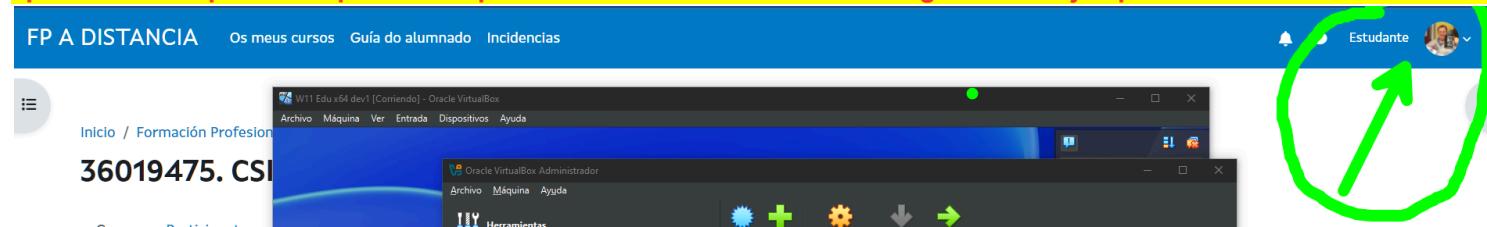


**AVISO:** Las tareas son trabajo individual y no deben ser compartidas con otros compañer@s. Si se detecta o se sospecha que una tarea es copia total o parcial de otra, serán anuladas las 2 tareas, independientemente de quién realizara la versión original. La nota de ambas tareas será de 0 puntos

En los apartados en los que es necesario entregar las capturas de pantalla, éstas deben tener como fondo de pantalla la plataforma de fpdistancia con tu usuario mostrando claramente la foto de tu perfil. Aquellos apartados/subapartados que no cumplan esta condición no serán corregidos. Por ejemplo:



## Actividad 1: EN NETBEANS

En el proyecto **Java "Deposito"** (ver por favor .zip anexado), hay definida una Clase llamada **CCuenta**, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con una Clase **Main**, donde se hace uso de la clase descrita.

Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

**DOCUMENTA TODO TU TRABAJO CON LAS EXPLICACIONES Y CAPTURAS DE PANTALLA QUE DEMUESTREN QUE HAS REALIZADO CADA TAREA CORRECTAMENTE.**

### 1. REFACTORIZACIÓN

1. Las clases deberán formar parte del paquete cuentas.
2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".
3. Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.
4. Encapsular los atributos de la clase CCuenta.
5. Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.

### 2. GIT

1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.
2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.
3. Mostrar el historial de versiones para el proyecto mediante un comando desde consola.

### 3. JAVADOC

1. Insertar comentarios JavaDoc en la clase CCuenta.
2. Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.

## Actividad 2: EN ECLIPSE

En el proyecto Java "**Deposito**" (ver por favor .zip anexado), hay definida una Clase llamada *CCuenta*, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con una Clase *Main*, donde se hace uso de la clase descrita.

Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

### 1. REFACTORIZACIÓN

1. Las clases deberán formar parte del paquete cuentas.
2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".
3. Introducir el método *operativa\_cuenta*, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.
4. Encapsular los atributos de la clase *CCuenta*.
5. Añadir un nuevo parámetro al método *operativa\_cuenta*, de nombre cantidad y de tipo float.

### 2. GIT

1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.
2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.
3. Mostrar el historial de versiones para el proyecto mediante un comando desde consola.

### 3. JAVADOC

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

Los **criterios de puntuación**, para cada apartado (5 puntos por apartado repartidos de forma proporcional) serán los siguientes:

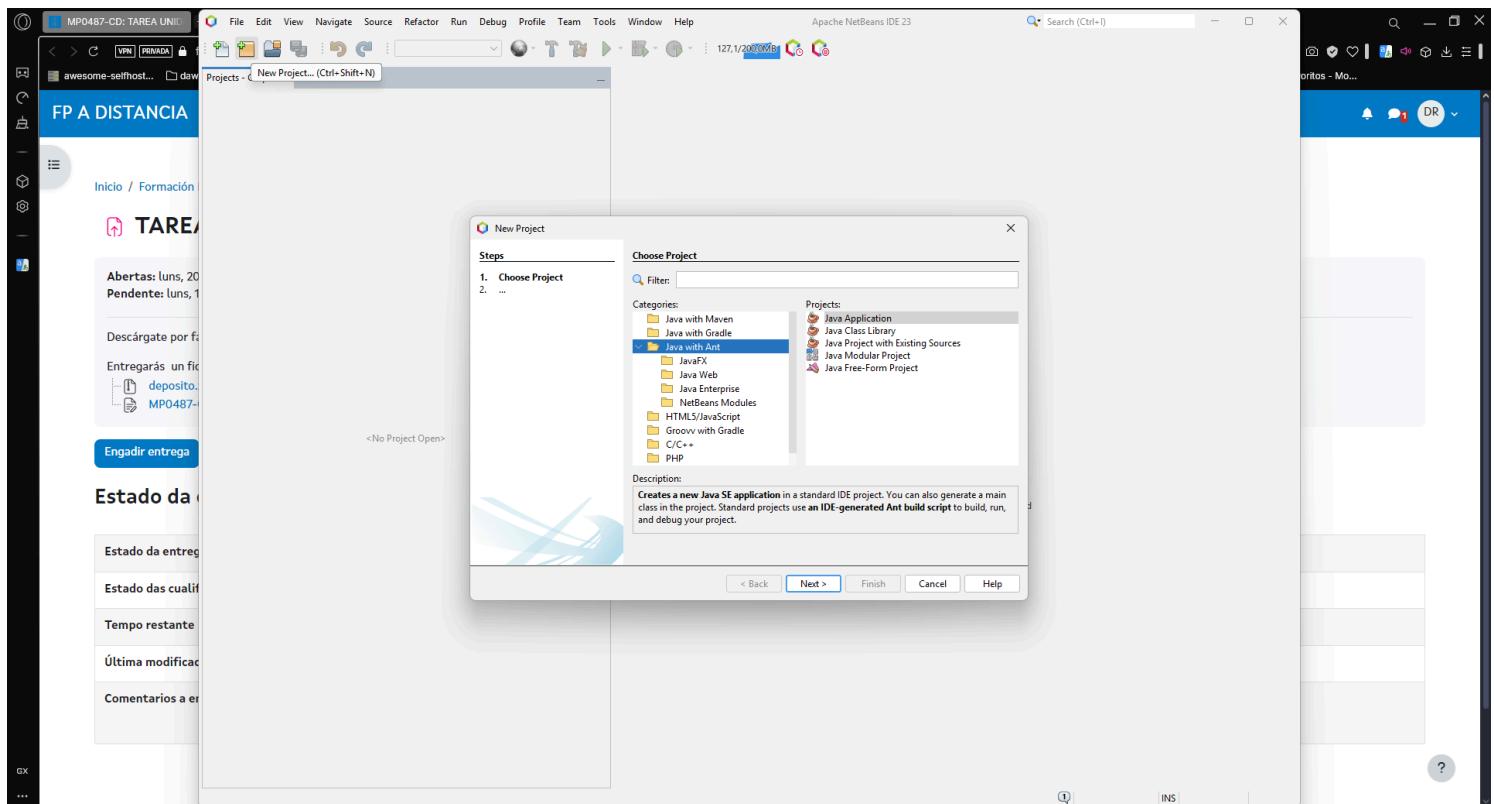
1. Cambia el nombre de la variable "miCuenta" por "cuenta1". = 1 punto.
2. Introduce el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1. = 1 punto.
3. Encapsula los cuatro atributos de la clase CCuenta. = 1 punto.
4. Añadir un nuevo parámetro al 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.

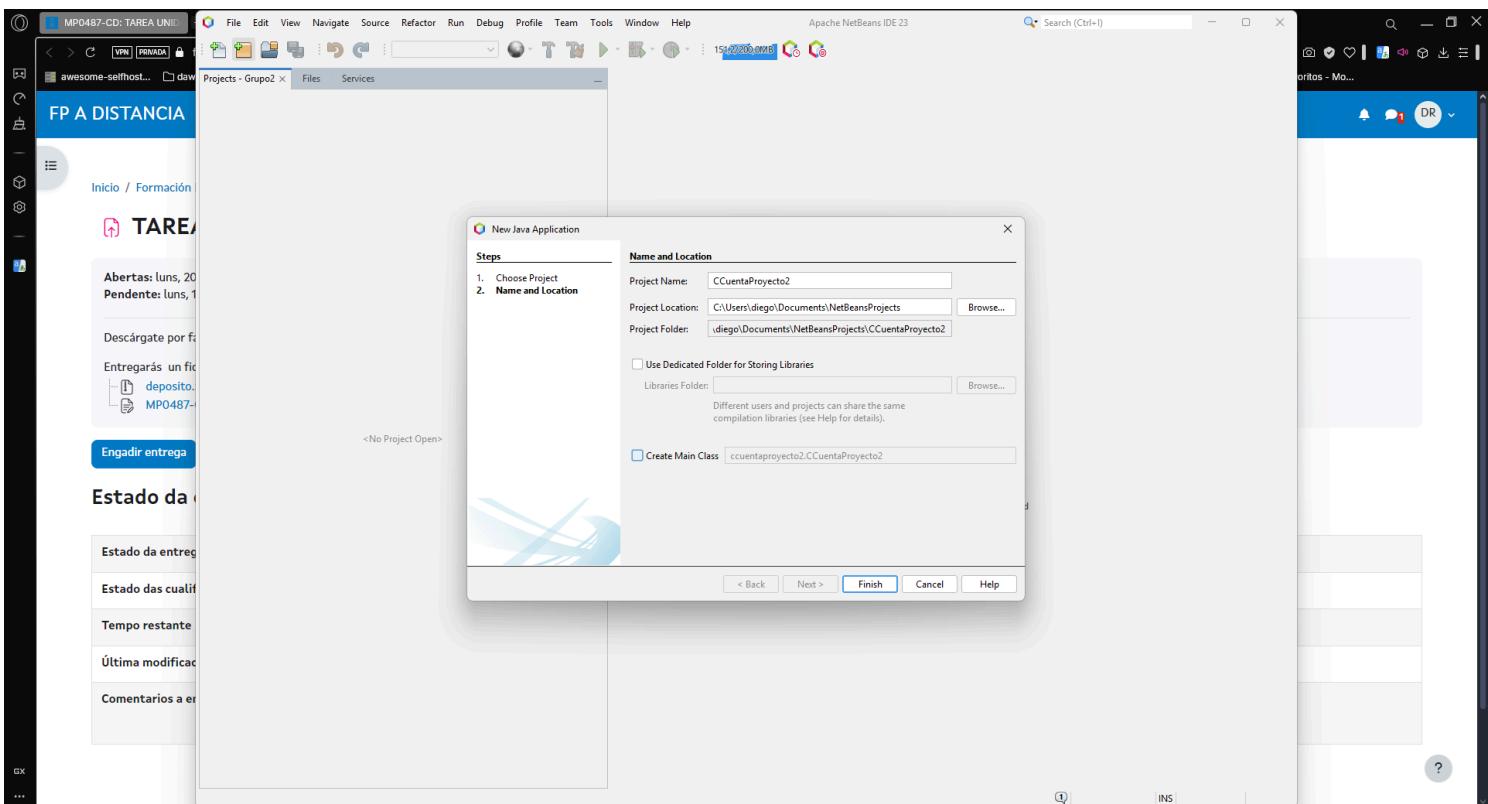
## NETBEANS

### 1. Refactorización

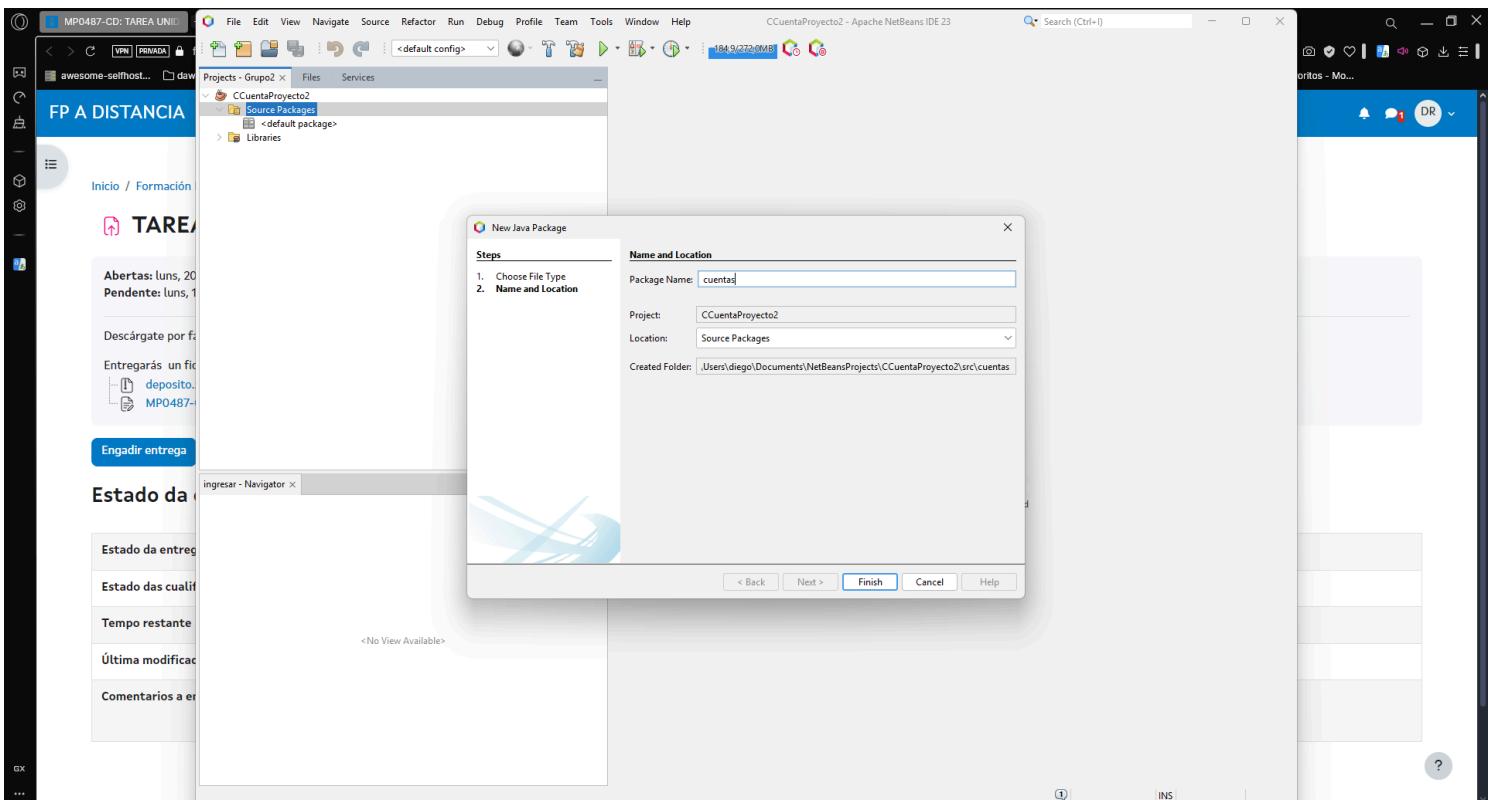
- 1.1. Las clases deberán formar parte del paquete cuentas.
- 1.2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".
- 1.3. Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.
- 1.4. Encapsular los atributos de la clase CCuenta.
- 1.5. Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.

Crear proyecto en Netbeans



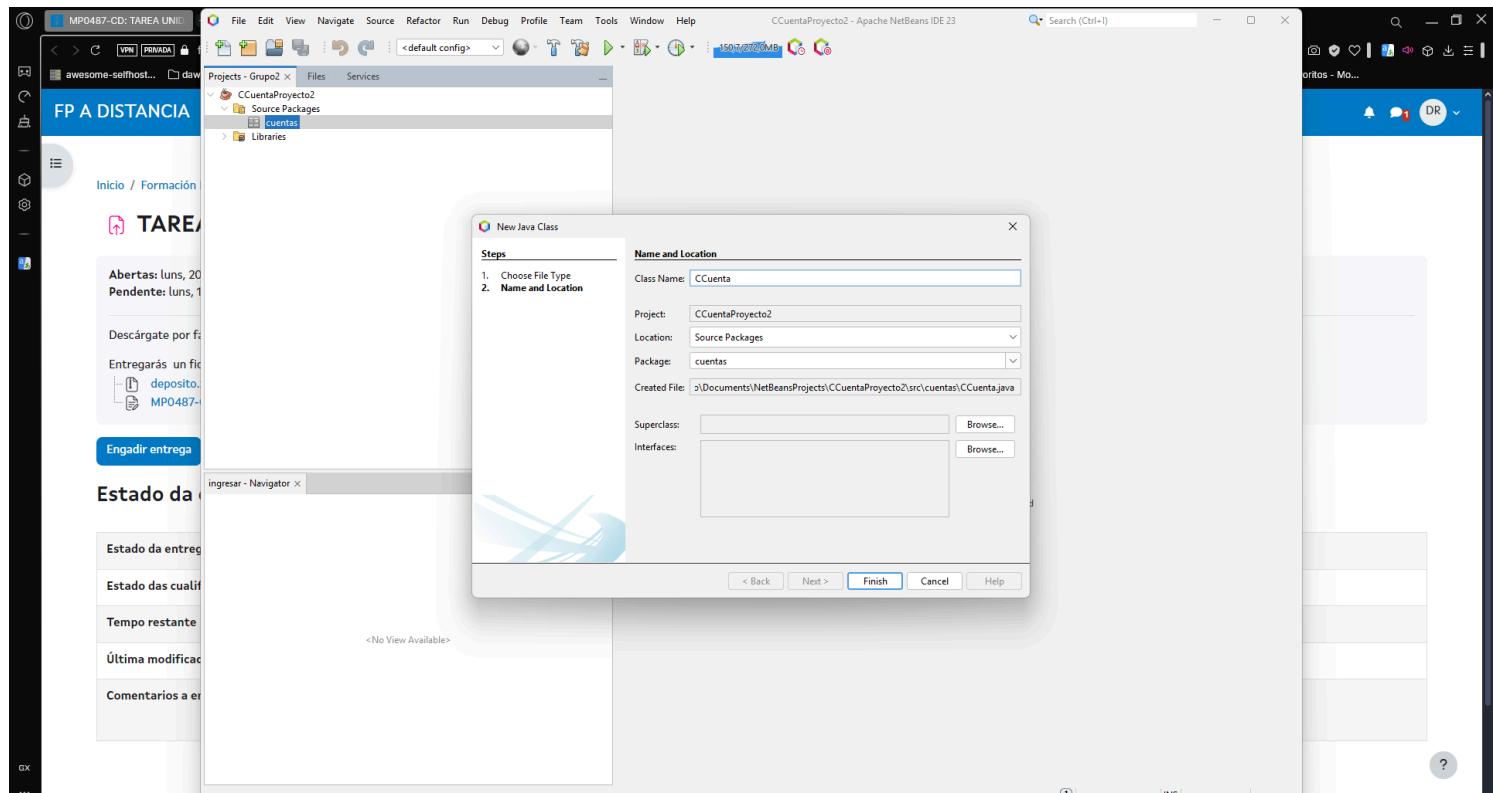


## Crear paquete cuentas

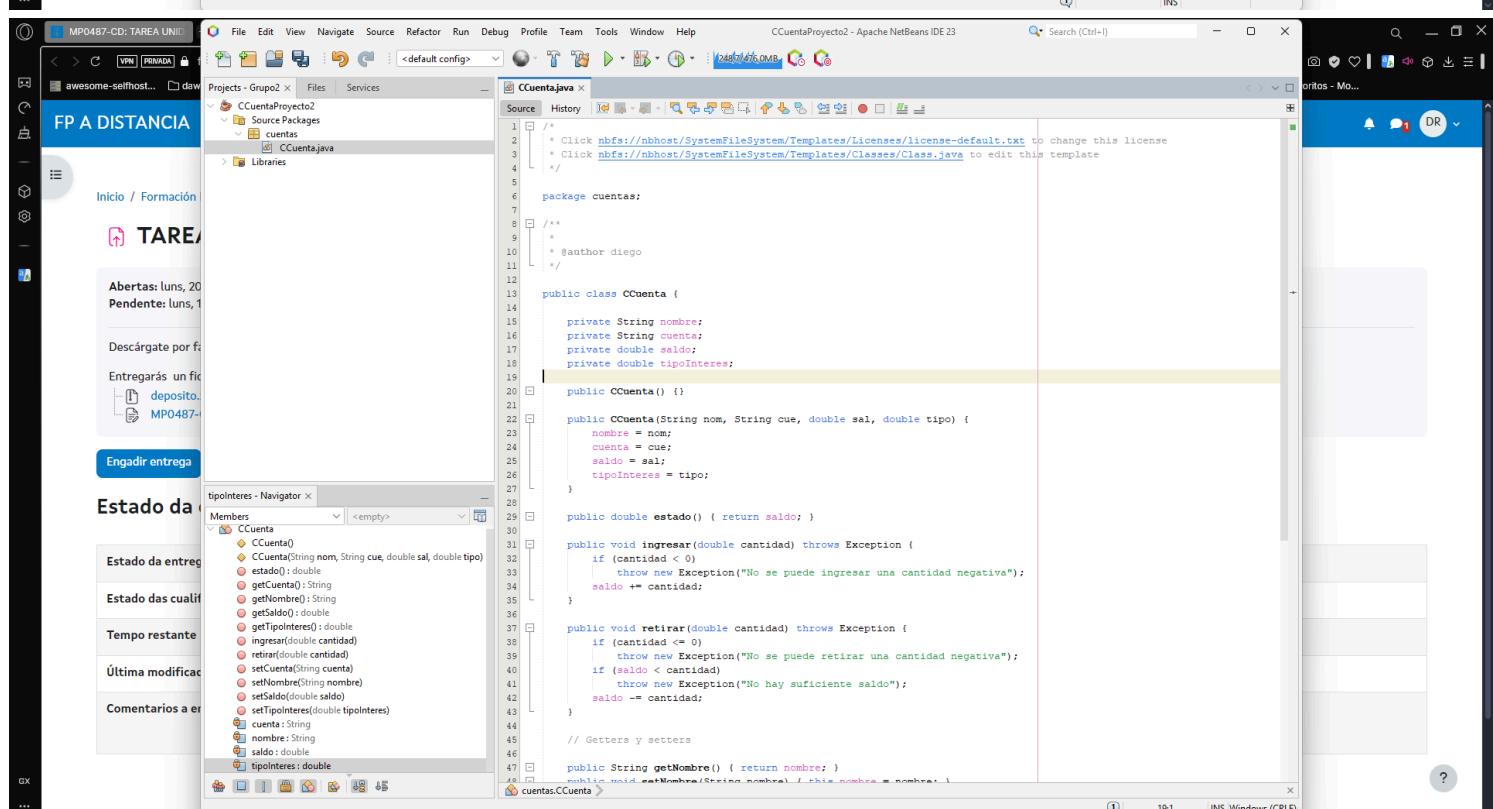


Para cumplir con los requerimientos del enunciado, creamos las clases en el paquete cuentas y realizamos las siguientes modificaciones en el código aportado en el enunciado:

### - CCuenta.java



The screenshot shows the NetBeans IDE interface with a project titled "CuentaProyecto2". A dialog box titled "New Java Class" is open, showing the steps "Choose FileType" and "Name and Location". In the "Name and Location" section, the "Class Name" is set to "CCuenta", "Project" is "CuentaProyecto2", "Location" is "Source Packages", and "Package" is "cuentas". The "Created File" path is displayed as `..\Documents\NetBeansProjects\CuentaProyecto2\src\cuentas\CCuenta.java`. The "Finish" button is highlighted.

The screenshot shows the NetBeans IDE interface with the "CCuenta.java" file open in the code editor. The code defines a class `CCuenta` with attributes `nombre`, `cuenta`, `saldo`, and `tipoInteres`. It includes methods for constructor, deposit, withdrawal, and getters/setters. The code is annotated with Javadoc and includes a copyright notice for "diego". The code editor shows syntax highlighting and code completion suggestions.

