



Trabajo 3. Mantenibilidad y refactorización.
SonarCloud y GitHub Actions

EVOLUCIÓN Y MANTENIMIENTO DEL SOFTWARE

©2024

Ángel Panizo & Juan Manuel Garitagoitia & Jessica Díaz Fernández

1 Objetivos del trabajo

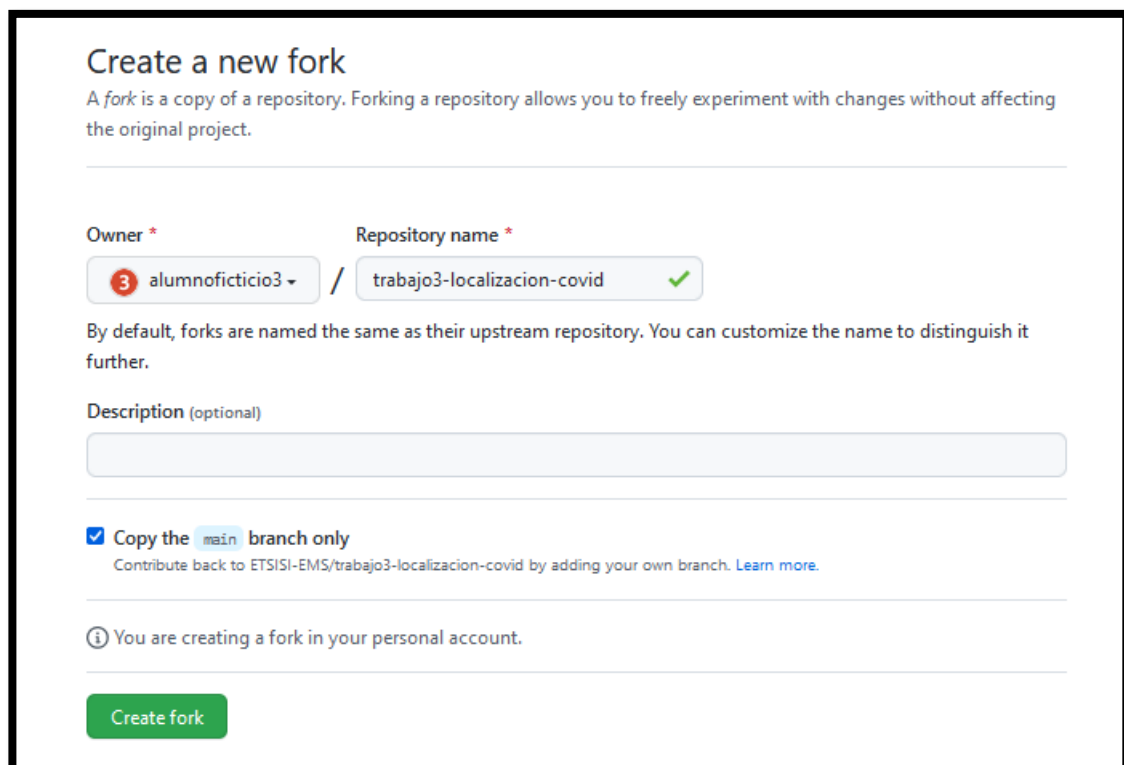
El alumno aplicará conceptos sobre principios de diseño de software mantenible, identificación de code smells (deterioro del código) y técnicas de refactorización del código fuente. También usará los conocimientos de control de versiones y CI aprendidos durante la asignatura.

2 Introducción

En esta práctica vamos a simular que una consultora contrató a los alumnos de EMS para mejorar la mantenibilidad de uno de sus sistemas Java. Para ello se proponen seguir los 10 principios de mantenibilidad vistos en clase. Primero, actuando como **DevOps engineer**, los alumnos deberán preparar un pipeline capaz de analizar la calidad del código para que, en el caso de no alcanzar unos mínimos de calidad, se prevenga su incorporación a la rama principal. Segundo, actuando como **desarrolladores**, los alumnos deberán probar su pipeline y refactorizar el código para que cumpla los estándares de calidad establecidos previamente.

3 Obtención del repositorio de código

Se recomienda realizar este trabajo en equipo. Los equipos trabajarán con un fork del siguiente repositorio GitHub <https://github.com/ETSISI-EMS/trabajo3-localizacion-covid>



Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner * Repository name *

alumnoficticio3 / trabajo3-localizacion-covid ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

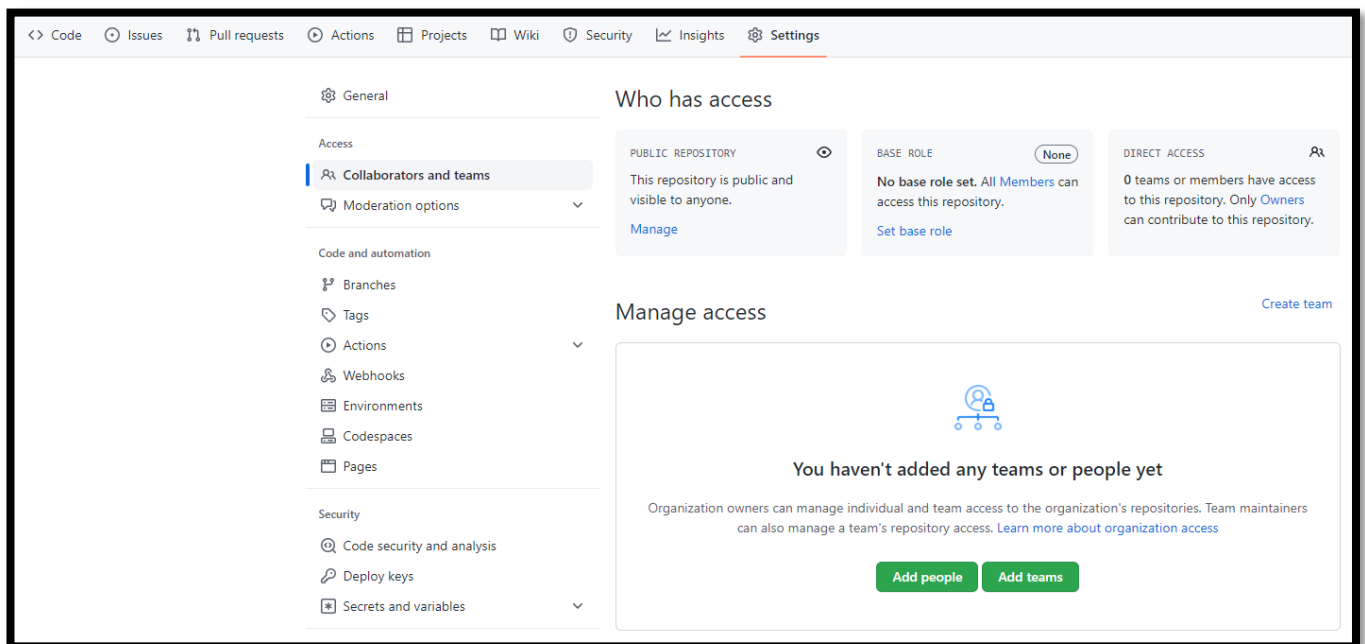
Description (optional)

☒ Copy the `main` branch only
Contribute back to ETSISI-EMS/trabajo3-localizacion-covid by adding your own branch. [Learn more.](#)

i You are creating a fork in your personal account.

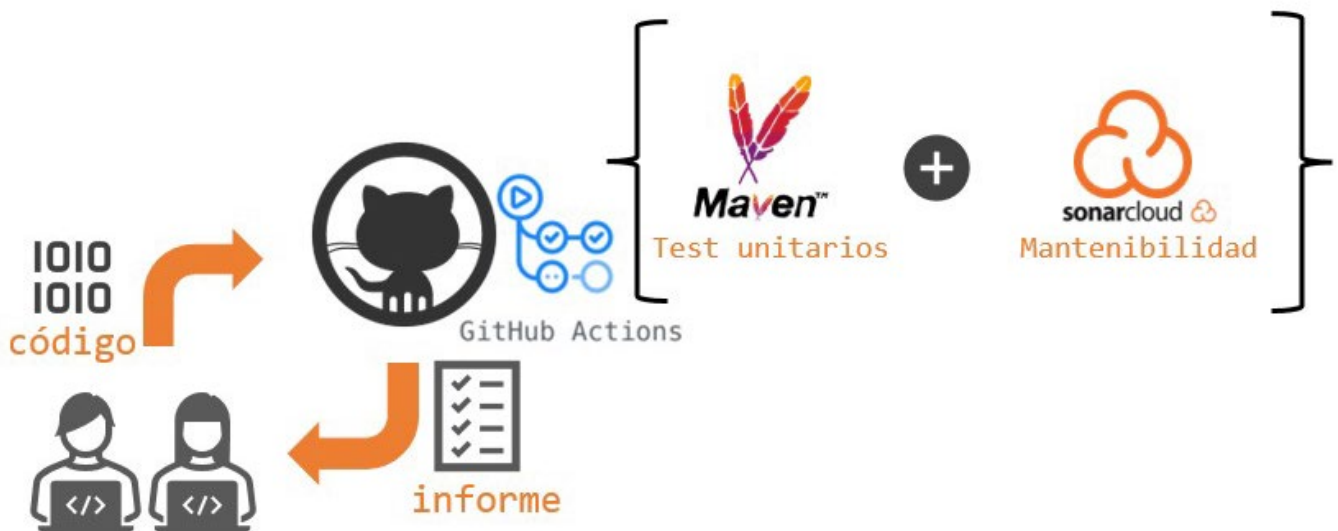
Create fork

Para trabajar en equipo, el propietario tiene que invitar a colaborar al resto de miembros del equipo. Para ello, en las configuraciones (settings) del repositorio, seleccionar “Collaborators and teams” y “Add people”.



4 Parte 1: Preparación del pipeline

Este trabajo propone el uso de **GitHub** para el hosting de repositorios Git, **GitHub Actions** como herramienta de CI, y **Maven** y **JUnit**¹ para automatizar ciertas tareas. Como novedad, en este trabajo se introduce el uso del servicio en la nube **SonarCloud** para comprobar la mantenibilidad y calidad del código. A continuación, se muestra un esquema del pipeline de CI y la parte del fichero pom.xml que contiene ciertas configuraciones que el alumnado debe reconocer y/o modificar durante la realización del trabajo. Es particularmente relevante la configuración del nombre de la organización `<sonar.organization>etsisi-ems</sonar.organization>` que coincidirá tanto en GitHub como en Sonar. En esta ocasión, por temas organizativos del alumnado, no utilizará la organización ETSISI-EMS, si no que hará uso de su propio espacio (cuenta de usuario).



A continuación, se muestra un extracto del fichero pom.xml

¹ El alumnado no tendrá que especificar ningún test ya que se dan definidos y listos para ser ejecutados en el pipeline de CI con el objetivo de verificar que las refactorizaciones introducidas por el alumnado no han cambiado la funcionalidad básica de la aplicación.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.practica</groupId>
  <artifactId>trabajo3-localizacion-covid</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <name>trabajo3-localizacion-covid</name>
  <url>http://www.example.com</url>

