

Android

Arquitectura Limpia

# Graddle

- Gradle es una herramienta para automatizar el proceso de construcción de nuestro proyecto(compilar,testing,empaquetado...). Esta basado en Groovy el cual tiene una sintaxis muy similar a la de java. **Groovy** es un lenguaje de programación orientados a objetos. Cuando compilamos nuestro software, gradle mira si hay cambios en el código fuente con respecto a la ultima compilación, de esta forma se ahorra la tarea de volver a compilar.

▼  HelloWorld


▶  .idea


▼  app


▶  build


 libs

▶  src


 .gitignore


 app.iml


 build.gradle


 proguard-rules.txt


▶  gradle


 .gitignore


 build.gradle


 gradle.properties


 gradlew

 gradlew.bat

 HelloWorld.iml

 local.properties

 settings.gradle

▶  External Libraries

## apply plugin: 'android'

```
android {
    compileSdkVersion 19
    buildToolsVersion "19.0.3"

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 19
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            runProguard false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
        }
    }
}

dependencies {
    compile 'com.android.support:appcompat-v7:19.+'
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
2.3 Gingerbread	10	97.4%
4.0 Ice Cream Sandwich	15	95.2%
4.1 Jelly Bean	16	87.4%
4.2 Jelly Bean	17	76.9%
4.3 Jelly Bean	18	73.9%
4.4 KitKat	19	40.5%
5.0 Lollipop	21	24.1%
5.1 Lollipop	22	4.7%
6.0 Marshmallow	23	

## Ice Cream Sandwich

### Contacts Provider

- Social APIs
- User profile
- Invite intent
- Large photos

### Calendar Provider

- Calendar APIs
- Event intents

### Voicemail Provider

- Add voicemails to the device

### Multimedia

- Media effects for images and videos
- Remote control client
- Improved media player

### Camera

- Face detection
- Focus and metering areas
- Continuous auto focus
- Camera broadcast intents

### Connectivity

- Android Beam for NDEF push with NFC
- Wi-Fi P2P connections
- Bluetooth health profile
- Network usage and controls

### Accessibility

- Explore-by-touch mode
- Accessibility for views
- Accessibility services
- Improved text-to-speech engine support

### User Interface

- Spell checker services
- Improved action bar
- Grid layout
- Texture view
