

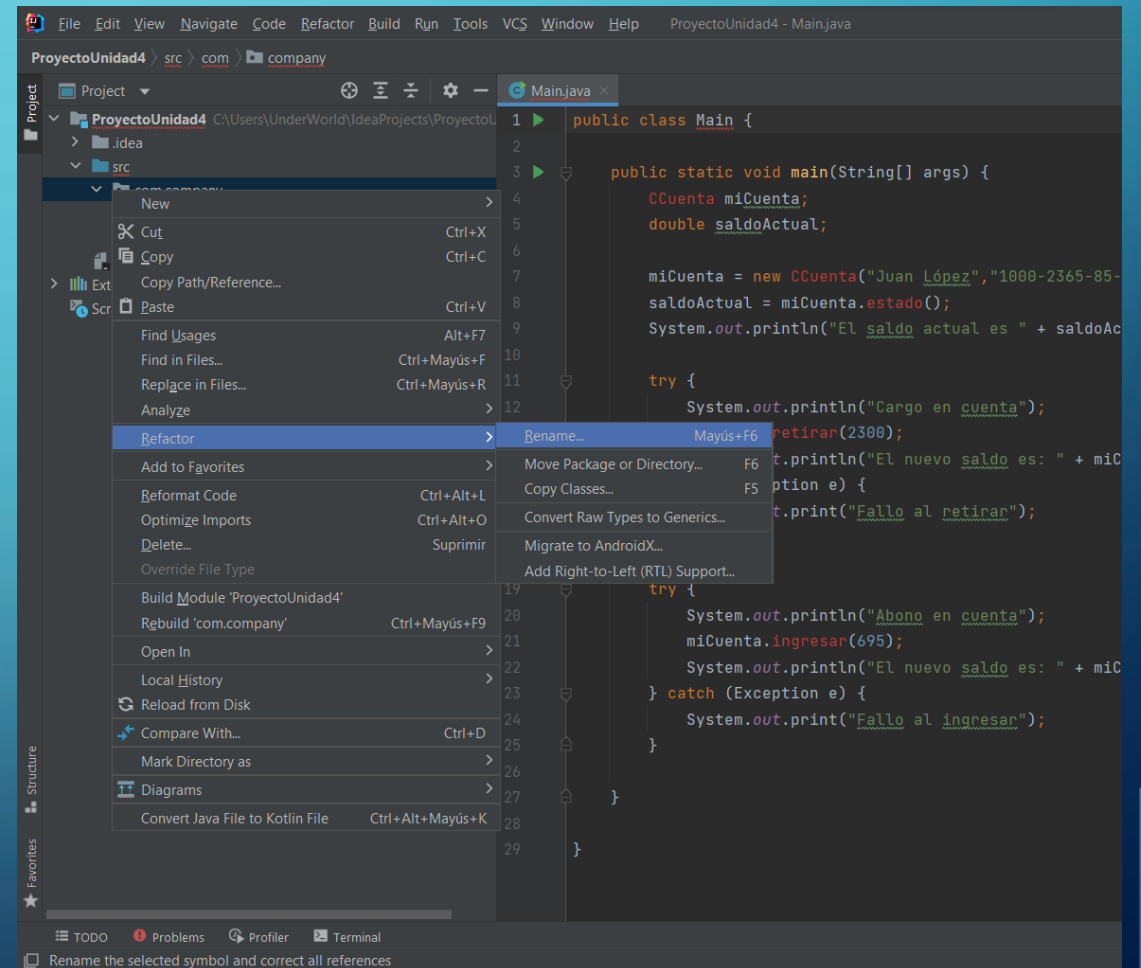
A decorative graphic on the left side of the slide consisting of white and light blue lines that resemble a circuit board or a tree structure, with small circles at the end of the branches.

PRÁCTICA FINAL UT4: OPTIMIZACIÓN Y DOCUMENTACIÓN.

