PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

TEMA 7:

Intents.

ÍNDICE

PMDM

Intents.

1. Los Intent explícitos

1.- Intents explícitos

- Para pasar de una actividad a otra utilizamos los "Intent explícitos".
- Un "Intent" es declarar la intención de lo que se quiere hacer. En este caso, la intención es ejecutar otra actividad distinta a la que nos encontramos. Esto lo realiza el siguiente código:

  val intent: Intent = Intent(contextoActividadActual, OtraActividad::class.java)

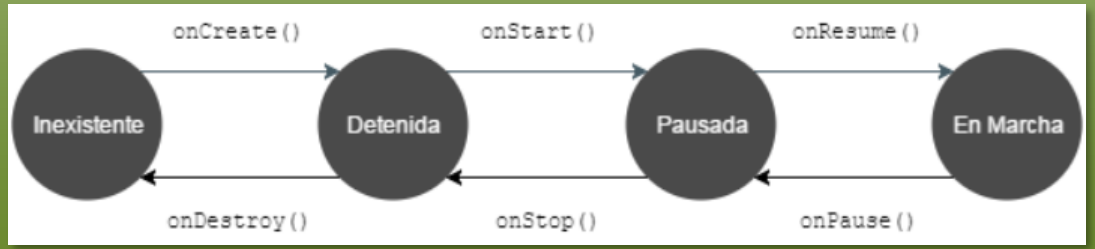
  donde contextoActividadActual puede ser *this* o *baseContext*.
- Por último, después de declarar la intención, hay que lanzar la actividad con:
  - *startActivity(intent)*

1.- Intents explícitos

- Vamos a crear un proyecto con dos actividades: *Correr* y *HacerFlexiones*.
- Correr tendrá un botón que nos llevará a la actividad de HacerFlexiones, y al pulsar "Atrás" volveremos a la actividad "Correr".
- Además vamos a editar todos los métodos que representan los estados del ciclo de vida de la actividad "Correr", añadiéndoles que muestren un mensaje por pantalla (Toast) al entrar en cada estado.

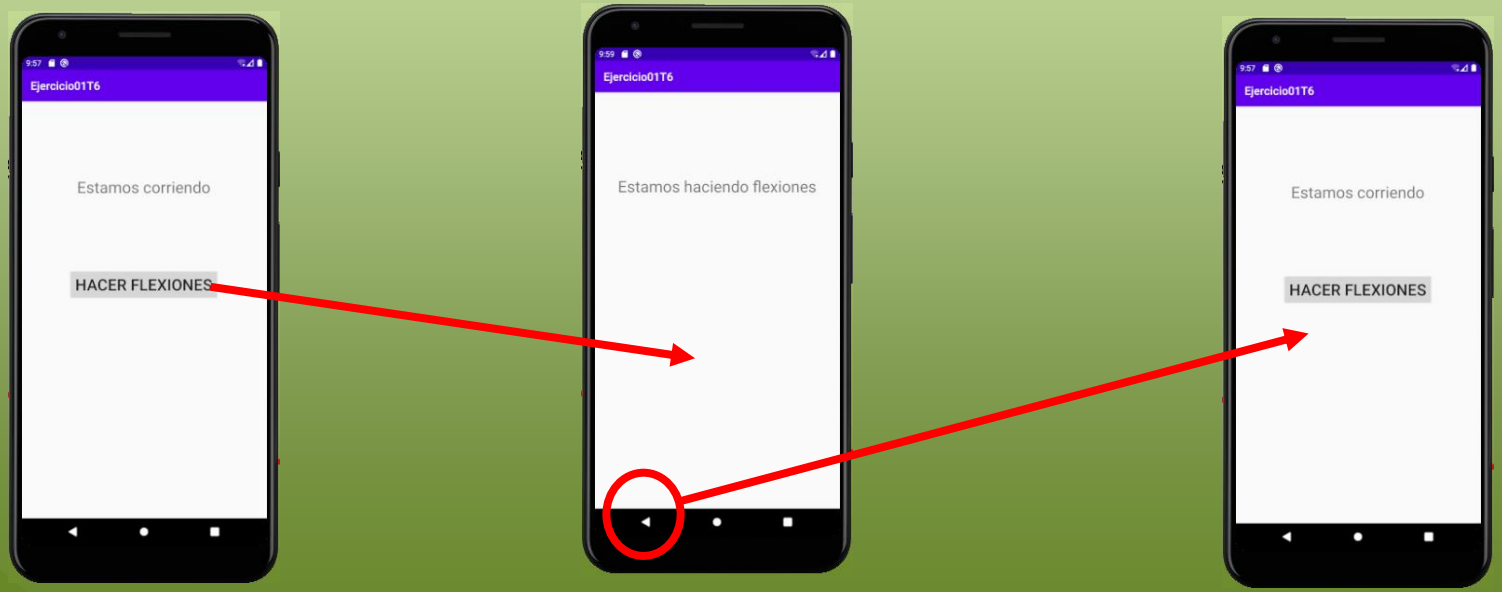
EJERCICIOS

- **Ejercicio 01:** Crea un proyecto Android que contenga dos Actividades (Empty Views Activity).
- Sobrescribe todos los métodos del ciclo de vida de una actividad, de manera que cada una de ellas muestre un Toast por pantalla.



