



## SOFTWARE ARCHITECTURE

2023-24

Jose Emilio Labra Gayo

Pablo González

Cristian Augusto Alonso

Jorge Álvarez Fidalgo



Escuela de  
Ingeniería  
Informática



Universidad de Oviedo

## Lab 6

TDD: Test-driven development

Code coverage(SonarCloud)

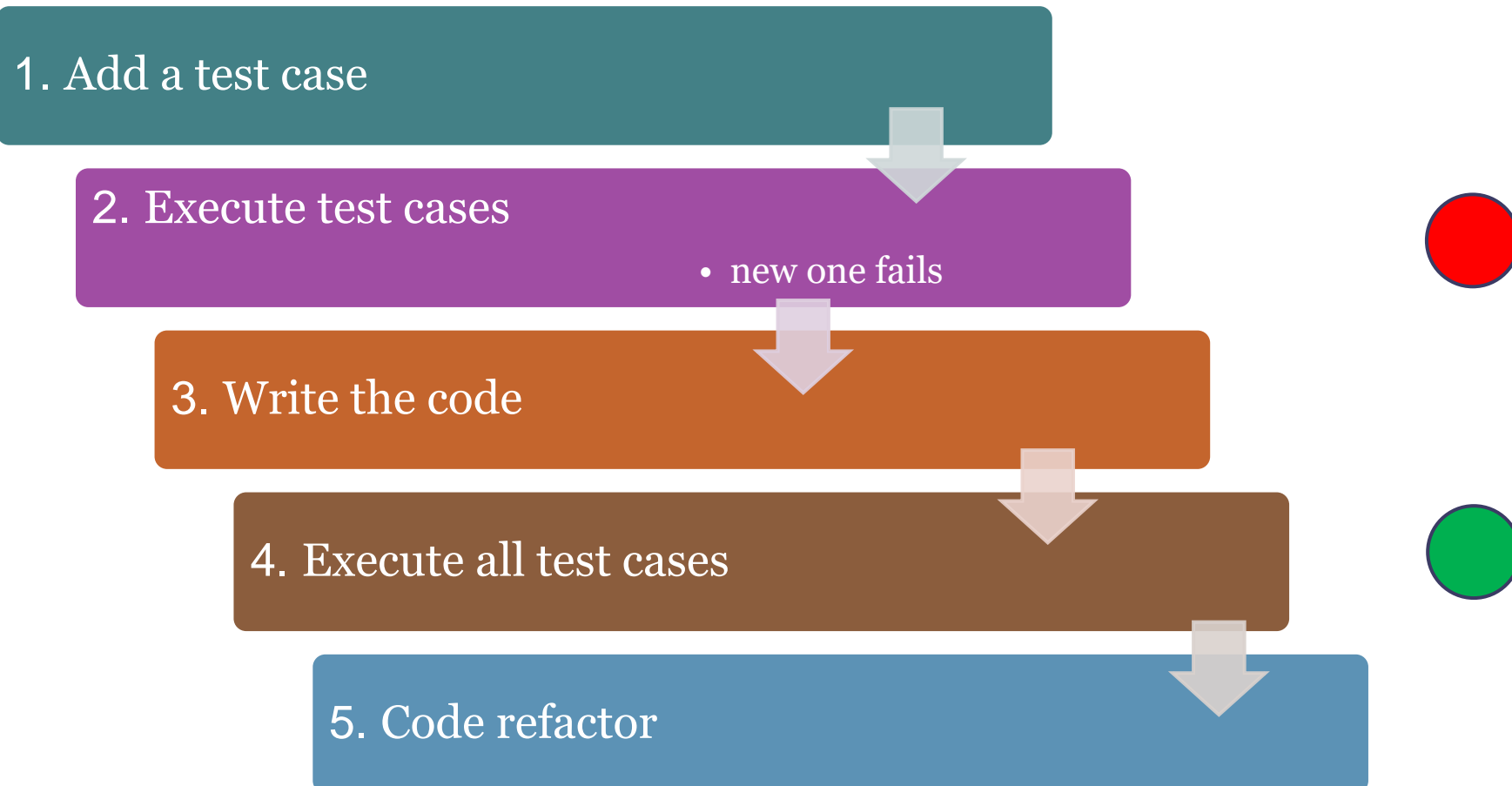
Continuous integration (GitHub Actions)

Tools to static analyze the code (SonarCloud)