  <properties>
    <java.version>21</java.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>21</maven.compiler.source>
    <maven.compiler.target>21</maven.compiler.target>
    <sonar.organization>etsisi-ems</sonar.organization>
    <sonar.host.url>https://sonarcloud.io</sonar.host.url>
  </properties>

```

Coincide con el nombre que se le asignará al proyecto en SonarCloud

Se configurará con la cuenta de usuario (u organización) del alumno que crea el proyecto SonarCloud y que compartirá con los demás miembros del equipo

4.1 SonarCloud

Como herramienta para medir la calidad del código se usará *SonarCloud*² que es una herramienta SaaS (Software as a Service) que realiza un *análisis estático* del código, es decir, analiza el código fuente, no así el comportamiento dinámico, esto es, el programa en ejecución. SonarCloud es un software muy completo que permite identificar tanto *code smells* como problemas de seguridad y rendimiento en más de 29 lenguajes de programación. SonarCloud cuenta con tres herramientas para controlar la calidad del código:

- **Rules:** Las reglas sirven para detectar un problema en el código, ya sea un *code smell*, un problema de seguridad... etc. Por ejemplo, una regla puede contar el número de parámetros de una función y avisar si supera cierto tamaño. Otro ejemplo podría ser comprobar si hay un "import" no que se use.
- **Quality Profiles:** Son conjuntos de reglas que se aplican en un proyecto concreto. Cada organización, o incluso proyecto, tiene unas características concretas y las reglas aplicadas no tienen por qué ser iguales. A través de este mecanismo, SonarCloud permite personalizar nuestro negocio, por ejemplo, podemos definir un quality profile que analice problemas de seguridad en el código y omita el análisis de *code smells* o donde solo se compruebe que no haya código muerto ignorando todo lo demás.
- **Quality Gates:** Son condiciones que marcan la calidad mínima que deben cumplir nuestros proyectos y están asociadas a un quality profile concreto. Por ejemplo, una quality gate puede medir que no haya ningún problema de seguridad en nuestro código o que el número de *code smells* sea menor que un threshold.

² <https://www.sonarsource.com/products/sonarcloud>

4.2 Principios de mantenibilidad

En concreto, en la práctica tendremos que crear un *Quality Profile* y *Quality Gates* para cumplir los siguientes principios de mantenibilidad:

- Write short units of code:
 1. Un método no debería tener más de 30 líneas de código. *Methods should not have too many lines*
 2. Las clases no deberían tener demasiados atributos (max 20). *Classes should not have too many fields*
 3. Las clases no deberían tener demasiados métodos (max 35). *30 líneas/método * 35 = 1050 líneas por clase*
 4. Los ficheros de código no deberían tener muchas líneas. *Files should not have too many lines of code*
- Write simple units of code:
 5. Un método no debería tener una complejidad ciclomática mayor que 10. *Methods should not be too complex*
 6. Un método no debería tener una complejidad cognitiva mayor que 15. *Cognitive Complexity of methods should not be too high*
 7. Evitar expresiones demasiado complejas. *Expressions should not be too complex*
 8. Evitar if-then-else anidados. *Boolean expressions should not be gratuitous*
 9. Las sentencias "switch" no deben tener demasiadas cláusulas "case" (más de 4) – deberían sustituirse por estructuras de datos. *Ternary operators should not be nested*
Switch statements should not have too many case clauses
- Write code once
 10. No deberían existir bloques de código repetidos. Un bloque de código está constituido por 10 líneas. *Two branches in a conditional structure should not have exactly the same implementation*
 11. Diferentes métodos no deberían tener implementación idéntica. *Methods should not have identical implementations*
- Keep interfaces small:
 12. Un método no debería tener demasiados parámetros (en general no más de 5, constructores podrían tener hasta 7). *Methods should not have too many parameters*
- Clean code:
 13. No puede haber métodos privados que no se usen (no sean llamados desde ninguna parte del código). *Unused "private" methods should be removed*
 14. No puede haber atributos que no se usen. *Unused "private" fields should be removed*
 15. No puede haber variables locales que no se usen. *Unused local variables should be removed*
 16. No puede haber parámetros que no se usen. *Unused method parameters should be removed*
Unused type parameters should be removed
 17. No puede haber "imports" que no se usen. *Unnecessary imports should be removed*