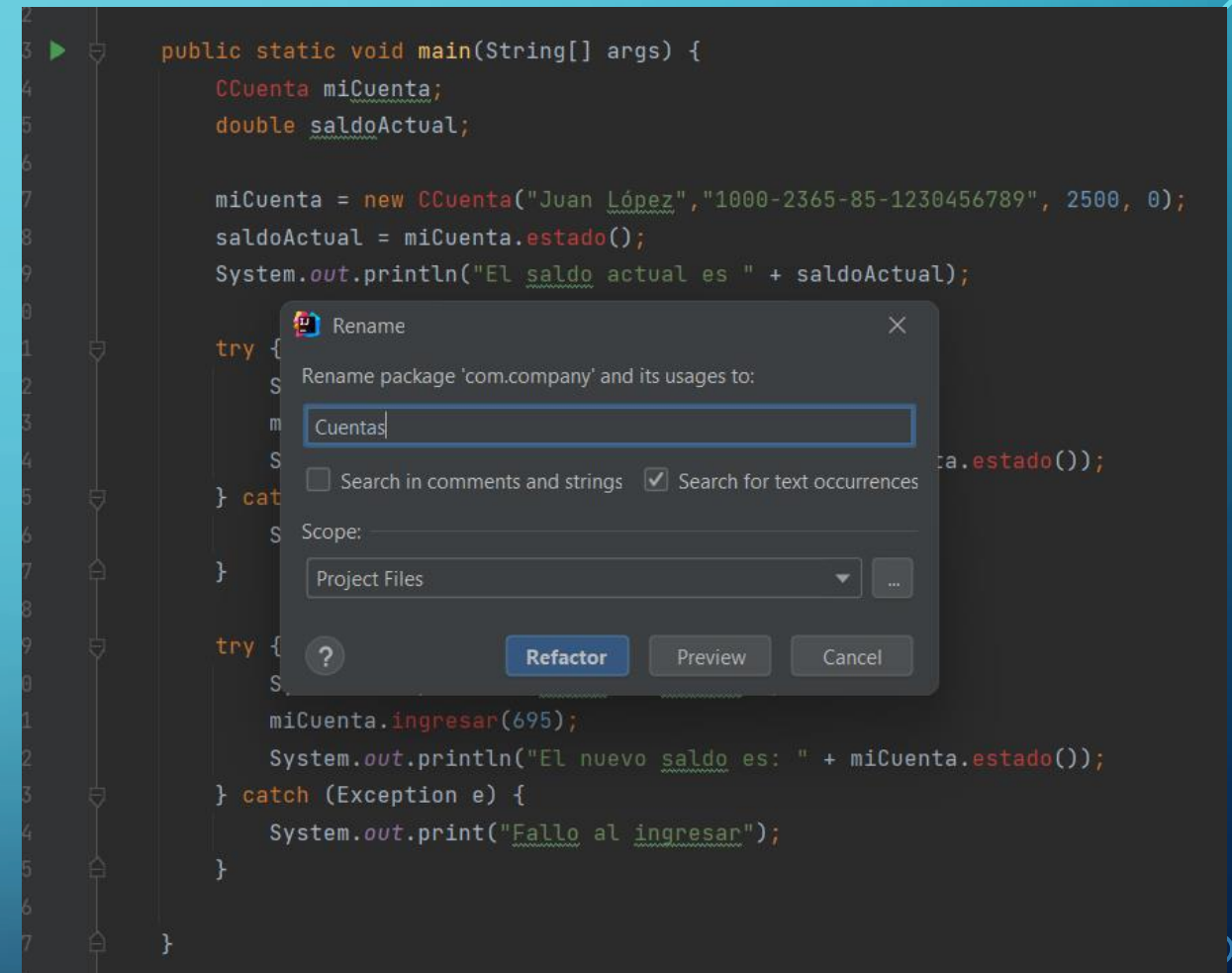
FABIO YAMUZA GONZALEZ

REFACTORIZAR CODIGO

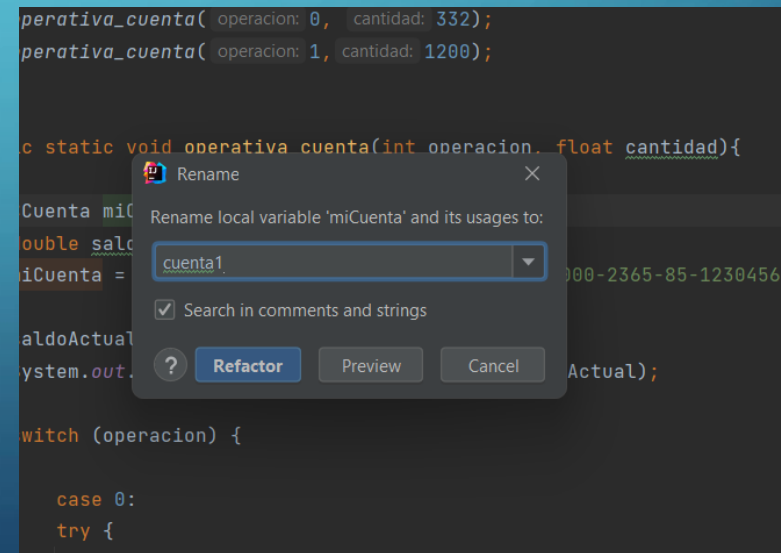
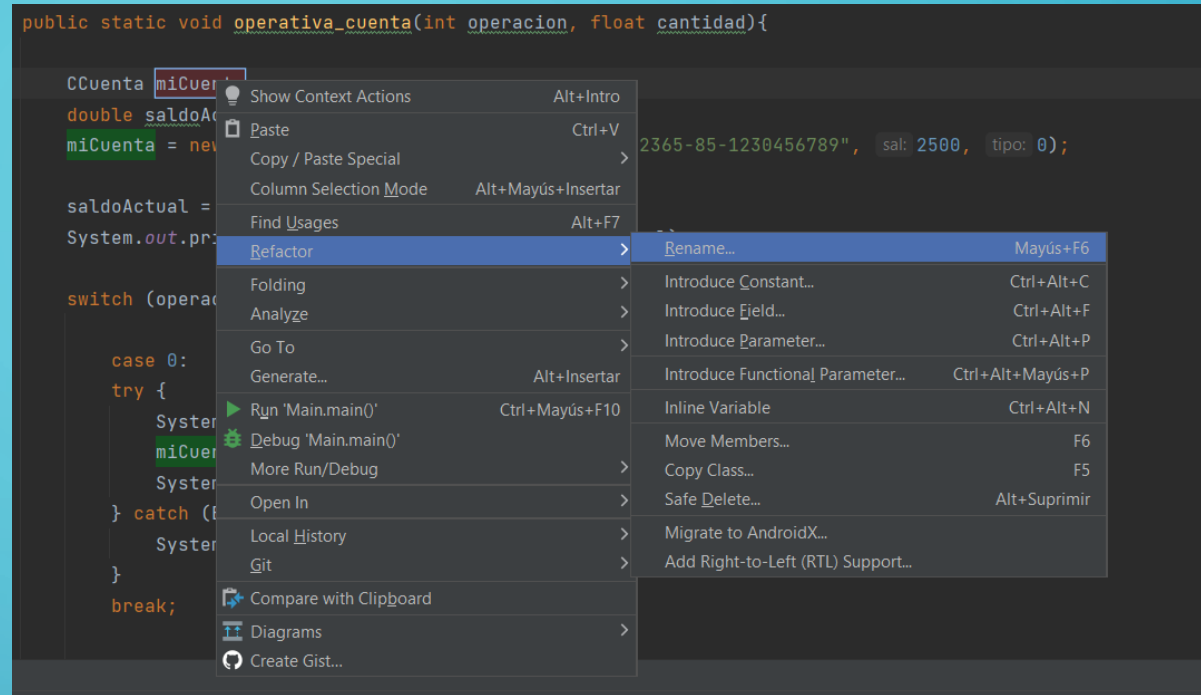
- Se han seguido una serie de pasos para refactorizar el código de la aplicación por defecto que venia incluida en el trabajo de la UT4, lo primero que se ha hecho es cambiar el nombre del paquete en el que viene el main y la clase Ccuenta



- En la imagen anterior nos dirigimos a la izquierda del entorno y buscamos entre las carpetas del proyecto el paquete contenedor del main y el Ccuenta hacemos click derecho en el paquete y seleccionamos refactor nos saldrá otro menú emergente y esta vez damos click en rename...
- Escribimos el nuevo nombre del paquete clickeamos en refactor y ya estaría refactorizado el nombre del paquete en el proyecto entero

