



ANDROID PARA INGENIEROS

2ª EDICIÓN

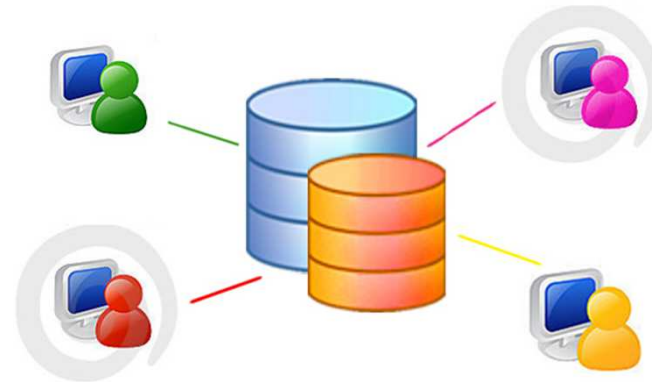
4ª SESIÓN

Curso 2015-2016

ÍNDICE

1. Persistencia de Datos. Ficheros
2. Bases de Datos
3. Conectar con la Web
4. Gráficos
5. Publicar Aplicaciones. Google Play
6. Otros

1- PERSISTENCIA DE DATOS



1- PERSISTENCIA DE DATOS

- La persistencia de datos hace referencia a la propiedad de los datos para que sobrevivan de alguna manera.
- Hasta ahora los datos solo persisten en memoria RAM, es decir, mientras la aplicación se está ejecutando.
- Se trata de almacenar los datos en la memoria interna del dispositivo para poder utilizarlos posteriormente o compartirlos con otras aplicaciones.

1- PERSISTENCIA DE DATOS

- Android dispone de dos modelos de persistencia de datos.
 - Los ficheros
 - Las Bases de Datos

1- PERSISTENCIA DE DATOS. FICHEROS

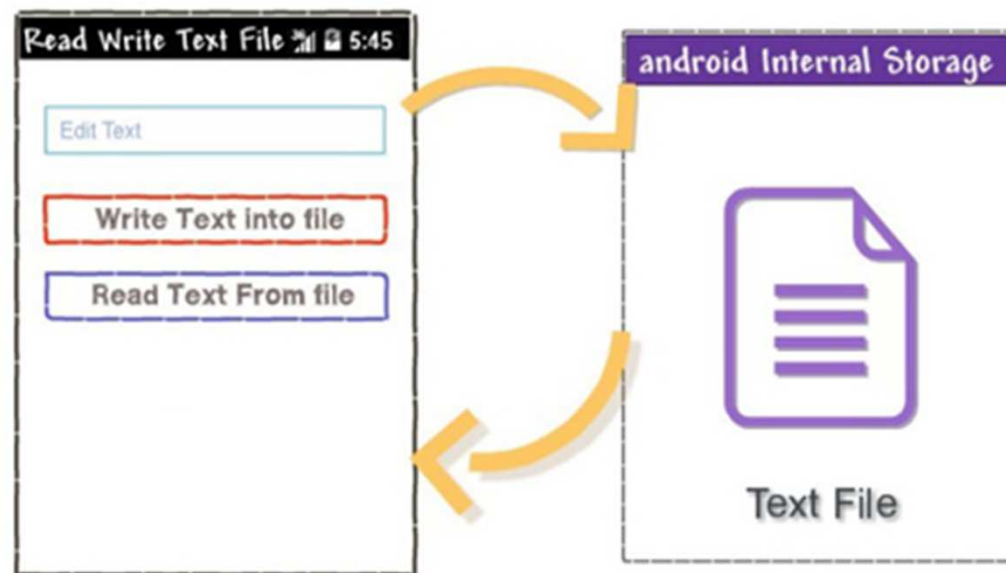
- Un fichero o archivo informático es un conjunto de bits que son almacenados en un dispositivo.
- Un fichero es identificado por su nombre y el directorio que lo contiene.
- Los ficheros son ubicados en directorios, con un nombre único dentro del directorio.
- Los ficheros suelen tener una extensión que ayuda a identificar su contenido (por ejemplo .txt)

1- PERSISTENCIA DE DATOS. FICHEROS

- Las operaciones típicas de un fichero son:
 - Creación de un fichero
 - Apertura de un fichero
 - Lectura de un fichero
 - Escritura de un fichero
 - Cierre de un fichero

1- PERSISTENCIA DE DATOS. FICHEROS

- Vamos a crear un esquema como el siguiente en Android para escribir y leer en ficheros.