# TDD

- Software development process where requirements are converted to specific test cases
- The opposite to software development that allows not tested software to be deployed
- Technique proposed by Kent Beck

# TDD - Phases



# TDD -Characteristics

- Simple code created to satisfy the test case
- We get clean code as a result
- And a test-suite
- Helps focus to know what we want to implement

# SonarCloud - Coverage

- Tool that includes code coverage as a metric in the code evaluation process
- Code coverage: Measure to show what code lines has been executed by a test suite
- Some terminology about SonarCloud:
  - LC: `lines_to_cover - uncovered_lines`
  - EL: `lines_to_cover`

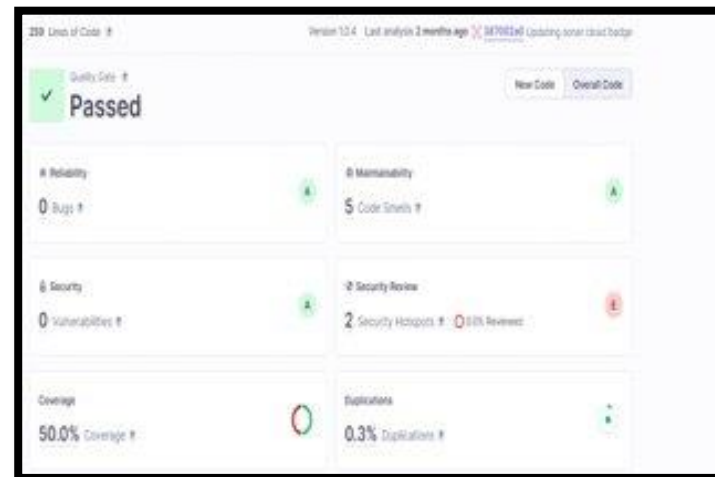
# SonarCloud

- Coverage ratio is calculated with the formula:

$$LC/EL$$

- After the tests, it generates a file that allows to do the analysis

- [https://sonarcloud.io/summary/overall?id=Arquisoft\\_wiq](https://sonarcloud.io/summary/overall?id=Arquisoft_wiq)  
???



# TDD - Example test

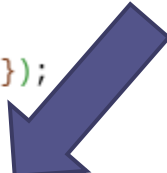
- Testing a basic component in React.js (App.test.js)

```
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 - Example test

- Checking that the AddUser component works well:
  - Sometimes we need to mock some part of the application
  - If we didn't mock the api, our test would depend on the *userservice*
  - As these are unitary tests, we simulate that part of the app

```
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  });
```





# Continuous Integration (CI)

- Development practice that promotes developers to **integrate** code into a shared repository several times a day
- Every task to build the software is executed when some condition is met
  - For instance, a push a pull request, or the creation of a new release

# Continuous Integration (CI)

- Detect and solve problems continuously
- Always available
- Immediate execution of unit test cases and E2E tests.
- Automatic deployment
- Project quality monitorization.

# Continuous Integration (CI)

- Examples:
  - Jenkins
  - Pipeline
  - Hudson
  - Apache Continuum
  - Travis
  - **GitHub Actions**

# Continuous Integration (CI) -Uses

- Common usages:
  - Maintenance of the code in a repository
  - Building automation
  - Quick building
  - Execute test cases in a cloned production environment
  - Show results of last build.

# GitHub Actions

- Continuous integration service for projects stored in GitHub
- Free for free software projects
- Configuration is in one or multiple YAML files inside the `.github/workflows` directory that is localized in the root directory of the project

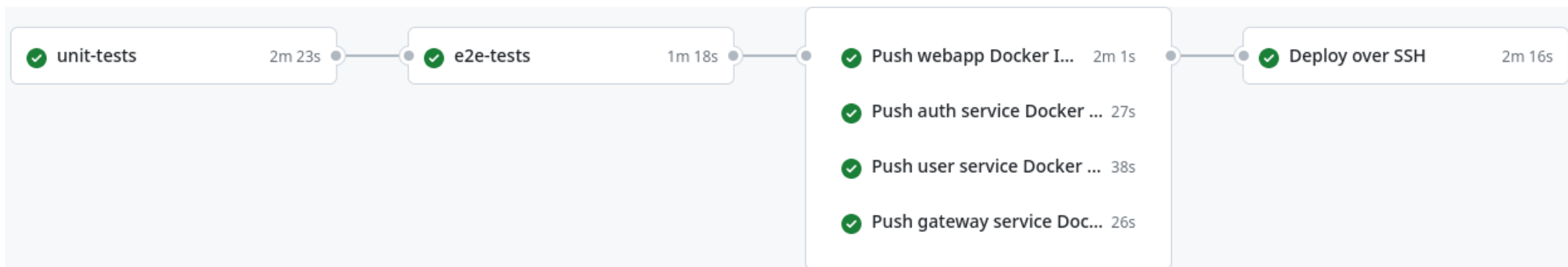
# GitHub Actions

- `.yml` specifies:
  - Conditions for firing the process
  - List of jobs
    - Each executed in a specific environment
  - Steps to carry out the job (checkout, install dependencies, build and 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 gateway-service 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 gateway-service 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

- Each job can have a specific purpose
  - Test a part of the app, deploy, etc.
- GitHub actions can be used to automate other parts of the repository.
  - Example: autoreply to new issues created in the repository

# GitHub Actions

- We have jobs also to build the docker images and publish them to github
- Check the full [documentation](#) for the CI configuration

