

# FEUD

Fundación de egresados U. Distrital  
*Construyendo Profesionales*



***Somos el Centro de Entrenamiento Autorizado por marcas representativas en Gobierno TI y empresa, con el portafolio más amplio en Latinoamérica:***



Todas las marcas, nombres comerciales, marcas de servicios y logotipos a los que se hace referencia en el presente documento pertenecen a sus respectivas empresas. ITIL® es una marca registrada de AXELOS Limited. El Swirl logo™ es una marca de AXELOS Limited. PMP, PMI, PMI-RMP, PMI-ACP, PMI-SP, PgMP, CAPM, REP Logo son marcas registradas del Project Management Institute.

**FEUD**  
Fundación de egresados U. Distrital  
*Construyendo Profesionales*



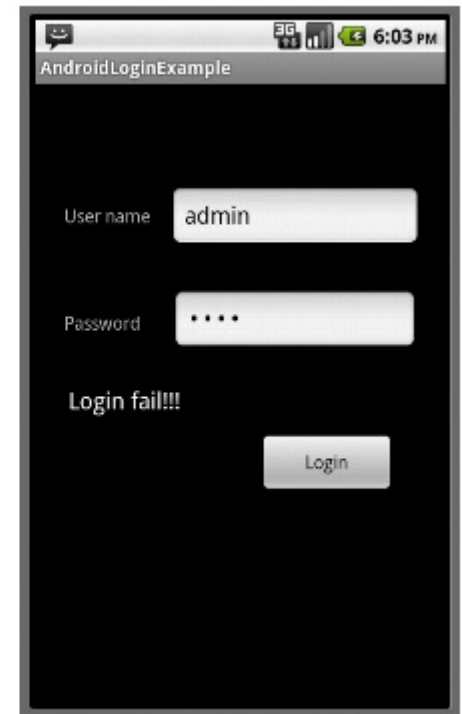
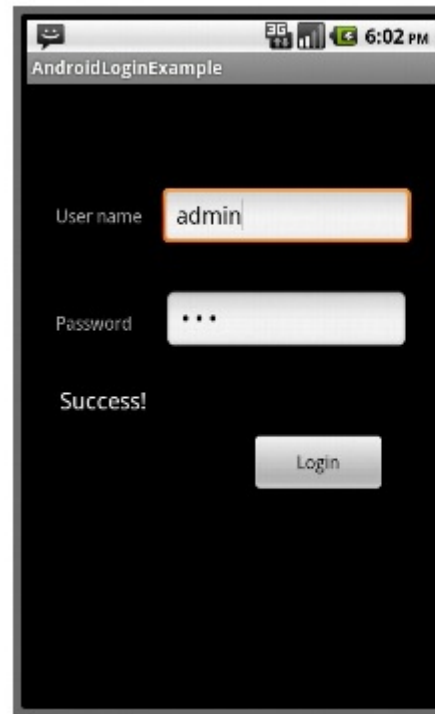
# Desarrollo de aplicaciones utilizando SDK - Android

Diplomado Desarrollo de Software para móviles

# ANDROID: ACTIVIDADES

# Actividades

- Una actividad puede verse de manera simple como una pantalla que interactúa con el usuario



# Actividades

- Una aplicación puede estar compuesta por varias actividades.
- Puede ser iniciada por un Intent
  - Intent i = **new** Intent(**this**, ActivityTwo.**class**);
  - startActivity(i);
- Las actividades comunican datos entre si mediante Bundles adjuntos al Intent

# Actividades: Usando Bundles

//Desde Activity1 (llama a segunda actividad)

```
Intent intent = new Intent(this,myActivity2.class);  
Bundle bundle = new Bundle();  
bundle.putString("myValue", myValue);  
intent.putExtras(bundle);  
navigation.this.startActivity(intent);
```

//In Activity2 (ejecutada por Actividad1)

```
Bundle bundle = getIntent().getExtras();  
act2MyValue= bundle.getString("myValue");
```

# ANDROID: LAYOUTS

# Que es un Layout?

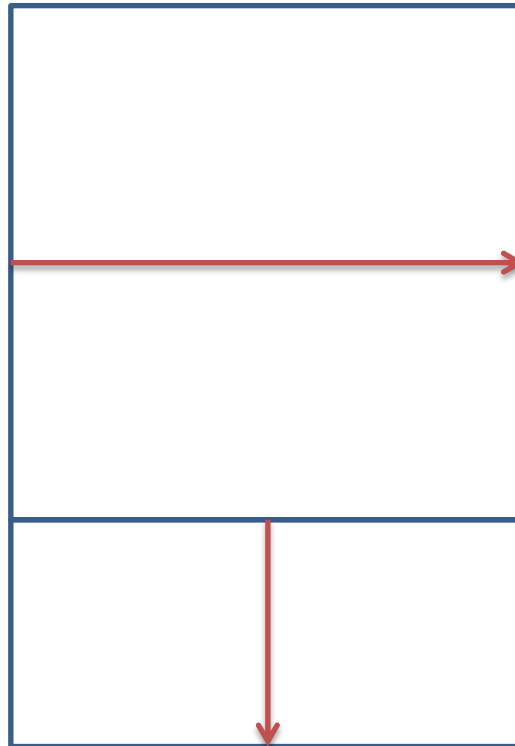
- Una agrupación de un View (Cualquier elemento de interacción)
- Extiende de ViewGroup
- Pueden crearse Layouts personalizados (creando una clase que extienda de ViewGroup)



# Tipos más comunes de Layouts

- Linear Layout
- Relative Layout
- Table Layout

# Linear Layout



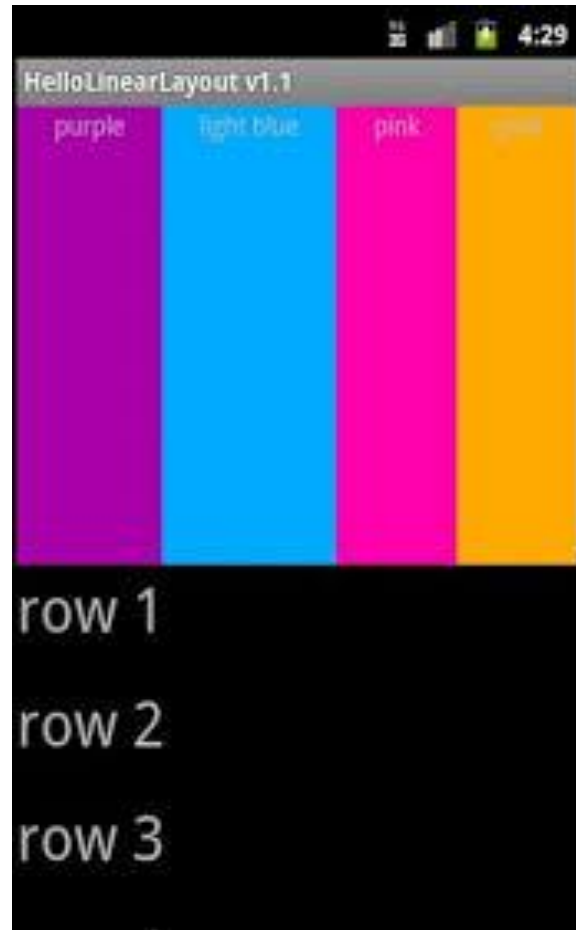
# Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Orientacion Horizontal -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

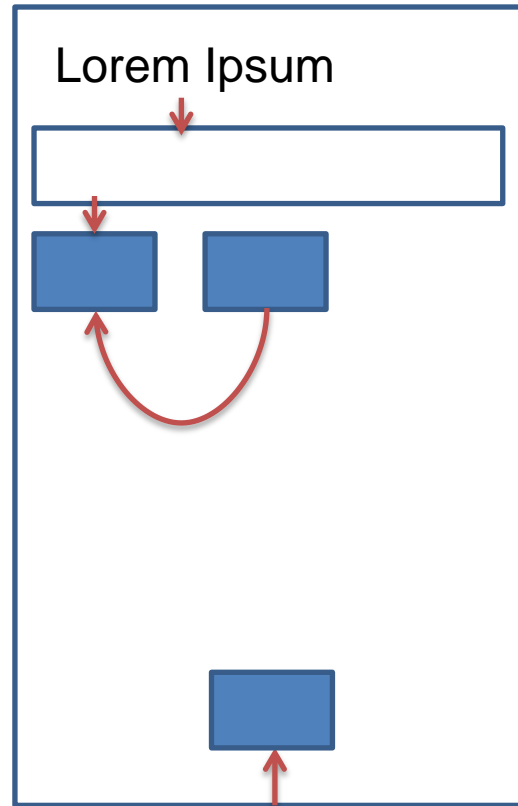
    <!-- Orientacion Vertical -->
    <LinearLayout android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="#2a2a2a"
        android:layout_marginTop="25dip"
    </LinearLayout>
</LinearLayout>
```



# Linear Layout



# Relative Layout



# Relative Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <TextView android:id="@+id/label" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/email" />

    <EditText android:id="@+id/inputEmail" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:layout_below="@id/label" />

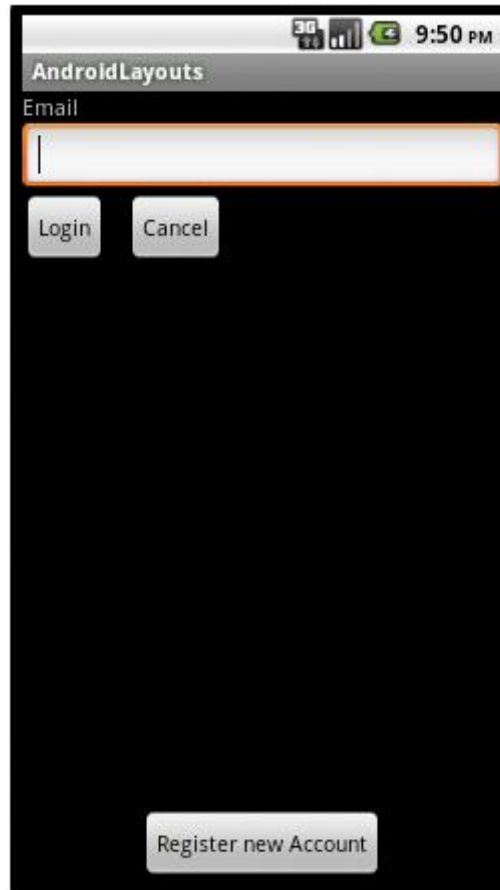
    <Button android:id="@+id/btnLogin" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/inputEmail"
        android:layout_alignParentLeft="true" android:layout_marginRight="10px"
        android:text="@string/login" />

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_toRightOf="@id/btnLogin"
        android:layout_alignTop="@id/btnLogin" android:text="@string/cancel" />

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" android:text="@string/register"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```



# Relative Layout



# Table Layout

F1 C1		
F2 C1	F2 C2	F2 C3
F3 C1		F3 C2



# Table Layout

<TableLayout

xmlns:android="<http://schemas.android.com/apk/res/android>"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:shrinkColumns="\*" android:stretchColumns="\*"

android:background="#ffffff">

<!-- Fila 1 con una sola columna -->

<TableRow

android:layout\_height="wrap\_content"

android:layout\_width="fill\_parent"

android:gravity="center\_horizontal">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="18dp" android:text="@string/row1"

android:layout\_span="3"

android:padding="18dip" android:background="#b0b0b0"

android:textColor="#000"/>

</TableRow>

# Table Layout

<!-- Fila 2 con 3 columnas -->

<TableRow

android:id="@+id/tableRow1"

android:layout\_height="wrap\_content"

android:layout\_width="match\_parent">

<TextView

android:id="@+id/TextView04" android:text="@string/row2col1"

android:layout\_weight="1" android:background="#dcdcdc"

android:textColor="#000000"

android:padding="20dip" android:gravity="center"/>

<TextView

android:id="@+id/TextView04" android:text="@string/row2col2"

android:layout\_weight="1" android:background="#d3d3d3"

android:textColor="#000000"

android:padding="20dip" android:gravity="center"/>

<TextView

android:id="@+id/TextView04" android:text="@string/row2col3"

android:layout\_weight="1" android:background="#cac9c9"

android:textColor="#000000"

android:padding="20dip" android:gravity="center"/>

</TableRow>

# Table Layout

<!-- Fila 3 con 2 columnas -->

<TableRow

android:layout\_height="wrap\_content"

android:layout\_width="fill\_parent"

android:gravity="center\_horizontal">

<TextView

android:id="@+id/TextView04" android:text="@string/row3col1"

android:layout\_weight="1" android:background="#b0b0b0"

android:textColor="#000000"

android:padding="20dip" android:gravity="center"/>

<TextView

android:id="@+id/TextView04" android:text="@string/row3col2"

android:layout\_weight="1" android:background="#a09f9f"

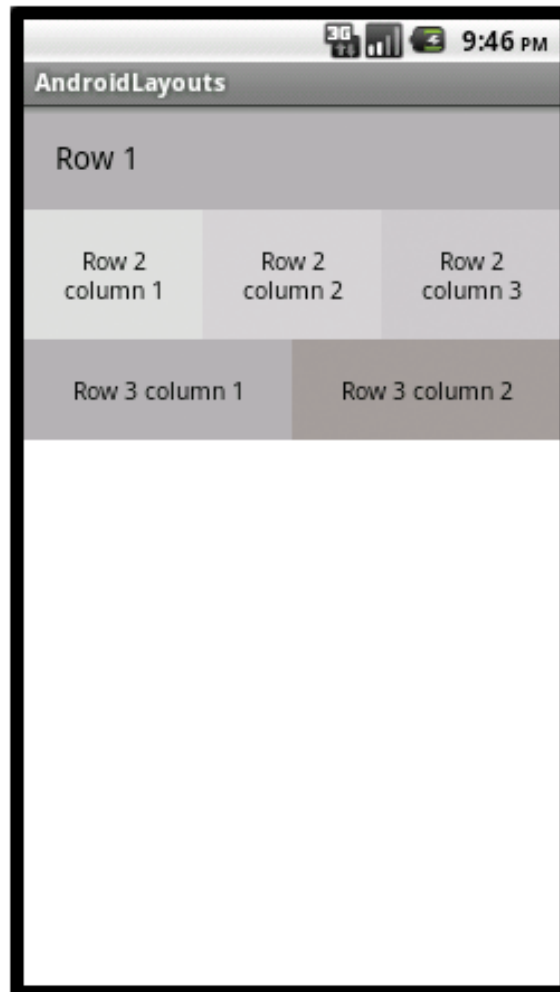
android:textColor="#000000"

android:padding="20dip" android:gravity="center"/>

</TableRow>

</TableLayout>

# Table Layout



Row 1		
Row 2 column 1	Row 2 column 2	Row 2 column 3
Row 3 column 1		Row 3 column 2

# ANDROID: SERVICIOS

# Servicios

- Son elementos que realizan alguna función pero generalmente no requieren de interfaz gráfica
- Existen 2 tipos de servicios
  - Unbounded
  - Bounded

## Servicios: Unbounded

- No están ligados a la existencia una aplicación, aunque ésta la cree.
- Debe iniciarse y finalizarse con
  - `startService()`
  - `stopService()`

## Servicios: Bounded

- Son servicios que requieren de vincularse con mínimo una aplicación para ejecutarse
- Al no tener vínculos el servicio desaparece cuando nadie lo usa
- Se requiere del uso de un `ServiceConnection` que determina lo que ocurre al vincular, desvincular un servicio



## Servicios: Bounded

- Para iniciarlo y desvincularlo (y finalizarlo si no hay otra actividad usándolo)
- `bindService()`
- `unbindService()`

# Ejemplo: Servicio Bounded

```
public class LocalService extends Service {  
  
    private final IBinder mBinder = new LocalBinder();  
    private final Random mGenerator = new Random();  
  
    public class LocalBinder extends Binder {  
        LocalService getService() {  
            return LocalService.this;  
        }  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        return mBinder;  
    }  
  
    public int getRandomNumber() {  
        return mGenerator.nextInt(100);  
    }  
}
```

# Ejemplo: Servicio Bounded

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, LocalService.class);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

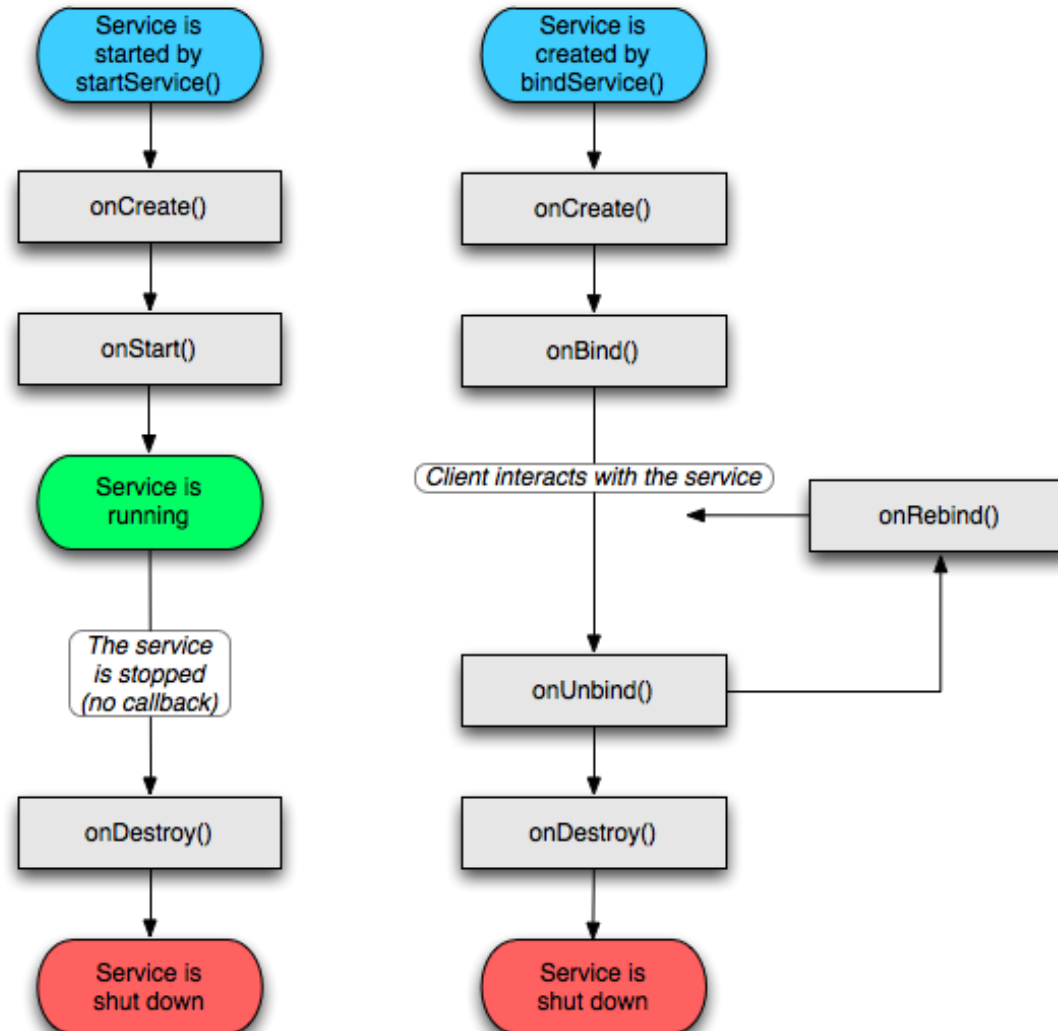
    @Override
    protected void onStop() {
        super.onStop();
        if (mBound) {
            unbindService(mConnection);
            mBound = false;
        }
    }
}
```

```

public void onClick(View v) {
    if (mBound) {
        int num = mService.getRandomNumber();
        Toast.makeText(this, "number: " + num,
            Toast.LENGTH_SHORT).show();
    }
}

private ServiceConnection mConnection = new
ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName
className,
        IBinder service) {
        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mBound = true;
    }
    @Override
    public void
onServiceDisconnected(ComponentName arg0) {
        mBound = false;
    }
};
}

```



# ANDROID: BROADCAST RECEIVERS

# Broadcast Receiver

- Se encarga de recibir mensajes broadcast o bien de una aplicación o bien del sistema operativo.
- Pueden crearse BroadCast Receivers personalizados.
- Deben registrarse:
  - Desde el Manifest
  - Desde una Actividad
    - EN ESTE CASO SIEMPRE HAY QUE QUITAR EL REGISTRO

# Ejemplo: BroadcastReceiver

```
public class ReceptorLlamada extends BroadcastReceiver{

    @Override
    public void onReceive(Context arg0, Intent arg1) {
        Bundle extras = arg1.getExtras();
        String estado =
extras.getString(TelephonyManager.EXTRA_STATE);

        if(estado.equals(TelephonyManager.EXTRA_STATE_RINGING)
){
            String numeroTel =
extras.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
            Log.d("Test_B_R",numeroTel);
        }
    }
}
```



# Registrando en Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    <!-- ..... -->
    <application
        <!-- ..... -->
        <activity

            android:name="co.com.compania.introcm.broadcastreceiver.DummyActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="ReceptorLlamada">
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE" >
                </action>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

# Registrando en Actividad

```
public class MainActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        IntentFilter filter = new  
IntentFilter(Intent.ACTION_SCREEN_ON);  
        filter.addAction(Intent.ACTION_SCREEN_OFF);  
        PowerDownReceiver receiver = new  
PowerDownReceiver();  
        registerReceiver(receiver, filter);  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
        unregisterReceiver(receiver);  
    }  
}
```

# ANDROID: CONTENT PROVIDERS

```

public class Main extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Cursor contactos = getResolver().query(
            ContactsContract.Contacts.CONTENT_URI,
            null,
            null,
            null,
            null);
        while(contactos.moveToNext()){
            int nombreIndice =
                contactos.getColumnIndex(PhoneLookup.DISPLAY_NAME);
            String nombre = contactos.getString(nombreIndice);
            Log.d("CONTENT_PROVIDER", nombre);
        }
    }
}

```

# Content Providers: Cursor

```
Cursor contactos = getContentResolver().query(  
    ContactsContract.Contacts.CONTENT_URI, //Donde se obtiene  
(FROM)  
    null, //Columnas (Arreglo de Strings) null para todas (SELECT)  
    null, // Filas a retornar (WHERE)  
    null, // Argumentos si se solicitan con = ? en la anterior; lista de  
Strings  
    null // Ordenamiento (ORDERBY)  
);
```

# Content Providers: Cursor

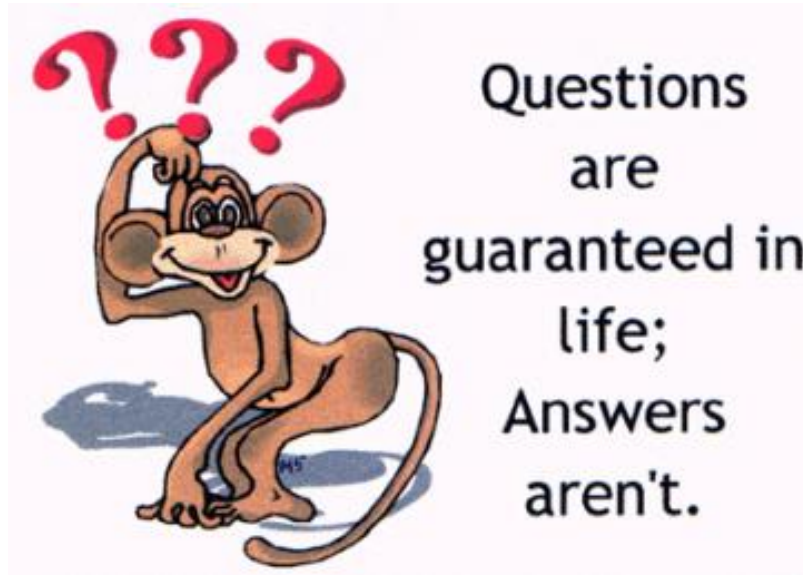
```
Cursor contactos = getContentResolver().query(  
    ContactsContract.Contacts.CONTENT_URI, //Donde se obtiene  
(FROM)  
    null, //Columnas (Arreglo de Strings) null para todas (SELECT)  
    null, // Filas a retornar (WHERE)  
    null, // Argumentos si se solicitan con = ? en la anterior; lista de  
Strings  
    null // Ordenamiento (ORDERBY)  
);
```

Una URI (RFC 2396) es una cadena de texto que permite identificar un recurso de información. Está formada así:

<standard\_prefix>://<authority>/<data\_path>/<id>

# Content Providers: Cursor Completo

```
String[] campos = new String[] {"codigo", "nombre"};
String[] args = new String[] {"usu1"};
Cursor contactos = getContentResolver().query(
    ContactsContract.Contacts.CONTENT_URI, //Donde se obtiene
    (FROM)
    campos, //Columnas (Arreglo de Strings) null para todas
    (SELECT)
    "usuario=?", // Filas a retornar (WHERE)
    args, // Argumentos si se solicitan con = ? en la anterior; lista de
    Strings
    campos[0] + " DESC" // Ordenamiento (ORDERBY)
);
```



# ¿Preguntas?

Diego Alberto Rincón Yáñez MSc.  
Twitter: @d1egoprog.