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cuentas;

/**
 *
 * @author diego
 */
public class CCuenta {
    private String nombre;
    private String cuenta;
    private double saldo;
    private double tipoInteres;
    public CCuenta() {}
    public CCuenta(String nom, String cue, double sal, double tipo) {
        nombre = nom;
        cuenta = cue;
        saldo = sal;
        tipoInteres = tipo;
    }
    public double saldo() { return saldo; }
    public void ingresar(double cantidad) throws Exception {
        if (cantidad < 0)
            throw new Exception("No se puede ingresar una cantidad negativa");
        saldo += cantidad;
    }
    public void retirar(double cantidad) throws Exception {
        if (cantidad >= 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (saldo < cantidad)
            throw new Exception("No hay suficiente saldo");
        saldo -= cantidad;
    }
    // Getters y setters
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }
    public String getCuenta() { return cuenta; }
    public void setCuenta(String cuenta) { this.cuenta = cuenta; }
    public double getSaldo() { return saldo; }
    public void setSaldo(double saldo) { this.saldo = saldo; }
    public double getTipoInteres() { return tipoInteres; }
    public void setTipoInteres(double tipoInteres) { this.tipoInteres = tipoInteres; }
}

```

```
package cuentas;

public class CCuenta {

    private String nombre;
    private String cuenta;
    private double saldo;
    private double tipointeres;

    public CCuenta() {}

    public CCuenta(String nom, String cue, double sal, double tipo) {
        nombre = nom;
        cuenta = cue;
        saldo = sal;
        tipointeres = tipo;
    }

    public double estado() { return saldo; }

    public void ingresar(double cantidad) throws Exception {
        if (cantidad < 0)
            throw new Exception("No se puede ingresar una cantidad negativa");
        saldo += cantidad;
    }

    public void retirar(double cantidad) throws Exception {
        if (cantidad <= 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (saldo < cantidad)
            throw new Exception("No hay suficiente saldo");
        saldo -= cantidad;
    }

    // Getters y setters

    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getCuenta() { return cuenta; }
    public void setCuenta(String cuenta) { this.cuenta = cuenta; }

    public double getSaldo() { return saldo; }
    public void setSaldo(double saldo) { this.saldo = saldo; }

    public double getTipointeres() { return tipointeres; }
    public void setTipointeres(double tipointeres) { this.tipointeres = tipointeres; }

}
```

- **Main.java**

NetBeans IDE 23 showing the creation and content of the Main.java class.

**New Java Class Dialog:**

```

Steps:
1. Choose File Type
2. Name and Location
Name and Location
  Class Name: Main
  Project: CCuentaProyecto2
  Location: Source Packages
  Package: cuentas
  Created File: iego/Documents/NetBeansProjects/CCuentaProyecto2/src/cuentas/Main.java
  Superclass: 
  Interfaces: 
  
```

**Navigator - Members:**

```

Members <empty>
  CCuenta
    CCuenta()
    CCuenta(String nom, String cue, double sal, double tip)
    estado(): double
    getNombre(): String
    getSaldo(): double
    getTipointeres(): double
    ingresar(double cantidad)
    retirar(double cantidad)
    setCuenta(String cuenta)
    setNombre(String nombre)
    setSaldo(double saldo)
    setTipointeres(double tipointeres)
    cuenta: String
    nombre: String
    saldo: double
    tipointeres: double
  
```

**Code Editor (Main.java):**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cuentas;

/**
 *
 * @author diego
 */
public class Main {
    public static void main(String[] args) {
        CCuenta cuenta = new CCuenta("Antonio López", "1000-2365-85-1230456789", 2500, 0);
        operativa_cuenta(cuenta, 655f);
    }
    private static void operativa_cuenta(CCuenta cuenta, float cantidad) {
        double saldoActual = cuenta.estado();
        System.out.println("El saldo actual es " + saldoActual);
        try {
            cuenta.retirar(2300);
        } catch (Exception e) {
            System.out.println("Fallo al retirar");
        }
        try {
            System.out.println("Ingreso en cuenta");
            cuenta.ingresar(cantidad);
        } catch (Exception e) {
            System.out.println("Fallo al ingresar");
        }
    }
}

```

```

package cuentas;

public class Main {

    public static void main(String[] args) {
        CCuenta cuenta1 = new CCuenta("Antonio López", "1000-2365-85-1230456789", 2500, 0);
        operativa_cuenta(cuenta1, 695f);
    }

    private static void operativa_cuenta(CCuenta cuenta, float cantidad) {
        double saldoActual = cuenta.estado();
        System.out.println("El saldo actual es " + saldoActual);

        try {
            cuenta.retirar(2300);
        } catch (Exception e) { System.out.println("Fallo al retirar"); }

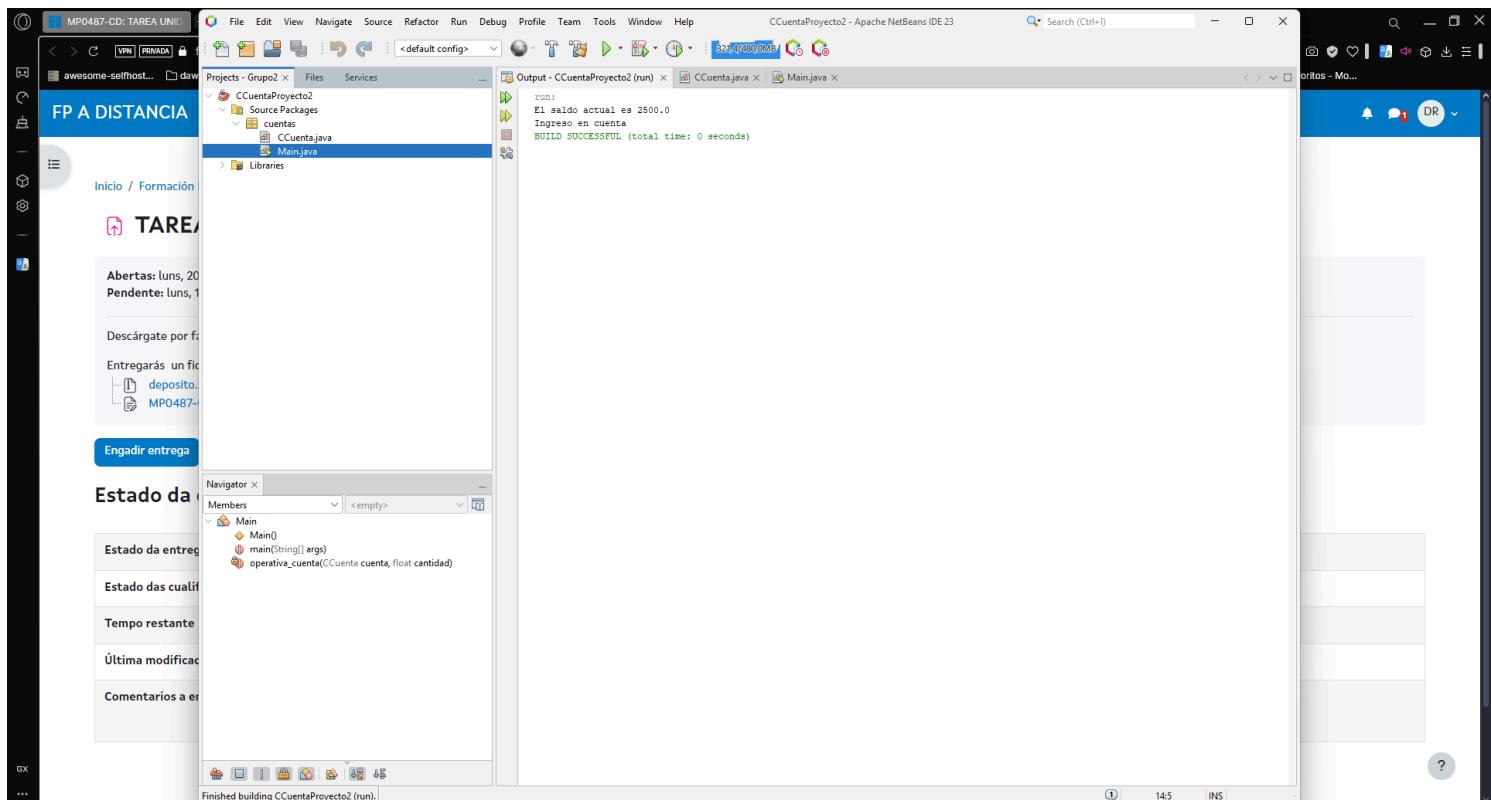
        try {
            System.out.println("Ingreso en cuenta");
            cuenta.ingresar(cantidad);
        } catch (Exception e) { System.out.println("Fallo al ingresar"); }
    }
}

```

#### Explicación de los cambios:

- Paquete cuentas: Ambas clases están ahora en el paquete cuentas.
- Renombrado de variable: miCuenta se cambió a cuenta1.
- Método operativa\_cuenta: Agrupa todas las operaciones relacionadas con cuenta1 y recibe el parámetro cantidad.
- Encapsulación en CCuenta: Se añadieron getters y setters para todos los atributos.
- Corrección en constructor: Se inicializa correctamente tipointeres.
- Uso del parámetro cantidad: Se utiliza en el método ingresar dentro de operativa\_cuenta.

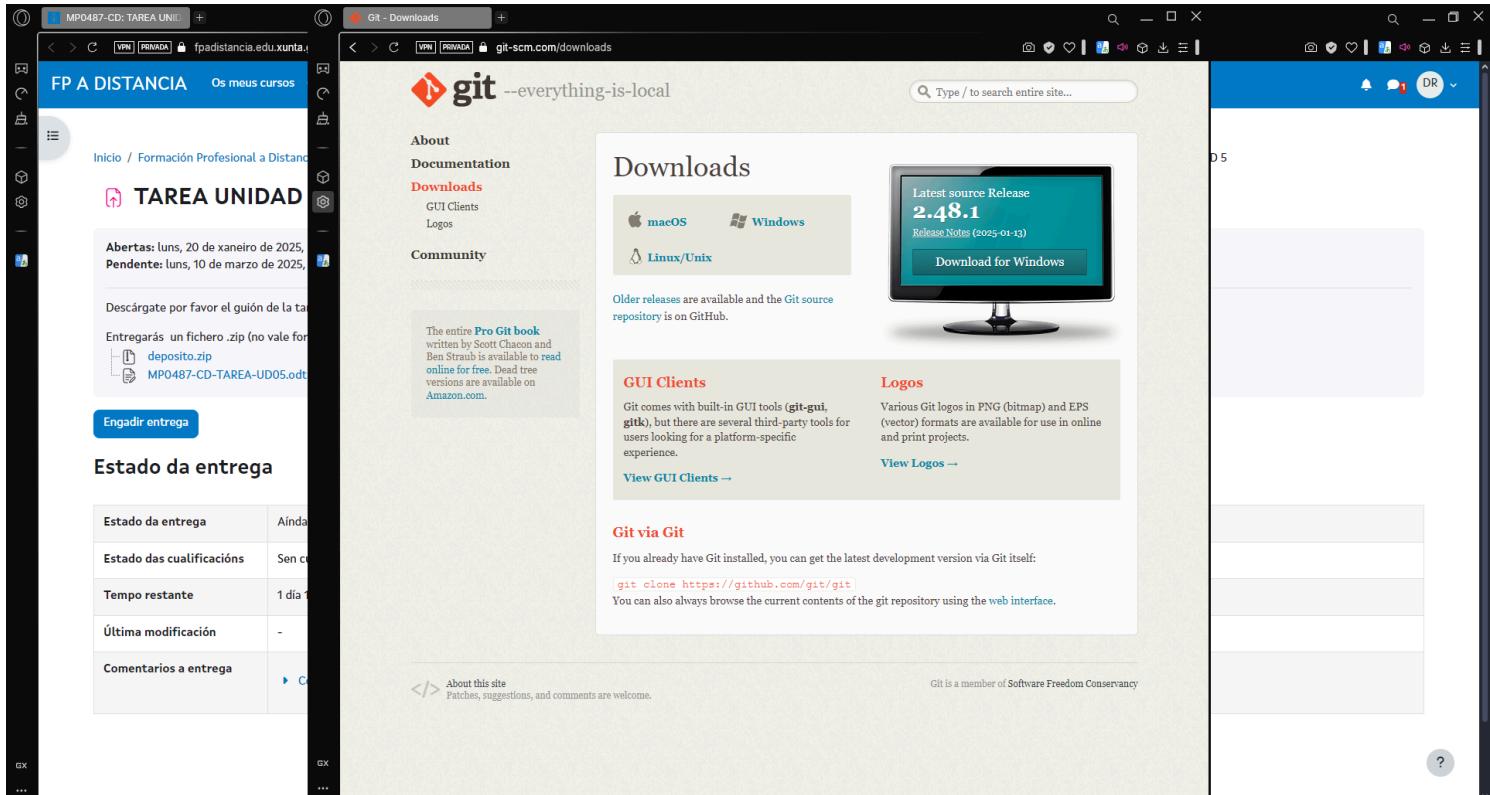
#### Ejecución de la clase Main.java (CCuenta.java > Run File)



## 2. Git

- 2.1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.
- 2.2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.
- 2.3. Mostrar el historial de versiones para el proyecto mediante un comando desde consola.

Descargar Git. No lo documento porque ya lo tengo instalado.

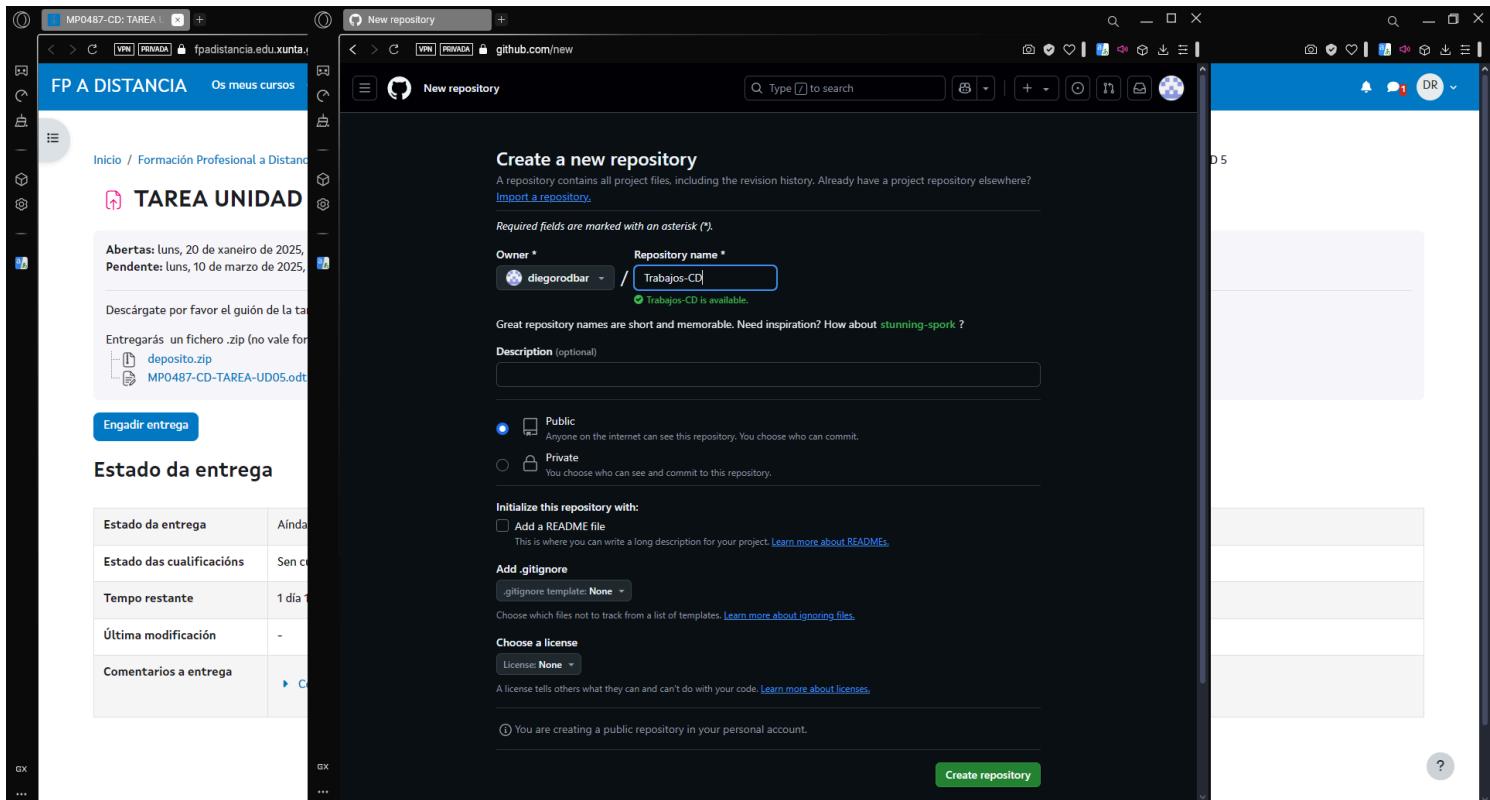


Configurar usuario y email (desde consola o NetBeans)

```
git config --global user.name "TuNombre"
```

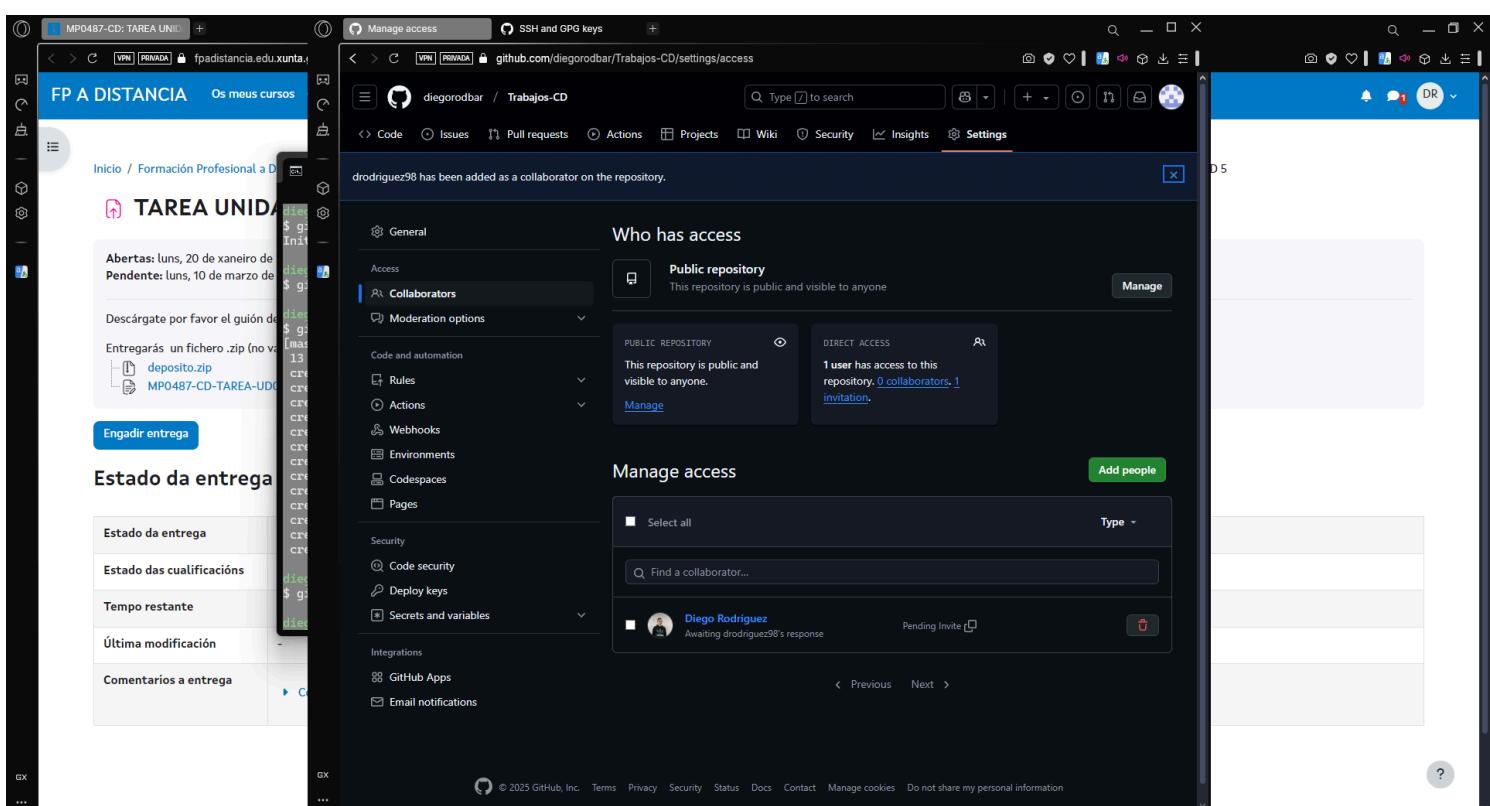
```
git config --global user.email "tua@email.com"
```

## Crear repositorio público en GitHub



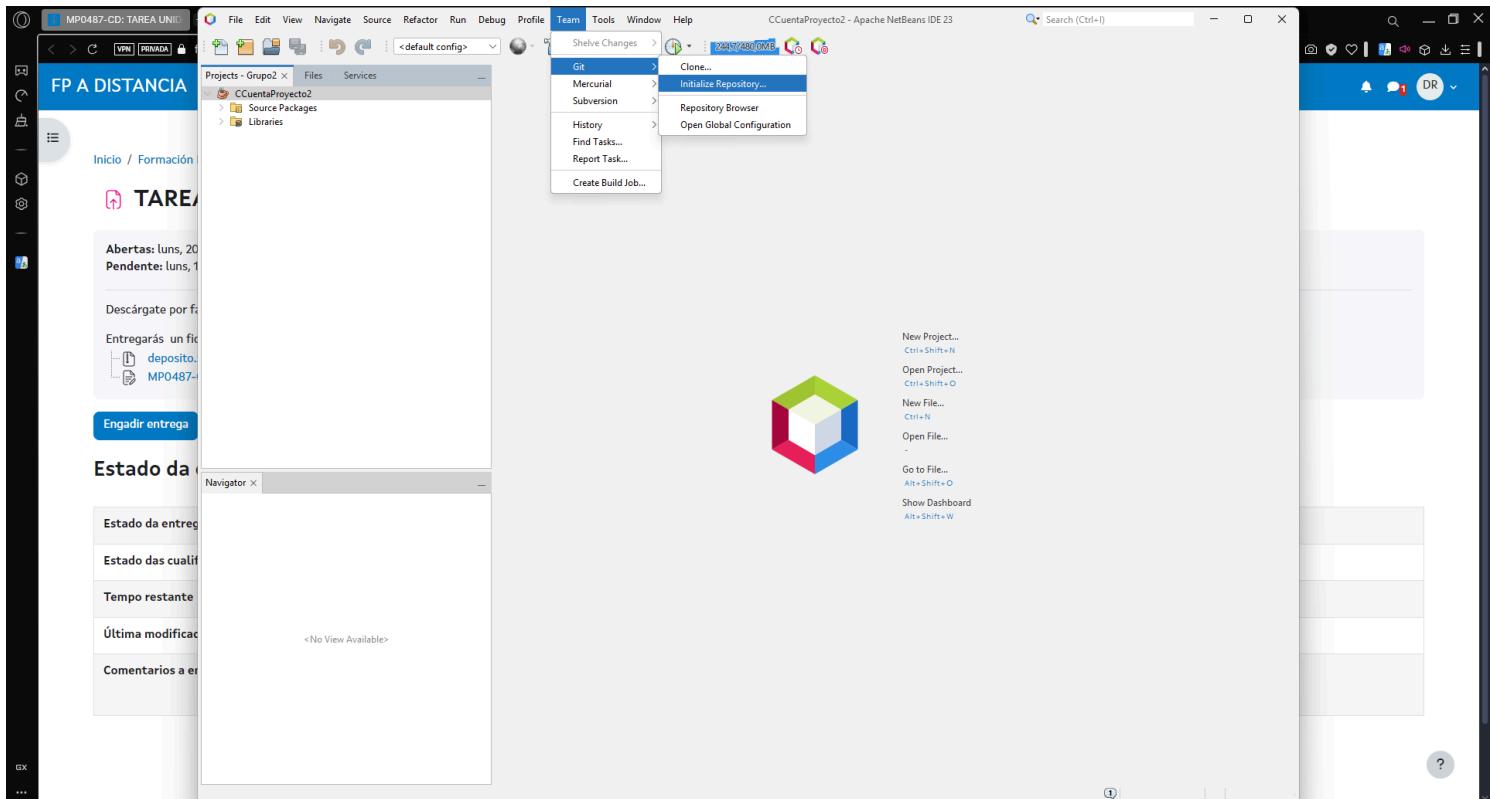
The screenshot shows a dual-browser setup. On the left, a browser window displays a course management system for 'FP A DISTANCIA'. On the right, a larger browser window is open to the GitHub 'New repository' page. In the GitHub interface, the repository name 'Trabajos-CD' is entered in the search bar. The 'Public' option is selected. At the bottom right of the GitHub form, there is a prominent green 'Create repository' button.

En mi equipo tengo vinculado Git a mi cuenta principal de GitHub. Para poder subir contenido a este repositorio añado mi cuenta principal como colaborador del nuevo repositorio. Es necesario aceptar la invitación, si no no permitirá subir nada.

