMessage).toBeInTheDocument();
8  });
```

TDD - Test de ejemplo

Comprobamos que AddUser funciona bien:

Algunas veces tenemos que moquear parte de la prueba.

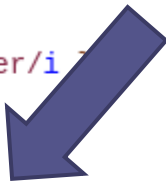
Si no mockeamos el api, nuestro test depende de UserService de los resultados de la *rest-api*.

Como se trata de test unitarios debemos eliminar esta dependencia.

```

14  it('should add user successfully', async () => {
15    render(<AddUser />);
16
17    const usernameInput = screen.getByLabelText(/Username/i);
18    const passwordInput = screen.getByLabelText(/Password/i);
19    const addButton = screen.getByRole('button', { name: /Add User/i });
20
21    // Mock the axios.post request to simulate a successful response
22    mockAxios.onPost('http://localhost:8000/adduser').reply(200);
23
24    // Simulate user input
25    fireEvent.change(usernameInput, { target: { value: 'testUser' } });
26    fireEvent.change(passwordInput, { target: { value: 'testPassword' } });
27
28    // Trigger the add user button click
29    fireEvent.click(addButton);
30
31    // Wait for the Snackbar to be open
32    await waitFor(() => {
33      expect(screen.getByText(/User added successfully/i)).toBeInTheDocument();
34    });
35  });

```



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 prueba

Despliegue automático

Monitorización de la calidad de código

Integración Continua - ejemplos

Jenkins

Pipeline

Hudson

Apache Continuun

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 de código abierto

La configuración se mantiene en uno o varios ficheros `.yaml` dentro del directorio **`.github/workflows`** , que podemos localizar en la raíz del directorio

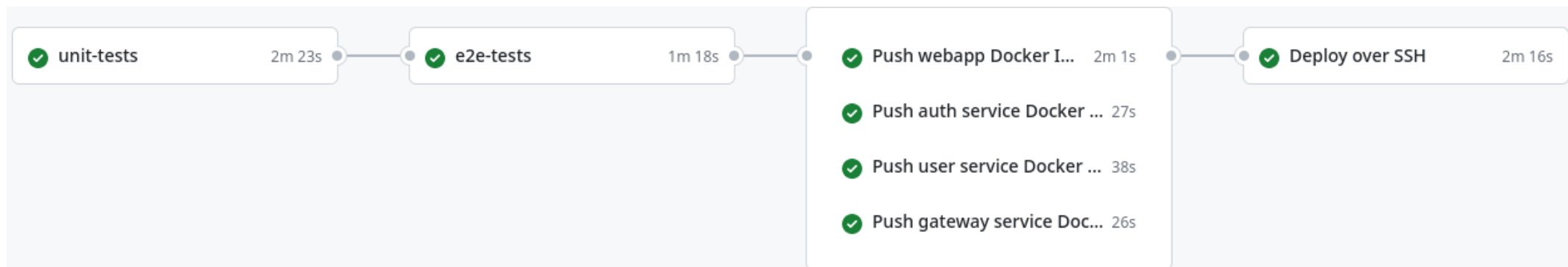
GitHub Actions

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

```

7 jobs:
8   unit-tests:
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12      - uses: actions/setup-node@v4
13        with:
14          node-version: 20
15      - run: npm --prefix users/authservice ci
16      - run: npm --prefix users/userservice ci
17      - run: npm --prefix gatewayservice ci
18      - run: npm --prefix webapp ci
19      - run: npm --prefix users/authservice test -- --coverage
20      - run: npm --prefix users/userservice test -- --coverage
21      - run: npm --prefix gatewayservice test -- --coverage
22      - run: npm --prefix webapp test -- --coverage
23      - name: Analyze with SonarCloud
24        uses: sonarsource/sonarcloud-github-action@master
25      env:
26        GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
27        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 se crea un nuevo issue.

GitHub Actions

- También tenemos jobs para crear imágenes de Docker y publicarlas
- Comprueba la [documentación](#) para más configuraciones

```
42 docker-push-webapp:
43     name: Push webapp Docker Image to GitHub Packages
44     runs-on: ubuntu-latest
45     permissions:
46         contents: read
47         packages: write
48     needs: [e2e-tests]
49     steps:
50     - uses: actions/checkout@v4
51     - name: Publish to Registry
52       uses: elgohr/Publish-Docker-Github-Action@v5
53     env:
54         API_URI: http://${{ secrets.DEPLOY_HOST }}:8000
55     with:
56         name: arquisoft/wiq_0/webapp
57         username: ${ github.actor }
58         password: ${ secrets.GITHUB_TOKEN }
59         registry: ghcr.io
60         workdir: webapp
61         buildargs: API_URI
```


Análisis estático del código

Analiza el código sin compilarlo

Detecta bugs, code smells, vulnerabilidades del sistema, etc

Útil para medir la calidad del código.

