

Introducción a Android

Profesor: Ana Isabel Vegas

INDICE

1. ANDROID STUDIO	3
JDK 8.....	3
ANDROID STUDIO	4
ACTUALIZAR SDK MANAGER	4
ACTUALIZACIONES.....	7
2. CREAR EL PRIMER PROYECTO CON ANDROID STUDIO	8
3. ESTRUCTURA DE UN PROYECTO CON ANDROID STUDIO	11
CARPETA JAVA.....	12
CARPETA RES	12
FICHERO ANDROIDMANIFEST.XML	13
4. COMPONENTES DE UNA APLICACIÓN ANDROID	14
ACTIVITY	14
VIEW.....	14
SERVICE.....	15
CONTENT PROVIDER	15
BROADCAST RECEIVER.....	15
WIDGET	16
INTENT	16
5. EJEMPLO HOLA MUNDO PERSONALIZADO	17
ÍNDICE DE GRÁFICOS	24

1. ANDROID STUDIO

Android Developer recomienda Android Studio como el único IDE para desarrollar aplicaciones con Android, ¿por qué?

La siguiente tabla muestra las diferencias entre ADT y Android Studio:



Feature	Android Studio	ADT
Build system	Gradle 	Ant 
Maven-based build dependencies	Yes	No
Build variants and multiple-APK generation (great for Android Wear)	Yes	No
Advanced Android code completion and refactoring	Yes	No
Graphical layout editor	Yes	Yes
APK signing and keystore management	Yes	Yes
NDK support	Coming soon	Yes

Gráfico 1. Android Studio vs ADT

JDK 8

Android Studio requiere tener instalado previamente JDK 7 o posterior. Mi recomendación es instalar JDK 8.












Java SE Development Kit 8u40		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux x86	146.84 MB	 jdk-8u40-linux-i586.rpm
Linux x86	166.85 MB	 jdk-8u40-linux-i586.tar.gz
Linux x64	145.14 MB	 jdk-8u40-linux-x64.rpm
Linux x64	165.17 MB	 jdk-8u40-linux-x64.tar.gz
Mac OS X x64	221.9 MB	 jdk-8u40-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.09 MB	 jdk-8u40-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.68 MB	 jdk-8u40-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	130.57 MB	 jdk-8u40-solaris-x64.tar.Z
Solaris x64	89.91 MB	 jdk-8u40-solaris-x64.tar.gz
Windows x86	175.71 MB	 jdk-8u40-windows-i586.exe
Windows x64	180.19 MB	 jdk-8u40-windows-x64.exe

Gráfico 2. Descarga jdk 8

Una vez descargado de la pagina de Oracle tan solo hay que ejecutar el archivo.

ANDROID STUDIO

Ahora nos descargamos Android Studio desde la pagina developers.android.

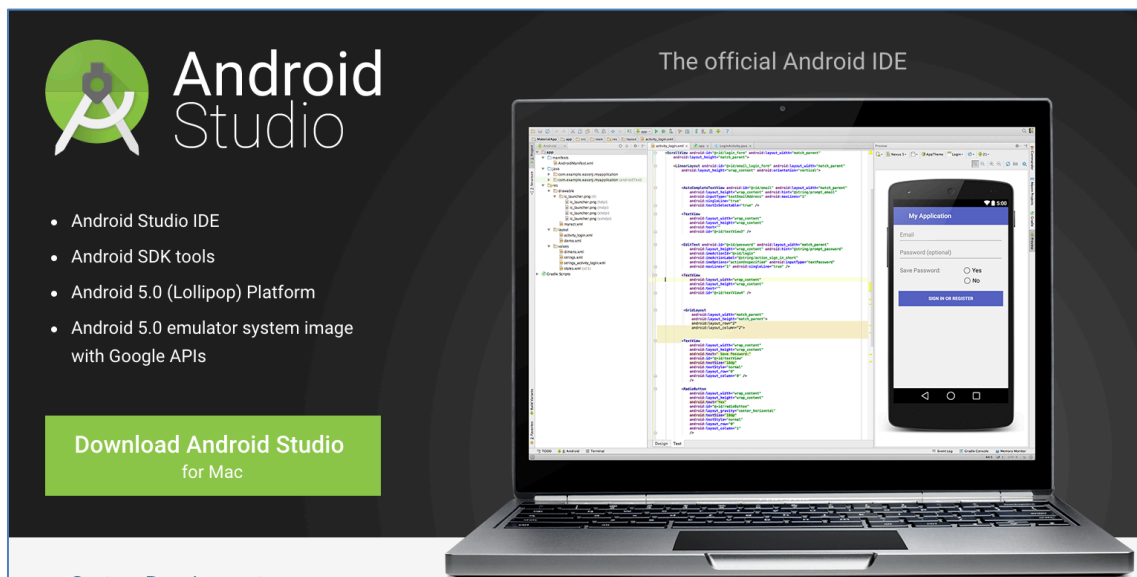


Gráfico 3. Descarga Android Studio

Una vez instalado debemos asegurarnos de tener descargadas las librerías necesarias para la versión a desarrollar.