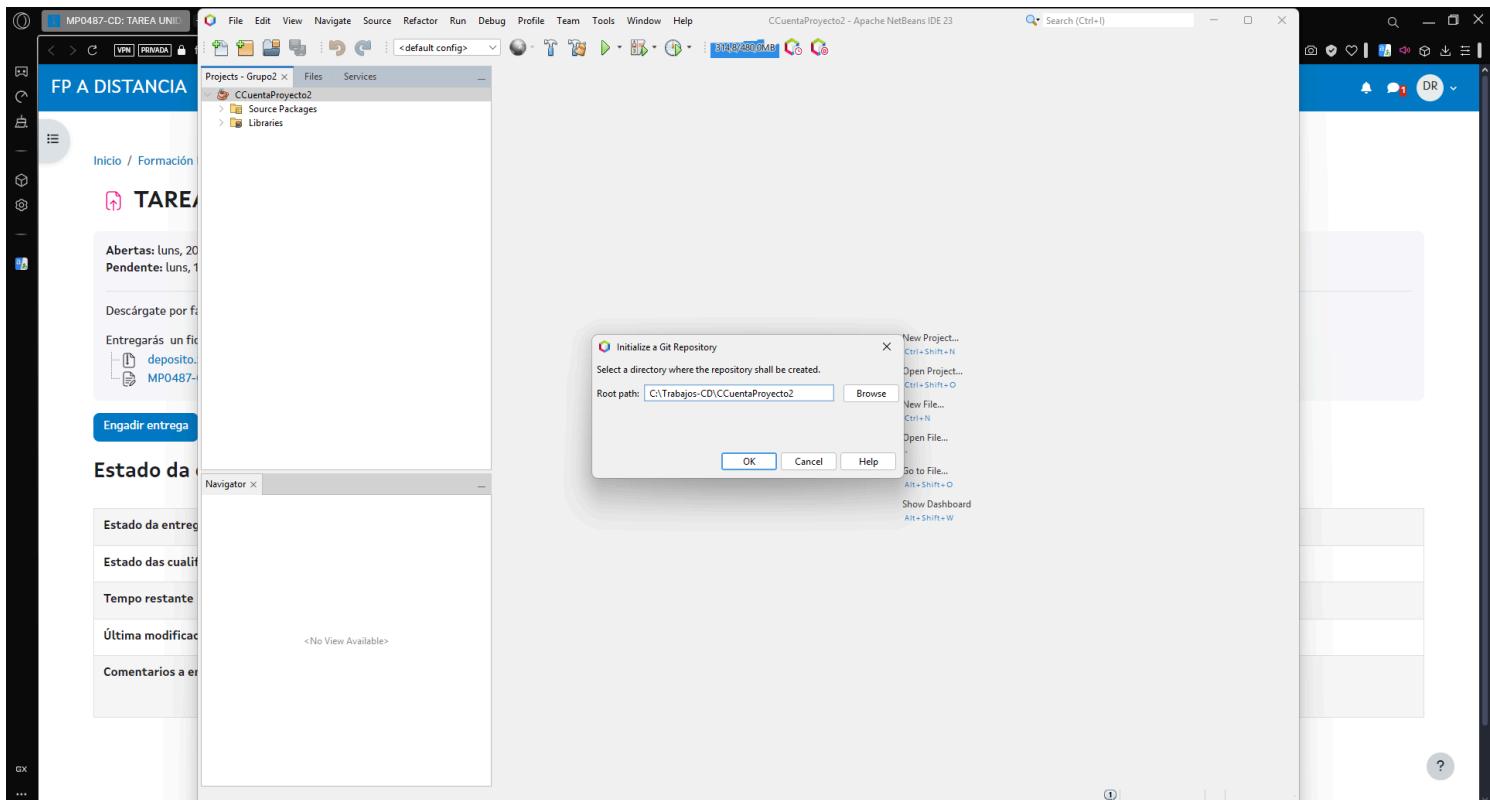


The screenshot shows the 'Manage access' settings for the 'Trabajos-CD' repository on GitHub. Under the 'Who has access' section, it is noted that 'dierodobar' has been added as a collaborator. In the 'Manage access' section, 'Diego Rodriguez' is listed with a status of 'Awaiting dierodobar's response' and a 'Pending Invite' button. The GitHub footer at the bottom includes links for Terms, Privacy, Security, Status, Docs, Contact, Manage cookies, and Do not share my personal information.

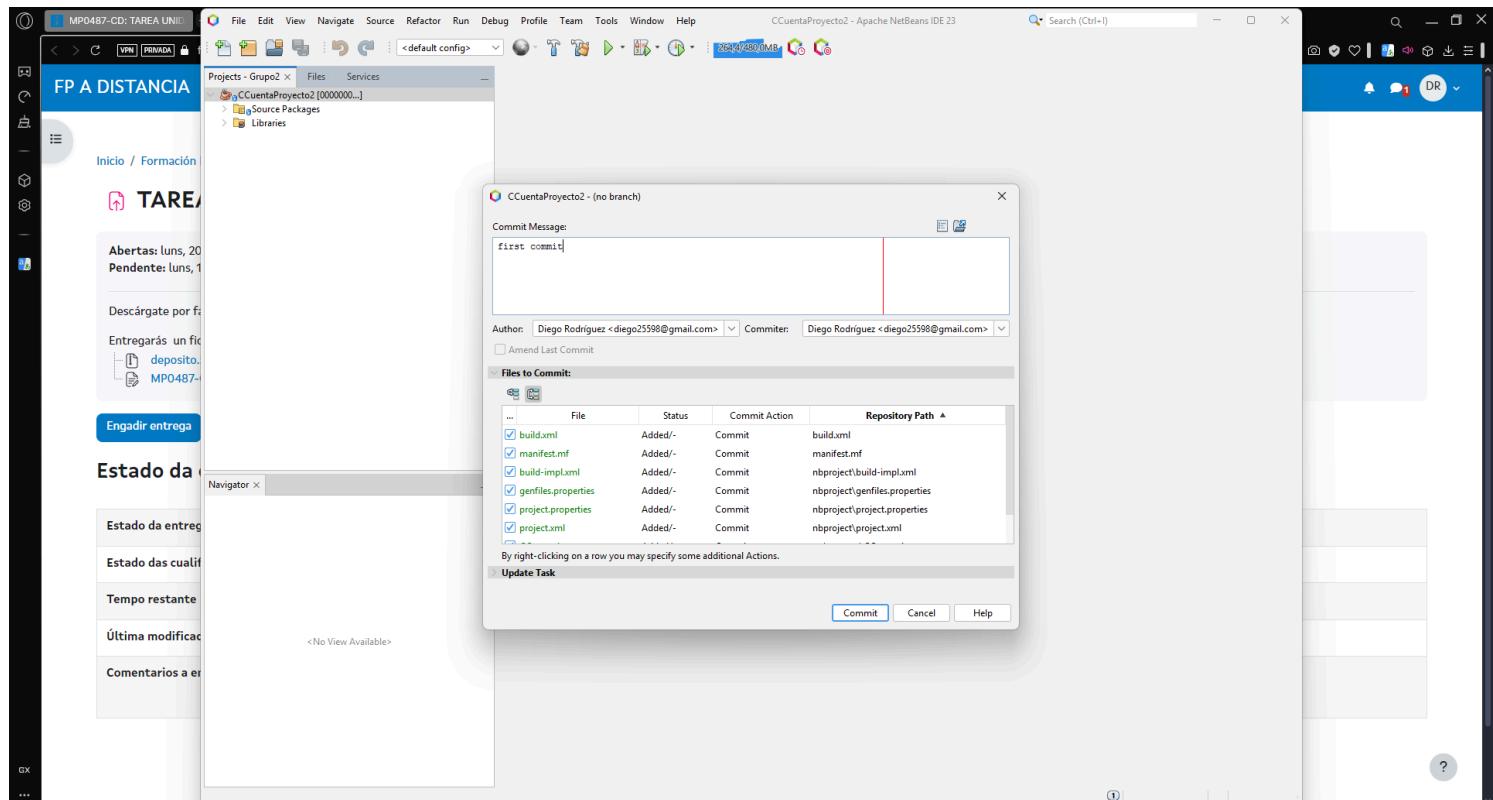
En Netbeans acceder a Team > Git > Initialize repository. Equivale a git init.



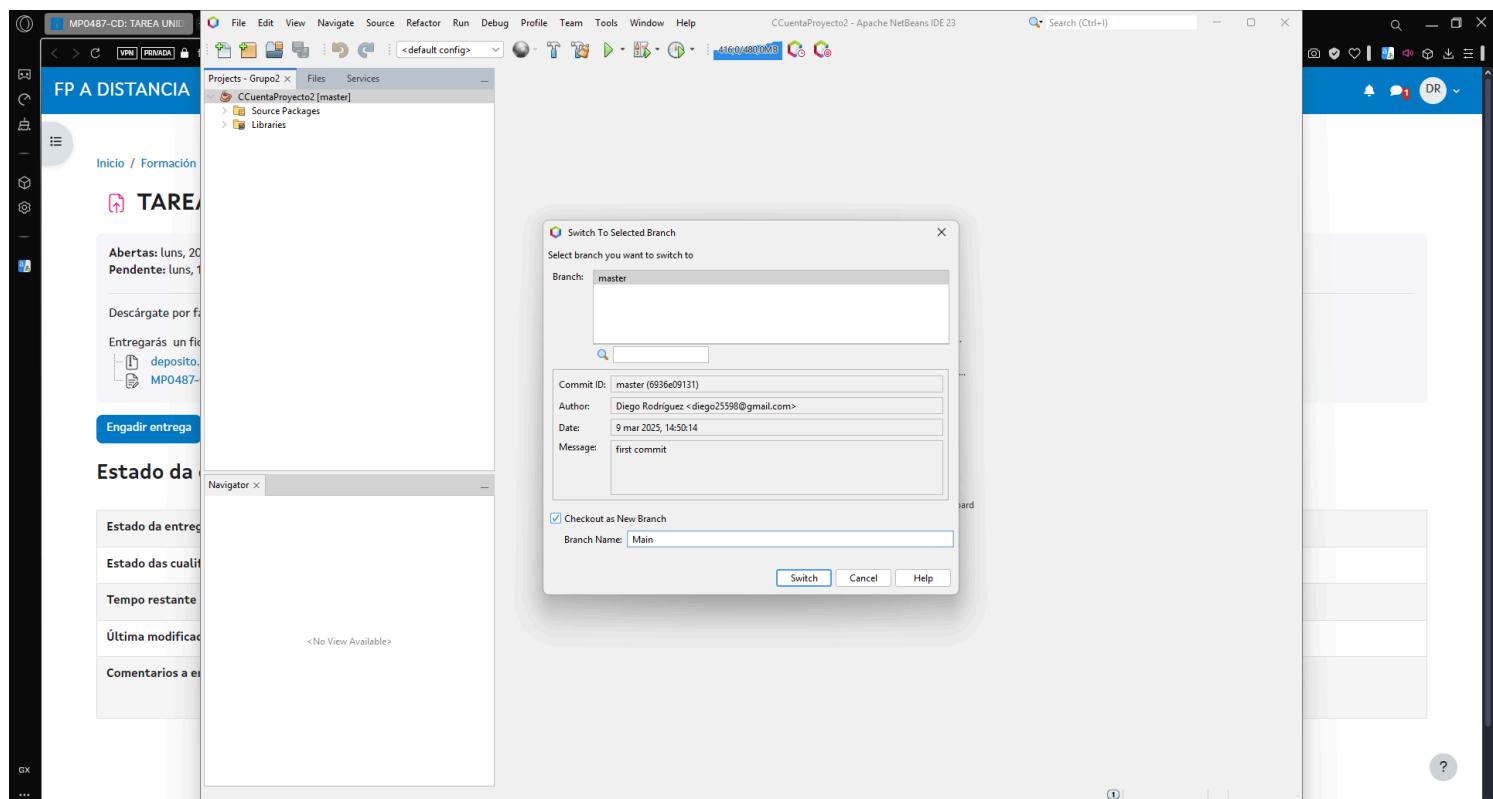
Indicar el directorio raíz del repositorio.

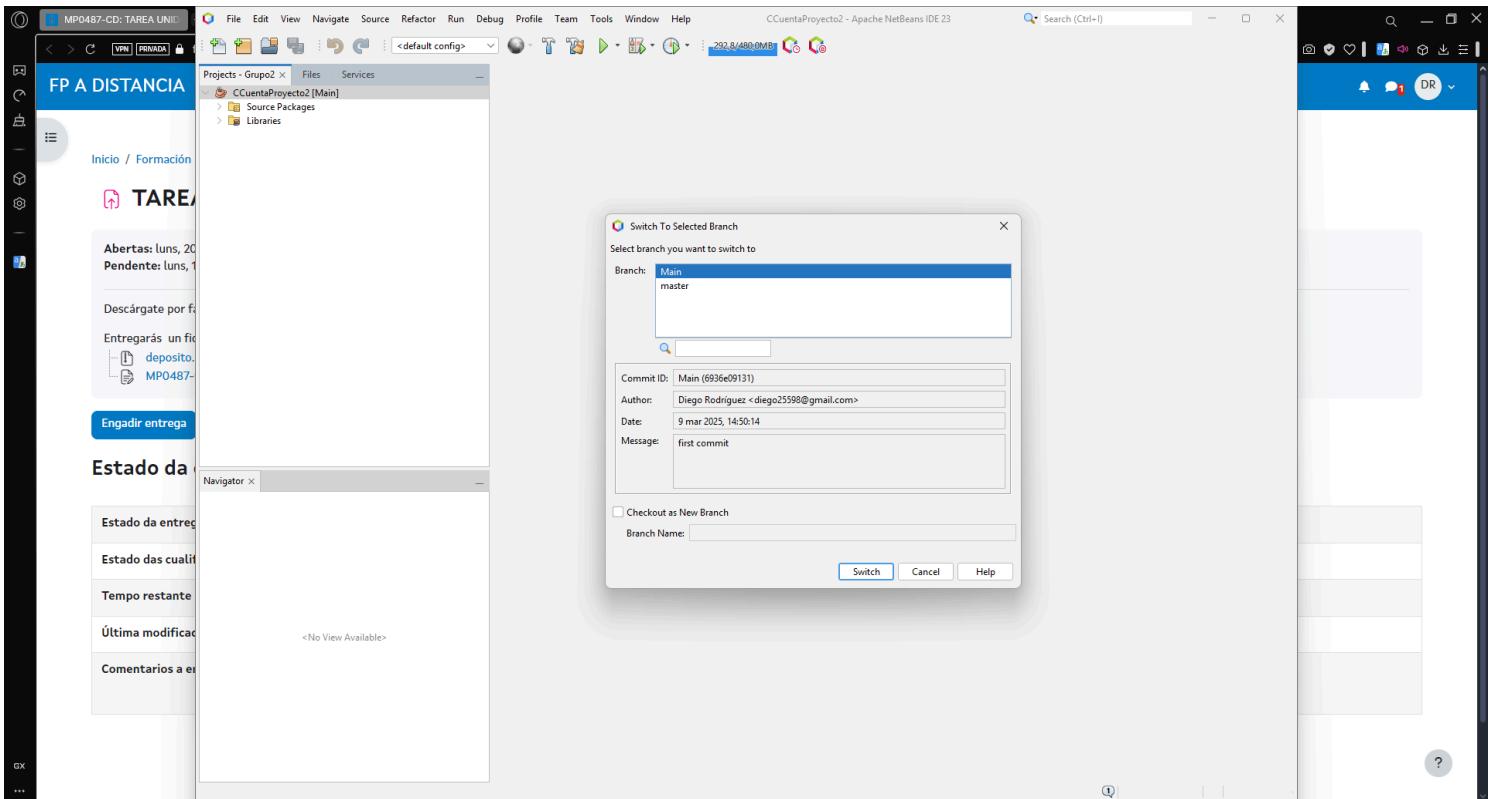


Indicar que queremos añadir al siguiente commit todo el contenido del proyecto (Team > Add) y hacer el primer commit (Team > Commit).



Crear y cambiar a la rama Main antes de subir el proyecto a GitHub (Team > Branch/Tag > Switch to branch > Checkout as new branch).

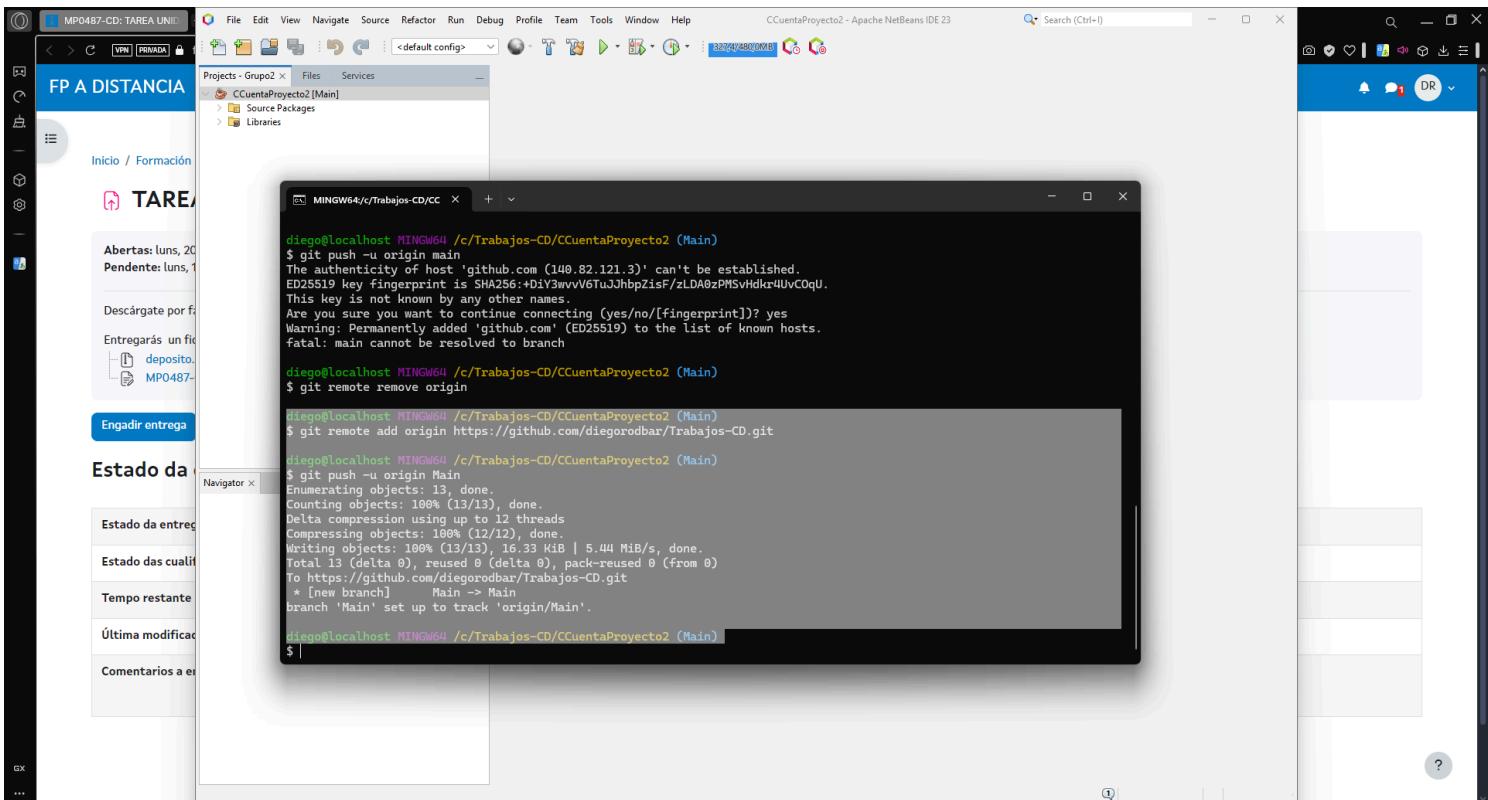




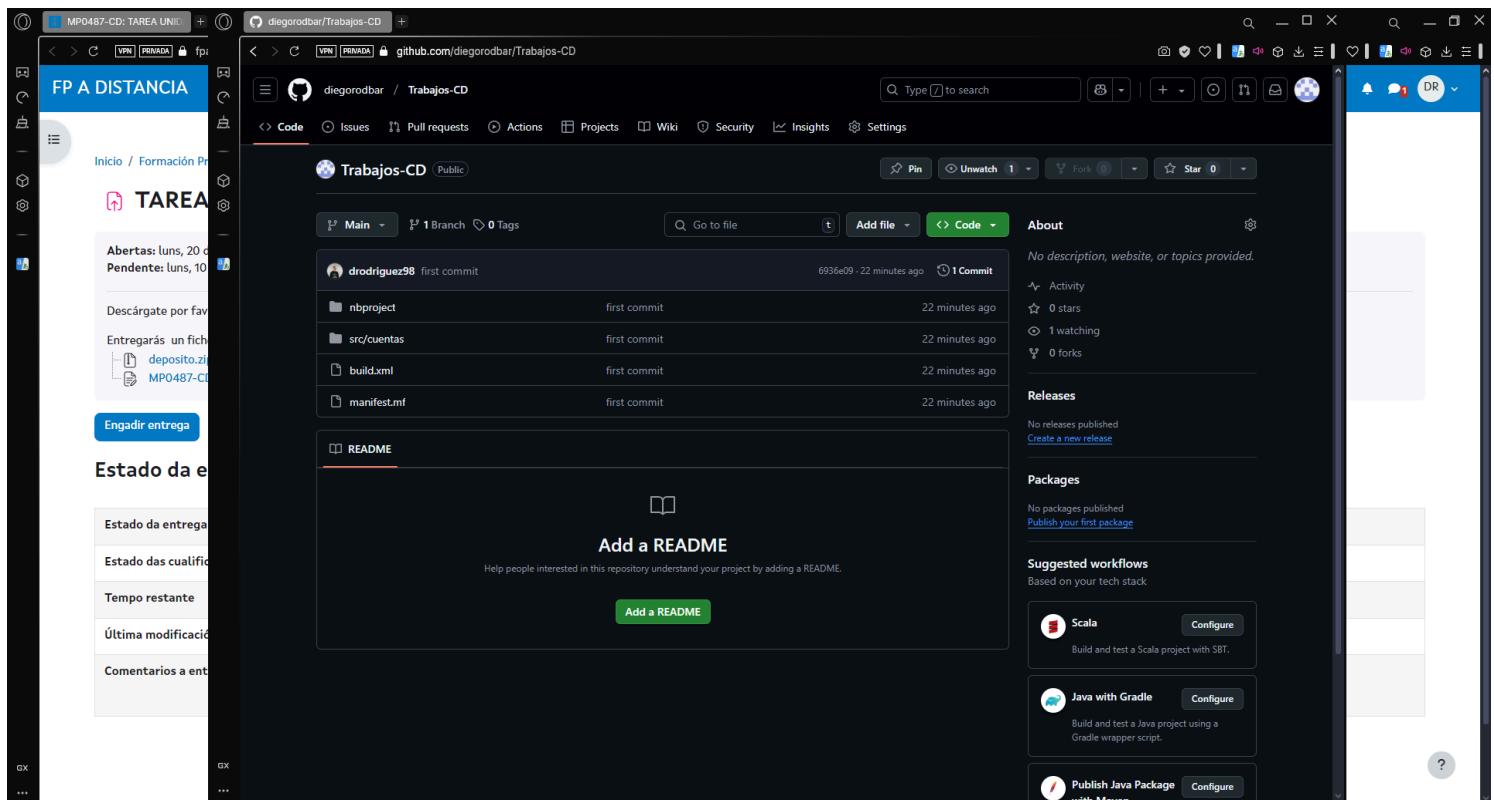
Añadir el repositorio remoto desde Git Bash y subir el proyecto a GitHub.

```
git remote add origin https://github.com/diegorodbar/Trabajos-CD.git
```

```
git push -u origin Main
```



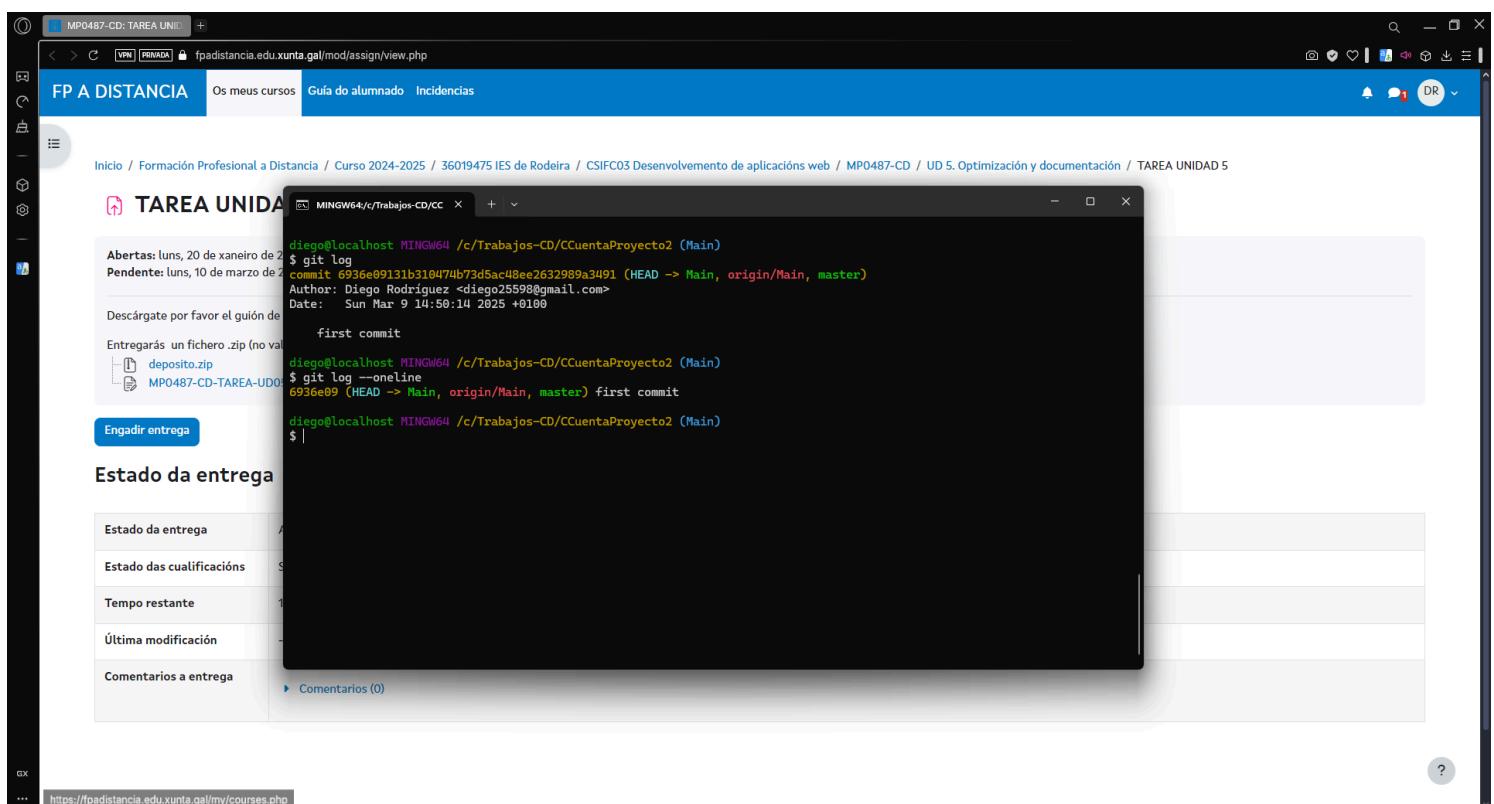
Comprobar que se ha subido el proyecto al repositorio remoto.



