**DOCUMENTACION EJERCICIOS I & II**

Edgar Núñez Arana

**Tabla de contenido**

[**EJERCICIO I** 2](#_Toc167954321)

[**Modelo (*M)*** 2](#_Toc167954322)

[**Controlador (C)** 3](#_Toc167954323)

[**Vista (V)** 6](#_Toc167954324)

[**Conclusión** 8](#_Toc167954325)

# **EJERCICIO I**

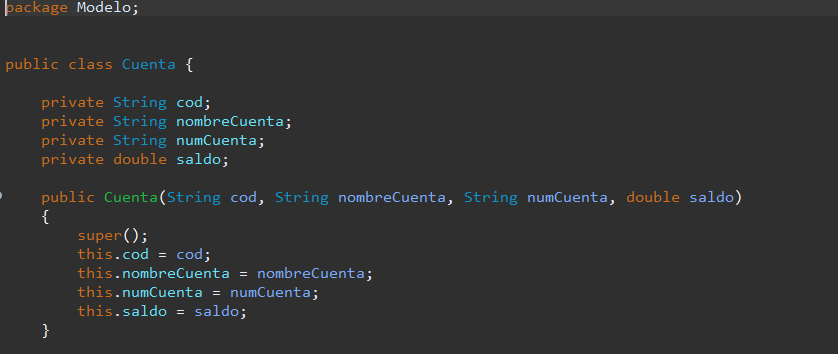
**En este ejercicio haremos una gestión de un banco desde Eclipse donde podremos crear cuentas y actuar sobre ellas, ver sus datos, borrarlas, o modificarlas para ello las reconoceremos con un ID que será el código identificativo de cada cuenta, todo ello estará en el formato *MVC* para el correcto funcionamiento y administración de cada clase del banco.**

## **Modelo (*M******)***

El apartado modelo (M) es el package, como se puede ver en la imagen 1, interno de nuestro banco el cual tiene la clase cuenta.java y es donde se guardara nuestras cuentas en formato .dat.

IMAGEN 1

Además la clase Cuenta tendrá unos atributos que serán los datos a ver de cada cuenta como puede ser Nombre del titular de la cuenta, numero de cuenta, saldo (tipo double) y por último el código identificativo que será el mas importante ya que con el podremos reconocer a que cuenta nos dirigimos. En esta clase Cuenta también tendremos su constructor y los “getter” y “setters”, véase en la imagen 2 (hay algunos getter y setter no todos).



