ENTORNOS DE DESARROLLO

Tarea ED04. CURSO 2022/2023.

CRISTINA CORINO LOPEZ

Contenido disponible en:

<https://github.com/criscorino/ED04>

**-INDICE –**

[**REFACTORIZACIÓN** 3](#_Toc129711689)

[**GIT** 8](#_Toc129711690)

[**JAVADOC** 13](#_Toc129711691)

# **- Listado de Ilustraciones -**

[Ilustración 1. Cambio de nombre del paquete (I) 3](#_Toc129885142)

[Ilustración 2. Cambio de nombre del paquete (II) 3](#_Toc129885143)

[Ilustración 3. Resultado de cambio de nombre del paquete 3](#_Toc129885144)

[Ilustración 4. Cambio de nombre de la variable (I) 4](#_Toc129885145)

[Ilustración 5. Cambio de nombre de la variable (II) 4](#_Toc129885146)

[Ilustración 6. Resultado de cambio de nombre de la variable 4](#_Toc129885147)

[Ilustración 7. Introducción del método (I) 5](#_Toc129885148)

[Ilustración 8. Introducción del método (II) 5](#_Toc129885149)

[Ilustración 9. Resultado introducción del método 5](#_Toc129885150)

[Ilustración 10. Encapsular atributos (I) 6](#_Toc129885151)

[Ilustración 11. Encapsular atributos (II) 6](#_Toc129885152)

[Ilustración 12. Resultado de encapsular atributos 6](#_Toc129885153)

[Ilustración 13. Añadir un nuevo parámetro al método (I) 7](#_Toc129885154)

[Ilustración 14. Añadir un nuevo parámetro al método (II) 7](#_Toc129885155)

[Ilustración 15. Resultado de añadir un nuevo parámetro al método 7](#_Toc129885156)

[Ilustración 16. Creamos cuenta en Github 8](#_Toc129885157)

[Ilustración 17. Creamos el repositorio ED04 8](#_Toc129885158)

[Ilustración 18. Dirección del repositorio 8](#_Toc129885159)

[Ilustración 19. Descarga de la aplicación Github Desktop 9](#_Toc129885160)

[Ilustración 20. Log y clonación del repositorio 9](#_Toc129885161)

[Ilustración 21. Vinculación con el repositorio 9](#_Toc129885162)

[Ilustración 22. Primer commit 10](#_Toc129885163)

[Ilustración 23. Realziamos commit 10](#_Toc129885164)

[Ilustración 24. Realizamos el Publish branch 10](#_Toc129885165)

[Ilustración 25. Github online 11](#_Toc129885166)

[Ilustración 26. Historial de versiones desde la aplicación de escritorio 11](#_Toc129885167)

[Ilustración 27. Instalamos Git por comando 11](#_Toc129885168)

[Ilustración 28. Configuración global 12](#_Toc129885169)

[Ilustración 29. Historial de versiones 12](#_Toc129885170)

[Ilustración 30. Generar JavaDoc 14](#_Toc129885171)

[Ilustración 31. JavaDoc 14](#_Toc129885172)

[Ilustración 32. Clase CCuenta en el JavaDoc 14](#_Toc129885173)

[Ilustración 33. Commit desde la CMD 15](#_Toc129885174)

[Ilustración 34. Commit disponible online 15](#_Toc129885175)

**REFACTORIZACIÓN**

**Las clases deberán formar parte del paquete cuentas.**

**Cambiar el nombre de la variable "miCuenta" por "cuenta1".**

**Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.**

**Encapsular los atributos de la clase CCuenta.**

**Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.**

**GIT**

**Configurar GIT para el proyecto. Crear un repositorio público en GitHub.**

**Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.**

**Mostrar el historial de versiones para el proyecto mediante un comando desde consola.**

**JAVADOC**

**Insertar comentarios JavaDoc en la clase CCuenta.**

**Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.**

***Criterios de puntuación. Total 10 puntos.***

1. **Cambia el nombre de la variable "miCuenta" por "cuenta1". = 1 punto.**
2. **Introduce el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1. = 1 punto.**
3. **Encapsula los cuatro atributos de la clase CCuenta. = 1 punto.**
4. **Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float. = 1 punto.**
5. **Configurar GIT para el proyecto. Crear un repositorio público en GitHub.= 1 punto.**
6. **Realiza, al menos, una operación commit, comentando el resultado de la ejecución. = 1 punto.**
7. **Mostrar el historial de versiones para el proyecto mediante un comando desde consola.= 1 punto.**
8. **Inserta comentarios Javadoc en la clase Ccuenta. = 1 punto.**
9. **Genera documentación Javadoc para todo el proyecto. = 1 punto.**
10. **Comprueba que la documentación generada por Javadoc, abarca todos los métodos y atributos de la clase Ccuenta. = 1 punto.**

# **REFACTORIZACIÓN**

**Las clases deberán formar parte del paquete cuentas.**

Por error este apartado se realizó en último lugar. Para modificar el nombre del paquete debemos clicar sobre el y dentro de la opción Refactor clicamos sobre Rename…:

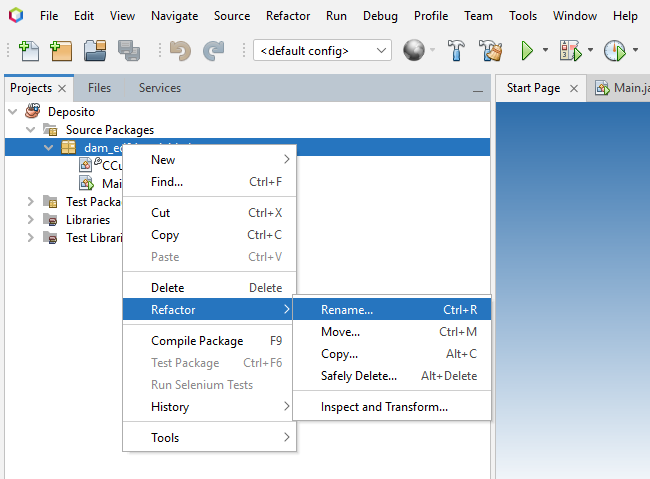


Ilustración . Cambio de nombre del paquete (I)

Indicamos a continuación el nombre que quedemos y marcamos la opción de que también cambie el nombre sobre los comentarios:

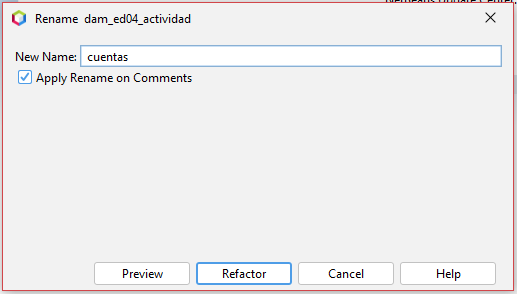


Ilustración . Cambio de nombre del paquete (II)

De esta forma podemos ver que queda de la siguiente forma:

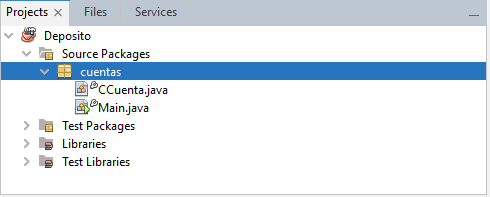


Ilustración . Resultado de cambio de nombre del paquete

**Cambiar el nombre de la variable "miCuenta" por "cuenta1".**

Seleccionamos la variable miCuenta y desde el menú Refactor, en Rename… modificamos el nombre de la variable y clicamos en Refactor, como vemos a continuación:

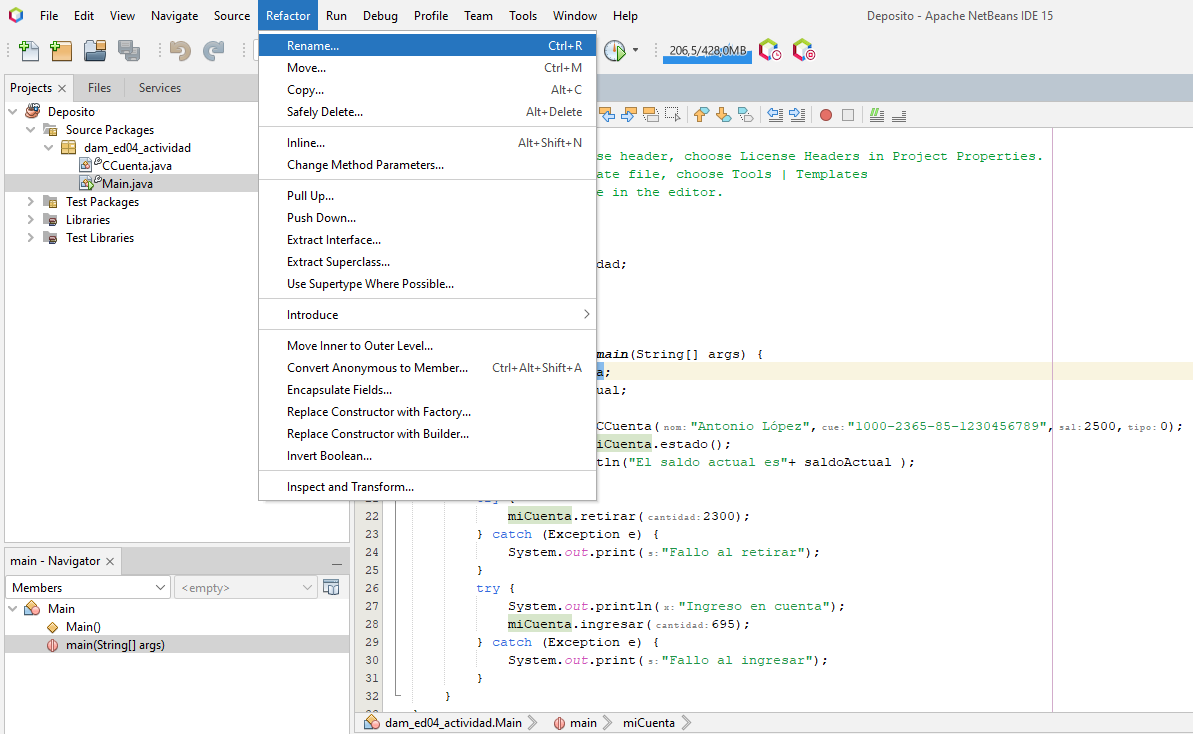


Ilustración . Cambio de nombre de la variable (I)

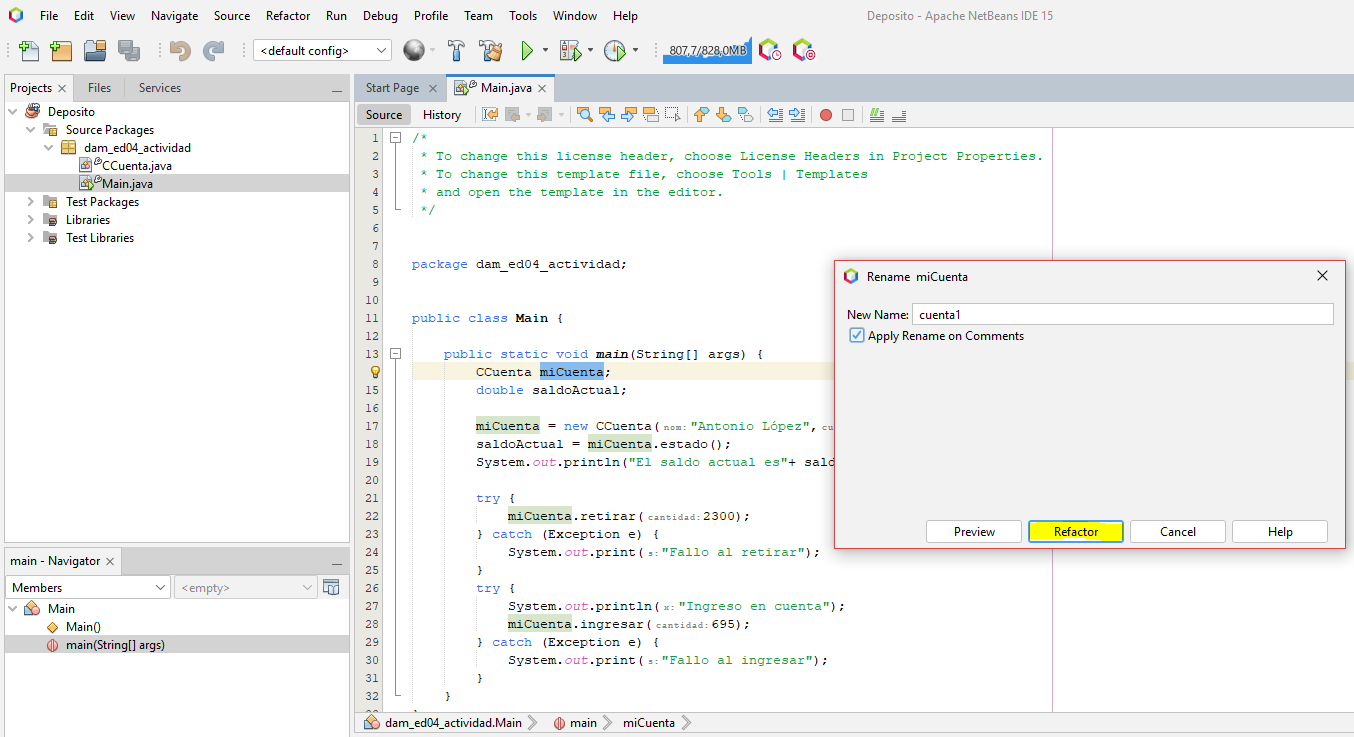


Ilustración . Cambio de nombre de la variable (II)

