



## Trabajo 3. Mantenibilidad y refactorización

# EVOLUCIÓN Y MANTENIMIENTO DEL SOFTWARE ©2022

*Jessica Díaz Fernández & Juan Manuel Garitagoitia & Ángel Panizo*

## 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 aprendidos durante la asignatura.

## 2 Condiciones para la realización del trabajo

El trabajo se realiza en equipo. Los equipos trabajarán refactorizando el código contenido en el repositorio GitHub que se crea con la aceptación de las siguientes tareas de GitHub Classroom, según el grupo de clase:


- GIWM31: <https://classroom.github.com/a/5ufKRxNn>
- GIWM32: <https://classroom.github.com/a/sbJ3dAFJ>
- GIWT31: <https://classroom.github.com/a/GWaOqKCs>
- GIWT32: <https://classroom.github.com/a/dlDiCOGs>
- Grupo EF: <https://classroom.github.com/a/H6ZFfvPH>

El alumno debe entregar, además del repositorio de código (el alumno NO debe realizar ninguna acción especial ya que los profesores tienen acceso a todos los repositorios del curso ETSISI-EMS) un documento **PDF** con la descripción de cada uno de los apartados descritos en el enunciado.

El alumno utilizará la herramienta **BetterCodeHub (BCH)** para el análisis de la calidad y mantenibilidad del código de acuerdo a los principios estudiados en el Tema 4. La información básica de la herramienta está disponible en <https://bettercodehub.com/> <https://bettercodehub.com/docs/configuration-manual>. BCH dispone de un plan gratuito para el análisis de repositorios **públicos**. La asignatura de EMS tiene una licencia para poder usar esta herramienta con repositorios **privados** alojados en la organización <https://github.com/ETSISI-EMS>.

BCH debe complementarse con el análisis proporcionado por el plugin de **Eclipse PMD** o herramientas similares de otros IDEs. SonarQube sería equivalente a BCH aunque en esta ocasión no se dispone de este servidor para utilizar esta herramienta.

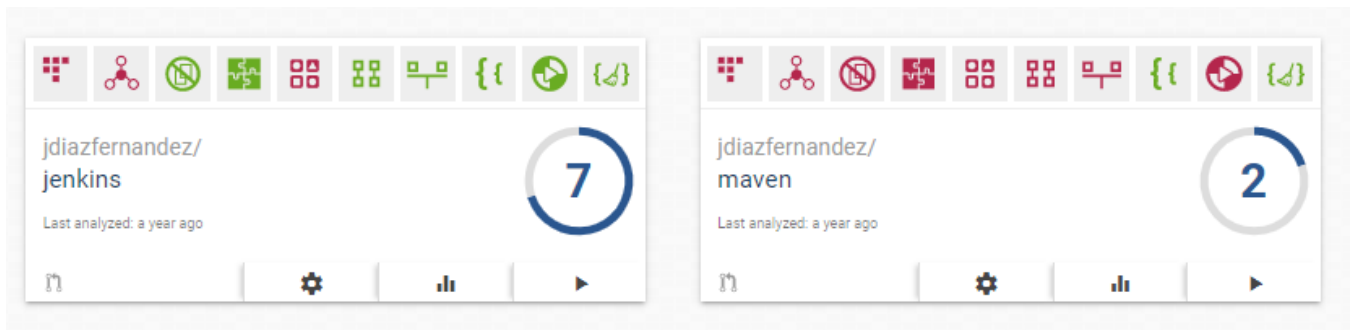
El repositorio de código proporcionado contiene un fichero **Readme.md** y **BetterCodeHub.md** que el alumno debe leer ya que proporcionan indicaciones básicas para la realización del trabajo. En particular son importantes los pasos aquí destacados (ver BetterCodeHub.md):

 **How to get started.**

---

1. Make sure that you are a member of [ETSISI-EMS](#). Membership of the ETSISI-EMS organization allows you to store and analyse private repo's on GitHub.
2. Visit [bettercodehub.com](https://bettercodehub.com), login with your GitHub handle and accept the 3 scopes, please!
3. Press Play button to get the first feedback on the 10 guidelines, browse the report, pick refactor candidates and turn them into GitHub Issues.
4. Enable Push & Pull Request support by clicking on the PR icon on the card, this will make BetterCodeHub analyse your commits and pull requests.
5. Send a PR to add your badge code to be added to the scoreboard.md

Por tanto, el primer paso es acceder a <https://bettercodehub.com/> y logarse con la cuenta de GitHub. Una vez se hayan aceptado las condiciones de BCH se podrán analizar los repositorios privados de la organización ETSISI-EMS. Podéis analizar únicamente repositorios públicos de vuestra cuenta personal (ej. código de terceros, proyectos open-source como los de la siguiente figura), así como públicos y privados dentro la organización.