 18. No puede haber código comentado. *Sections of code should not be commented out*
 19. Las constantes deben estar escritas en mayúsculas. *Constant names should comply with a naming convention*
^[A-Z][A-Z0-9](_[A-Z0-9]+)*\$ = con guión bajo intermedio*
 20. Los nombres de las clases deben empezar por mayúsculas y seguir una nomenclatura camel case.
 21. Los nombres de los atributos deben empezar en minúsculas y seguir una nomenclatura camel case.
 22. Los nombres de los métodos deben empezar en minúsculas y seguir una nomenclatura camel case.

4.3 Pipeline de CI

En este trabajo usaremos Github Actions para implementar CI. Vamos a configurar un pipeline para que ejecute un análisis en SonarCloud y verifique que se cumplen todas las Quality Gates configuradas. Es decir, el pipeline fallará (no genere un artefacto) si no se cumple alguna de las Quality Gates configuradas. A grandes rasgos los pasos a seguir son los siguientes:

20. Class names should comply with a naming convention
21. Field names should comply with a naming convention
22. Method names should comply with a naming convention

^[A-Z][a-zA-Z0-9]\$ = cadenas de caracteres que empiezan por mayúsculas y contienen exclusivamente caracteres en mayúscula o minúscula y números*

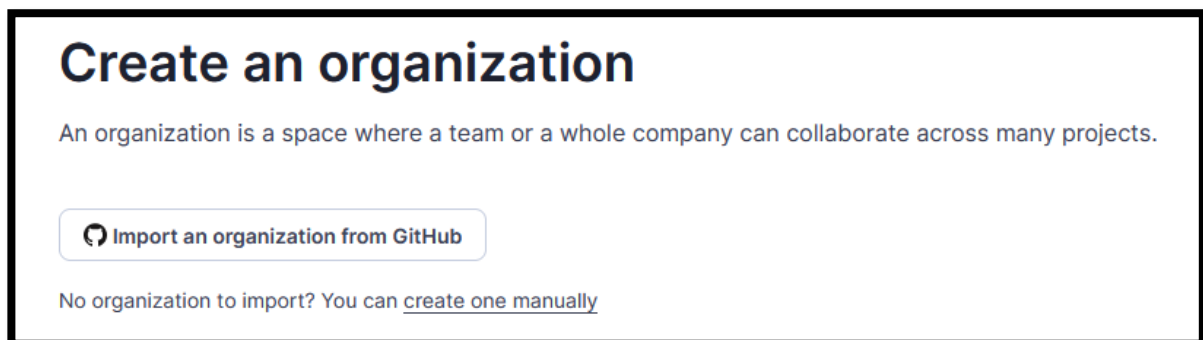
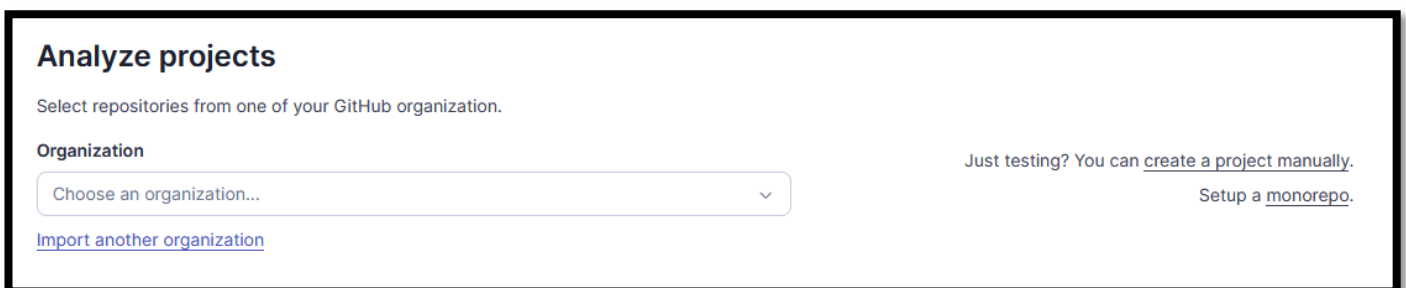
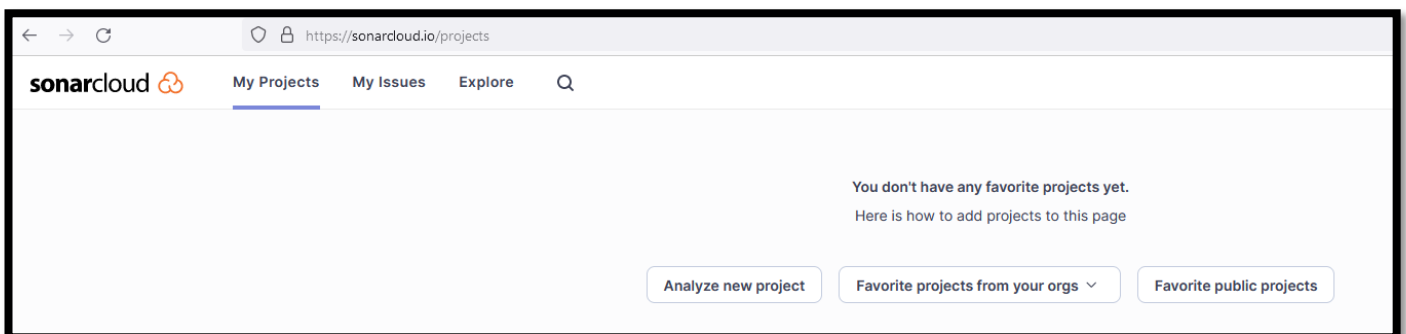
1. [Configuración de los principios de mantenibilidad en SonarCloud](#)
 - A. Autenticación en SonarCloud
 - B. Configurar el repositorio GitHub a analizar.
 - C. Configurar permisos en SonarCloud
 - D. Configurar un Quality Profile
 - E. Configurar Quality Gates
2. [Configuración de un Pipeline de CI con GitHub Actions y SonarCloud](#)
 - F. Activar en Sonar CI con GitHub Actions
 - G. Configurar Secretos para el pipeline en GitHub Actions
 - H. Definir pipeline en GitHub Actions
3. [Ejecución del Pipeline de CI](#)

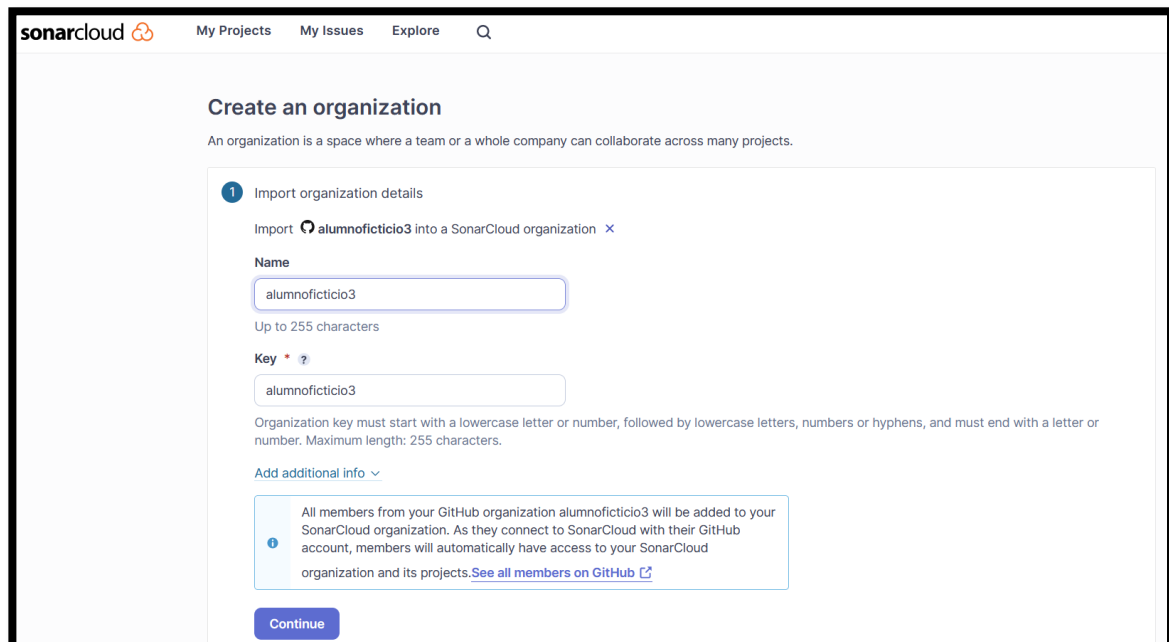
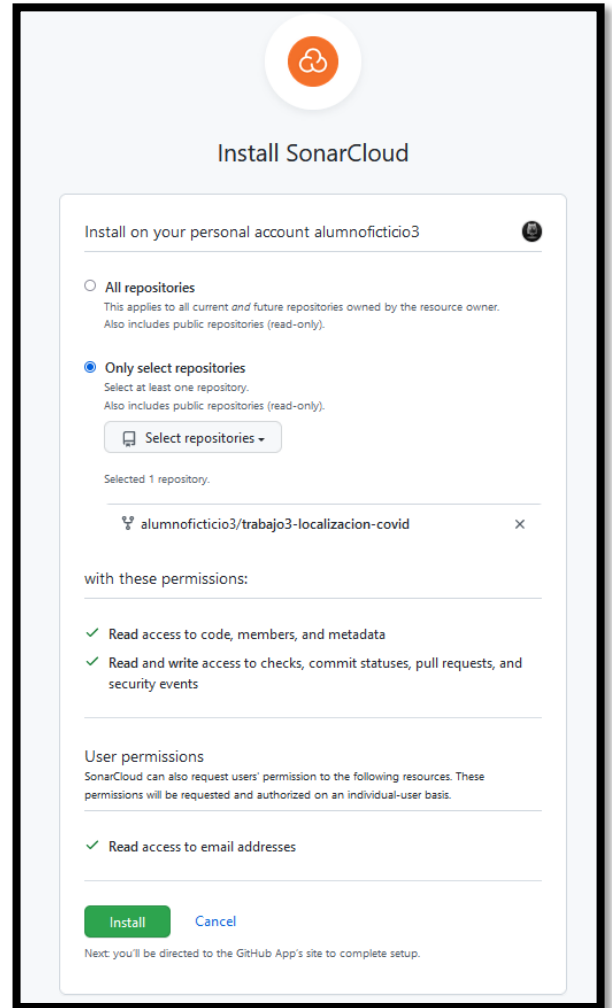
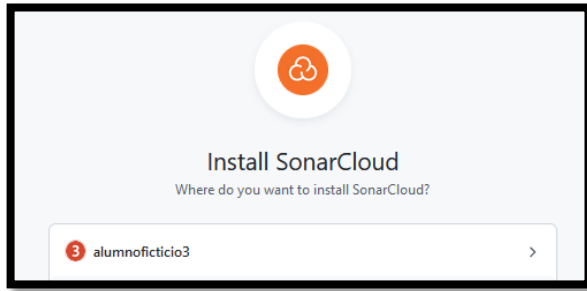
A continuación, se detallan cada uno de estos pasos:

4.3.1 Configuración de los principios de mantenibilidad en SonarCloud

A. Autenticación en SonarCloud

1. En <https://sonarcloud.io/login> autorizamos enlazar nuestra cuenta de GitHub con SonarCloud.
2. En la página de inicial, “Analyze new project”, click en el enlace Import another organization, botón “Import an organization from GitHub”. A continuación, seleccionar la cuenta de usuario para instalar SonarCloud. El alumno podría elegir entre instalar SonarCloud en todos sus repositorios o elegir únicamente el repositorio usado en este trabajo.





Create an organization

An organization is a space where a team or a whole company can collaborate across many projects.

1 Import organization details

alumnoficticio3

2 Choose a plan

☒ **Free plan** 0 €
 All projects you analyze will be public.
 Anyone can browse source code.

☐ **Paid plan** from 10 €
 ✓ Unlimited private projects
 ✓ Strict control over who can view your private data
 ✓ No commitments, cancel anytime
 ✓ 14 days free trial.
[Learn more](#)

Create Organization

B. Configurar el repositorio GitHub a analizar

1. Seleccionar en el menú el repositorio de GitHub a analizar y click en botón “Set Up”.

Analyze projects - Select repositories

Organization *

alumnoficticio3 alumnoficticio3 [Import another organization](#)

☒ trabajo3-localizacion-covid

Don't see your repo? Check your [GitHub app configuration](#).

1 repository selected
 1 repository will be created as a public project on SonarCloud

Set Up

Just testing? You can [create a project manually](#).
 Setup a [monorepo](#).

2. Configurar lo que va a ser considerado “código nuevo”

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code.

This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology.

Learn more: [New Code Definition](#)

Set a new code definition for your organisation to use it by default for all new projects
 This can help you use the Clean as You Code methodology consistently across projects.
[alumnoficticio1 - Administration - New Code](#)

The new code for this project will be based on:

☒ **Previous version**
 Any code that has changed since the previous version is considered new code.
 Recommended for projects following regular versions or releases.

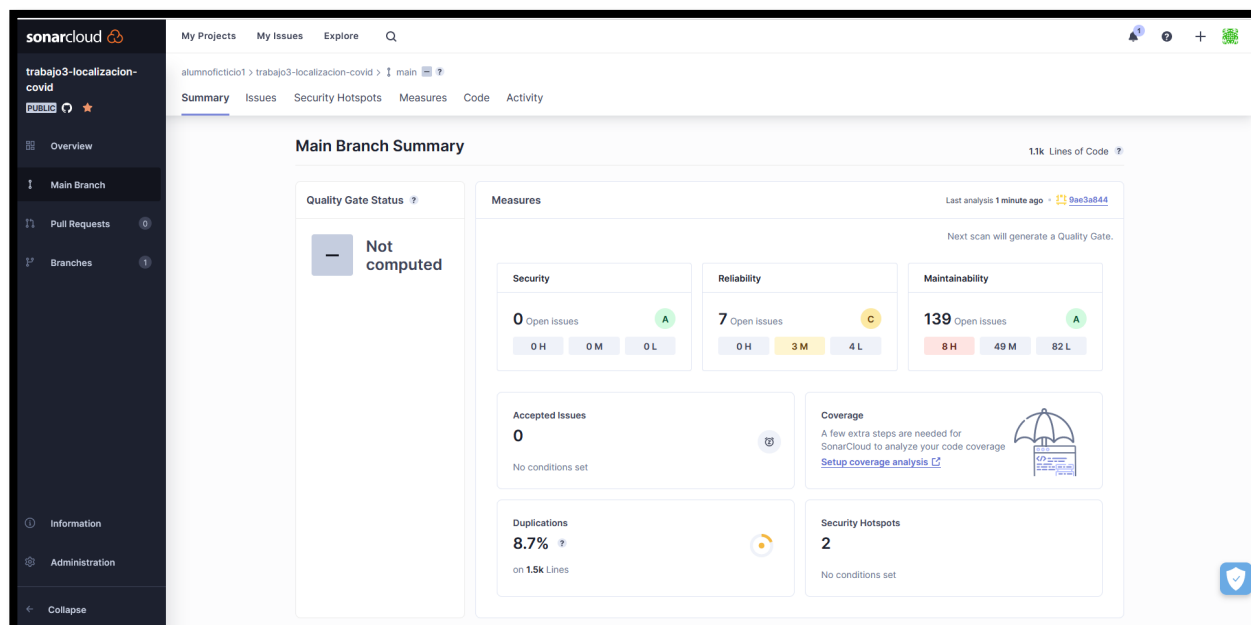
☐ **Number of days**
 Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
 Recommended for projects following continuous delivery.

You can change this at any time in the project administration