ACTUALIZAR SDK MANAGER

En la pantalla inicial seleccionamos Configure ->

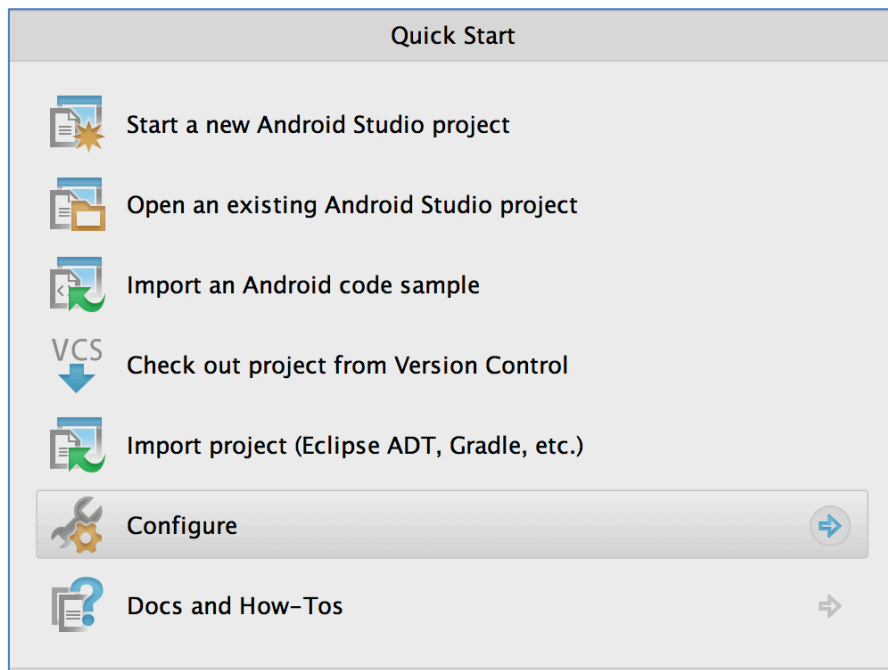


Gráfico 4. Opción Configure

Pulsar sobre SDK Manager

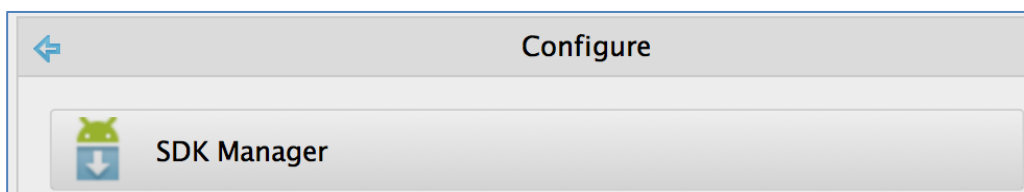


Gráfico 5. SDK Manager

Aquí vemos las versiones instaladas las que todavía no lo están.

Name	API	Rev.	Status
Tools			
Android SDK Tools		24.4.1	Installed
Android SDK Platform-tools		23.1	Installed
Android SDK Build-tools		23.0.2	Installed
Android SDK Build-tools		23.0.1	Installed
Android SDK Build-tools		22.0.1	Installed
Android SDK Build-tools		21.1.2	Installed
Android SDK Build-tools		20	Installed
Android SDK Build-tools		19.1	Installed
Tools (Preview Channel)			
Android SDK Tools		25.0.7	Not installed
Android 6.0 (API 23)			
Documentation for Android SDK	23	1	Installed
SDK Platform	23	2	Installed
Samples for SDK	23	2	Installed
Android TV ARM EABI v7a System Image	23	2	Installed
Android TV Intel x86 Atom System Image	23	2	Installed
Android Wear ARM EABI v7a System Image	23	2	Installed
Android Wear Intel x86 Atom System Image	23	2	Installed
ARM EABI v7a System Image	23	3	Installed
Intel x86 Atom_64 System Image	23	8	Installed
Intel x86 Atom System Image	23	8	Installed
Google APIs	23	1	Installed
Google APIs ARM EABI v7a System Image	23	7	Installed
Google APIs Intel x86 Atom_64 System Image	23	12	Installed
Google APIs Intel x86 Atom System Image	23	12	Installed
Sources for Android SDK	23	1	Installed

Gráfico 6. SDK Manager

Los paquetes que nos hemos descargado están en la ruta que vemos desde:
Configure -> Project Defaults -> Project Structure

Ahí nos muestra la ruta de la carpeta sdk y el jdk que estamos utilizando.

