Logotipo

Descripción generada automáticamente con confianza media

INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ EN EL ESTADO DE CAMPECHE

INGENIERÍA EN SISTEMAS COMPUTACIONALES

**PRÁCTICA NO. 5.- DESARROLLO DE APLICACIÓN PARA MÓVILES CON FORMULARIOS.**

PARCIAL 2

PROFESOR: JOSE LUIS LIRA TURRIZA

ALUMNO:

PABLO JESUS CHE MARTINEZ 7252

MATERIA: TÓPICOS AVANZADOS DE PROGRAMACIÓN

SEMESTRE: 4 GRADO: 4 GRUPO: A

CICLO ESCOLAR: 20212022P

PRÁCTICA No.5.- Desarrollo de aplicaciones para móviles con formularios.

**- INTRODUCCIÓN**

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Ya hemos visto que existen 2 configuraciones CLDC y CDC, cada una con unas características propias que veremos en profundidad más adelante. Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM.

En esta práctica se pretende implementar los conceptos de formularios implementados a aplicaciones para J2ME que es la versión móvil de la familia de Java.

**-OBJETIVO**

El estudiante podrá crear aplicaciones con interfaces de usuario basado en formularios.

**LUGAR**

AULA

**-SEMANA DE EJECUCIÓN**

Semana DOCE (Parcial 3)

**- MATERIAL Y EQUIPO**

• Sistema Operativo

• Procesador de Textos

• Software para el desarrollo de aplicaciones “eclipse”.

• Cañón

• Plumones

• Pizarrón.

• Equipos de cómputo para todos los estudiantes de la asignatura.

**- DESARROLLO DE LA PRÁCTICA**

Toda aplicación con un mínimo de seguridad necesita una pantalla de autenticación de usuario (Login) que nos da como resultado dos respuestas: pantalla de bienvenida si el usuario existe o pantalla de error si el usuario no existe.

1. De acuerdo al planteamiento, crear una aplicación en eclipse de tipo J2ME en el que se definirán tres pantallas: formLogin, formLoginOk y formLoginError.

2. También se definen dos cajas de texto y una etiqueta para mostrar la petición al usuario o el

mensaje de error: txtUsuario, txtPassword, strMensaje.

3. También se definen los comandos o acciones a realizar sobre las pantallas: cmdEntrar, cmdSalir y cmdAtras.

4. Define un método mostrarPantalla (Displayable nextDisplay) que recibe la siguiente pantalla a mostrar.

5. Implementa un método validarUsuario que obtiene los valores de las cajas de texto y compara con los valores definidos para ser correctos y llama al método dentro del commandAction.

6. Muestra los resultados con pantallas que demuestren su funcionamiento.

**Desarrollo de la práctica en Java**

El programa tiene los componentes que se crean por defecto al crear un proyecto en Android studio, pero en este caso vamos a crear 3 actividades más con su respectivo layout para diseñar.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La primera actividad es formLogin. Su diseño en XML es el siguiente.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Dentro del archivo XML se utilizan dos Layout, el constraint layout para mayor comodidad al momento de diseñar, son dos, uno para la parte superior y una para la parte inferior, el primero contendrá un fondo amarillo y un TextView que mostrará el texto Login ITESCAM. Para el layout inferior se presenta un texto que da la bienvenida al usuario, dos textos uno que dice usuario y contraseña y sus respectivos EditText para recibir la entrada, en ellos se va a ingresar el usuario y la contraseña y por último un botón que sirve para ingresar y/o validar que exista el dato ingresado.

Texto

Descripción generada automáticamente

Primeramente, creamos 2 variables de tipo EditText y 1 de tipo botón, para que en el método onCreate hagamos referencia a los elementos creados en el archivo XML, una vez referenciados se le asigna un OnClickListener al botón creado para que se ejecute una acción al momento de hacer clic en él, este será el método validar usuario.

