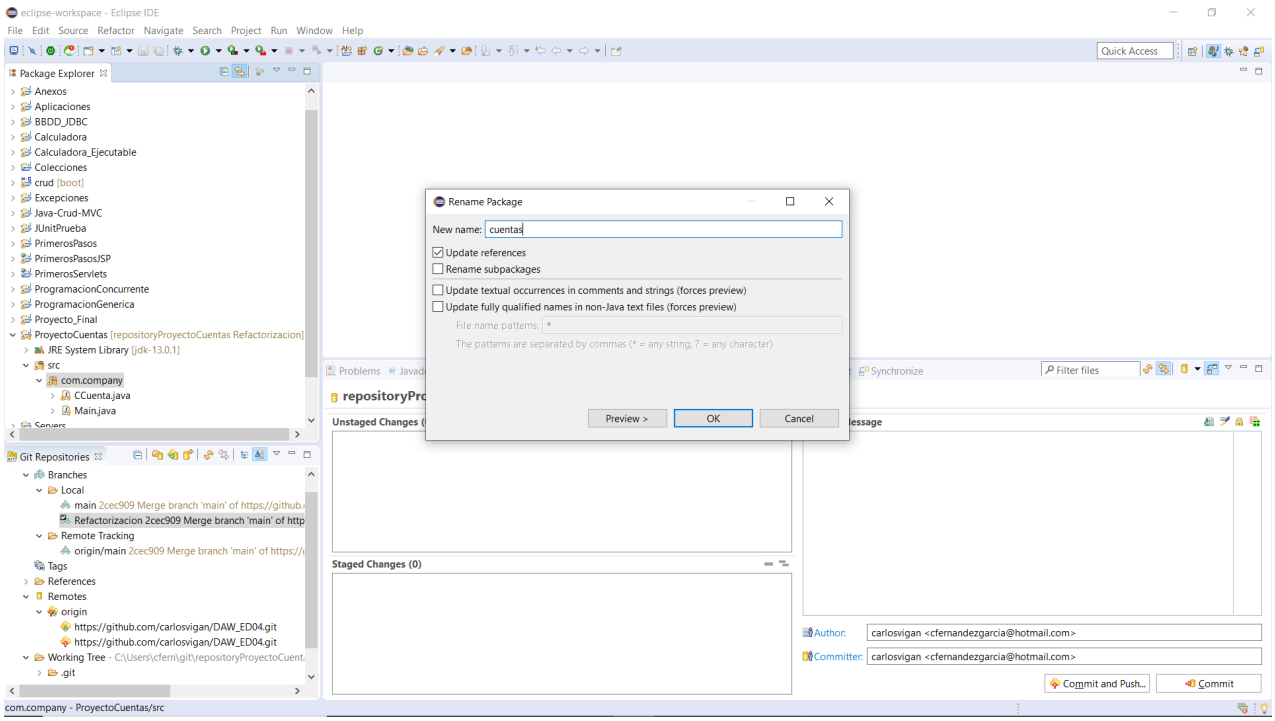
Práctica Final UT4: Optimización y Documentación.

En el proyecto Java que se adjunta, hay definida una Clase llamada *CCuenta*, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con un Clase *Main*, donde se hace uso de la clase descrita.