- Switch widget
- Improved popup menus
- System themes
- Controls for system UI visibility
- Hover event support
- Hardware acceleration for all windows

### Enterprise

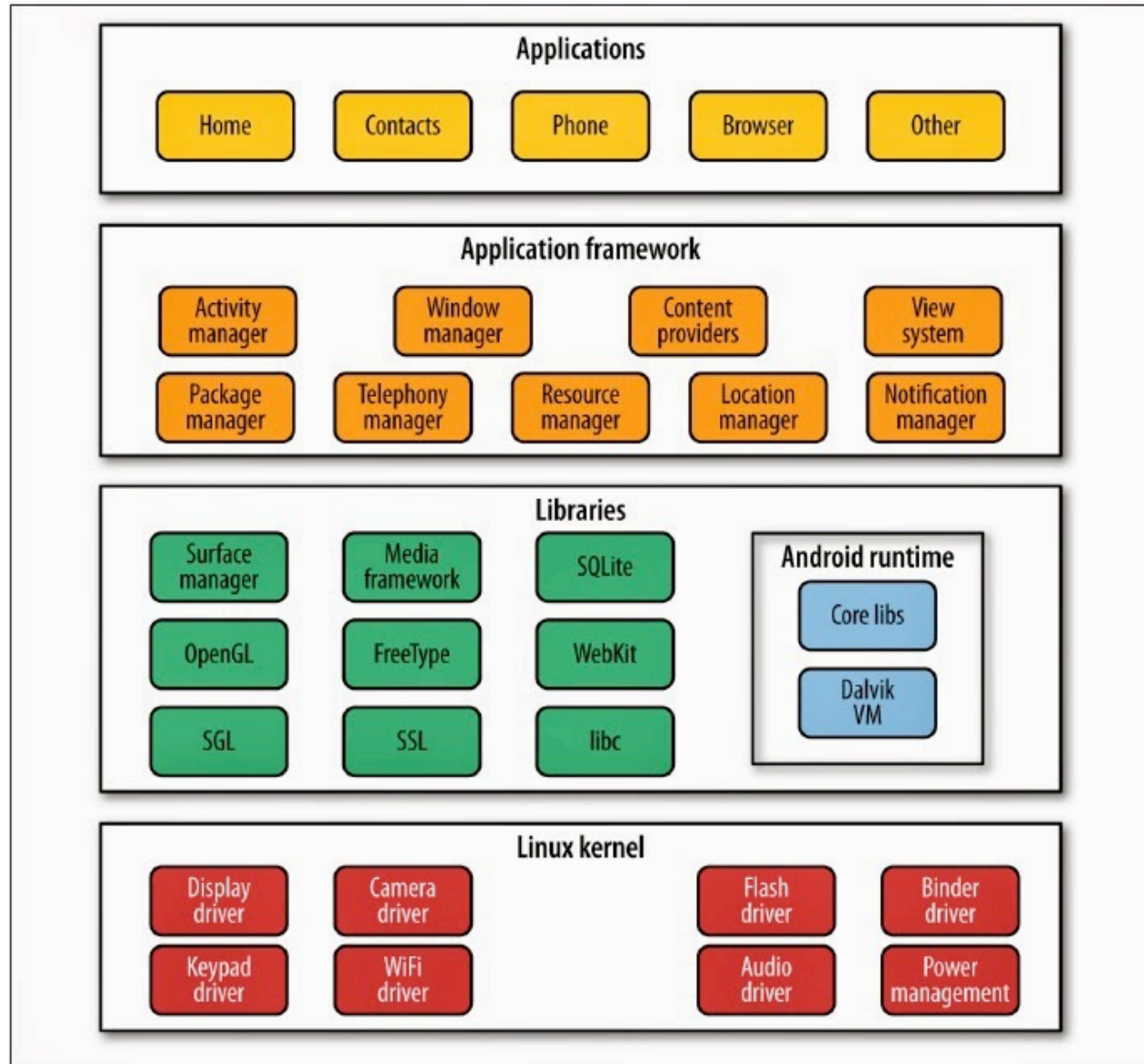
- VPN services
- Device policies
- Certificate management

### Device Sensors

- Improved sensors
- Temperature sensor
- Humidity sensor

<https://developer.android.com/about/versions/android-4.0.html>

# Android



## Application Building Blocks



### Activity

- UI Component Typically Corresponding to one screen.

### IntentReceiver

- Responds to notifications or status changes. Can wake up your process.

### Service

- Faceless task that runs in the background.

### ContentProvider

- Enable applications to share data.



MyApp/

Project

build.gradle

settings.gradle

app/

Module

build.gradle

build/

libs/

src/

main/

Sourceset

java/

com.example.myapp/

res/

drawable/

layout/

...

AndroidManifest.xml

