



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)

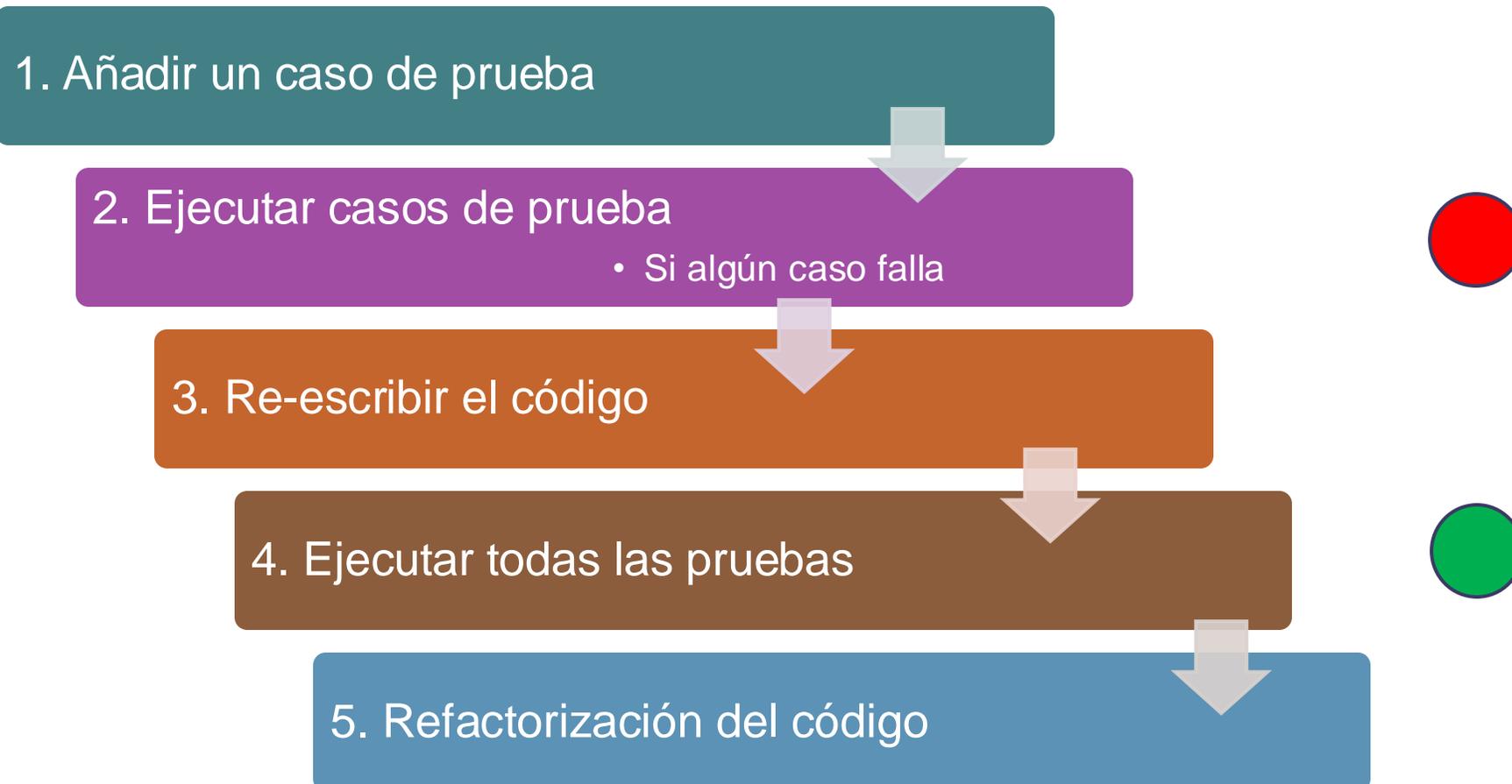
Integración continua (GitHub Actions)

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 indica la proporción de líneas de 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`)

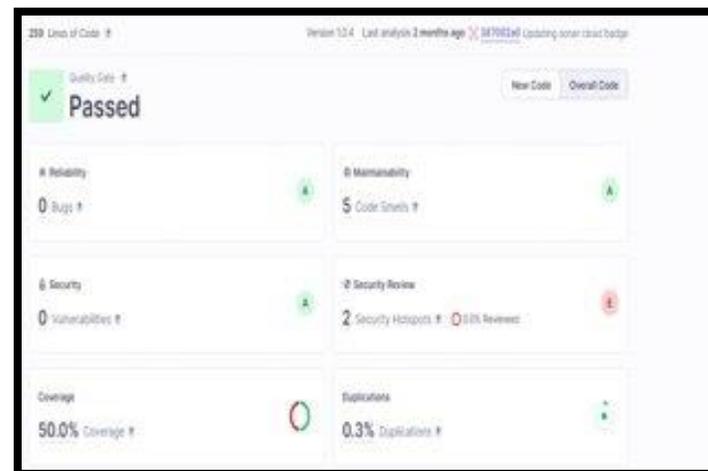
EL = Número total de líneas ejecutables (`lines_to_cover`)

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_wichat_???