SDK Location

Android SDK location:
The directory where the Android SDK is located. This location will be used for new projects, and for existing projects that do not have a local.properties file with a sdk.dir property.

JDK location:
The directory where the Java Development Kit (JDK) is located.

Gráfico 7. SDK Location

ACTUALIZACIONES

Para asegurarnos que Android Studio se mantiene actualizado en la parte inferior pulsamos sobre el enlace check y nos dice si ya los esta.

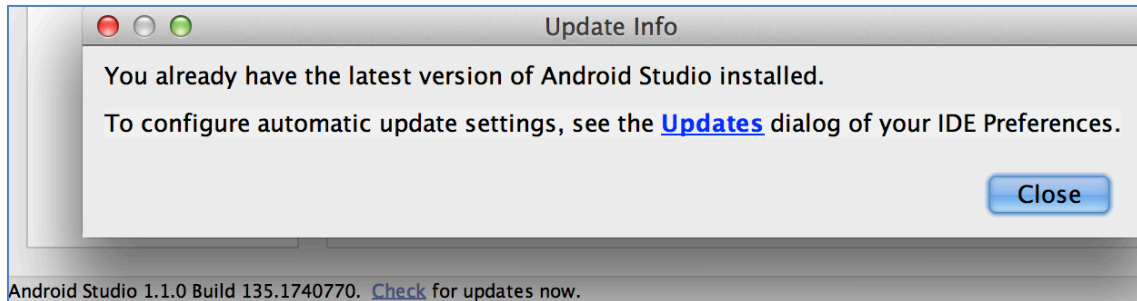


Gráfico 8. Comprobar actualizaciones

Pulsamos sobre Updates y elegimos Canary Channel que es el que me aporta las ultimas novedades.

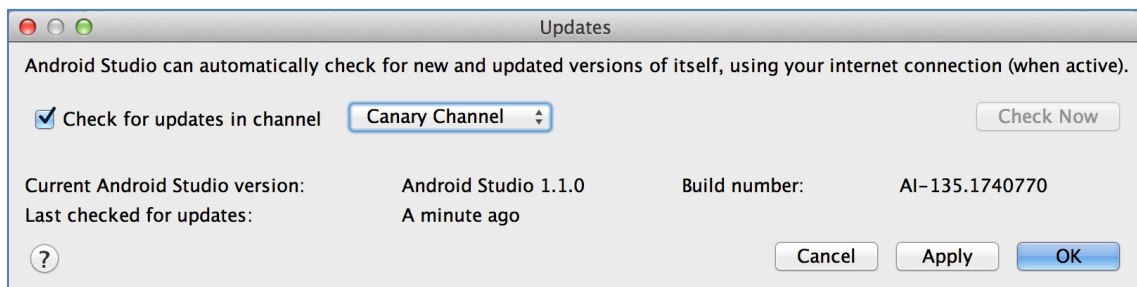


Gráfico 9. Canary Channel

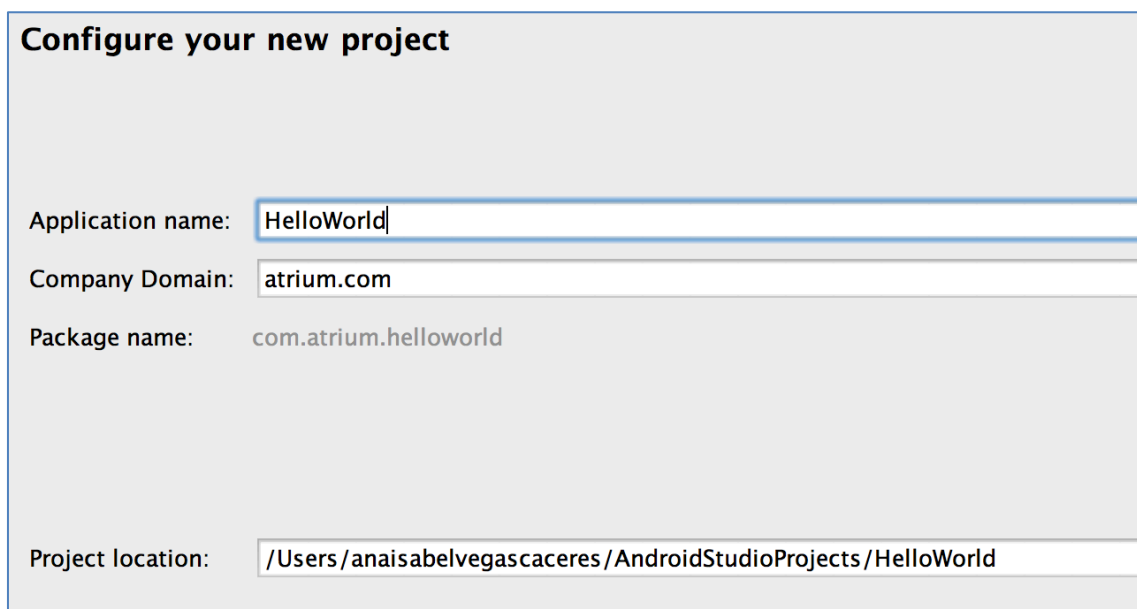
2. CREAR EL PRIMER PROYECTO CON ANDROID STUDIO