Texto

Descripción generada automáticamente

El método validar usuario obtiene los textos de edit text contraseña y usuario, en caso de que estos estén vacíos se muestra un toast que diga que no hay datos, caso contrario si el dato ingresado es igual a usuario 7267 y contraseña 12345 se hará un cambio de pantalla al formLoginOk, en caso de que este no sea igual se enviará al formLoginError.

La segunda actividad es formLoginOk. Su diseño en XML es el siguiente.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Hay dos layout, uno de ellos es el principal y uno que sirve como encabezado el cual da la bienvenida al usuario 7267 y más abajo un texto que confirma que el usuario fue correcto, el segundo layout se pinta de verde en caso de que los datos sean correctos.

Texto

Descripción generada automáticamente

Su código en Java no tiene ninguna función, esta clase solo contiene el método onCreate que se crea por defecto.

La tercera actividad es formLoginError. Su diseño en XML es el siguiente.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Hay dos layout, uno de ellos es el principal y uno que sirve como encabezado para mostrar error y más abajo un texto que le hace saber al usuario que los datos ingresados son incorrectos, el segundo layout se pinta de rojo en caso de que los datos no sean correctos.

**Pruebas en Java.**

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

Descripción generada automáticamente

Si le damos ingresar sin poner datos nos arroja un toast

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

Descripción generada automáticamente

Ingresamos datos erróneos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ingresamos datos correctos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

**Desarrollo de la práctica en C# (Xamarin)**

El programa tiene los componentes que se crean por defecto al crear un proyecto en Xamarin, pero en este caso vamos a crear 3 actividades más con su respectivo archivo XAML.

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

Descripción generada automáticamente

La primera Pantalla es formLogin. Su diseño en XAML es el siguiente

Texto

Descripción generada automáticamente

Dentro del archivo XAML contendrá un fondo amarillo y un TextView que mostrará el texto Login ITESCAM en la parte superior. Para la parte inferior se presenta un texto que da la bienvenida al usuario, dos textos uno que dice matricula y contraseña y sus respectivos Entry para recibir la entrada, en ellos se va a ingresar el usuario y la contraseña y por último un botón que sirve para ingresar y/o validar que exista el dato ingresado.

**Texto

Descripción generada automáticamente**

El método validar usuario obtiene los textos de los Entry matricula y usuario, en caso de que estos estén vacíos se muestra un DisplayAlert que diga que no hay datos, caso contrario si el dato ingresado es igual a usuario 7267 y contraseña 12345 se hará un cambio de pantalla al formLoginOk, en caso de que este no sea igual se enviará al formLoginError.

La segunda pantalla es formLoginOk. Su diseño en XAML es el siguiente.

**Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media**

Hay un encabezado el cual da la bienvenida al usuario 7267 y más abajo un texto que confirma que el usuario fue correcto, el encabezado se pinta de verde en caso de que los datos sean correctos.

**Texto

Descripción generada automáticamente**

Su código en en C# no tiene ninguna función, esta clase solo contiene el método constructor que inicializa los componentes

La tercera pantalla es formLoginError. Su diseño en XAML es el siguiente.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