**Carpeta build:** los elementos que contiene son códigos generados automáticamente por Android Studio cada vez que se realiza la compilación de nuestro proyecto.

**Carpeta libs:** contiene las librerías Java externas que utiliza nuestra aplicación. Android Studio hace referencia a estas librerías en el fichero build.gradle.

**Carpeta src:** contiene la información más importante y será la que estudiaremos para entender la estructura de una aplicación de Android.

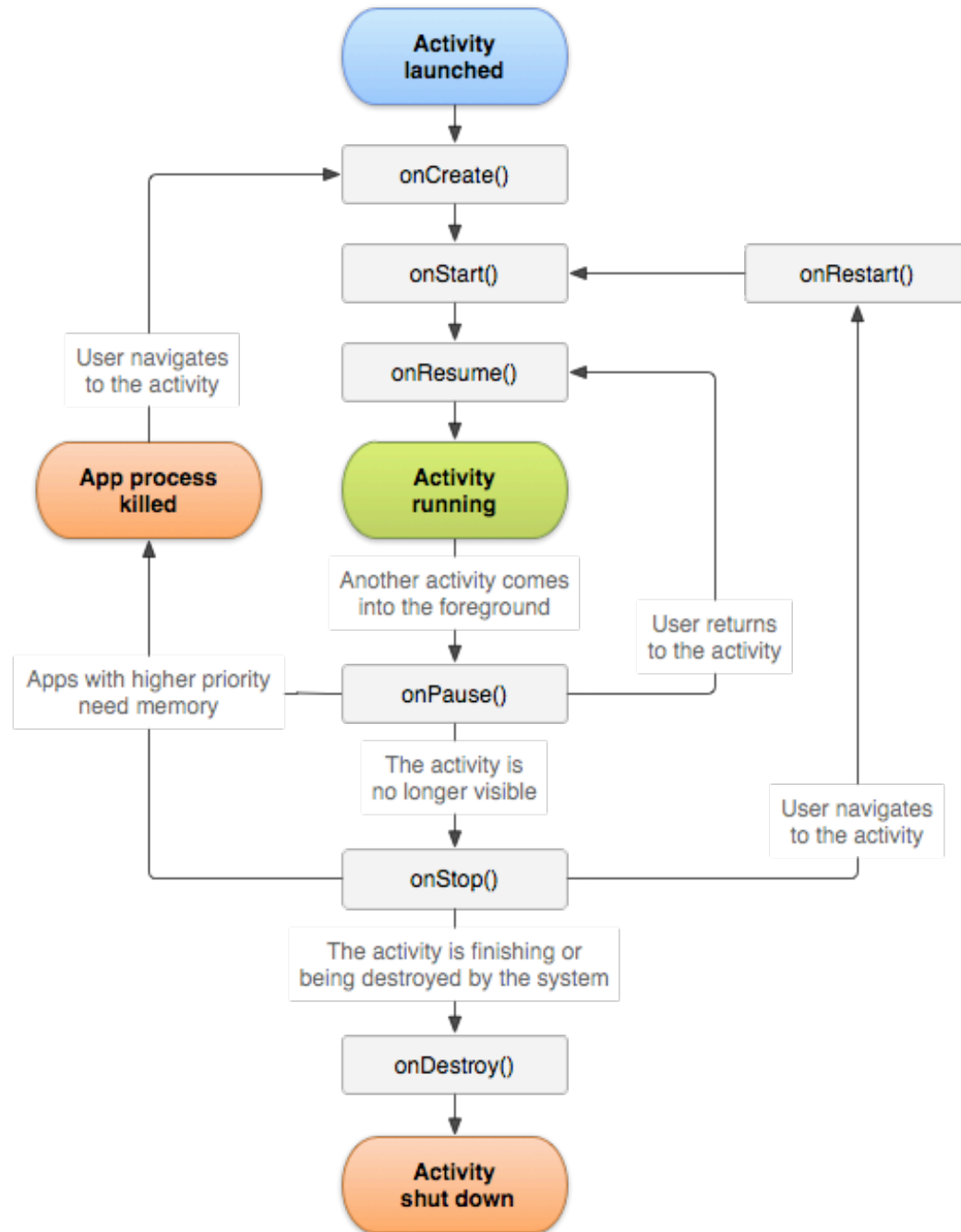
**Carpeta drawable:** aquí agregaremos las imágenes que utilizaremos en nuestra aplicación.

**Carpeta layout:** aquí encontraremos el archivo activity\_main.xml. Este contiene el diseño de la interfaz de nuestra actividad principal.

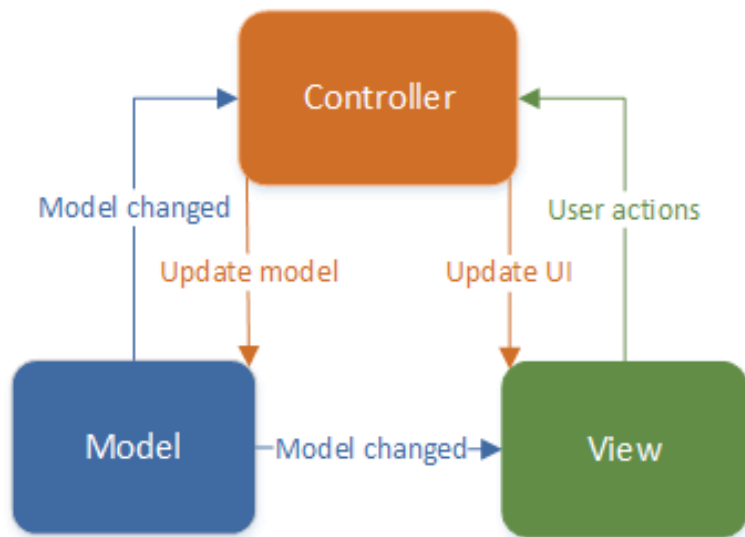
**Carpeta menú:** esta carpeta es creada de forma automática y contendrá un menú básico para nuestra actividad.

**Carpeta values:** aquí encontraremos los archivos dims.xml, styles.xml y strings.xml.

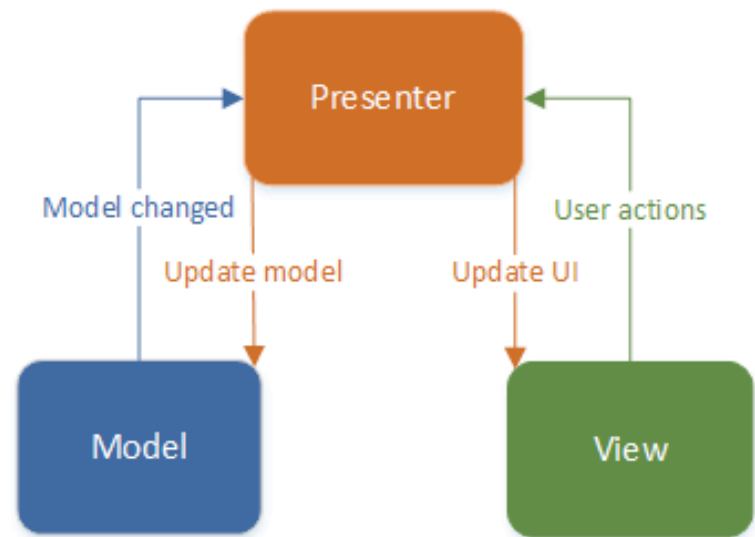
# Ciclo de vida activity



# MVC



# MVP

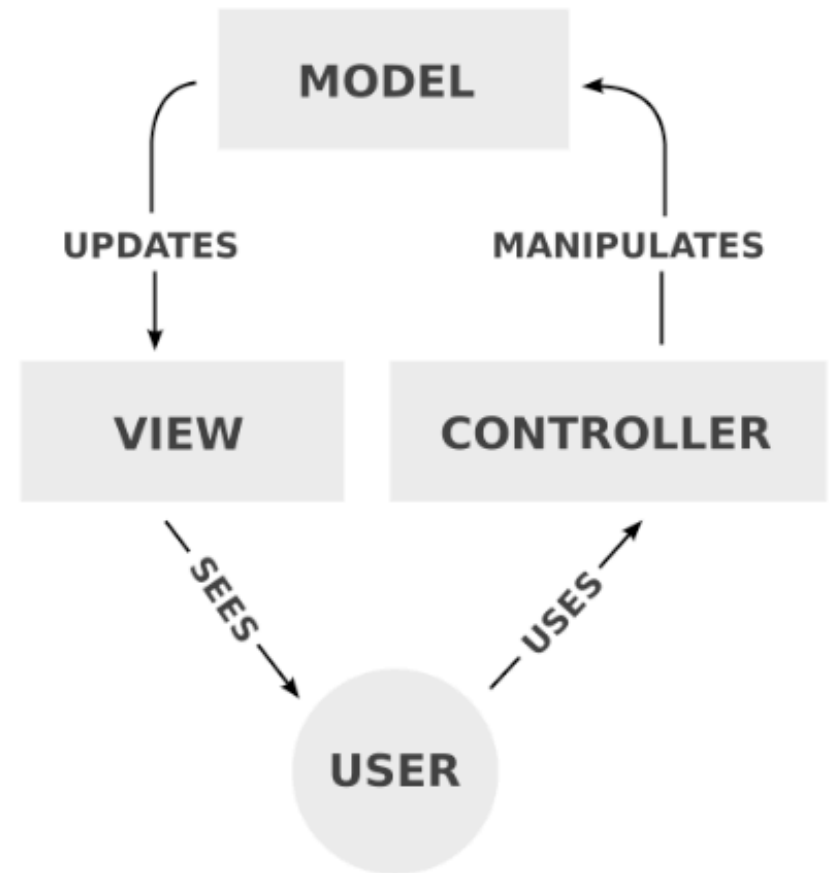


# MVC

Model: What to render

View: How to render

Controller: Events, user input



<https://platzi.com/blog/arquitectura-android-app/>

# ButterKnife