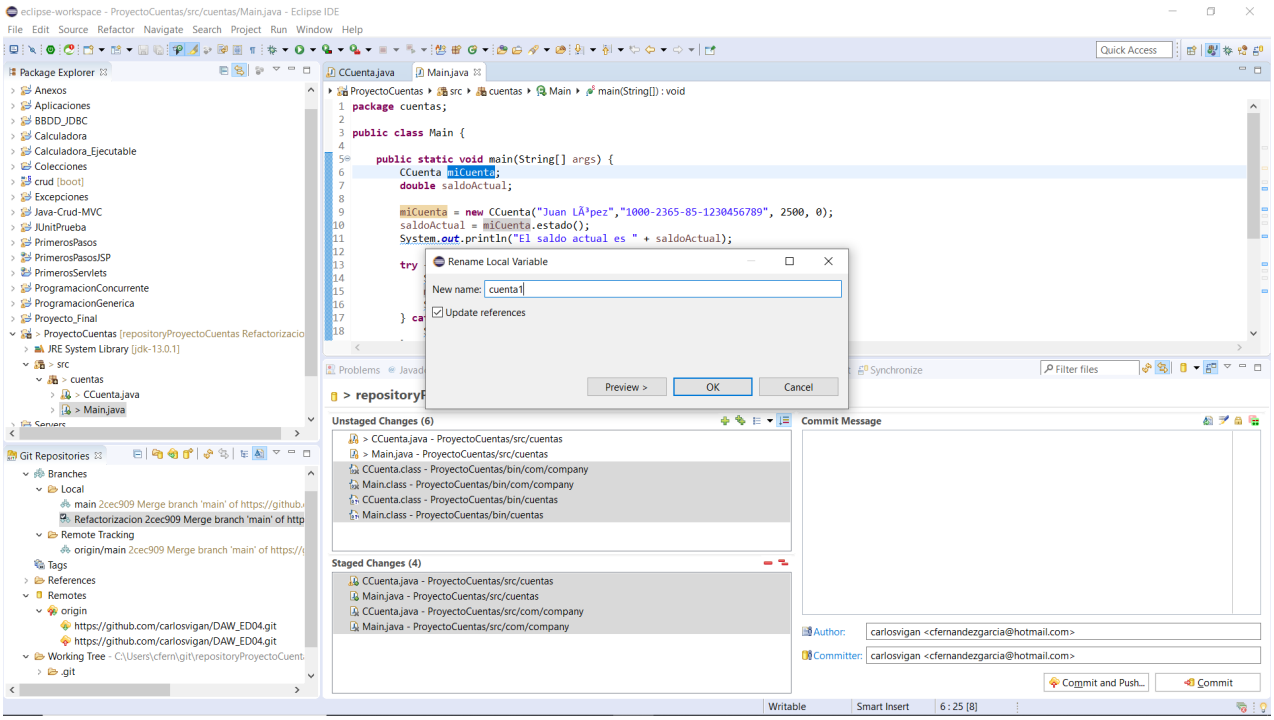
Basándonos en ese proyecto, vamos a realizar las siguientes actividades.

REFACTORIZACIÓN

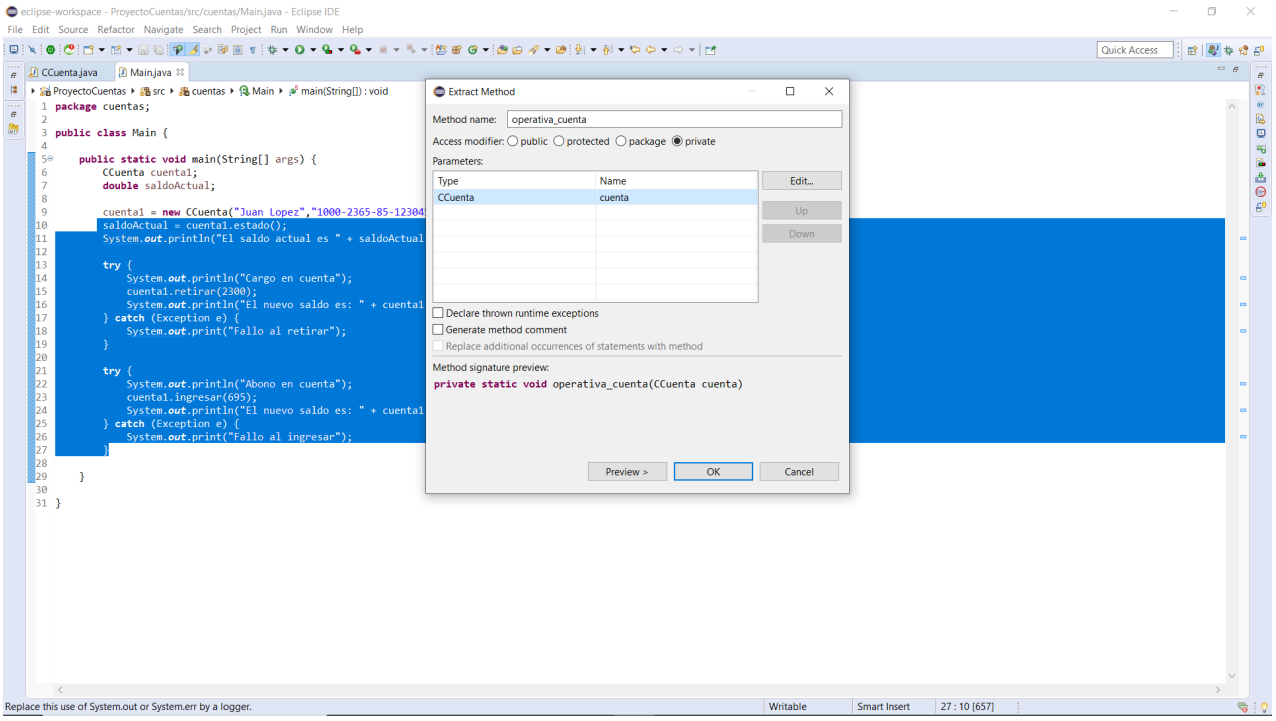
1. Las clases deberán formar parte del paquete cuentas.