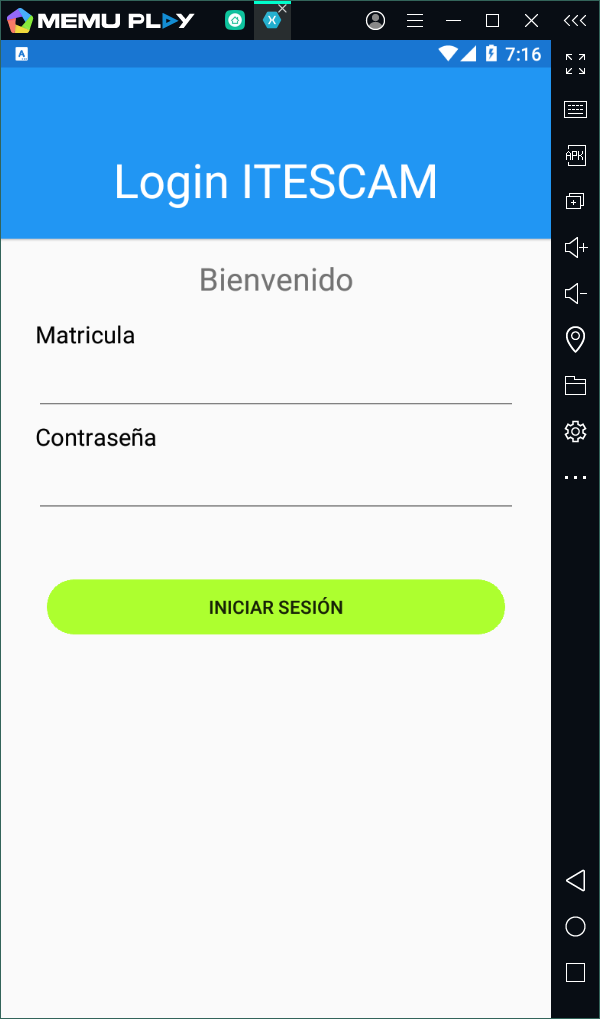
Hay un encabezado para mostrar error y más abajo un texto que le hace saber al usuario que los datos ingresados son incorrectos, el encabezado se pinta de rojo en caso de que los datos no sean correctos.

**Texto

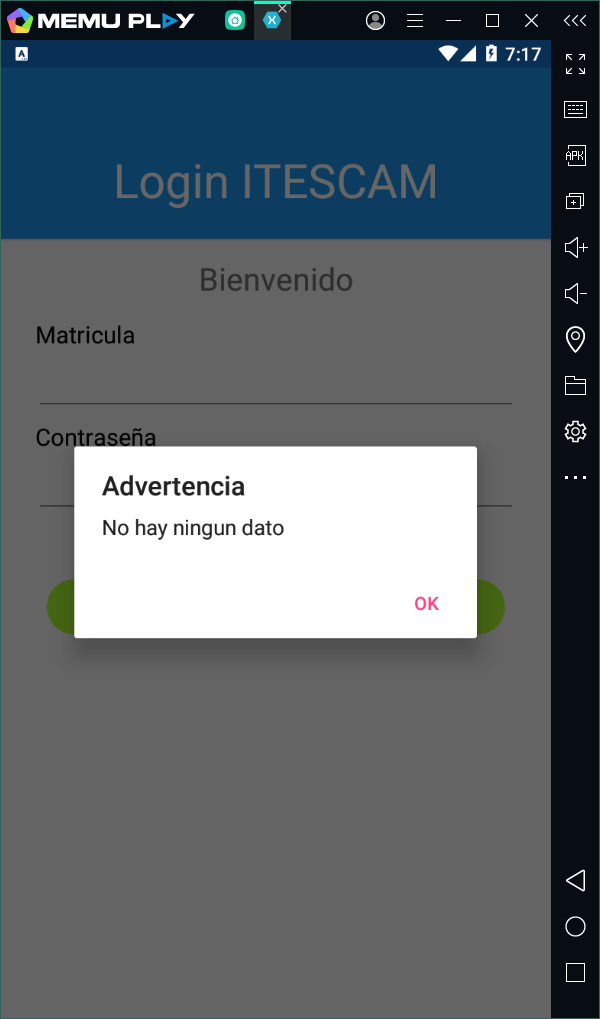
Descripción generada automáticamente**

Su código en en C# no tiene ninguna función, esta clase solo contiene el método constructor que inicializa los componentes