Para mostrar el historial de versiones del proyecto desde la consola, utilizar los siguientes comandos en Git Bash o terminal.

git log

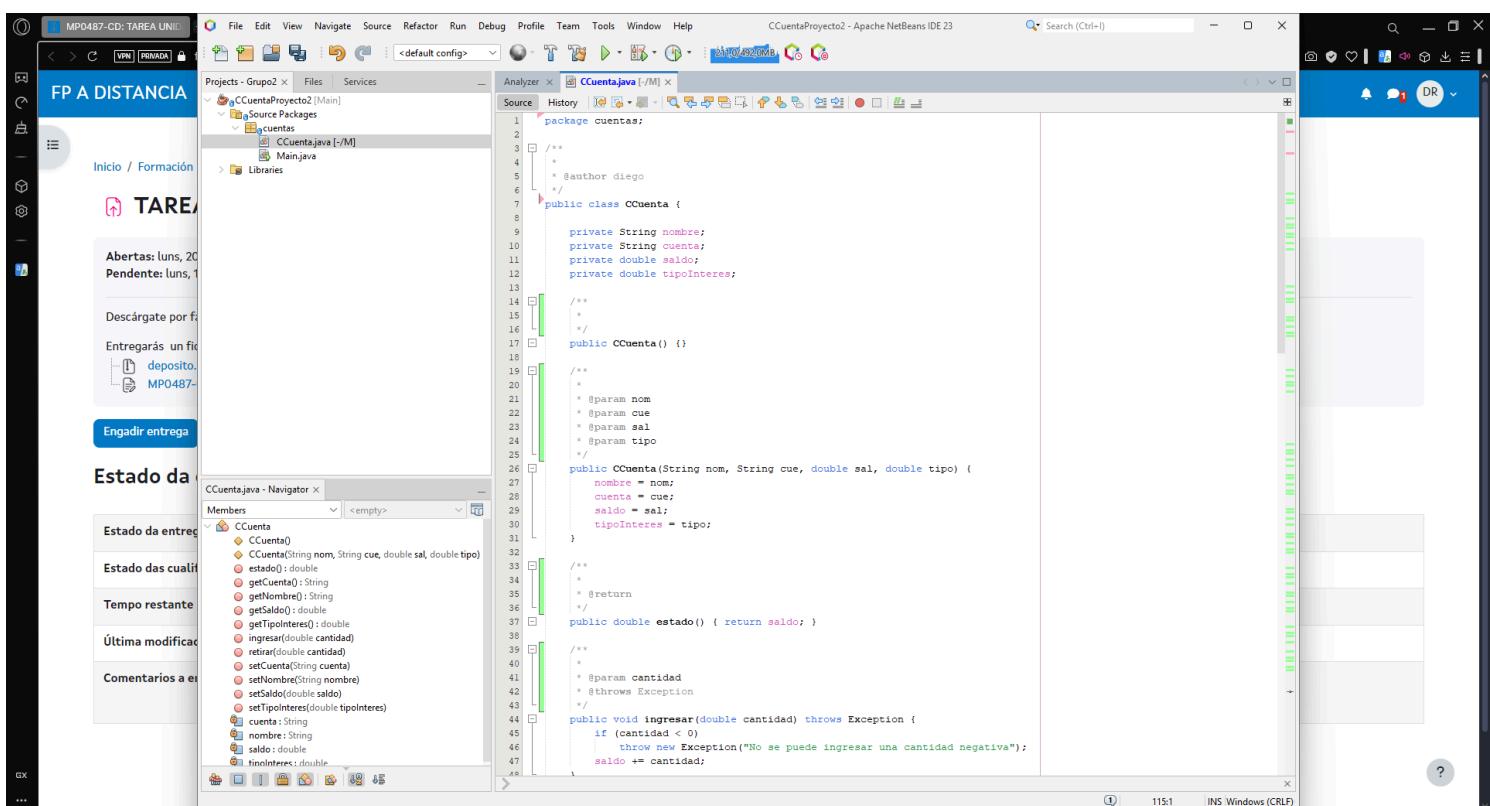
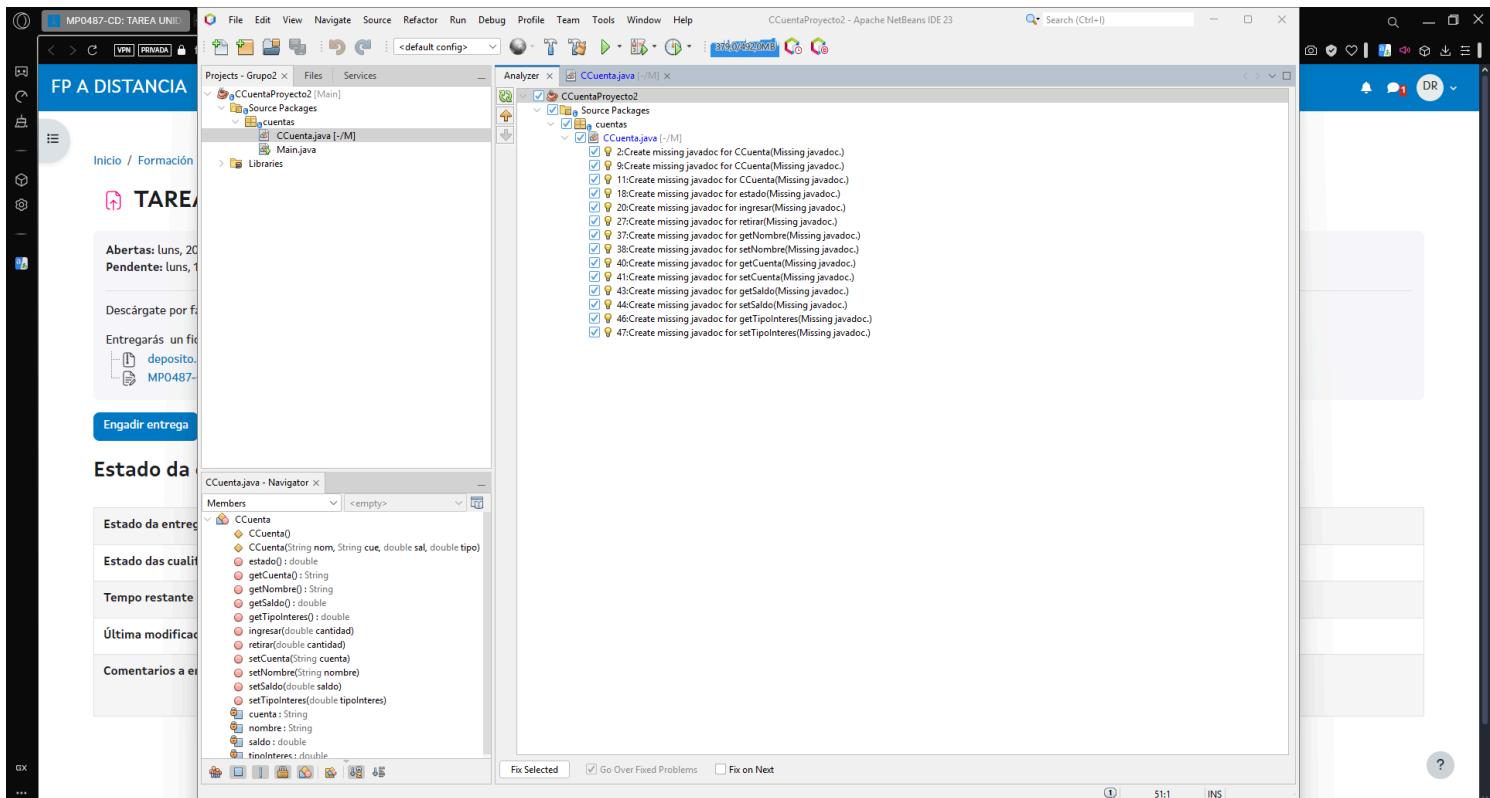
git log --oneline



### 3. Javadoc

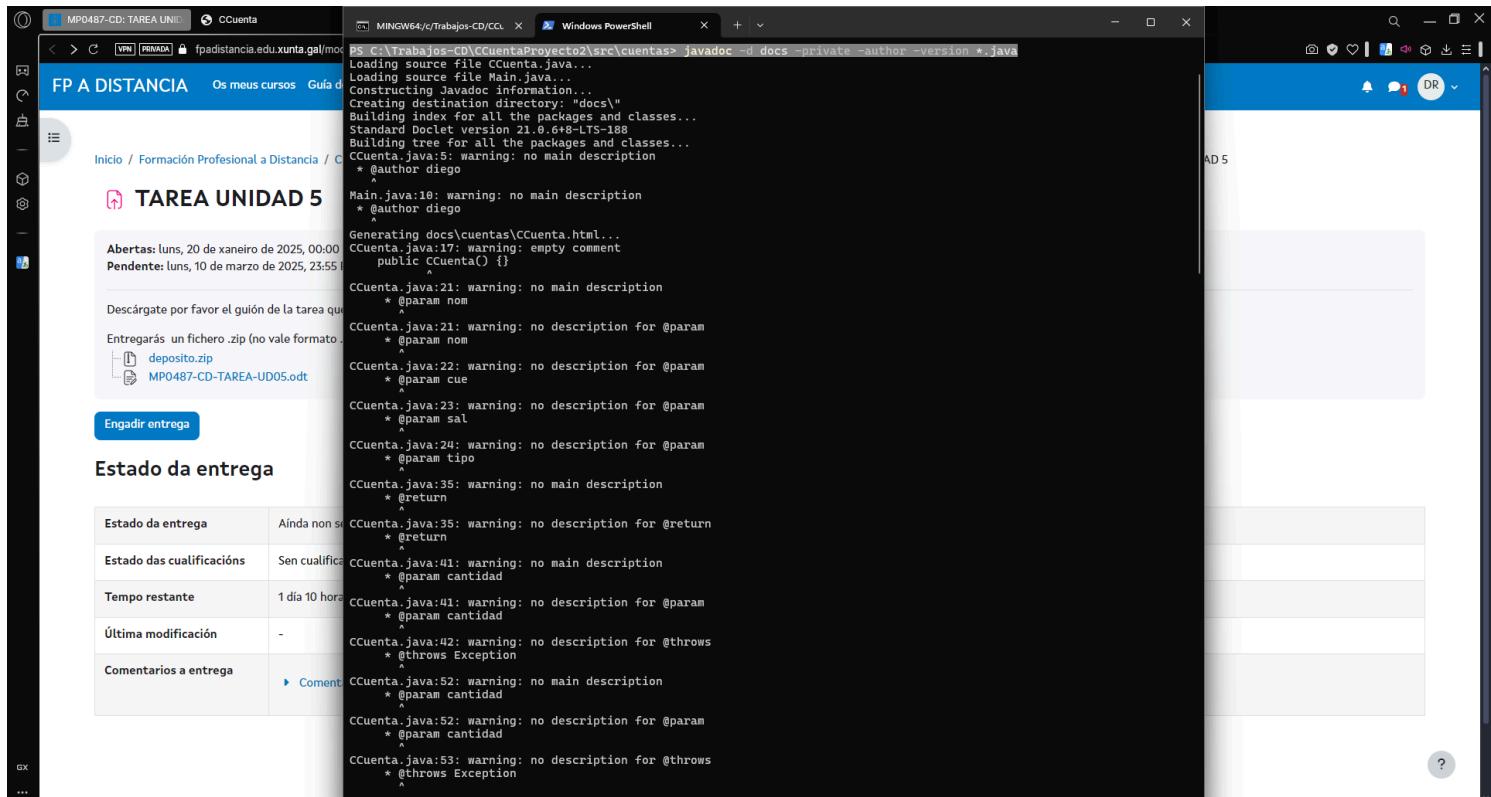
- 3.1. Insertar comentarios JavaDoc en la clase CCuenta.
  - 3.2. Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.

Se pueden generar automáticamente accediendo a Tools > Analyze Javadoc > Fix selected, seleccionando todos los archivos.

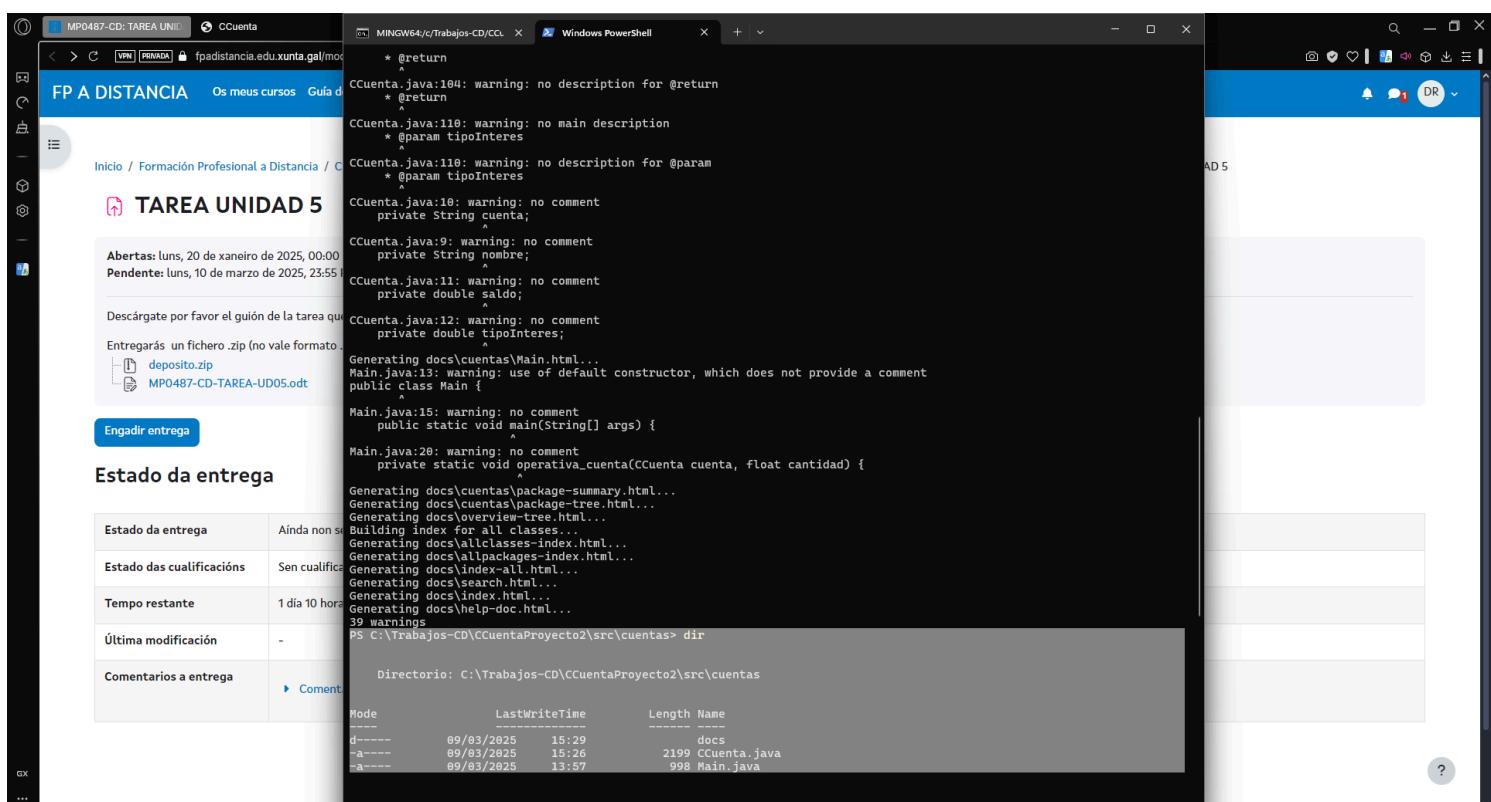


Desde terminal generar la documentación. Ejecutar el siguiente comando desde el paquete (directorio) cuentas:

```
javadoc -d docs -private -author -version *.java
```

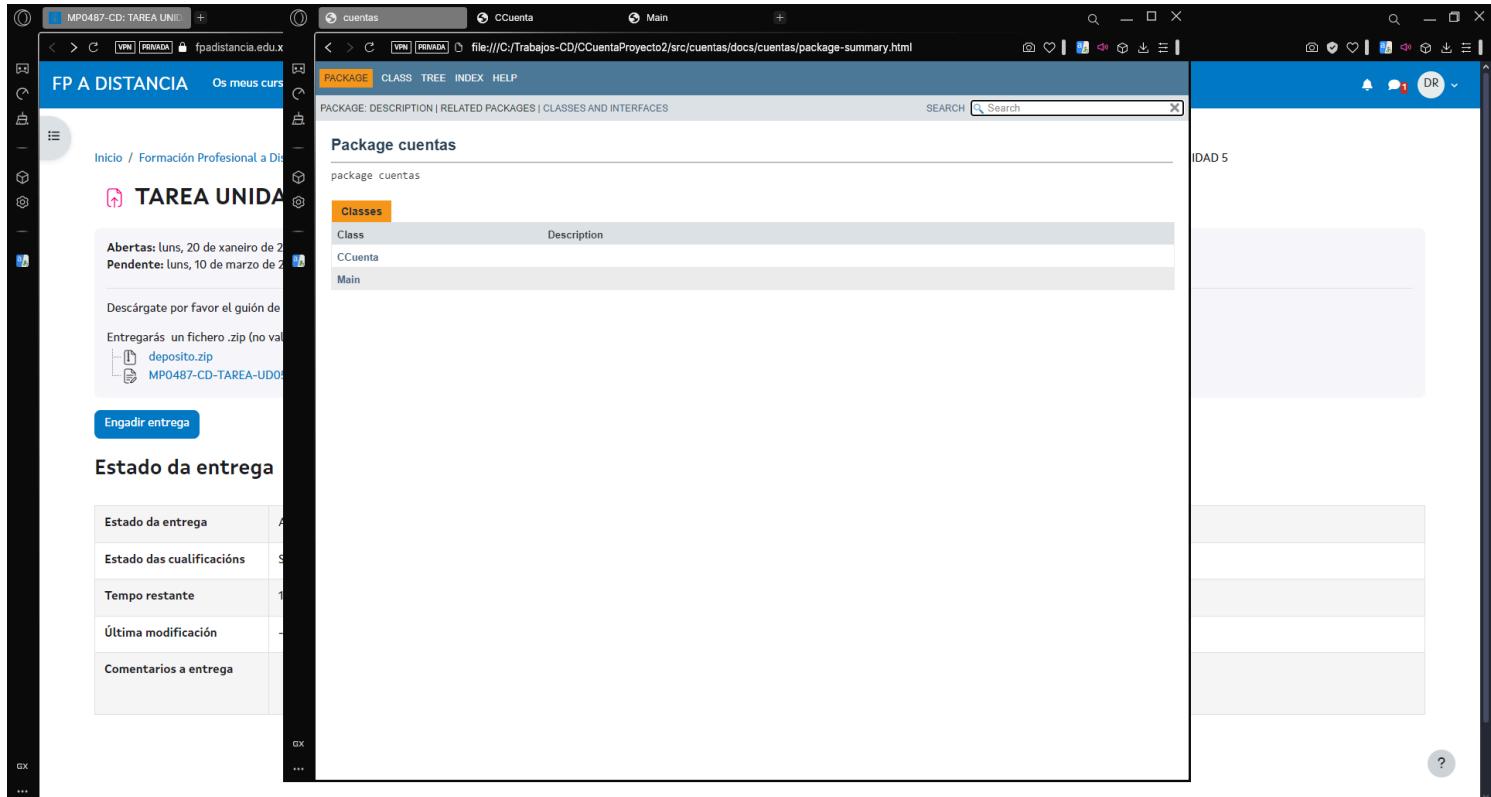


The screenshot shows a web browser window for 'FP A DISTANCIA' with the URL [fpdistancia.edu.xunta.gal/mod/assign/index.php?taskinstanceid=10000000000000000000000000000000](http://fpdistancia.edu.xunta.gal/mod/assign/index.php?taskinstanceid=10000000000000000000000000000000). The page displays information about the task and a table for submission status. To the right, a terminal window titled 'Windows PowerShell' runs the command `PS C:\Trabajos-CD\CCuentaProyecto2\src\cuentas> javadoc -d docs -private -author -version *.java`, producing numerous warnings about missing documentation for various class members.



The screenshot shows a web browser window for 'FP A DISTANCIA' with the URL [fpdistancia.edu.xunta.gal/mod/assign/index.php?taskinstanceid=10000000000000000000000000000000](http://fpdistancia.edu.xunta.gal/mod/assign/index.php?taskinstanceid=10000000000000000000000000000000). The page displays information about the task and a table for submission status. To the right, a terminal window titled 'Windows PowerShell' runs the command `PS C:\Trabajos-CD\CCuentaProyecto2\src\cuentas> javadoc -d docs -private -author -version *.java`, producing numerous warnings about missing documentation for various class members. It also shows the creation of documentation files like Main.html and package-summary.html.