Observad en la opción “**Your repositories**” que esté seleccionado el icono de la organización. El repositorio creado para esta tarea a través de GitHub Classroom es un repositorio privado. La figura os muestra el análisis realizado sobre un repositorio de ejemplo.

Iconos que representan cada uno de los 10 principios de mantenibilidad

Resultado del análisis (sobre 10)

Arrancar un análisis manual

Ver los resultados del último análisis

Activado, lanzará el análisis cada vez que haya un cambio en el

Configuraciones. Lugar además donde recuperar el “compliance badge”

**Settings**

**Compliance badge** Better Code 2 / 10

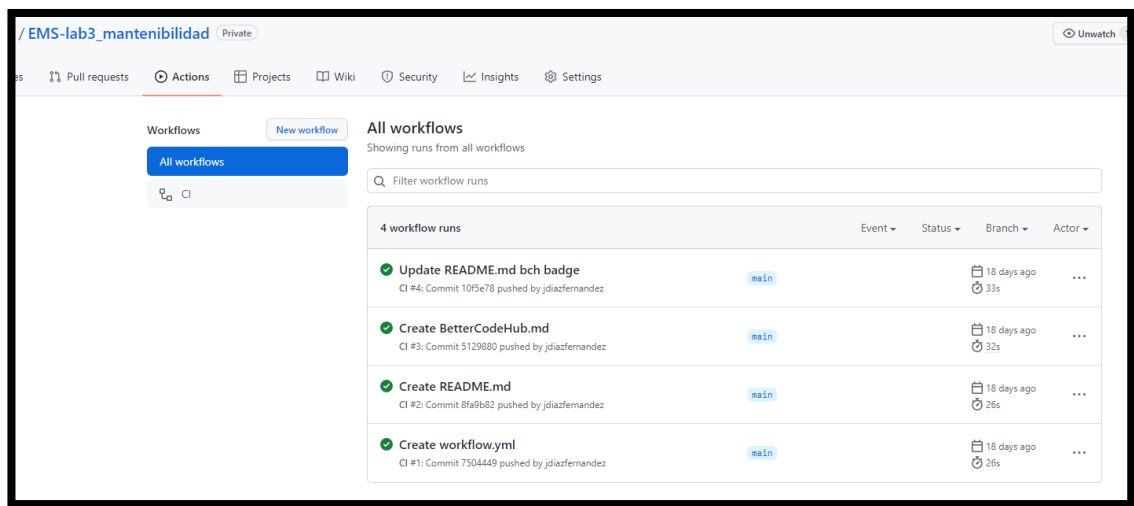
Grab your badge for Markdown:

```
[![BCH compliance](https://bettercodehub.com/edge/badge/ETSISI-EMS/trabajo3-localizacion-covid?branch=main&token=ef1250f5ea10e04807d24452f4809cc89bfcc5df)](https://bettercodehub.com/)]
```

La interfaz y funcionalidad es muy sencilla. Lo primero que veis es el resultado de análisis según los principios de mantenibilidad que contempla esta herramienta (que son los estudiados en clase), en este caso un nivel de conformidad de 2 sobre 10. Los iconos representan cada uno de los principios de mantenibilidad (en rojo los que no se cumplen y en verde los que se cumplen).

El primer paso sobre vuestro repositorio será realizar un análisis manual. Es muy recomendable que activéis que se lance un análisis cada vez que realicéis un cambio en el repositorio (remoto). Además, los profesores han configurado un pipeline de CI usando GitHub Actions que se ejecuta cada vez que se hace un push al repositorio. El pipeline ejecuta un comando mvn (mvn package) estudiado en los laboratorios del Trabajo 1, que, entre otras tareas, compila y ejecuta los test que los profesores han programado. **Investigad** el menú GitHub Actions, el

fichero yml que especifica el pipeline de CI, y comprobad que este pipeline se ejecuta con cada cambio en el repositorio remoto.