Application name: nombre del proyecto

Company Domain: dominio de la empresa para la que desarrollamos la aplicación. De esta forma los SEO nos ayudan a localizar sus aplicaciones y a la vez obtenemos un nombre de paquete único que es el que utiliza Google Play como id de la aplicación.

Package name: paquete que genera

Project location: ubicación del proyecto.



The screenshot shows the 'Configure your new project' dialog box in Android Studio. It contains four input fields with the following values:

Field	Value
Application name:	HelloWorld
Company Domain:	atrium.com
Package name:	com.atrium.helloworld
Project location:	/Users/anaisabelvegascaceres/AndroidStudioProjects/HelloWorld

Gráfico 10. Crear un proyecto

Al pulsar en Next elijo para que tipo de dispositivo quiero desarrollar: teléfonos y tablets, tv o wear.

Para ver los usos pulso sobre el link Help me choose

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
2.2 Froyo	8	99,5%
2.3 Gingerbread	10	90,4%
4.0 Ice Cream Sandwich	15	82,6%
4.1 Jelly Bean	16	61,3%
4.2 Jelly Bean	17	40,9%
4.3 Jelly Bean	18	33,9%
4.4 KitKat	19	< 0.1%
5.0 Lollipop	21	

Gráfico 11. Uso de las diferentes versiones

Elijo Android 6 ya que tiene retro-compatibilidad.

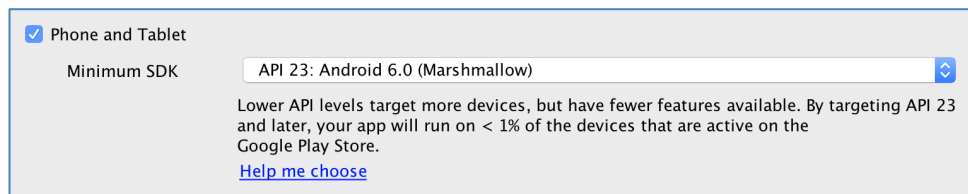


Gráfico 12. Mínimo SDK

En la siguiente pantalla elegimos un blank activity.



Gráfico 13. Blank Activity

Activity Name:	MainActivity
Layout Name:	activity_main
Title:	MainActivity
Menu Resource Name:	menu_main

Gráfico 14. Datos de la activity



Todo el código de este ejemplo lo encontrareis en **HelloWorld.zip**

La explicación de este tema está en el video **Primer proyecto en Android Studio**

3. ESTRUCTURA DE UN PROYECTO CON ANDROID STUDIO

La siguiente imagen muestra la estructura del proyecto creado con Android Studio:

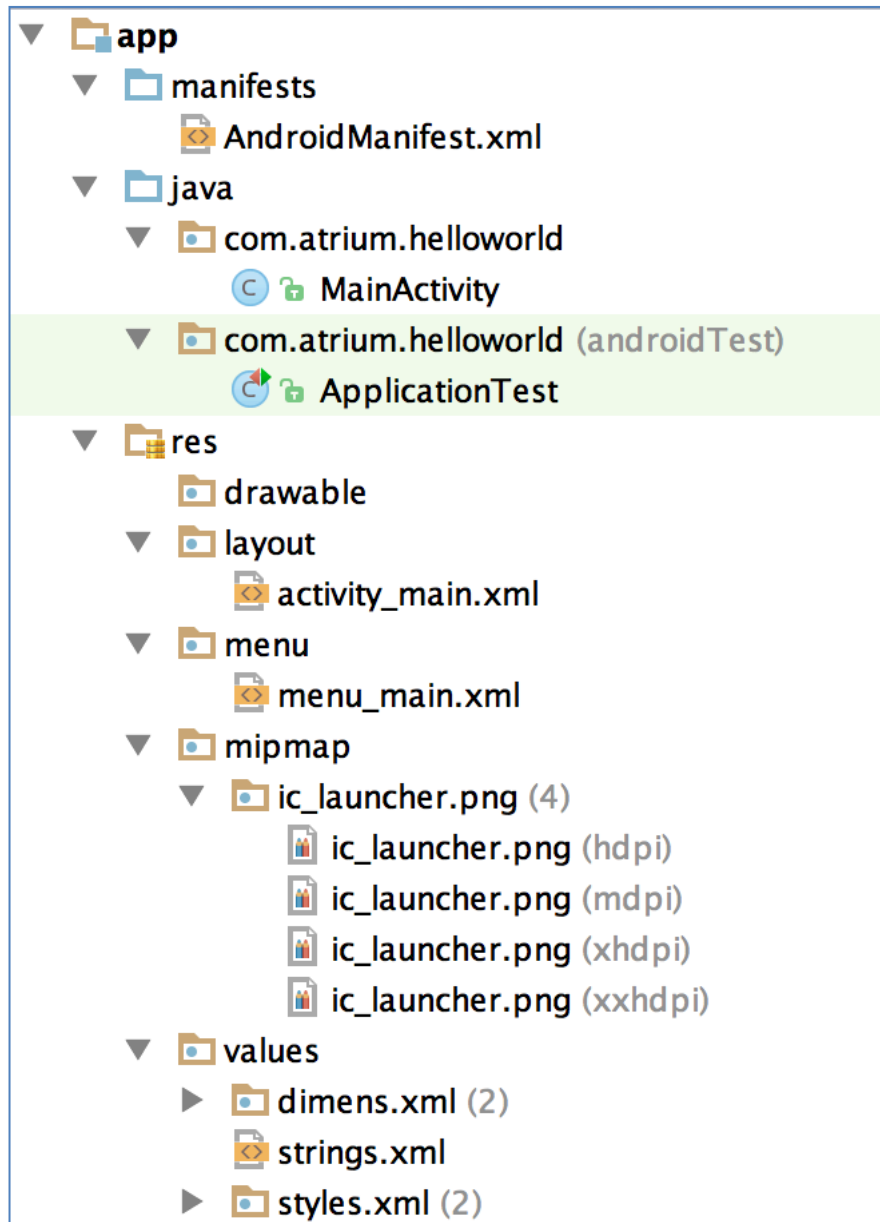


Gráfico 15. Estructura proyecto en Android Studio

CARPETA JAVA

Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc.

Inicialmente, Android Studio creará por nosotros el código básico de la pantalla (Activity) principal de la aplicación, siempre bajo la estructura del paquete java definido.

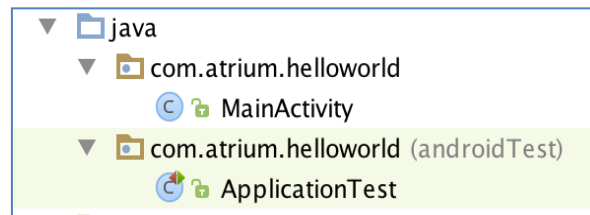


Gráfico 16. Carpeta src

CARPETA RES

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos se deberán distribuir entre las siguientes carpetas:

- `/res/drawable/`. Contienen las imágenes de la aplicación. Se puede dividir en `/drawable-ldpi`, `/drawable-mdpi` y `/drawable-hdpi` para utilizar diferentes recursos dependiendo de la resolución del dispositivo.
- `/res/layout/`. Contienen los ficheros xml de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en `/layout` y `/layout-land` para definir distintos layouts dependiendo de la orientación del dispositivo.
- `/res/menu/`. Contiene la definición en xml de los menús de la aplicación.
- `/res/mipmap/`. Aquí se almacenan los iconos de la aplicación en diferentes tamaños.
- `/res/values/`. Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (`strings.xml`), estilos (`styles.xml`), colores (`colors.xml`), etc.

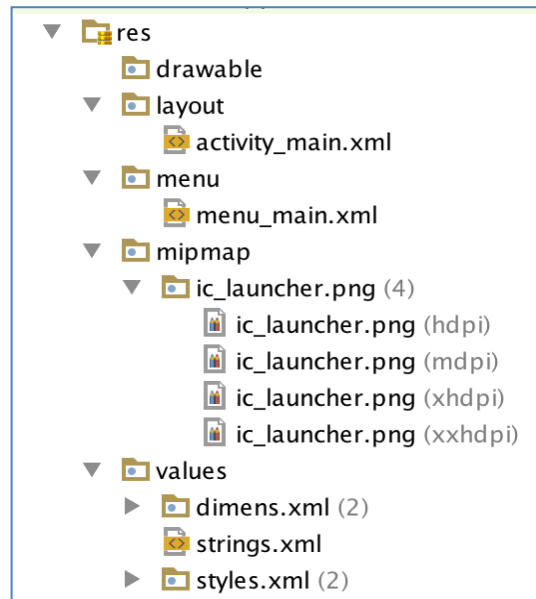


Gráfico 17. Carpeta res

FICHERO ANDROIDMANIFEST.XML

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, ...), sus componentes (pantallas, mensajes, ...), o los permisos necesarios para su ejecución.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="HelloWorld" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Gráfico 18. Fichero AndroidManifest.xml

4. COMPONENTES DE UNA APLICACIÓN ANDROID

Existen una serie de elementos clave que resultan imprescindibles para desarrollar aplicaciones con Android.