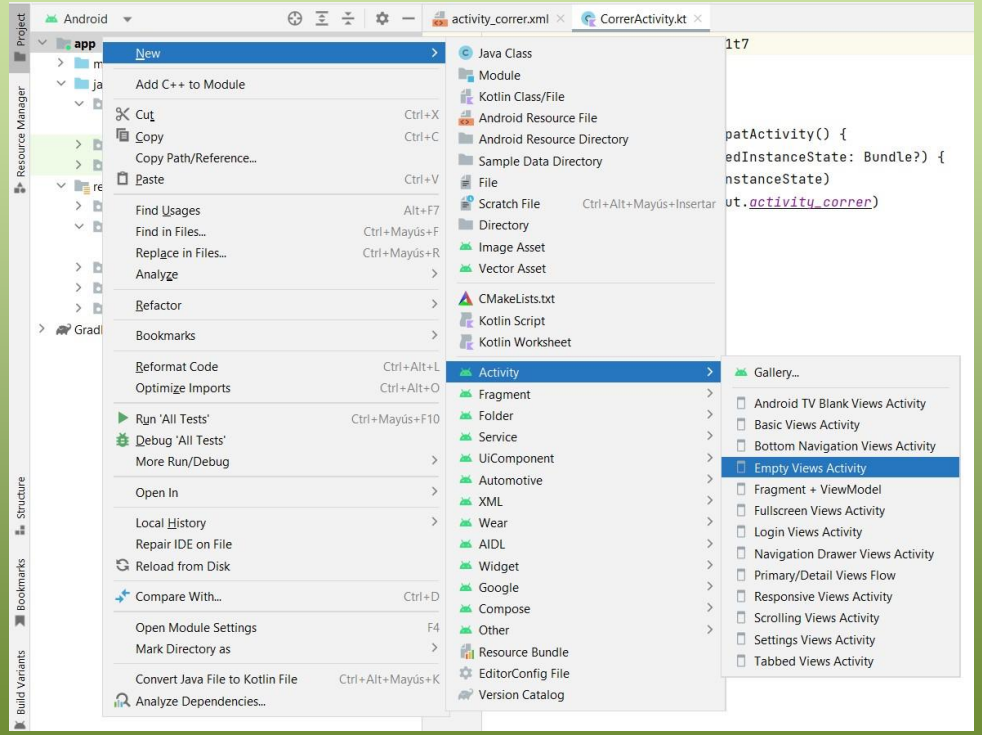
EJERCICIOS

- Quedaría de la siguiente manera:



EJERCICIOS

- Lo primero será diseñar el layout de nuestra actividad principal y, después, crear nuestra segunda activity, tal y como muestra la imagen.



EJERCICIOS

- Programamos el evento de pulsar el botón de la actividad "Correr". En él creamos el Intent para llamar a la actividad "HacerFlexiones".

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_correr)

    val btnHacerFlexiones: Button = findViewById<Button>(R.id.btnHacerFlexiones)

    btnHacerFlexiones.setOnClickListener { it: View!
        val intent: Intent = Intent( packageContext: this, HacerFlexionesActivity::class.java)
        startActivity(intent)
    }
}
```

EJERCICIOS

- Por último, vamos a sobrescribir los estados por los que pasa la actividad "Correr" para que nos avise con un mensaje cada vez que pase por uno de ellos:

```
override fun onStart() {
    super.onStart()
    Toast.makeText(applicationContext, text: "2. onStart", Toast.LENGTH_SHORT).show()
}

override fun onResume() {
    super.onResume()
    Toast.makeText(applicationContext, text: "3. onResume", Toast.LENGTH_SHORT).show()
}

override fun onPause() {
    super.onPause()
    Toast.makeText(applicationContext, text: "4. onPause", Toast.LENGTH_SHORT).show()
}

override fun onStop() {
    super.onStop()
    Toast.makeText(applicationContext, text: "5. onStop", Toast.LENGTH_SHORT).show()
}

override fun onDestroy() {
    super.onDestroy()
    Toast.makeText(applicationContext, text: "6. onDestroy", Toast.LENGTH_SHORT).show()
}
```

1.- Intents explícitos

- También podemos pasar parámetros de un Activity a otro Activity. Para ello, una vez creado el Intent, utilizaremos el método *putExtra* para asociarle a ese Intent los parámetros que le queramos pasar.
- Al método putExtra le debemos pasar dos parámetros:
  - El **nombre** (o clave) de la variable que le queramos pasar.
  - El **valor** de esa variable.

*intent.putExtra("**nombreUsuario**", **"Pepe"**);*

*intent.putExtra("**edad**", **23**);*

1.- Intents explícitos

- Una vez creado el Intent en una Activity, deberemos recogerlo junto a sus parámetros en la Activity a la que saltamos. Para ello utilizaremos los métodos *intent (getIntent())* y *getTipoExtra()*, siendo *Tipo* el tipo de dato del parámetro que queremos recoger. El método *hasExtra()* nos permite comprobar si el Intent contiene el parámetro que queremos recuperar.
- Ejemplos:

```
val intent: Intent = intent
if(intent.hasExtra("nombreUsuario") {
        val nombre: String? = intent.getStringExtra("nombreUsuario")
}
if(intent.hasExtra("edad") {
        val edad: Int = intent.getIntExtra("edad", defaultValue)
}
```

1.- Intents explícitos

- Otra forma de recoger los datos del Intent sería creando un objeto de la clase *Bundle* (es una especie de HashMap) cuyos elementos serán los parámetros del Intent que le hemos pasado a la Activity. El objeto Bundle del Intent lo obtenemos mediante el método *extras (getExtras()).*

```
val intent: Intent = intent
val extras: Bundle? = intent.extras
if(extras != null) {
    val nombre: String? = extras.getString("nombreUsuario")
    val edad: Int = extras.getInt("edad", defaultValue)
}
```

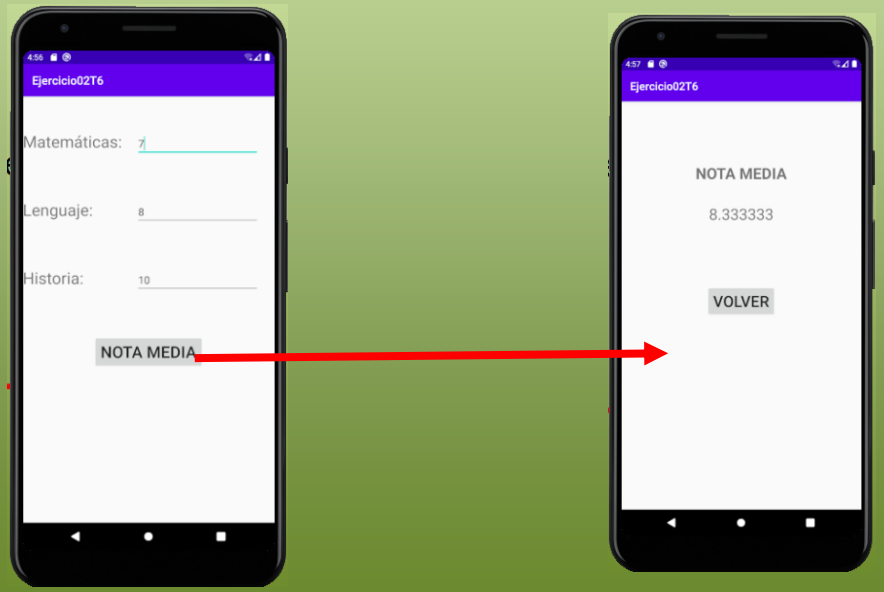
EJERCICIOS

- **Ejercicio 02:** Crea un proyecto Android que contenga dos Actividades (Empty Views Activity): una para introducir las calificaciones y otra para mostrar la nota media.
- Pista: Necesitarás el siguiente código para el botón *Volver*.