Back

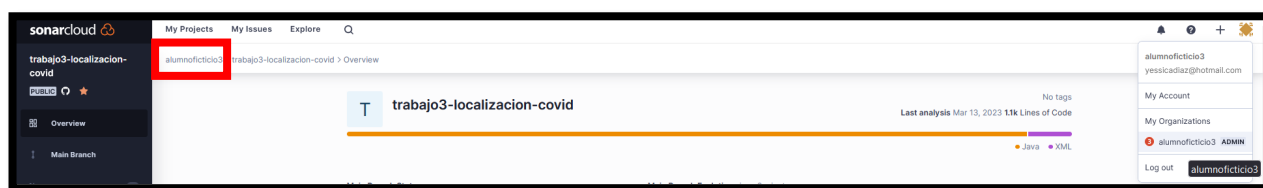
Create project

Sonar Cloud ofrece un primer análisis para el proyecto seleccionado (trabajo3-localizacion-covid), pero el objetivo del estudiante es configurar este análisis de acuerdo a las políticas y principios definidos en este trabajo.

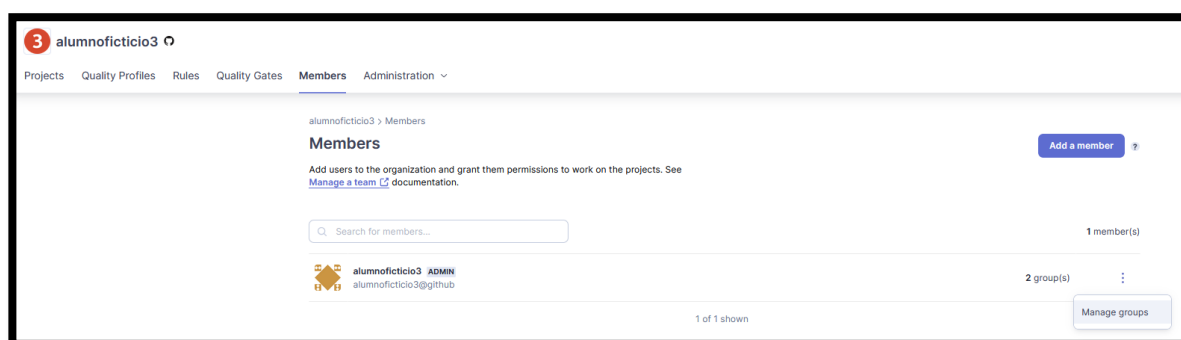


C. Configurar permisos en SonarCloud

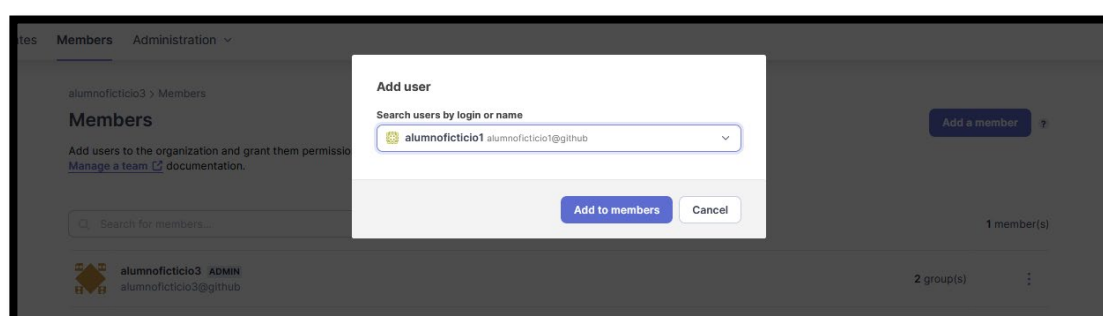
1. Volvemos a la pestaña de la organización, bien desde el menú Account o bien navegando en el path



2. Pestaña de “Members”, buscamos nuestro usuario. Click “Manage groups” y comprobar que tenemos marcada la opción de “Owners”.

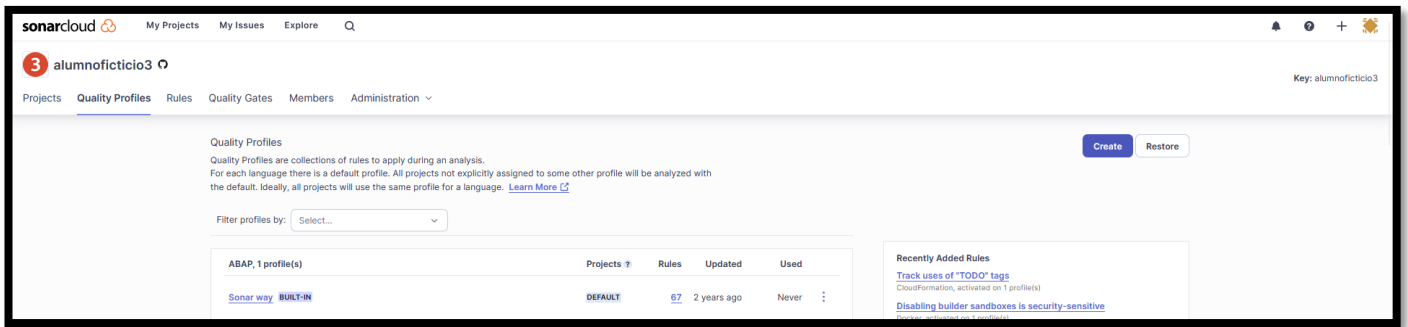


3. Añadir como miembros y/o propietarios al resto de miembros del equipo



D. Configurar un Quality Profile

1. Pestaña Quality Profile. Click en “Create” y rellenar el formulario con los siguientes datos

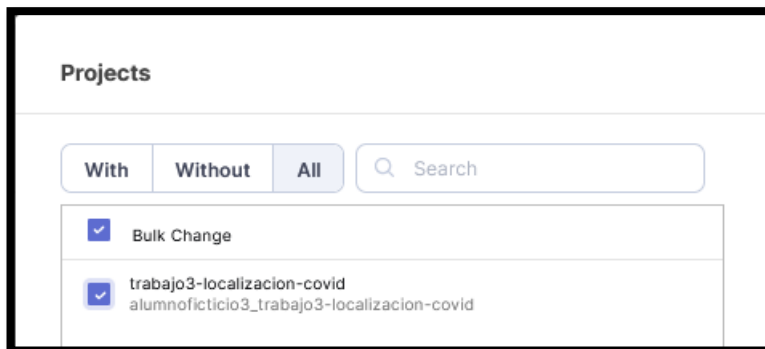


The 'New Profile' form is shown. It has the following fields:

- Name ***: Text input with 'Lab3' entered.
- Language ***: Dropdown menu with 'Java' selected.
- Parent:**: Dropdown menu with 'Select...' selected.

At the bottom, there are 'Create' and 'Cancel' buttons.

2. Seleccionar el proyecto pulsando en “Change Projects” y elegir de la lista (OJO! Si no aparece tu proyecto, tienes que buscarlo en la pestaña de All). Si es necesario para colaborar en equipo, configurar permisos en el botón “Grant permissions to more users”.



alumnoficticio1 > Quality Profiles > Java

Lab3

Updated: 2 minutes ago Used: Never [See Changelog](#)

Inheritance [Change Parent](#)

Lab3	0 active rules	699 inactive rules	0 overridden rules
------	----------------	--------------------	--------------------

Projects [Change Projects](#)

[trabajo3-localizacion-covid](#)

1 of 1 shown

Permissions

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

[Grant permissions to more users](#)

Rule Breakdown

Clean Code attributes	Active	Inactive
Consistency	0	135
Intentionality	0	453
Adaptability	0	55
Responsibility	0	18

Software qualities	Active	Inactive
Security	0	56
Reliability	0	178
Maintainability	0	434

⚠️ 540 Sonar way rules not included ?

[Activate More](#)

- Click en "Activate More" y rellenar el *Quality Profile* con las *Rules* necesarias para cumplir con los principios de mantenibilidad especificados en el [Apartado 4.2](#).
- SonarCloud incluye un gran número de reglas ya predefinidas. Utilizad los filtros (especialmente los tags) para encontrar las que os interesen. Si pulsamos sobre el nombre de una regla nos muestra una descripción más detallada.

Projects Quality Profiles Rules Quality Gates Members Billing & Upgrade Administration

maintainability

▼ Severity ?

- High 2
- Medium 7
- Low 35

▼ Type

- Bug 0
- Vulnerability 0
- Code Smell 44
- Security Hotspot 0

Add to selection Ctrl + click

▼ Tag

Search for tags...

- cert 53
- pitfall 45
- convention 44
- clumsy 37
- tests 32
- suspicious 30
- performance 29
- cwe 28
- confusing 24
- error-handling 21
- bad-practice 20

on the same line

Consistency

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

Java Maintainability Code Smell convention Activate

Consistency

Comments should not be located at the end of lines of code

Java Maintainability Code Smell convention Activate

Consistency

Constant names should comply with a naming convention

Java Maintainability Code Smell convention Activate

Consistency

Field names should comply with a naming convention

Java Maintainability Code Smell convention Activate

Intentionality

Fields should not be initialized to default values

Java Maintainability Code Smell convention Activate

Consistency

Files should end with a newline

Java Maintainability Code Smell convention Activate

Consistency

Interface names should comply with a naming convention

Java Maintainability Code Smell convention Activate

E. Configurar Quality Gates

- Click sobre "Quality Gates". Crear una nueva Quality Gate pulsando en el botón de "Create" con el nombre Lab3.

Projects Quality Profiles Rules Quality Gates Members

Quality Gates ?

[Create](#)

Sonar way BUILT-IN

[Copy](#)

<https://www.etsisi.upm.es/estudios/grados/61w/lq>

2. Por defecto Sonar preconfigura ciertas condiciones, así que debemos borrarlas antes de activar condiciones nuevas.

Conditions on New Code					
Conditions on New Code apply to all branches and to Pull Requests.					
Metric	Operator	Value			
Coverage	is less than	80.0%			
Duplicated Lines (%)	is greater than	3.0%			
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			

