



Software Architecture

Lab. 08

TDD: Test-driven development

Code coverage(SonarCloud)

Continuous integration (GitHub Actions)

Tools to static analyze the code (SonarCloud)

2022-23

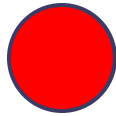
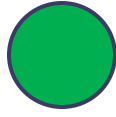
Jose Emilio Labra Gayo
Pablo González
Irene Cid
Cristian Augusto

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:

1. Add a test case
2. Execute test cases -> new one fails 
3. Write the code
4. Execute all test cases 
5. Code refactor

TDD

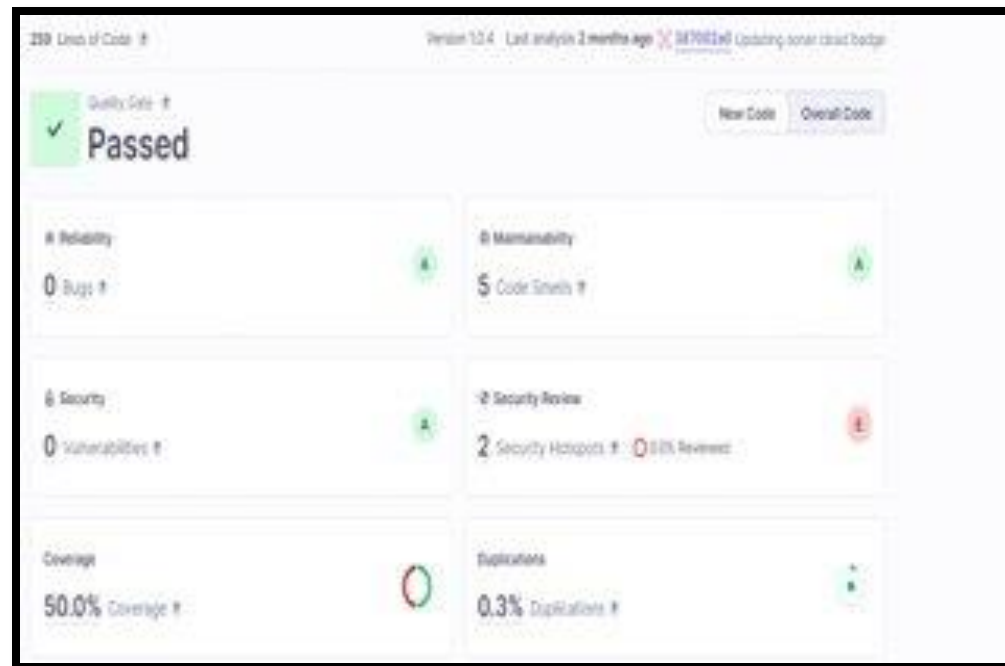
- Simple code created to satisfy the test case
- We get clean code as a result
- And a test-suite
- Helps focus to know what we want to implement

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:
$$\frac{LC}{EL}$$
- After the tests, it generates a file that allows to do the analysis
 - https://sonarcloud.io/summary/overall?id=Arquisoft_lomap_???



TDD - Example test

- Checking that the UserList component works well:
 - We create a list of users
 - We pass it to the UserList component
 - We check that both name and email are rendered

```
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 - Example test

- Checking that the EmailForm 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 restapi
 - As these are unitary tests, we simulate that part of the app



```
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 })
```


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)

- 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

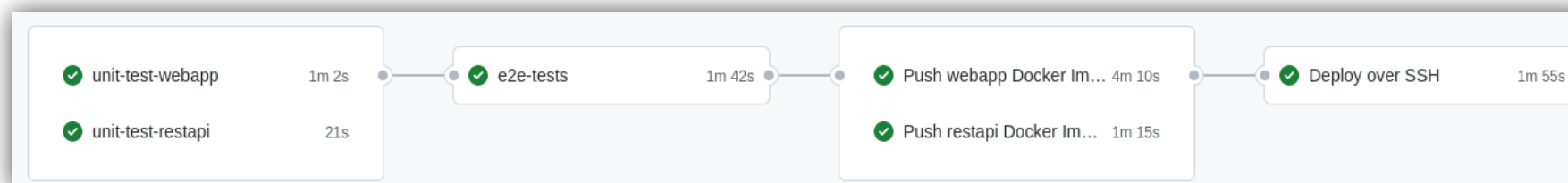
GitHub Actions

- .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)