```
//EVENTO BOTÓN VOLVER
btnVolver.setOnClickListener { it: View!
    finish()
}
```

EJERCICIOS

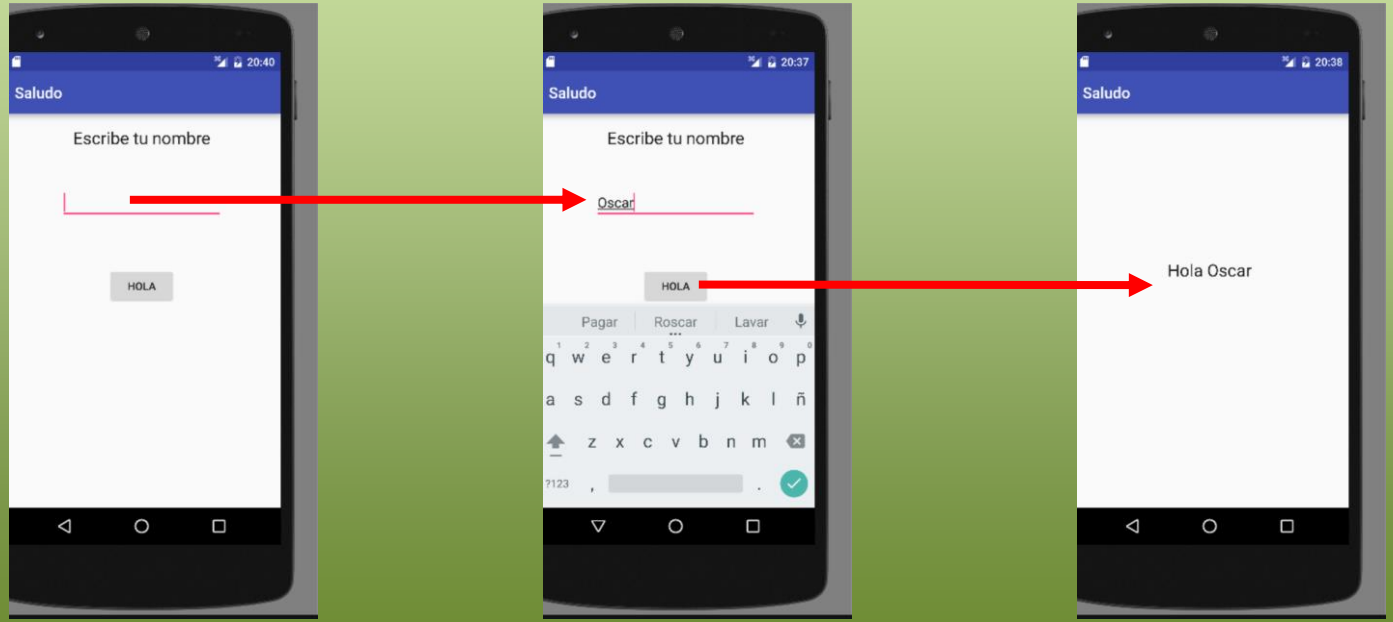
- Quedaría de la siguiente manera:



EJERCICIOS

- **Ejercicio 03:** Crea un proyecto Android que contenga dos Actividades (Empty Views Activity), tal y como se muestra en la siguiente diapositiva.

EJERCICIOS


20:40
Saludo
Escribe tu nombre
HOLA
20:37
Saludo
Escribe tu nombre
Oscar
HOLA
Pagar
Roscar
Lavar
20:38
Saludo
Hola Oscar

PMDM

Intents.

2. Los Intents implícitos

2.- Intents implícitos

- Los Intents implícitos son las herramientas gracias a las que podemos reutilizar aplicaciones ya instaladas en nuestro dispositivo.
- Con ellos declaramos la intención de comenzar una acción genérica sin nombrar el componente que debe realizarla.
- Por ejemplo, si quiero utilizar la aplicación de cámara de fotos de mi dispositivo, escribiría el siguiente código:

```
val intentFoto: Intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
startActivity(intentFoto)
```

2.- Intents implícitos

- Fíjate que el parámetro que le pasamos al Intent es la acción que deseo realizar.
- Todas las acciones posibles aparecen en la documentación de la API de Android:

https://developer.android.com/guide/components/intents-common?hl=es-419

2.- Intents implícitos

- En los Intents explícitos, podíamos pasarle datos mediante el método putExtra.
- En cambio, en los Intents implícitos, podemos utilizar el método setData o putExtra, al que tenemos que pasarle un objeto de tipo URI (Identificador de Recursos Uniforme).
- Para crear datos de tipo URI bastará con "parsear" nuestro dato simple. Ejemplo:

```
val intent: Intent = Intent (Intent.ACTION_DIAL)
intent.setData(Uri.parse ("tel:111222333"))
startActivity (intent)
```

2.- Intents implícitos

- A partir de Android 6.0 (nivel de API 23), los usuarios conceden permisos a las apps mientras se ejecutan, no cuando instalan la app. Este enfoque simplifica el proceso de instalación de la app, ya que el usuario no necesita conceder permisos cuando instala o actualiza la app. También brinda al usuario mayor control sobre la funcionalidad de la app; por ejemplo, un usuario podría optar por proporcionar a una app de cámara acceso a esta, pero no a la ubicación del dispositivo. El usuario puede revocar los permisos en cualquier momento desde la pantalla de configuración de la app.
- Los permisos del sistema se dividen en dos categorías: normal y peligroso.
  - Los **permisos normales** no ponen en riesgo la privacidad del usuario de forma directa. Si tu app tiene un permiso normal en su manifiesto, el sistema concede el permiso automáticamente.
  - Los **permisos peligrosos** pueden permitir que la app acceda a información confidencial del usuario. Deben estar declarados en su manifiesto y, además, el usuario debe autorizarlo explícitamente desde tu app.

2.- Intents implícitos

- Un ejemplo de intent implícito con permisos normales sería el *ACTION_DIAL* (Acción marcar).
- Esta acción nos permitiría mostrar una interfaz de usuario con el número que se estaría marcando, pero sin llegar a realizar la llamada.
- Tendría que ser el usuario el que iniciase la llamada pulsando el botón de llamar (de ahí que sea considerado tipo de "permiso normal").

EJERCICIOS

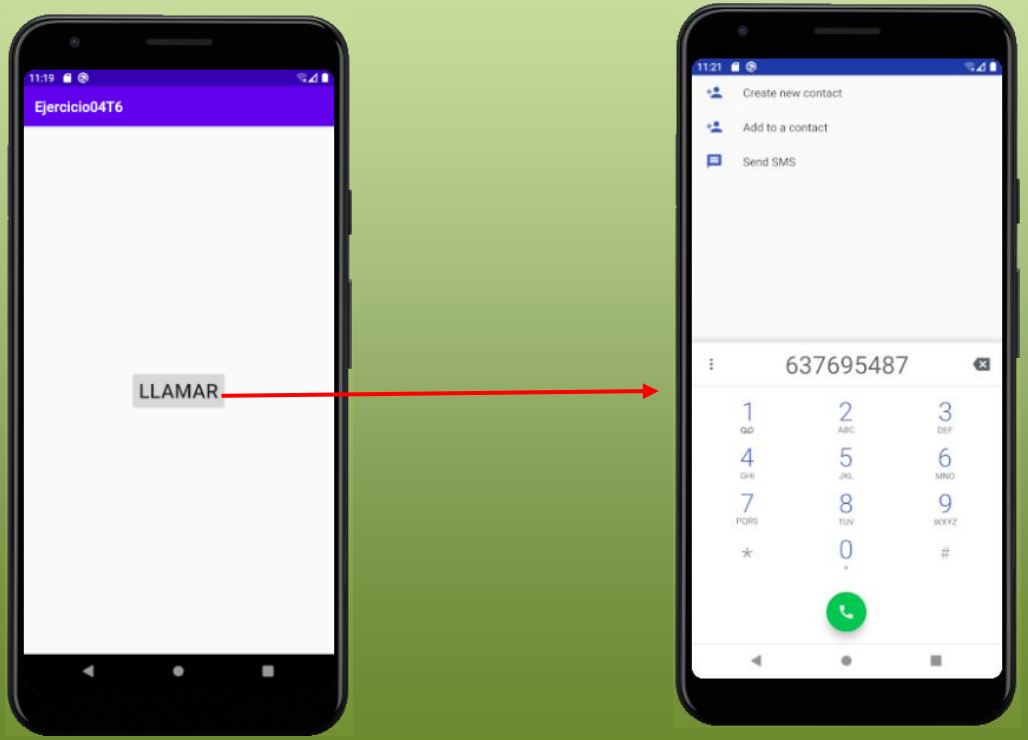
- **Ejercicio 04:** Crea un proyecto Android que contenga una única actividad de tipo Empty Views Activity con un botón en el centro. Al pulsar este botón, aparecerá marcado un número de teléfono esperando a que el usuario pulse el botón de llamar.

EJERCICIOS

- Lo primero será añadir el botón a nuestra activity, haciendo uso de los muelles para situarlo en el centro.
- Una vez implementado el diseño, tendremos que realizar el tratamiento del botón.
- El evento click deberá contener el código que se muestra en la imagen:

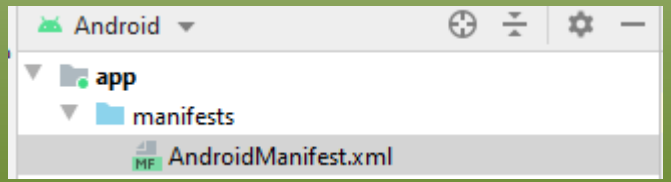
```
val btnLlamar: Button = findViewById<Button>(R.id.btnLlamar)
btnLlamar.setOnClickListener { it: View!
    val intentLlamar: Intent = Intent(Intent.ACTION_DIAL)
    intentLlamar.setData(Uri.parse( uriString: "tel:637695487"))
    startActivity(intentLlamar)
}
```

EJERCICIOS


11:19
Ejercicio04T6
LLAMAR
11:21
Create new contact
Add to a contact
Send SMS
637695487
1
2
3
4
5
6
7
8
9
*
0
#

2.- Intents implícitos

- Ahora vamos a hacer la versión catalogada como peligrosa: vamos a hacer un intent implícito a través del cual directamente llamamos por teléfono a un número.
- Todas las acciones consideradas peligrosas deben tener su correspondiente permiso en el archivo *manifest.xml* y, además, durante la ejecución, el usuario nos tiene que dar permiso explícito.



2.- Intents implícitos

vity_main.xml | AndroidManifest.xml | MainActivity.kt

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.ejercicio5t7">

    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-feature android:name="android.hardware.telephony" android:required="true" />
```

```kotlin
private fun llamar(){
    val permiso: String = "android.permission.CALL_PHONE"
    if(ContextCompat.checkSelfPermission( context: this, permiso) == PackageManager.PERMISSION_GRANTED) {
        val intentLlamar: Intent = Intent(Intent.ACTION_CALL)
        intentLlamar.setData(Uri.parse( uriString: "tel:637695487"))
        startActivity(intentLlamar)
    } else {
        Toast.makeText(applicationContext, R.string.no_permisos_msg, Toast.LENGTH_SHORT).show()
        ActivityCompat.requestPermissions( activity: this, arrayOf(permiso), requestCode: 0)
    }
}
```

EJERCICIOS

- **Ejercicio 05:** Crea un proyecto Android que contenga una única actividad de tipo Empty Views Activity con un botón en el centro. Al pulsar este botón se realizará, automáticamente, una llamada telefónica.

EJERCICIOS

- **Ejercicio 06:** Ahora vamos a crear una aplicación que utilice Intents implícitos para:
  - Visualizar un página web.
  - Mostrar el mapa de una coordenada GPS.
  - Enviar programáticamente un email.