3. Añadir tantas condiciones como sean necesarias para cumplir los requisitos especificados. Para ello usar el botón de “Add Condition”. Las condiciones se pueden imponer sobre todo el código o solo sobre el código nuevo. Una vez especificadas todas las condiciones necesarias, tenemos que configurar el proyecto (o proyectos) sobre el que actuará la Quality Gate creada. **Para ello demos seleccionar nuestro proyecto en el recuadro de “Projects”** (OJO! Si tu proyecto no aparece es necesario buscarlo en la pestaña de “All”).

Projects Quality Profiles Rules **Quality Gates** Members

Quality Gates ? [Create](#)

10-maintainability-principles

alumnoetsisi

Sonar way **DEFAULT BUILT-IN**

alumnoetsisi

Conditions ?

No Conditions

[Rename](#) [Copy](#) [Set as Default](#) [Delete](#)

[Add Condition](#)

Add Condition

☒ On New Code ☐ On Overall Code

Quality Gate fails when

Search for metrics...

[Add Condition](#) [Cancel](#)

Quality Gates ? [Create](#)

Lab3

Sonar way **DEFAULT BUILT-IN**





[Rename](#) [Copy](#) [Set as Default](#) [Delete](#)

[Add Condition](#)

Conditions ?





Conditions on New Code

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

Metric	Operator	Value	Edit	Delete
Duplicated Blocks	is greater than	0		
Code Smells	is greater than	0		

Conditions on Overall Code

Conditions on Overall Code apply to long-lived branches only.

Metric	Operator	Value	Edit	Delete
Code Smells	is greater than	0		
Duplicated Blocks	is greater than	0		

Projects ?

[With](#) [Without](#) [All](#)

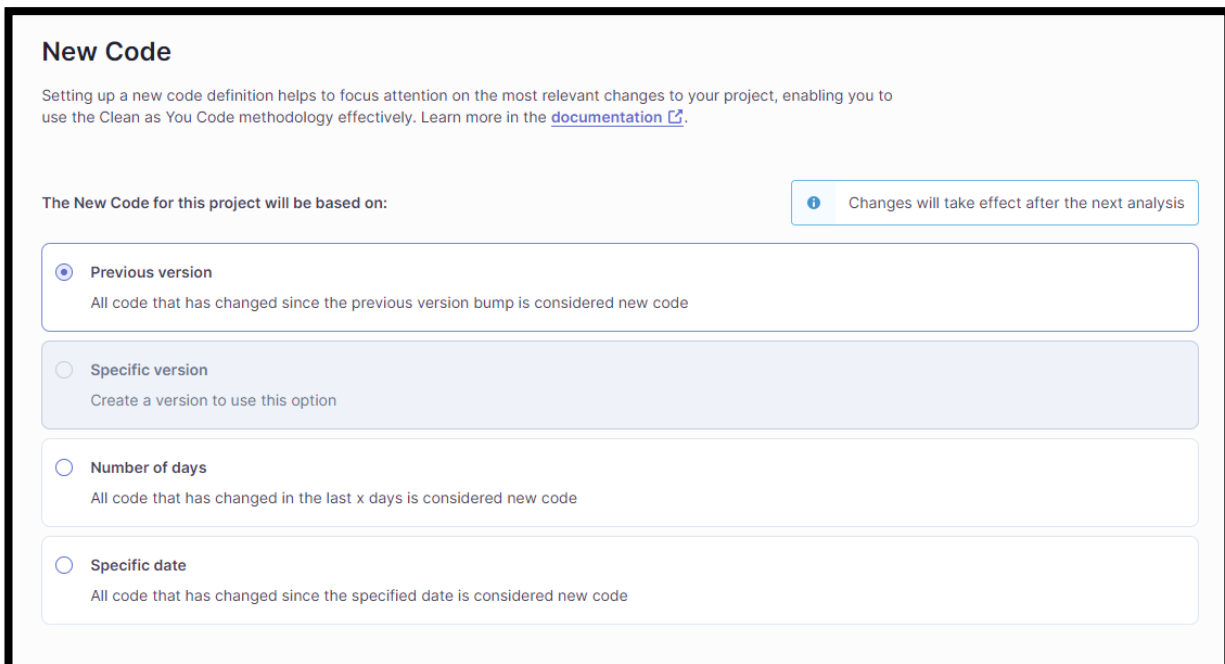
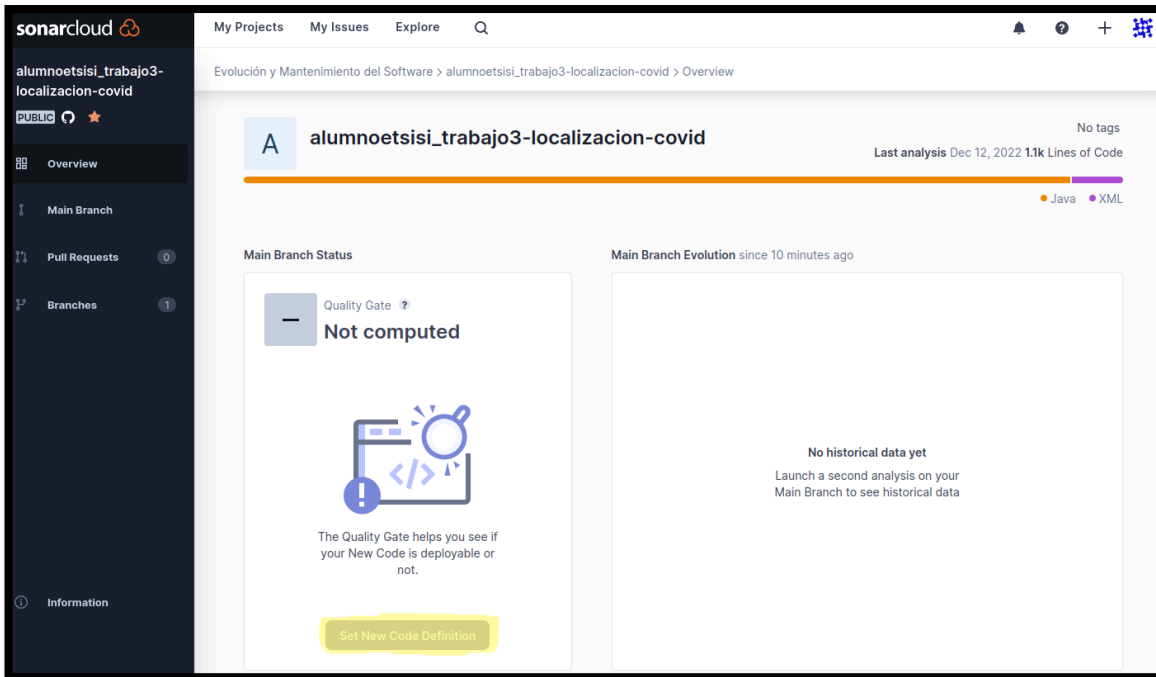
☒ trabajo3-localizacion-covid

alumnoficticio3_trabajo3-localizacion-covid

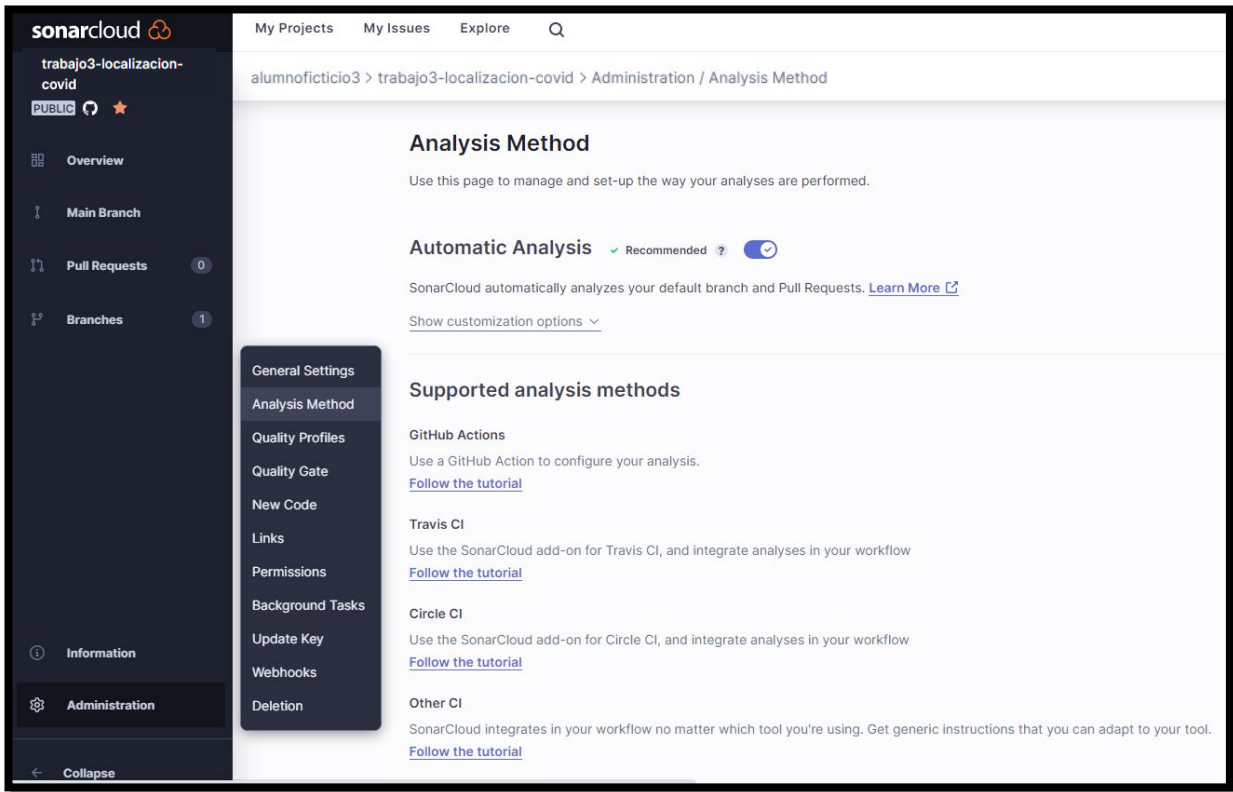
4.3.2 Configuración de un Pipeline de CI con GitHub Actions y SonarCloud

F. Activar en Sonar CI con GitHub Actions

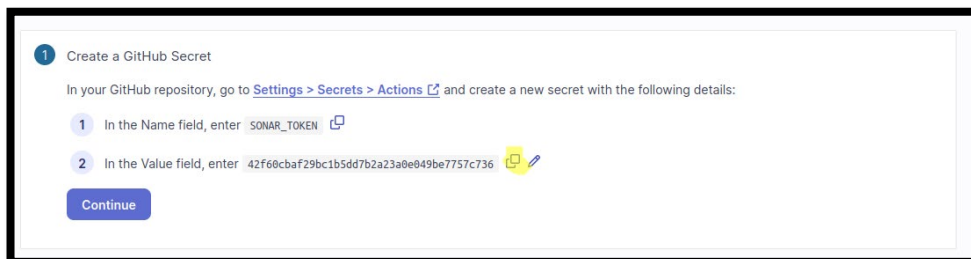
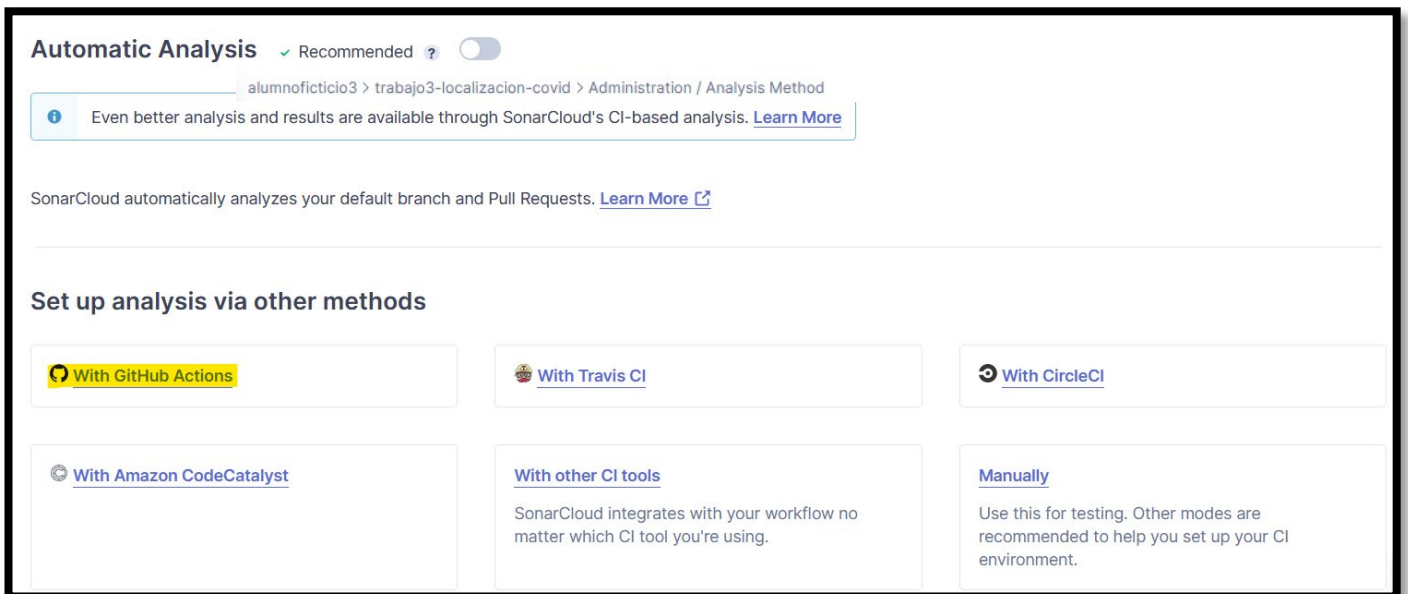
1. Pestaña “Projects” y click en nuestro proyecto. Click en el botón “Set New Code Definition” y seleccionar “Previous Version”. Si no aparece el botón es que ya se definió con anterioridad y no es necesario definirlo de nuevo.



2. Menú “Administration”, seleccionar “Analysis Method” y desmarcar el análisis automático

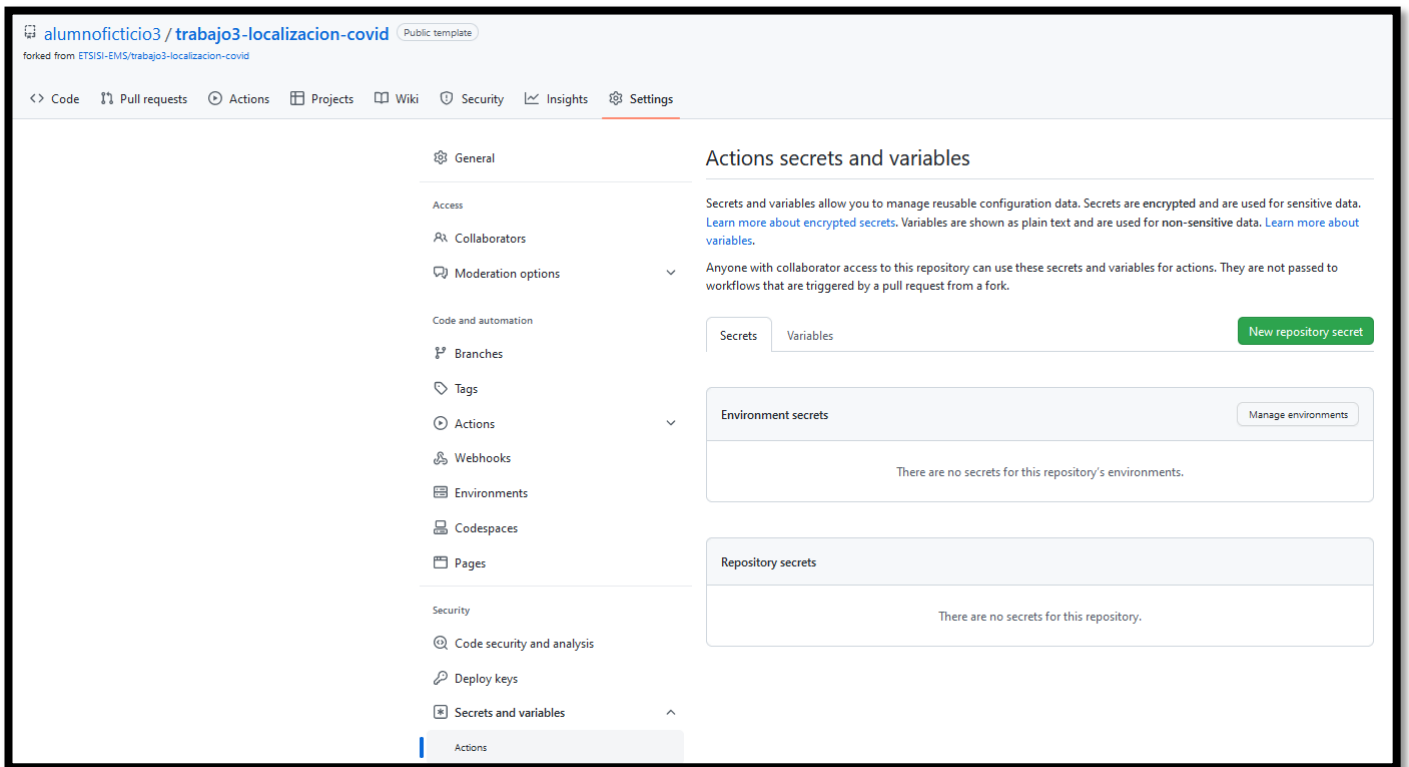


3. Click en “With Github Actions” y copiar el token de SonarCloud

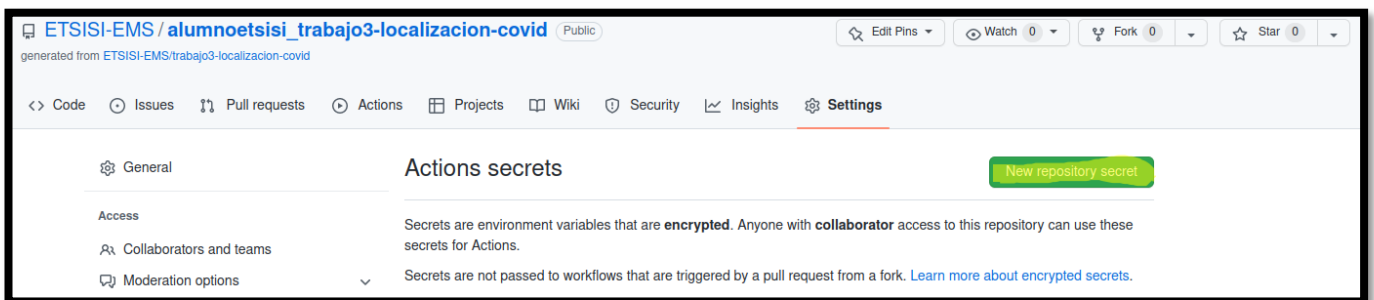


G. Configurar Secretos para el pipeline en GitHub Actions

1. Para analizar un proyecto Sonar con GitHub Actions es necesario configurar un secreto. En GitHub, en los settings del repositorio, menú Secret->Actions



2. Crear un secreto nuevo: “New repository secret” con el Name **SONAR_TOKEN** y en Secret el token copiado desde SonarCloud en el paso anterior. Click en “Add secret”.



Actions secrets / New secret

Name *

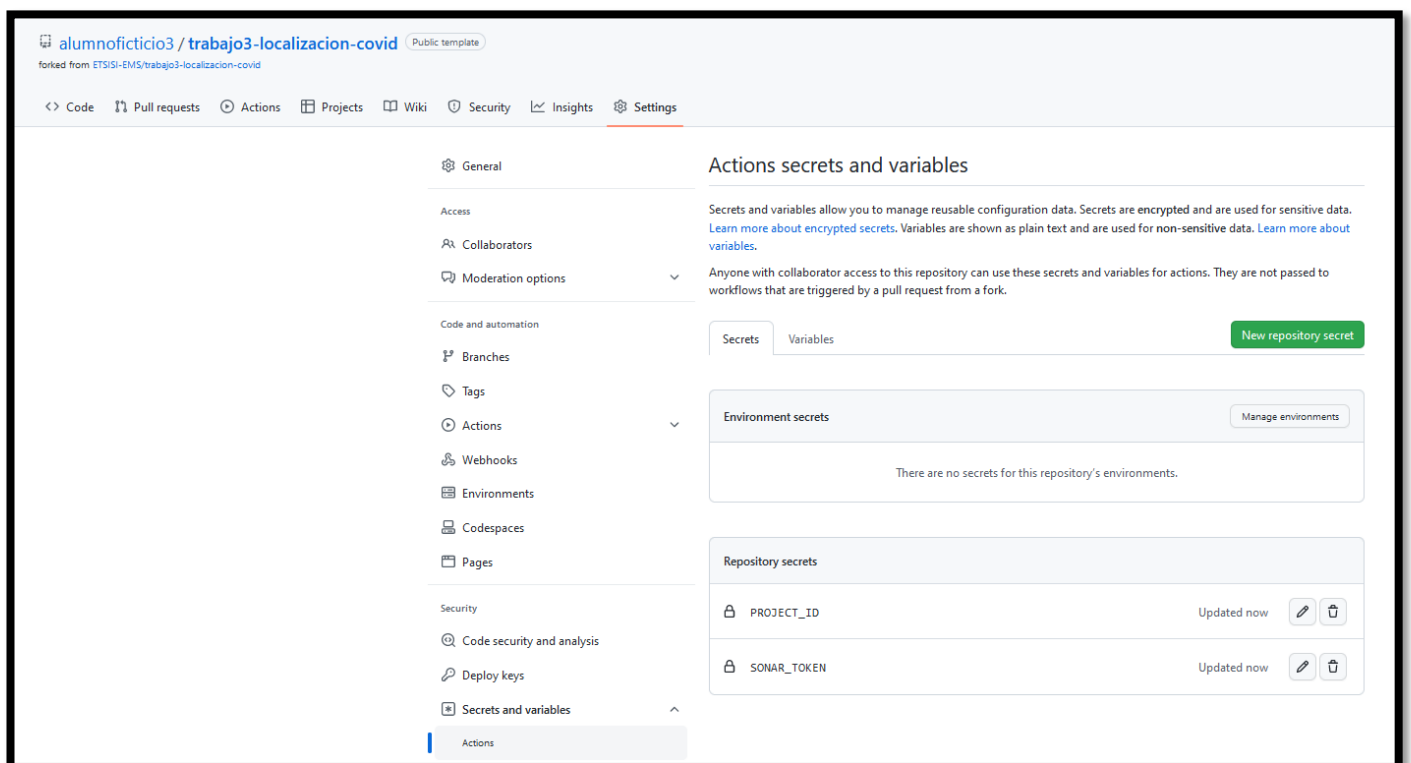
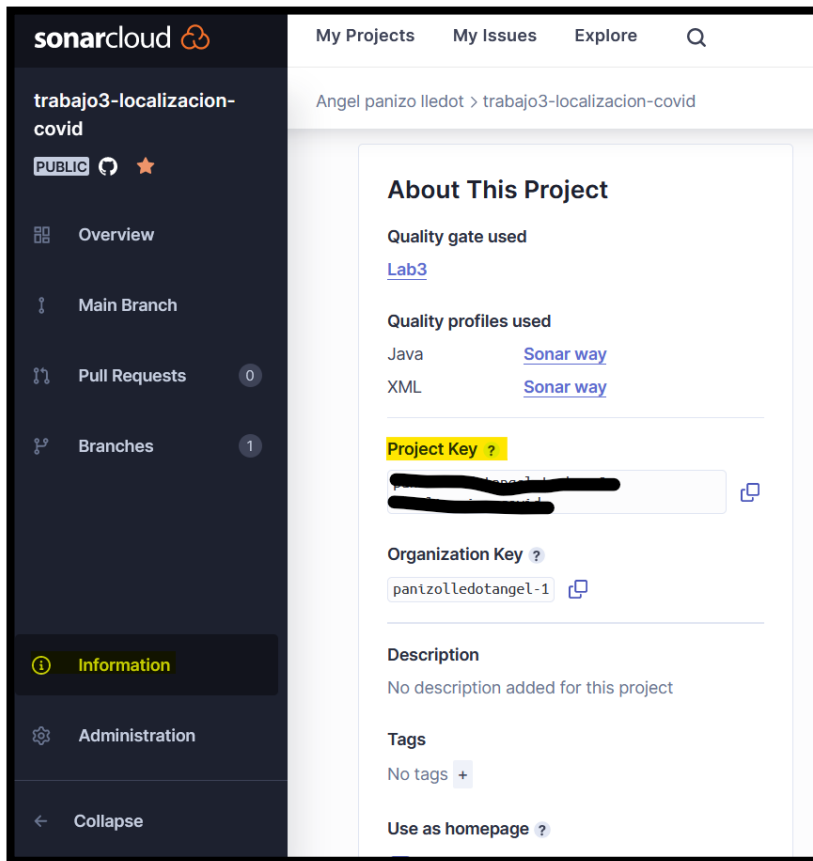
SONAR_TOKEN

Secret *

<token copiado de SonarCloud>

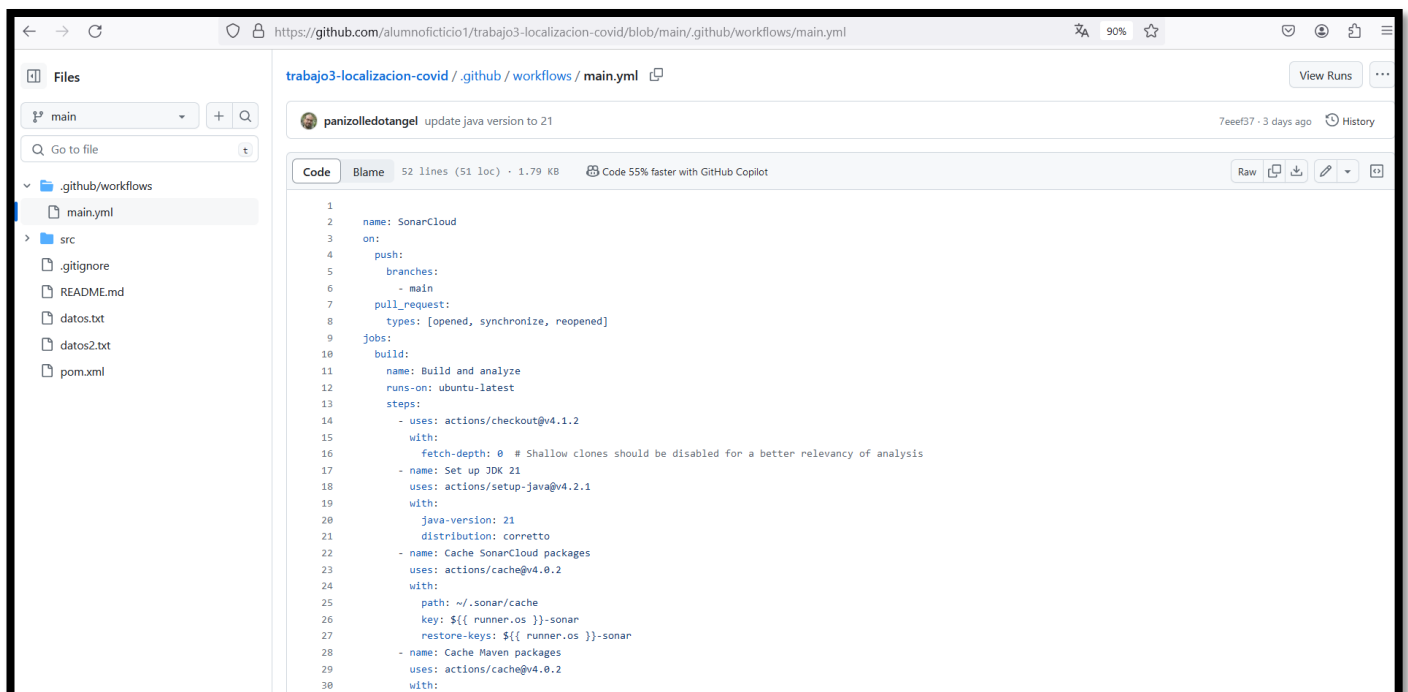
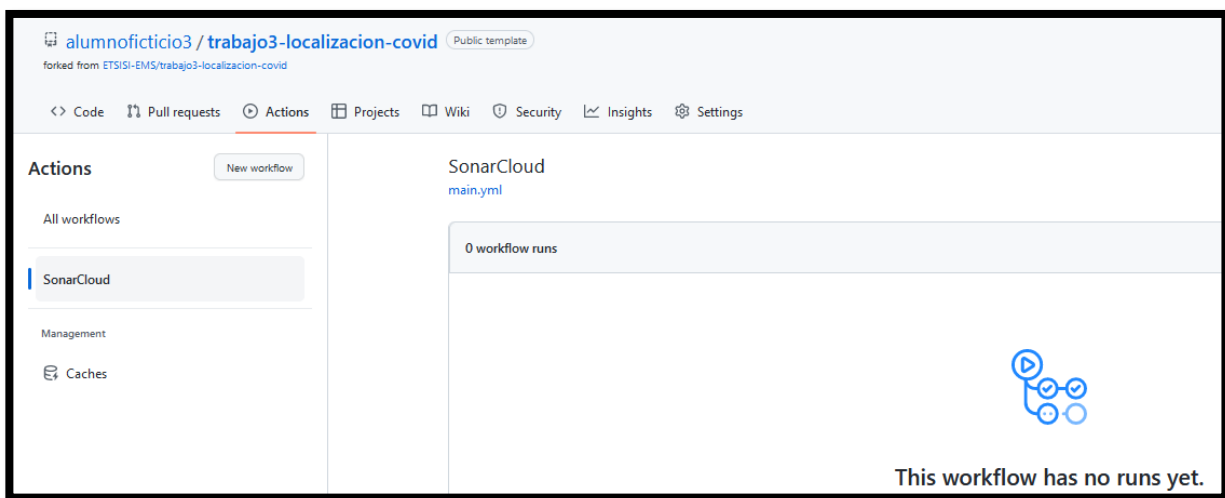
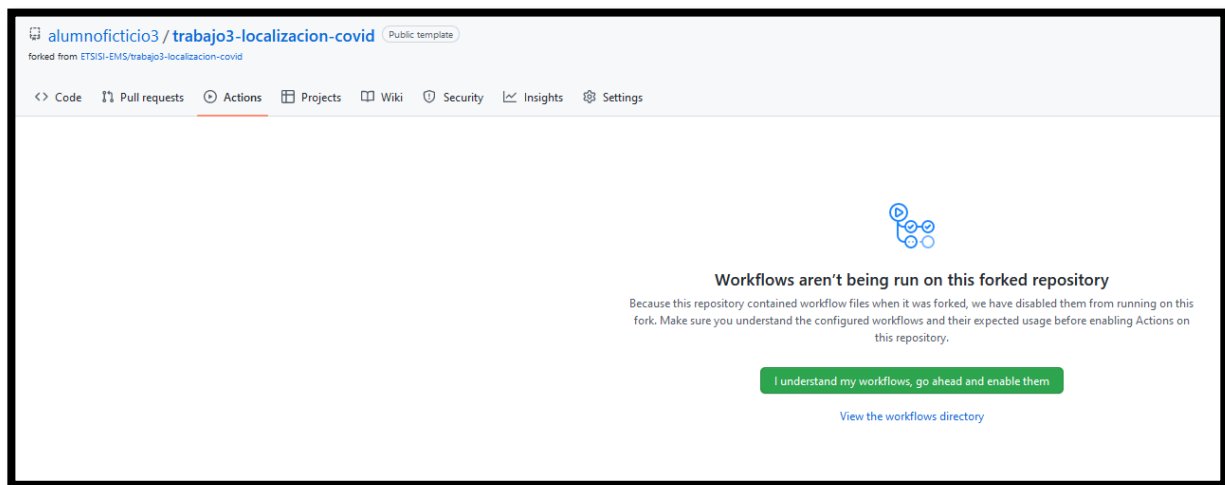
Add secret

3. Crear otro secreto con Name **PROJECT_ID** y en Secret copiar el id de nuestro proyecto en SonarCloud. Para conocer el id de nuestro proyecto, visitar el proyecto en SonarCloud, click en “Information” y copiar el valor de “Project Key”



H. Definir el pipeline en GitHub Actions

1. Pestaña de "Actions". Click en "I understand my workflows, go ahead and enable them".



2. Configurar la organización de SonarCloud en el fichero pom.xml. El id de tu organización es el campo Organization_Key de la pestaña de información del proyecto de sonar cloud.

1

alumnoficticio1 Update pom.xml with sonarcloud organization

Code

Blame

79 lines (73 loc) · 2.54 KB

Code 55% faster with GitHub Copilot

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <project xmlns="http://maven.apache.org/POM/4.0.0"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7
8     <groupId>com.practica</groupId>
9     <artifactId>trabajo3-localizacion-covid</artifactId>
10    <version>0.0.1-SNAPSHOT</version>
11
12    <name>trabajo3-localizacion-covid</name>
13    <url>http://www.example.com</url>
14
15    <properties>
16        <java.version>21</java.version>
17        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18        <maven.compiler.source>21</maven.compiler.source>
19        <maven.compiler.target>21</maven.compiler.target>
20        <sonar.organization>alumnoficticio1</sonar.organization>
21        <sonar.host.url>https://sonarcloud.io</sonar.host.url>
22    </properties>
23
```