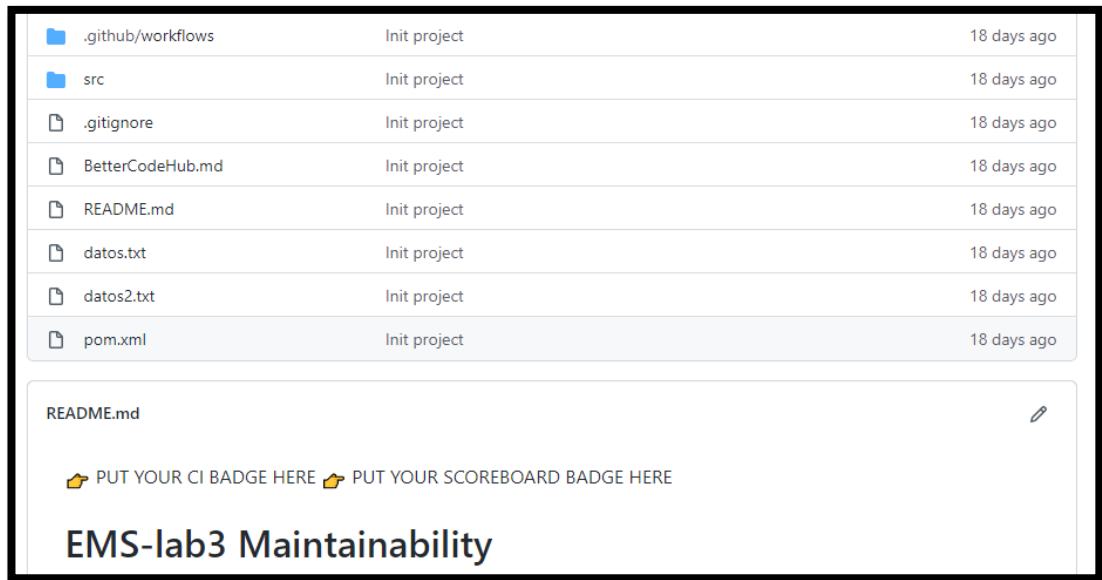
```
name: CI

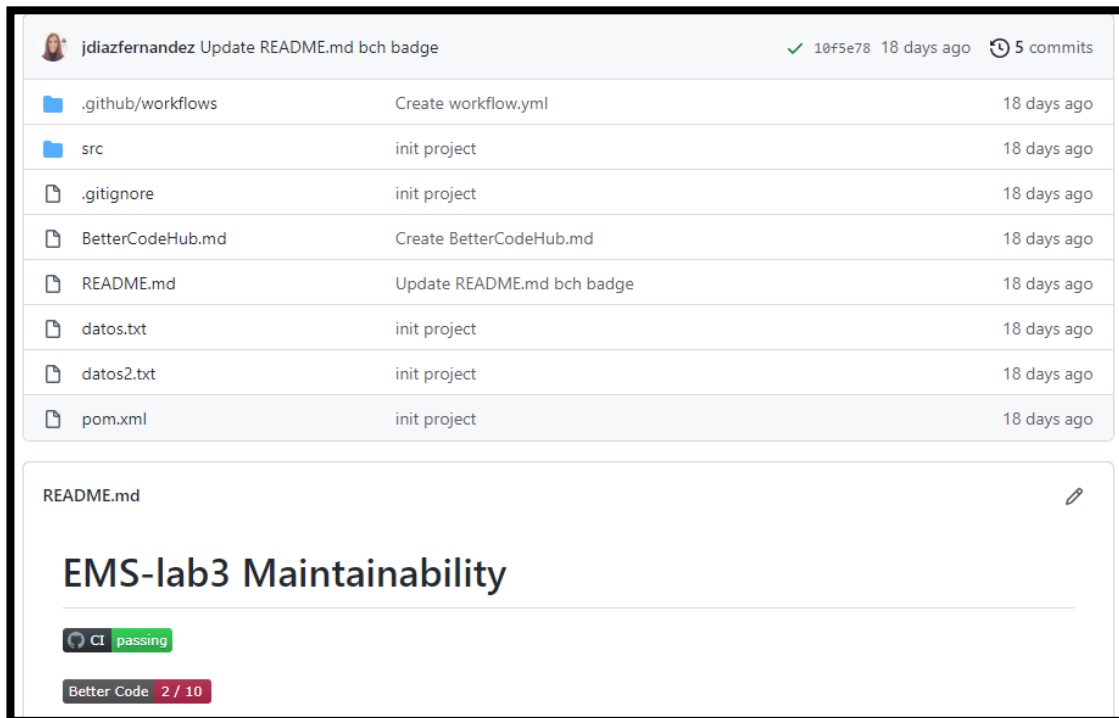
on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

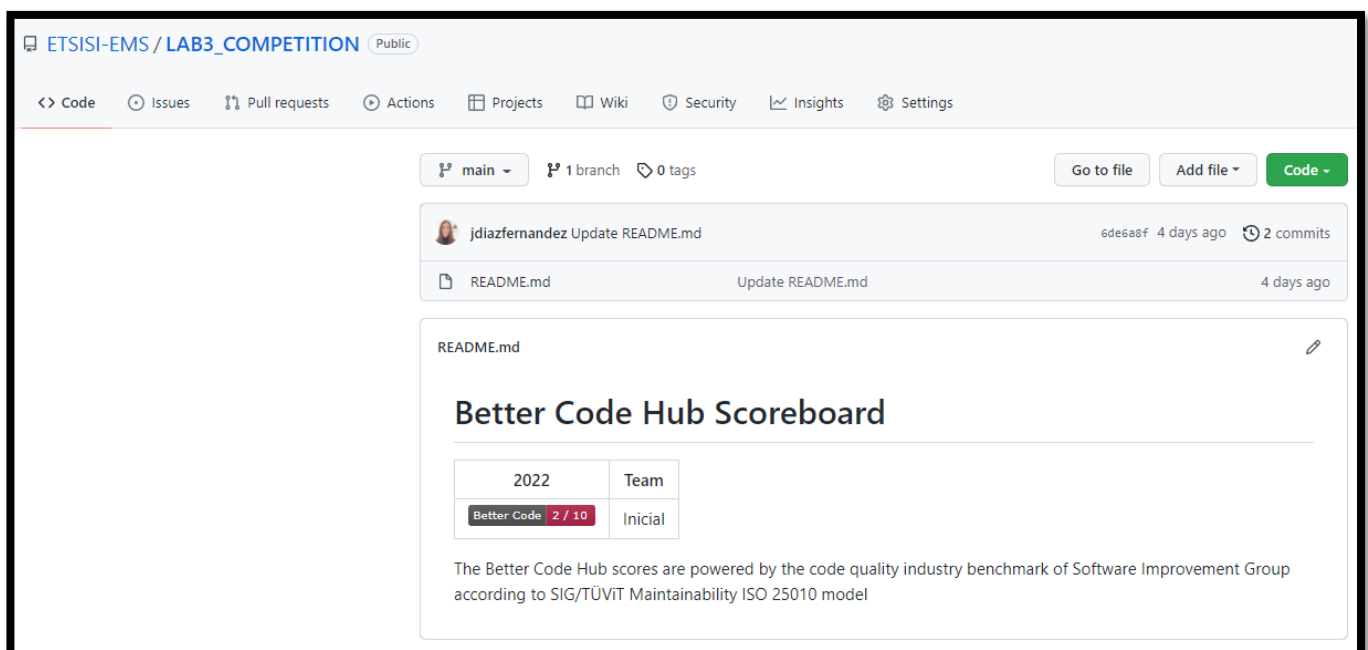
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 1.8
        uses: actions/setup-java@v1
        with:
          java-version: 1.8
      - name: Build with Maven
        run: mvn -B package --file pom.xml
```