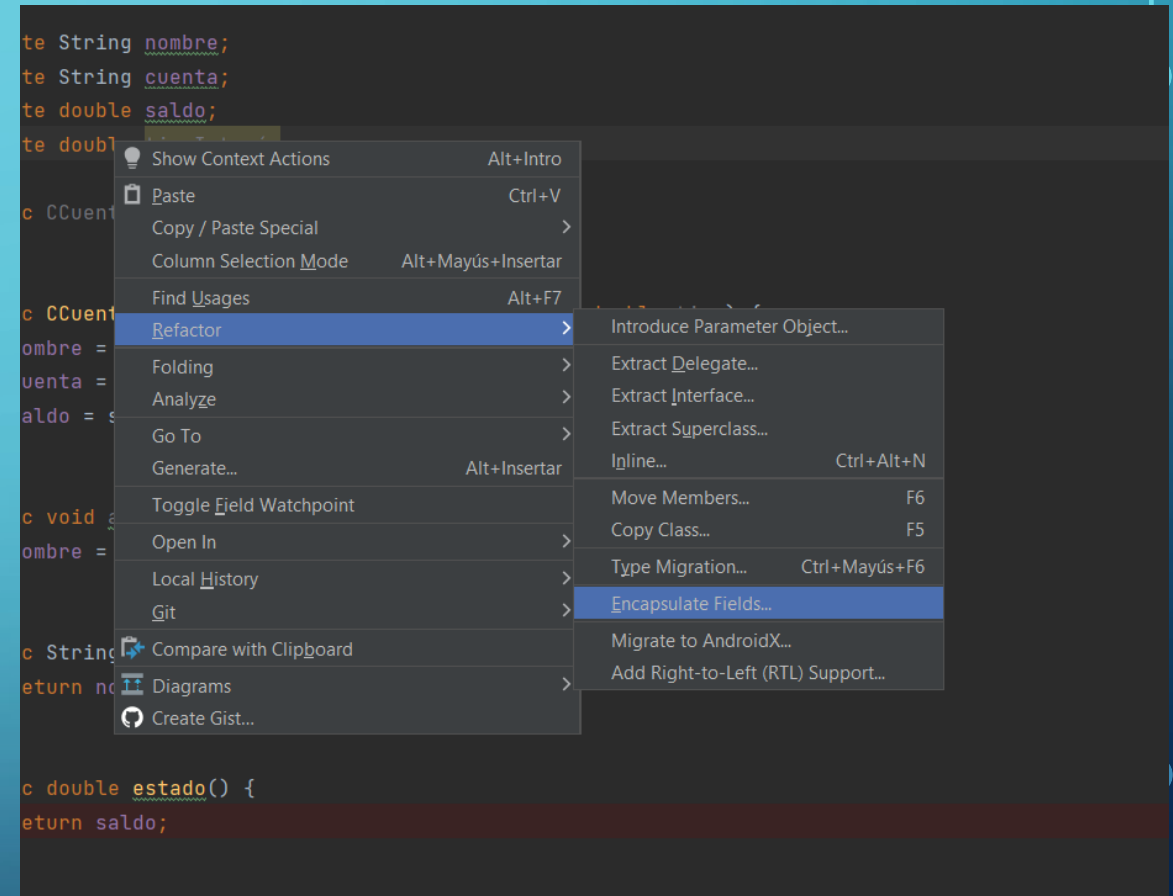


- Ahora vamos refactorizar el nombre de una variable del main y utilizaremos el mismo método mostrado en el punto anterior click derecho en la variable que queremos renombrar, click en refactor, click en rename, añadimos el nombre nuevo y click de nuevo en refactor



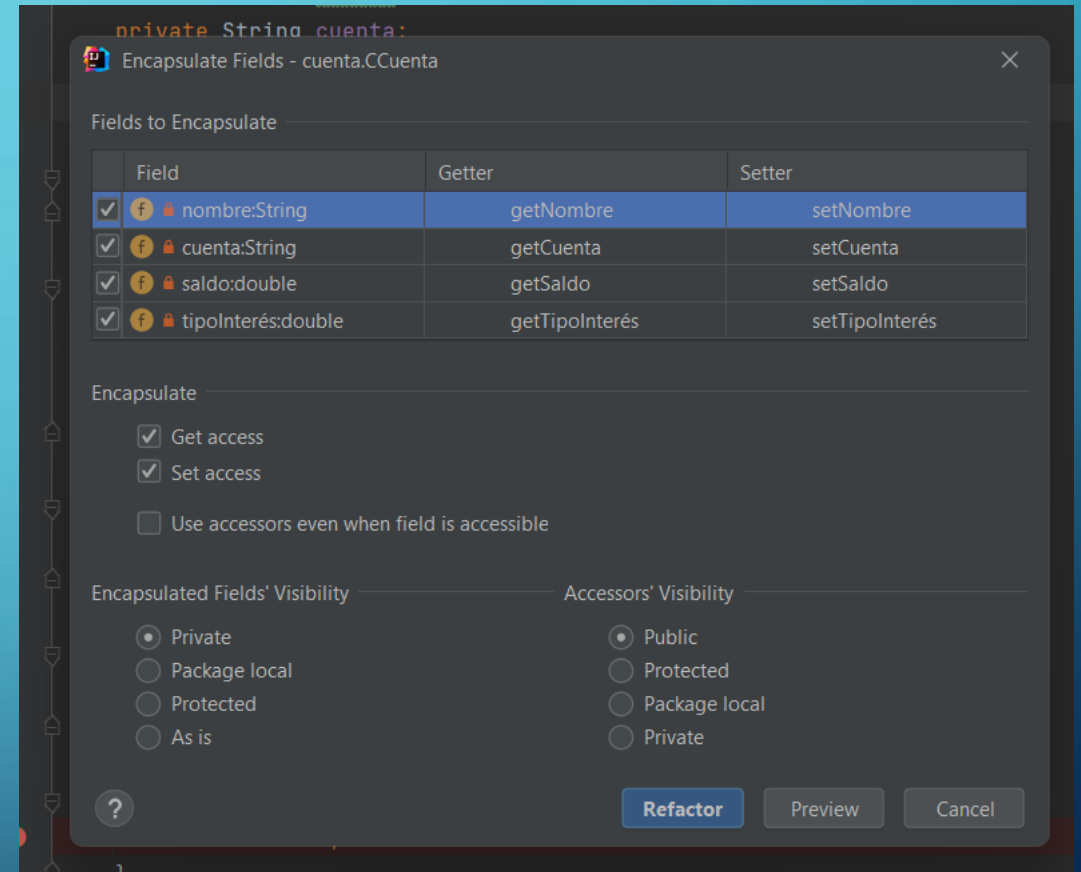
- Los siguientes dos pasos de refactorización al no usar la opción refactor no se han realizado capturas ya que se ha hecho de manera manual en el mismo main (estos cambios se podrán ver directamente en el repositorio master creado en mi cuenta github), los cambios han sido la creación de un método llamado `operativa_cuenta`, que englobe las sentencias de la clase main que operan con el objeto `cuenta1`.
- A este método se le parametrizaran dos variables una de tipo entero llamada `operación` y otra de tipo float llamada `float`, este método se utilizara para comprobar tanto el ingreso a una cuenta como retirada de dinero.