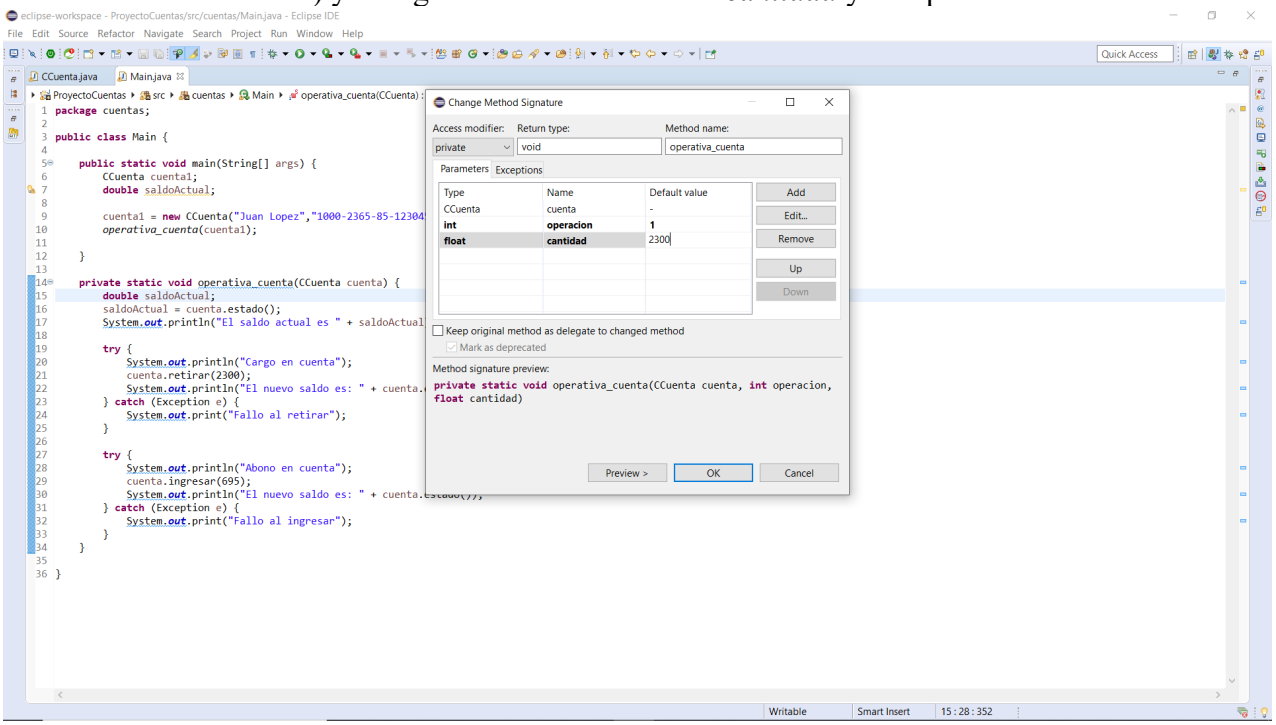
2. **(1 pto)** Cambiar el nombre de la variable "*miCuenta*" por "*cuenta1*".