- Activity
- View
- Service
- Content Provider
- Broadcast Receiver
- Widget
- Intent

ACTIVITY

Las actividades (activities) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana en cualquier otro lenguaje visual.

VIEW

Los objetos view son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análogo por ejemplo a los controles de Java.

De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Se definen en un fichero XML similar a lo que sería la definición de una página web en HTML.

SERVICE

Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano.

En concepto, son exactamente iguales a los servicios presentes en cualquier otro sistema operativo (demonios en Unix o servicios Windows).

Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (activities) si se necesita en algún momento la interacción con del usuario.

En Android disponemos de dos tipos de servicios:

- Servicios locales; que pueden ser utilizados por aplicaciones del mismo terminal.
- Servicios remotos; pueden ser utilizados desde otros terminales.

CONTENT PROVIDER

Compartir información entre teléfonos móviles resulta un tema vital.

Android define un mecanismo estándar para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros.

Con este mecanismo podremos acceder a datos de otras aplicaciones, como la lista de contactos, o proporcionar datos a otras aplicaciones.

BROADCAST RECEIVER

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, ...) o por otras aplicaciones (cualquier aplicación puede generar mensajes (intents, en terminología Android) broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

WIDGET

Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

INTENT

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente.

Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones.

Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.



RECUERDA QUE...

- Los componentes de aplicación son los bloques esenciales de una aplicación Android.
- Cada componente es un punto a través del cual el sistema puede entrar en la aplicación.

5. EJEMPLO HOLA MUNDO PERSONALIZADO

Vamos a crear un proyecto en el cual se le presente al usuario una pantalla con una caja de texto donde introduzca su nombre y un botón.

Al pulsar el botón se le enviará a otra pantalla donde se mostrará un mensaje de saludo personalizado, con el nombre escrito por el usuario.

Para la realización de este ejemplo vamos a seguir unos pasos que detallamos a continuación:

Paso 1. Crear el proyecto

En primer lugar vamos a crear un nuevo proyecto Android. Llamaremos a la aplicación “HolaUsuario”, indicaremos como mínimo SDK “Android 6.0 (Marshmallow)”, elijeremos una actividad en blanco con nombre “MainActivity”.

Paso 2. Diseñar la pantalla principal

En Android, el diseño y la lógica de una pantalla están separados en dos ficheros distintos.

- Por un lado, en el fichero `/res/layout/activity_main.xml` tendremos el diseño puramente visual de la pantalla definido como fichero XML
- Por otro lado, en el fichero `/java/paquetejava/MainActivity.java`, encontraremos el código java que determina la lógica de la pantalla.

Paso 3. Modificar activity_main.xml

En este XML se definen los elementos visuales que componen la interfaz de nuestra pantalla principal y se especifican todas sus propiedades.

Lo primero que nos encontramos es un elemento `LinearLayout`. Los layout son elementos no visibles que determinan cómo se van a distribuir en el espacio los controles que incluyamos en su interior. En este caso, un `LinearLayout`

distribuirá los controles uno tras otro y en la orientación que indique su propiedad `android:orientation`.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:text="@string/nombre"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/TxtNombre"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="text" />

    <Button
        android:id="@+id/BtnHola"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/saludar" />