- El ultimo punto de refactorización es el encapsulamiento de los atributos de la cuenta CCuenta, se vuelve a usar la opción refactor, nos dirigimos a donde están declaramos los parámetros damos click derecho, refactor, y click en este, saldrá un submenú y escogeremos la opción encapsulate fields.



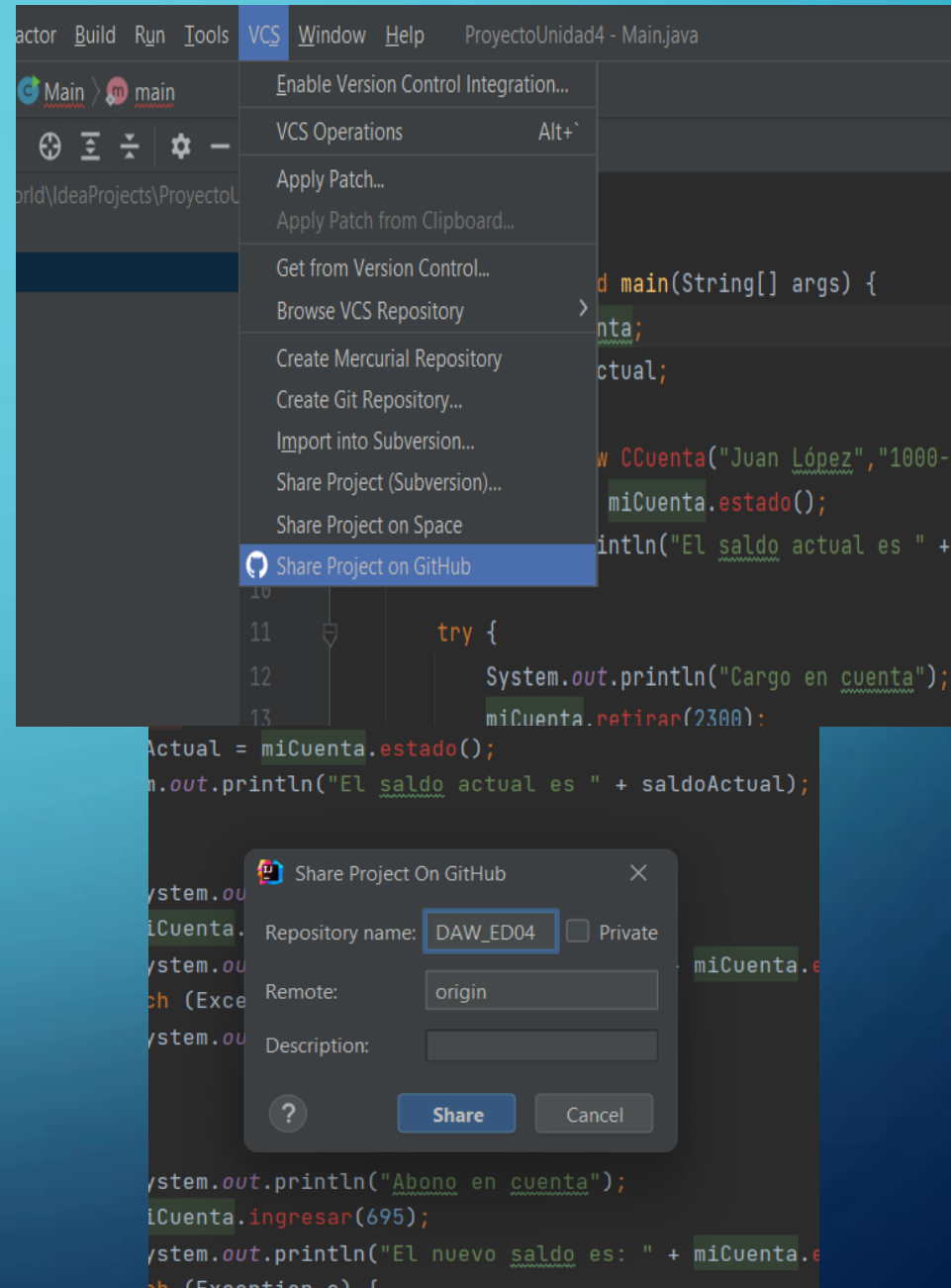
Se nos abrirá un menú en el cual marcaremos todos los atributos de la clase y en el apartado encapsulate marcaremos get y set pero desmarcamos Use accessors even when fields is accessible ya que este nos modificara el constructor parametrizado que tenemos ya creado, una vez encapsulado los atributos borraremos los siguientes métodos ya que sería redundante tenerlos.

```
public void asignarNombre(String nom) {  
    nombre = nom;  
}  
  
public String obtenerNombre() {  
    return nombre;  
}  
  
public double estado() {  
    return saldo;  
}  
  
public String obtenerCuenta() {  
    return cuenta;  
}
```

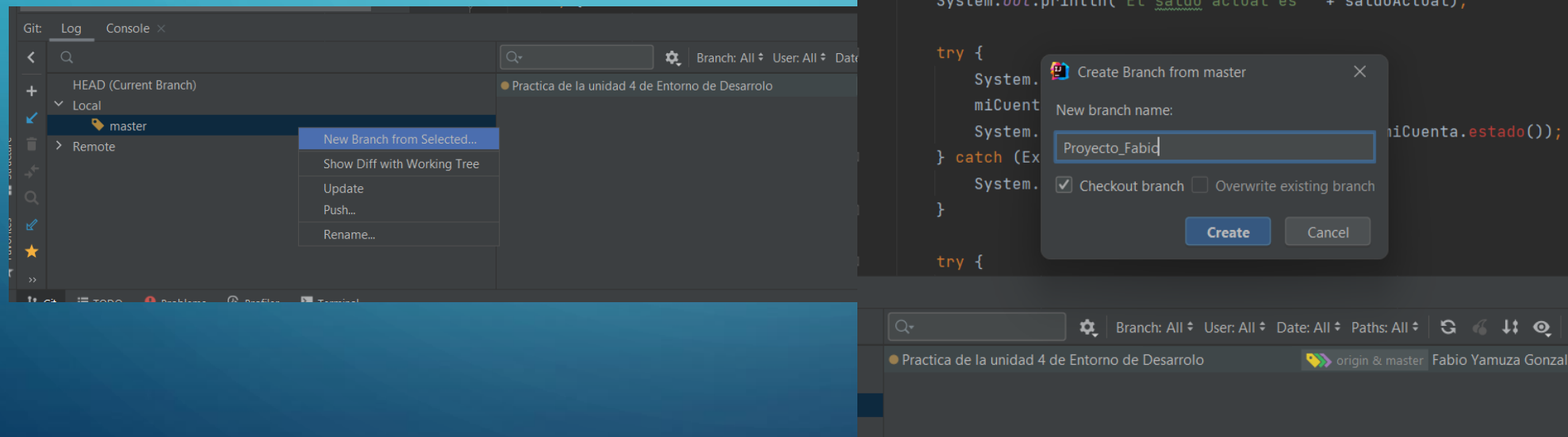


GIT

- El primer paso es crear un repositorio con este proyecto en git, escogeremos entre los menus superiores el que pone VCS y escogeremos share Project on GitHub, nos saltara una venta en la cual tendremos que darle nombre al repositorio y daremos click en share.