1- PERSISTENCIA DE DATOS. FICHEROS

- Para manejar los ficheros tenemos que realizar los siguientes tres pasos:
 - Añadir permisos de usuario
 - Crear y escribir archivos
 - Leer los archivos

1- PERSISTENCIA DE DATOS. FICHEROS

- Para entender bien el procedimiento vamos a crear una aplicación como el esquema que hemos visto antes que nos permita escribir la entrada del usuario en un archivo y leerla.
- En primer lugar debemos añadir permisos de usuario para escribir en la memoria de nuestro dispositivo. Vamos a «AndroidManifest.xml» como hemos hecho otras veces y añadimos:

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

1- PERSISTENCIA DE DATOS. FICHEROS

- El segundo paso es crear y/o escribir en el fichero. En primer lugar vamos a definir el nombre que queramos que tenga y la ruta donde estará (path). Justo al empezar nuestro «MainActivity» vamos a declarar estas dos variables globales:

```
private String MI_ARCHIVO = "miArchivo.txt";  
private String PATH = "/sdcard/hola/";
```

- Después y dentro del escuchador del botón de escribir vamos a llamar a la función «escribirArchivo()».

1- PERSISTENCIA DE DATOS. FICHEROS

- La función «escribirArchivo()» hará lo siguiente:

```
// Método para escribir  
private void escribirArchivo() {  
    //reading text from file  
    try{
```

```
        File aux_file = new File(PATH),  
        aux_file.mkdirs();
```

```
        File file = new File(PATH, MI_ARCHIVO);
```

```
        // true --> activar el modo concatenar
```

```
        FileWriter writer = new FileWriter(file, false);  
        writer.write( edt_entrada.getText().toString() );  
        writer.flush();  
        writer.close();
```

```
        Toast.makeText(this, "Guardado correctamente", Toast.LENGTH_SHORT).show();  
    }
```

```
    catch(IOException e) {
```

```
        Toast.makeText(this, "Error al guardar " + e.toString(), Toast.LENGTH_SHORT).show();  
    }
```

```
}
```

Si no existe el directorio lo crea

El «Writer» nos permite escribir en nuestro fichero. False-> borra en contenido y escribe. True-> concatena al final del archivo.

1- PERSISTENCIA DE DATOS. FICHEROS

- El último paso es leer el fichero.
- Después y dentro del escuchador del botón de leer vamos a llamar a la función «leerArchivo()».

1- PERSISTENCIA DE DATOS. FICHEROS

○ La función «leerArchivo()» hará lo siguiente:

```
// Método para leer
private void leerArchivo() {
    //reading text from file
    try{
        File file = new File(PATH, MI_ARCHIVO);

        FileReader reader = new FileReader(file);
        BufferedReader br = new BufferedReader(reader);
        String s;
        while((s = br.readLine()) != null) {
            Toast.makeText(this, s, Toast.LENGTH_SHORT).show();
        }
        br.close();
        reader.close();
    }
```

```
        Toast.makeText(this, "Leído correctamente", Toast.LENGTH_SHORT).show();
    }
    catch(IOException e) {
        Toast.makeText(this, "Error al leer " + e.toString(),
        Toast.LENGTH_SHORT).show();
    }
}
```

El «Reader» nos permite leer nuestro fichero.

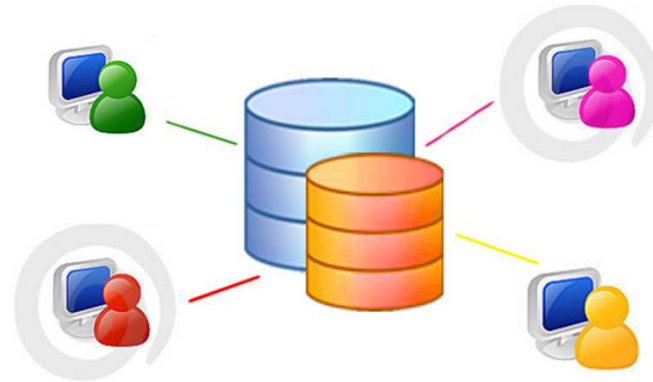
Además necesitamos un buffer donde se cargará lo que se va leyendo.