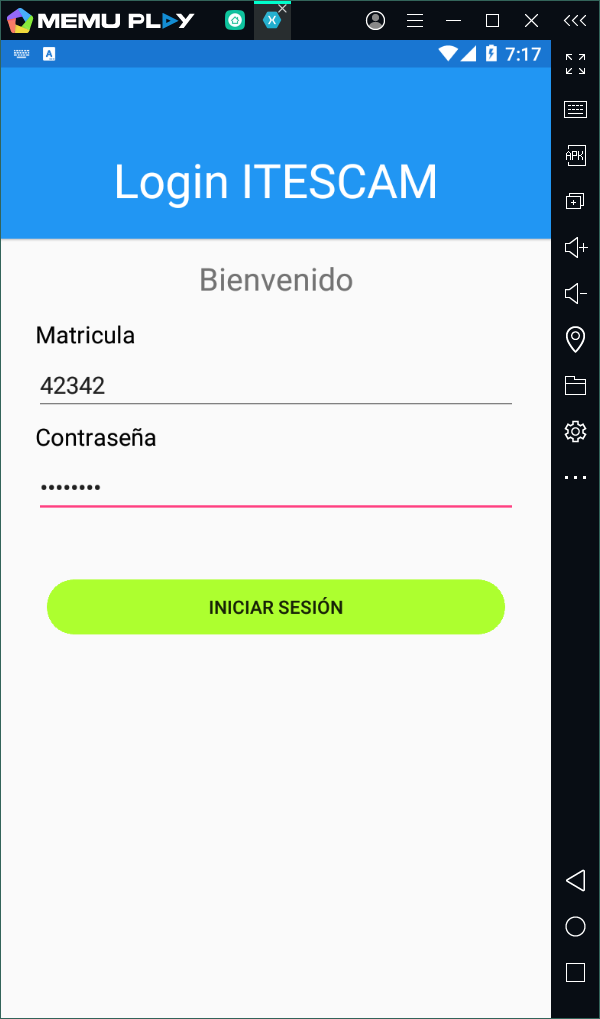
**Pruebas en Xamarin.**

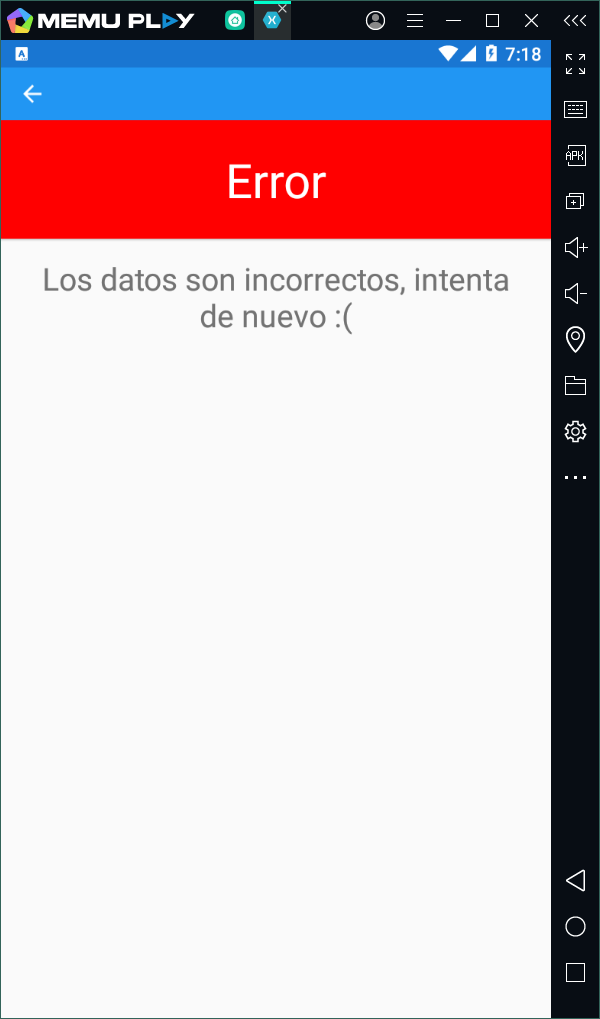
****

Si le damos ingresar sin poner datos nos arroja un DisplayAlert.