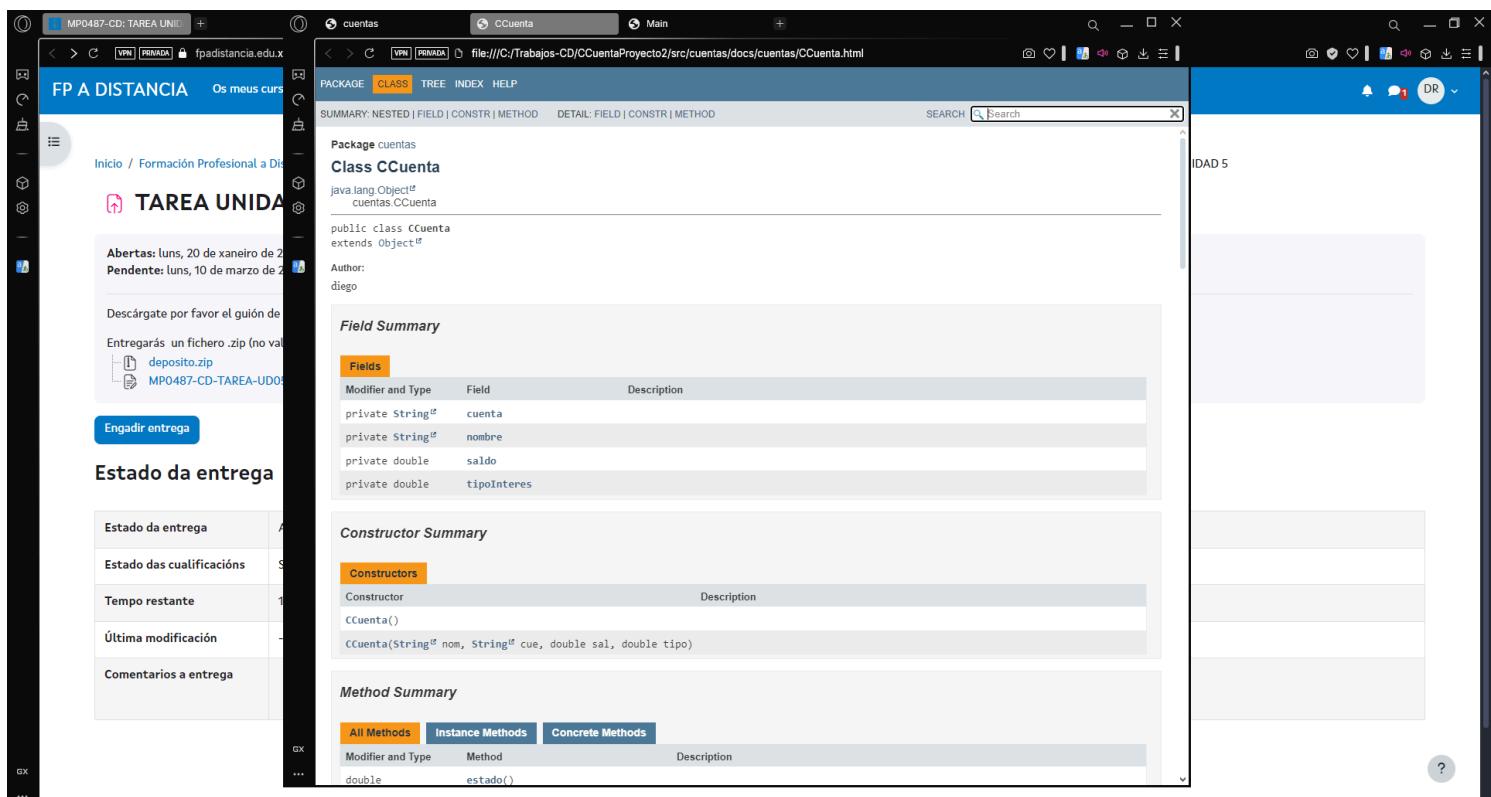
Abrir con el navegador el archivo index.html generado en el nuevo directorio docs y comprobar que la documentación se ha generado correctamente.



**PACKAGE: cuentas**

SEARCH  Search

Class	Description
CCuenta	
Main	



**Class CCuenta**

java.lang.Object<sup>if</sup>  
cuentas.CCuenta

public class CCuenta  
extends Object<sup>if</sup>

Author:  
diego

**Field Summary**

**Fields**

Modifier and Type	Field	Description
private String <sup>if</sup>	cuenta	
private String <sup>if</sup>	nombre	
private double	saldo	
private double	tipointeres	

**Constructor Summary**

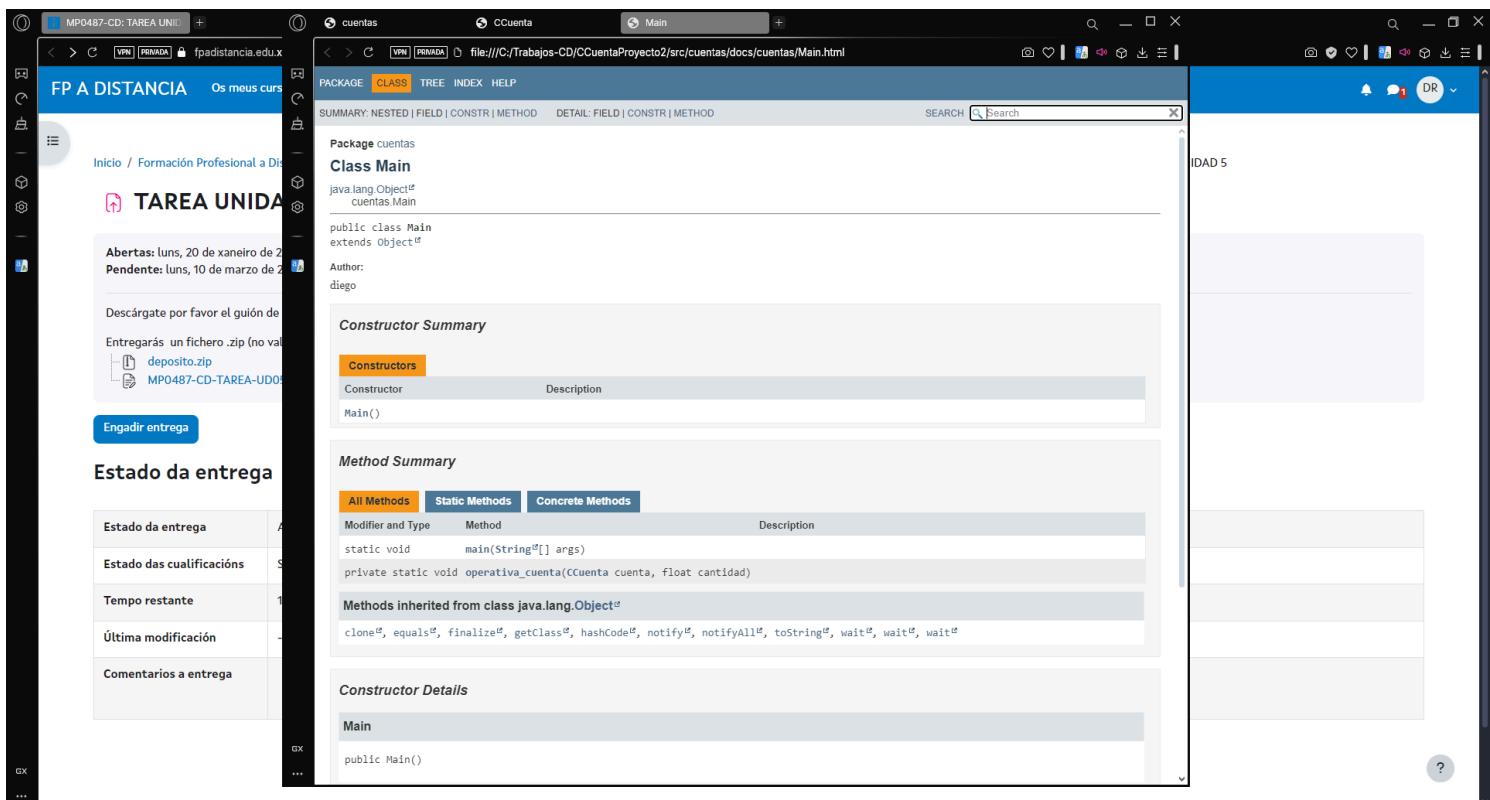
**Constructors**

Constructor	Description
CCuenta()	
CCuenta(String <sup>if</sup> nom, String <sup>if</sup> cue, double sal, double tipo)	

**Method Summary**

**All Methods**   **Instance Methods**   **Concrete Methods**

Modifier and Type	Method	Description
double	estado()	



**FP A DISTANCIA** Os meus cursos

**TAREA UNIDA**

Abertas: luns, 20 de xaneiro de 2024  
Pendente: luns, 10 de marzo de 2024

Descárgate por favor el guión de la tarea.

Entregarás un fichero .zip (no vale deposito.zip)

deposito.zip  
MPO487-CD-TAREA-UD05

Engadir entrega

**Estado da entrega**

Estado da entrega	A
Estado das cualificacións	S
Tempo restante	1
Última modificación	-
Comentarios a entrega	

GX ...

**cuentas** **CCuenta** **Main**

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH  Search

**Package cuentas**  
**Class Main**  
java.lang.Object<sup>1</sup>  
cuentas.Main

```
public class Main
extends Object1

Author: diego
```

**Constructor Summary**

**Constructors**

Constructor	Description
Main()	

**Method Summary**

**All Methods** **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
static void	main(String[] args)	
private static void	operativa_cuenta(CCuenta cuenta, float cantidad)	
<b>Methods inherited from class java.lang.Object<sup>1</sup></b>		
clone <sup>1</sup> , equals <sup>1</sup> , finalize <sup>1</sup> , getClass <sup>1</sup> , hashCode <sup>1</sup> , notify <sup>1</sup> , notifyAll <sup>1</sup> , toString <sup>1</sup> , wait <sup>1</sup> , wait <sup>1</sup> , wait <sup>1</sup>		

**Constructor Details**

**Main**

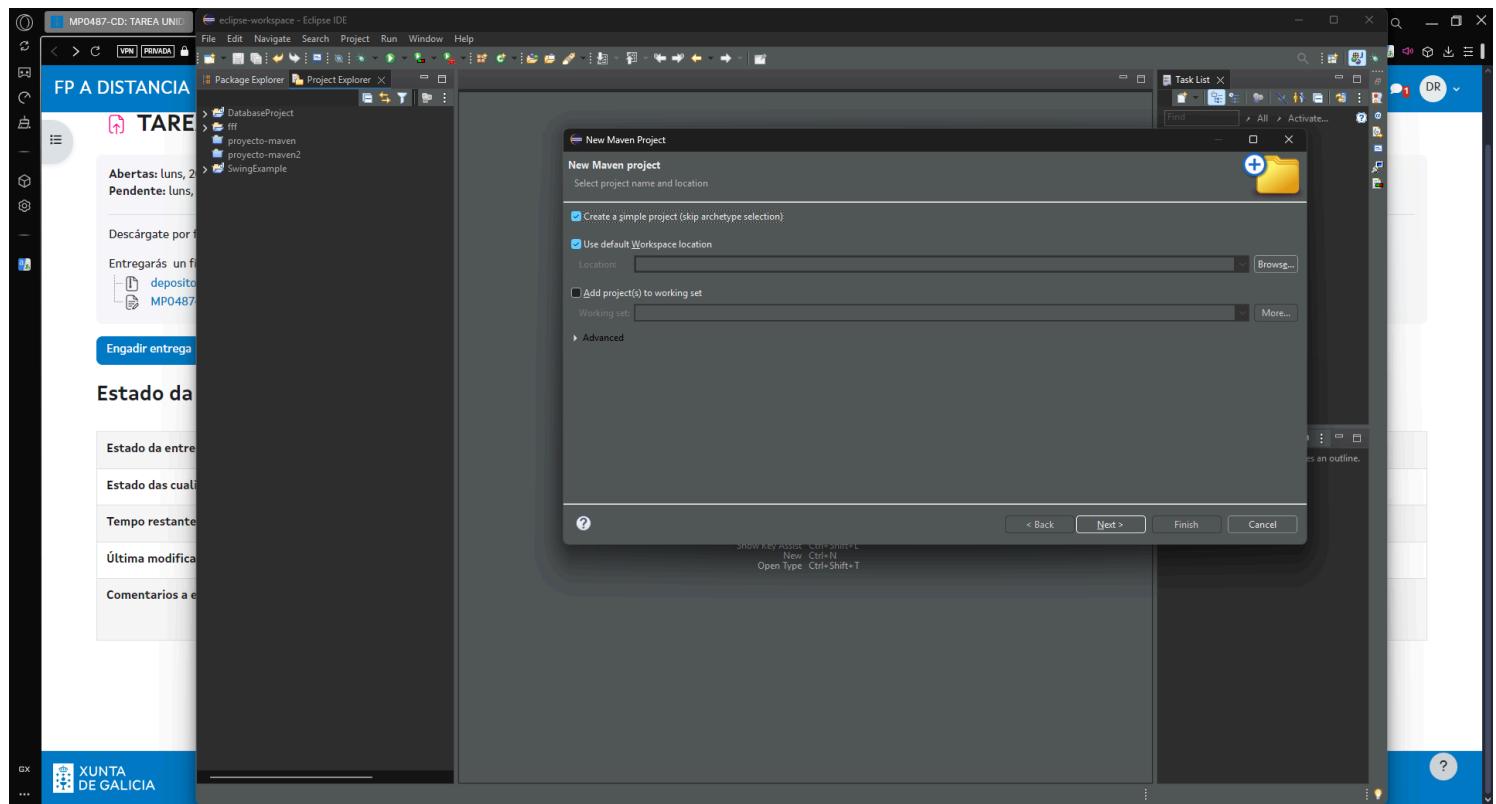
```
public Main()
```

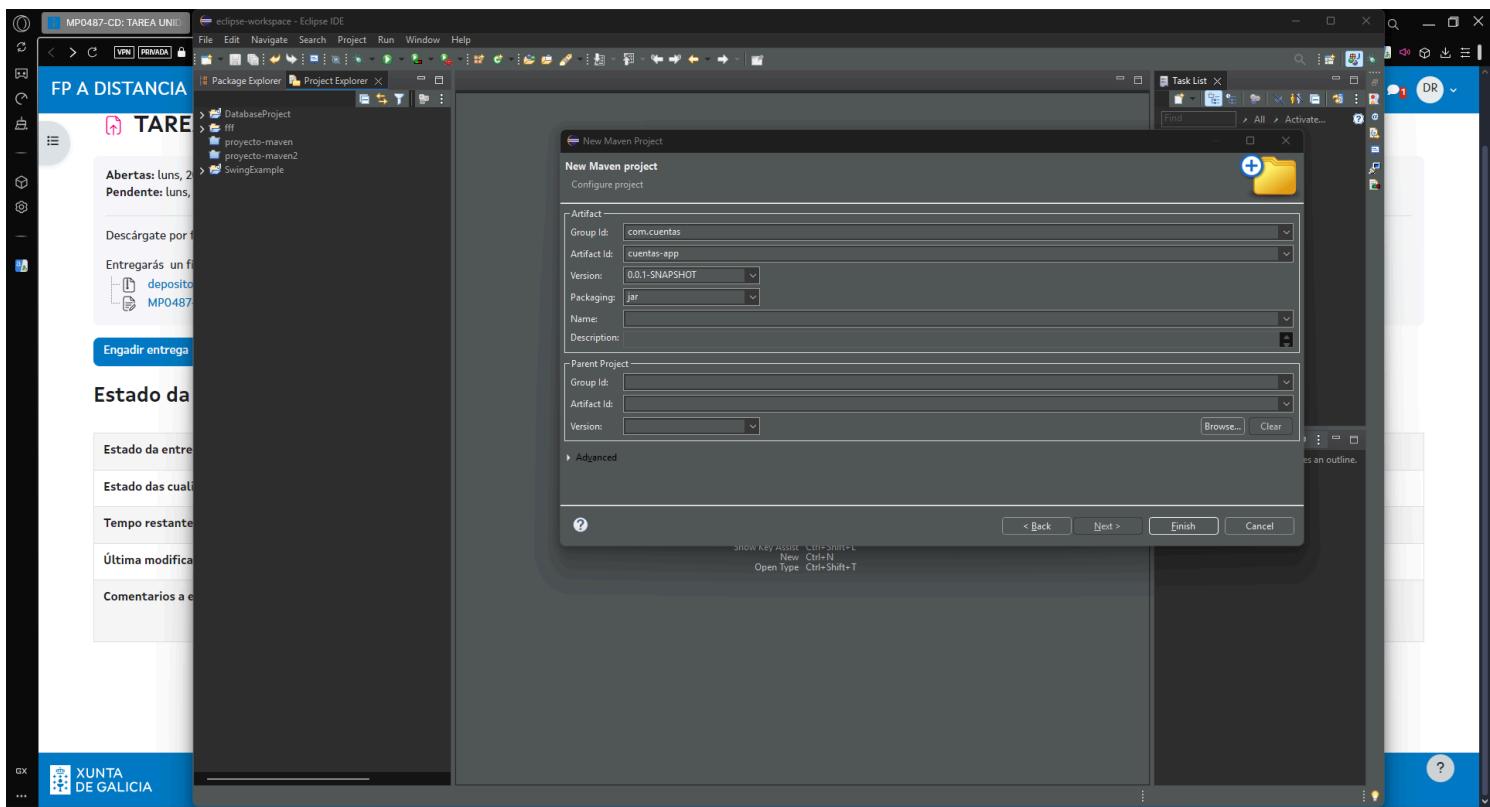
## ECLIPSE

### 1. Refactorización

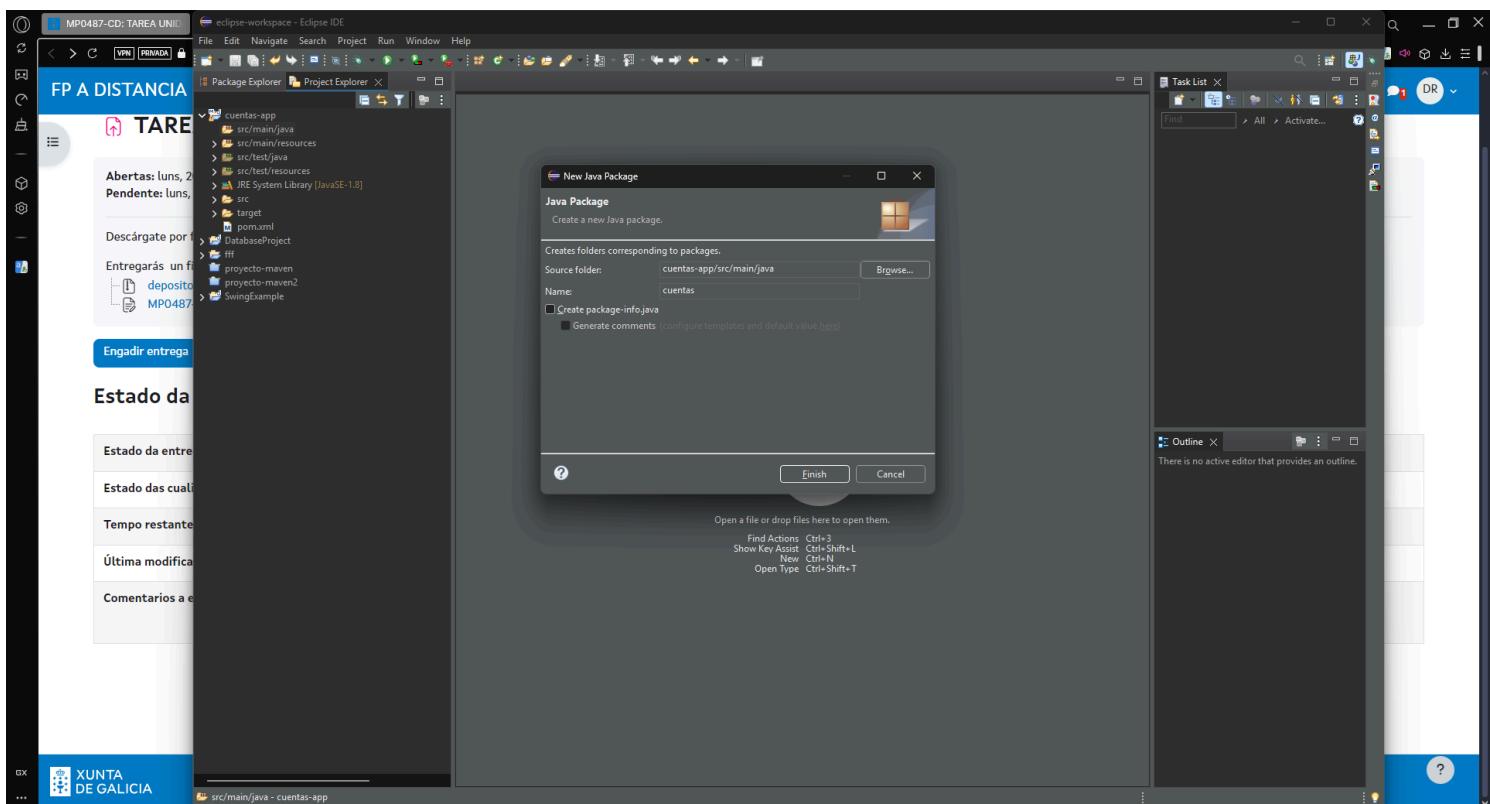
- 1.1. Las clases deberán formar parte del paquete cuentas.
- 1.2. Cambiar el nombre de la variable "miCuenta" por "cuenta1".
- 1.3. Introducir el método operativa\_cuenta, que englobe las sentencias de la clase Main que operan con el objeto cuenta1.
- 1.4. Encapsular los atributos de la clase CCuenta.
- 1.5. Añadir un nuevo parámetro al método operativa\_cuenta, de nombre cantidad y de tipo float.

Crear proyecto en Eclipse (File > New > Maven Project).



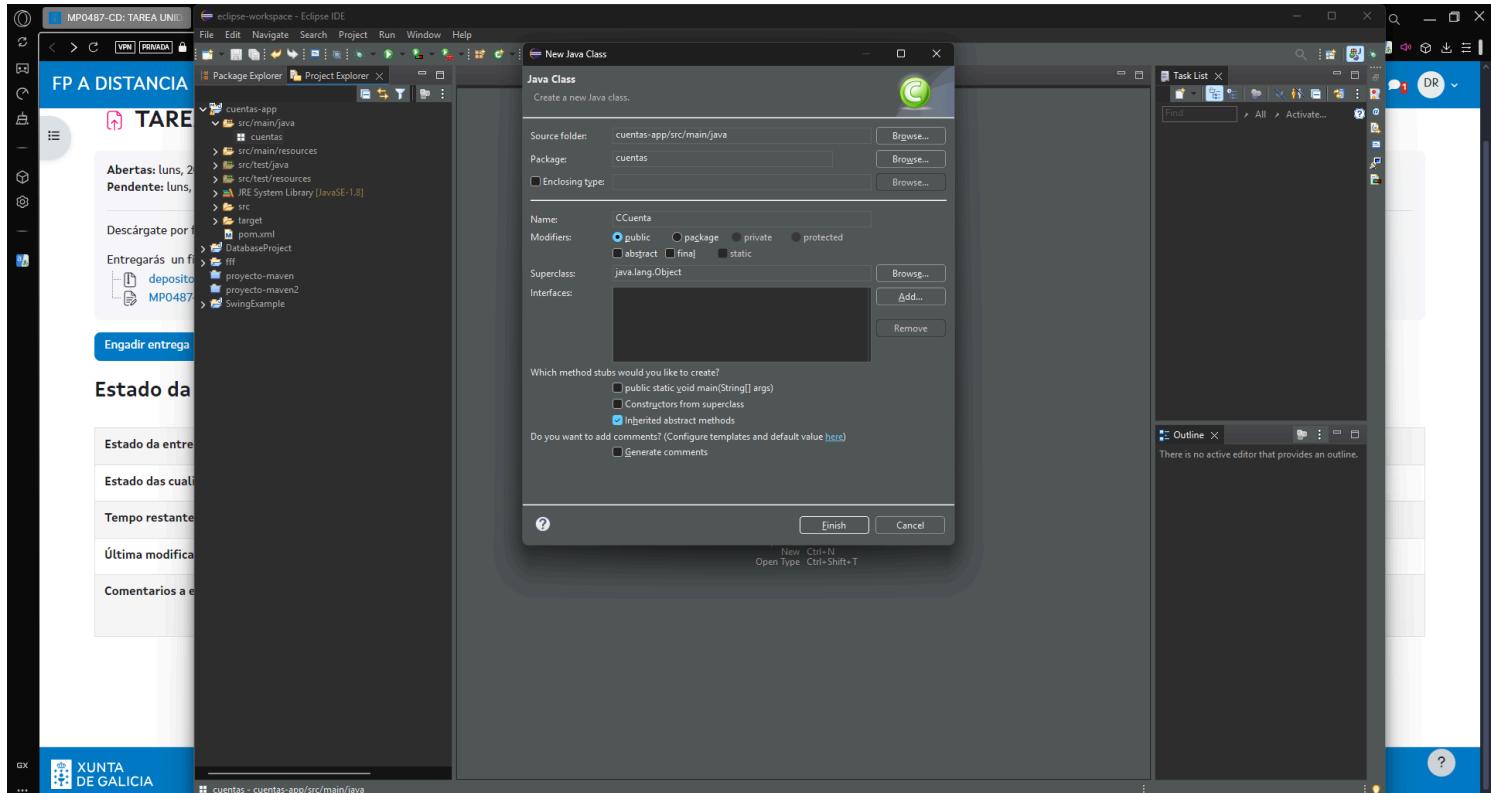
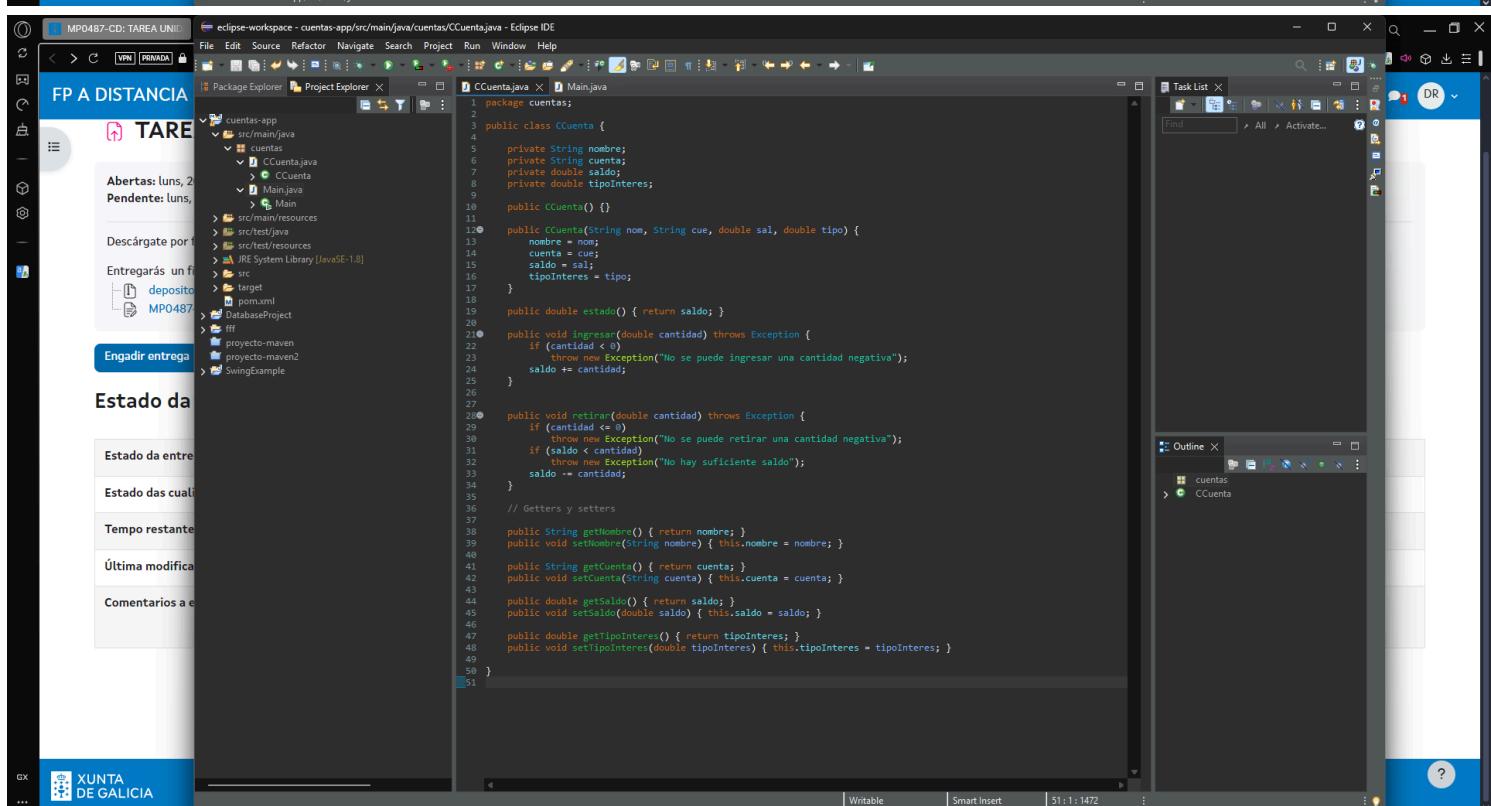