TDD - Test de ejemplo

- Probar un componente básico en React.js

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders welcome message', () => {
  render(<App />);
  const welcomeMessage = screen.getByText(/Welcome to the 2025 edition of the Software Architecture course/i);
  expect(welcomeMessage).toBeInTheDocument();
});
```

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 addUserButton = 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(addUserButton);
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

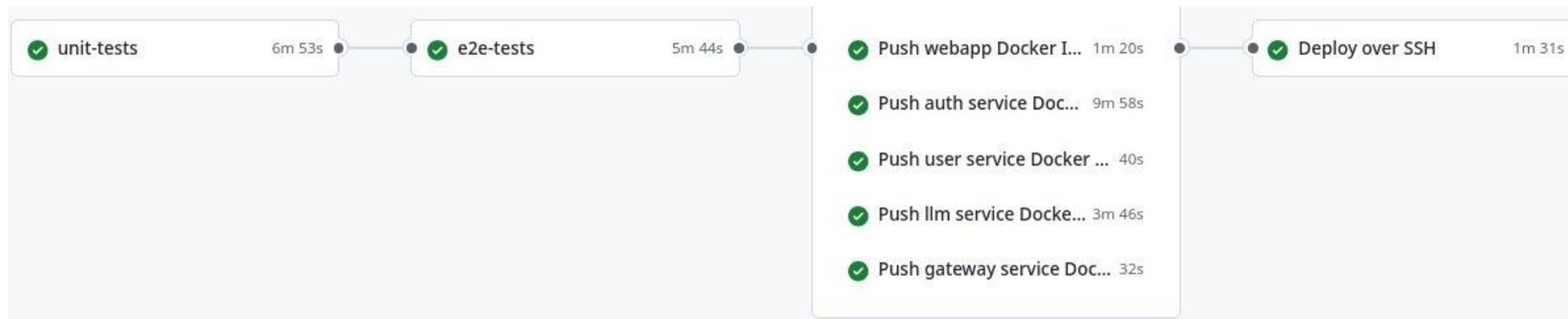
GitHub Actions

- 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

```

jobs:
  unit-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 22
      - run: npm --prefix users/authservice ci
      - run: npm --prefix users/userservice ci
      - run: npm --prefix llmservice ci
      - run: npm --prefix gateway-service ci
      - run: npm --prefix webapp ci
      - run: npm --prefix users/authservice test -- --coverage
      - run: npm --prefix users/userservice test -- --coverage
      - run: npm --prefix llmservice test -- --coverage
      - run: npm --prefix gateway-service test -- --coverage
      - run: npm --prefix webapp test -- --coverage
      - name: Analyze with SonarQube
        uses: SonarSource/sonarqube-scan-action@master
        env:
          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

```
docker-push-webapp:  
  name: Push webapp Docker Image to GitHub Packages  
  runs-on: ubuntu-latest  
  permissions:  
    contents: read  
    packages: write  
  needs: [e2e-tests]  
  steps:  
  - uses: actions/checkout@v4  
  - name: Publish to Registry  
    uses: elgohr/Publish-Docker-Github-Action@v5  
    env:  
      API_URI: http://${{ secrets.DEPLOY_HOST }}:8000  
    with:  
      name: arquisoft/wichat_0/webapp  
      username: ${{ github.actor }}  
      password: ${{ secrets.GITHUB_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

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

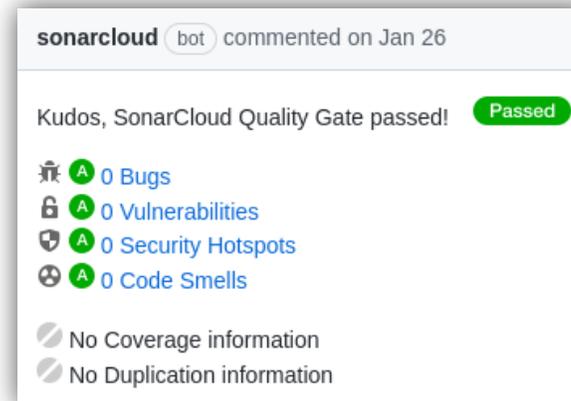
- Automáticamente sobre los archivos modificados, antes de enviarlos al repositorio.