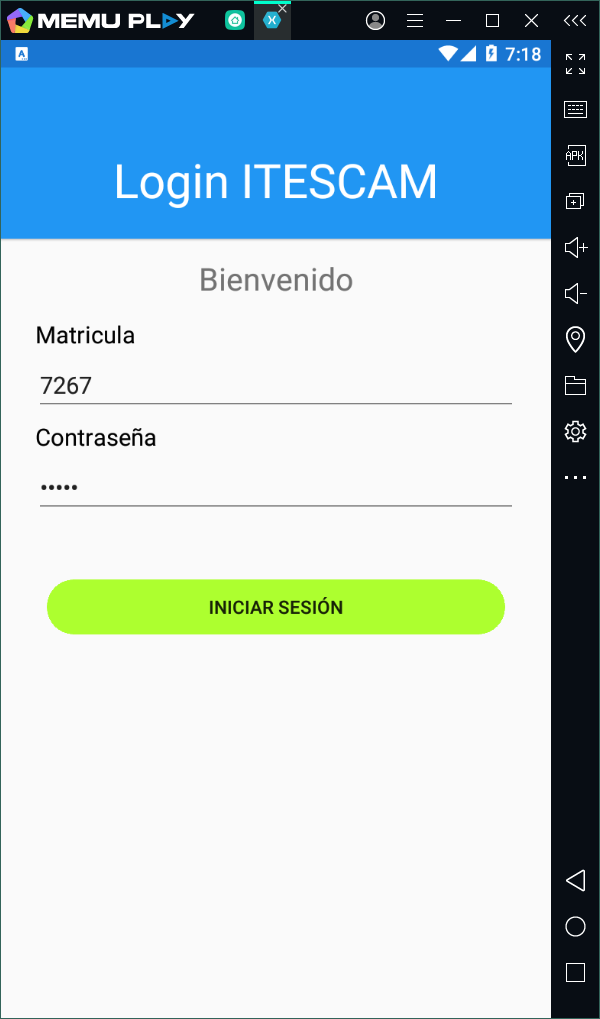


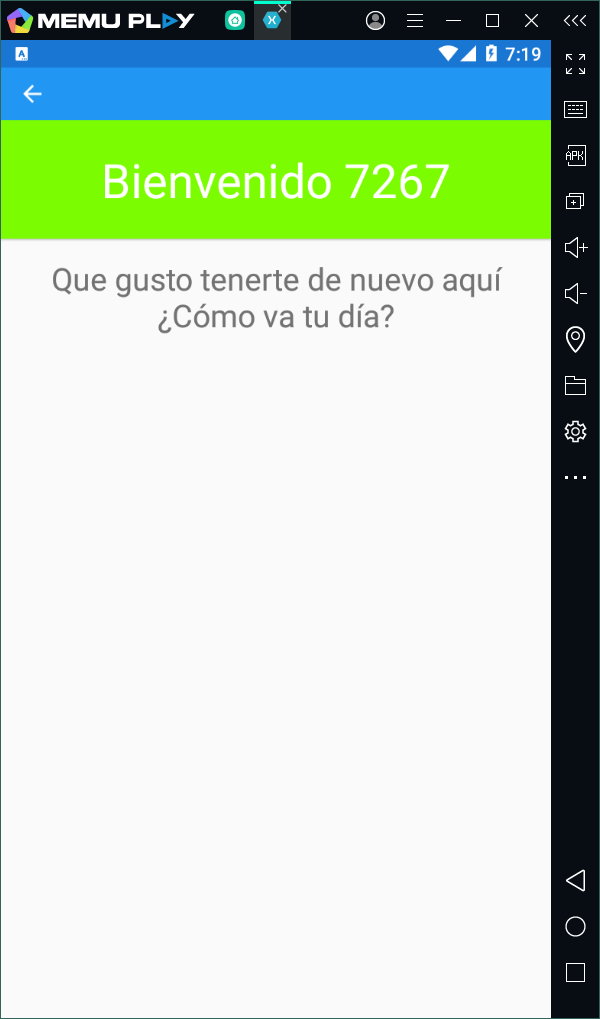
Ingresamos datos erróneos.