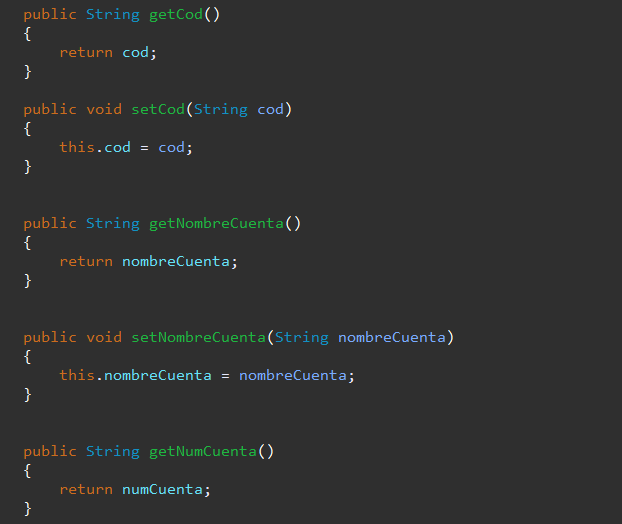


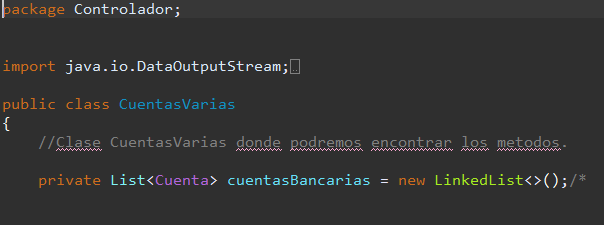
IMAGEN 2

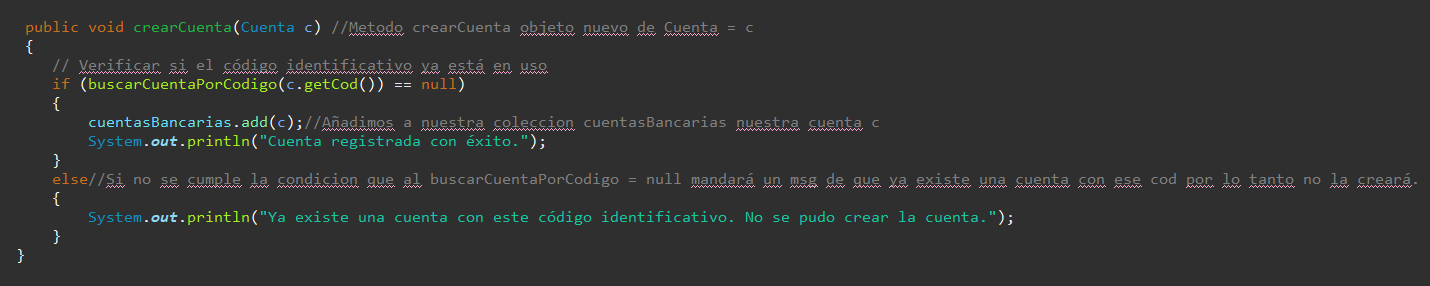
## **Controlador (C)**

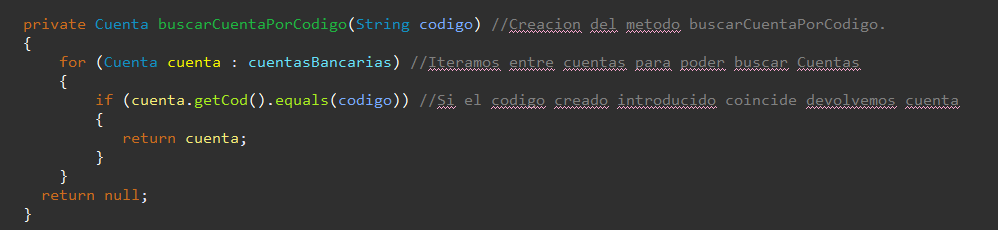
**En el apartado controlador (C) es el package, imagen 