Con esto podemos ver que nos modifica lo siguiente:

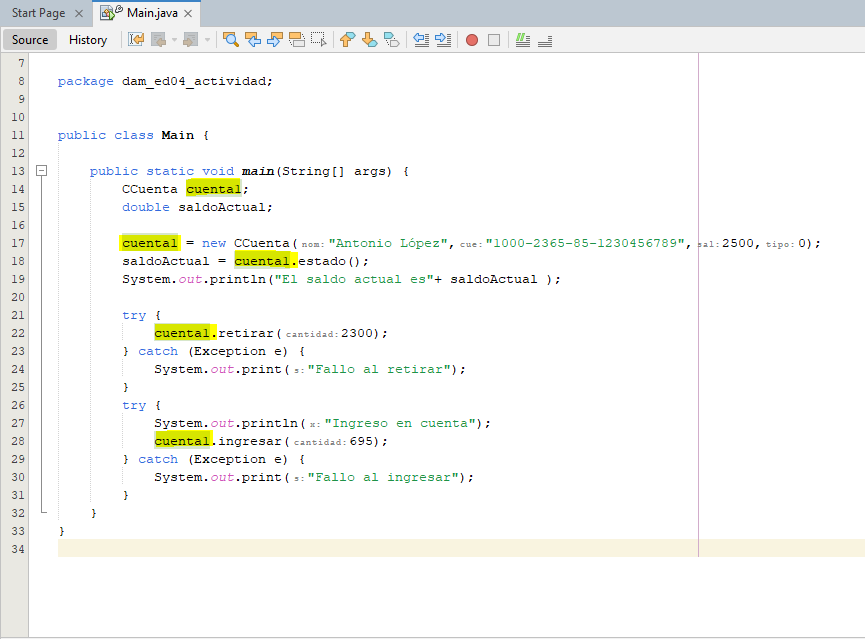


Ilustración . Resultado de cambio de nombre de la variable

**Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.**

A continuación, seleccionamos las sentencias mencionadas y desde el menú Refactor, en Introduce, vamos a Method…

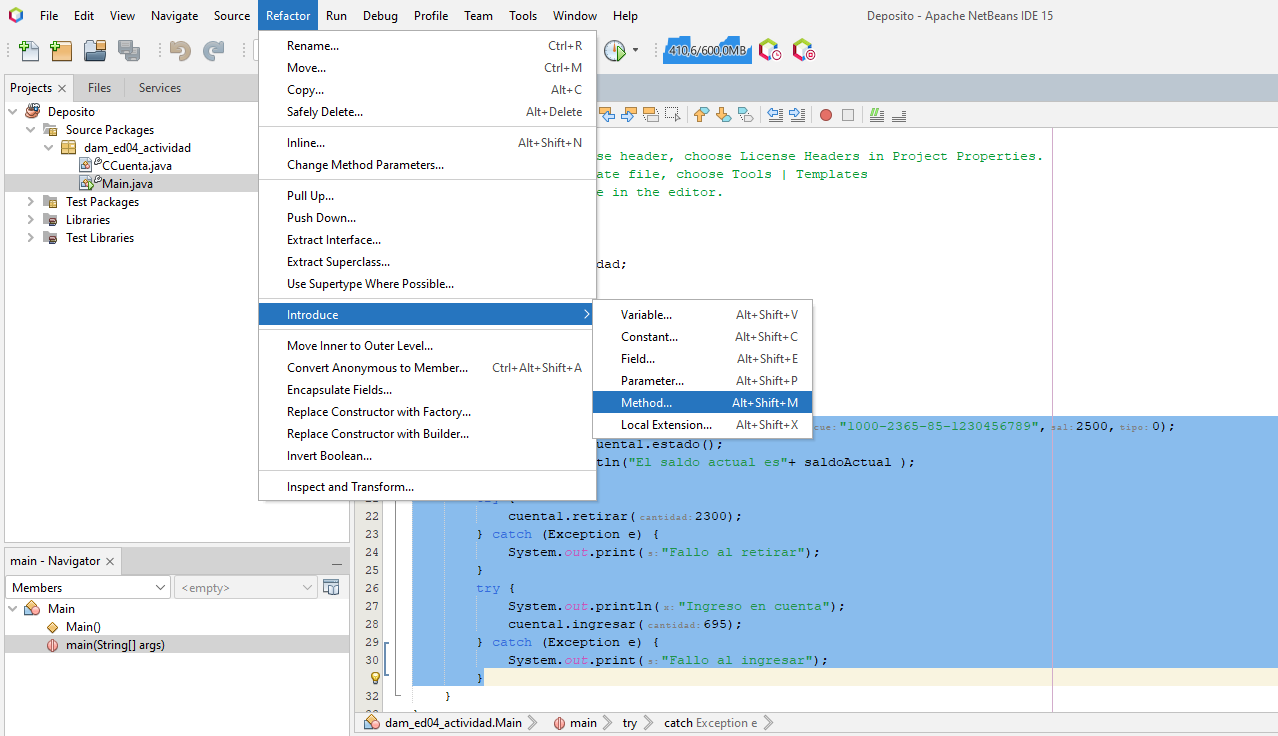


Ilustración . Introducción del método (I)

Como nombre indicaremos operativa\_cuenta, tendrá acceso privado, con ello de damos a Ok:

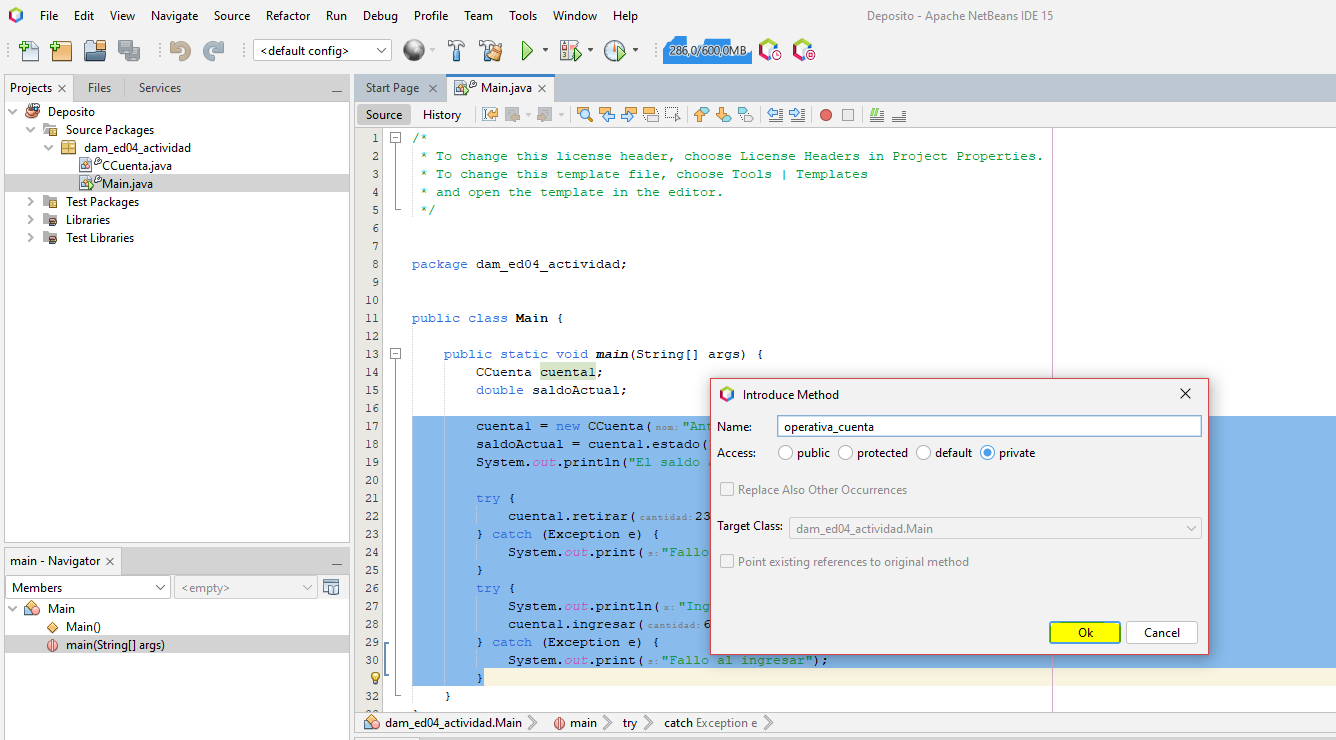


Ilustración . Introducción del método (II)

Con esto podemos ver en el código que nos incluye lo siguiente:

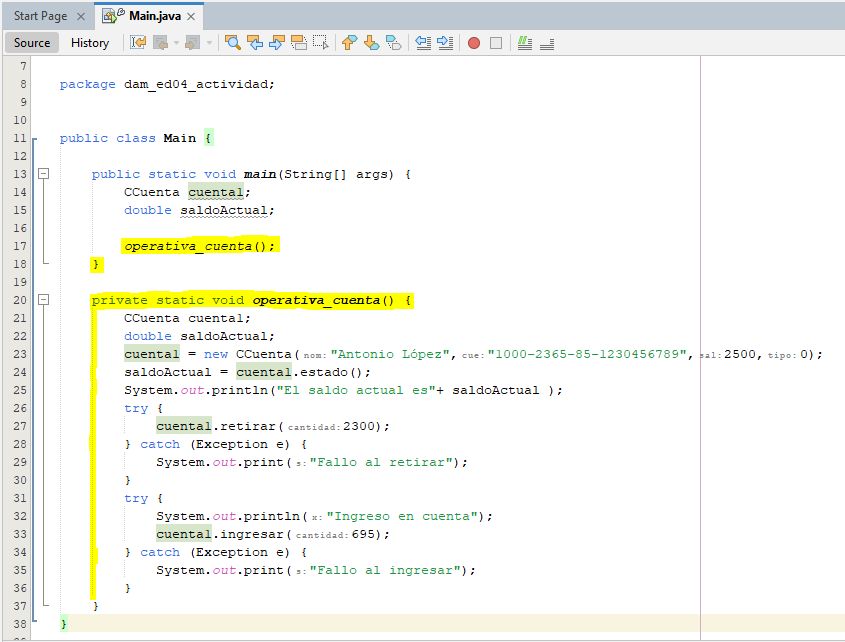


Ilustración . Resultado introducción del método

**Encapsular los atributos de la clase CCuenta.**

Para encapsular los atributos iremos a CCuenta.java ya que es ahí donde les podemos encontrar. Les seleccionamos y desde el menú Refactor hacemos clic en Encapsulate Fields..