Crear paquete cuentas (src/main/java > New > Package).



Para cumplir con los requerimientos del enunciado, creamos las clases en el paquete cuentas y realizamos las mismas modificaciones que en Netbeans en el código aportado en el enunciado. Los cambios aplicados ya se han mencionado en el apartado de Netbeans.

### - CCuenta.java

```

package cuentas;

public class CCuenta {
    private String nombre;
    private String cuenta;
    private double saldo;
    private double tipoInteres;

    public CCuenta() {}

    public CCuenta(String nom, String cue, double sal, double tipo) {
        nombre = nom;
        cuenta = cue;
        saldo = sal;
        tipoInteres = tipo;
    }

    public double estado() { return saldo; }

    public void ingresar(double cantidad) throws Exception {
        if (cantidad < 0)
            throw new Exception("No se puede ingresar una cantidad negativa");
        saldo += cantidad;
    }

    public void retirar(double cantidad) throws Exception {
        if (cantidad < 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (saldo < cantidad)
            throw new Exception("No hay suficiente saldo");
        saldo -= cantidad;
    }

    // Getters y setters

    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

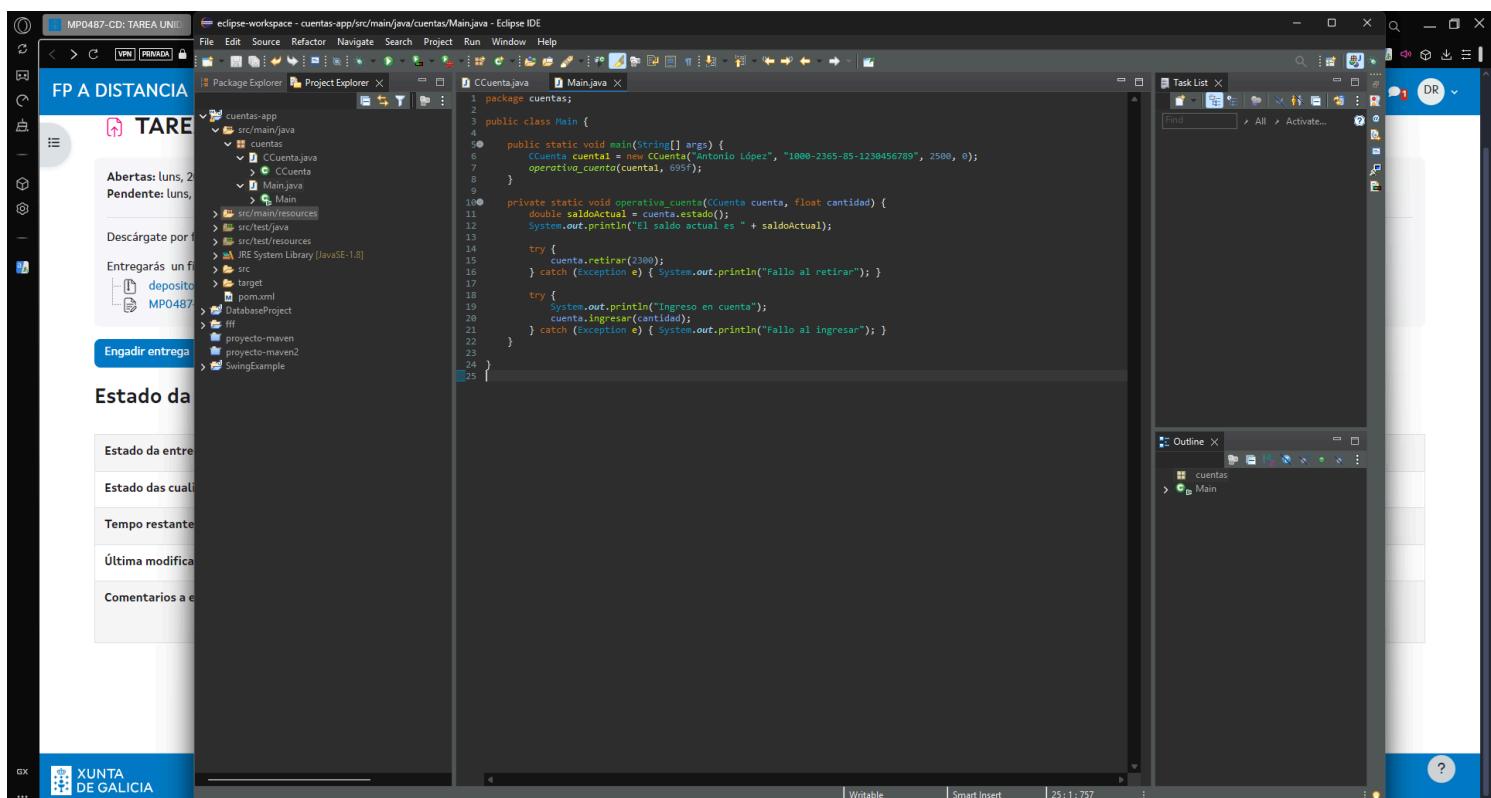
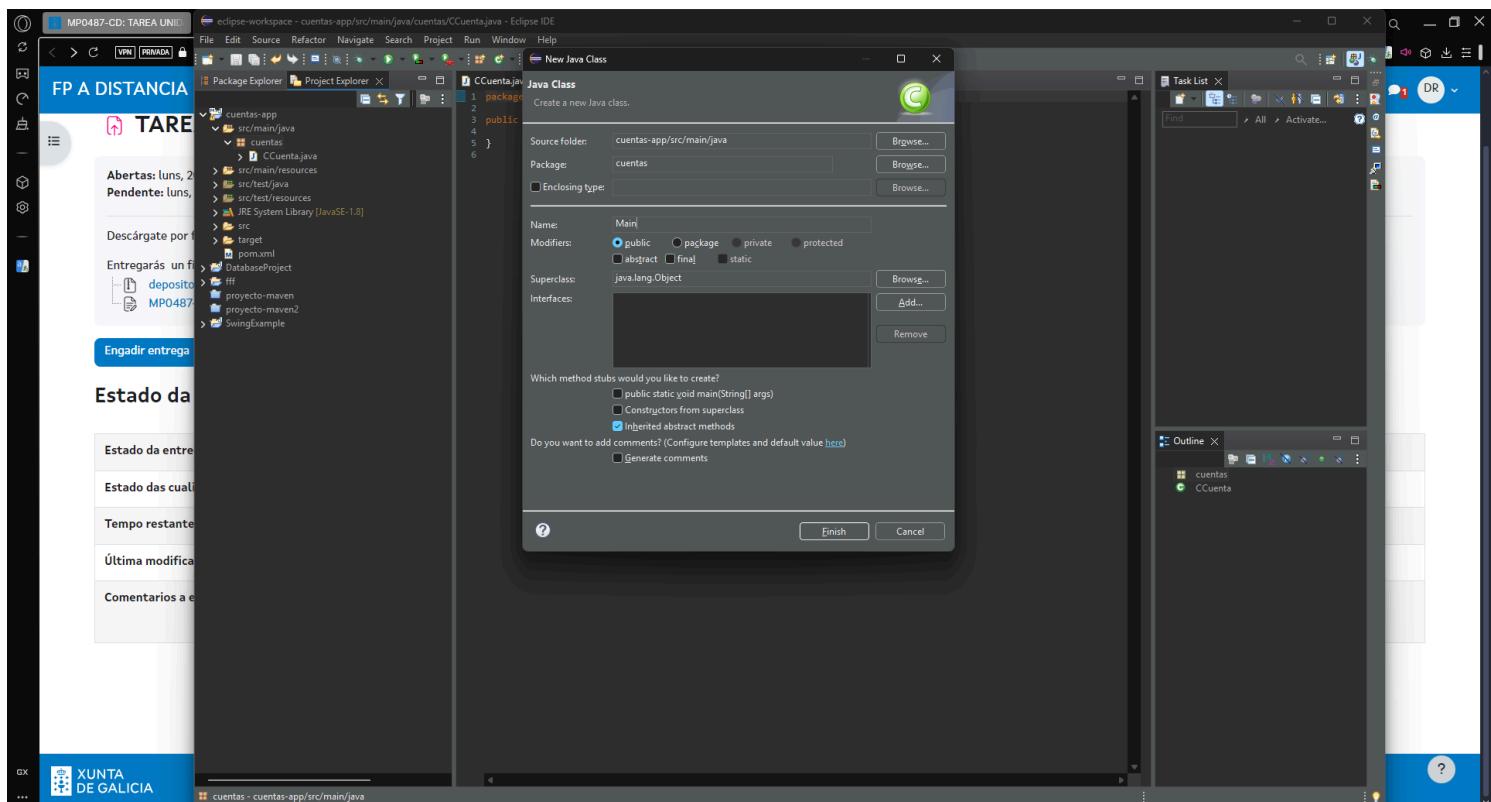
    public String getCuenta() { return cuenta; }
    public void setCuenta(String cuenta) { this.cuenta = cuenta; }

    public double getSaldo() { return saldo; }
    public void setSaldo(double saldo) { this.saldo = saldo; }

    public double getTipoInteres() { return tipoInteres; }
    public void setTipoInteres(double tipoInteres) { this.tipoInteres = tipoInteres; }
}

```

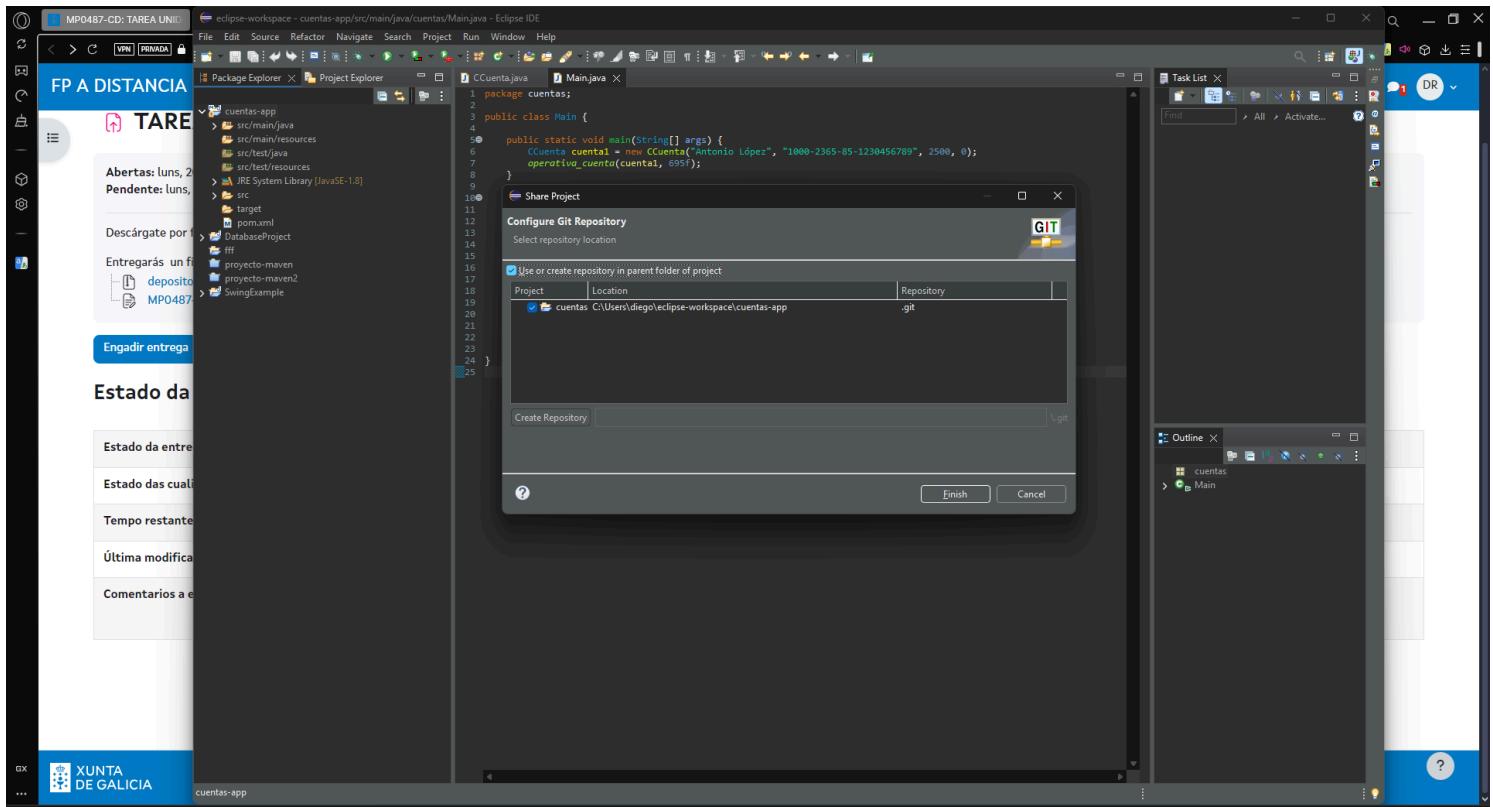
## - Main.java



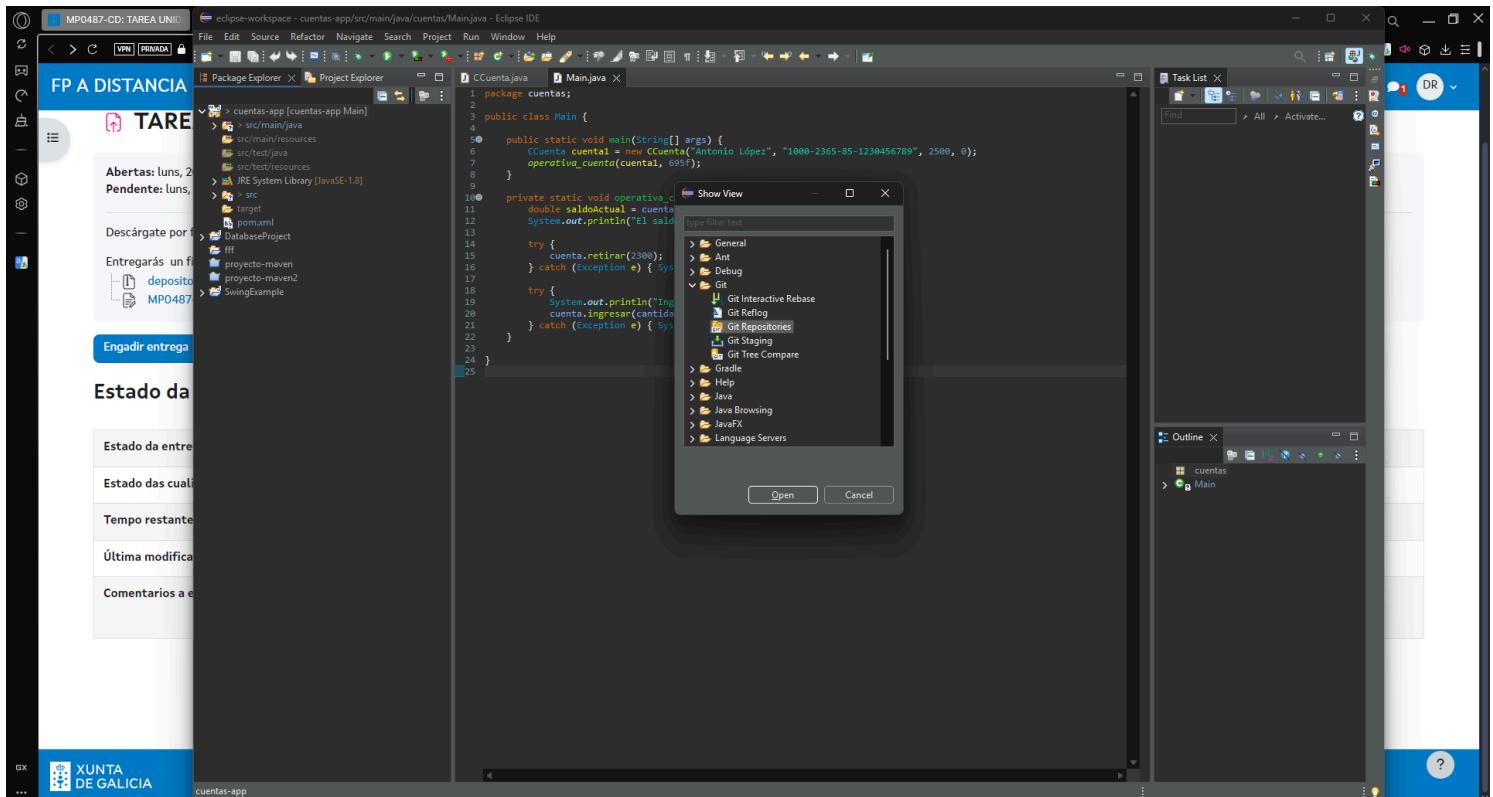
## 2. Git

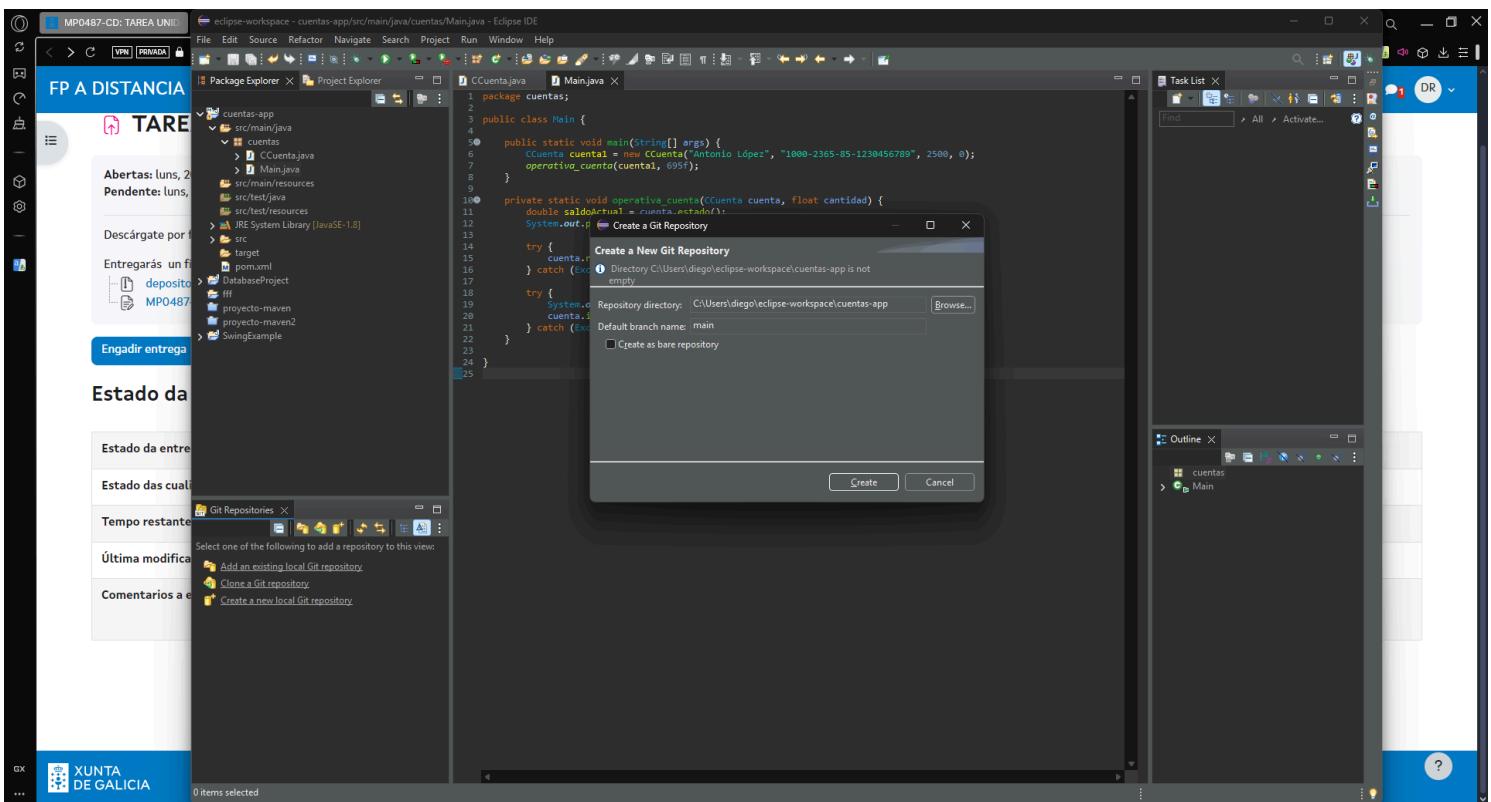
- 2.1. Configurar GIT para el proyecto. Crear un repositorio público en GitHub.
- 2.2. Realizar, al menos, una operación commit. Comentando el resultado de la ejecución.
- 2.3. Mostrar el historial de versiones para el proyecto mediante un comando desde consola.

Iniciar repositorio local (Package Explorer > Team > Share Project).