3, el cual es el encargado de dar las distintas funciones de nuestro banco**.

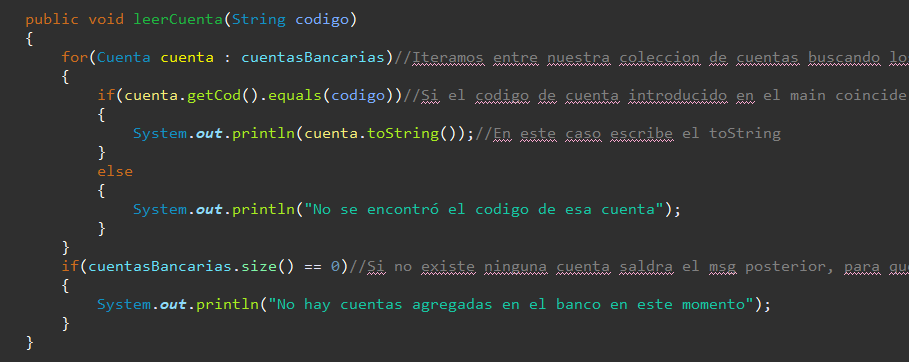
IMAGEN 3

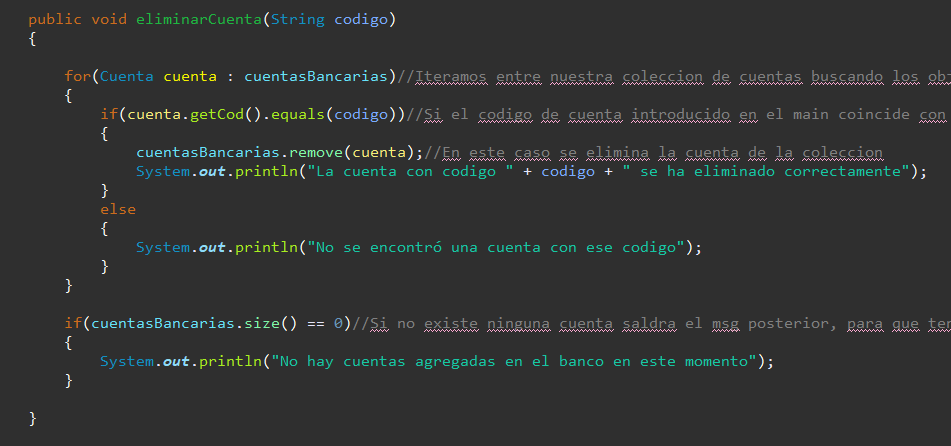
**Para ello crearemos una colección para poder actuar sobre ella.**