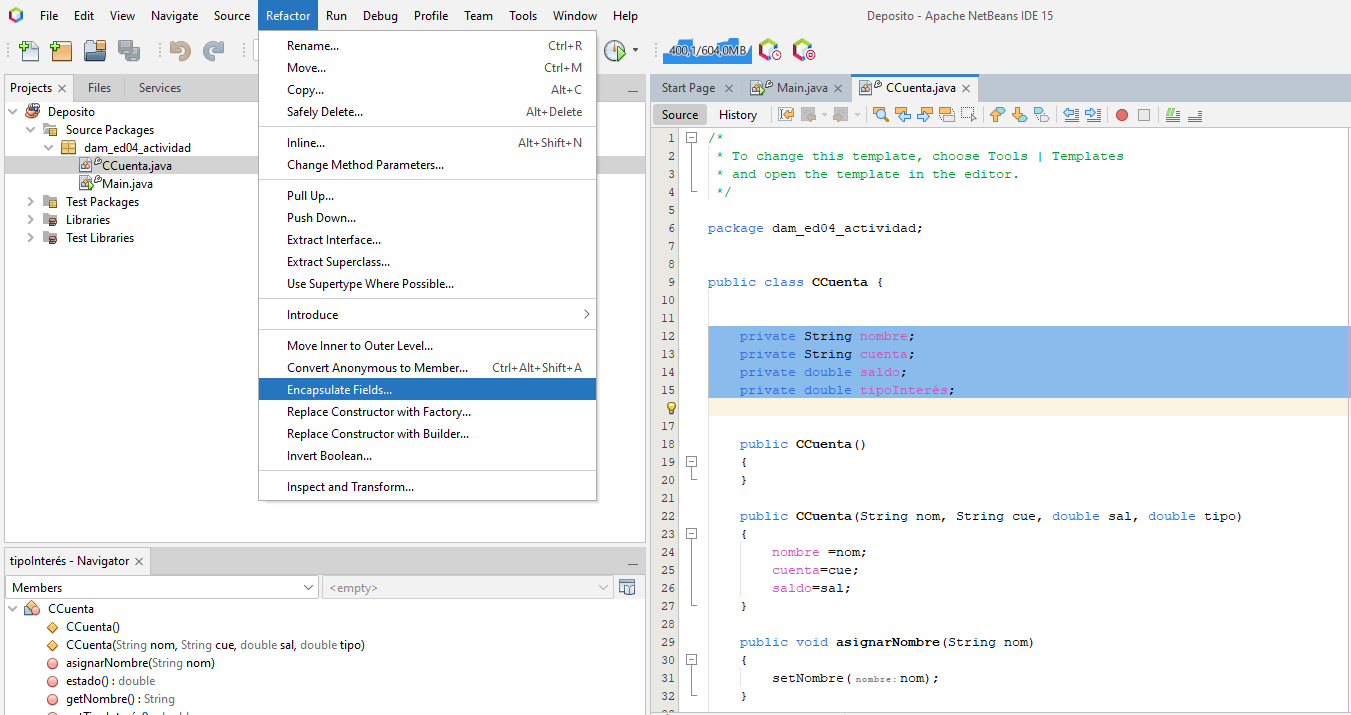


Ilustración . Encapsular atributos (I)

Nos aparecerá la siguiente pantalla donde únicamente clicaremos sobre Refactor.

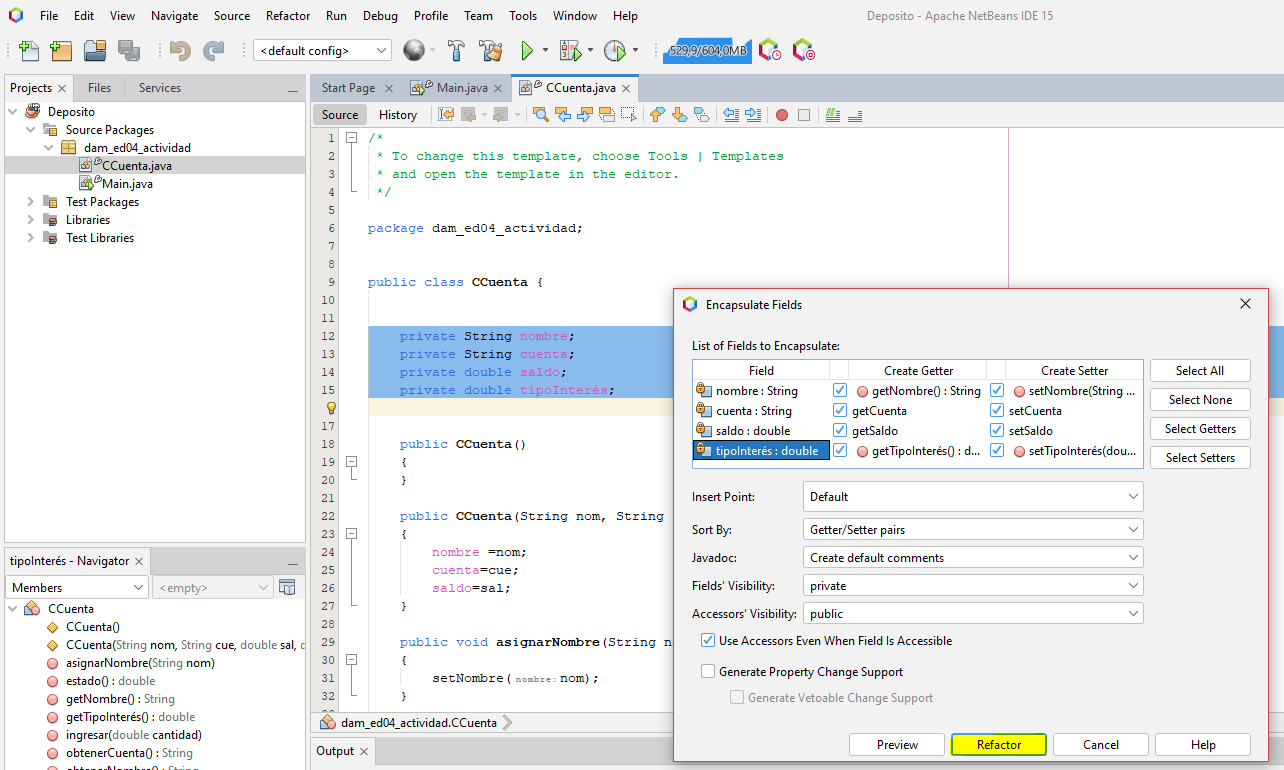


Ilustración . Encapsular atributos (II)

Con esto podemos ver el siguiente resultado:

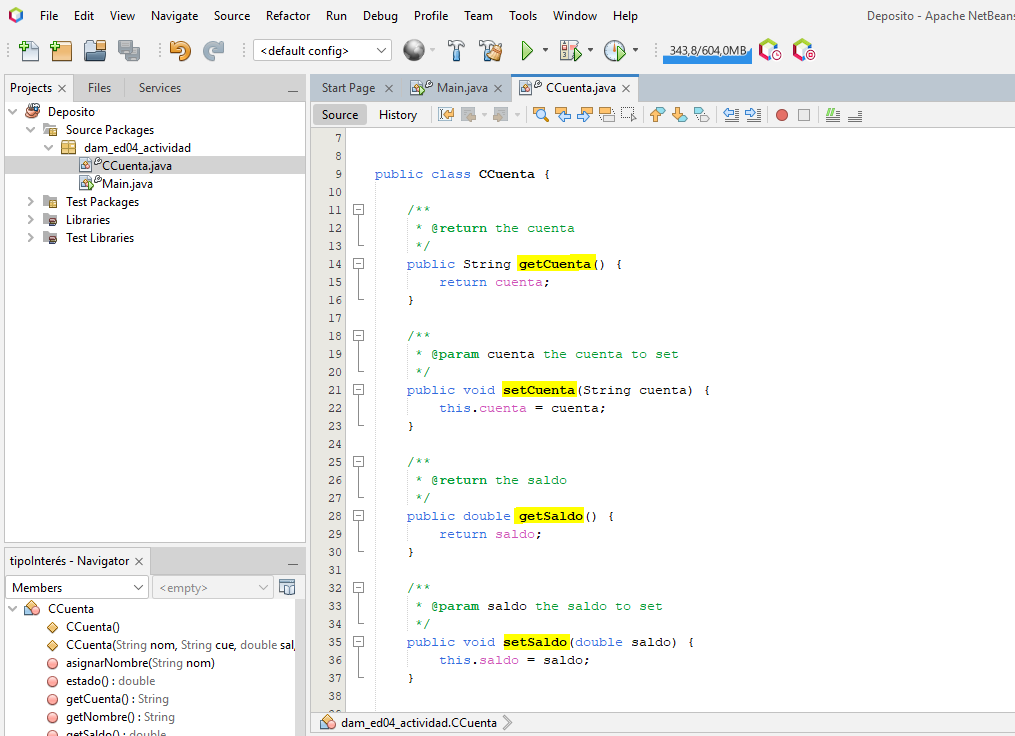


Ilustración . Resultado de encapsular atributos

**Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.**

Para añadir un nuevo parámetro, seleccionamos el método y en el menú Refactor, clicamos sobre Change Method Parameters…

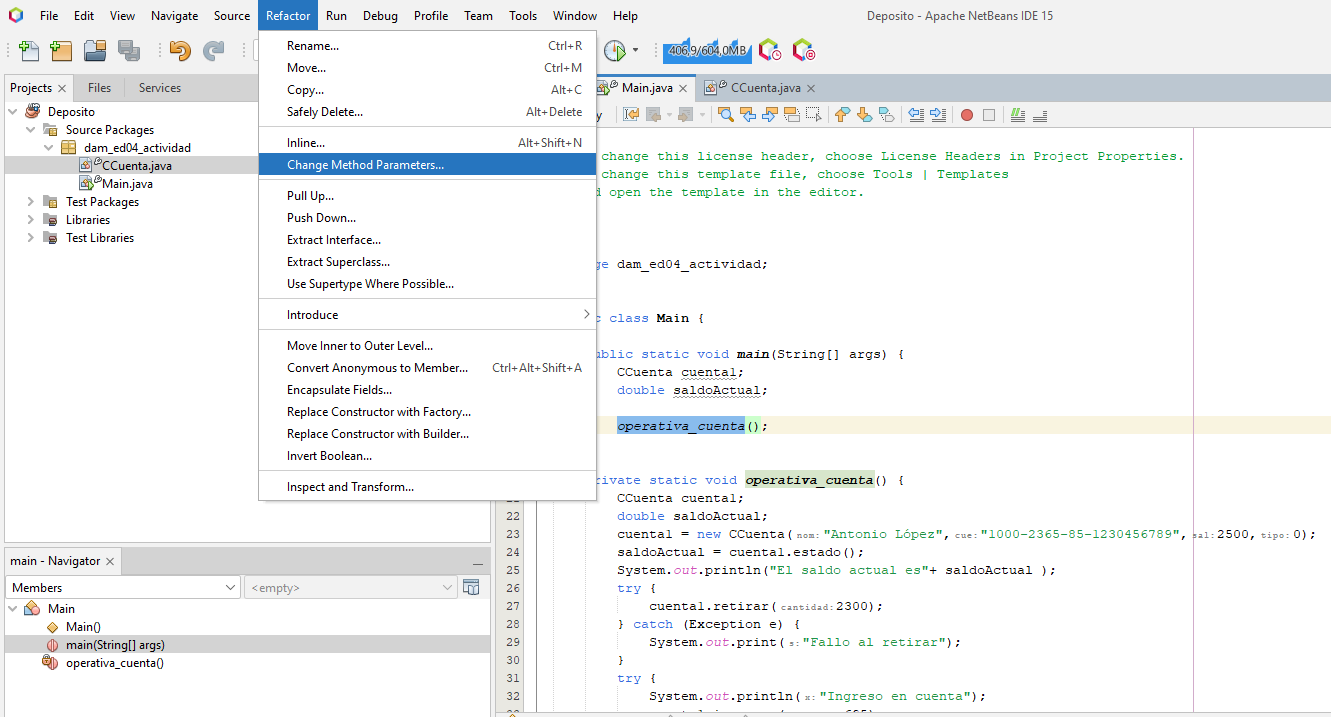


Ilustración . Añadir un nuevo parámetro al método (I)

A continuación, creamos un parámetro a traves de add con name “cantidad” y que su type sea “float”.