</LinearLayout>
```

Gráfico 19. Fichero `content_main.xml`

Dentro del layout hemos incluido 3 controles: una etiqueta (`TextView`), un cuadro de texto (`EditText`), y un botón (`Button`). En todos ellos hemos establecido las siguientes propiedades:

- `android:id`. ID del control, con el que podremos identificarlo más tarde en nuestro código. Vemos que el identificador lo escribimos precedido de `"@+id/"`. Esto tendrá como efecto que al compilarse el proyecto se genere automáticamente una nueva constante en la clase `R` para dicho control.
- `android:text`. Texto del control. El texto de un control se puede especificar directamente o bien utilizar alguna de las cadenas de texto definidas en los recursos del proyecto (fichero `strings.xml`), en cuyo caso indicaremos su identificador precedido del prefijo `"@string/"`.
- `android:layout_height` y `android:layout_width`. Dimensiones del control con respecto al layout que lo contiene. Esta propiedad tomará normalmente los

valores "wrap_content" para indicar que las dimensiones del control se ajustarán al contenido del mismo, o bien "fill_parent" para indicar que el ancho o el alto del control se ajustará al ancho o alto del layout contenedor respectivamente.

Paso 4. Diseñar la segunda pantalla

De igual forma definiremos la interfaz de la segunda pantalla, creando una nueva actividad llamada SaludoActivity, y añadiendo esta vez tan solo una etiqueta (TextView) en el layout para mostrar el mensaje personalizado al usuario.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingView..."
    tools:showIn="@layout/activity_saludo"
    tools:context="com.grupoatrium.holausuario.SaludoActivity">

    <TextView
        android:id="@+id/TxtMensaje"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="$mensaje"/>

</RelativeLayout>
```

Gráfico 20. Fichero content_saludo.xml

Paso 5. Crear lógica para MainActivity

Una vez definida la interfaz de las pantallas de la aplicación deberemos implementar la lógica de la misma

Lo primero que encontramos en nuestro fichero java es la definición de una nueva clase MainActivity que extiende a Activity.

El único método que sobrescribiremos de esta clase será el método onCreate, llamado cuando se crea por primera vez la actividad. En este método lo único que encontramos en principio, además de la llamada a su implementación en la clase padre, es la llamada al método setContentView(R.layout.activity_main).

Con esta llamada estaremos indicando a Android que debe establecer como interfaz gráfica de esta actividad la definida en el recurso R.layout.activity_main, que no es más que la que hemos especificado en el fichero /res/layout/ activity_main.xml.

Una vez más vemos la utilidad de las diferentes constantes de recursos creadas automáticamente en la clase R al compilar el proyecto.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        // Localizar los controles
        final EditText txtNombre = (EditText) findViewById(R.id.TxtNombre);
        final Button btnHola = (Button) findViewById(R.id.BtnHola);

        // Asociar un listener al boton Hola
        btnHola.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {

                // Crear un intent
                Intent intent = new Intent(MainActivity.this, SaludoActivity.class);

                // Crear la informacion que vamos a pasar a la nueva Actividad
                Bundle b = new Bundle();
                b.putString("NOMBRE", txtNombre.getText().toString());

                // Añadimos la informacion al intent
                intent.putExtras(b);

                // Iniciamos la nueva actividad
                startActivity(intent);
            }
        });
    }
}
```

Gráfico 21. Fichero MainActivity.java

Paso 6. Crear lógica para SaludoActivity

vamos a crear una nueva actividad para la segunda pantalla de la aplicación análoga a ésta primera, para lo que crearemos una nueva clase SaludoActivity que extienda de Activity y que implemente el método onCreate indicando que utilice la interfaz definida en R.layout.activity_saludo.

```
public class SaludoActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_saludo);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        // Localizar el control
        TextView txtMensaje = (TextView) findViewById(R.id.TxtMensaje);

        // Recuperar la informacion recibida en el intent
        Bundle bundle = getIntent().getExtras();

        // Mostrar el mensaje en pantalla
        txtMensaje.setText("Hola " + bundle.getString("NOMBRE"));

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
        });
    }
}
```

Gráfico 22. Fichero SaludoActivity.java

Paso 7. Modificar AndroidManifest.xml

Cualquier aplicación Android utiliza un fichero especial en formato XML (AndroidManifest.xml) para definir, entre otras cosas, los diferentes elementos que la componen.

Todas las actividades de nuestra aplicación deben quedar convenientemente recogidas en este fichero.

La actividad principal ya debe aparecer puesto que se creó de forma automática al crear el nuevo proyecto Android, por lo que debemos añadir tan sólo la segunda.

```

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HolaUsuario"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="HolaUsuario"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".SaludoActivity"
            android:label="SaludoActivity"
            android:theme="@style/AppTheme.NoActionBar" >
        </activity>
    </application>
</manifest>

```

Gráfico 23. Fichero AndroidManifest.xml

Paso 8. Texto de los controles

Los textos se deben modificar en el fichero strings.xml