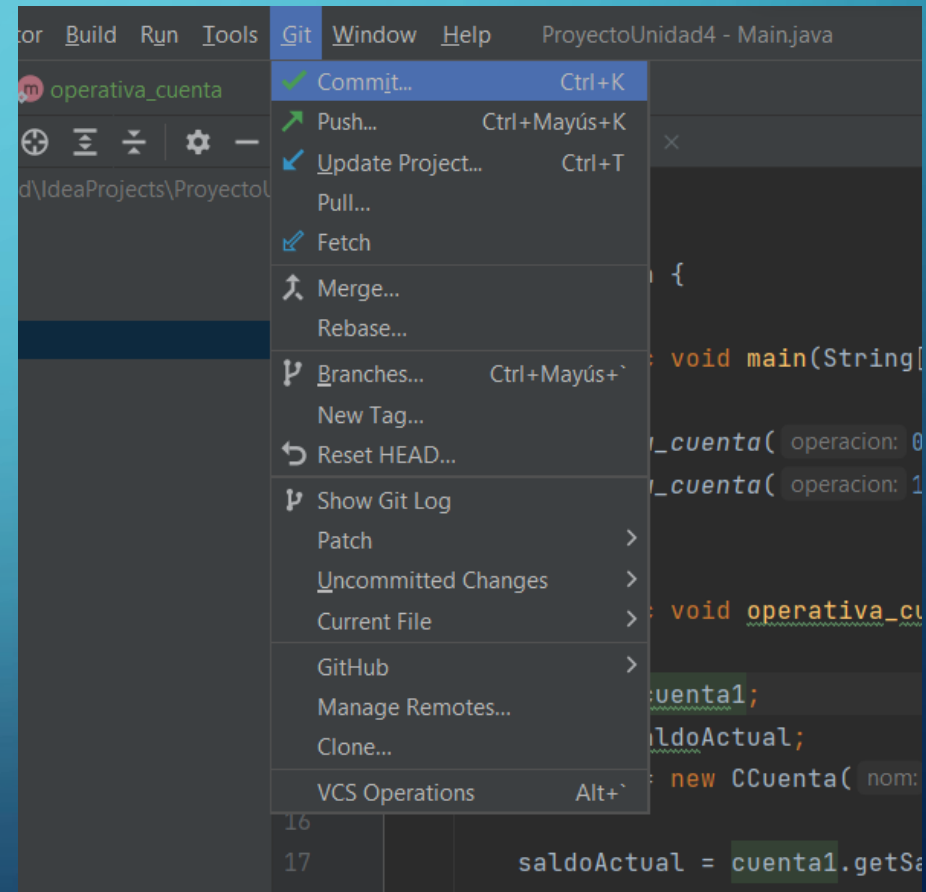


- Ahora vamos a crear una rama secundaria, en esta rama es donde vamos a realizar todos los cambios que se han explicado en el primera apartado del proyecto en el de refactorización, nos dirigimos a la esquina inferior derecha y clickeamos en git, damos click derecho en la rama master y escogemos new Branch from selected y le damos nombre.

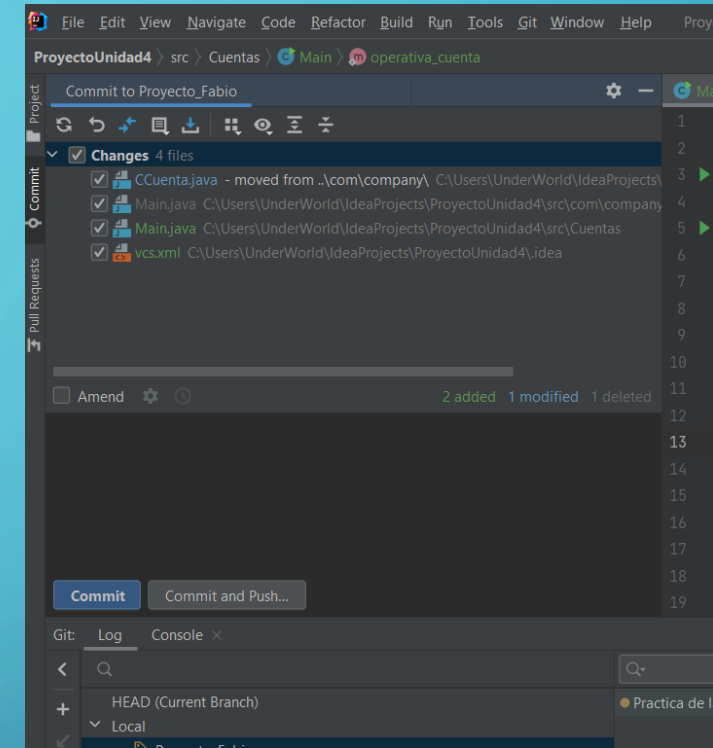
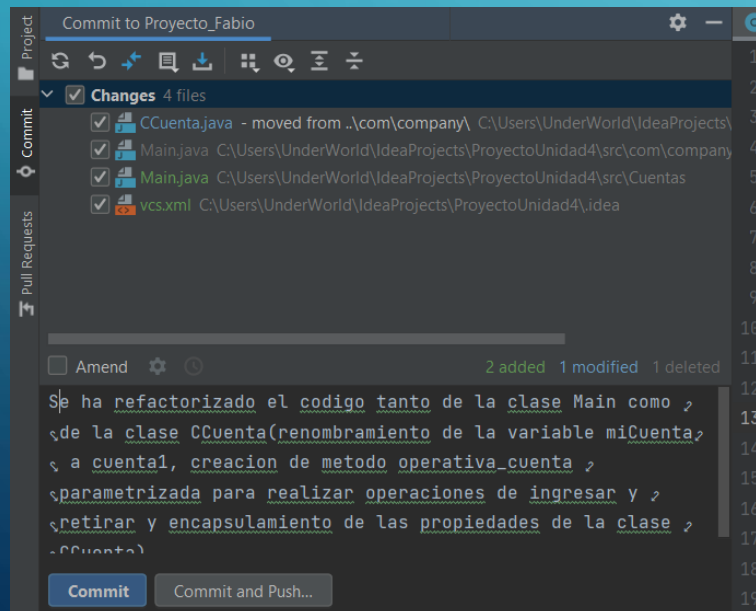


COMMIT

- En este apartado vamos a hacer un commit de los cambios de refactorización en nuestra rama secundaria dando un breve resumen comentado de los cambios realizados, click en la opción git del entorno y escogemos el que nos pone commit.