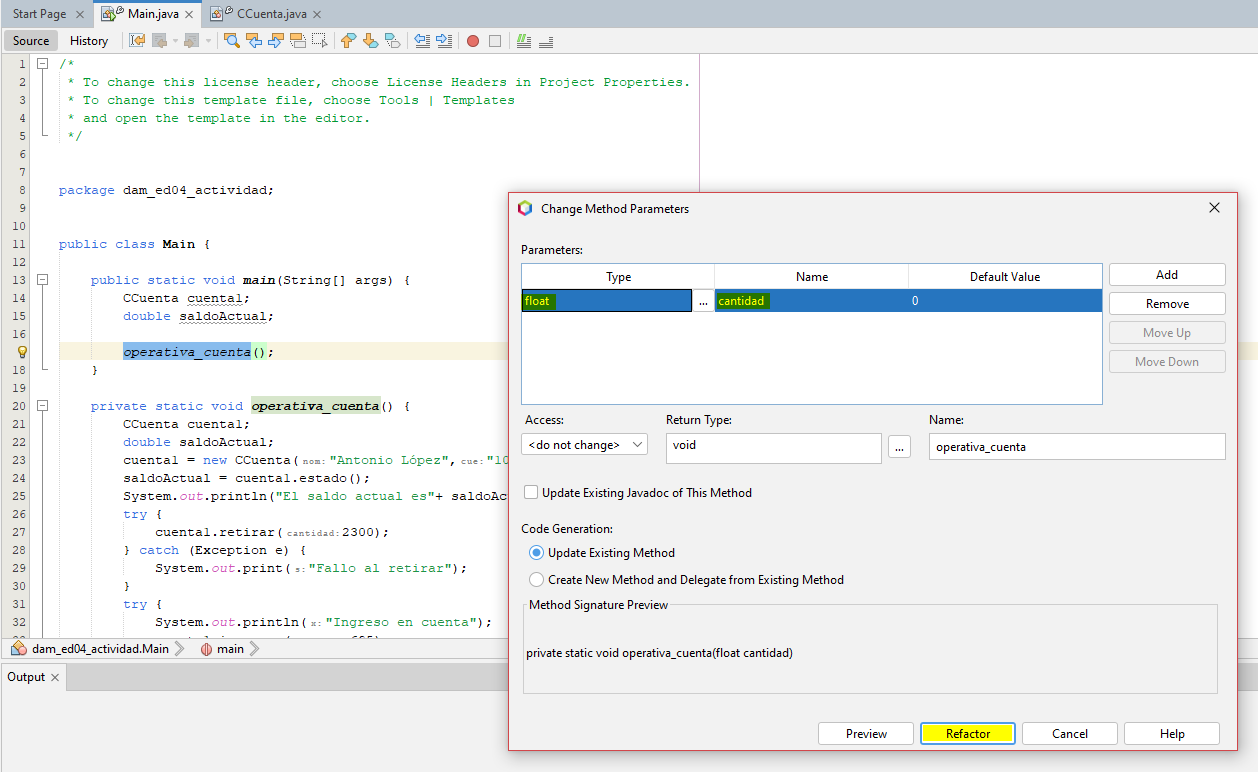


Ilustración . Añadir un nuevo parámetro al método (II)

Con esto podemos ver el siguiente resultado:

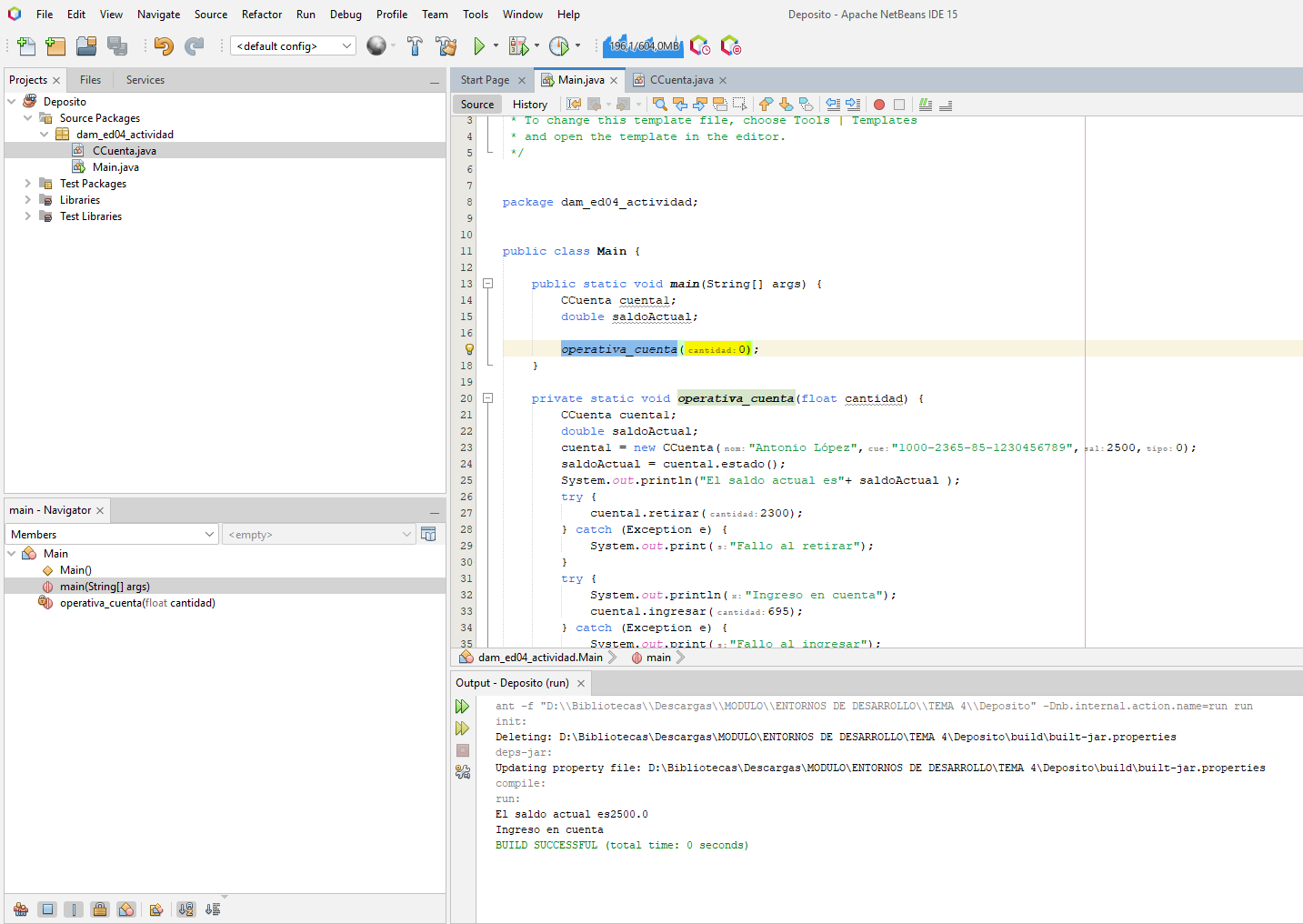


Ilustración . Resultado de añadir un nuevo parámetro al método

# **GIT**

**Configurar GIT para el proyecto. Crear un repositorio público en GitHub.**

Accedemos a la página web <https://github.com/> y nos registramos. Una vez creada la cuenta podemos ver nuestro espacio donde ya estamos logeados con criscorino y a continuación hacemos clic en Create Repository.

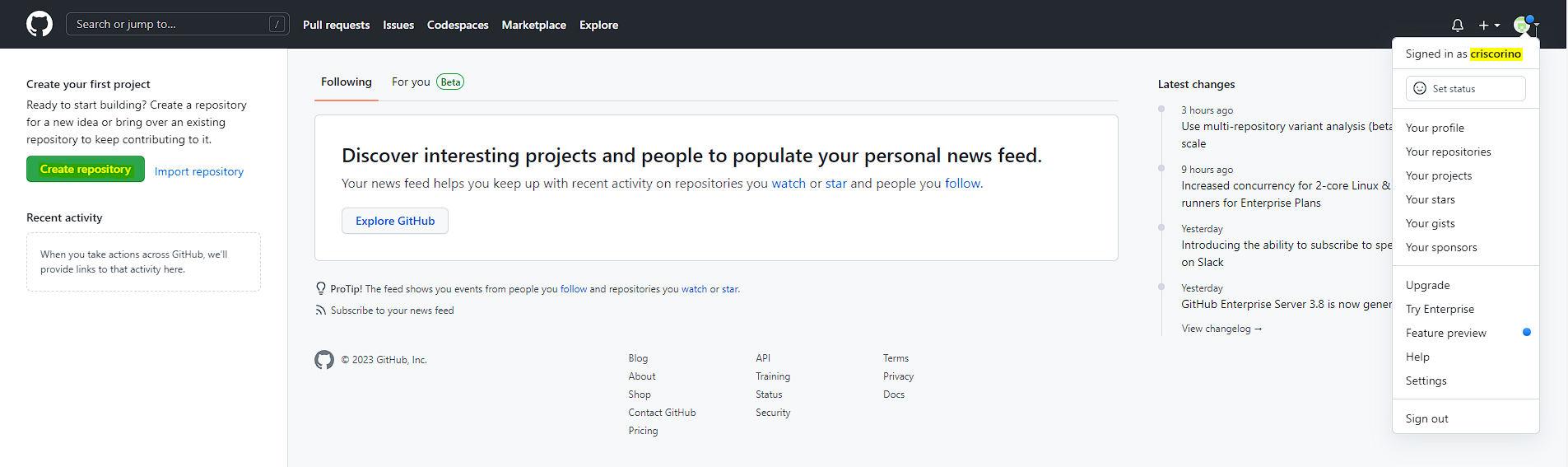


Ilustración . Creamos cuenta en Github

Creamos un nuevo repositorio con el nombre que queramos, en este caso ED04 haciendo referencia a la actividad y la dejamos pública.

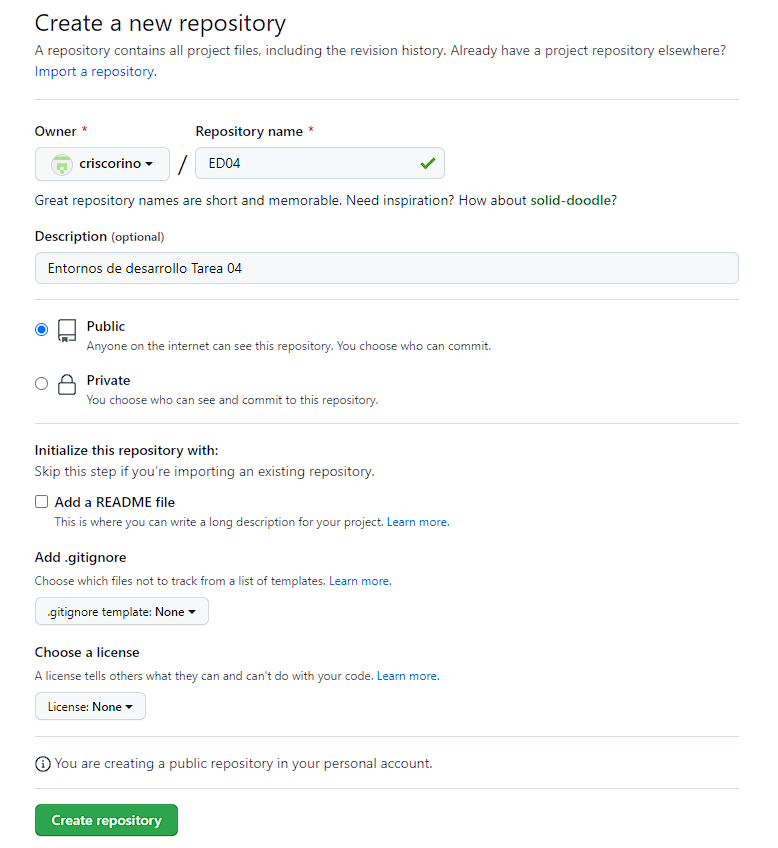


Ilustración . Creamos el repositorio ED04

Una vez creado el repositorio se genera el siguiente código que vamos a copiar en portapapeles para su posterior utilización:

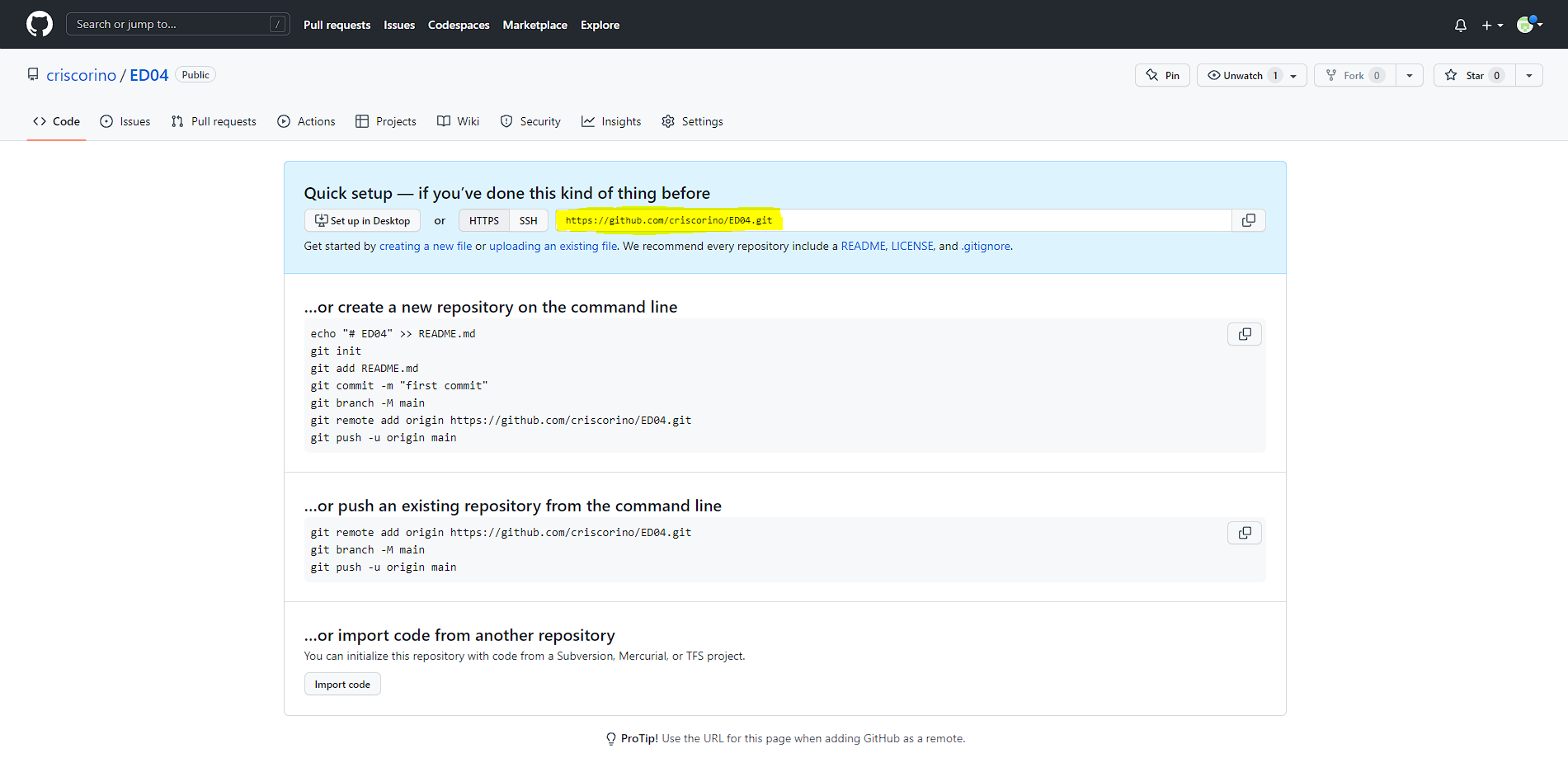


Ilustración . Dirección del repositorio

**Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.**

Una forma que tenemos para poder realizar un commit es descargar la aplicación de escritorio de GitHub a través de <https://desktop.github.com/>:

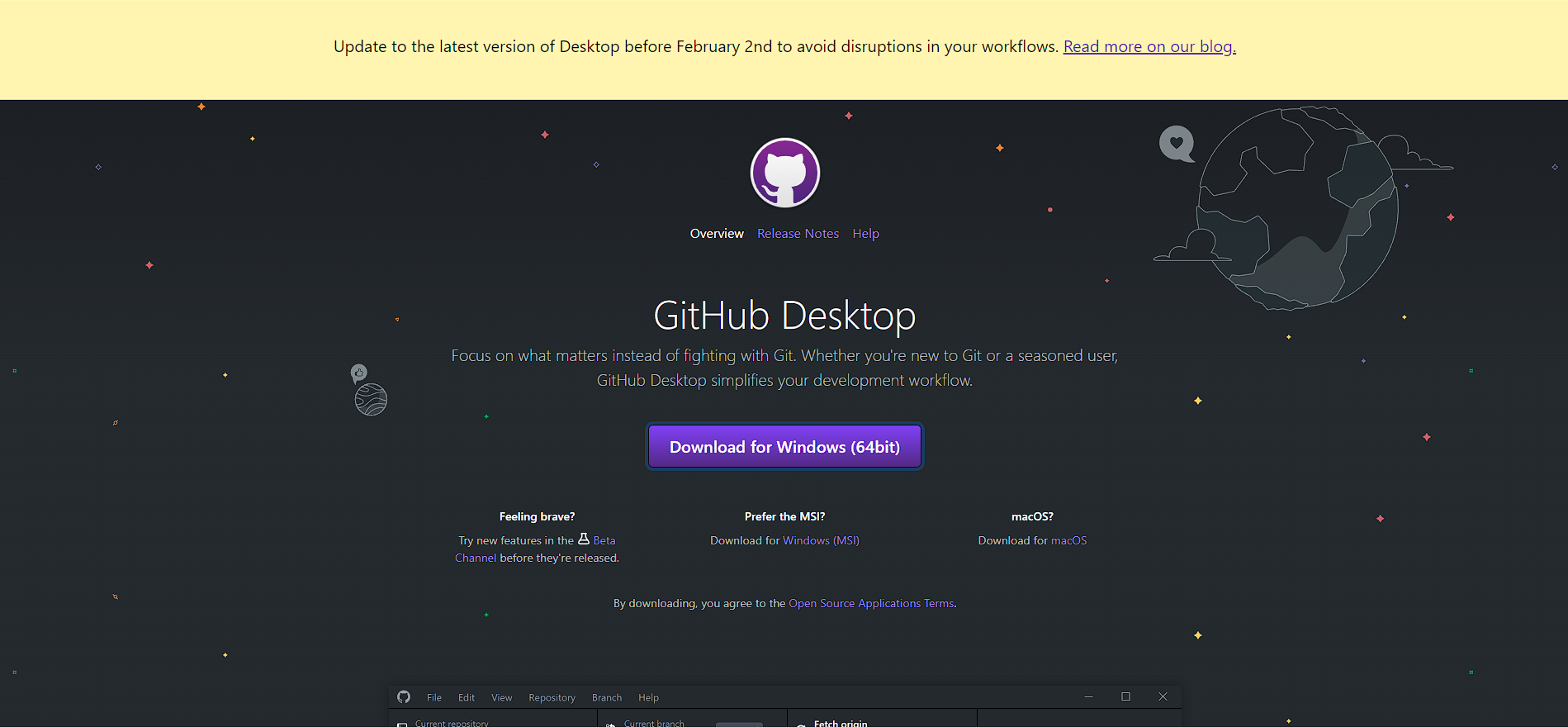


Ilustración . Descarga de la aplicación Github Desktop

Una vez iniciada la aplicación clicamos sobre “Create a tutorial repository…” y seleccionamos el repositorio que creamos anteriormente en la web:

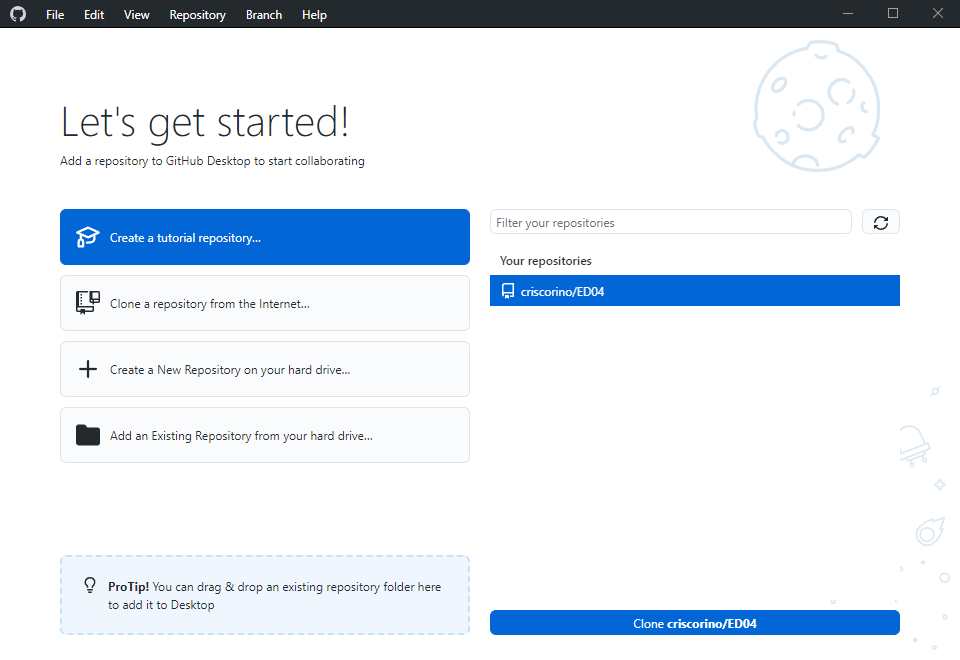


Ilustración . Log y clonación del repositorio

A continuación, tras vincular la aplicación de escritorio con GitHub utilizamos clonar el repositorio mediante URL, pegamos la URL e indicamos la carpeta local de los archivos.

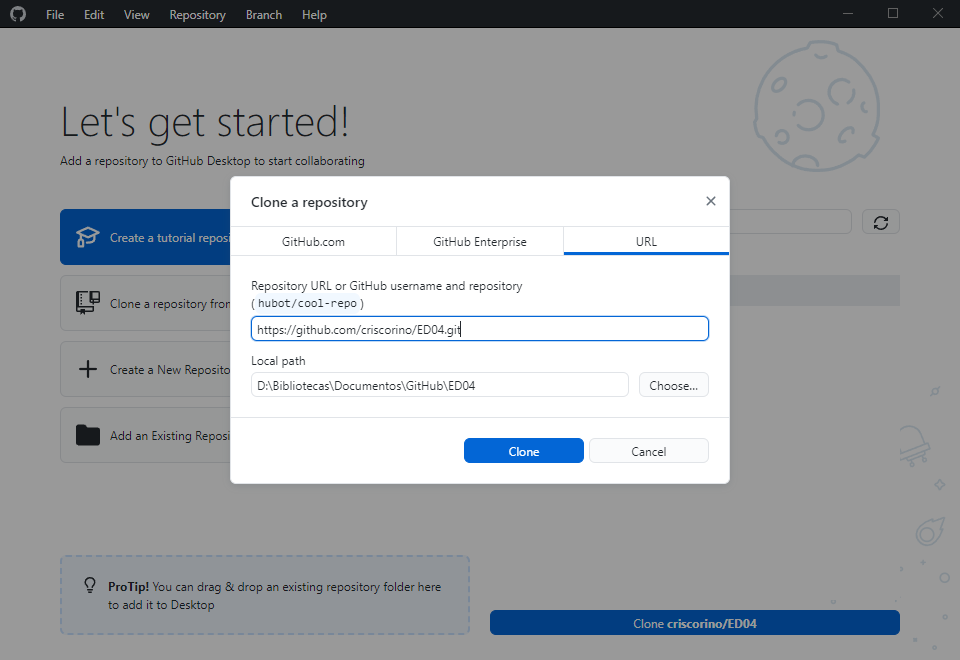


Ilustración . Vinculación con el repositorio

Para realizar un Commit indicaremos en Summary un pequeño resumen, si queremos también se puede añadir una descripción y clicamos sobre “Commit to main”.