```

<resources>
    <string name="app_name">HolaUsuario</string>
    <string name="action_settings">Settings</string>
    <string name="nombre">Introduce nombre</string>
    <string name="saludar">Mostrar saludo</string>
    <string name="title_activity_saludo">SaludoActivity</string>
</resources>

```

Gráfico 24. Fichero strings.xml

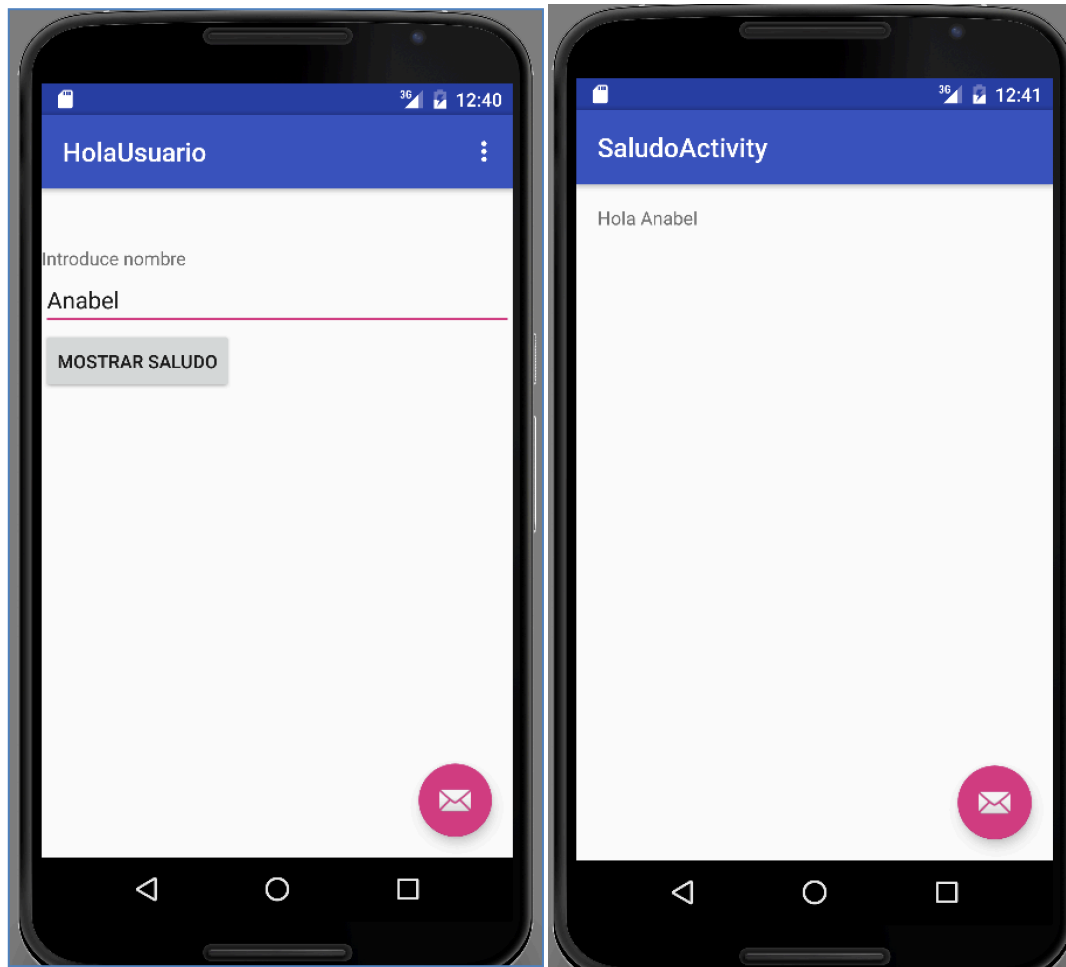


Gráfico 25. Resultado tras la ejecución de la aplicación

ÍNDICE DE GRÁFICOS

Gráfico 5. Android Studio vs ADT	3
Gráfico 6. Descarga jdk 8	3
Gráfico 7. Descarga Android Studio	4
Gráfico 8. Opción Configure	5
Gráfico 9. SDK Manager	5
Gráfico 10. SDK Manager	6
Gráfico 11. SDK Location	6
Gráfico 12. Comprobar actualizaciones	7
Gráfico 13. Canary Channel	7
Gráfico 14. Crear un proyecto	8
Gráfico 15. Uso de las diferentes versiones	9
Gráfico 16. Mínimo SDK	10
Gráfico 17. Blank Activity	10
Gráfico 18. Datos de la activity	10
Gráfico 26. Estructura proyecto en Android Studio	11
Gráfico 20. Carpeta src	12
Gráfico 21. Carpeta res	13
Gráfico 24. Fichero AndroidManifest.xml	13
Gráfico 27. Fichero content_main.xml	18
Gráfico 28. Fichero content_saludo.xml	19
Gráfico 29. Fichero MainActivity.java	20
Gráfico 30. Fichero SaludoActivity.java	21
Gráfico 31. Fichero AndroidManifest.xml	22
Gráfico 32. Fichero strings.xml	22
Gráfico 33. Resultado tras la ejecución de la aplicación	23