Vamos leyendo el archivo línea a línea hasta el final

EJERCICIO

- Crear una aplicación Android que nos permita escribir y leer en un fichero lo que escriba el usuario.
 - Usar un «TextEdit» para que usuario introduzca información.
 - Guardar el fichero en /sdcard/resultado/
 - Con nombre miArchivo.txt
 - Permitir con un «CheckBox» que la información que introduce el usuario se concatene al contenido o se sobrescriba.

2- BASES DE DATOS



2- BASES DE DATOS

- Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenadas sistemáticamente para su posterior uso.
- Existen dos tipos de bases de datos según la variabilidad.
 - Bases de datos estáticas, de solo lectura.
 - Bases de datos dinámicas, que pueden ser modificadas. Se puede leer, actualizar, borrar y editar datos.

2- BASES DE DATOS

- Otra forma de clasificar las Bases de Datos es dependiendo a su modelo de administración de sus datos.
 - **Bases de Datos jerárquicas**, los datos se organizan en forma de árbol. Un nodo padre con diferentes nodos hijos que a su vez pueden tener más hijos.
 - **Bases de Datos de red**, parecido al modelo jerárquico pero se permite tener varios nodos padre.
 - **Bases de Datos transaccionales**, su único fin es el envío y recepción de datos a gran velocidad. No se preocupan por la redundancia y duplicación de datos.

2- BASES DE DATOS

- **Bases de Datos relacionales**, su principal idea es el uso de “relaciones” entre las tablas. Es muy usado actualmente para modelar problemas reales y administrar datos dinámicamente.
- **Bases de Datos orientadas a objetos**, es un modelo bastante reciente que trata de almacenar los objetos completos (estado y comportamiento). Incorpora los conceptos importantes del paradigma de objetos: encapsulación, herencia y polimorfismo.

2- BASES DE DATOS

- Las bases de datos son un herramienta muy potente en las aplicaciones informáticas.
- Hasta hace muy poco resultaba muy costoso y complejo incorporar bases de datos a nuestras aplicaciones.
- No obstante, Android incorpora la librería «SQLite» que nos permite manejar bases de datos con el lenguaje SQL de una forma sencilla.

2- BASES DE DATOS

- SQL es el lenguaje de programación más usado para las bases de datos.
- No resultado difícil entender los ejemplos que vamos a ver si, queremos realizar cosas más complejas deberemos consultar algún manual de SQL.
- Para manipular una base de datos en Android usaremos la clase «SQLiteOpenHelper» que nos facilita la creación y manejo de bases de datos.

2- BASES DE DATOS

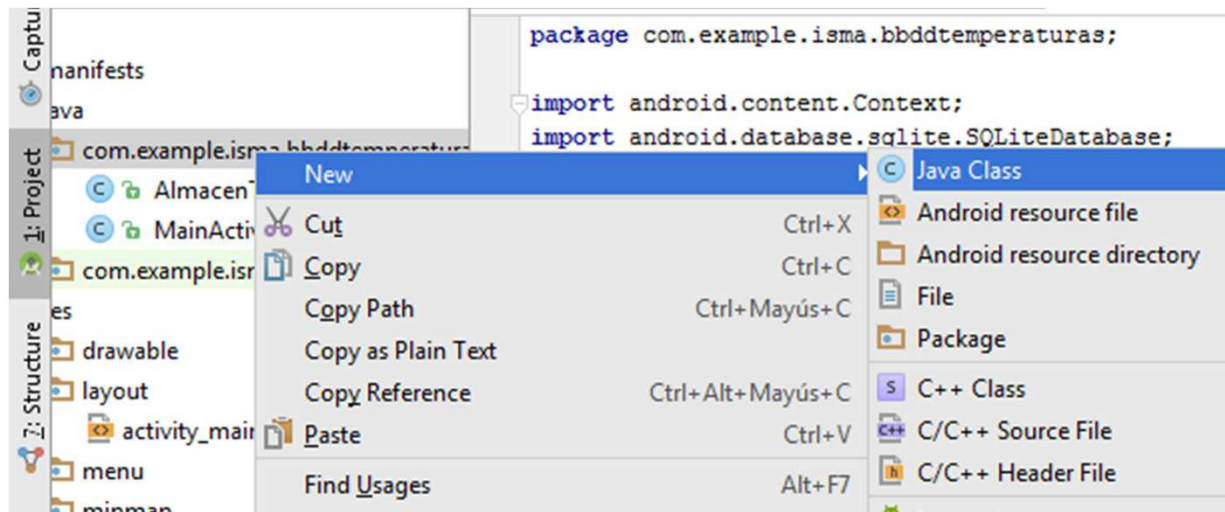
- Para usar y manejar una base de datos «SQLite» vamos a realizar los siguiente tres pasos:
 - Crear una clase que representará nuestra base de datos.
 - Implementar los métodos de creación de la base de datos.
 - Implementar los métodos con las operaciones que vamos a realizar sobre nuestra base de datos.