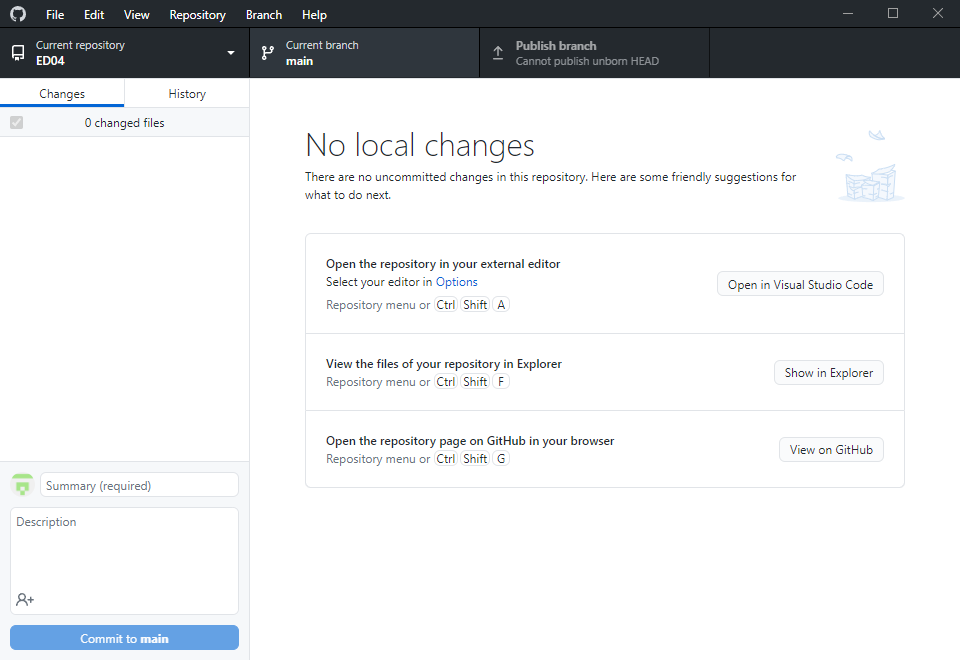


Ilustración . Primer commit

Arrastramos los archivos utilizados para la practica a la carpeta local y a continuación lo desde la aplicación realizamos el “commit to main”.

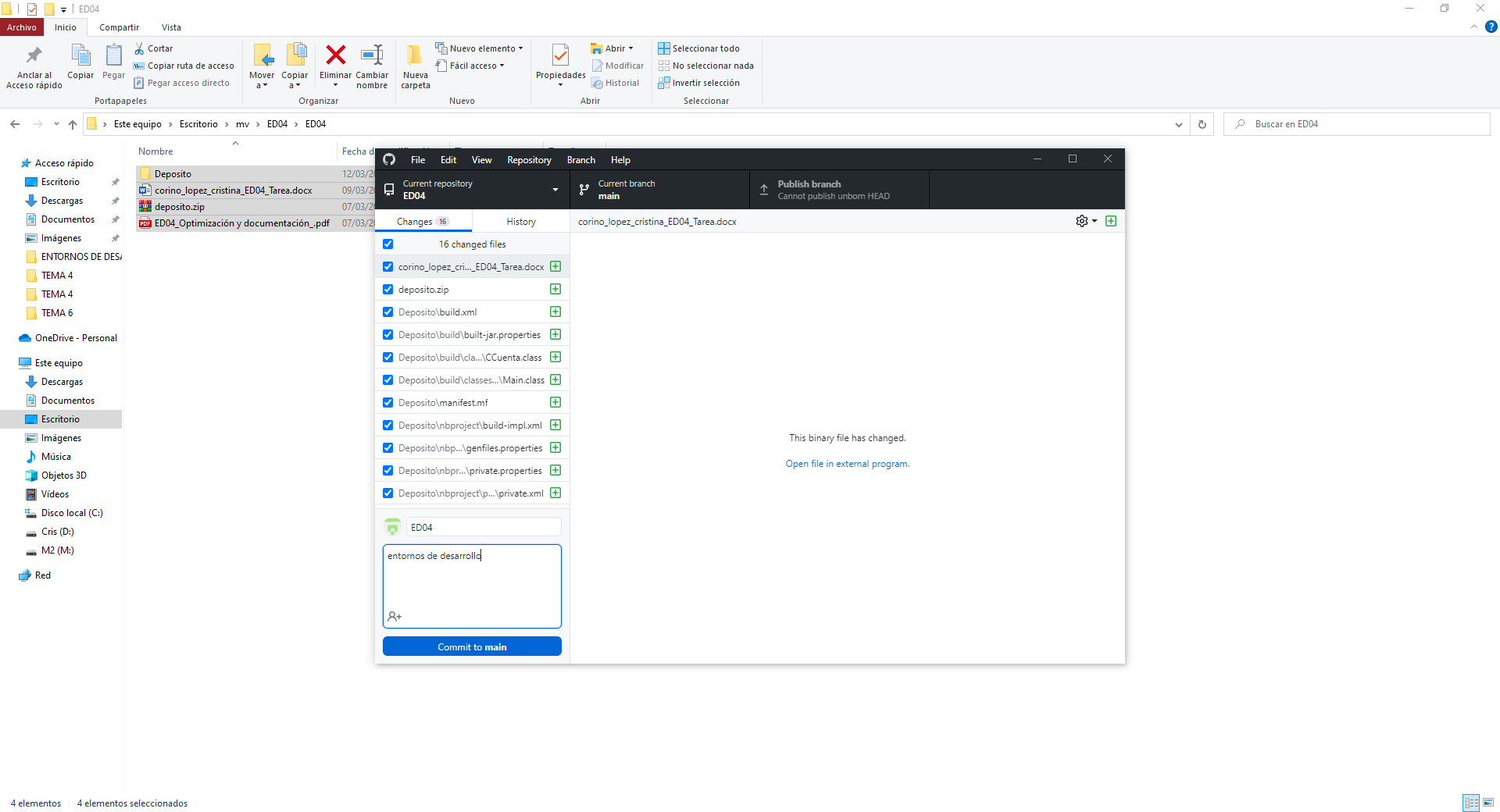


Ilustración . Realziamos commit

Por un lado, una vez subidos los archivos la aplicación se ve de la siguiente forma:

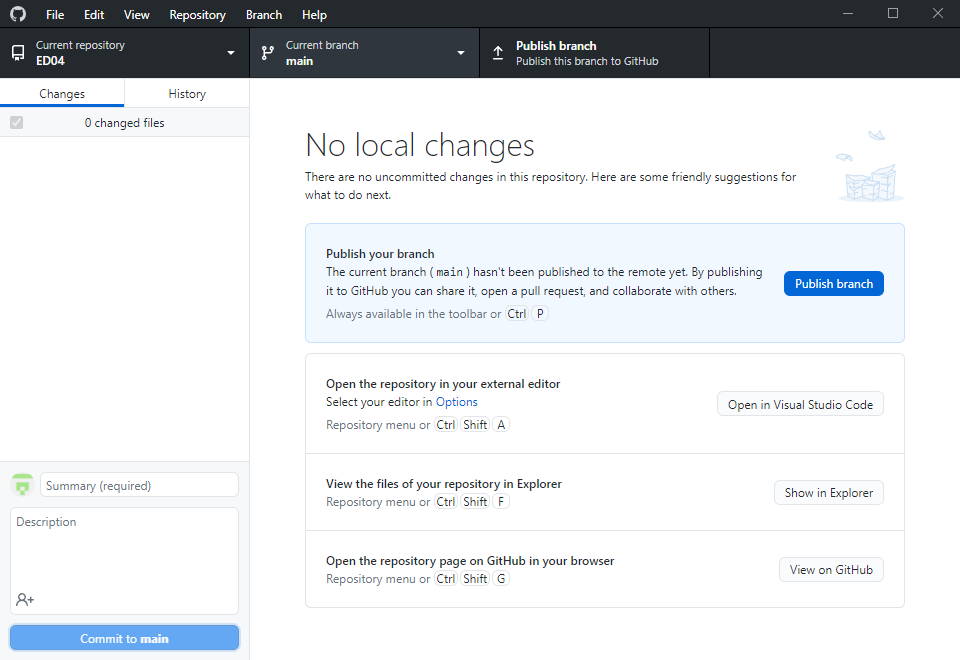


Ilustración . Realizamos el Publish branch

Y por otro lado, si actualizamos la web vemos que los documentos están disponibles en GitHub

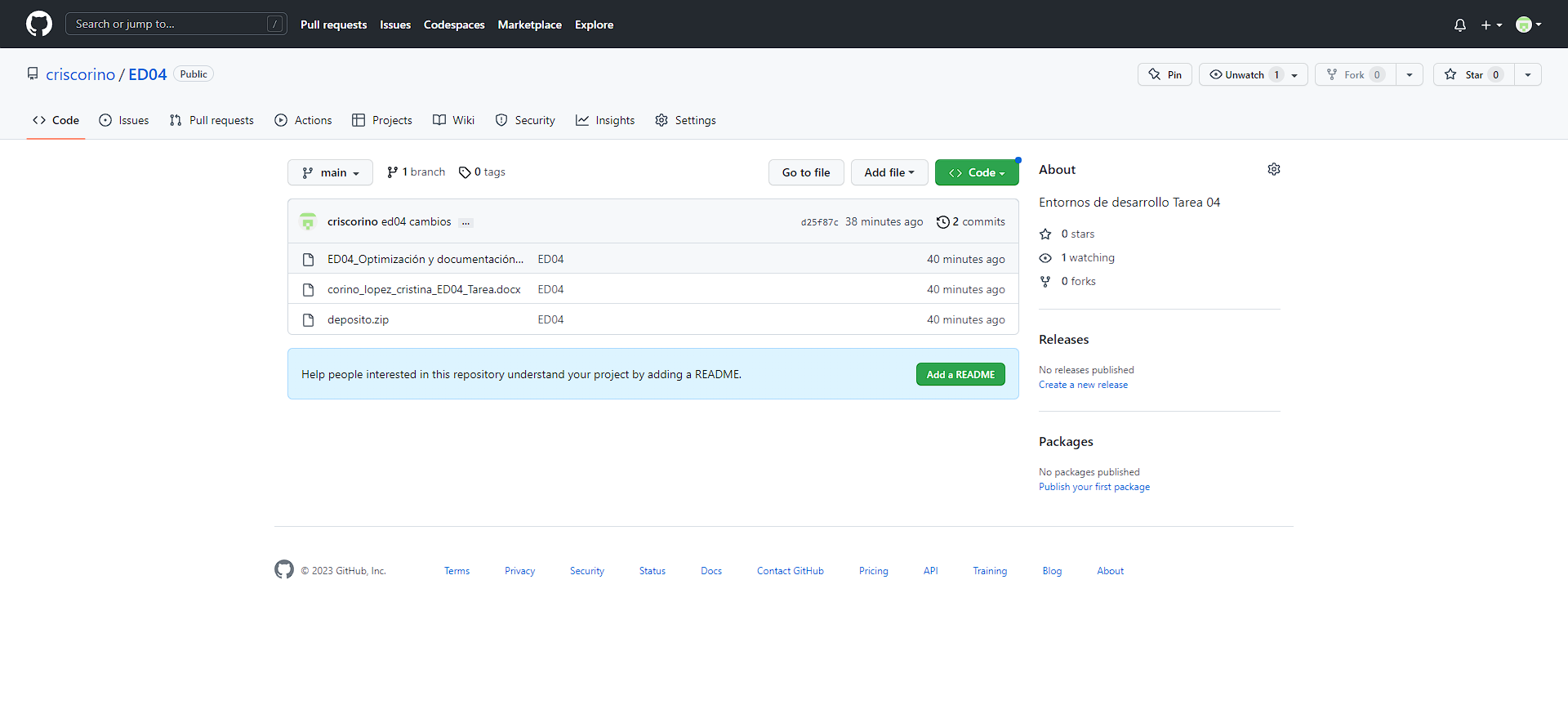


Ilustración . Github online

**Mostrar el historial de versiones para el proyecto mediante un comando desde consola.**

Para mostrar el historial de versiones del proyecto desde la propia aplicación es muy sencillo. Únicamente debemos clicar en el botón “History” para ver las modificaciones realizadas. Para ver más en detalle únicamente debemos clicar sobre la modificación y ver los datos:

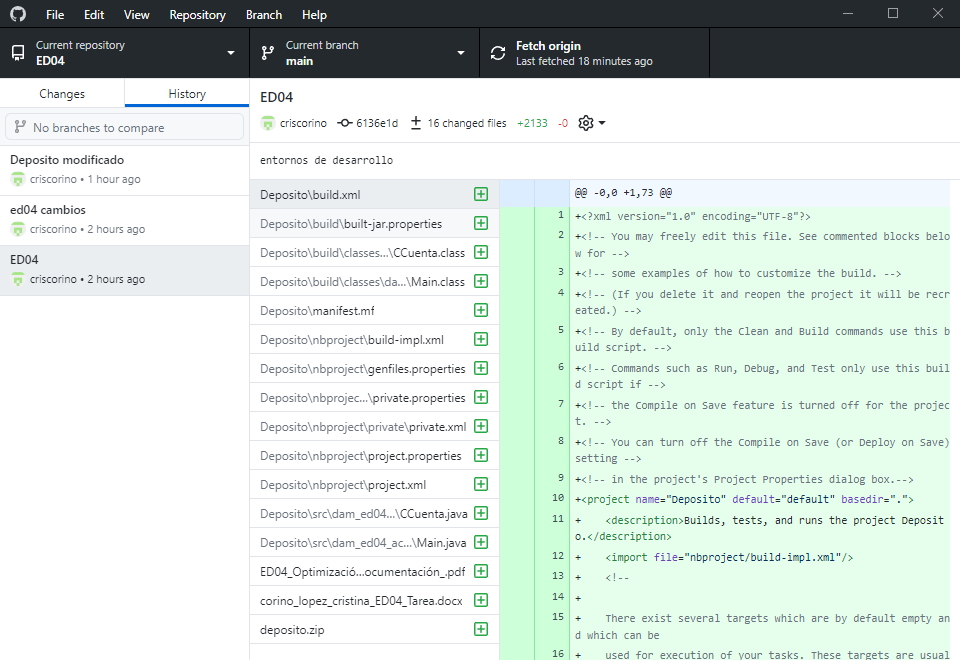


Ilustración . Historial de versiones desde la aplicación de escritorio

Para poder verlo desde la consola de comandos es algo más complicado. Deberemos ir a la cmd e indicar el siguiente código para instalar el programa Git CMD “winget install --id Git.Git -e --source winget”:

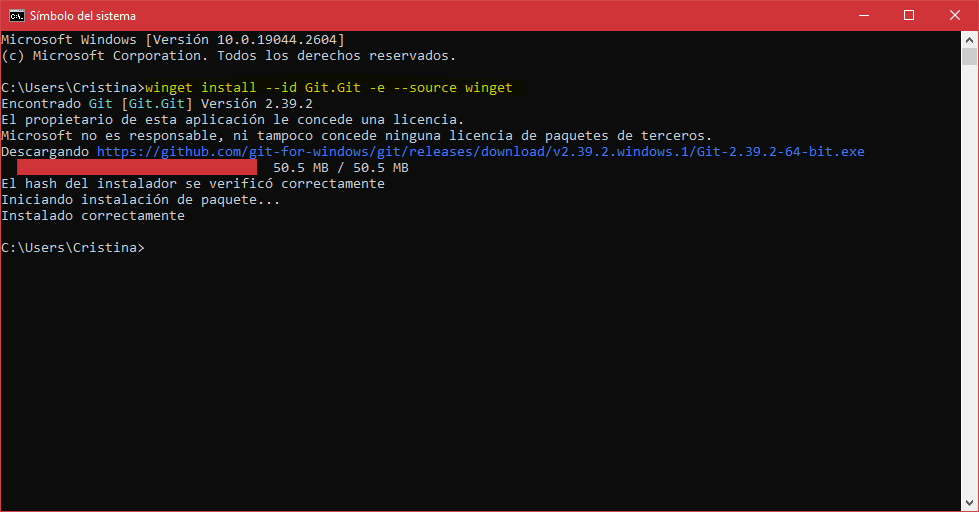


Ilustración . Instalamos Git por comando

Abrimos la CMD y realizamos la configuración global con los siguientes comandos.

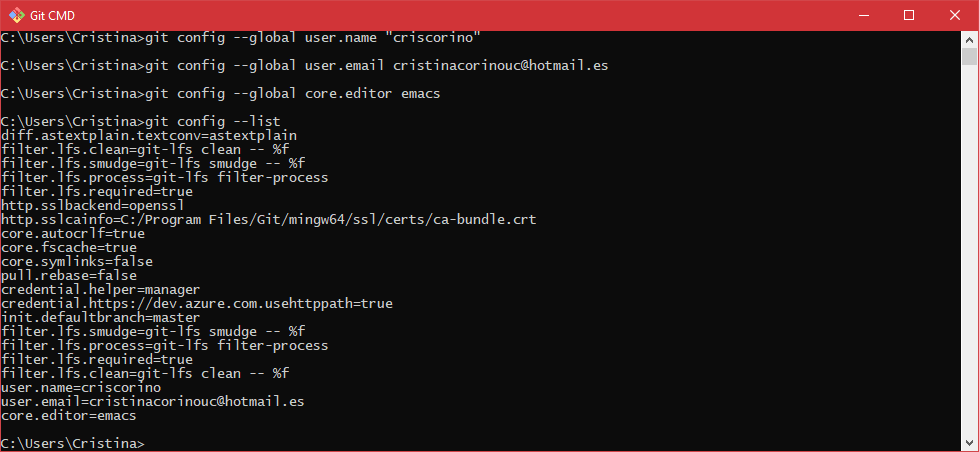


Ilustración . Configuración global

Por último, nos podemos sobre la carpeta local contenedora y utilizamos el comando “git log”, podremos ver en pantalla las tres modificaciones que habíamos visto también en la aplicación:

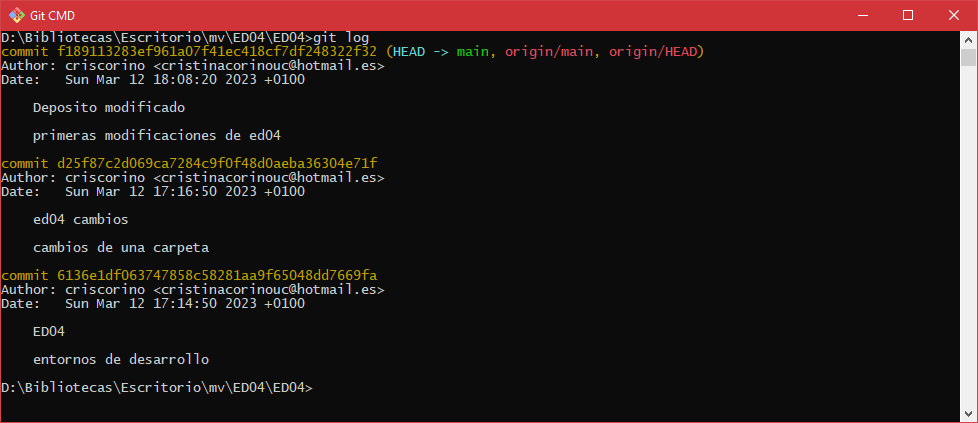


Ilustración . Historial de versiones

# **JAVADOC**

**Insertar comentarios JavaDoc en la clase CCuenta.**

Insertamos comentarios dentro de la clase CCuenta como vemos a continuación:

****

**Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.**

Tras añadir los comentarios generamos la documentación JavaDoc en Run – Generate JavaDoc:

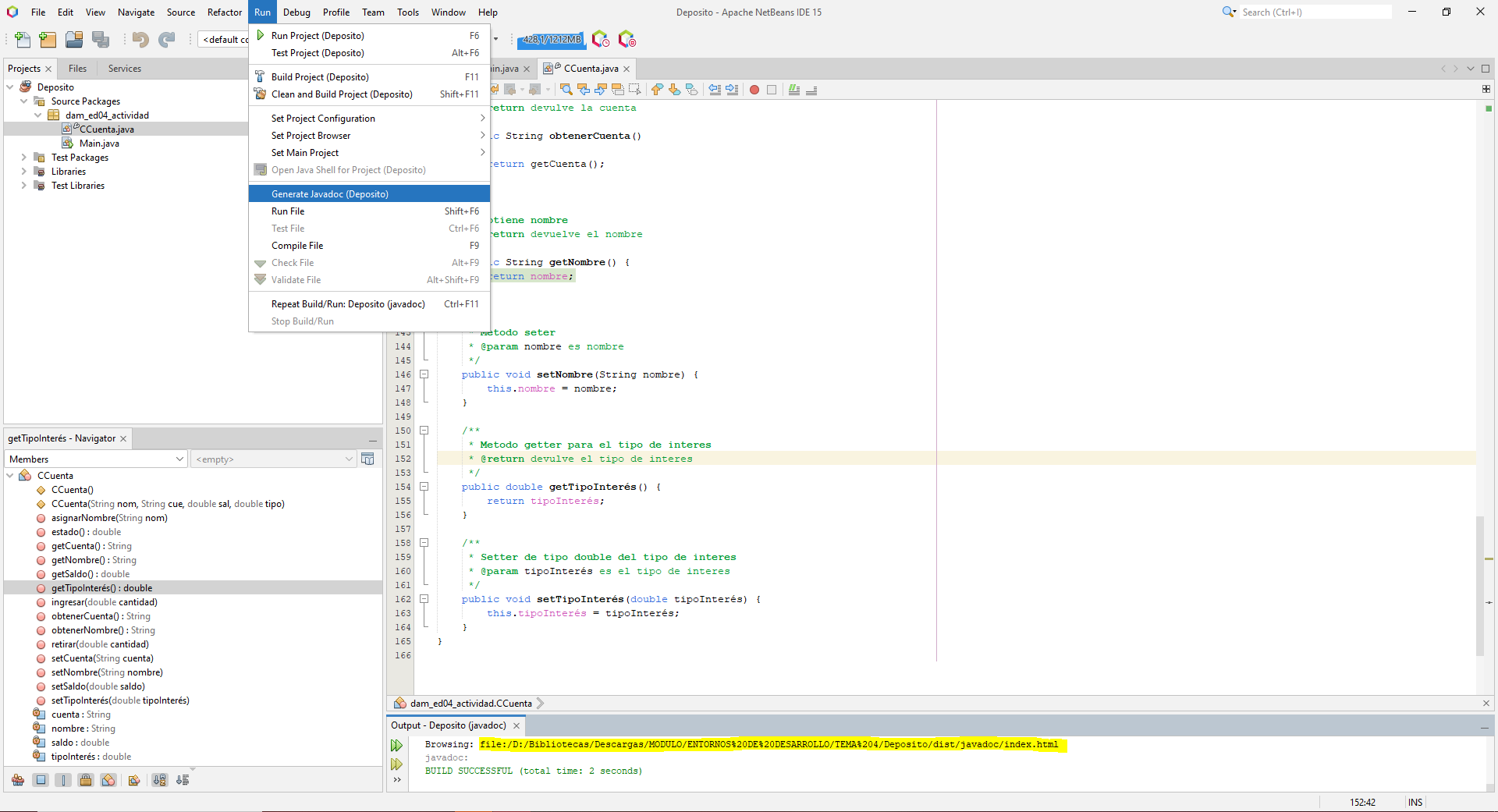


Ilustración . Generar JavaDoc

Se abrirá una nueva ventana del navegador con lo siguiente:

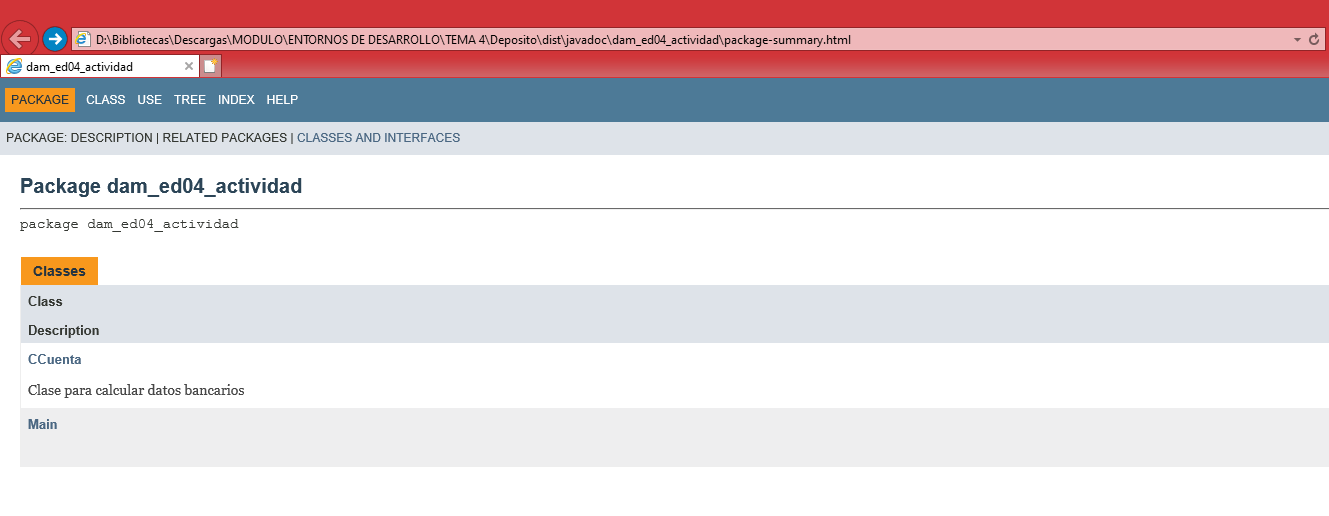


Ilustración . JavaDoc

Si clicamos sobre la clase CCuenta podemos ver los comentarios que fuimos introduciendo:

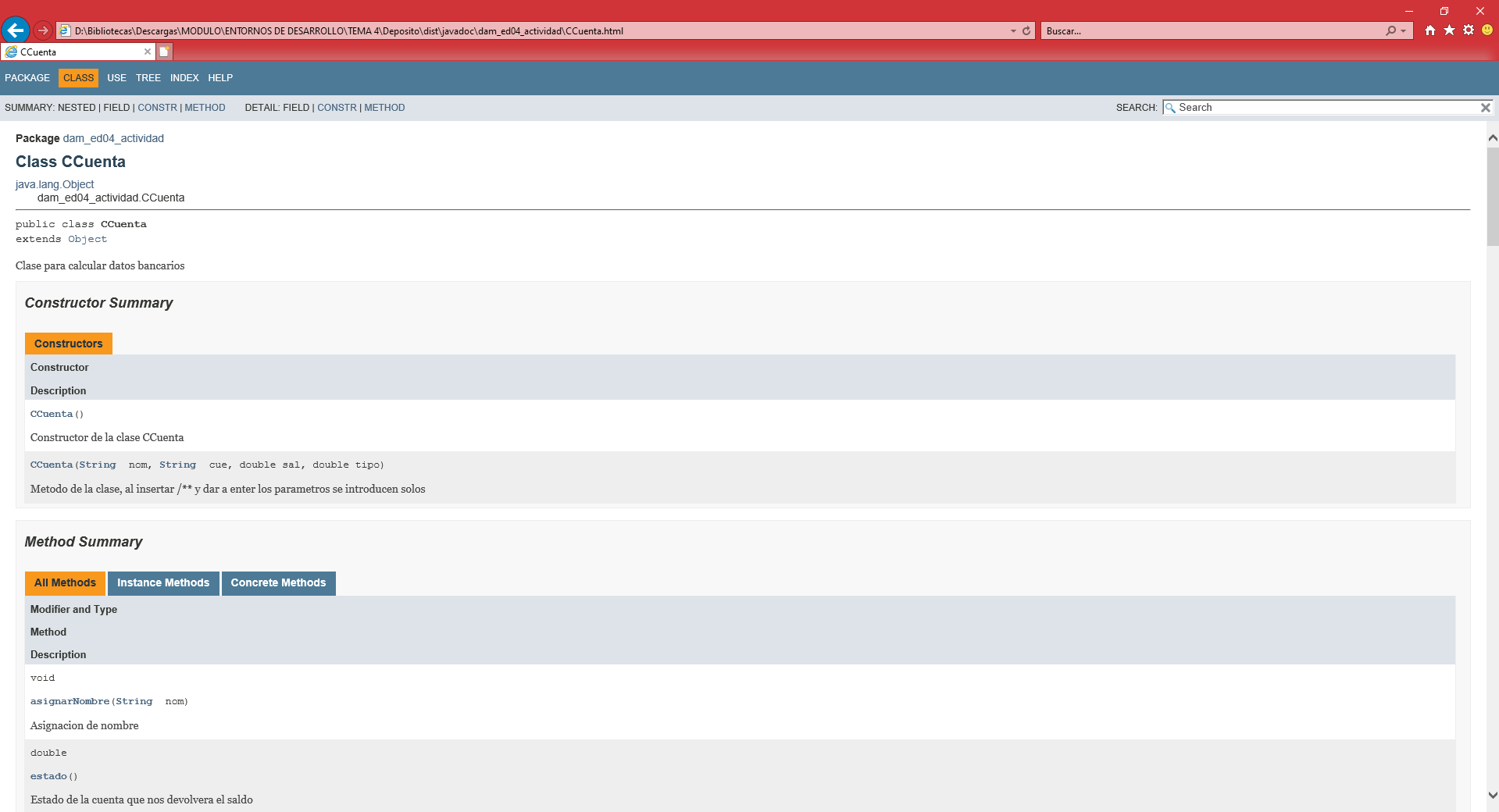


Ilustración . Clase CCuenta en el JavaDoc

Para finalizar la tarea insertamos toda esta información actualizada de nuevo en la carpeta GitHub para que la tarea tenga toda su información accesible desde allí. En esta ocasión para ver otra opción vamos a utilizar el commit desde la GIT CMD con los siguientes comandos:

* **git commit -m “Javadoc”**, de esta forma generamos el commit con el mensaje “Javadoc”.
* **git push origin main**, de esta forma realizamos el push para que los cambios que se encuentran en local se suban a la red.

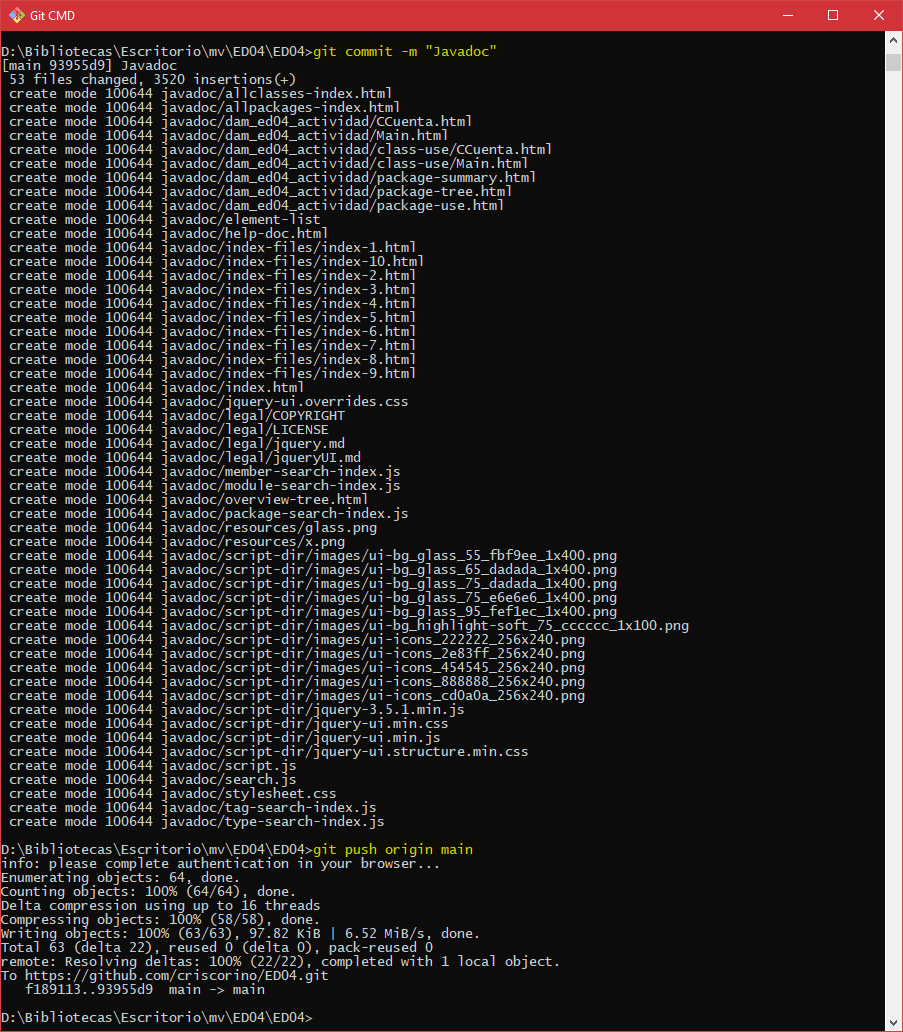


Ilustración . Commit desde la CMD

De esta forma podemos ver así la información actualizada en la web:

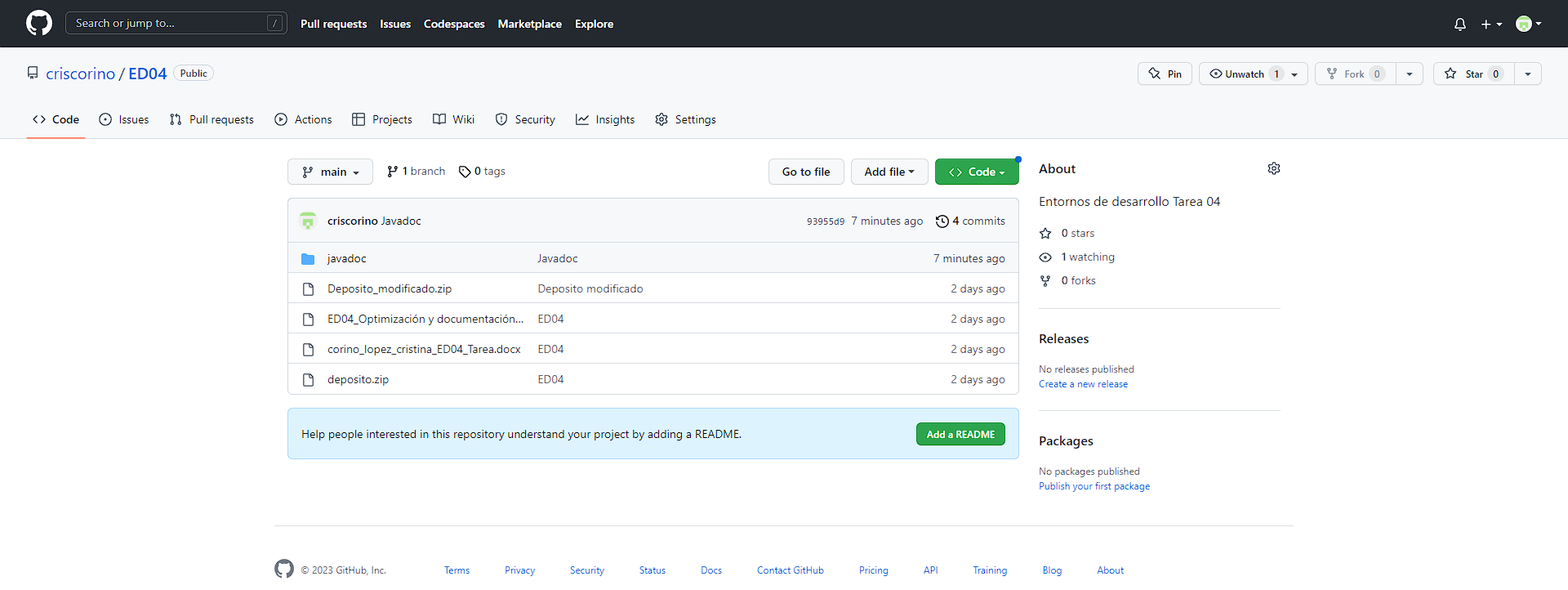


Ilustración . Commit disponible online