SOFTWARE
ARCHITECTURE

Escuela de Ingeniería Informática

Universidad de Oviedo

Lab 6

TDD: Test-driven development
Code coverage (SonarCloud)
Continuous integration (GitHub Actions)
Static analysis tools (SonarCloud)

2024-25

Jose Emilio Labra Gayo
Pablo González
Cristian Augusto Alonso
Diego Martín

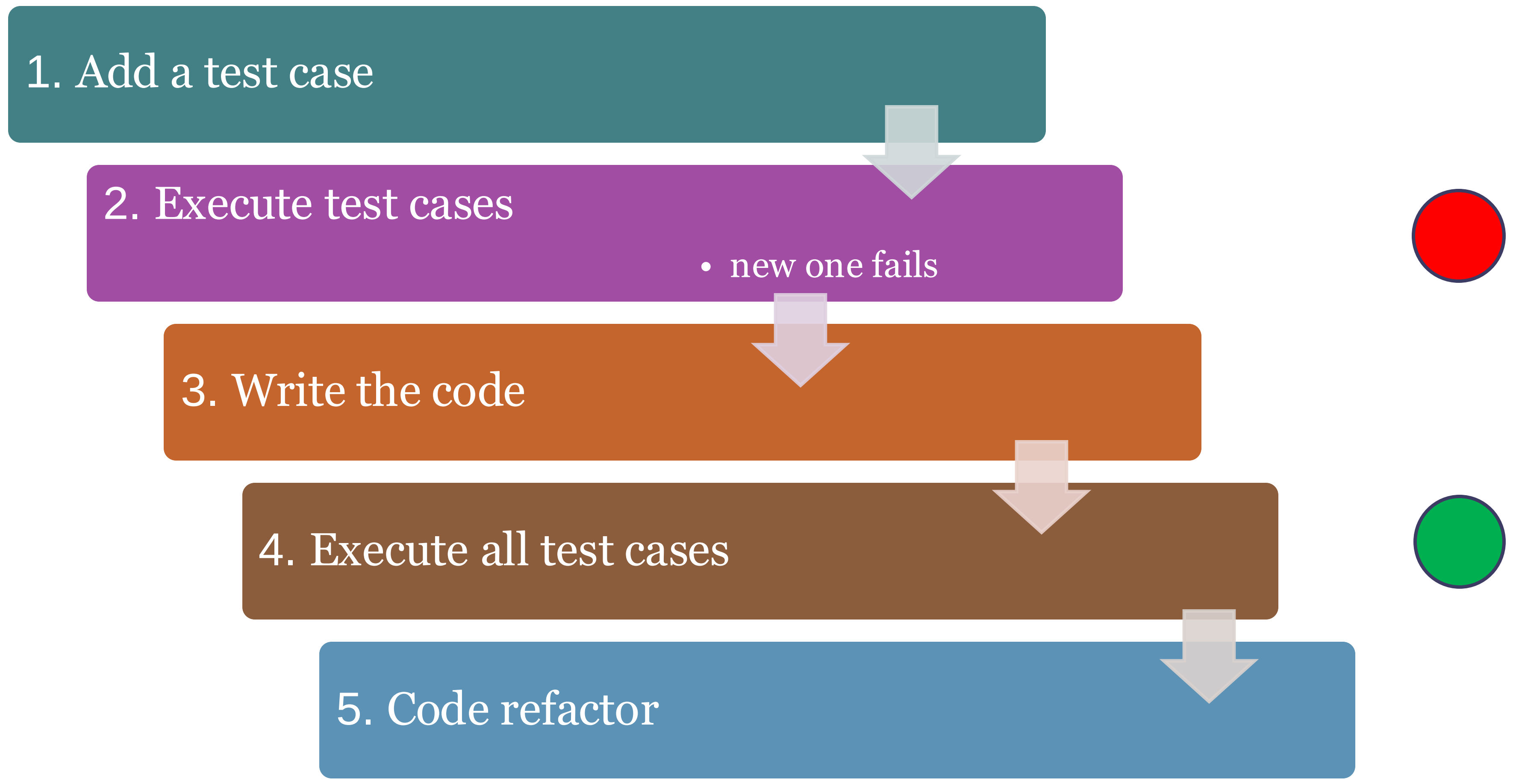
TDD

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 - Features

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

Code Coverage (SonarCloud)

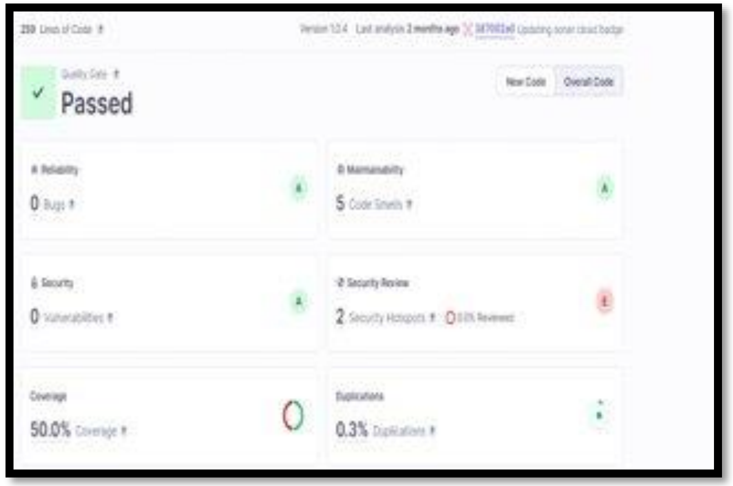
- Code coverage: Measure to show what code lines has been executed by a test suite
- Tool that includes code coverage as a metric in the code evaluation process
- Some terminology about SonarCloud:
  - LC: lines_to_cover – uncovered_lines
  - EL: lines_to_cover

Code Coverage in 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_wichat_???



259 Lines of Code

Version 1.0.4 Last analysis 2 months ago [illegible] Updating sonar cloud badge

Quality Gate

Passed

New Code | Overall Code

Reliability: 0 Bugs — A

Maintainability: 5 Code Smells — A

Security: 0 Vulnerabilities — A

Security Review: 2 Security Hotspots — 0.0% Reviewed — E

Coverage: 50.0% Coverage

Duplications: 0.3% Duplications

TDD – Example test

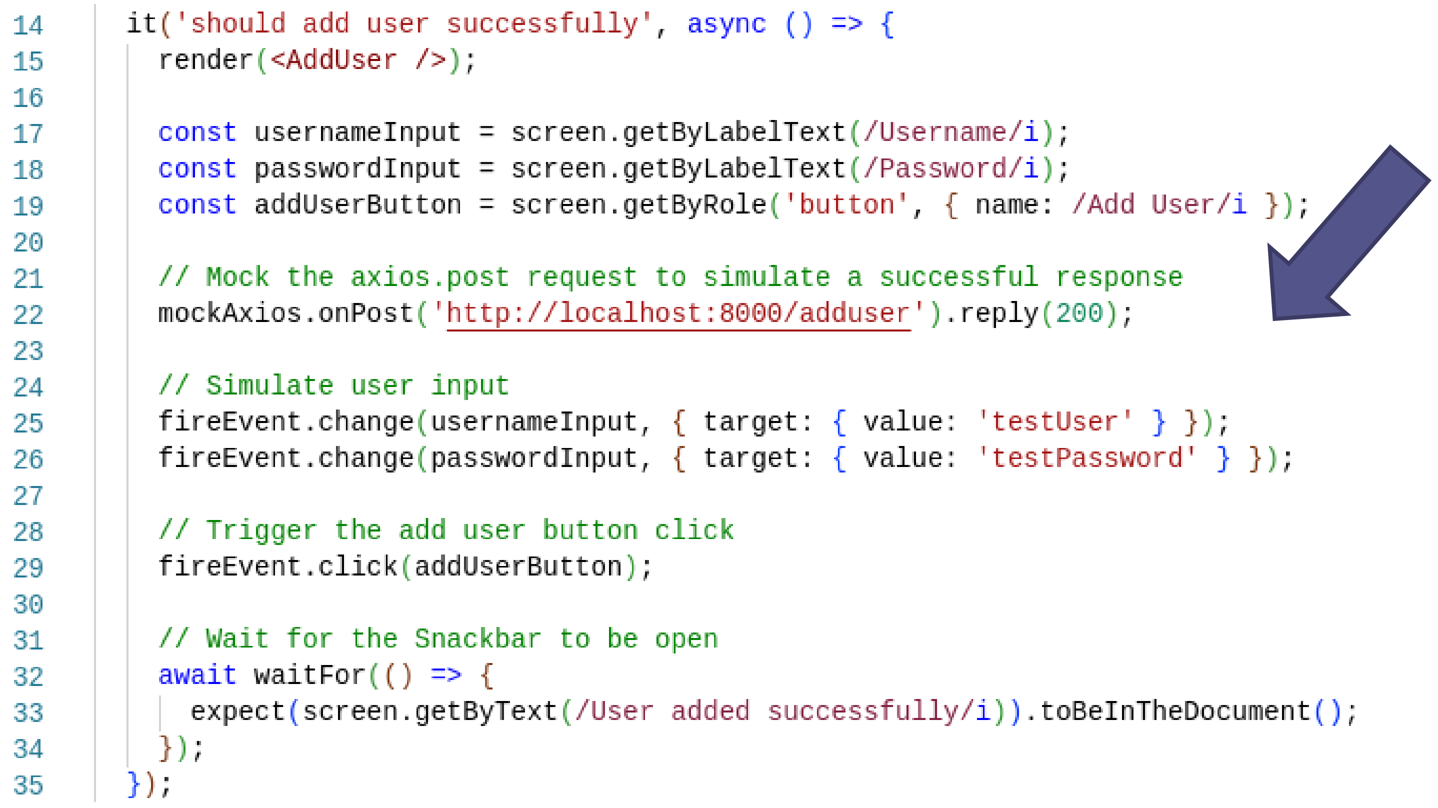
- Testing a basic component in React.js (App.test.js)

```
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 - 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



```
it('should add user successfully', async () => {
  render(<AddUser />);

  const usernameInput = screen.getByLabelText(/Username/i);
  const passwordInput = screen.getByLabelText(/Password/i);
  const addUserButton = screen.getByRole('button', { name: /Add User/i });

  // Mock the axios.post request to simulate a successful response
  mockAxios.onPost('http://localhost:8000/adduser').reply(200);

  // Simulate user input
  fireEvent.change(usernameInput, { target: { value: 'testUser' } });
  fireEvent.change(passwordInput, { target: { value: 'testPassword' } });

  // Trigger the add user button click
  fireEvent.click(addUserButton);

  // Wait for the Snackbar to be open
  await waitFor(() => {
    expect(screen.getByText(/User added successfully/i)).toBeInTheDocument();
  });
});
```

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 Continuun
  - 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.

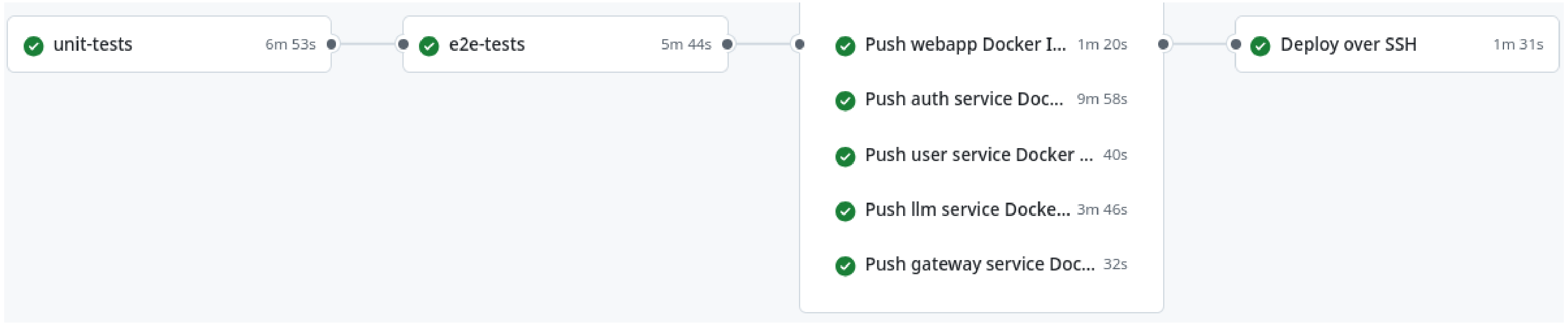
GitHub Actions

- 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)

```yaml
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 gatewayservice 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 gatewayservice test -- --coverage
    - run: npm --prefix webapp test -- --coverage
    - name: Analyze with SonarQube
      uses: SonarSource/sonarqube-scan-action@master
      env:
        SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```



✓ unit-tests 6m 53s

✓ e2e-tests 5m 44s

✓ Push webapp Docker I... 1m 20s

✓ Push auth service Doc... 9m 58s

✓ Push user service Docker ... 40s

✓ Push llm service Docke... 3m 46s

✓ Push gateway service Doc... 32s

✓ Deploy over SSH 1m 31s

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

```
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
```

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

Static Analysis - 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, Visal Studio, Eclipse…)

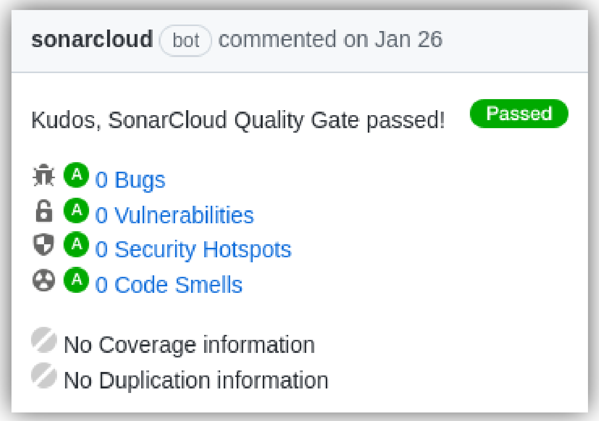
The análisis is performed locally (before the changes are submitted to the repository), can be executed:

- 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 bot commented on Jan 26

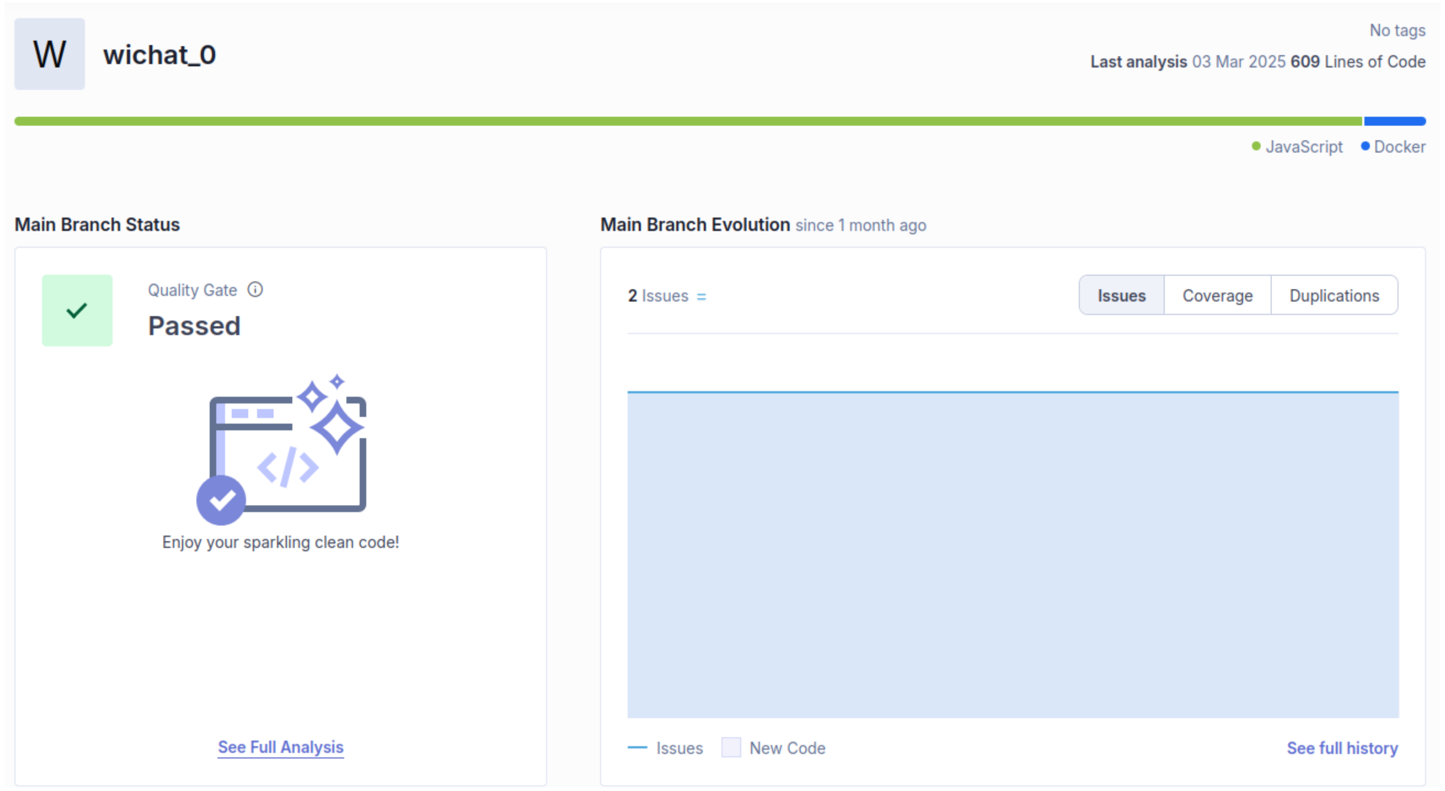
Kudos, SonarCloud Quality Gate passed! Passed

A 0 Bugs
A 0 Vulnerabilities
A 0 Security Hotspots
A 0 Code Smells

No Coverage information
No Duplication information

SonarCloud

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


W wichat_0

No tags

Last analysis 03 Mar 2025 609 Lines of Code

● JavaScript ● Docker

Main Branch Status

Quality Gate

Passed

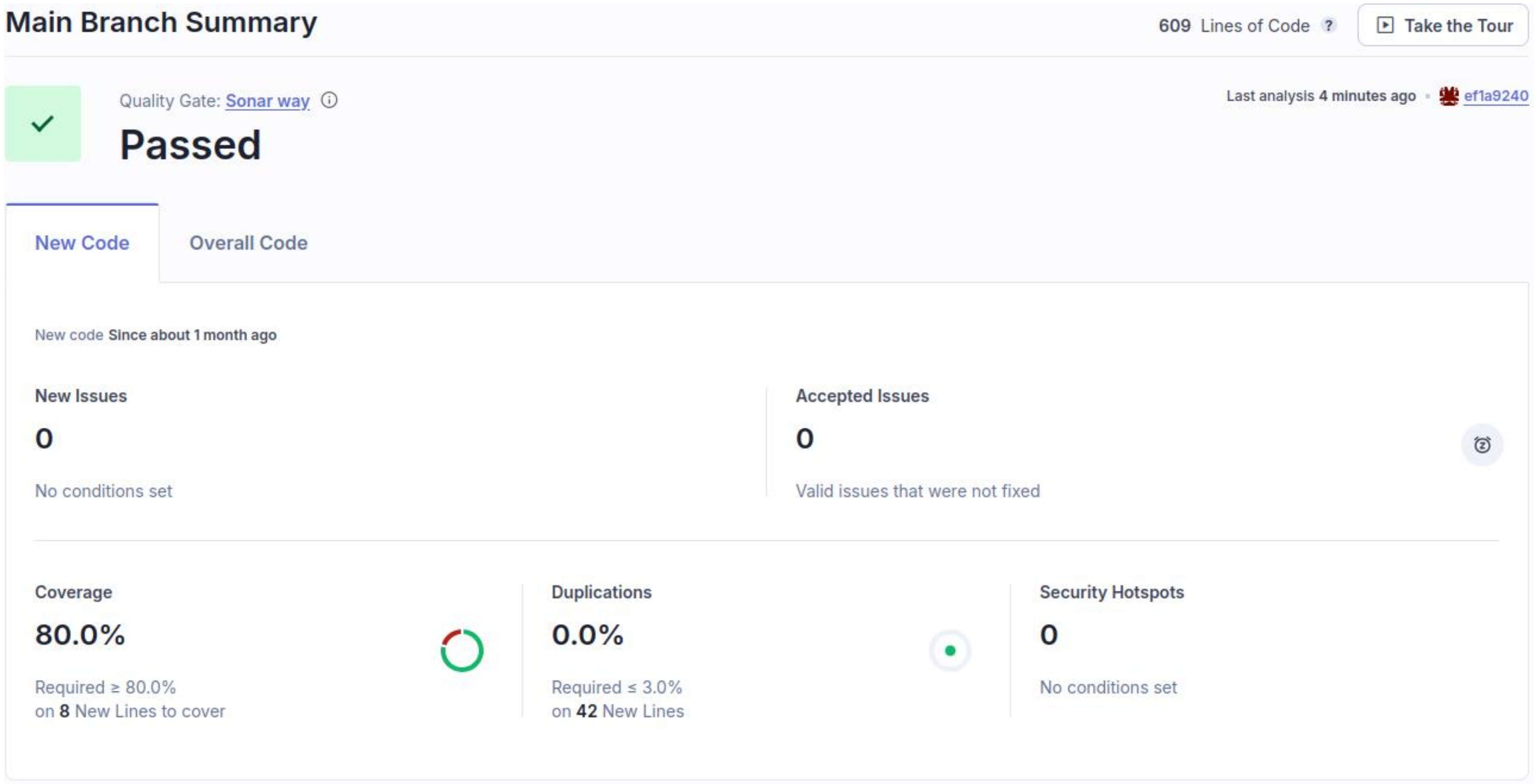
Enjoy your sparkling clean code!

See Full Analysis

Main Branch Evolution since 1 month ago

2 Issues =

SonarCloud: Project certification and Quality evolution



Main Branch Summary

609 Lines of Code ? | Take the Tour

Quality Gate: Sonar way

Passed

Last analysis 4 minutes ago · ef1a9240

New Code | Overall Code

New code Since about 1 month ago

New Issues
0
No conditions set

Accepted Issues
0
Valid issues that were not fixed

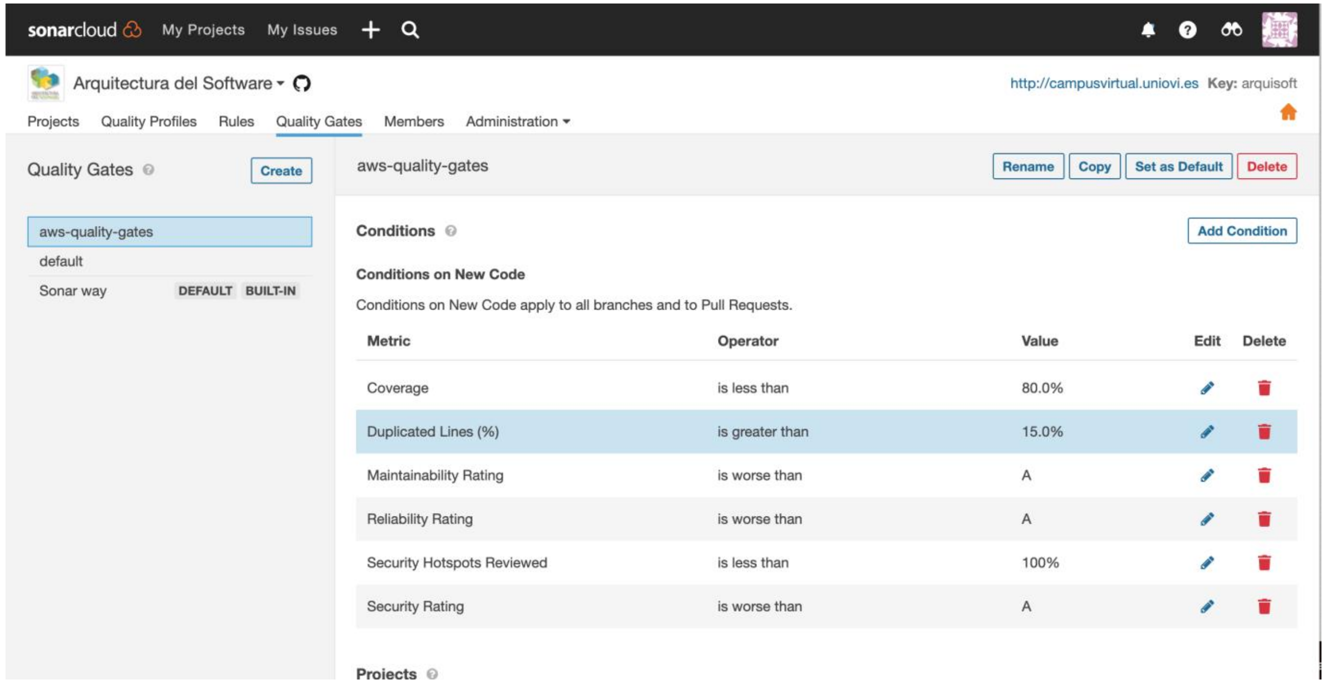
Coverage
80.0%
Required ≥ 80.0%
on 8 New Lines to cover

Duplications
0.0%
Required ≤ 3.0%
on 42 New Lines

Security Hotspots
0
No conditions set

SonarCloud: Quality Gates

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



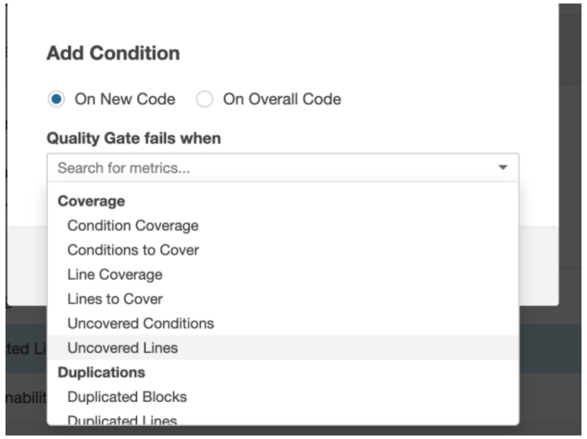
aws-quality-gates

Conditions

Conditions on New Code

Conditions on New Code apply to all branches and to Pull Requests.

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		



Add Condition

On New Code [✓] · On Overall Code

Quality Gate fails when

Search for metrics...

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, ..

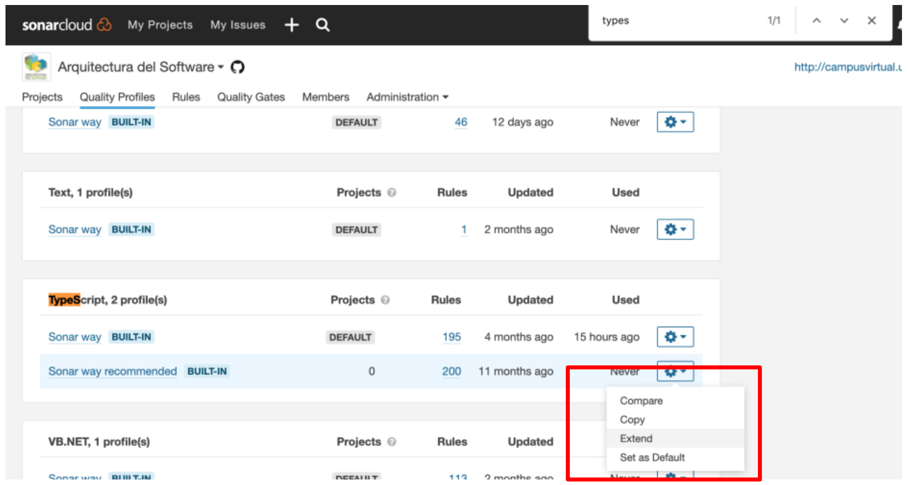
wichat_0 default project uses code coverage with SonarCloud

SonarCloud: Profiles and Rules

Rules are defined at profile level

We can add, desactivate, update rules creating a new profile :

Copy a parent profile - change it - associate it to the project



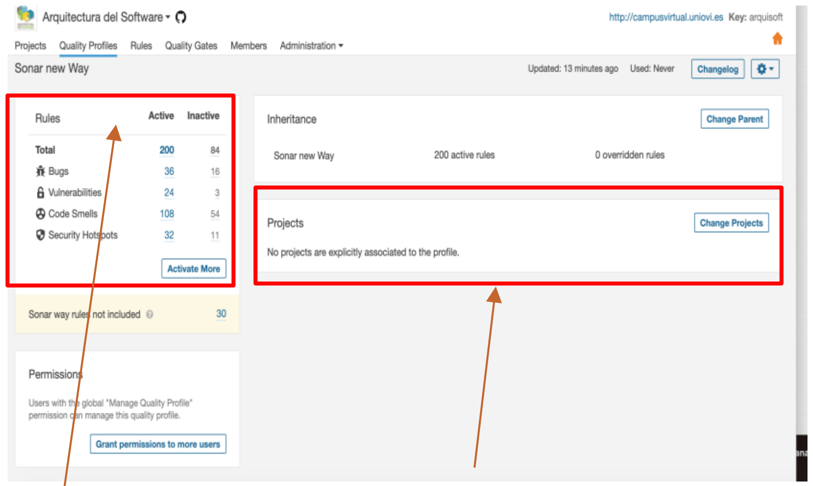
	Projects	Rules	Updated	Used
Sonar way BUILT-IN	DEFAULT	46	12 days ago	Never

Text, 1 profile(s)	Projects	Rules	Updated	Used
Sonar way BUILT-IN	DEFAULT	1	2 months ago	Never

TypeScript, 2 profile(s)	Projects	Rules	Updated	Used
Sonar way BUILT-IN	DEFAULT	195	4 months ago	15 hours ago
Sonar way recommended BUILT-IN	0	200	11 months ago	Never

VB.NET, 1 profile(s)	Projects	Rules	Updated
Sonar way BUILT-IN	DEFAULT	113	2 months ago

Create a new profile



Sonar new Way

Updated: 13 minutes ago Used: Never

Rules	Active	Inactive
Total	200	84
Bugs	36	16
Vulnerabilities	24	3
Code Smells	108	54
Security Hotspots	32	11

Sonar way rules not included 30

Inheritance

Sonar new Way 200 active rules 0 overridden rules

Projects

No projects are explicitly associated to the profile.