Se puede bloquear la subida de código que no cumpla con ciertas características de calidad

Análisis estático en SonarCloud



SonarCloud contiene herramientas para análisis estático del código

Necesita:

- Servidor de Git como GitHub

- Acceso al repositorio

- Un lenguaje soportado

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.

Sonarlint



Análisis estático desde el propio IDE (disponible en los más populares ej. IntelliJ, Visual Code, Visual Studio, Eclipse...)
Provee de análisis estático de forma local (antes de subirlo al repositorio), se ejecuta:

- De forma manual

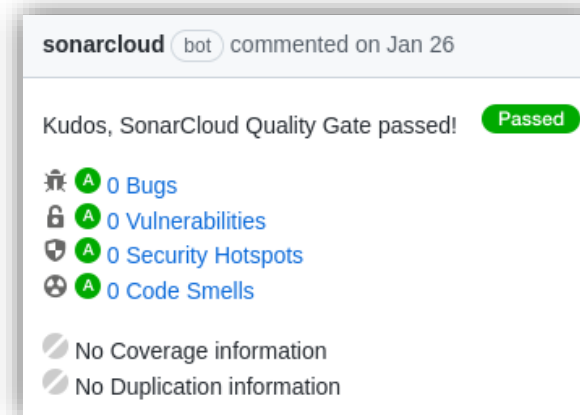
- Automáticamente sobre los archivos modificados, antes de enviarlos al repositorio.

Más información, IDEs-lenguajes soportados y cómo instalarlo en la [página oficial](#)

SonarCloud - wiq_x

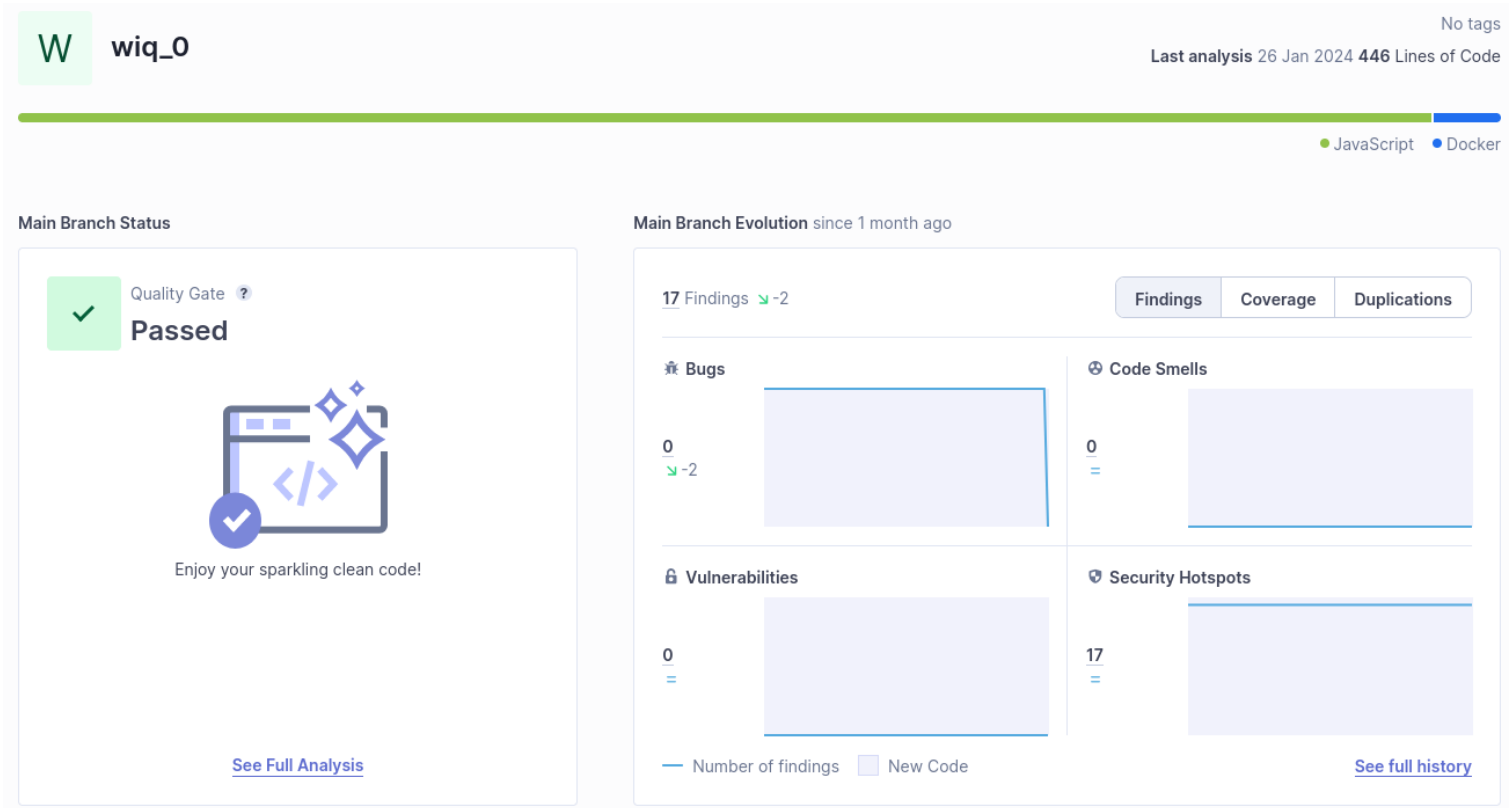
Cuando los cambios son enviados al repositorio (ejemplo: una nueva pull request)