EJERCICIOS

- Como en esta Activity vamos a tener varios botones, vamos a implementar el tratamiento de otra forma, para tenerlo todo más ordenado.
- Los pasos a seguir en el tratamiento de los botones son tres:
  - Implementaremos la interface *View.OnClickListener.*
  - Establecemos el listener *setOnClickListener* para cada botón.
  - Sobreescribimos el método *onClick()*.

EJERCICIOS

- **interface View.OnClickListener**
- Nuestra activity debe implementar la interfaz, para poder sobreescribir más adelante el método onClick.

```
package com.example.ejercicio6t7

import ...

class MainActivity : AppCompatActivity(), View.OnClickListener {
```

EJERCICIOS

- **setOnClickListener**
- Recogemos cada botón y establecemos el listener *setOnClickListener*.

```
btnIr = findViewById<Button>(R.id.btnIr)
btnVerMapa = findViewById<Button>(R.id.btnVerMapa)
btnEnviar = findViewById<Button>(R.id.btnEnviar)

btnIr.setOnClickListener(this)
btnVerMapa.setOnClickListener(this)
btnEnviar.setOnClickListener(this)
```

EJERCICIOS

- **onClick**
- Sobreescribimos el método *onClick()*.

```
override fun onClick(view: View) {
    when (view.id) {
        R.id.btnIr -> visitarURL()
        R.id.btnVerMapa -> visitarGoogleMaps()
        R.id.btnEnviar -> enviarMail()
    }
}
```

EJERCICIOS

- Soluciones para cada intent:

```
private fun enviarMail() {
    val destinatario: String? = editTextEmail.text.toString()
    if(destinatario == null || destinatario == "") {
        Toast.makeText(applicationContext, R.string.msg_no_email, Toast.LENGTH_SHORT).show()
    } else {
        val asunto: String = "Mi primer email"
        val texto: String = "Hola, este es mi primer email enviado desde una app"
        val type: String = "text/plain"

        val intent: Intent = Intent(Intent.ACTION_SEND)
        intent.putExtra(Intent.EXTRA_EMAIL, destinatario)
        intent.putExtra(Intent.EXTRA_SUBJECT, asunto)
        intent.putExtra(Intent.EXTRA_TEXT, texto)
        intent.type=type
        startActivity(intent)
    }
}
```

```
private fun visitarURL() {
    val url: String? = editTextUrl.text.toString()
    if(url == null || url == "") {
        Toast.makeText(applicationContext, R.string.msg_no_url, Toast.LENGTH_SHORT).show()
    } else {
        val intent: Intent = Intent(Intent.ACTION_VIEW)
        intent.setData(Uri.parse(url))
        startActivity(intent)
    }
}
```

```
private fun visitarGoogleMaps() {
    val latitud: Double? = editTextLatitud.text.toString().toDoubleOrNull()
    val longitud: Double? = editTextLongitud.text.toString().toDoubleOrNull()
    if(latitud == null || longitud == null) {
        Toast.makeText(applicationContext, R.string.msg_no_coordenadas, Toast.LENGTH_SHORT).show()
    } else {
        val intent: Intent = Intent(Intent.ACTION_VIEW)
        intent.setData(Uri.parse( uriString: "geo:$latitud,$longitud"))
        startActivity(intent)
    }
}
```

PMDM

Intents.

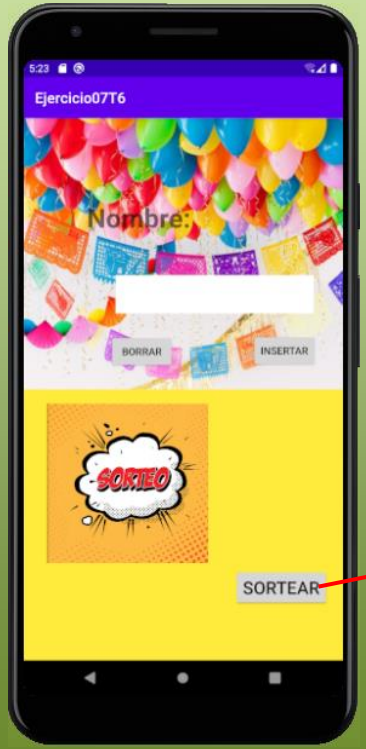
3. Ejercicios de consolidación

EJERCICIOS

- **Ejercicio 07:** Crea una aplicación donde se irán insertando nombres de socios en un ArrayList.
- Tendremos tres botones:
  - Borrar. Limpiará el EditText.
  - Insertar. Insertará el socio en el ArrayList. Una vez insertado, limpiará el EditText y mostrará un Toast para avisar de que todo ha ido bien.
  - Sortear. Saltará a otro Activity. Recibirá por parámetros el nombre del socio/a ganador/a. Si el arrayList está vacío, avisará mediante un Toast.