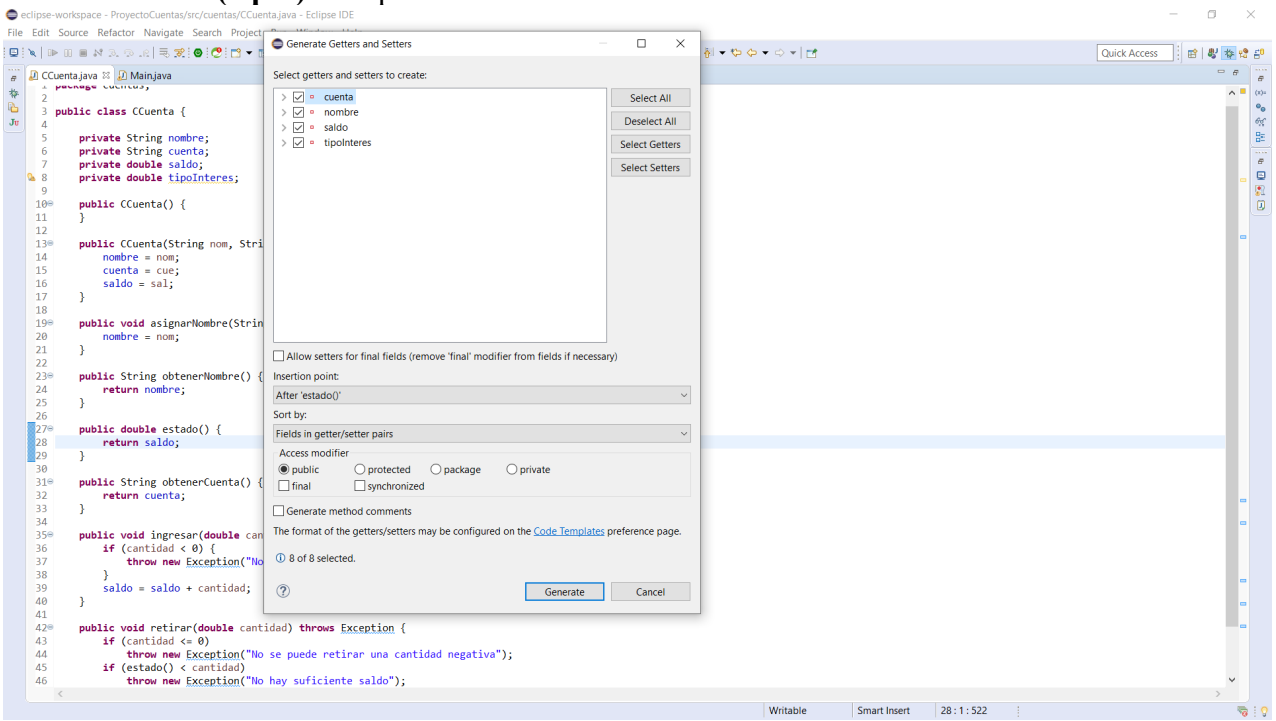
3. **(1 pto)** Introducir el método *operativa_cuenta*, que englobe las sentencias de la clase Main que operan con el objeto *cuenta1*.



4. **(1 pto)** Añadir dos nuevos parámetros al método *operativa_cuenta*, el primero tendrá el nombre *operacion* y de tipo entero (nos indicará el tipo de operación a realizar) y el segundo tendrá el nombre *cantidad* y de tipo float.



5. **(1 pto)** Encapsular los atributos de la clase *CCuenta*.



GIT

1. **(0.5 ptos)** Configurar GIT para el proyecto. Crear un repositorio público en GitHub para este proyecto llamado DAW_ED04.
2. **(0.5 ptos)** Sube el código a una rama que tengas como principal.
3. **(1 pto)** Crea una rama a parte para realizar los cambios indicados en el apartado anterior (Refactorización). Realiza, al menos, una operación *commit*. Comentando el resultado de la ejecución y/o cambios.
4. **(1 pto)** Luego, integra los cambios desde la rama secundaria a la rama principal realizando las operaciones pertinentes.
5. **(1 pto)** Realiza los pasos 3 y 4 para el siguiente apartado (JavaDoc).