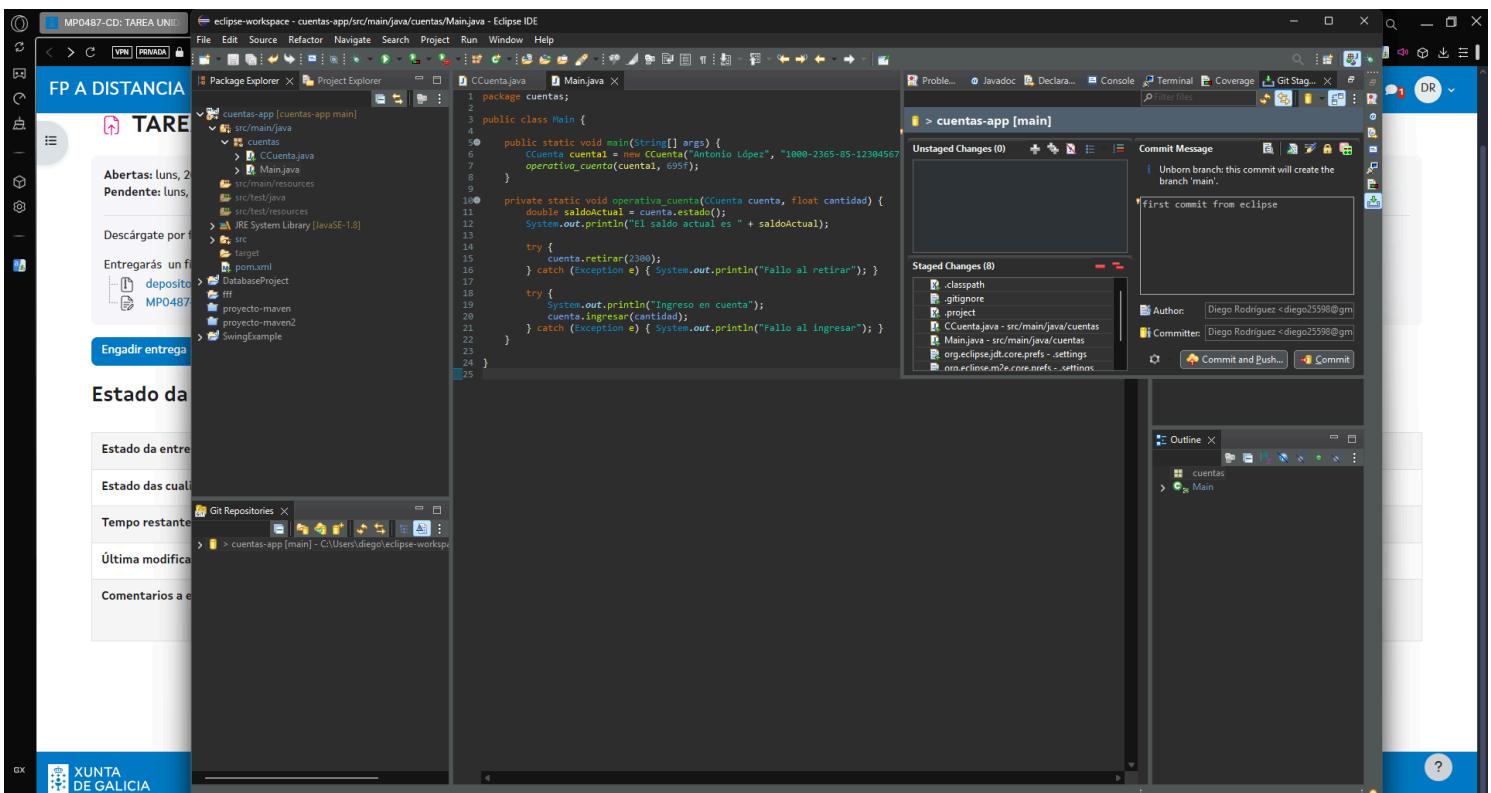


Vincular repositorio local con GitHub desde Eclipse (Window > Show View > Other > Git Repositories > Create a new local Git repository).





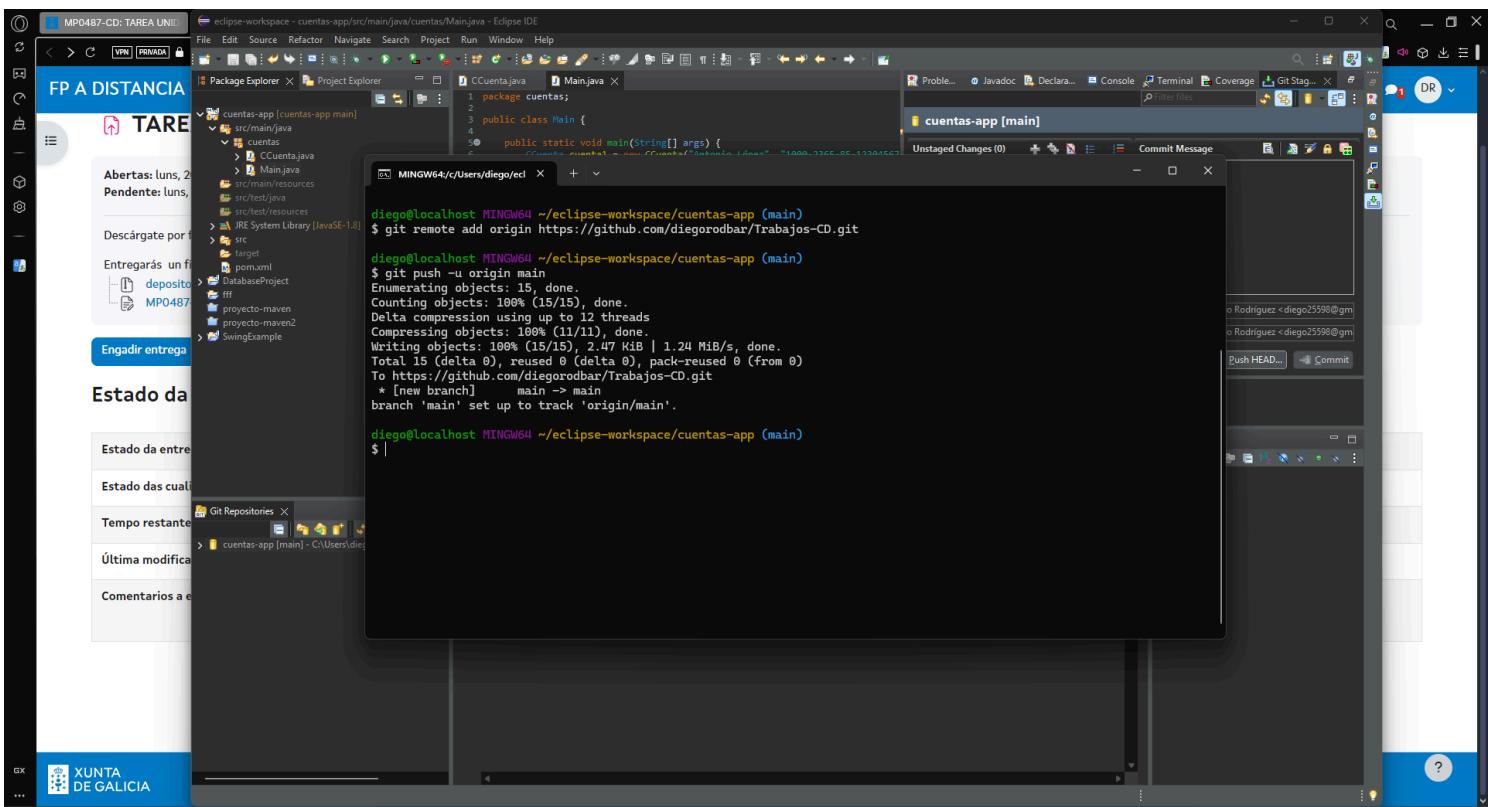
Indicar que queremos añadir al siguiente commit todo el contenido del proyecto (Repositorio Local > Commit ) y hacer el primer commit.



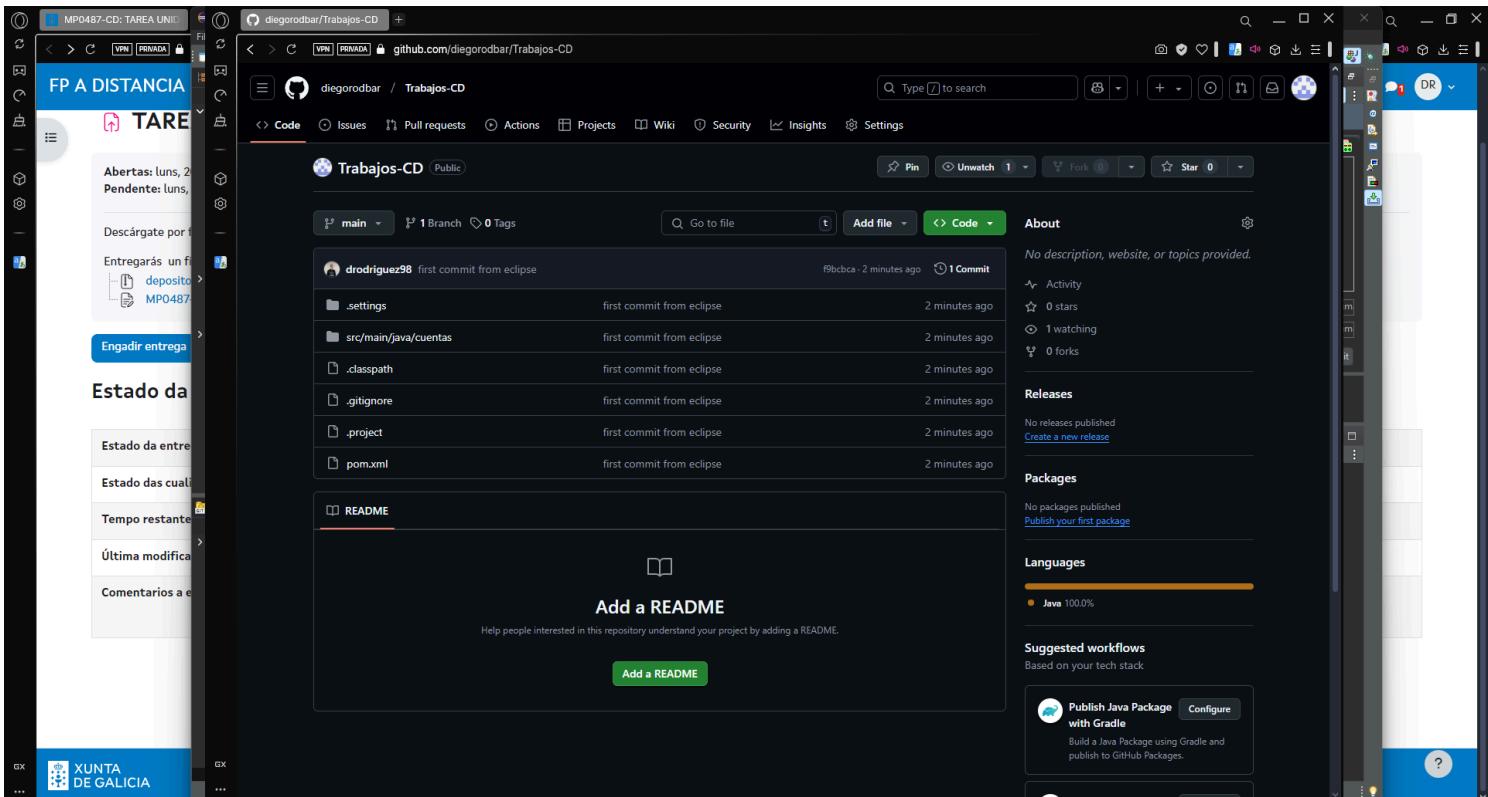
Añadir el repositorio remoto desde Git Bash y subir el proyecto a GitHub. Si crease un nuevo repositorio utilizaría los siguientes comandos igual que en el apartado de Netbeans.

```
git remote add origin https://github.com/diegorodbar/Trabajos-CD.git
```

```
git push -u origin main
```



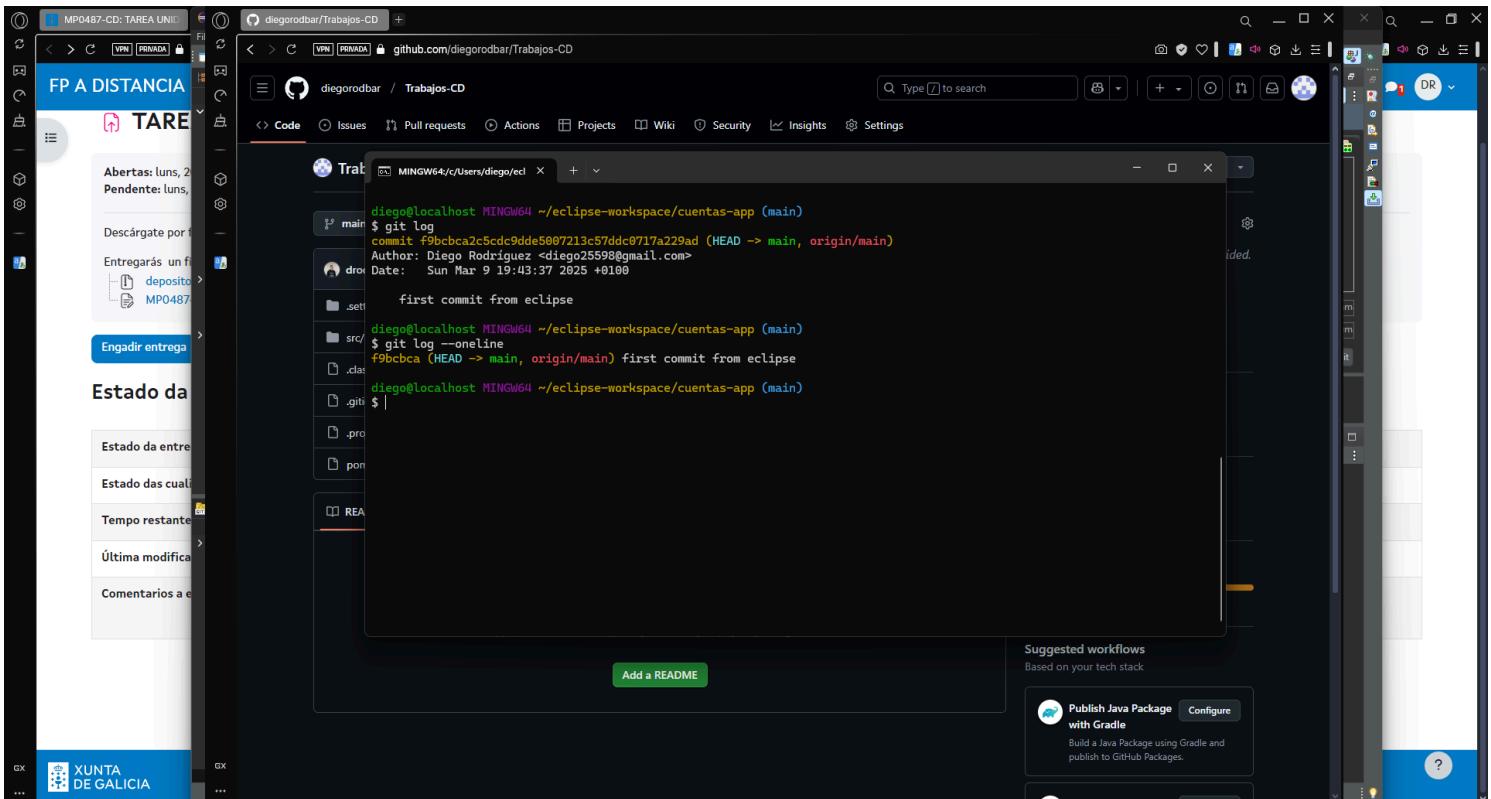
Comprobar que se ha subido el proyecto al repositorio remoto.



Para mostrar el historial de versiones del proyecto desde la consola, utilizar los siguientes comandos en Git Bash o terminal.

git log

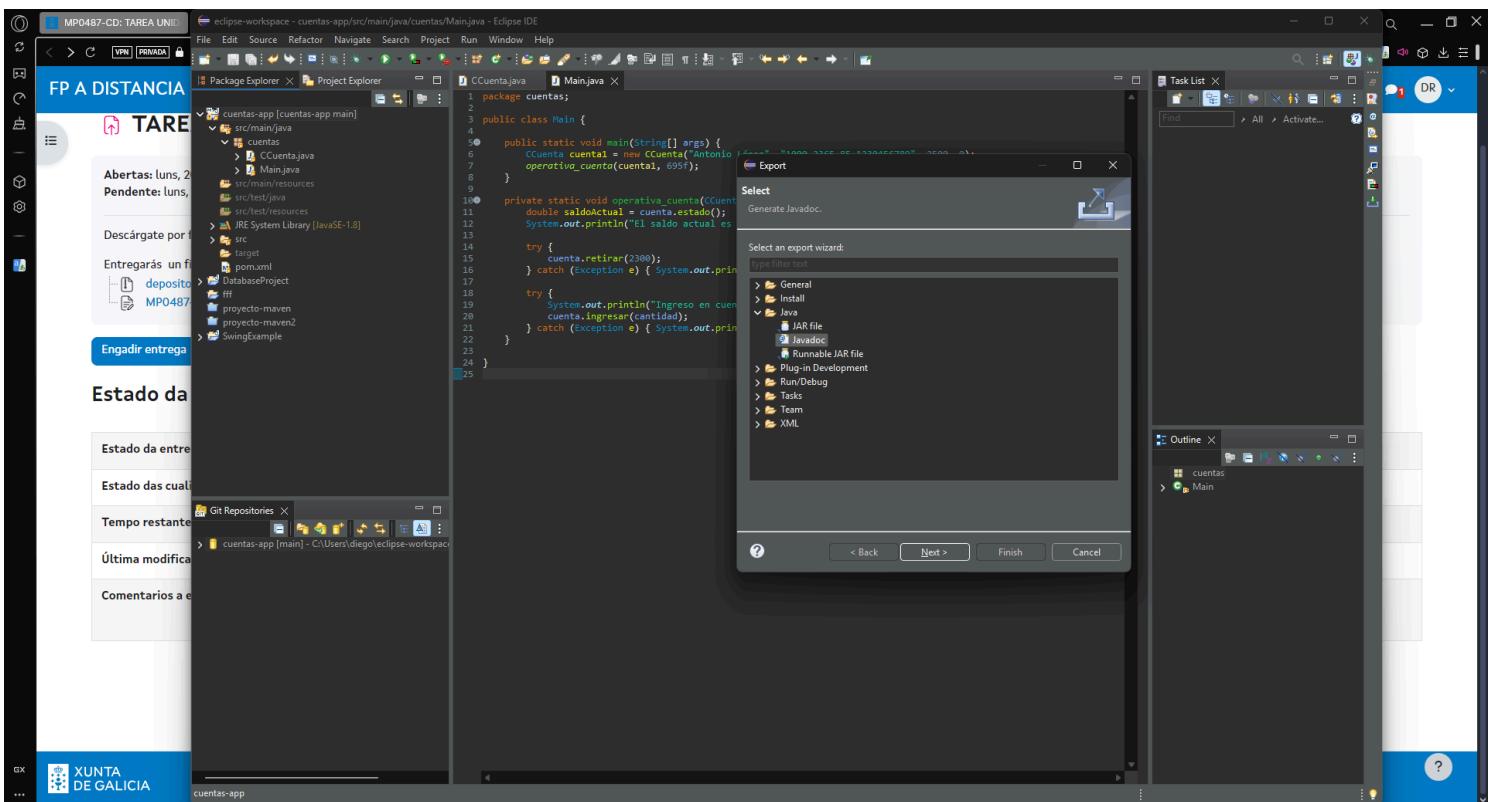
git log --oneline

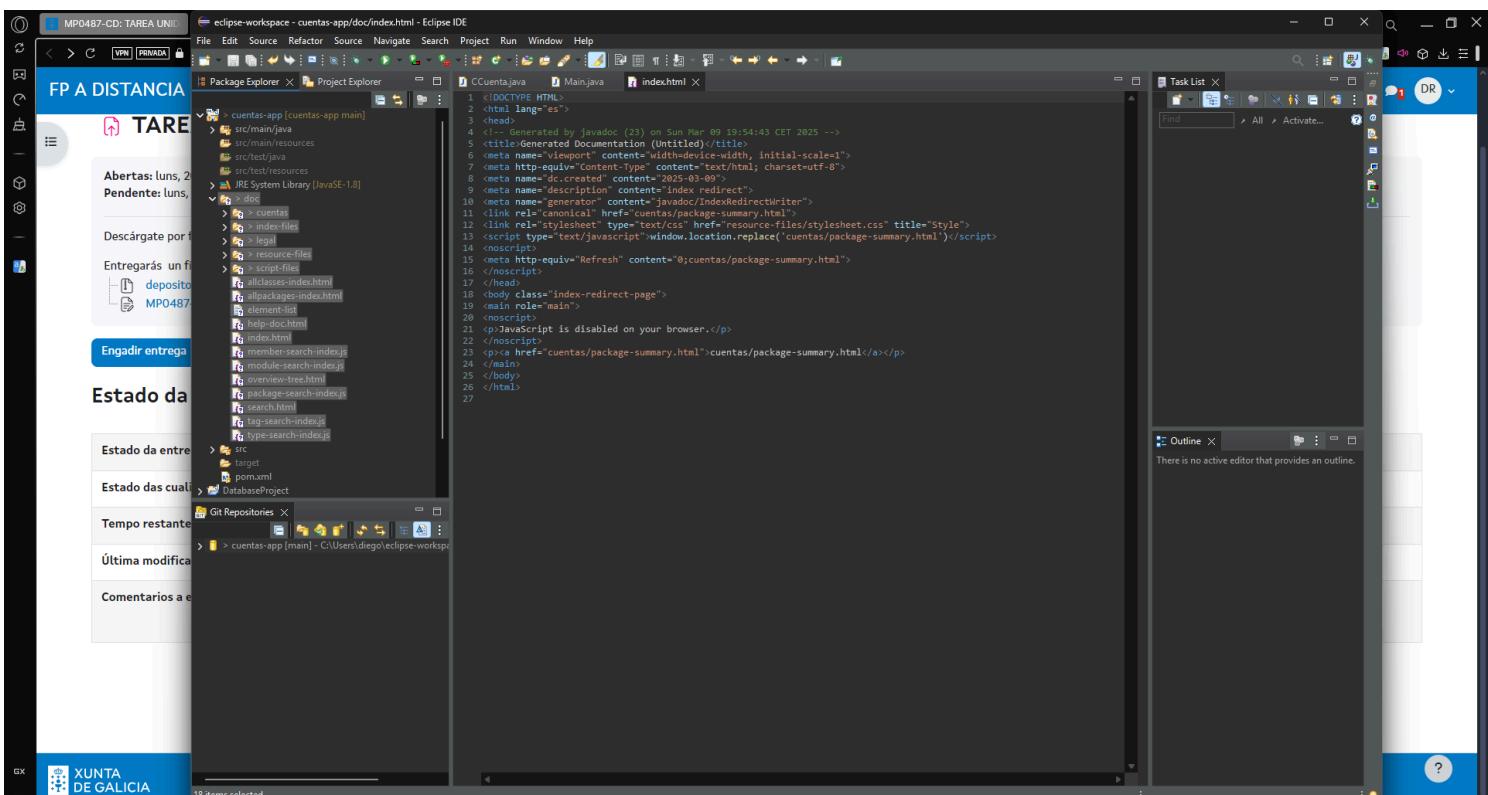
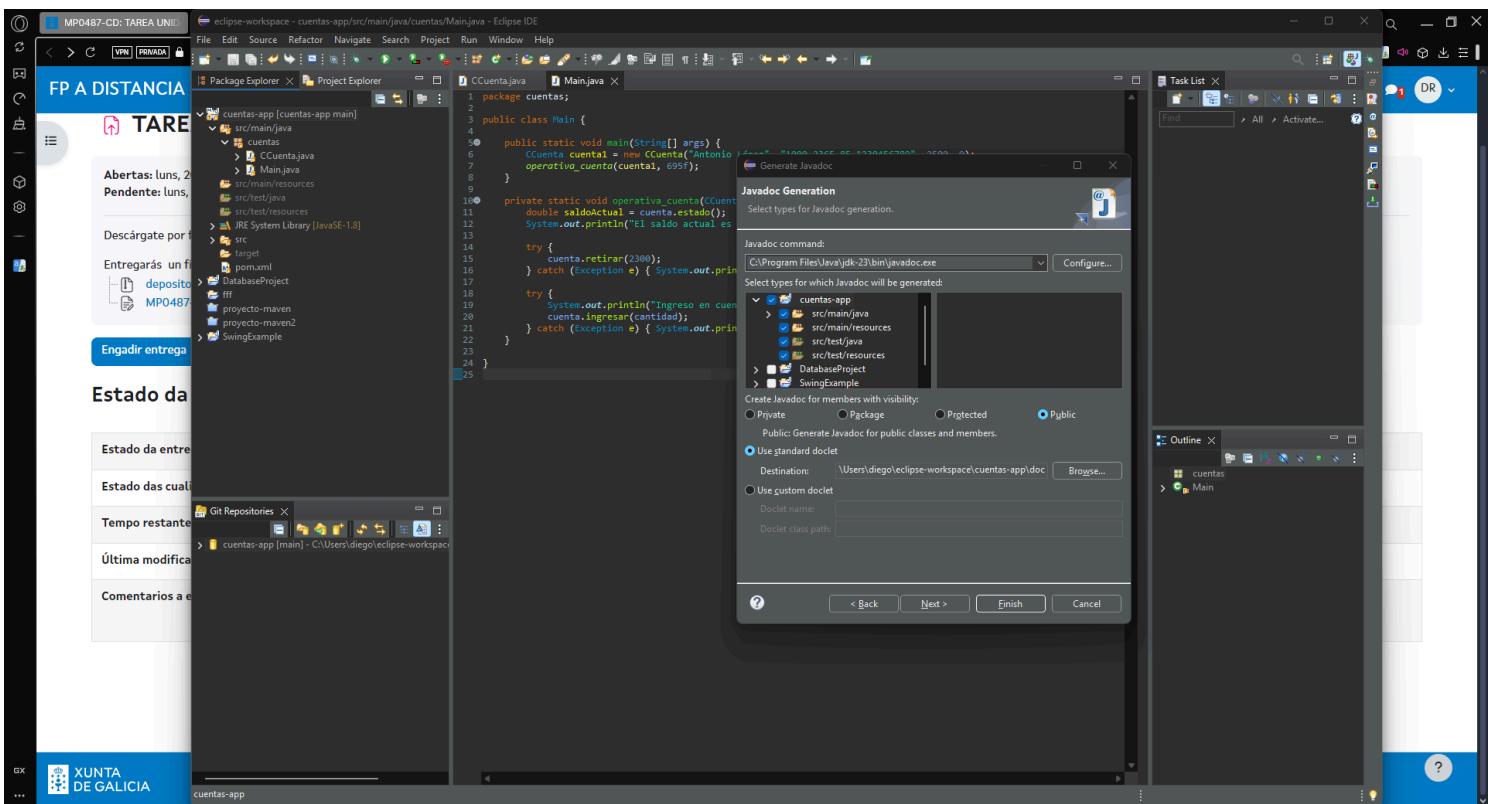