- Pista: Utiliza un número aleatorio para generar el socio/a ganador/a (el ganador/a se encontrará en la posición del arrayList correspondiente al aleatorio generado):
  - *val aleatorio: Int = Random.nextInt (o, arrayList.size)*

EJERCICIOS


5:23
Ejercicio07T6
Nombre:
BORRAR
INSERTAR
SORTEO
SORTEAR

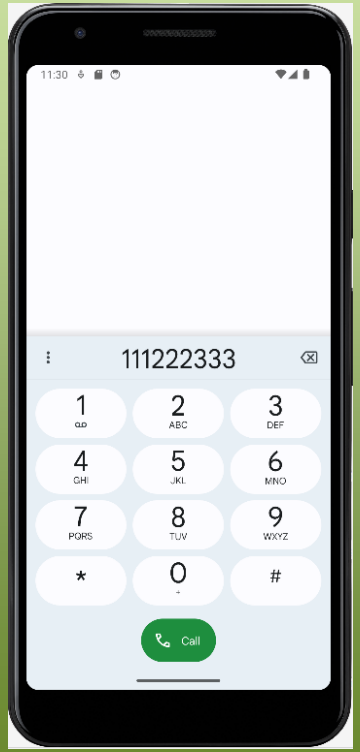

5:23
Ejercicio07T6
GANADOR!!!
Juan
Mariscal

EJERCICIOS

- **Ejercicio 08:** Crea un proyecto en Android Studio en el que utilices la plantilla *Tabbed Activity*. Tendrá las pestañas *Implícitos* y *Explícitos.*
- Cada fragment tendrá un botón en el centro de la pantalla.
- Al pulsar el botón de la pestaña "Implícitos" aparecerá marcado en pantalla un número de teléfono.
- Al pulsar el botón de la pestaña "Explícitos" saltará a otra activitity (con un único botón en el centro para volver).