```
compile 'com.jakewharton:butterknife:6.1.0'
```

```
@InjectView(R.id.sample_textview)  
TextView sample_textview;
```

```
ButterKnife.inject(this);
```

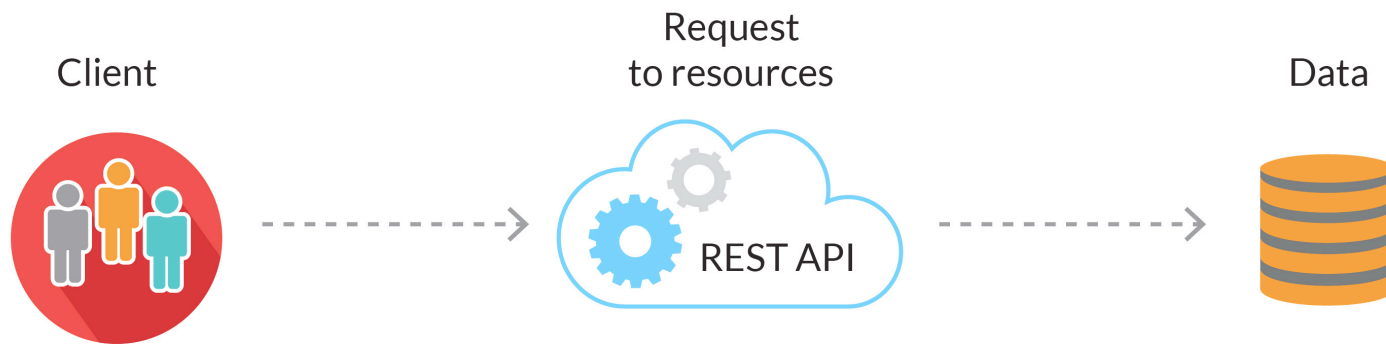
```
@OnClick(R.id.sample_textview)  
public void showToastMessage(){  
    Toast.makeText(MainActivity.this, "This is a message from the activity",  
    Toast.LENGTH_SHORT).show();  
}
```

Apps conectadas a  
Internet





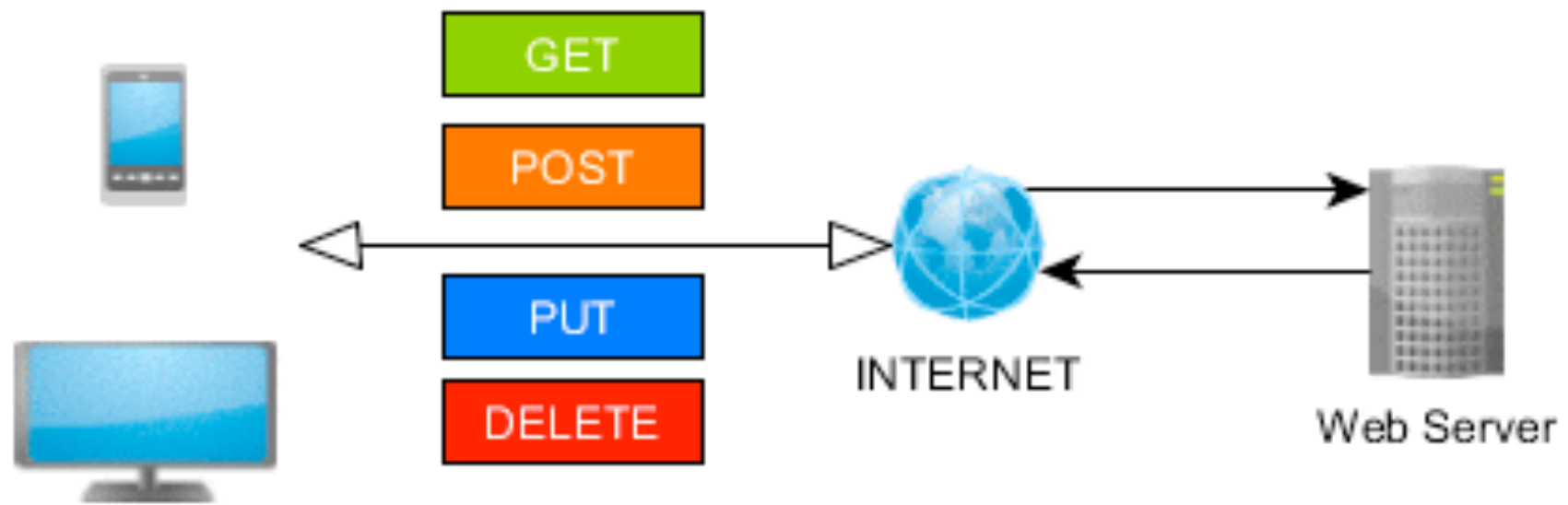
# HTTP / REST



Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.

Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

Los objetos en REST siempre se manipulan a partir de la URI.



# Recursos

<https://jsonplaceholder.typicode.com/photos>

<https://jsonplaceholder.typicode.com/users>