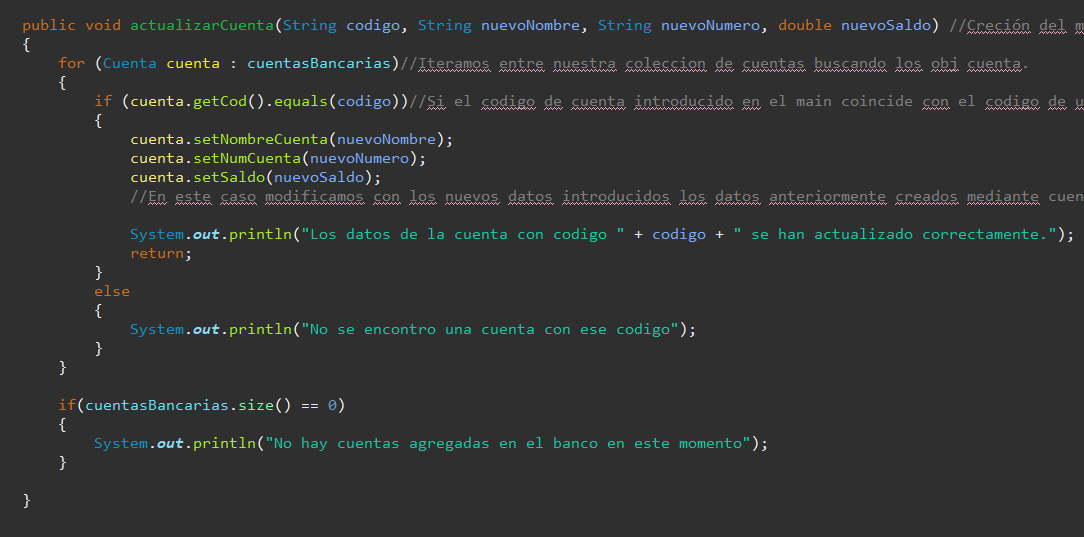












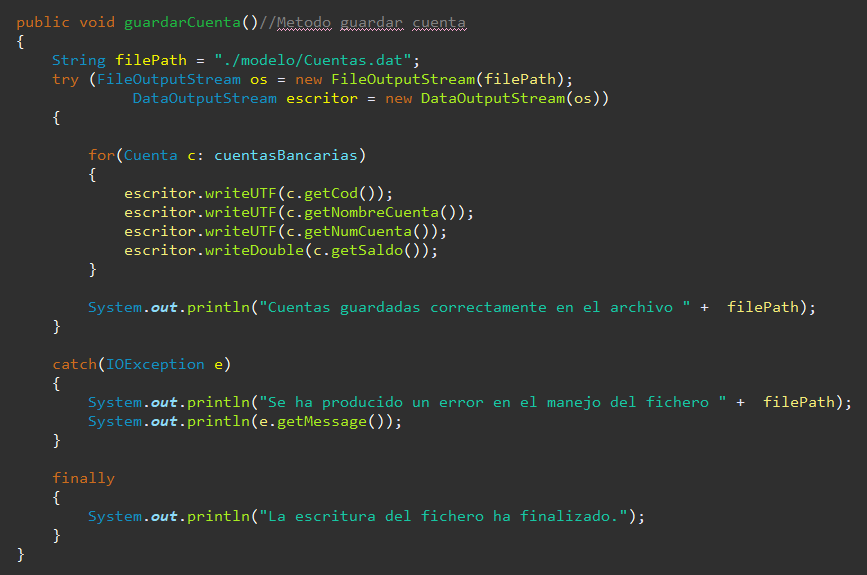
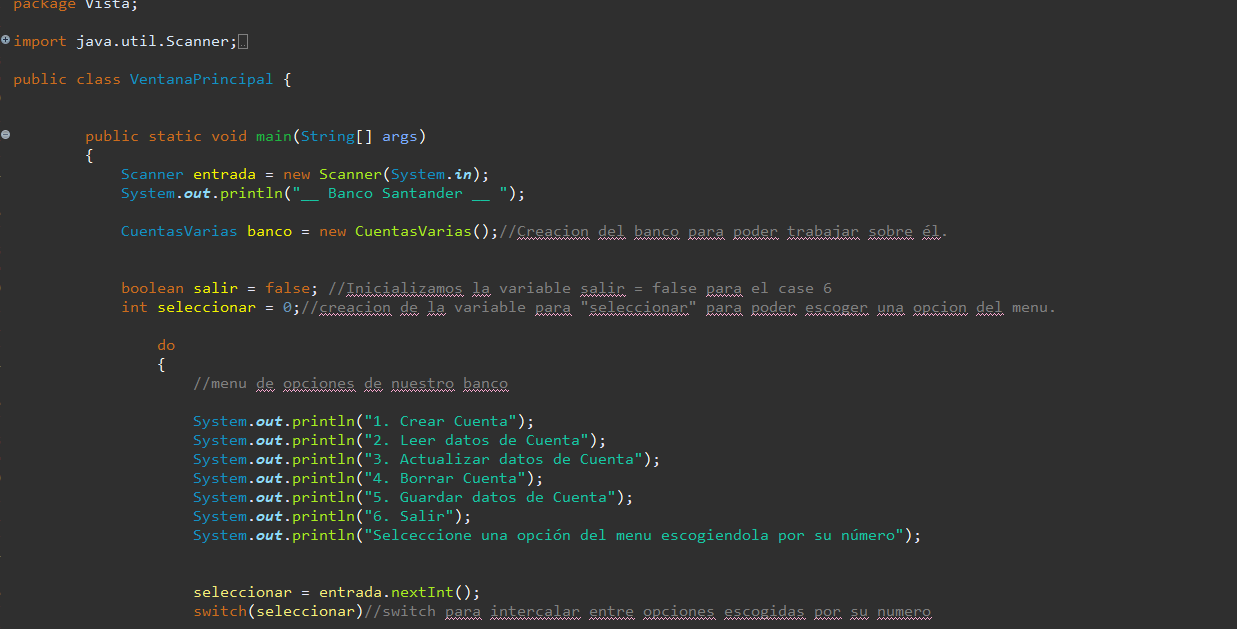