JAVADOC

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

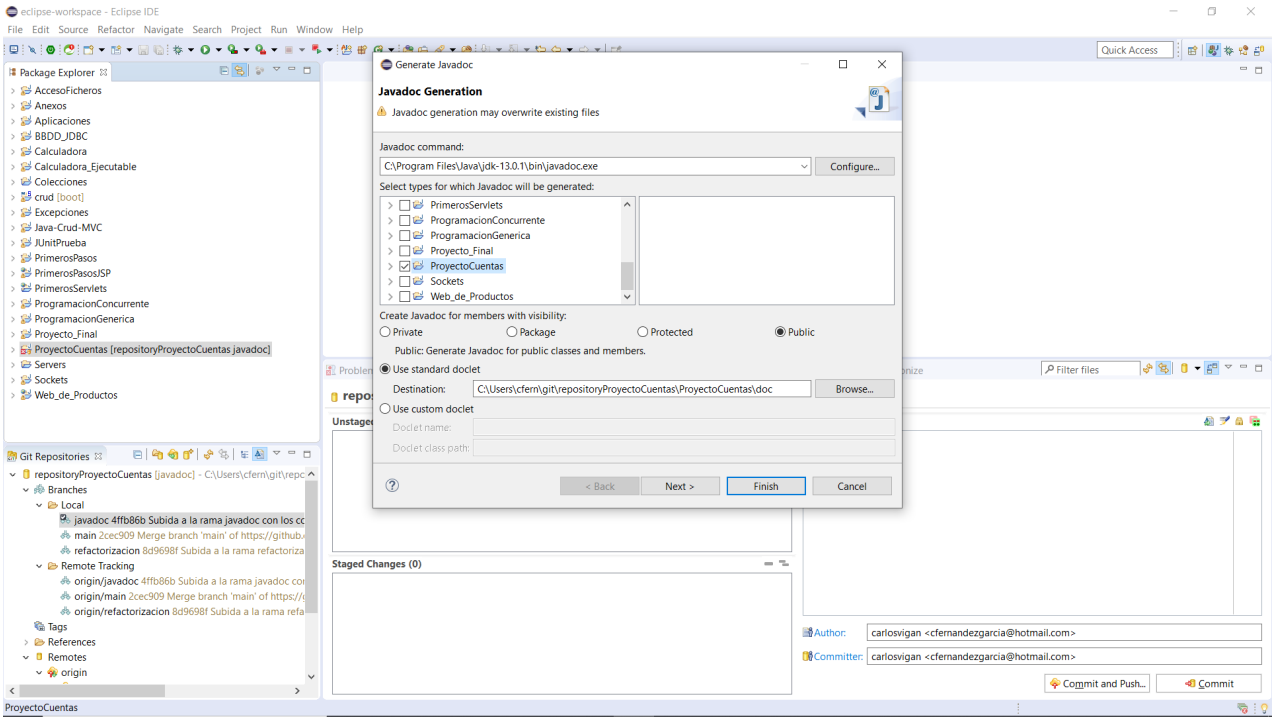
```
package cuentas;
/**

 * Esta clase contiene el método main y lanza las diferentes operativas que se desean
realizar sobre la cuenta.

 * @author: Carlos Fernández García

 * @version: 1.0
 */
public class Main {
       /**
     * Metodo main: Primer método que se ejecuta al lanzar la aplicación. Desde aqui
vamos a llamar al método encargado de realizar las difentes operativas
     * @param args Matriz de tipo String que recoge los valores que introduzcas a la
hora de ejecutar tu aplicación.
     */
    public static void main(String[] args) {
        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta("Juan Lopez","1000-2365-85-1230456789", 2500, 0);
        operativa_cuenta(cuenta1, 1, 2300);
        operativa_cuenta(cuenta1, 2, 695);

    }
       /**
     * Metodo operativa_cuenta: Metodo que nos permite realizar las operativas de
retirar o ingresar dinero sobre la cuenta.
     * @param cuenta variable encargada de recibir el objeto instanciado de tipo
CCuenta.
     * @param operacion variable que nos permite indicar el tipo de operación que
vamos a realizar. 1 es retirar y 2 ingresar.
     * @param cantidad variable que recibe la cantidad de dinero a ingresar o retirar.
     */
      private static void operativa_cuenta(CCuenta cuenta, int operacion, float
cantidad) {
             double saldoActual;
             saldoActual = cuenta.getSaldo();
        System.out.println("El saldo actual es " + saldoActual);
        if (operacion==1)
        {
              try {
                  System.out.println("Cargo en cuenta");
                  cuenta.retirar(cantidad);
```

```
                System.out.println("El nuevo saldo es: " + cuenta.getSaldo());
            } catch (Exception e) {
                System.out.print("Fallo al retirar");
            }
           }
        if(operacion==2)
        {
              try {
                System.out.println("Abono en cuenta");
                cuenta.ingresar(cantidad);
                System.out.println("El nuevo saldo es: " + cuenta.getSaldo());
            } catch (Exception e) {
                System.out.print("Fallo al ingresar");
            }
        }
      }

}
```

```
package cuentas;
/**

 * Esta clase define el objeto CCuenta con sus correspondientes atributos y métodos

 * @author: Carlos Fernández García

 * @version: 1.0
 */
public class CCuenta {
      //Atributos de la clase
    private String nombre;
    private String cuenta;
    private double saldo;
    private double tipoInteres;
    /**

     * Constructor para la clase CCuenta
     */
    public CCuenta() {
    }
    /**

     * Constructor de la clase CCuenta
     * @param nom El parámetro nom define el titular de la cuenta
       * @param cue El parámetro cue define el número de cuenta
       * @param sal El parámetro saldo define el dinero que tiene la cuenta