```
curl https://jsonplaceholder.typicode.com/photos
```

# CURL

Curl es una librería de funciones para conectar con servidores para trabajar con ellos. El trabajo se realiza con formato URL. Es decir, sirve para realizar acciones sobre archivos que hay en URLs de Internet, soportando los protocolos más comunes, como http, ftp, https, etc.

# JSON

JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript



```
BufferedReader reader = new BufferedReader(new  
InputStreamReader(httpResponse.getEntity().getContent(),  
"UTF-8"));
```

```
String json = reader.readLine();
```

```
// Instantiate a JSON object from the request response  
JSONObject jsonObject = new JSONObject(json);
```



# Necesario

```
<uses-permission  
android:name="android.permission.INTERNET"/>
```

```
useLibrary 'org.apache.http.legacy'
```

# AsyncTask

Permite ejecutar tareas en segundo plano

**onPreExecute()** : en foreground. Acceso al UI

**doInBackground(Parametros...)** : Sin acceso a UI

**onProgressUpdate(Progress...)** : Avisa del cambio para loaders

**onPostExecute(Result)** : Después de el background.  
Acceso a UI

```
class MiTarea extends AsyncTask {  
    @Override  
    protected Integer doInBackground(Integer... n) {  
        return factorial(n[0]);  
    }  
    @Override  
    protected void onPostExecute(Integer res) {  
        salida.append(res + "\n");  
    }  
}
```

```
MiTarea tarea = new MiTarea();  
tarea.execute(n);
```

# Volley

**Volley** es una librería desarrollada por Google para optimizar el envío de peticiones Http desde las aplicaciones Android hacia servidores externos.

```
compile 'com.android.volley:volley:1.0.0'
```

- Procesamiento concurrente de peticiones.
- Priorización de las peticiones, lo que permite definir la preponderancia de cada petición.
- Cancelación de peticiones, evitando la presentación de resultados no deseados en el hilo principal.
- Gestión automática de trabajos en segundo plano, dejando de lado la implementación manual de un framework de hilos.
- Implementación de caché en disco y memoria.
- Capacidad de personalización de las peticiones.