EJERCICIOS


11:30
Ejercicio8T7
IMPLÍCITOS
EXPLÍCITOS
LLAMAR

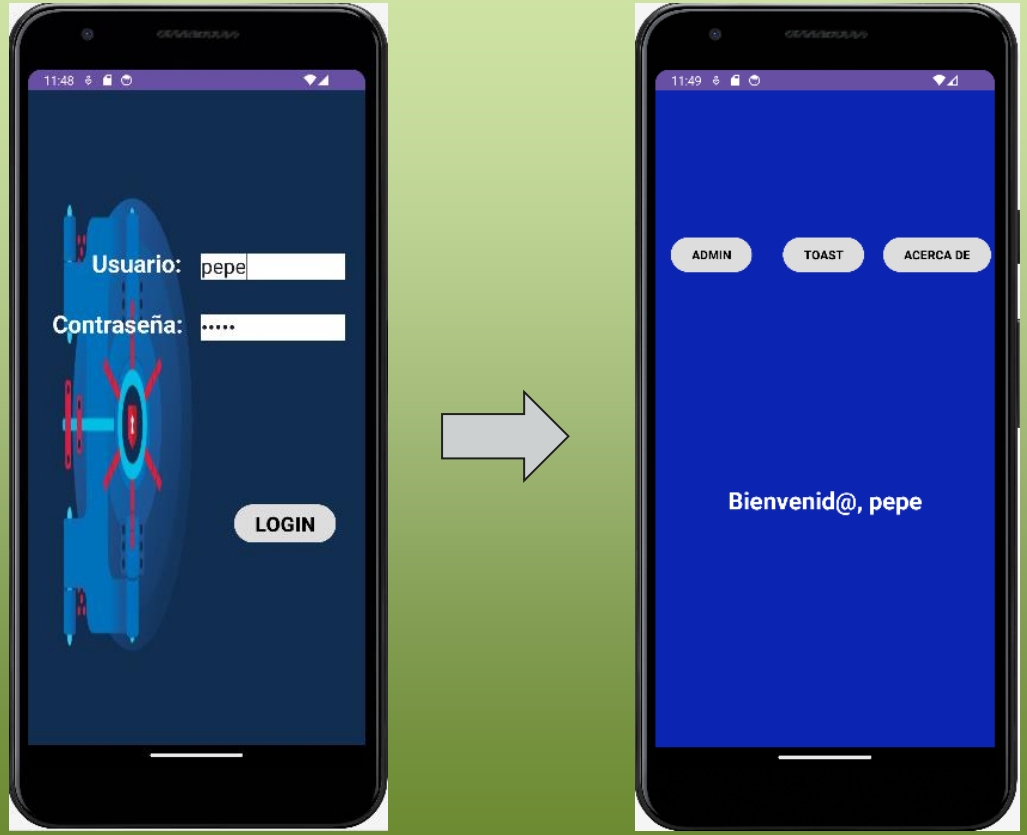

11:30
111222333
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#
Call


11:31
Ejercicio8T7
IMPLÍCITOS
EXPLÍCITOS
SALTO ACTIVITY


11:32
VOLVER

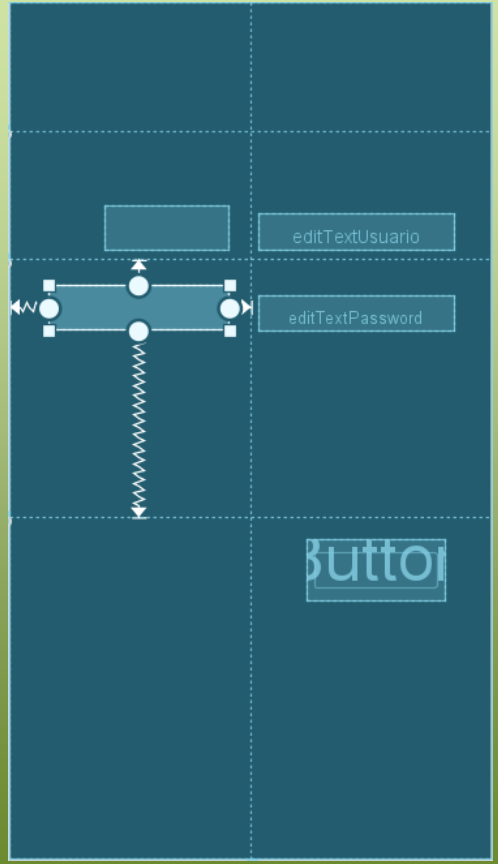
EJERCICIOS

- **Ejercicio 09:** Crea un proyecto en Android Studio donde se muestre una pantalla de login. Al introducir los datos y pulsar en *LOGIN*, nos llevará a otra activity.



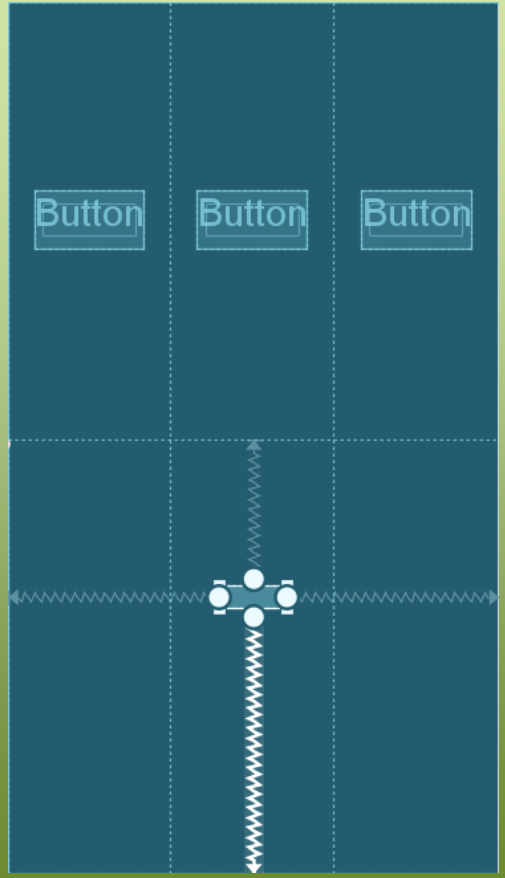
EJERCICIOS

- **Diseño Activity inicial**
- El EditText correspondiente a la contraseña será de tipo Password.
- La guía vertical estará al 50%.
- Tendremos tres guías horizontales: al 15%, 30% y 60%.



EJERCICIOS

- **Diseño Activity bienvenida**
- Dispondremos de tres botones (*Admin*, *Toast* y *Acerca de*) y un TextView, colocados tal y como se muestra en la imagen.
- Las guías verticales se situarán a un 33% y un 66% y la horizontal a un 50%.



EJERCICIOS

- **Programación Activity inicial.**
- Tendremos un atributo *arrayListUsuarios*, que será un ArrayList de la clase *Usuario*. La clase Usuario dispondrá de 3 atributos: *nomUsuario*, *pass* y *rol*. Todos ellos de tipo String.
- En el método *onCreate* cargaremos algunos datos para este ArrayList (deberás incluir, al menos, un usuario con rol "*admin*" y otro con rol "*usuario*").
- Al pulsar en el botón, buscaremos si los datos introducidos por el usuario se encuentran en nuestro ArrayList:
  - Si son correctos, saltamos a la siguiente Activity enviándole por parámetros el nombre del usuario logueado y su rol.
  - Si son incorrectos, mostraremos un Toast "*Usuario y/o contraseña incorrectos*".

EJERCICIOS

- **Programación Activity bienvenida.**
- Uno de sus atributos será de tipo *Usuario* (y sus valores se cargarán con los parámetros recibidos -la contraseña la podemos poner como vacía-).
- Carga de botones:
  - Si el usuario tiene rol "admin", mostraremos todos los botones.
  - Si el usuario tiene rol "usuario", mostraremos sólo el botón "*Acerca de*"
- Al arrancar nuestra Activity, el TextView se cargará con el mensaje "Bienvenid@, [nombre usuario logueado]".
- Cuando pulsemos los botones "Admin" y "Acerca de", se cambiará el mensaje del TextView con "Administrando: [nombre usuario logueado]" y "Autor: [tu nombre]" respectivamente.
- Cuando el usuario pulse el botón "Toast", se mostrará un Toast con el mensaje "Sesión activa: [nombre usuario logueado]".