Tenemos información acerca de la calidad de código del pull request que estamos mergeando.

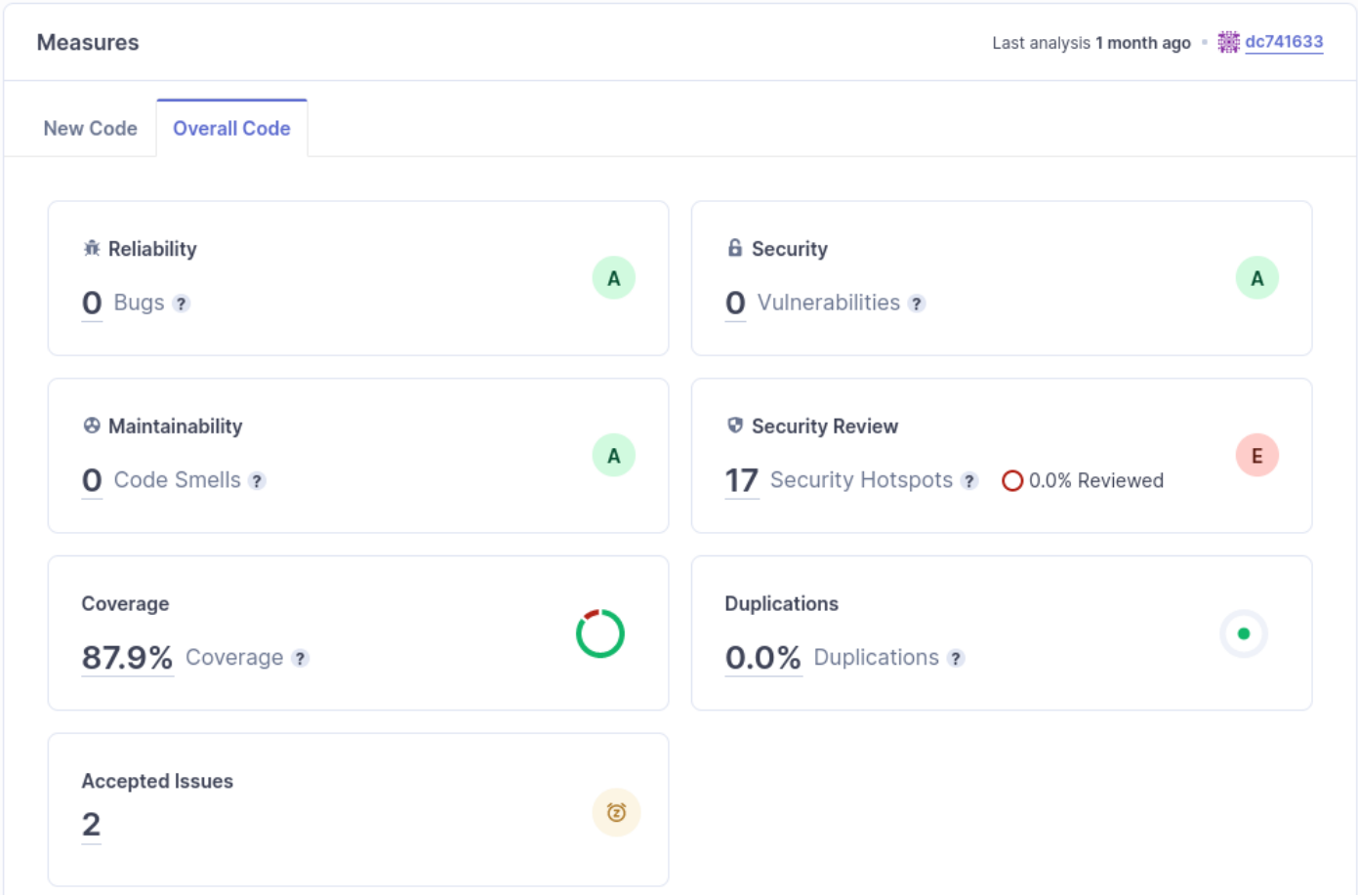


SonarCloud

En el *Project Dashboard* podemos ver el último 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 a Quality Gate named 'aws-quality-gates'. The page includes a sidebar with navigation links (Projects, Quality Profiles, Rules, Quality Gates, Members, Administration) and a main content area. The 'Quality Gates' section is active, showing a list of gates with 'aws-quality-gates' selected. The configuration for this gate is displayed, including a table of conditions.

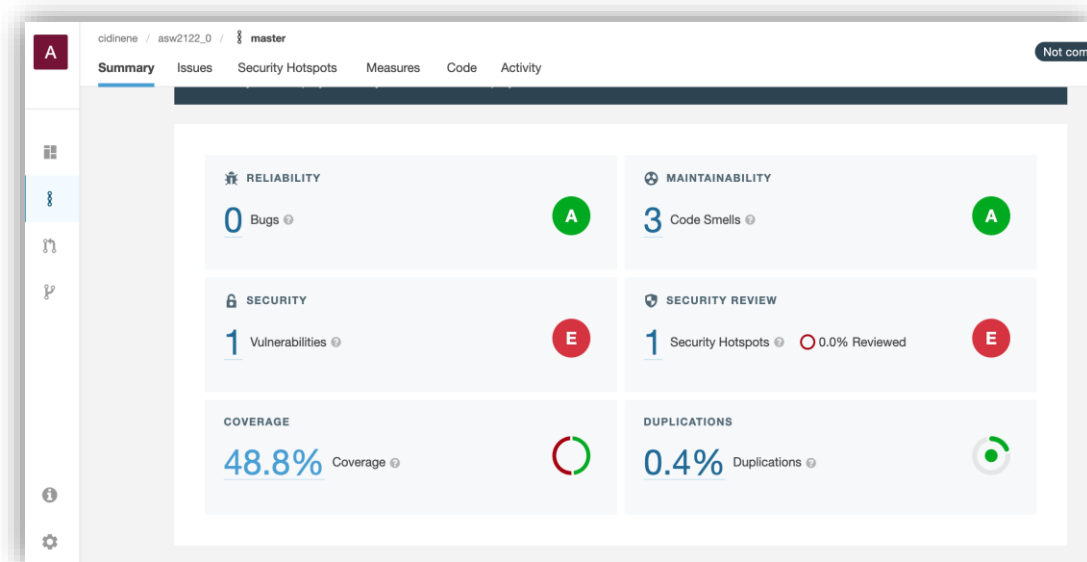
Metric	Operator	Value	Edit	Delete
Coverage	is less than	80.0%		
Duplicated Lines (%)	is greater than	15.0%		
Maintainability Rating	is worse than	A		
Reliability Rating	is worse than	A		
Security Hotspots Reviewed	is less than	100%		
Security Rating	is worse than	A		

The screenshot shows the 'Add Condition' dialog box. It has two radio buttons: 'On New Code' (selected) and 'On Overall Code'. Below these, there is a section titled 'Quality Gate fails when' with a search bar 'Search for metrics...'. A dropdown menu is open, showing a list of metrics under the 'Coverage' section: 'Condition Coverage', 'Conditions to Cover', 'Line Coverage', 'Lines to Cover', 'Uncovered Conditions', and 'Uncovered Lines'. There is also a 'Duplications' section with 'Duplicated Blocks' and '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, ..
- **wiq_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 displays 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 shown, including 'Sonar way', 'Text', 'TypeScript', and 'VB.NET'. The 'TypeScript' profile is selected, and its settings are displayed. A red box highlights the 'Settings' gear icon, with an arrow pointing to the 'Create a new profile' label.

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, with an arrow pointing to the 'Set the profile rules' label.

Right Screenshot: Shows the 'Projects' configuration page for the 'Sonar new Way' profile. A red box highlights the 'Projects' section, which indicates that no projects are explicitly associated with the profile. An arrow points to this section with the label 'Associate the profile to the project'.

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

^

⌵

×

🔔

?

👁

Arquitectura del Software

⌵

<http://campusvirtual.uniovi.es>

Key: arquisoft

Projects

Quality Profiles

Rules

Quality Gates

Members

Administration

Filters

Clear All Filters

Bulk Change

↑ ↓ to select rules

← → to navigate

🔄 1 / 200 rules

🔍 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

⬆

"===" 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

School of Computer Science, University of Oviedo

Ver las alertas mientras programamos

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