       * @param sal El parámetro tipo define el tipo de operacion que vas a realizar
sobre la cuenta
     */
    public CCuenta(String nom, String cue, double sal, double tipo) {
        nombre = nom;
        cuenta = cue;
        saldo = sal;
    }


    /**
     * Getter.
     * @return nombre: nombre del titular de la cuenta.
     */
      public String getNombre() {
            return nombre;
      }
      /**
     * Setter.
     * Asigna el nombre del titular de la cuenta al atributo nombre.
```

```
     */
      public void setNombre(String nombre) {
            this.nombre = nombre;
      }
       /**
     * Getter.
     * @return cuenta: número de la cuenta.
     */
      public String getCuenta() {
            return cuenta;
      }
      /**
     * Setter.
     * Asigna el número de cuenta de la cuenta al atributo cuenta.
     */
      public void setCuenta(String cuenta) {
            this.cuenta = cuenta;
      }
      /**
     * Getter.
     * @return saldo: saldo de la cuenta.
     */
      public double getSaldo() {
            return saldo;
      }
      /**
     * Setter.
     * Asigna el saldo  de la cuenta al atributo saldo.
     */
      public void setSaldo(double saldo) {
            this.saldo = saldo;
      }
      /**
     * Getter.
     * @return tipoInteres: tipo de interés.
     */
      public double getTipoInteres() {
            return tipoInteres;
      }
      /**
     * Setter.
     * Asigna el tipo de interes al atributo tipoInteres.
     */
      public void setTipoInteres(double tipoInteres) {
            this.tipoInteres = tipoInteres;
      }
      /**
     * Devuelve el saldo de la cuenta una vez ingresado el dinero ingresado y en el
caso de intentar ingresar una cantidad negativa lanza una excepción.
     * @param cantidad dinero a ingresar en la cuenta.
     */
    public void ingresar(double cantidad) throws Exception {
        if (cantidad < 0) {
            throw new Exception("No se puede ingresar una cantidad negativa");
        }
        saldo = saldo + cantidad;
    }
    /**
     * Devuelve el saldo de la cuenta una vez retirado el dinero y en el caso de
intentar retirar una cantidad negativa o una cantidad superior al saldo disponible de
la cuenta lanza sus correspondientes excepciónes.
     * @param cantidad dinero a retirar de la cuenta.
     */
    public void retirar(double cantidad) throws Exception {
        if (cantidad <= 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (getSaldo() < cantidad)
            throw new Exception("No hay suficiente saldo");
        saldo = saldo - cantidad;
```

```
    }
}
```



Entrega:

Se tendrán que realizar capturas de pantalla de cómo se han hecho los apartados “Refactorización” y “JavaDoc” y añadirlas a un documento PDF. Dicho documento súbelo a tu repositorio Git.
Sube a la plataforma EVAGD el enlace para poder acceder a tu repositorio público Git.