3

Commit changes

Update pom.xml with sonarcloud organization

Add an optional extended description...

☒ Commit directly to the `main` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

Dirígete a las Actions

Summary

Jobs

Build and analyze

Run details

Usage

Workflow file

Build and analyze

failed now in 1m 2s

Search logs

Set up job

Run actions/checkout@v2

Set up JDK 11

Cache SonarCloud packages

Cache Maven packages

Build and analyze

Query status

show status

fail_if_error

Post Cache Maven packages

Post Cache SonarCloud packages

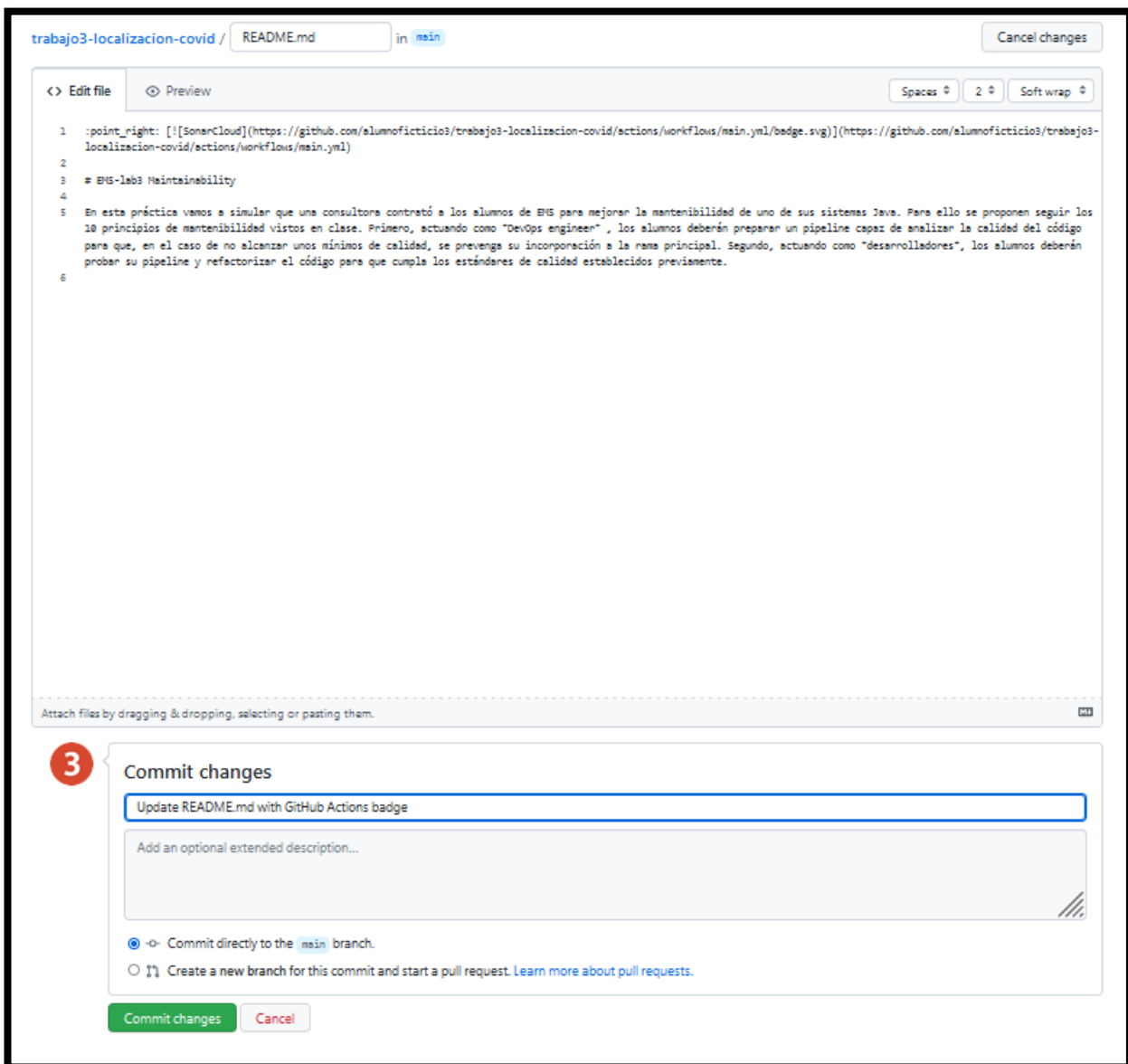
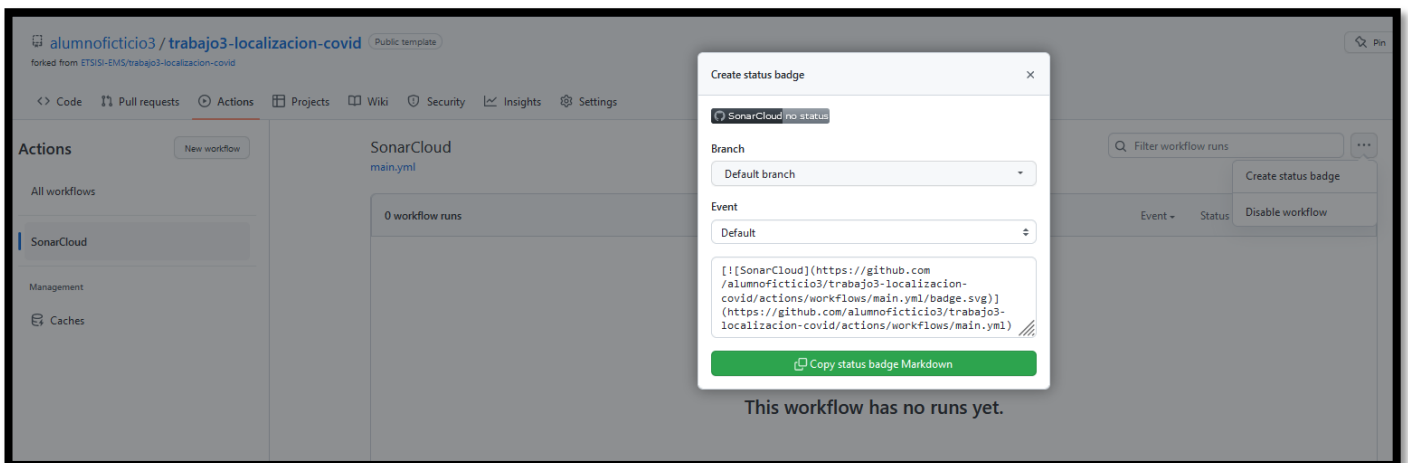
Post Set up JDK 11

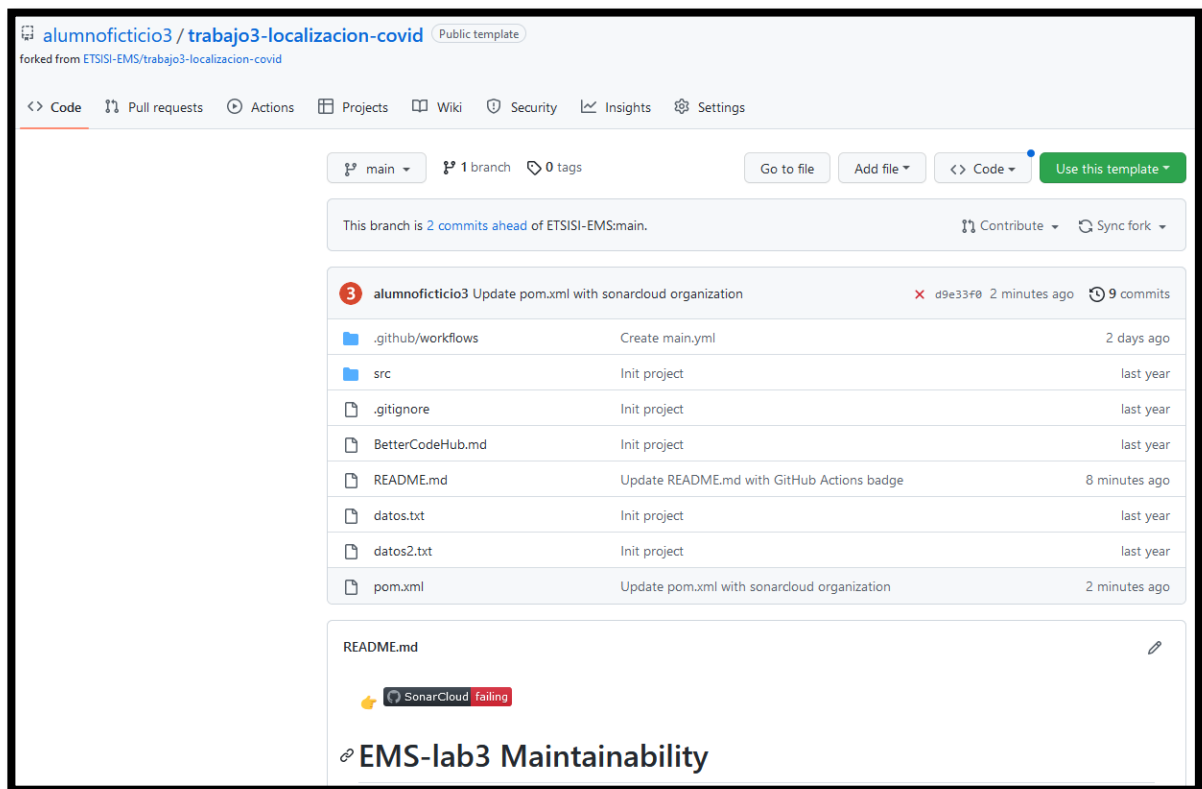
Post Run actions/checkout@v2

Complete job

¿Qué ha pasado? ¿Te esperabas este resultado?

3. El alumno podría copiar la insignia del workflow SonarCloud en el fichero README.md. Seleccionar el workflow, click en los “...” y click en “create status badge”. Editar el fichero README.md y pegar la insignia.

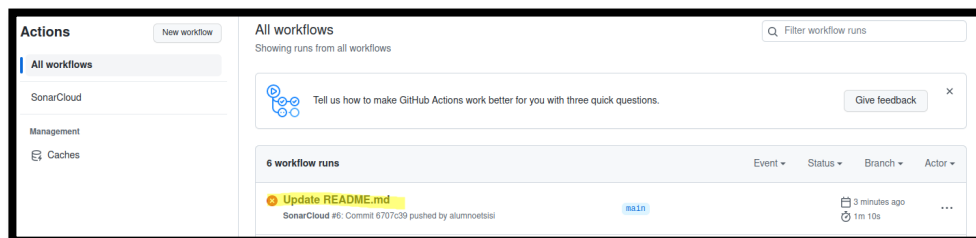




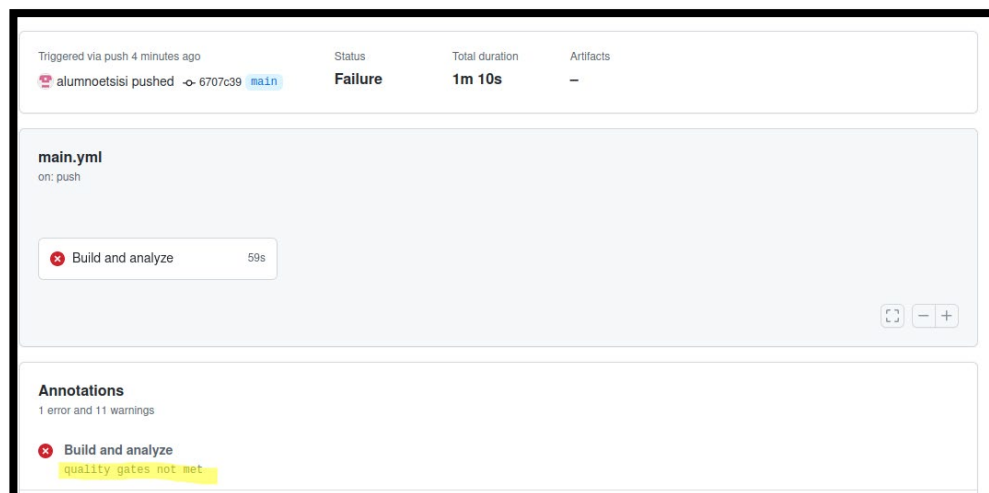
4.3.3 Ejecución del Pipeline de CI

Si se ha configurado todo correctamente, el pipeline debería fallar. Aun así, comprobar que el pipeline falla por un **incumplimiento** de las quality gates y no por otros errores.

1. Pestaña de “Actions” y click en la ejecución más reciente.



2. Debemos comprobar que la tarea “Build and analyze” contiene el mensaje “**quality gates not met**”. Si contiene otro mensaje de error deberemos arreglarlo.



Una vez que el pipeline está fallando porque no se cumplen las “quality gates”, visitar el proyecto de SonarCloud, “See Full Analysis” y comprobar que aparecen marcadas las partes del código que no cumplen con los principios de mantenibilidad (marcar Overall Code).

The screenshot shows the SonarCloud project page for 'refactor_ems'. The top navigation bar includes 'Summary', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity'. The main content area displays a 'Quality Gate' status of 'Failed' with a red 'X' icon. A red dot is visible in the top right corner of the page. The 'Overall Code' button is highlighted with an orange box. The page also shows '1.1k Lines of Code' and 'Version 0.0.1-SNAPSHOT'. The last analysis was 8 minutes ago with ID 'd9e33f09'. The metrics are as follows:

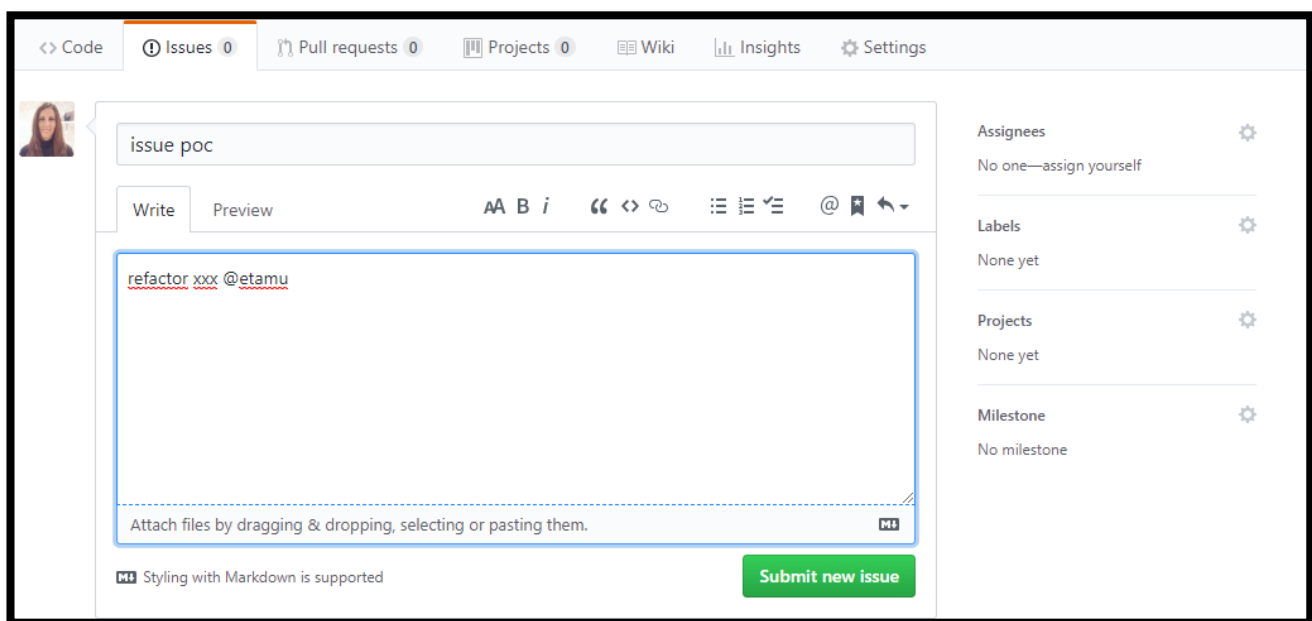
Metric	Value	Grade
Reliability	0 Bugs	A
Maintainability	0 Code Smells	A
Security	0 Vulnerabilities	A
Security Review	0 Security Hotspots	A
Coverage	0.0% Coverage	F
Duplications	0 Duplications	A

5 Parte 2: Refactorización del código

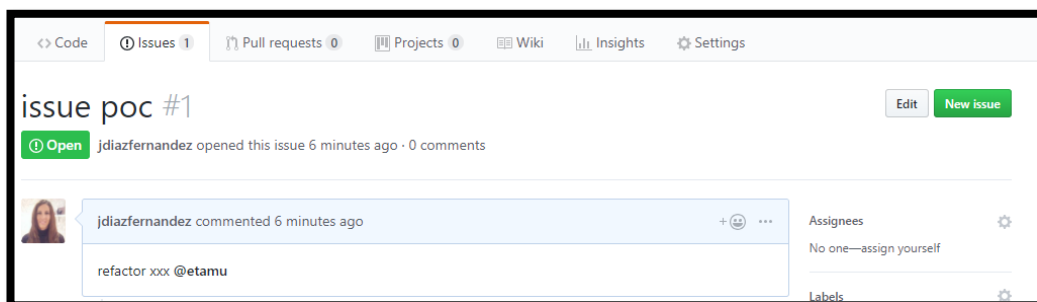
Una vez definido el pipeline que comprueba la calidad del código, toca empezar a arreglarlo para mejorar su mantenibilidad.

El código proporcionado permite gestionar la geolocalización de personas y la posibilidad de estar en contacto, funcionalidad que fue útil durante la pandemia COVID que sufrimos hace unos años (ver una descripción más extensa está disponible en el Anexo de este documento). Además, se proporciona una batería de tests en Junit5 y ficheros de texto con datos de prueba que **no se deben tocar**. Los alumnos han configurado un pipeline de CI usando GitHub Actions que se ejecuta cada vez que se hace un *push* a la rama “main” del repositorio.

Siguiendo el feedback que proporciona SonarCloud es necesario solucionar los problemas detectados. Para ello tenemos que organizar un flujo de trabajo en ramas para el equipo (podemos usar cualquiera de los vistos en el Tema 2). Además, cada refactorización debe quedar guardada en un **issue** de GitHub³. Un issue es la descripción de un cambio a realizar, por ej. en la siguiente figura se muestra que Jessica ha dado de alta un issue para refactorizar xxxx y se lo comenta al usuario @etamu (en el menú *Assignees* se puede especificar a qué miembro del equipo se le asigna ese issue).



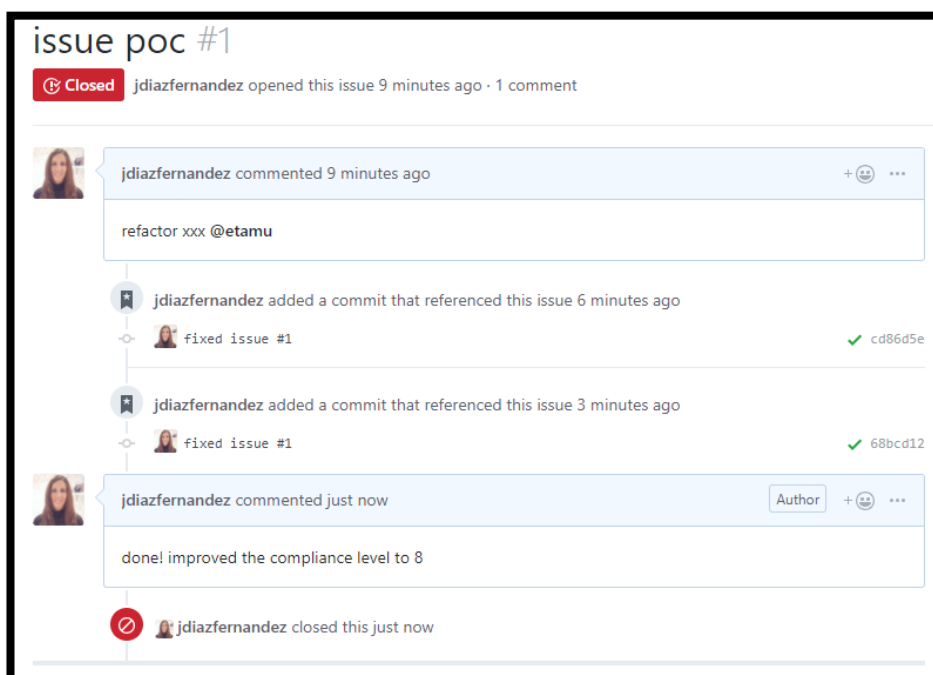
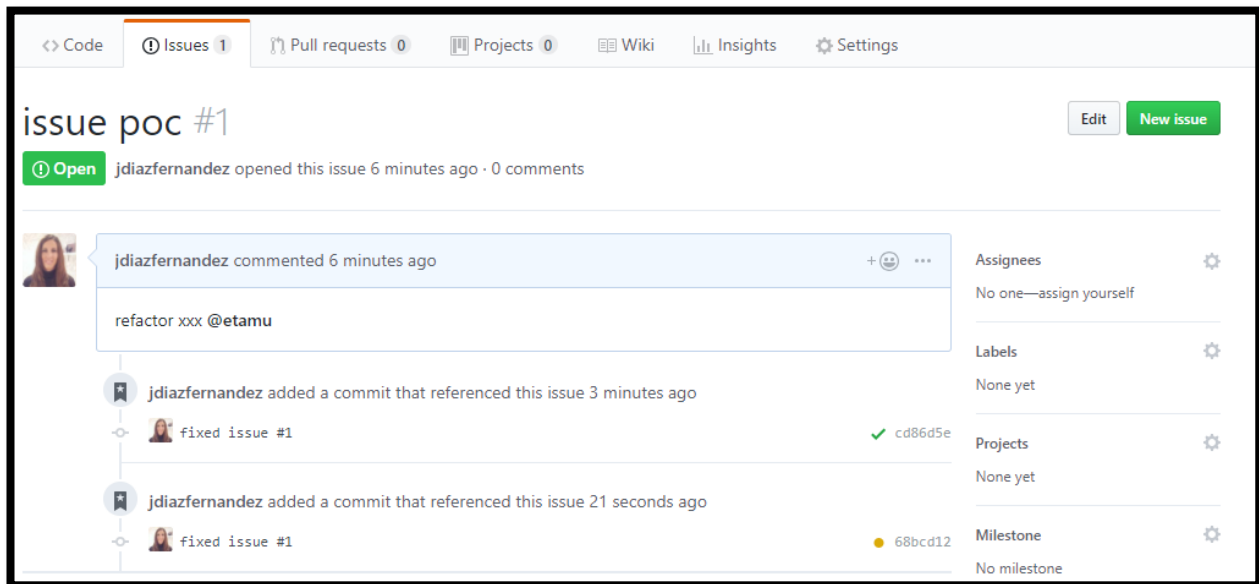
Una vez dado de alta el issue, GitHub asigna un ID (en este caso #1) como se ve en la figura:



Es **obligatorio** el uso de control de versiones a la hora de refactorizar. Cada refactorización/issue tendrá asociado un commit en el que se haya implementado el cambio. Es **obligatorio** relacionar los commits con los issue de la siguiente forma: en el mensaje del commit añadir a la descripción el #ID del issue. Una vez resuelto el issue es necesario hacer un merge con la rama main, hacer un push al repositorio remoto y cerrar el issue.

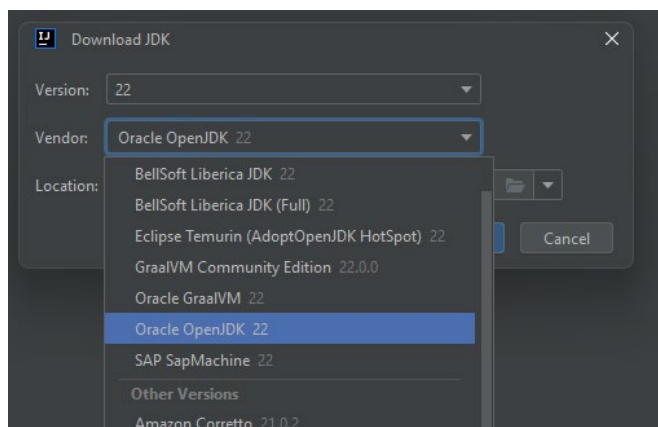
Desde la interfaz de GitHub se puede hacer el seguimiento de issues y sus commits.

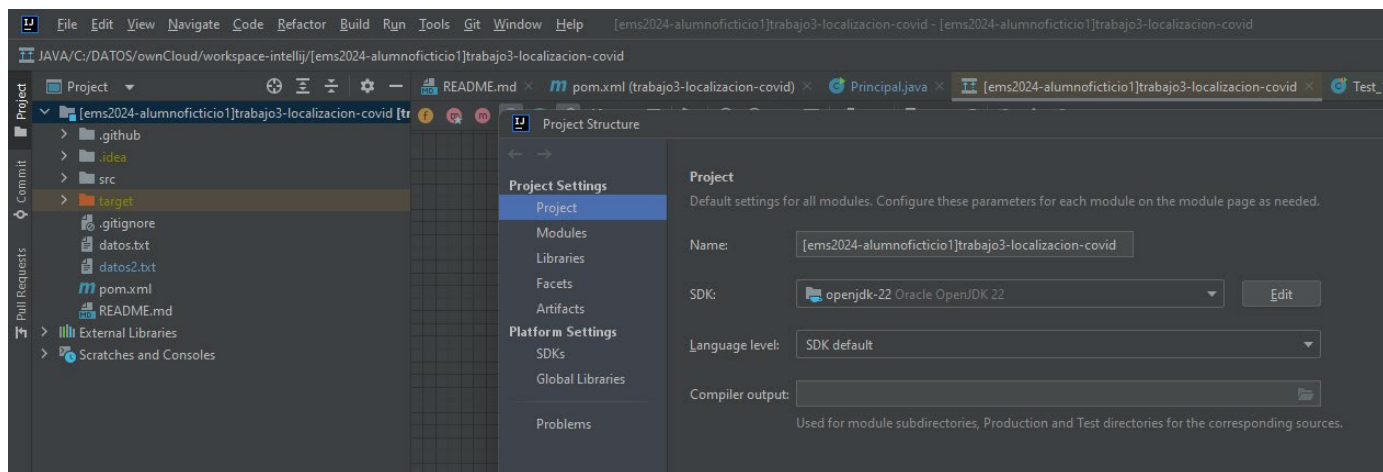
³ Si el repositorio no muestra la pestaña de Issues, comprobar en los settings → General → Features, que los issues estén activados.