```
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

- 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

- *uses: actions/checkout@v3.*
Uses an action created by the community.
In this case, it checks out the project to the runner
- *uses: actions/setup-node@v3*
with:
node-version: 18
Installs node in the runner
- *run: npm ci*
Runs a command (install the dependencies)
- *run: npm test*
Executes the unitary tests. If some fail, the CI will fail

GitHub Actions

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

```
docker-push-webapp:  
  name: Push webapp Docker Image to GitHub Packages  
  runs-on: ubuntu-latest  
  needs: [e2e-tests]  
  steps:  
    - uses: actions/checkout@v3  
    - name: Publish to Registry  
      uses: elgohr/Publish-Docker-Github-Action@v5  
      env:  
        API_URI: http://${{ secrets.DEPLOY_HOST }}:5000/api  
      with:  
        name: pglez82/asw2223_0/webapp  
        username: ${ github.actor }  
        password: ${ secrets.DOCKER_PUSH_TOKEN }  
        registry: ghcr.io  
        workdir: webapp  
        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

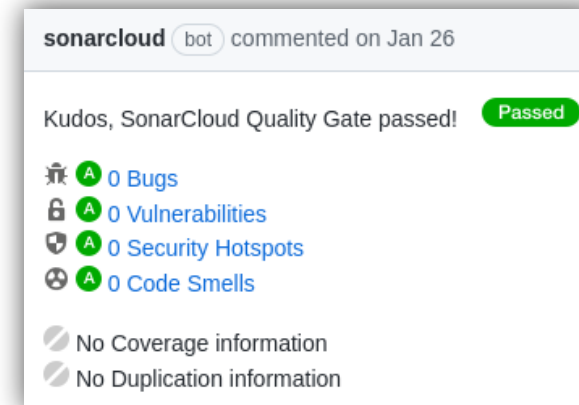
SonarCloud



- 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.

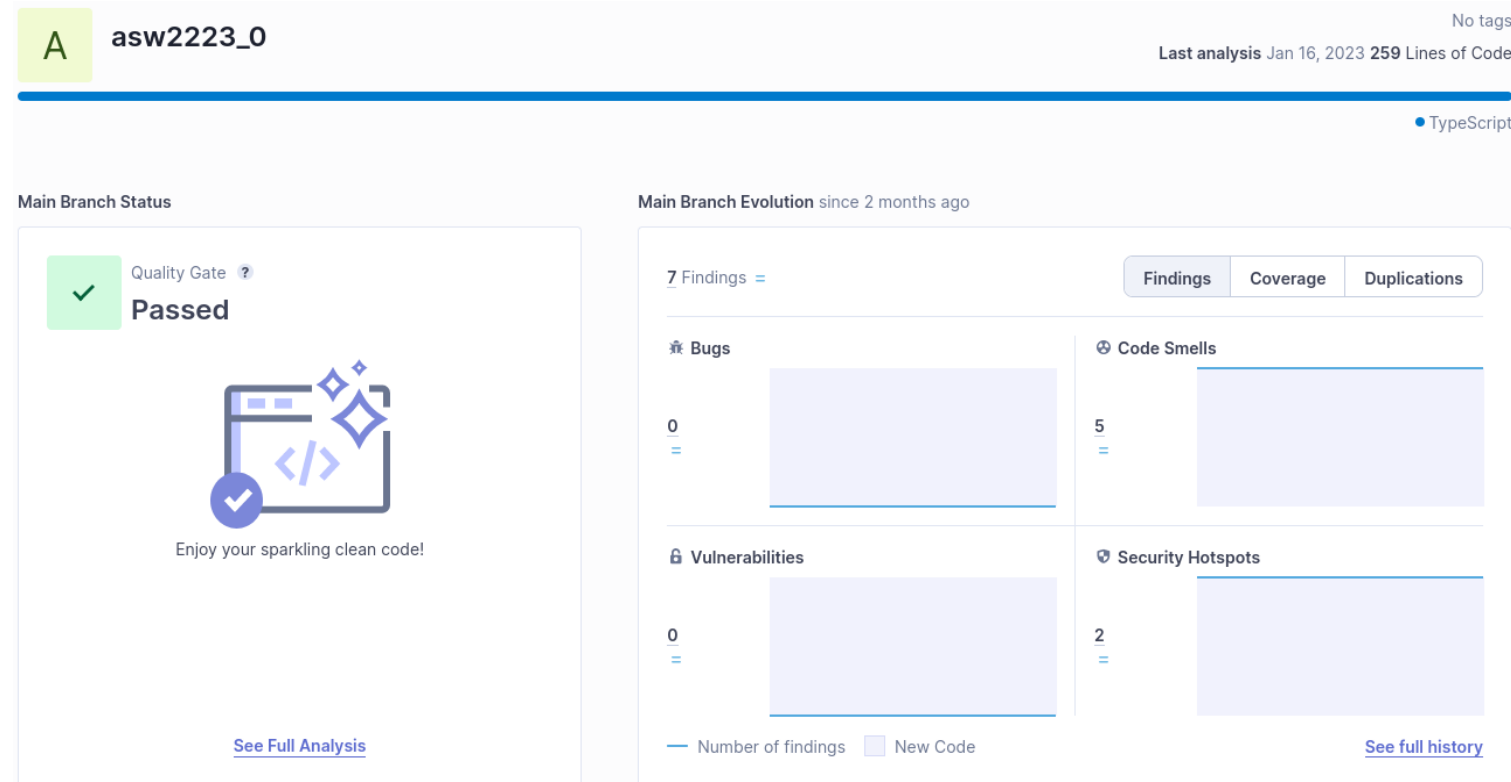
SonarCloud - lomap_0 configuration

- 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

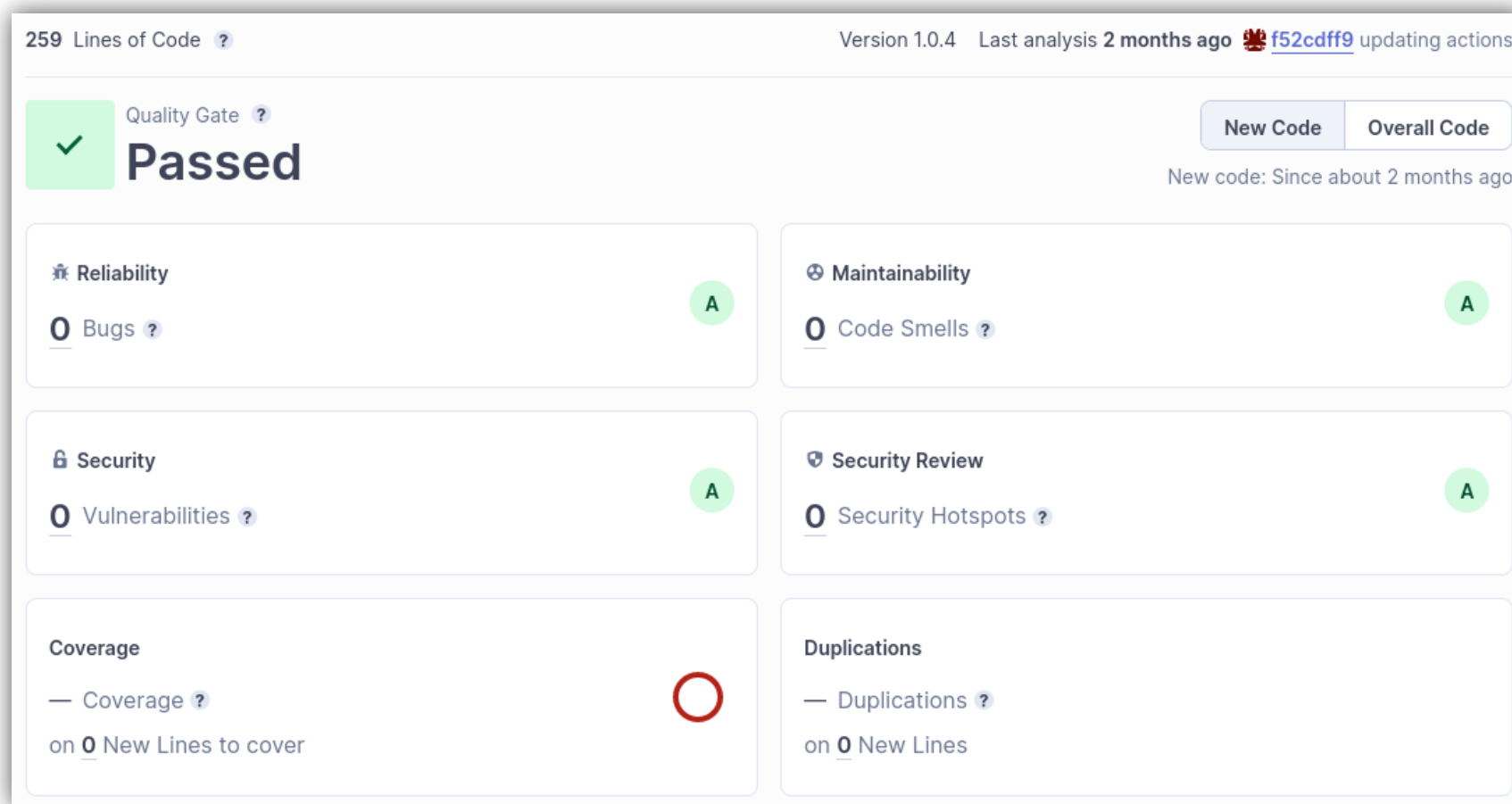


SonarCloud

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

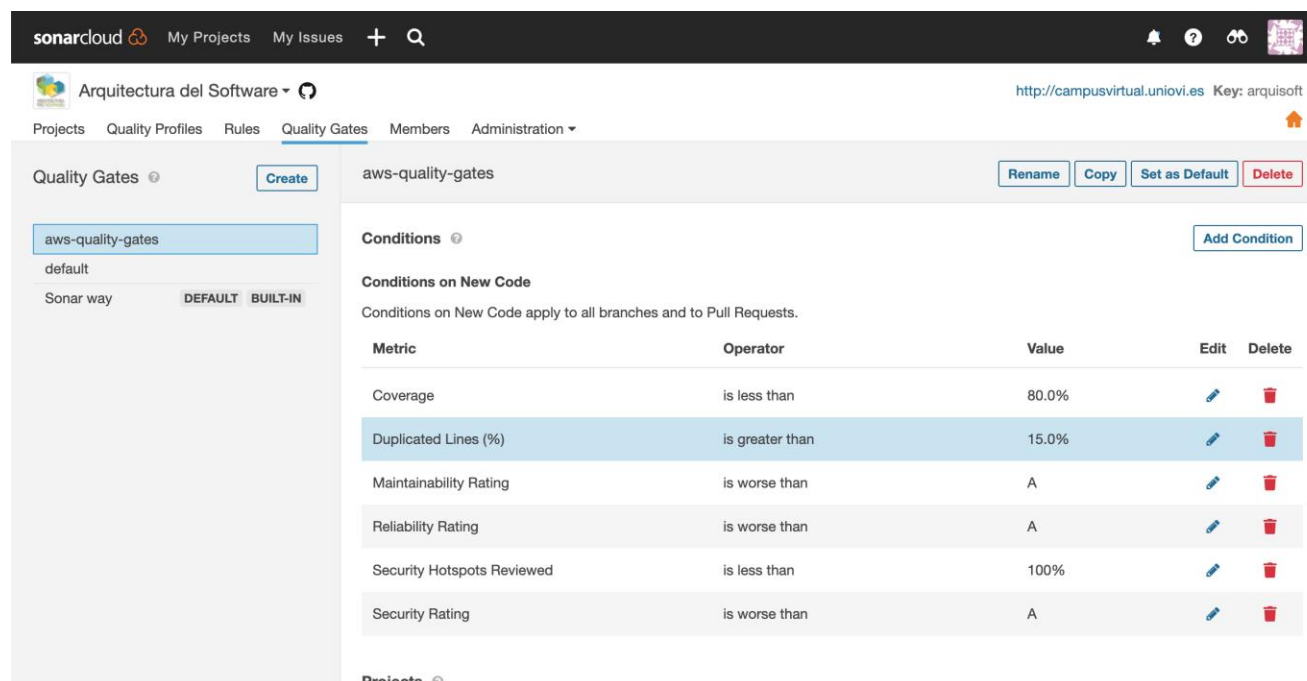


SonarCloud: Project certification and Quality evolution



SonarCloud: Quality Gates

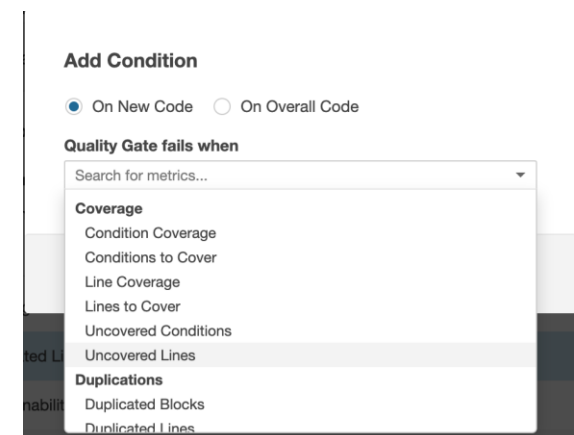
- At organization level, we can define the Quality Gates that our project must pass.