```
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
```



# Static analysis of the code

- Analyze the code without compiling it based in rules
- Detects bugs, code smells, system vulnerabilities, etc.
- Useful to control the code quality.
- If the code does not meet the quality requirements, then the commit can be blocked

# SonarCloud



- Static code analysis tool
- It needs:
  - Git server like GitHub
  - Repository access
  - An accepted language
- Two types of analysis configuration:
  - **Automated Analysis** (Default). Code coverage not available. Scanner running in SonarCloud servers
  - **CI-based analysis**. Sonar scanner running at the project server and sending reports to SonarCloud.

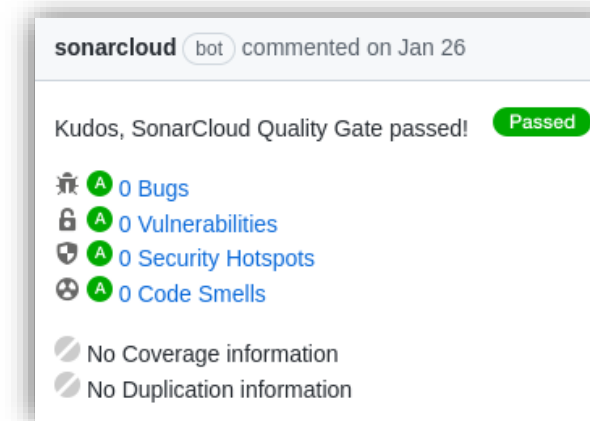
# Sonarlint



- SonarLint detects and highlights issues that can lead to bugs, vulnerabilities, and code smells in your IDE (available in the popular ones e.g. IntelliJ, Visual Code, Visual Studio, Eclipse...)
- The análisis is performed locally (before the changes are submitted to the repository), can be executed:
  - **Manually**
  - **Automatically over the changed files before the commit-push.**
- For further details regarding supported IDEs, languages and installation instructions, please visit the [oficial webpage](#)

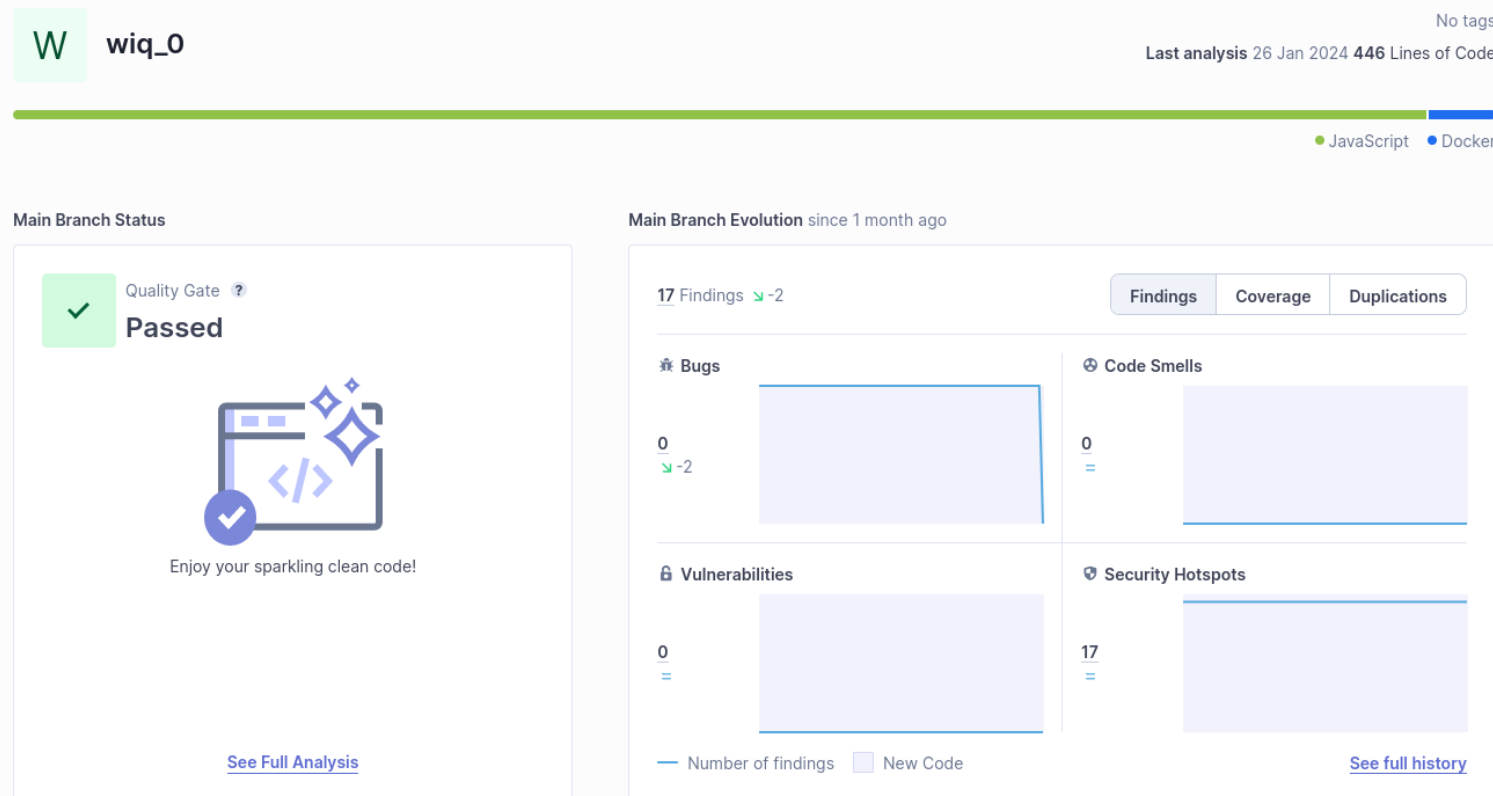