```
JSONArrayRequest jsArrayRequest = new JSONArrayRequest(
    Request.Method.GET,
    "http://example.json",
    null,
    new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONObject response) {
            items = parseJson(response);
            notifyDataSetChanged();
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.d(TAG, "Error Respuesta en JSON: " + error.getMessage());
        }
    }
);
```

# NetworkImageView

- Este view se puede usar en reemplazo del típico **ImageView**. La ventaja está en que el contenido de la imagen está directamente relacionado con la petición URL, lo que permite un manejo optimizado en la red y posibilitar cancelar la visualización en cualquier momento.

# Glide

```
compile 'com.github.bumptech.glide:glide:3.7.0'
```

```
ImageView imageView = (ImageView) findViewById(R.id.my_image_view);
```

```
Glide.with(this).load("http://goo.gl/gEgYUd").into(imageView);
```



# SharedPreferences

```
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

```
SharedPreferences.Editor editor = prefs.edit();  
editor.putString("email", "modificado@email.com");  
editor.putString("nombre", "Prueba");  
editor.commit();
```

```
String language = prefs.getString("language", "");
```

Comunicando tu  
aplicación

# long-time running tasks

**AsyncTask** : Se mueren con las Actividades / Fragmentos, son para tareas cortas.

## Service e IntentService

### **Service:**

- Un Servicio es un componente de la aplicación sin interfaz.
- Como componente de la aplicación, se ejecuta EN EL HILO PRINCIPAL de la aplicación.
- Para usarlo en segundo plano, hay que crear un nuevo hilo dentro del servicio.
- Puede vincularse (“bind”) a un Activity.
- Se inician y se paran mediante Intents, ya sean explícitos o de acción
- Es apto para aplicaciones como VOIP u otras que tengan que quedar permanentemente en segundo plano.

## **IntentService:**

- Un IntentService es un tipo especial de servicio que se ejecuta en segundo plano.
- Esta pensado para realizar una tarea que requiera un largo procesamiento en segundo plano y después pararse solo.
- Su funcionamiento está desaconsejado para realizar tareas que tengan que ejecutarse indefinidamente.

```
<application  
  android:name="@string/app_name"  
  android:theme="@android:style/Holo">
```

.....

```
  <service  
    name="com.myapp.id.MyService"  
    label="@string/service_label">  
    <intent-filter>  
      <action android:name =  
"com.myapp.id.action.ACTION_OPEN_SERVICE"/>  
    </intent-filter>  
  </service>
```

.....

```
</applicaton>
```

```
Intent intent = new Intent(this, MyService.class);  
startService(intent);
```

```
Intent intent = new  
Intent("com.myapp.id.action.ACTION_OPEN_SERVICE");  
startService(intent);
```

```
public class DownloadService extends Service {

    public DownloadService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        Log.d(TAG, "Servicio creado...");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "Servicio iniciado...");

        return START_NOT_STICKY;
    }

    @Override
    public void onDestroy() {
        Log.d(TAG, "Servicio destruido...");
    }

}
```

**START\_STICKY:** Crea de nuevo el servicio después de haber sido destruido por el sistema. En este caso llamará a onStartCommand() referenciando un intent nulo.

**START\_REDELIVER\_INTENT:** Crea de nuevo el servicio si el sistema lo destruyó. A diferencia de START\_STICKY, esta vez sí se retoma el último intent que recibió el servicio.



AlarmManager

# Servicio de alarmas

- Permite ejecutar acciones en momentos específicos aunque la app no este corriendo.

# RxJava en Android

# RXJava

- Es una librería que facilita la programación asíncrona y de eventos. Se basa en el uso de Observadores que nos ayudan a no tener problemas de threading y sincronización
- Alternativa a AsyncTask. Usada por Retrofit

# Bases

```
public interface Observer<T> {  
    void onComplete();  
    void onError(Throwable e);  
    void onNext(T t);  
}
```

- Observer
- Observable

```
public class Observable<T> {  
    ...  
}
```