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

Below the table, there is a 'Projects' section.



The 'Add Condition' dialog box shows options for 'On New Code' (selected) and 'On Overall Code'. Below this, there is a search bar for metrics and a list of available metrics.

Add Condition

☒ On New Code ☐ On Overall Code

Quality Gate fails when

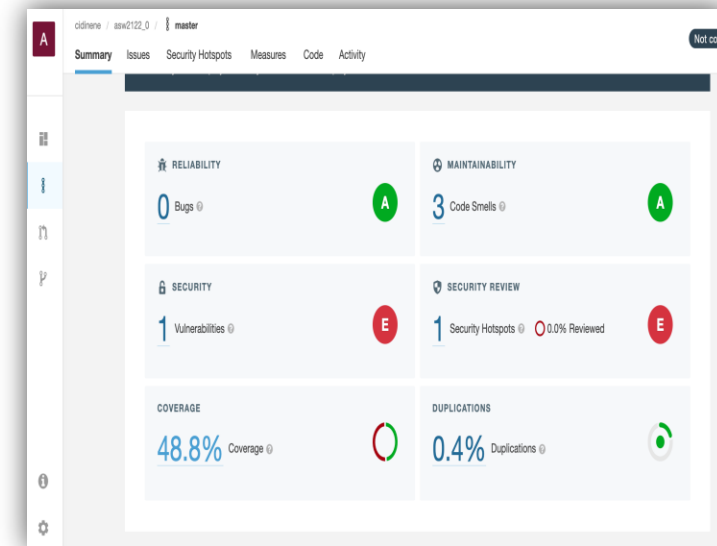
Search for metrics...

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

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, ..
- **Lo_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 image consists of two screenshots from the SonarCloud web interface, illustrating the process of creating and configuring a new profile.

Left Screenshot: Shows the 'Quality Profiles' page for the 'Arquitectura del Software' project. 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: 'Compare', 'Copy', 'Extend', and 'Set as Default'. The 'Copy' option is highlighted, indicating the step to create a new profile.

Right Screenshot: Shows the 'Rules' configuration page for the 'Sonar new Way' profile. The 'Rules' table is visible, showing a list of rules with columns for 'Active' and 'Inactive' counts. The 'Projects' section is also visible, showing that no projects are explicitly associated with the profile. The 'Change Projects' button is highlighted, indicating the step to associate the profile to the project.

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

🔍 📄 ☆ ⚙️ 👤 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

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

View alerts when coding

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