Por último, debéis realizar (antes de empezar a refactorizar) algunos pasos para actualizar los *badges* de vuestro proyecto, en particular, el badge de CI y el “compliance badge” de BCH. Las siguientes capturas muestran el estado inicial y el estado esperado (investigad dónde se obtienen los badges):

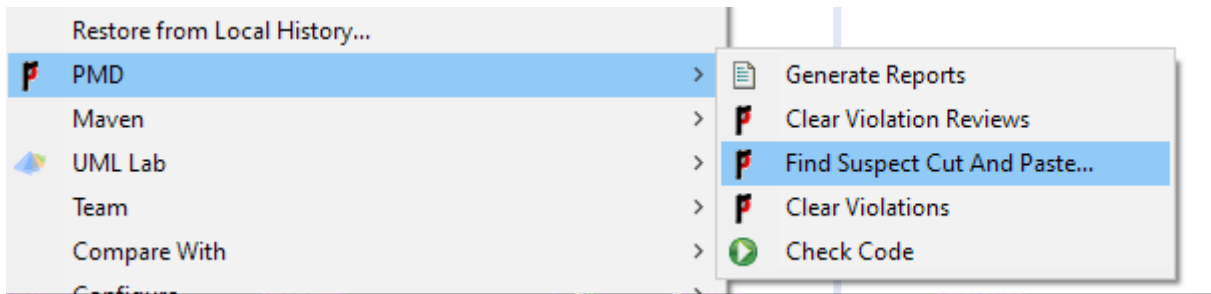




Finalmente, realizad una Pull Request con el “compliance badge” al repositorio [https://github.com/ETSISI-EMS/LAB3\\_COMPETITION](https://github.com/ETSISI-EMS/LAB3_COMPETITION), en particular sobre el fichero README.md (en notación *markdown* que debisteis utilizar en el lab 0 del Trabajo 1). Cada equipo debe proponer un cambio en este fichero que consiste en añadir una fila con el *badge* resultado del análisis realizado por BCH sobre su repositorio y el ID de su equipo. Así se obtendrá un listado por equipos, y el nivel de conformidad alcanzado por cada uno de ellos en la realización del trabajo, como muestra la siguiente figura:



En cuanto al plugin PMD de Eclipse, se disponen de opciones para detectar código duplicado o sospechoso de copy/paste (es posible configurar además el número mínimo de líneas para realizar este análisis) y la opción de “Check Code” que analiza violaciones de código categorizadas desde “blocker violations” en color rojo hasta “warning violations” en color azul fuerte.



Violations Overview					
Violations Outline Metrics View JUnit Console Git Repositories Type Hierarchy Call Hierarchy					
Element	# Violations	# Violations/KLOC	# Violations/Method	Pr ^	
> etsisi.ems2020.trabajo3.lineadehorizonte	98	1441.2	16.33	lineade	
▼ etsisi.ems2020.trabajo3.lineadehorizonte	280	1044.8	7.78	lineade	
▼ Ciudad.java	146	1050.4	14.60	lineade	
LooseCoupling	1	7.2	0.10	lineade	
LocalVariableNamingConventions	1	7.2	0.10	lineade	
CyclomaticComplexity	1	7.2	0.10	lineade	

### 3 Enunciado

Este trabajo consiste en mejorar fundamentalmente la mantenibilidad y calidad del código Java de una aplicación, identificando code smells y refactorizando el código fuente aplicando principios de mantenibilidad.

Las clases proporcionadas permiten gestionar una geolocalización de personas y la posibilidad de estar en contacto, todo ello por la nueva pandemia COVID que sufrimos desde hace dos años. Además, como ya se ha mencionado, se proporciona una batería de tests en Junit5 y ficheros de texto con datos de prueba que **no se deben tocar**. Los profesores han configurado un pipeline de CI usando GitHub Actions que se ejecuta cada vez que se hace un push al repositorio.