****

****

Ingresamos datos correctos.

****

****

**Conclusión.**

Durante el desarrollo de esta práctica fue la primera que realicé en Xamarin, con respecto a Java en Android studio el XAML aunque es parecido a XML reduce mucho las línea de código al diseñar, como ejemplo el diseño de la pantalla principal en XML se llevó 120 lineas y en XAML menos de 50, obviamente Android studio al ser una herramienta más elaborada y con más capacidad es más fácil diseñar que en Xamarin, por otro lado las implementaciones son iguales, validar los datos en ambos lenguajes Java y C# no cambia mucho, solamente el clic del botón es diferente en ambos, fuera de ello ambas herramientas sirven para trabajar aplicaciones móviles sin ningún problema, personalmente no conocía Xamarin y esta práctica me dio la oportunidad de conocer esta herramienta.

**- EVALUACIÓN Y RESULTADOS**

**-REFERENCIAS**

Sergio Galvez Rojas, Lucas Ortega Díaz. Java a tope: J2ME. Universidad de Málaga. 2003

Heriberto Vazquez. Empezando con J2ME. Link:

[http://programandodemadrugada.blogspot.mx/2011/10/1-empezando-con-j2me.html. F](http://programandodemadrugada.blogspot.mx/2011/10/1-empezando-con-j2me.html)ebrero 2016

**-ANEXOS**

Estructura del proyecto Java

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Pantalla principal

Interfaz de usuario gráfica

Descripción generada automáticamente

Pantalla en caso de que coincidan los datos ingresados

Interfaz de usuario gráfica

Descripción generada automáticamente

Pantalla en caso de que no coincidan los datos ingresados

Interfaz de usuario gráfica

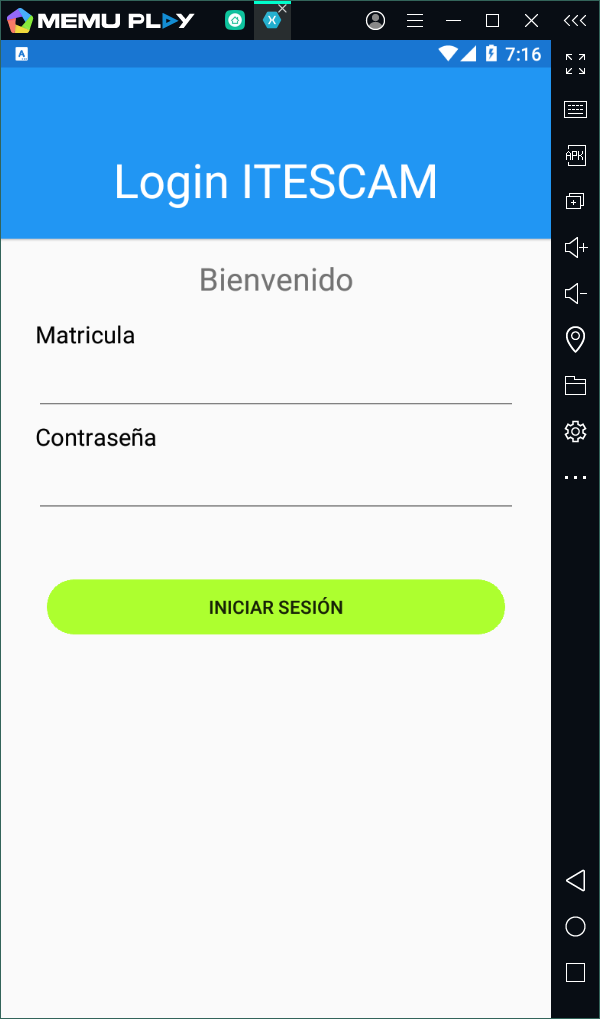
Descripción generada automáticamente

Estructura del proyecto Xamarin

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Pantalla principal Xamarin

****