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

SonarCloud - wiq_x

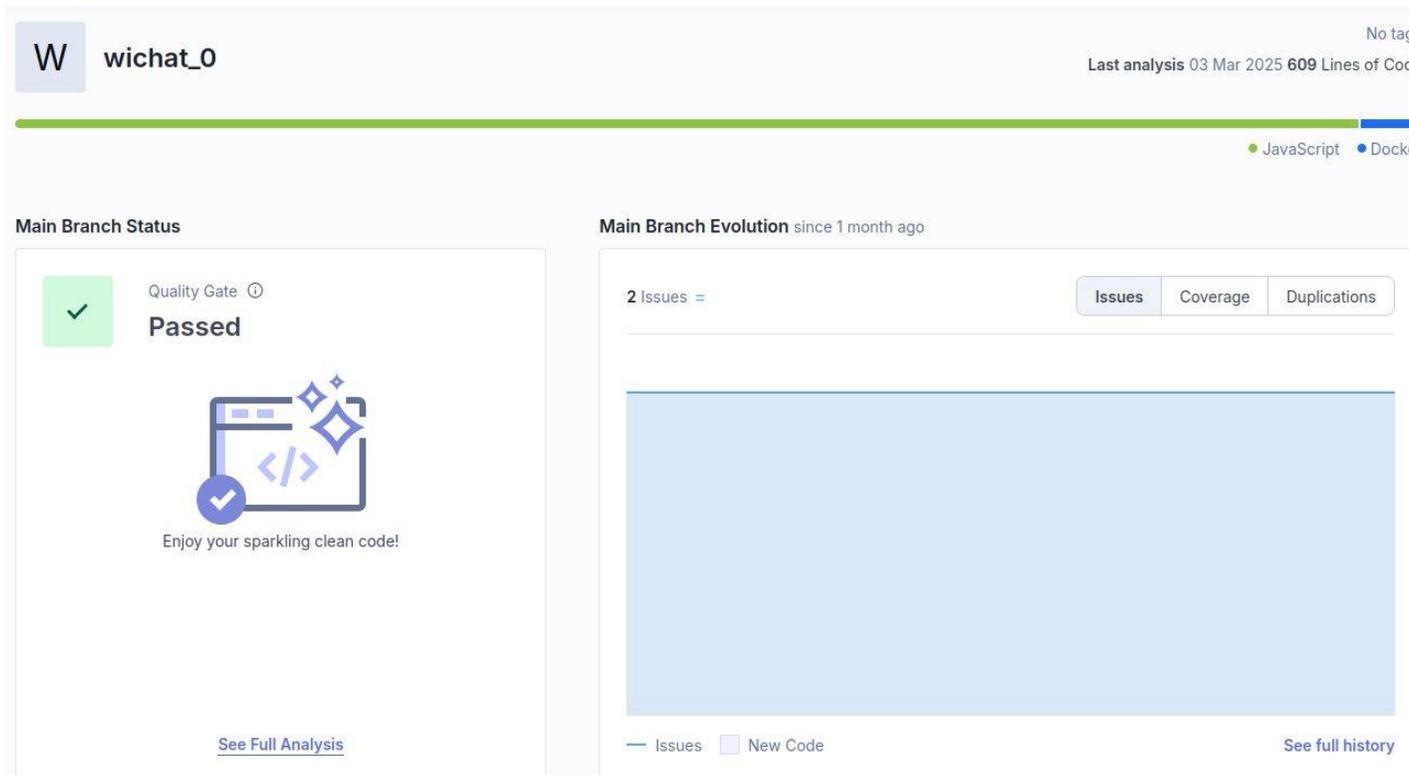
Cuando los cambios son enviados al repositorio (ejemplo: una nueva pull request)

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

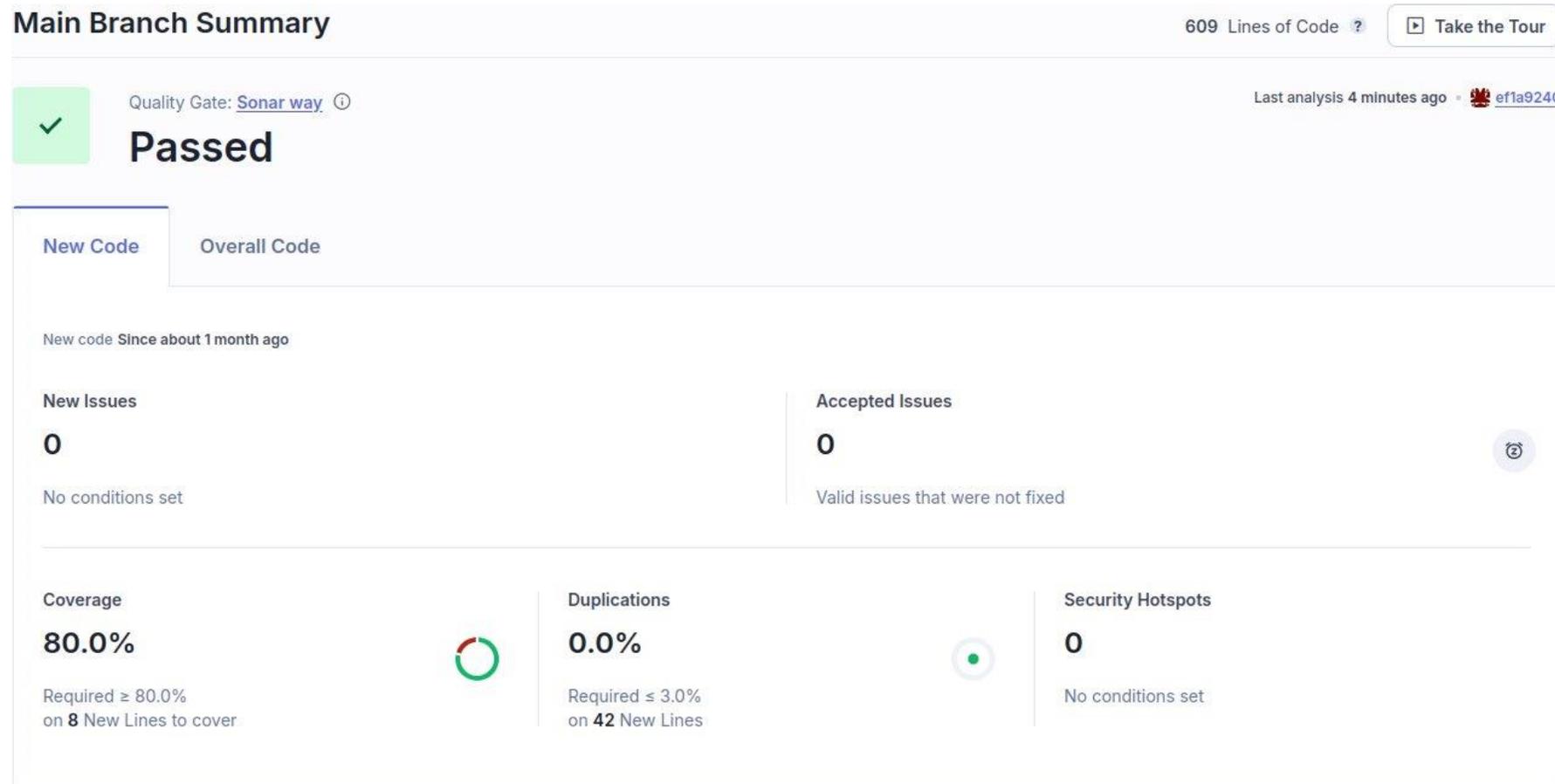