# SonarCloud - wiq\_0 configuration

- After changes are pushed to the repository (example, a new pull request)
- We have information about the code quality of the pull request that we are merging to our project

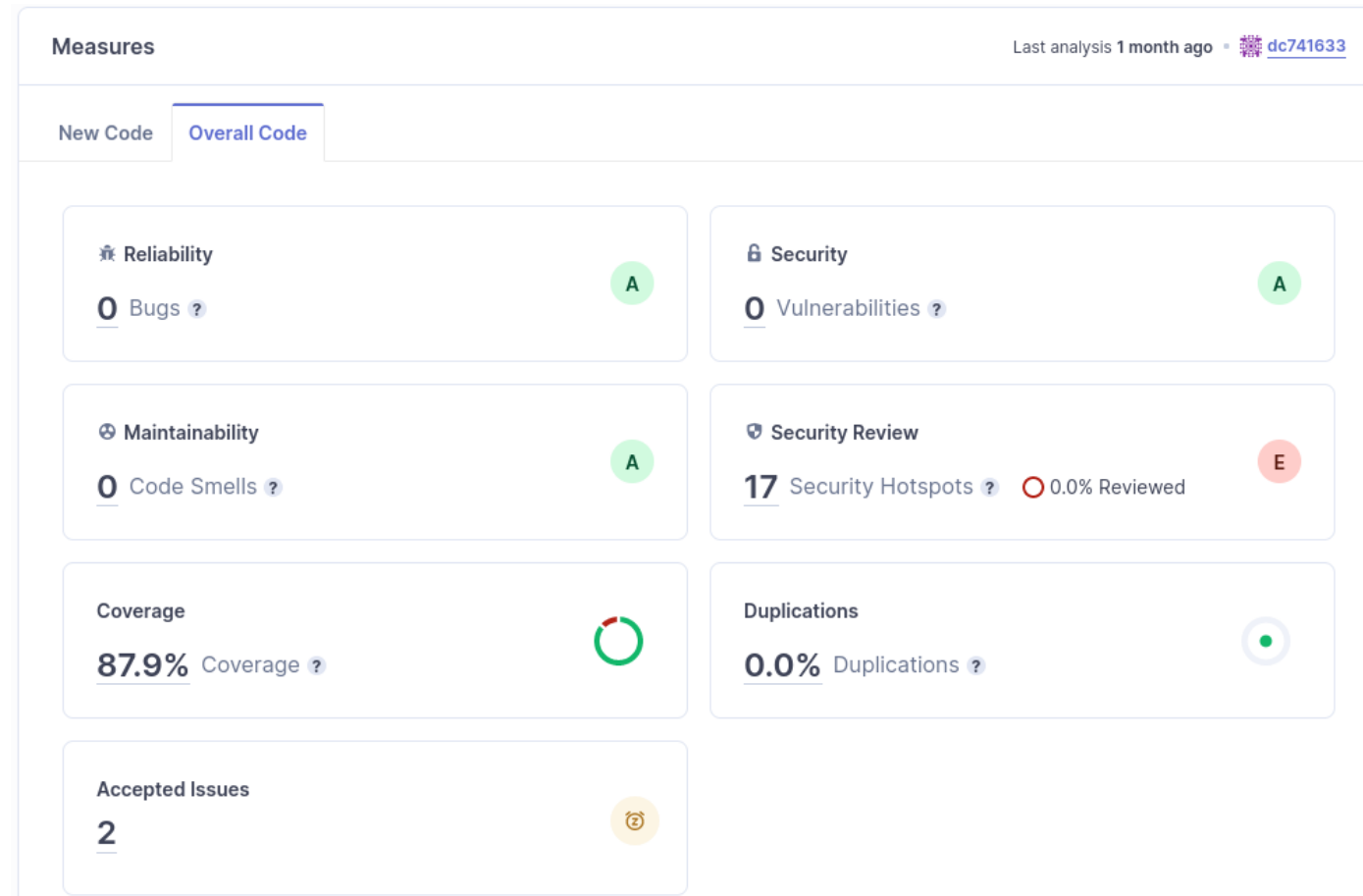


# SonarCloud

- In the Project Dashboard we can check project last analysis in the main branch, pull request and specific branches



# SonarCloud: Project certification and Quality evolution



# SonarCloud: Quality Gates

- At organization level, we can define the Quality Gates that our project must pass.

The screenshot shows the SonarCloud interface for configuring a Quality Gate named 'aws-quality-gates'. The page includes a sidebar with navigation options like 'Projects', 'Quality Profiles', 'Rules', 'Quality Gates', 'Members', and 'Administration'. The main content area displays the configuration for 'aws-quality-gates', including a table of conditions and a list of projects.

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 users to select a metric and define the failure condition. It includes radio buttons for 'On New Code' (selected) and 'On Overall Code'. A search bar for metrics is present, and a dropdown menu lists various metrics under 'Coverage' and 'Duplications'.

**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

Example AWS-Quality-Gates , we increase the procentage of duplicate lines that can be found before launch exception

# SonarCloud: Quality gates

- A **Quality Gate** is a set of conditions that our project should meet.
  - That conditions include different aspect: code coverage, static code analyse based in rules, code duplicated, ..
- **wiq\_o** default project uses code coverage with SonarCloud



# SonarCloud: Profiles and Rules

- Rules are defined at profile level
- We can add, deactivate, update rules creating a new profile :
  - Copy a parent profile - change it - associate it to the project

The left screenshot shows the 'Quality Profiles' page in SonarCloud. A list of profiles is displayed, including 'Sonar way BUILT-IN', 'Text, 1 profile(s)', 'TypeScript, 2 profile(s)', and 'VB.NET, 1 profile(s)'. A context menu is open over the 'Sonar way recommended' profile, with the 'Copy' option highlighted.

The right screenshot shows the configuration page for the 'Sonar new Way' profile. It features a 'Rules' table with columns for 'Active' and 'Inactive' rules, and a 'Projects' section where no projects are currently associated with the profile.

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

Create a new profile

Set the profile rules

Associate the profile to the project

# Rules configuration

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

# View alerts when coding

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