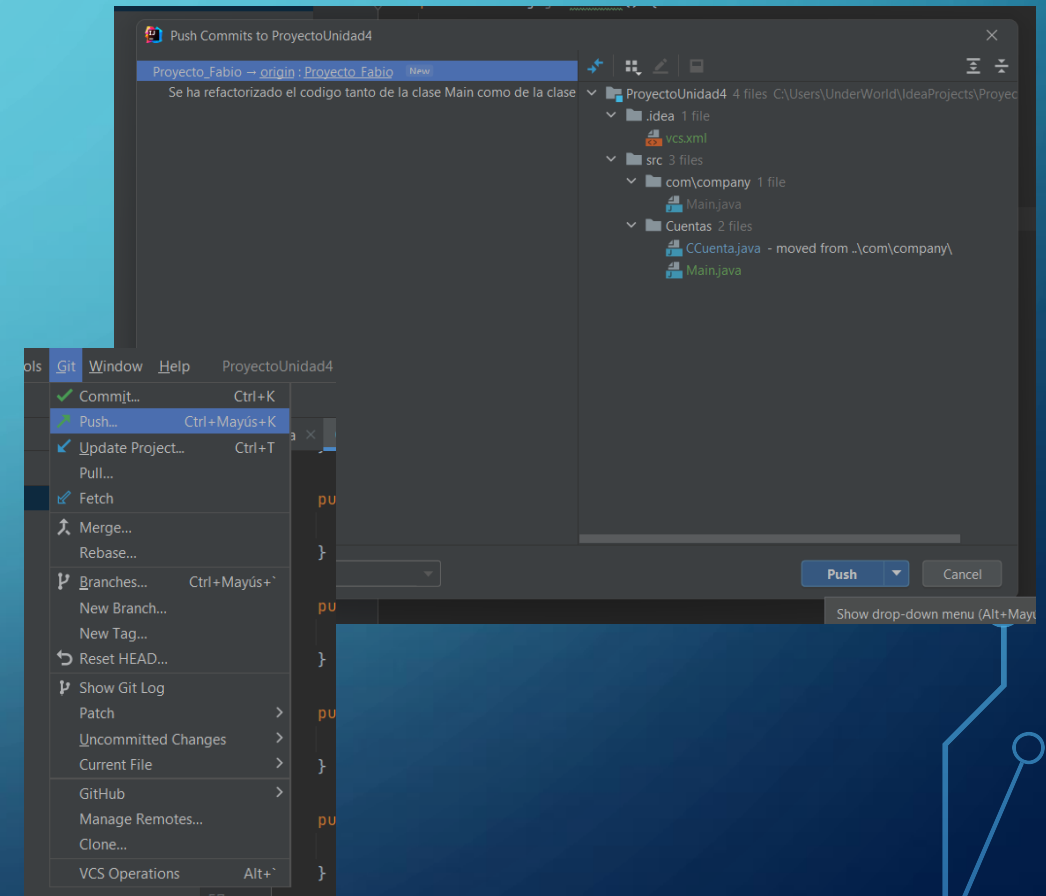
- Se seleccionan los archivos del proyecto de los que se quieren hacer el commit y en el cuadro de texto de abajo es donde se debe escribir la documentación de los cambios.