#### Tareas. Documentar cada tarea en el documento PDF a entregar

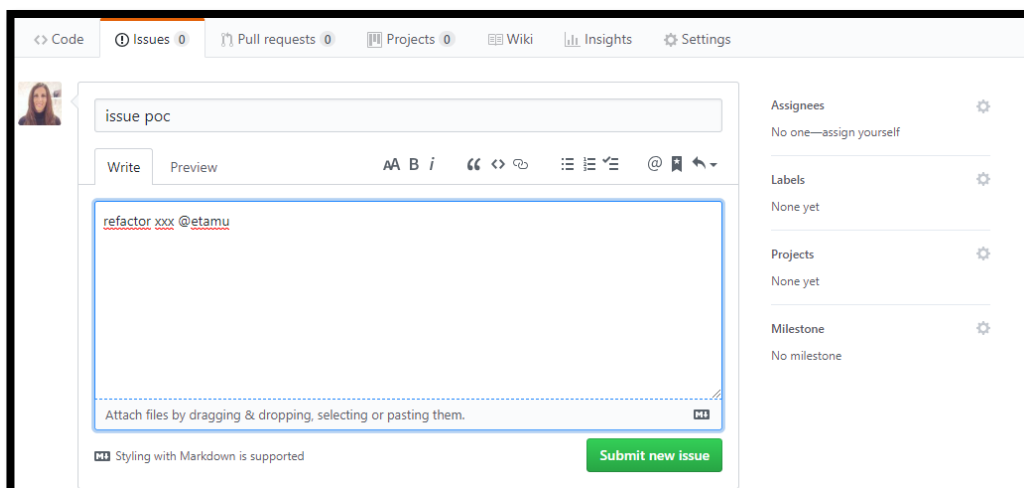
1. Identificar y **documentar** el máximo número posible de “**bad smells**” o indicadores de deterioro del código según los **principios de mantenibilidad** con la ayuda de la herramienta **BCH** (también **PMD** o similares) y **los apuntes de clase** (si se da el caso de que hay bad smells que BCH u otras herramientas no han detectado y el alumno sí).

Obtener el **diagrama de clases** de la aplicación a partir del código fuente (con UML Lab o Star UML). Adjuntar dicho diagrama en el documento y realizar las aclaraciones que considere el alumno, y si es necesario relacionar con los bad smells y principios de mantenibilidad identificados.

2. **Refactorizar** el código hasta obtener un nivel de conformidad aceptable al menos de **8 sobre 10 según BCH**. La máxima nota se obtendrá con un nivel de conformidad de 9 sobre 10. Se recuerda que una refactorización es cualquier cambio que suponga un aumento de calidad del código sin variar el comportamiento del mismo.

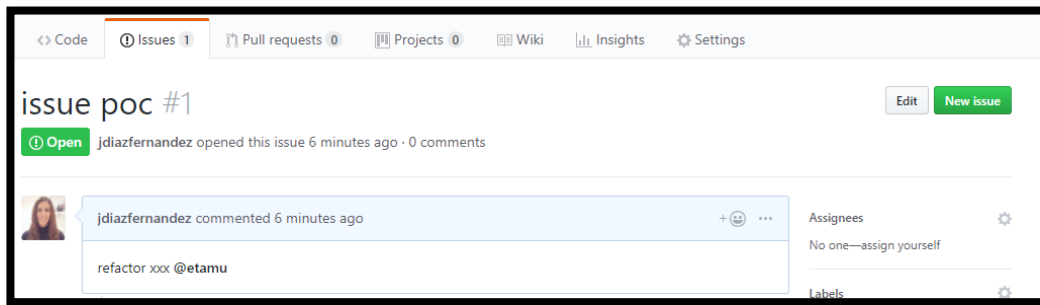
Especificar un **diagrama de clases** que recoja los cambios/mejoras anteriores en el caso de haya habido una refactorización de clases. Esta tarea es recomendable que se realice antes de implementar las refactorizaciones, para que el equipo tenga claro, a nivel de diseño, el objetivo de los cambios a implementar.

Cada refactorización quedará guardada en un **issue** de GitHub<sup>1</sup>. 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).

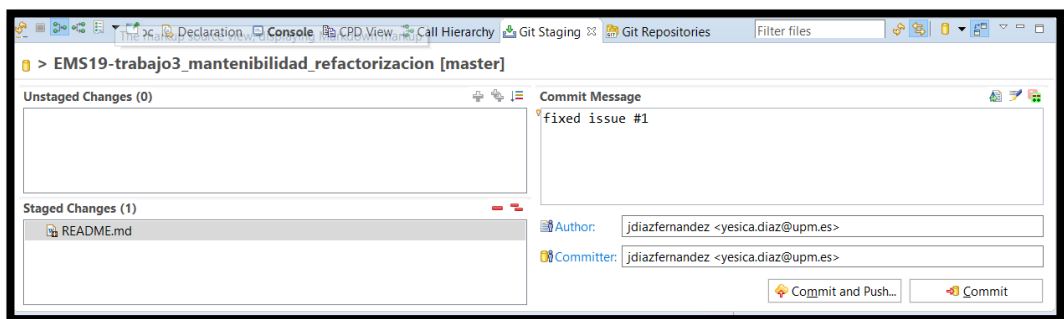
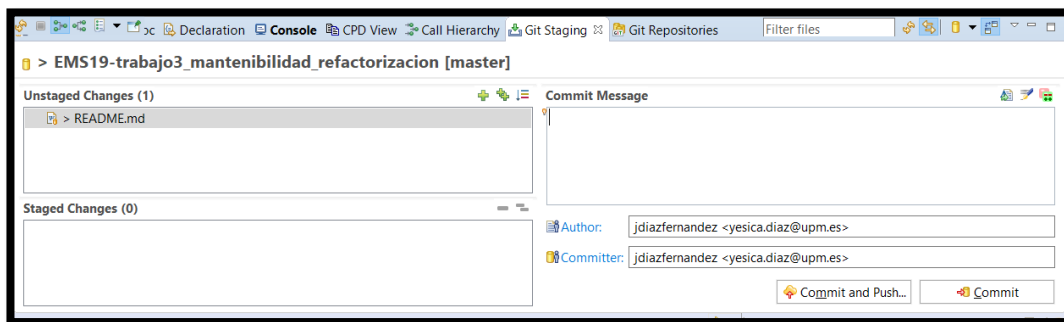


Una vez dado de alta el issue, GitHub asigna un ID (en este caso #1) como se ve en la figura:

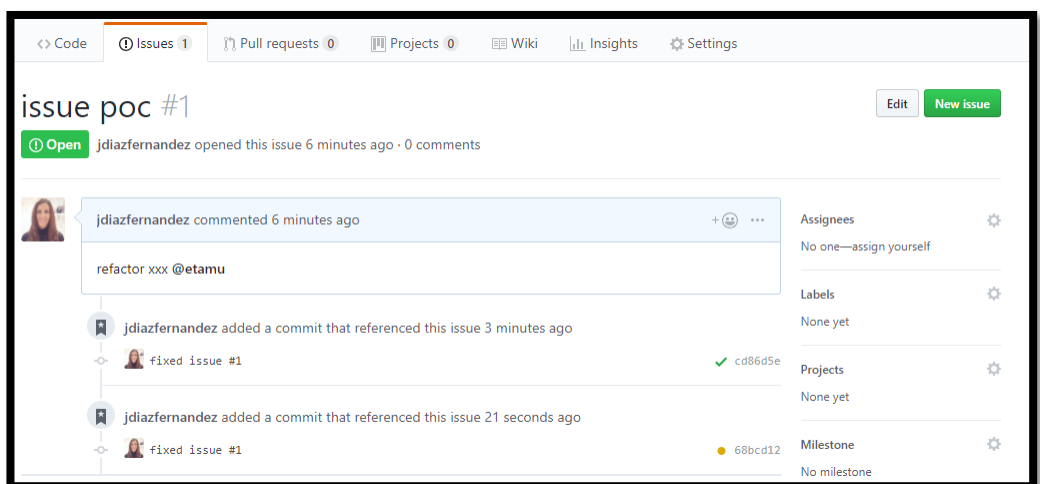
<sup>1</sup> No es requisito crear un proyecto en GitHub ni trabajar con ramas como se hizo en el Trabajo 2



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. El alumno podrá utilizar la consola de Git o un IDE, según sus preferencias. A continuación, se muestra el uso de Git desde el IDE Eclipse (add, commit y push). 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:

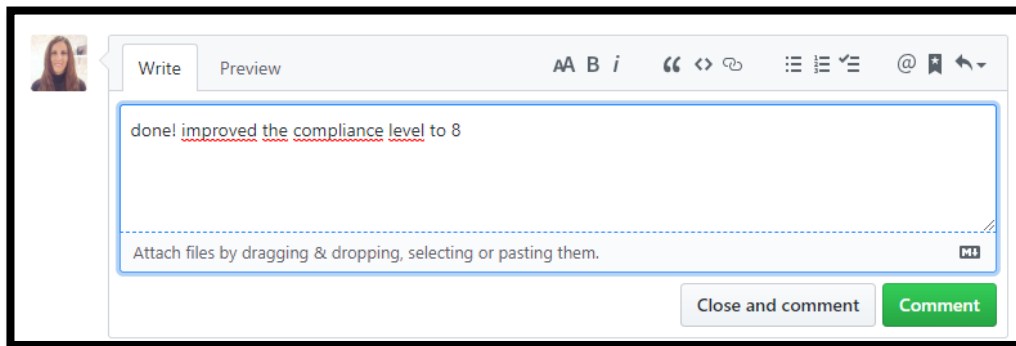


Desde GitHub se puede hacer el seguimiento de issues y sus commits:



Cerrar los issues cuando se hayan arreglado. Se recomienda añadir información del nivel de conformidad alcanzado con la refactorización realizada.