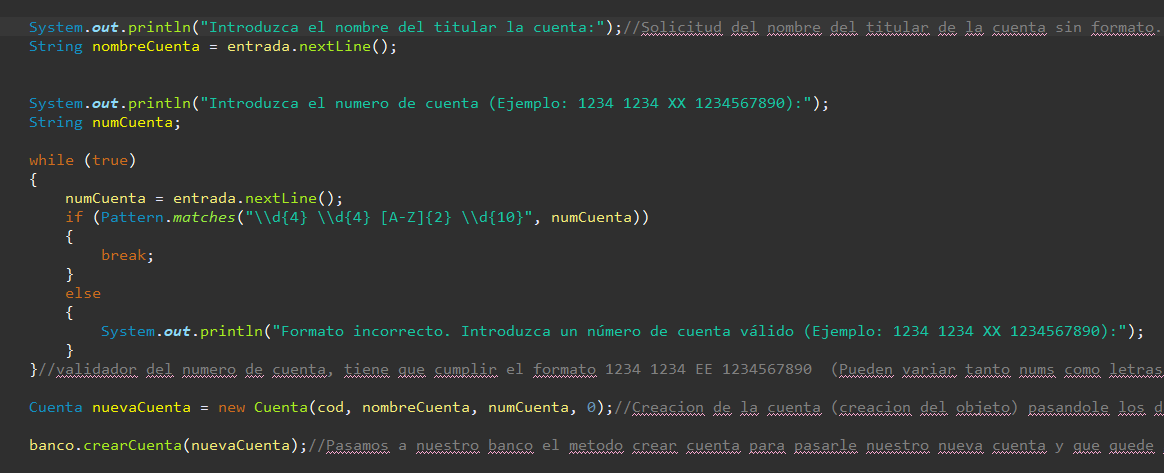
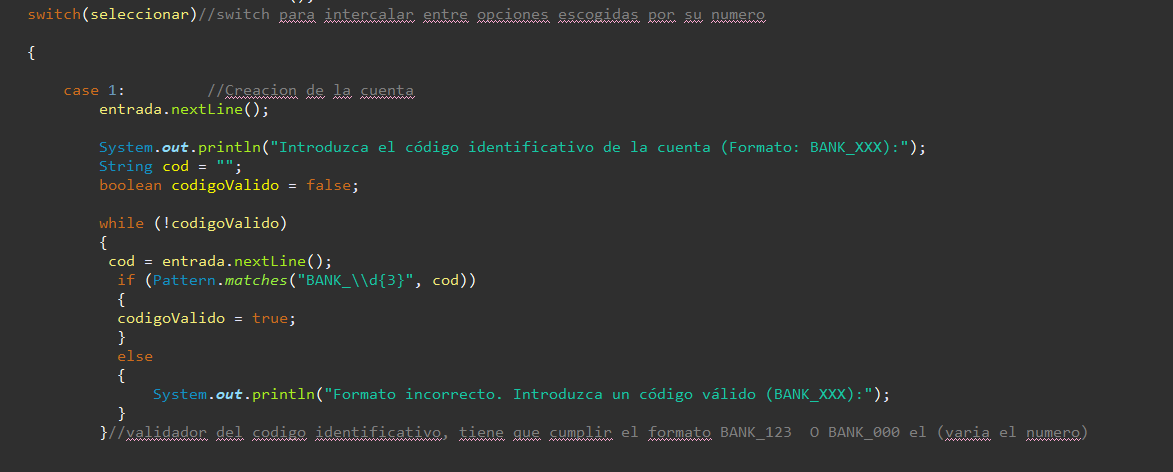
IMAGEN 4

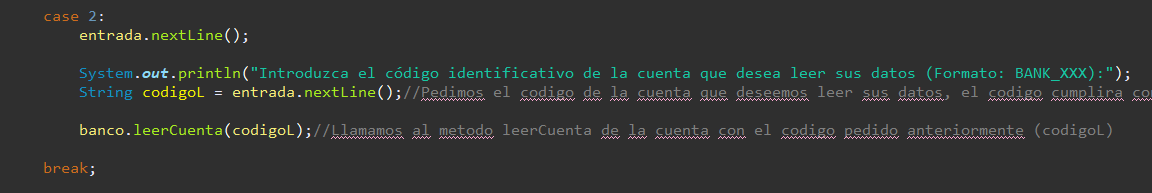
## **Vista (V)**

Por ultimo en el apartado vista, imagen 5,será el encargado de darle el feedback con el cliente donde este último podrá tener acciones con el programa escoger las opciones en el menú y crear borrar modificar cuentas…

Para ello crearemos un banco donde podremos realizar las acciones.