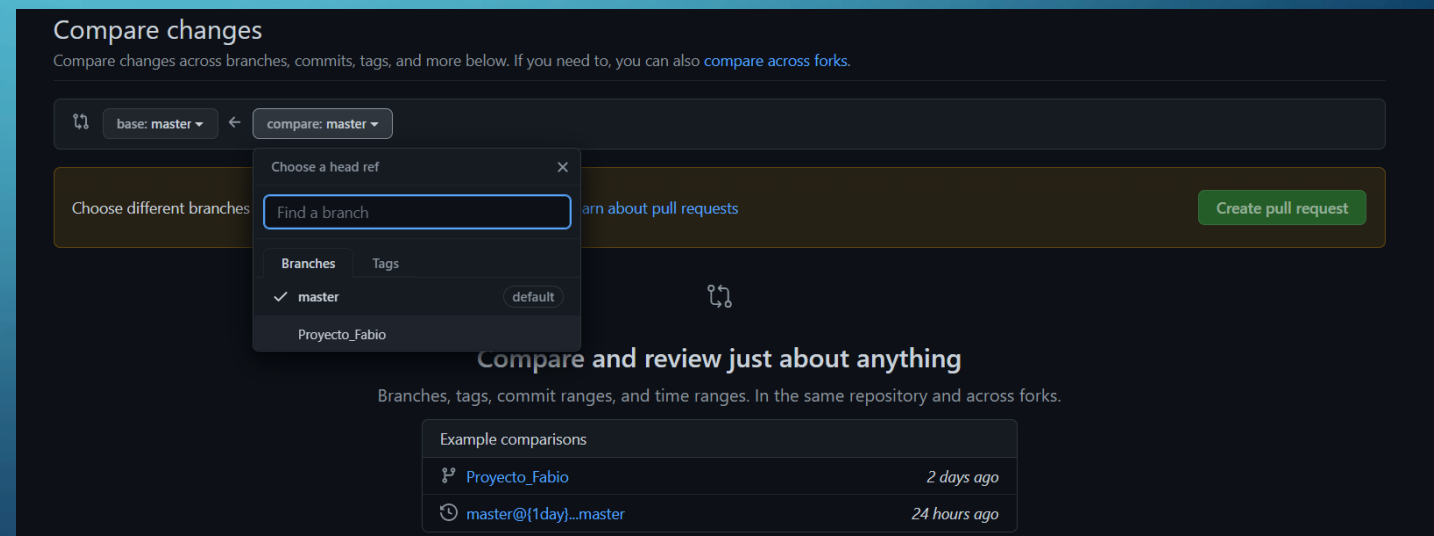
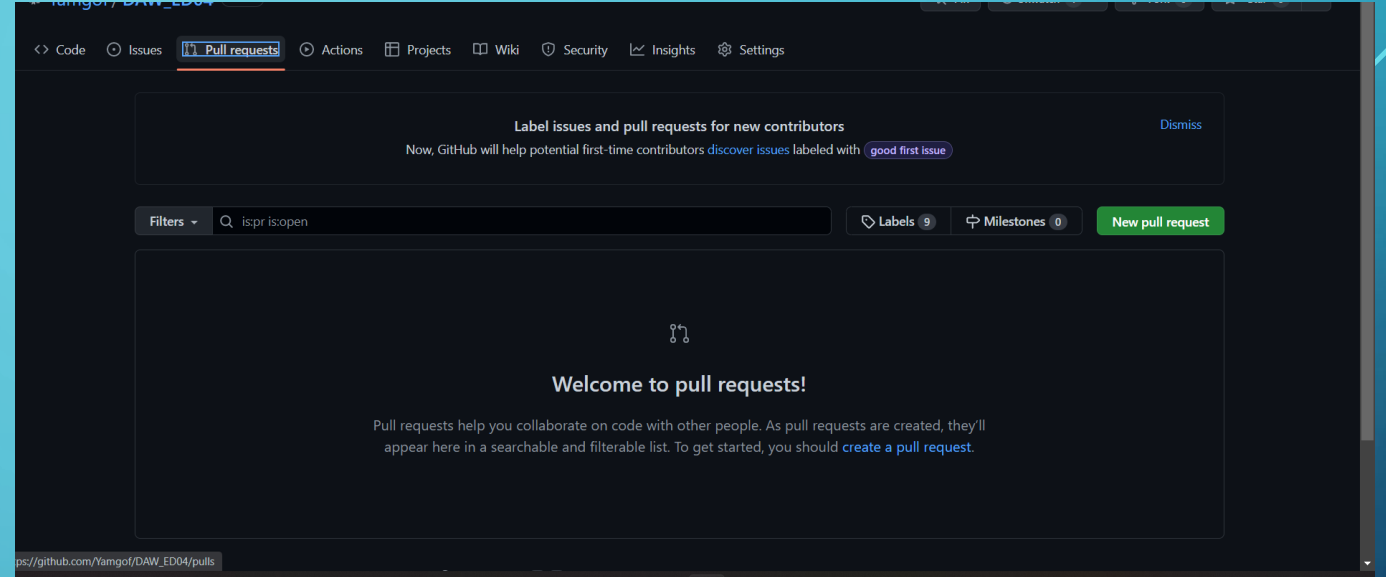
DE RAMA SECUNDARIA A PRINCIPAL

Ahora justo debajo de donde hicimos el commit tenemos la opción de push con la cual subiremos nuestros cambios de la rama secundaria a nuestra rama secundaria que tenemos en remoto.