Write Preview


done! improved the compliance level to 8

Attach files by dragging & dropping, selecting or pasting them.

Close and comment Comment


## issue poc #1

**Closed** jdiazfernandez opened this issue 9 minutes ago · 1 comment




jdiazfernandez commented 9 minutes ago

refactor xxx @etamu




jdiazfernandez added a commit that referenced this issue 6 minutes ago




fixed issue #1

cd86d5e




jdiazfernandez added a commit that referenced this issue 3 minutes ago




fixed issue #1

68bcd12



jdiazfernandez commented just now

done! improved the compliance level to 8



jdiazfernandez closed this just now

En el documento **adjuntar una captura de los issues implementados** y realizar las aclaraciones que considere el alumno. Por último, analizar el grado de conformidad (*compliance level*) del código proporcionado con la herramienta Better Code Hub (BCH) y adjuntar en el documento una captura del **análisis final de calidad y mantenibilidad del código**.

## 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

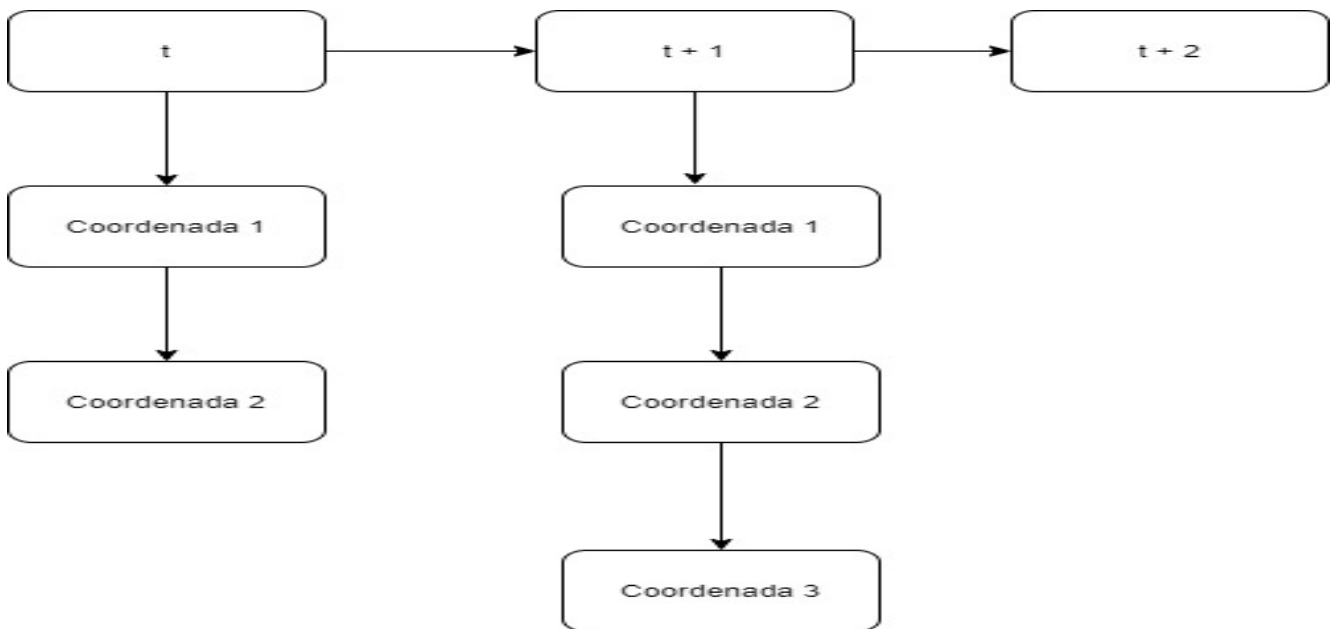
Coordenada → Representa una posición

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 un interface gráfico, 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.**