### 3. Javadoc

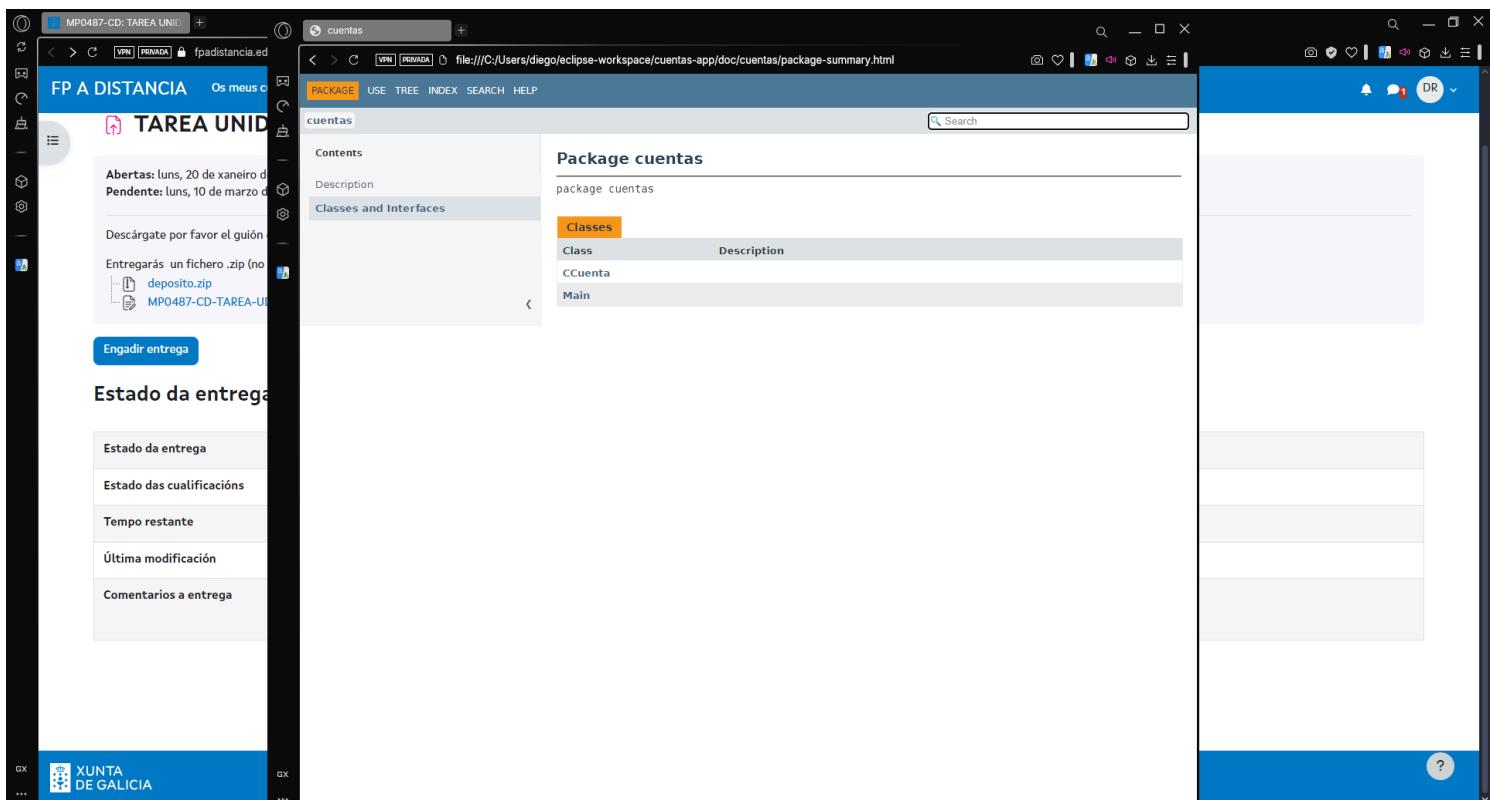
- 3.1. Insertar comentarios JavaDoc en la clase CCuenta.
- 3.2. Generar documentación JavaDoc para todo el proyecto y comprueba que abarca todos los métodos y atributos de la clase CCuenta.

Se pueden generar automáticamente haciendo clic derecho sobre el proyecto > Export > Java > Javadoc, seleccionando todos los archivos.

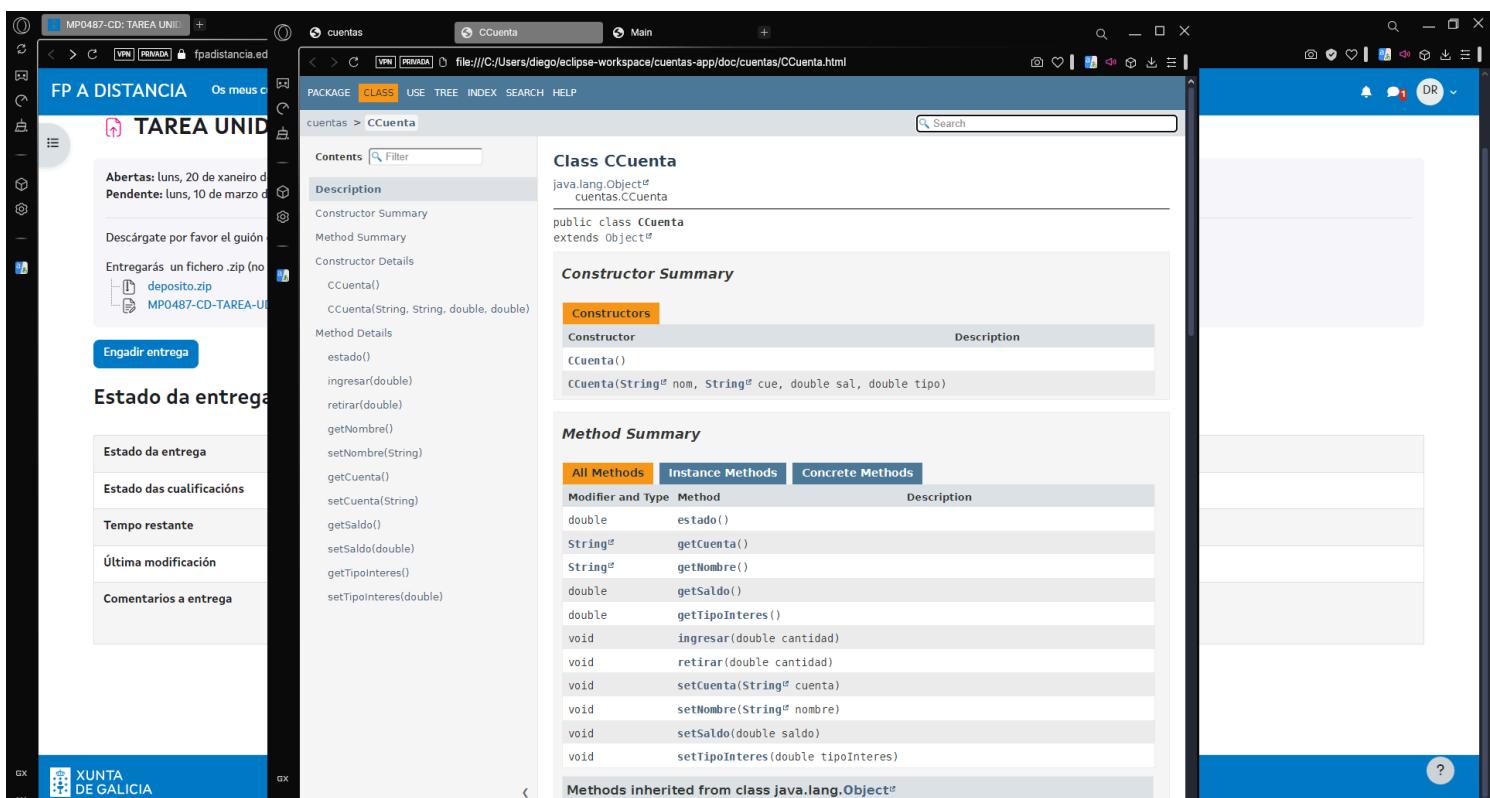




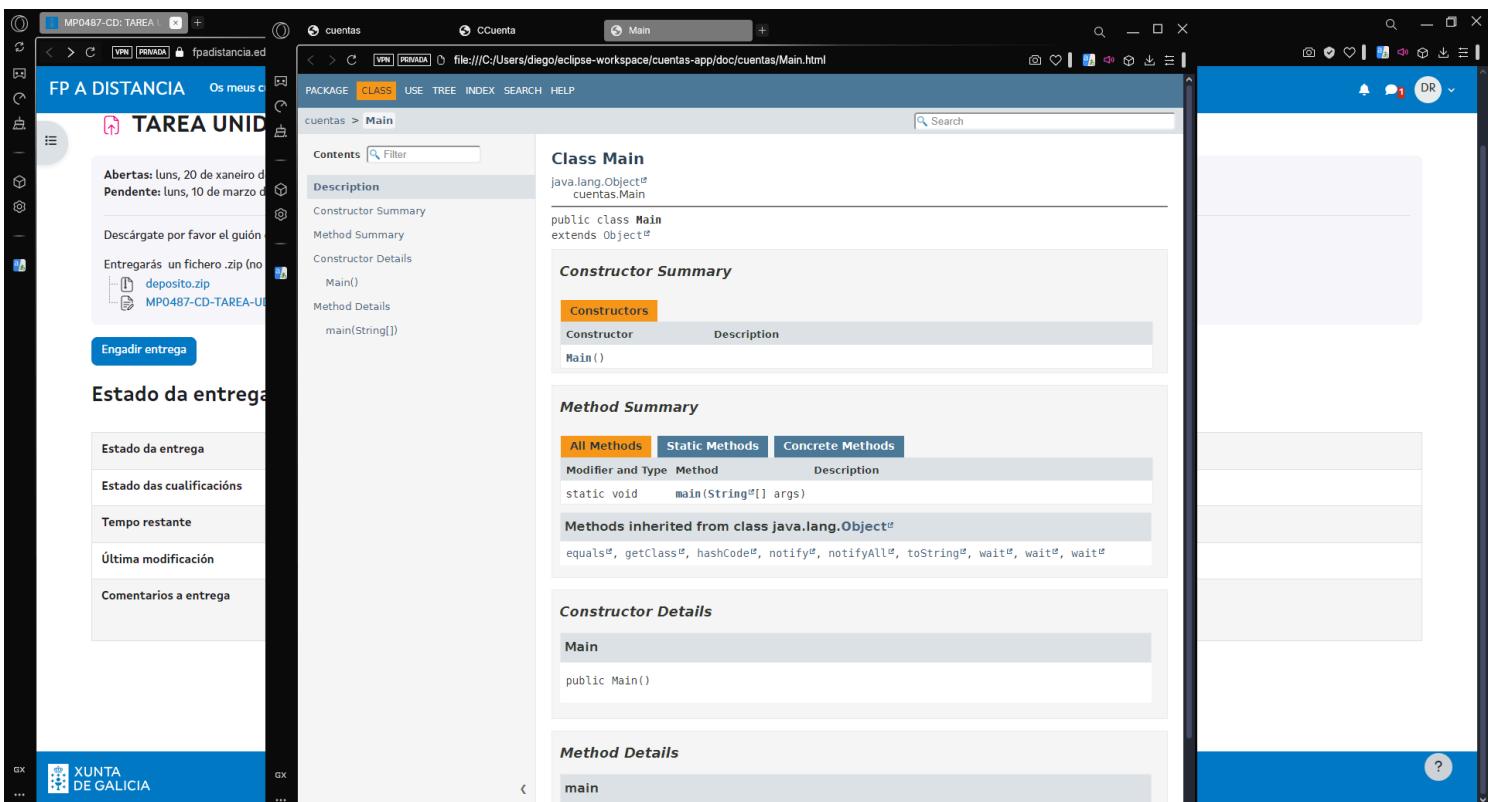
Abrir con el navegador el archivo index.html generado en el nuevo directorio doc y comprobar que la documentación se ha generado correctamente.



The screenshot shows a web browser with two tabs open. The left tab is a task submission form titled 'FP A DISTANCIA' for 'TAREA UNID'. It contains fields for due date ('Abertas: luns, 20 de xaneiro de 2024'), deadline ('Pendente: luns, 10 de marzo de 2024'), instructions ('Descárgate por favor el guión'), and submission details ('Entregarás un fichero .zip (no más de 10 MB)'). A blue button 'Engadir entrega' is visible. The right tab is an Eclipse help documentation for the package 'cuentas'. It shows the package summary for 'cuentas' and a table of classes, with 'CCuenta' and 'Main' listed. The interface includes a search bar and navigation links like 'PACKAGE', 'USE', 'TREE', 'INDEX', 'SEARCH', and 'HELP'.



This screenshot is similar to the previous one, showing the same task submission form on the left. The right side displays detailed Java class documentation for 'CCuenta' from the 'cuentas' package. The documentation includes a constructor summary, a method summary, and a table of methods with their descriptions. Methods listed include 'estado()', 'ingresar(double)', 'retirar(double)', 'getNombre()', 'setNombre(String)', 'getCuenta()', 'setCuenta(String)', 'getSaldo()', 'setSaldo(double)', 'getTipoInteres()', and 'setTipoInteres(double)'. The interface also includes a search bar and navigation links.



The screenshot shows a dual-monitor setup. On the left monitor, a web browser displays a task submission form for 'FP A DISTANCIA' titled 'TAREA UNIDA'. It includes fields for file uploads ('deposito.zip' and 'MPO487-CD-TAREA-UD05'), a note about zip files, and a button to add an entry. On the right monitor, an Eclipse IDE window is open, showing the Java code for the 'Main' class and its documentation. The code is as follows:

```

public class Main
extends Object

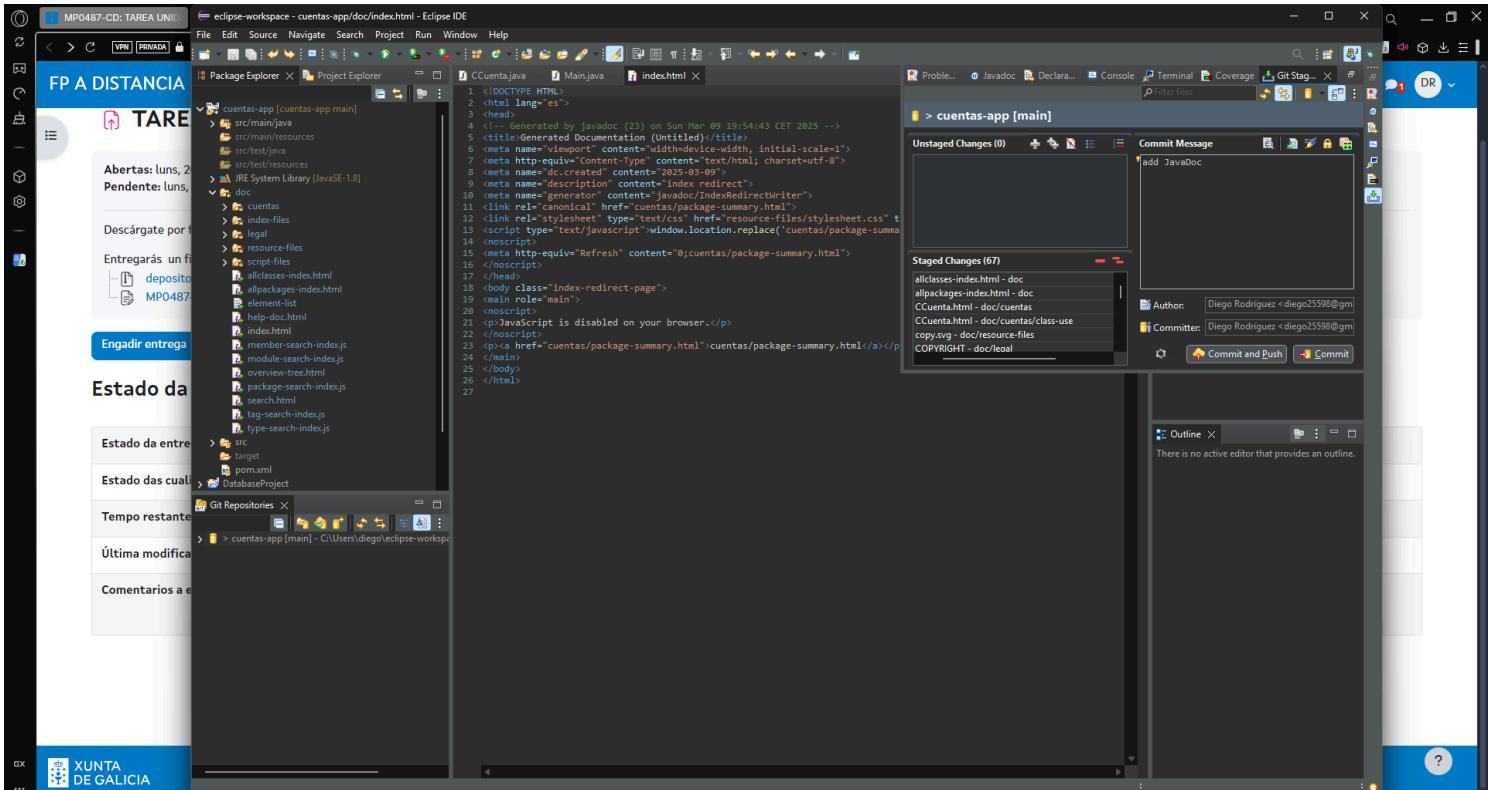
public Main()
{
}

main(String[])

```

The documentation includes sections for 'Constructor Summary' (with one constructor listed) and 'Method Summary' (with methods like main and equals). The 'Method Details' section shows the implementation for main.

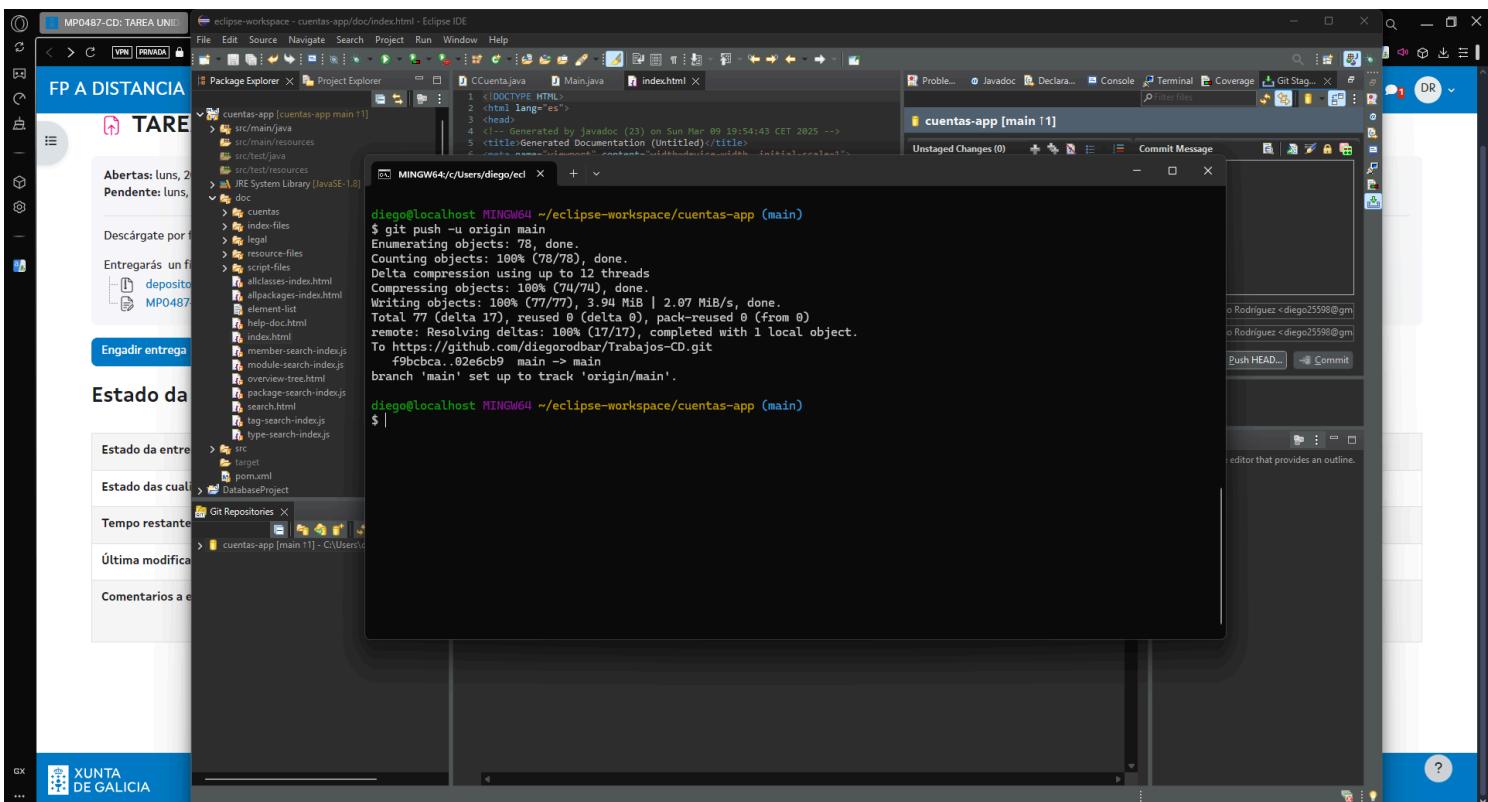
Hago commit para subir también a GitHub la documentación generada con Javadoc.



The screenshot shows the Eclipse IDE interface with the 'cuentas-app' project selected in the Package Explorer. The 'Git Repositories' view shows a repository named 'cuentas-app [main]'. In the center, the 'index.html' file is open, displaying the generated Java documentation. On the right, the 'Commit Message' editor is open with the message 'add JavaDoc'. The 'Staged Changes' and 'Unstaged Changes' panes show the files being committed. The commit message also includes author and committer details: 'Author: Diego Rodriguez <diego25598@gmail.com>' and 'Committer: Diego Rodriguez <diego25598@gmail.com>'.

Desde Git Bash vuelvo a ejecutar el siguiente comando.

```
git push -u origin main
```



El contenido del repositorio remoto en GitHub se actualiza correctamente:

