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:

FEUD
Fundación de egresados U. Distrital
Construyendo Profesionales

Desarrollo de aplicaciones utilizando SDK - Android

Diplomado Desarrollo de Software para móviles

Julio César Galeano García

2017

FEUD
Fundación de egresados U. Distrital
Construyendo Profesionales

Android

- Navigation Drawer y App Bar
- Servicios
- BroadcastReceiver
- ContentProvider (Base de datos internas)
- Uso del hardware del dispositivo
- Pruebas

Android: App Bar

La barra de acciones muestra el título de la actividad en uno de los laterales y un menú ampliado en el otro. Incluso en esta forma sencilla, también es útil para dar uniformidad a las aplicaciones.

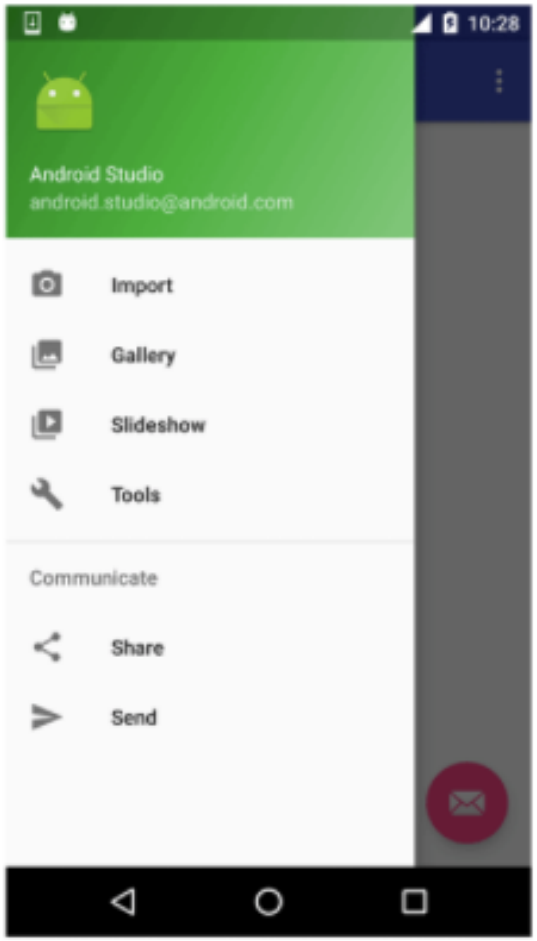
A partir de Android 3.0 (nivel de API 11), todas las actividades usan el ActionBar como barra de app. La ActionBar nativa se comporta de forma diferente según la versión del sistema Android que use un dispositivo.

Por este motivo, se debe usar Toolbar que tiene las funciones más recientes de la biblioteca de compatibilidad y están disponibles en todos los dispositivos que puedan usar la biblioteca de compatibilidad

https://developer.android.com/training/appbar/setting-up.html?hl=es

Android: Navigation Drawer

El panel lateral de navegación es un panel en el que se muestran las principales opciones de navegación de la app en el borde izquierdo de la pantalla. La mayor parte del tiempo está oculto, pero aparece cuando el usuario desliza un dedo desde el borde izquierdo de la pantalla o, mientras está en el nivel superior de la app, el usuario toca el ícono de la app en la barra de acciones.



Android: Navigation Drawer

```
public void onClick(View v) {
    new DownloadImageTask().execute("http://example.com/image.png");
}

private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    /** The system calls this to perform work in a worker thread and
      * delivers it the parameters given to AsyncTask.execute() */
    protected Bitmap doInBackground(String... urls) {
        return loadImageFromNetwork(urls[0]);
    }

    /** The system calls this to perform work in the UI thread and delivers
      * the result from doInBackground() */
    protected void onPostExecute(Bitmap result) {
        mImageView.setImageBitmap(result);
    }
}
```

Android: Servicios

Un Service es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario.

Un servicio continuará ejecutándose en segundo plano aunque el usuario cambie a otra aplicación.

Un servicio puede manejar transacciones de red, reproducir música, realizar I/O de archivos o interactuar con un proveedor de contenido, todo en segundo plano.

Existen dos tipos de servicio

- Unbounded
- Bounded

Android: Servicio Unbounded

Este servicio es iniciado y puede ejecutarse en segundo plano de manera indefinida, incluso si se destruye el componente que lo inició.

Por ejemplo, puede descargar o cargar un archivo a través de la red. Cuando la operación está terminada, el servicio debe detenerse por sí mismo.

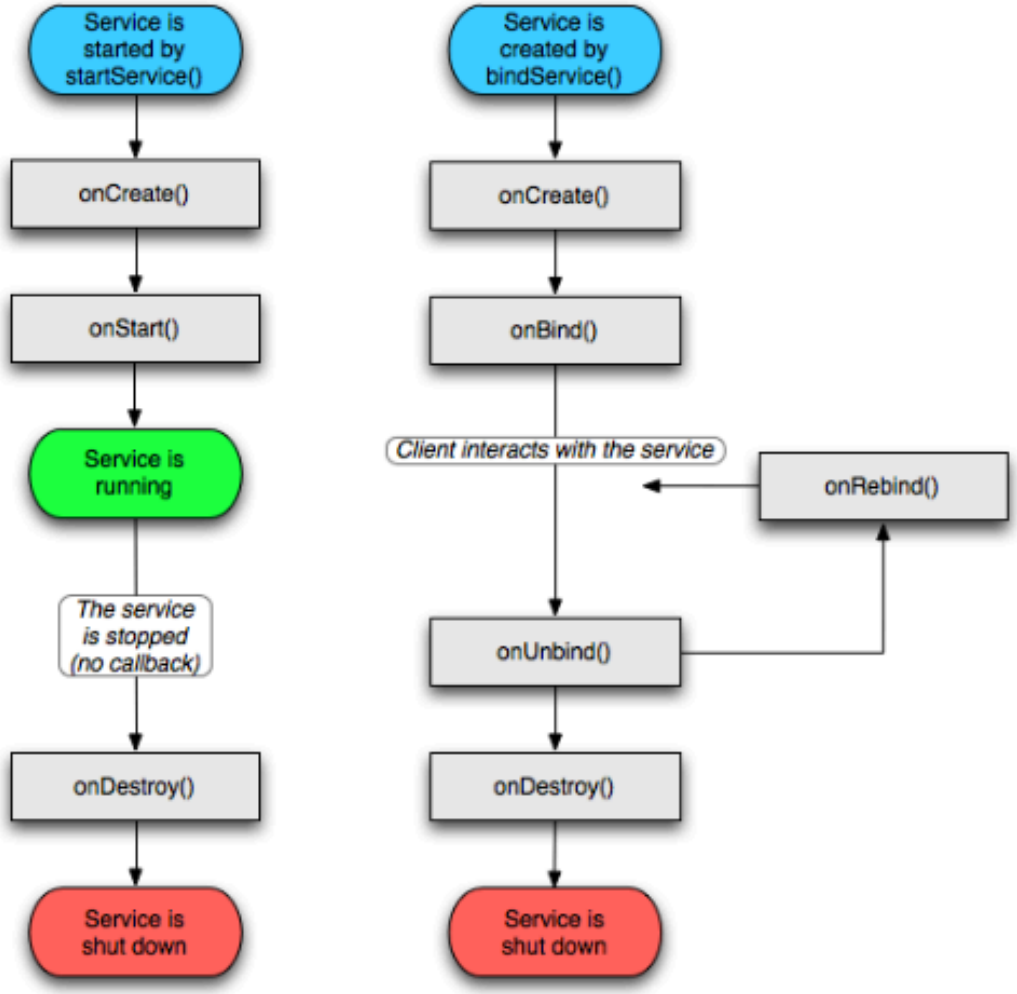
Deben iniciarse y finalizarse con

- startService()
- stopService()

Android: Servicio Bounded

- Son servicios que requieren 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.
- Para vincularlo y desvincularlo (y finalizarlo si no hay otra componente usándolo) se usa
  - bindService()
  - unbindService()

Android: Servicios



Android: BroadcastReceiver

Un broadcast receiver (receiver) es un componente de Android que le permite registrarse para eventos de sistema o de una aplicación.

Todos los receivers registrados son notificados por el runtime de Android una vez que sucede el evento.

Es posible crear BroadcastReceivers personalizados

Se pueden registrar en el manifest o desde una actividad, pero cuando se hace desde una actividad se debe quitar el registro manualmente.

Android: 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);
                }
        }
}
```

Android: BroadcastReceiver - 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>
```

Android: BroadcastReceiver

```
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: Ejercicio

Realizar ejercicio practico del tutorial a continuacion, numeral 6 y 7

http://www.vogella.com/tutorials/AndroidServices/article.html

Android: Content Provider

Los proveedores de contenido administran el acceso a un conjunto estructurado de datos. Encapsulan los datos y proporcionan mecanismos para definir la seguridad de los datos. Los proveedores de contenido son la interfaz estándar que conecta datos en un proceso con código que se ejecuta en otro proceso.

Cuando quieres acceder a datos en un proveedor de contenido, usas el objeto ContentResolver en el Context de tu aplicación para comunicarte con el proveedor como cliente. El objeto ContentResolver se comunica con el objeto del proveedor, una instancia de una clase que implementa ContentProvider. El objeto del proveedor recibe solicitudes de datos de clientes, realiza la acción solicitada y devuelve resultados.

Android: Content Provider

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


                Cursor contactos = getContentResolver().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);
                }
        }
}
```

Android: Content Provider

```
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)
);
```

Android: Content Provider

```
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)
);
```

Android: Content Provider

```
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)
);
```

Ejercicios

- Ejercicio de base de datos

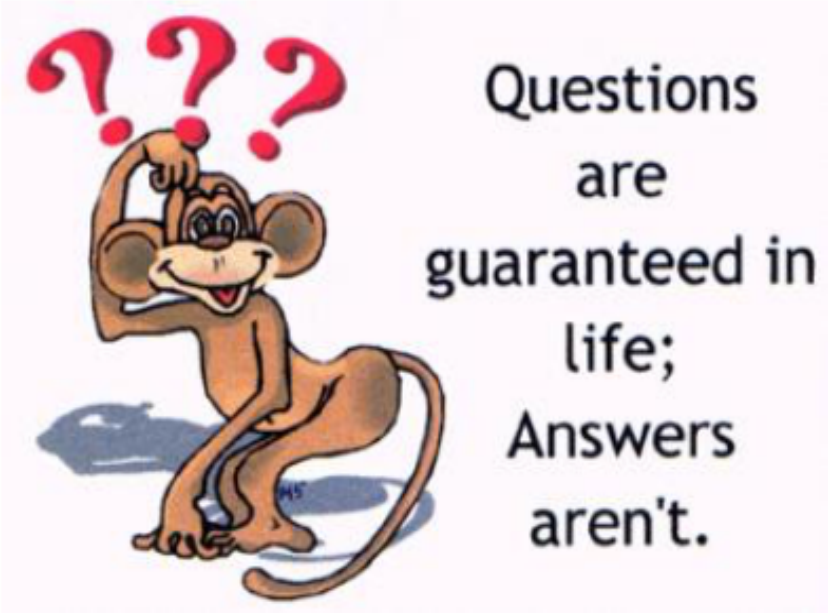
  https://www.101apps.co.za/articles/using-a-sqlite-database-in-android.html

- Ejercicio de creación de Content Provider para acceder a base de datos.

  https://www.101apps.co.za/articles/creating-our-custom-content-provider-part-1-building-the-provider.html

- Ejercicio de consumir el Content Provider con un Content Resolver

  https://www.101apps.co.za/articles/using-a-content-resolver-to-access-another-app-s-database.html



¿Preguntas?

Julio César Galeano García