SonarCloud

En el *Project Dashboard* podemos ver el último análisis de la rama principal, las pull-requests y las ramas específicas

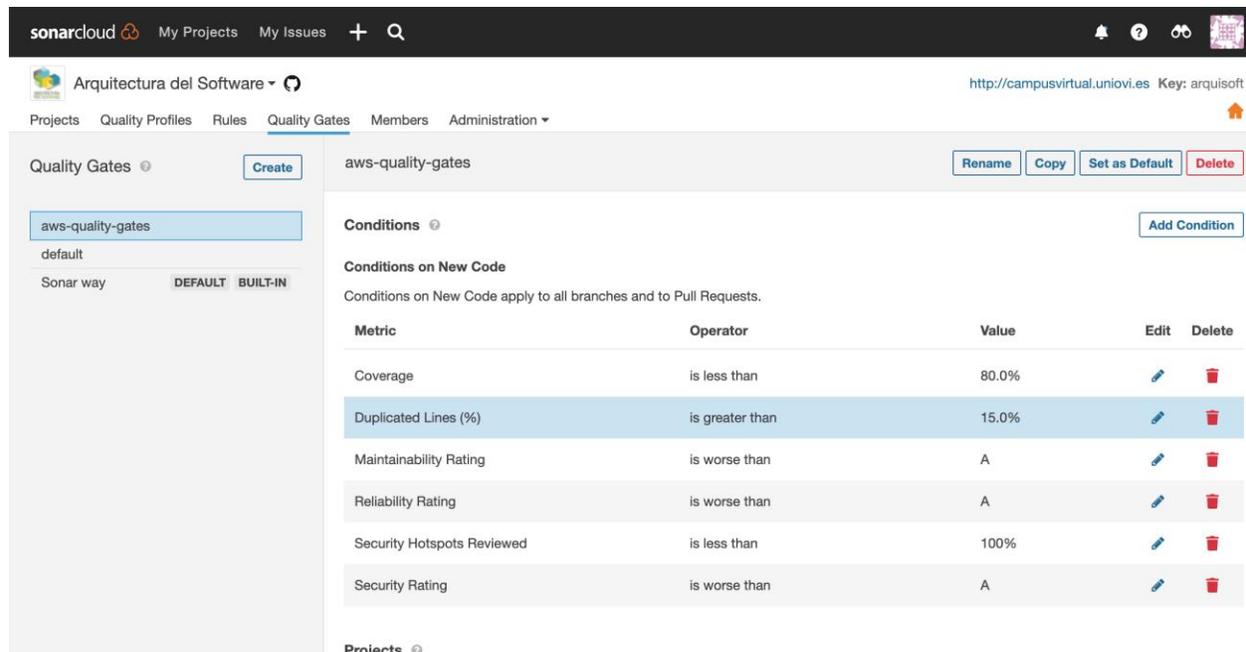


SonarCloud: Evolución de la calidad del proyecto



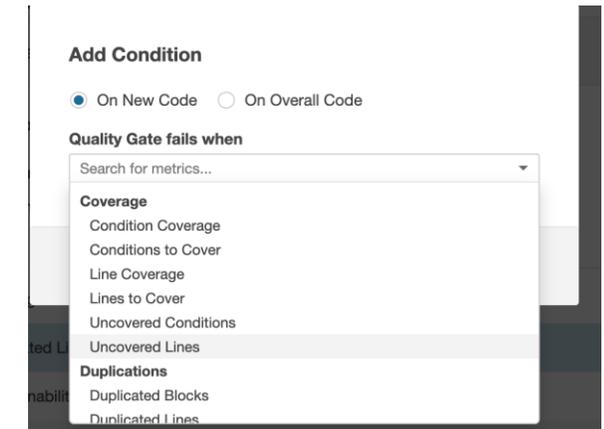
SonarCloud: Umbral de calidad

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



The screenshot shows the SonarCloud interface for configuring quality gates. The main area displays the 'aws-quality-gates' configuration with a table of conditions on new code.