2- BASES DE DATOS

- Para entender mejor vamos a crear una base de datos para almacenar mediciones de diferentes sensores en diferentes ciudades. Vamos a almacenar el nombre de la ciudad, temperatura y la fecha. Crearemos un método para añadir nuevas temperaturas para diferentes ciudades y otro para consultar las x temperaturas mayores.
- En primer lugar vamos a crear una clase de java «AlmacenTemperaturasSQLite» que herede de «SQLiteOpenHelper».

2- BASES DE DATOS

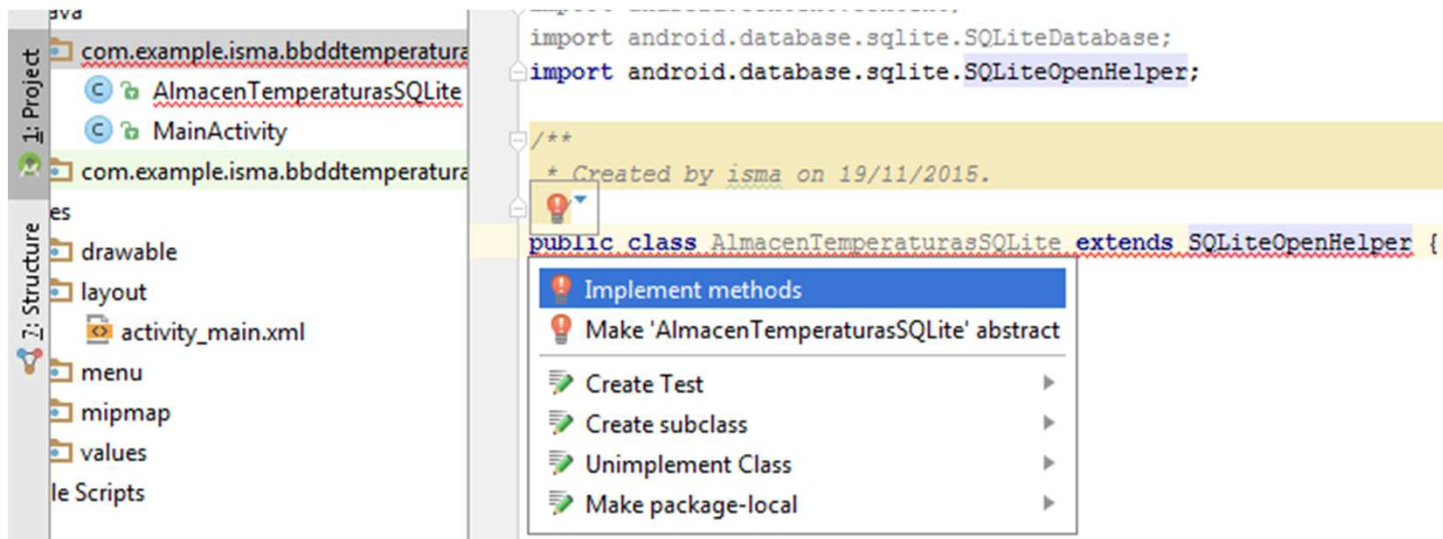
- Para crear una nueva clase de java en nuestro proyecto debemos pulsar el botón derecho aquí:



- Escribimos el nombre («AlmacenTemperaturasSQLite»)

2- BASES DE DATOS

- El segundo paso es heredar «extends» de la clase «SQLiteOpenHelper». Nos saldrá un error, pulsamos encima de «Implements Methods», así:



- Completamos los métodos que han surgido así:

2- BASES DE DATOS

```
public class AlmacenTemperaturasSQLite extends SQLiteOpenHelper {  
  
    private static final String nombreTabla = "temperaturas";  
  
    //Métodos de SQLiteOpenHelper  
    public AlmacenTemperaturasSQLite(Context context) {  
        super(context, nombreTabla, null, 1);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL("CREATE TABLE " + nombreTabla + " (" +  
            "_id INTEGER PRIMARY KEY AUTOINCREMENT, "+  
            "temperatura INTEGER, ciudad TEXT, fecha LONG)");  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int  
newVersion) {  
        // En caso de una nueva versión habría que actualizar las tablas  
    }  
}
```

2- BASES DE DATOS

- El última paso es implementar los métodos que necesitemos para utilizar nuestra base de datos.
 - Vamos a crear un método para almacenar las puntuaciones a partir de los puntos, nombre y fecha.
 - Y otro método para consultar las mejores X puntuaciones.
- De esta forma dentro de la clase que acabamos de crear:

2- BASES DE DATOS

```
//Métodos de AlmacenPuntuaciones
public void guardarTemperatura(double temperatura, String ciudad, long
fecha) {
    SQLiteDatabase db = getWritableDatabase();
    db.execSQL("INSERT INTO " + nombreTabla + " VALUES ( null, "+
        temperatura+", ""+ciudad+", "+fecha+"");
    db.close();
}

public Vector listaTemperaturas(int cantidad) {
    Vector result = new Vector();
    SQLiteDatabase db = getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT temperatura, ciudad FROM " +
        nombreTabla + " ORDER BY temperatura DESC LIMIT "
+cantidad, null);
    while (cursor.moveToNext()){
        result.add(cursor.getInt(0)+" " +cursor.getString(1));
    }
    cursor.close();
    db.close();
    return result;
}
}
```

2- BASES DE DATOS

- Ya hemos creado nuestra base de datos y algunos métodos para usarla, la última cuestión es como utilizarla.
- Dentro de nuestra clase «MainActivity» y dentro de «onCreate» podemos crear y usar la base de datos así:

2- BASES DE DATOS

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

    AlmacenTemperaturasSQLite miBD = new
    AlmacenTemperaturasSQLite(this);

    miBD.guardarTemperatura(25, "Madrid", 99999);
    miBD.guardarTemperatura(30, "Cadiz", 99999);

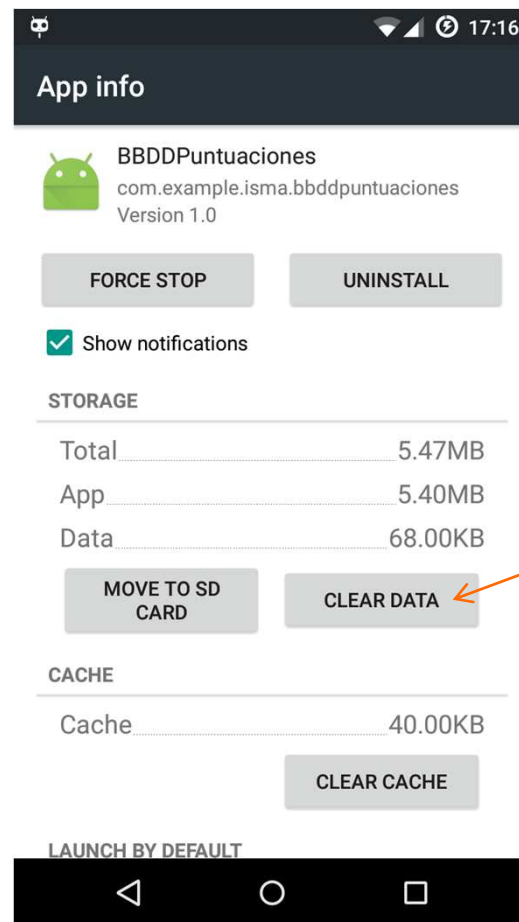
    Vector resultados = miBD.listaTemperaturas(100);