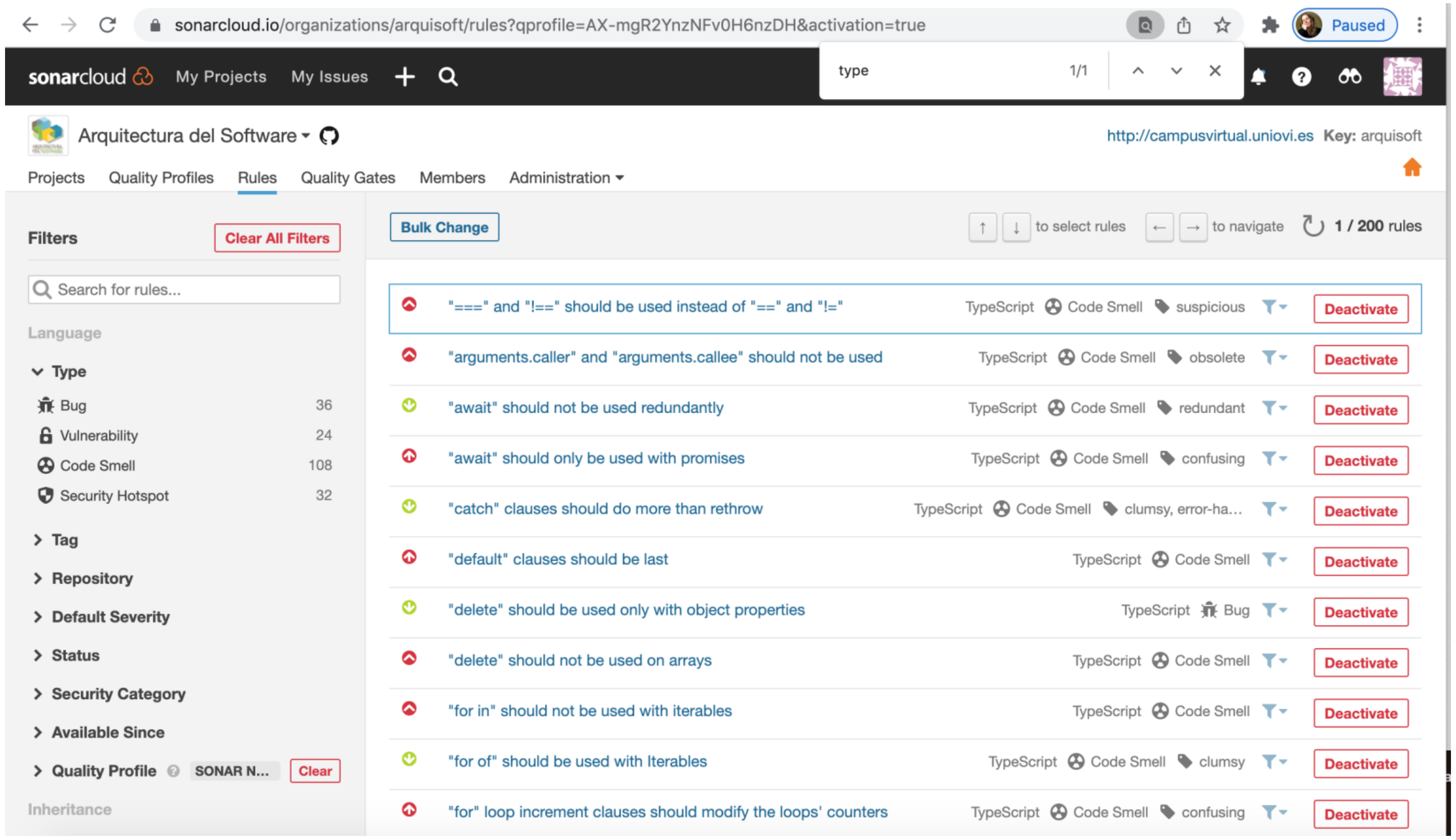
Permissions

Users with the global "Manage Quality Profile" permission can manage this quality profile.

Set the profile rules

Associate the profile to the project

Rules configuration



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

Arquitectura del Software

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

Type

Bug	36
Vulnerability	24
Code Smell	108
Security Hotspot	32

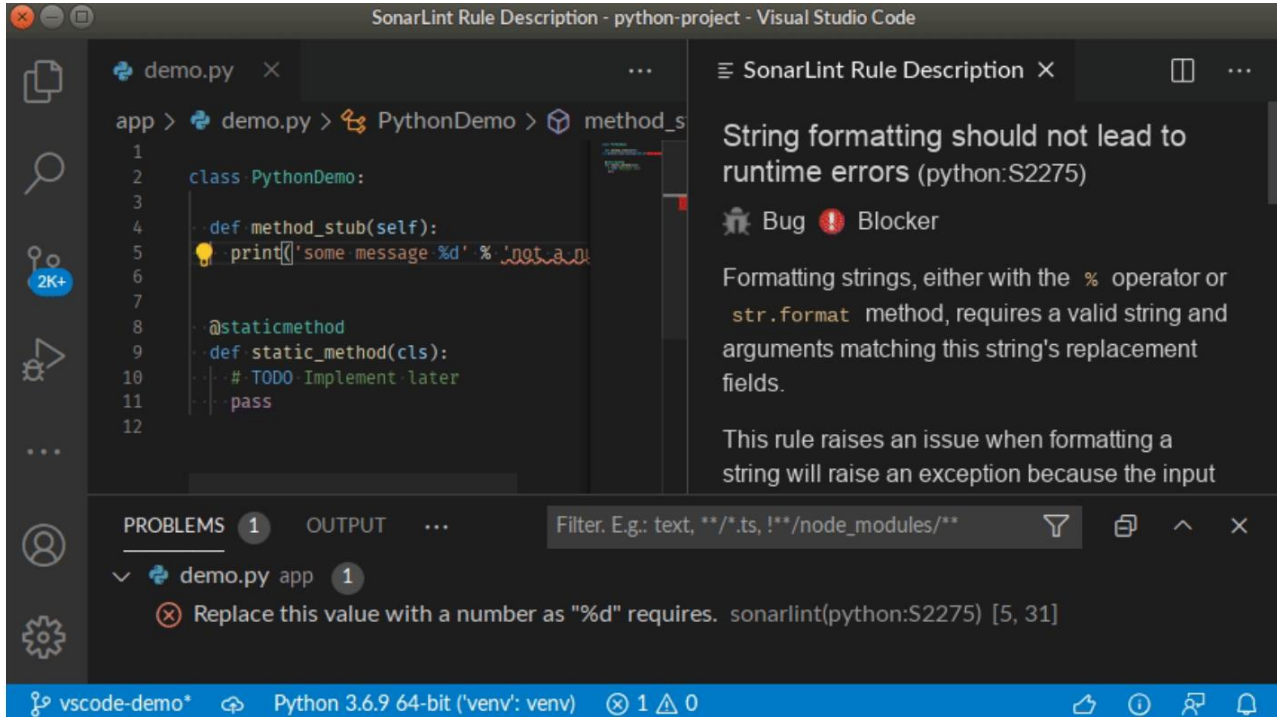
Quality Profile SONAR N... Clear

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

View alerts when coding

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


SonarLint Rule Description - python-project - Visual Studio Code
demo.py
app > demo.py > PythonDemo > method_s
SonarLint Rule Description
PROBLEMS 1
OUTPUT
Filter. E.g.: text, **/*.ts, !**/node_modules/**
vscode-demo*
Python 3.6.9 64-bit ('venv': venv)
1 0

```python
class PythonDemo:

    def method_stub(self):
        print('some message %d' % 'not a nu

    @staticmethod
    def static_method(cls):
        # TODO Implement later
        pass
```

String formatting should not lead to runtime errors (python:S2275)

Bug Blocker

Formatting strings, either with the `%` operator or `str.format` method, requires a valid string and arguments matching this string's replacement fields.

This rule raises an issue when formatting a string will raise an exception because the input

demo.py app 1

Replace this value with a number as "%d" requires. sonarlint(python:S2275) [5, 31]