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 is shown, allowing the user to select a metric for the quality gate. The dialog includes radio buttons for 'On New Code' (selected) and 'On Overall Code'. Below, a search bar is present, followed by a list of metrics under the 'Coverage' and 'Duplications' categories.

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

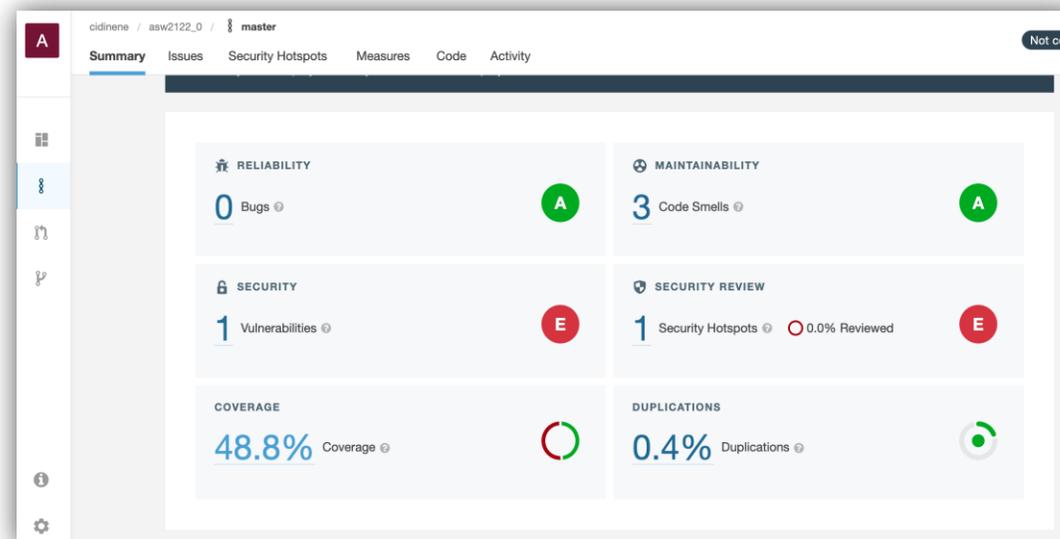
Duplications

- 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, ..
- **wichat_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 shows two screenshots of the SonarCloud interface. The left screenshot displays a list of quality profiles. The right screenshot shows the configuration page for a specific profile named 'Sonar new Way'.

Left Screenshot: Profile List

Profile Name	Type	Default	Rules	Updated	Used
Sonar way	BUILT-IN	DEFAULT	46	12 days ago	Never
Text, 1 profile(s)					
Sonar way	BUILT-IN	DEFAULT	1	2 months ago	Never
TypeScript, 2 profile(s)					
Sonar way	BUILT-IN	DEFAULT	195	4 months ago	15 hours ago
Sonar way recommended	BUILT-IN	DEFAULT	0	200	11 months ago
VB.NET, 1 profile(s)					
Sonar way	BUILT-IN	DEFAULT	119	2 months ago	Never

A red box highlights the gear icon for the 'Sonar way recommended' profile, which has opened a context menu with options: Compare, Copy, Extend, Set as Default, and Manage.

Right Screenshot: Profile Configuration

Rules Table:

Rules	Active	Inactive
Total	200	84
Bugs	36	18
Vulnerabilities	24	3
Code Smells	108	54
Security Hotspots	32	11

Projects Section:

No projects are explicitly associated to the profile.

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

sonarcloud My Projects My Issues + Q 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

Ver las alertas mientras programamos

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