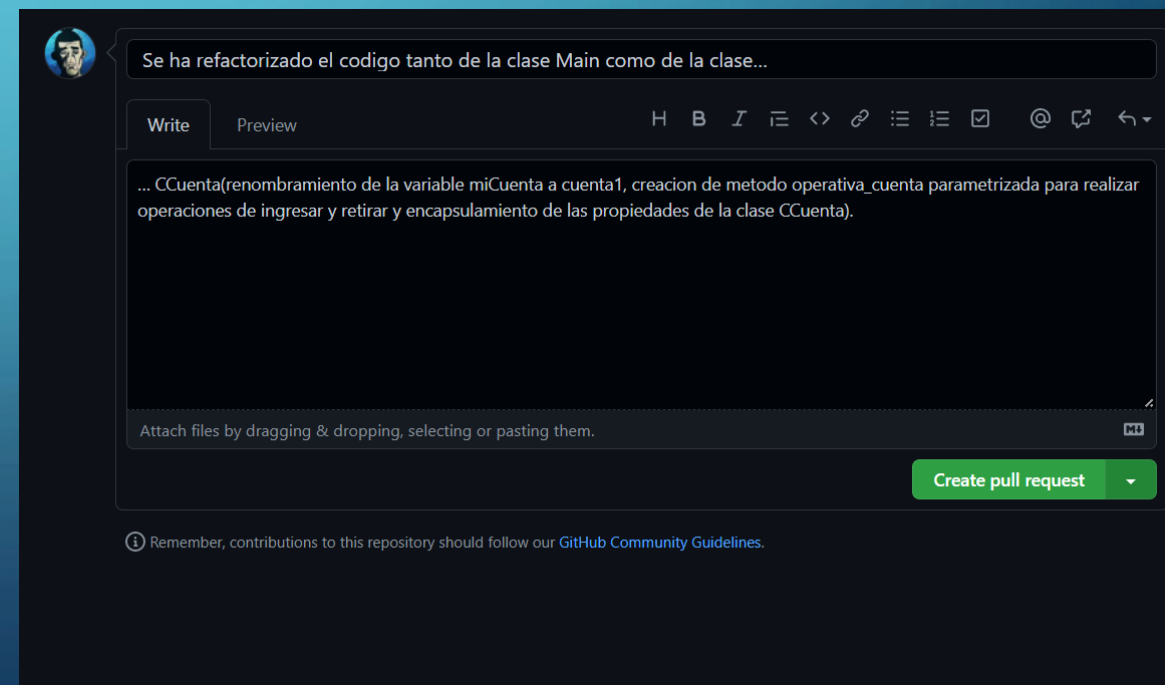
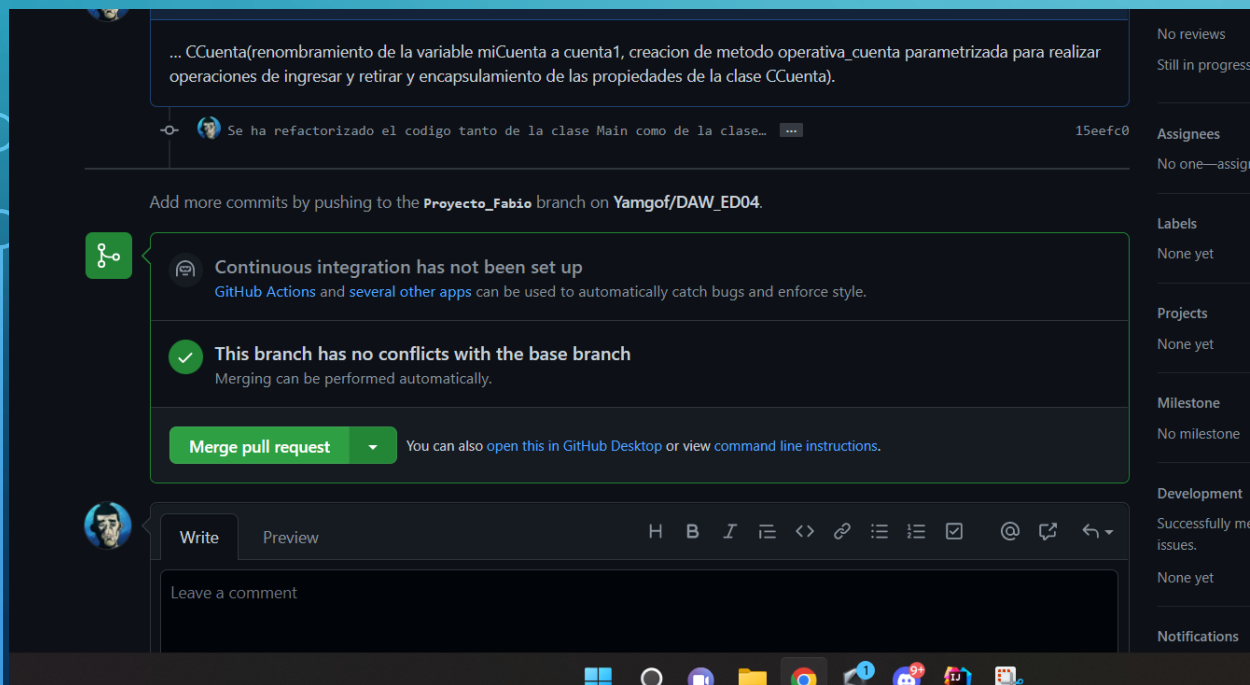
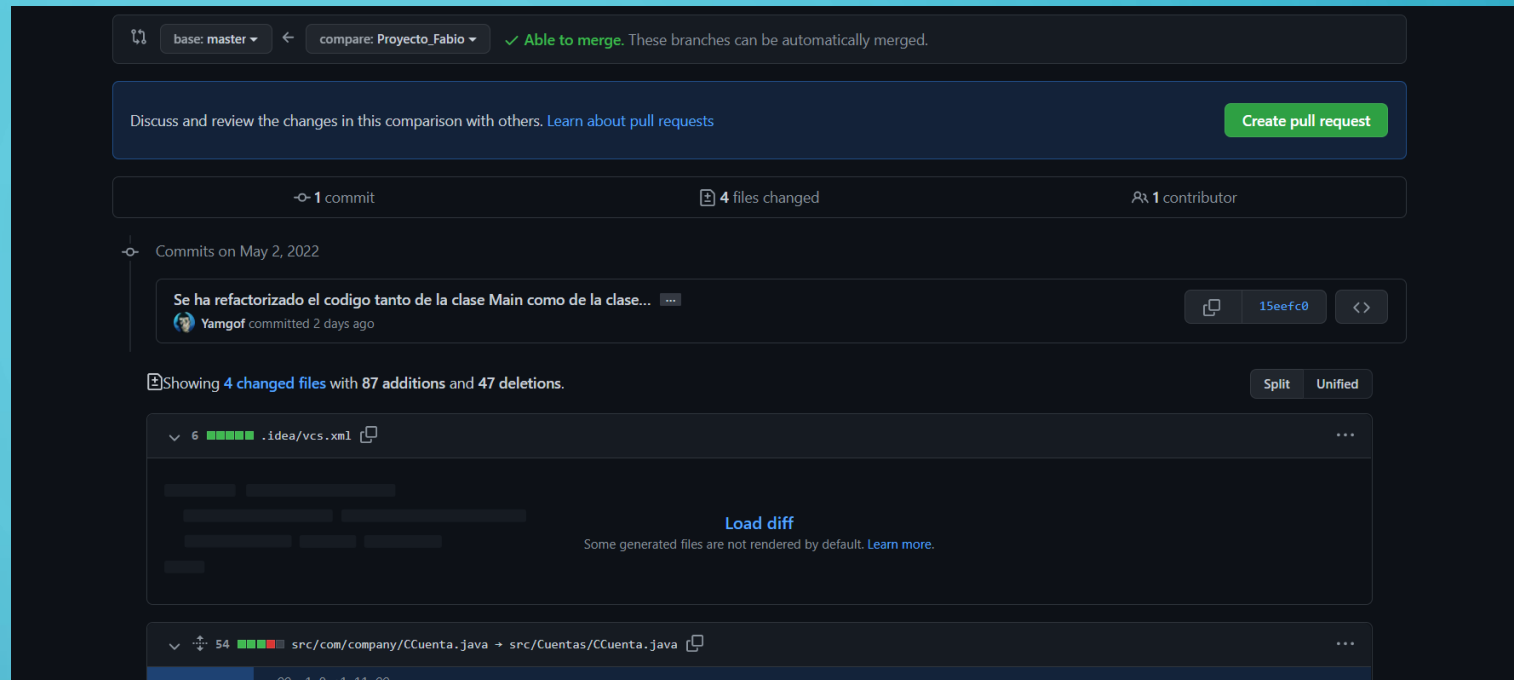
El siguiente paso si será hacer un pull request en el cual subiremos nuestros cambios de la rama secundaria a la rama principal pero esta vez si haremos esta operación desde el mismo github para manejar directamente nuestras ramas que están en remoto



- Dentro de la pagina de github abrimos nuestro repositorio y escogemos la acción pull request, una vez seleccionada tendremos que especificar de que rama a que rama haremos el pull, y se ha cogido la rama secundaria remota que aplique cambios a la rama master.



- Una vez escogido las ramas le daremos a create pull request, veremos el comentario del commit para saber los cambios que vamos a hacer pull una vez hecho esto solo tendremos que mergear los cambios.



JAVADOC

- Ahora vamos a generar un archivo javadoc, pero antes vamos a agregar una serie de comentarios en la que vamos a definir el nombre del autor y explicaremos la función de cada parámetro que le pasamos a los métodos y en que influye y explicaremos que va a devolver los métodos que nos esten devolviendo algo. Cuando tengamos esto nos dirigimos a tools y escogeremos generate javadoc, nos saldrá una ventana emergente en la que tendremos que poner que queremos que muestre el javadoc y donde queremos guardarlo en la siguiente diapositiva estarán las imágenes del proceso.