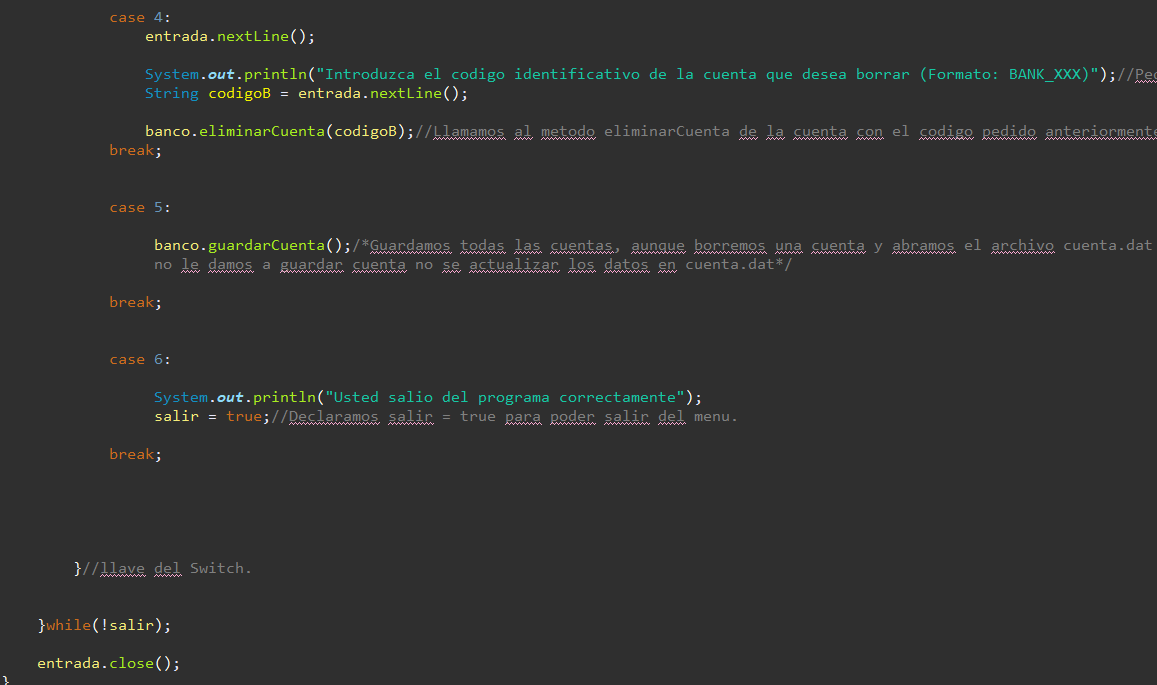
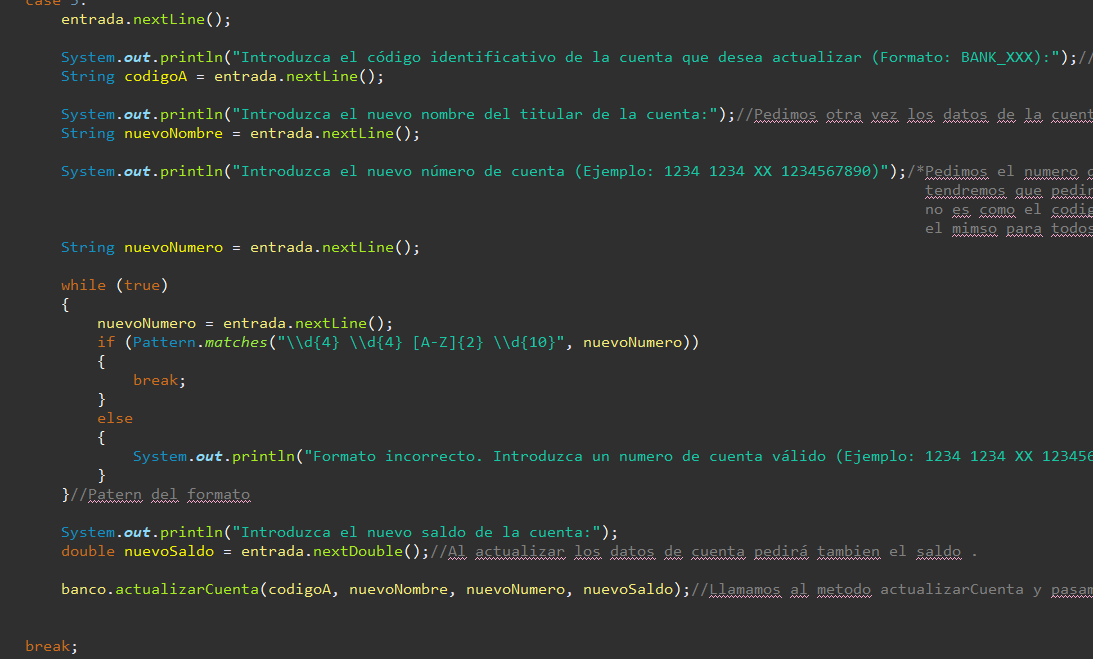


IMAGEN 5

Se puede apreciar que cada case será una opción que se podrá escoger mediante el menú.

## **Conclusión**

En conclusión con esta organización MVC podremos gestionar de una manera óptima y organizada las clases para su correcto funcionamiento de banco.

# **EJERCICIO II**

**En este ejercicio haremos una gestión de un banco desde NeatBeans el funcionamiento es parecido al del ejercicio I pero aquí podremos ver la interfaz gráfica de como se vería la aplicación de nuestro banco, imagen 6.**

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**