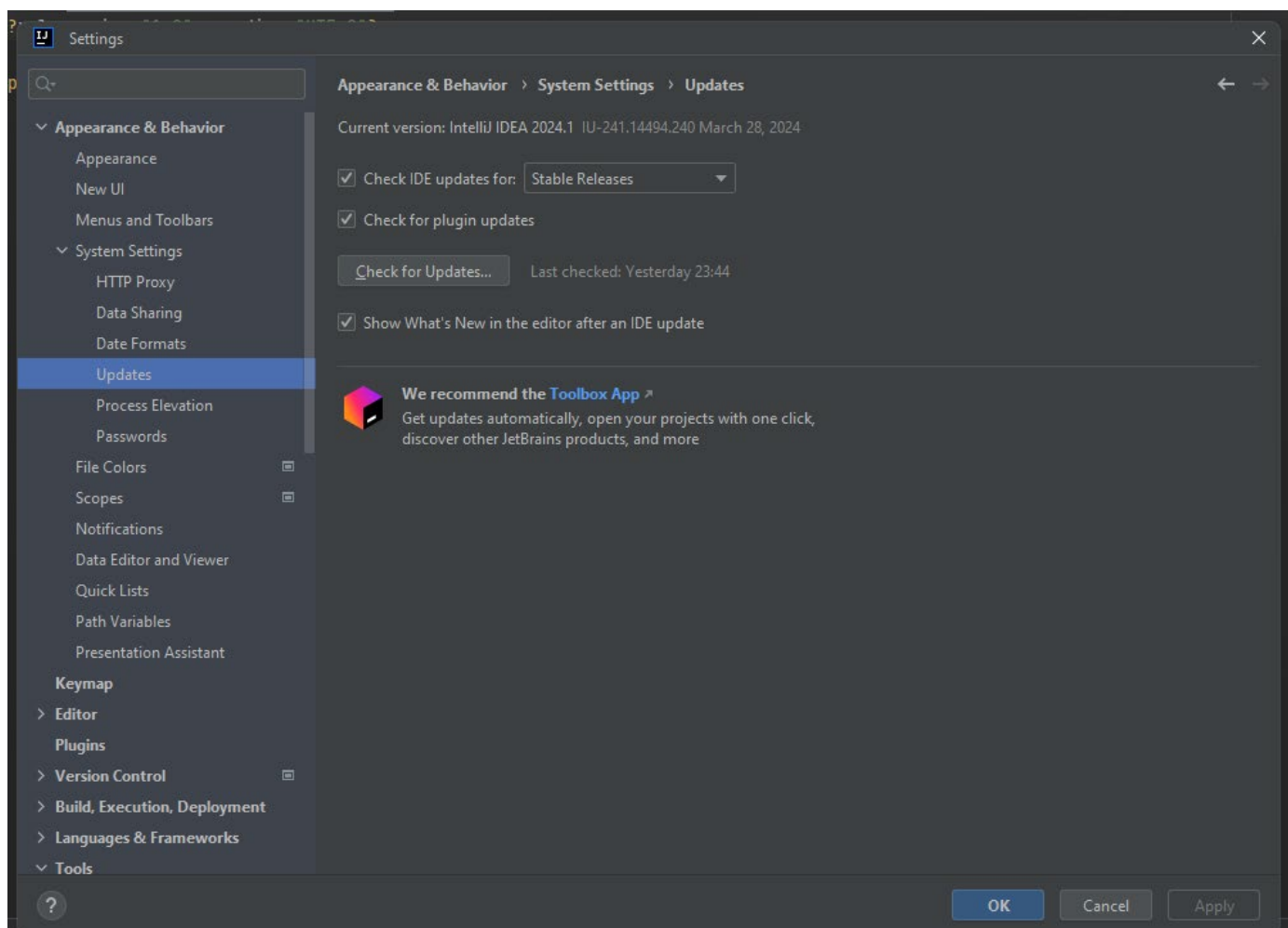
El proceso habrá terminado cuando el pipeline que configuramos se ejecute con éxito.

Para trabajar en la refactorización se recomienda el uso de IntelliJ. Si es necesario, configurar el proyecto en File/Project structure (de acuerdo al pom estamos trabajando con java.version 21). Podemos trabajar con el openjdk-22 disponible desde IntelliJ si clicáis en el menú desplegable del SDK para instalar el SDK que necesitéis en cada caso.

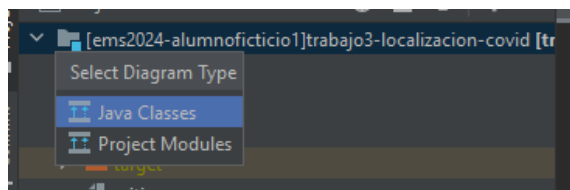




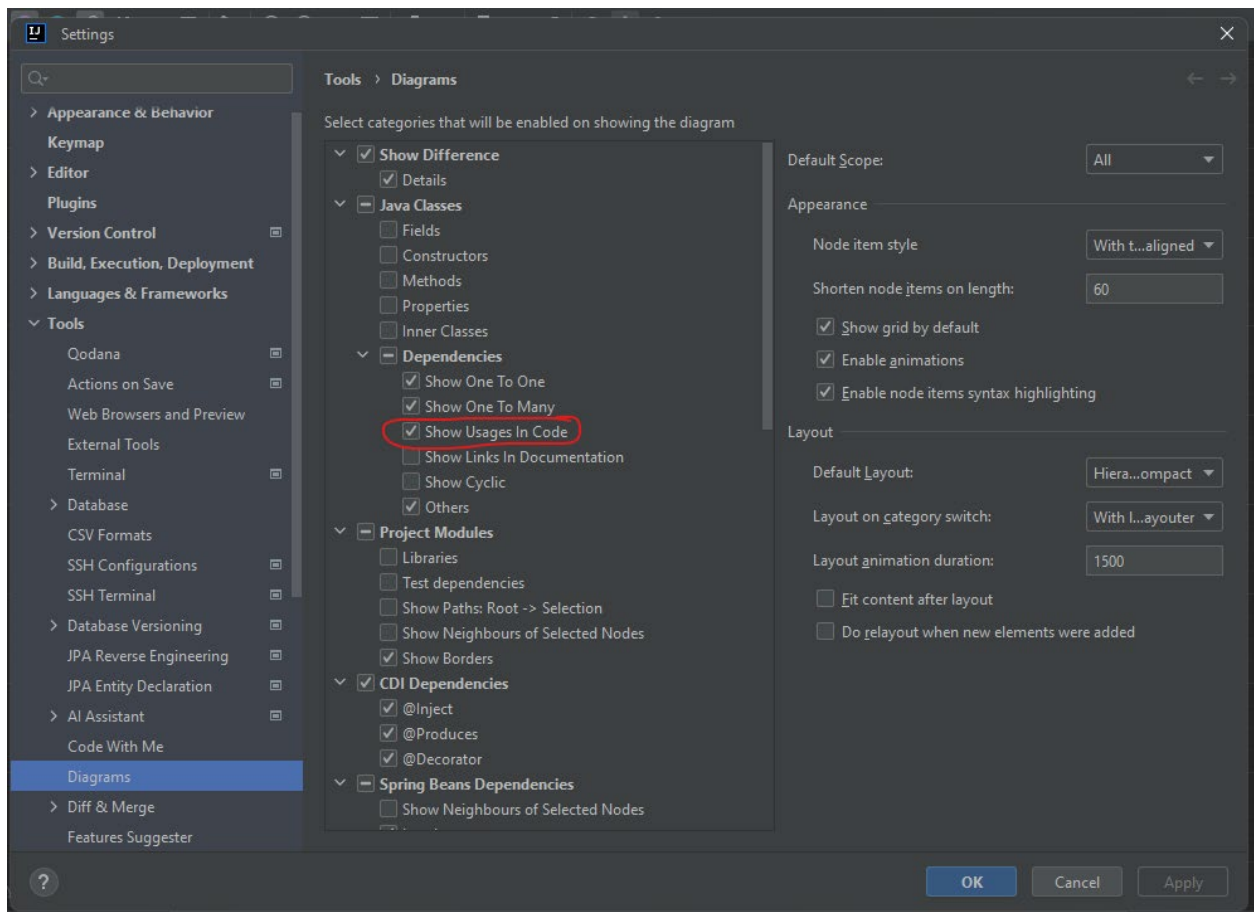
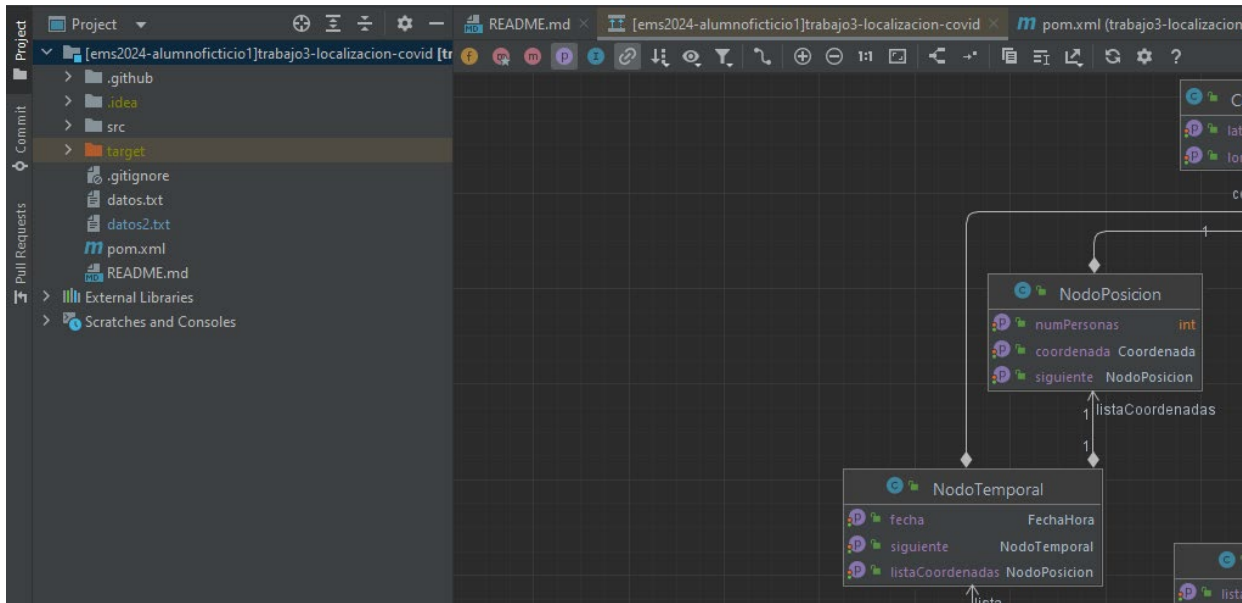
Si tenéis problemas porque no tengáis acceso a las últimas versiones de SDKs, aseguraros de tener el IDE actualizado:



Será útil en este ejercicio la versión Ultimate para obtener el diagrama de clases. Click botón derecho sobre el proyecto, Diagram/Show diagrams... Seleccionar Java Classes



En los menús del diagrama el estudiante puede seleccionar si hacer visibles los atributos (propiedades), métodos, y las dependencias (muestra dependencias de composición, herencia, etc.). Si queremos mostrar dependencias de uso, hay que configurarla en el botón de settings:



6 Parte 3: PMD

[Design | PMD Source Code Analyzer](#)

[La Ley de Demeter, la que casi nadie aplica ¿Es bueno hacer muchas llamadas encadenadas a métodos? - Javier Garzas](#)

1. Introducción: Geolocalización y población.

Dada la situación del Covid se ha hecho necesario el poder tener localizada a la población que así lo desee y saber qué personas han estado en contacto entre ellos. Para tal propósito se han creado una serie de clases con las que intentamos modelar este problema. Estas clases son las siguientes:

Persona → Contiene los datos de una persona

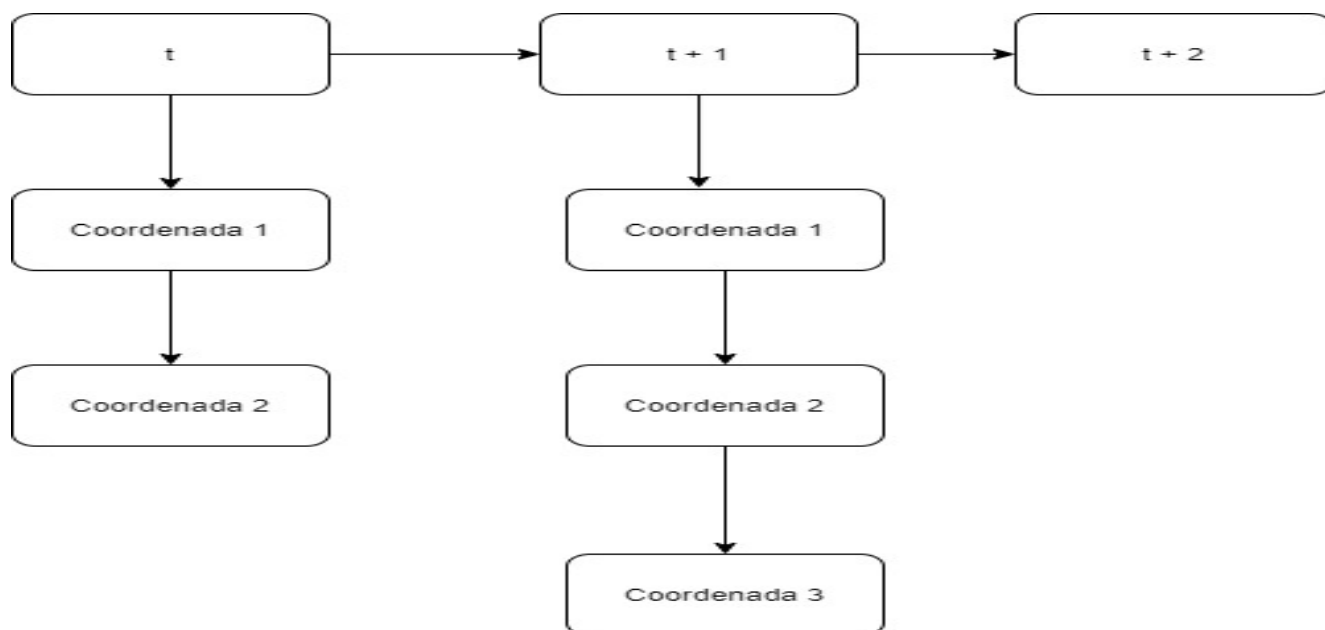
Coordenada → Representa una posición

FechaHora → Clase que representa la fecha y la hora

PosicionPersona → Clase que relaciona una persona mediante su documento en una coordenada

Además, hemos creado Localización y Población. La clase Población es una lista enlazada de personas y Localización es también una lista enlazada, pero de personas en distintas coordenadas.

Con todo esto generamos una lista de contactos. Esta lista es una secuencia de nodos temporales, donde iremos guardando instantes (FechaHora). Cada uno de esos instantes guardan una lista de coordenadas en el que se guardan las personas (sólo número) que han estado en esa coordenada. Esta lista de contactos tendrá esta apariencia:



Por último, gestionaremos los contactos mediante la clase ContactosCovid, que aglutina los datos de las personas, las localizaciones y la lista de contactos. Esta clase lo que se encarga, hasta que se lleve a cabo una interfaz gráfica, es la de cargar los datos que nos proporcionan, procesarlos y meterlos en las clases correspondientes.

Los datos se proporcionan en formato texto, separados por punto y coma. Pueden ser de dos tipo, persona y localización. El primero lleva los datos correspondientes a una persona, el segundo lleva los datos de una persona que ha estado en una fecha y hora en una localización (longitud y latitud).

Vuestra labor será la de refactorizar la aplicación para que siga haciendo lo que hace, pero cumpliendo los estándares vistos en clase.

La aplicación deberá seguir ejecutando los test tal y como hacía antes, de no ser así, la práctica se dará por inválida.