    for(int i=0; i< resultados.size(); i++){
        Toast.makeText(MainActivity.this, ""+resultados.get(i),
        Toast.LENGTH_SHORT).show();
    }
}
```

2- BASES DE DATOS

- Los datos de estas bases de datos solo pueden ser accesibles desde la misma aplicación.
- La información de la base de datos se guarda de una ejecución a otra.
- Podemos borrar el contenido de la base de datos si vamos a «AppInfo» de nuestra aplicación y pulsamos en «Clear Data».

2- BASES DE DATOS



2- BASES DE DATOS

- Si queremos crear y consultar datos de una Base de Datos externa podemos encontrar un ejemplo de como hacerlo aquí:

<http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>

EJERCICIO PROPUESTO

- Usando la base de datos que hemos creado antes, crear un aplicación para nuestro dispositivo Android que vaya guardando las temperaturas de diferentes ciudades.
 - Usar dos «TextEdit» para que usuario introduzca la ciudad y la temperatura.
 - Utilizar un botón para guardar en la base de datos esta nueva temperatura usando la hora actual del sistema en milisegundos (puedes usar «*new Date().getTime()*»).
 - Utilizar un botón mostrar para mostrar las 10 temperaturas más altas (usar el componente «Toast»).
 - (Un poco más avanzado) Mostrar las temperaturas con su ciudad en una lista («Spinner»).

3- CONECTAR CON LA WEB



3- CONECTAR CON LA WEB

- Internet es un conjunto descentralizado de redes de comunicación interconectadas.
- Utilizan el protocolo TCP/IP que garantiza que las red heterogéneas se puedan comunicar y ofrece una unidad lógica a nivel mundial.
- Sus orígenes se remontan a 1969 cuando se estableció la primera conexión entre ordenadores.

3- CONECTAR CON LA WEB

- Existen diferentes servicios y protocolos en Internet como son:
 - Envío de correos electrónicos (SMTP)
 - Transmisión de archivos (FTP y P2P)
 - Conversaciones en línea (IRC)
 - Mensajería instantánea
 - Telefonía (VoIP)
 - ...
- El servicio más extendido de Internet es la Web (World Wide Web WWW).

3- CONECTAR CON LA WEB

- La Web es un sistema distribuido de documentos de hipertexto accesibles vía Internet.
- Se utiliza un navegador Web para visualizar los sitios Web compuestos por texto, imágenes, multimedia, etc. y navegar a través de estas páginas usando hiperenlaces.
- Utiliza el protocolo HTTP

3- CONECTAR CON LA WEB

- El protocolo HTTP (Hypertext Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW)
- HTTP define métodos que indican que acción se desea efectuar sobre un recurso (texto, imagen, multimedia, etc.)
 - GET- pide un recurso
 - POST- envía datos para que sean procesados
 - PUT- actualiza el estado de un recurso
 - DELETE- borra el recurso especificado
 - TRACE- solicita al servidor información de uso del recurso especificado.
 - CONNECT- permite comprobar si existe conexión hasta un recurso.

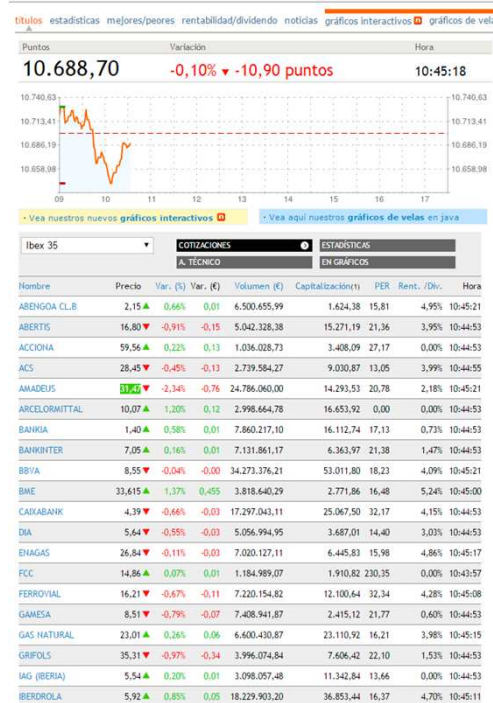
3- CONECTAR CON LA WEB

- El protocolo HTTP tiene los siguientes códigos de respuesta (responseCode) al trabajar con un recurso:
 - 1xx: Conexión rechazada
 - 2xx: Operación exitosa
 - **200 → OK**
 - 3xx: Redirección
 - 4xx: Error por parte del cliente
 - 404 → Hay comunicación, pero no se encuentra el recurso solicitado (es bastante típico)
 - 5xx: Error por parte del servidor

Para obtener más información sobre los códigos de respuesta HTTP consultar:

http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

- Otra forma de utilizar los datos de la Web es a través de una aplicación para obtener datos útiles y poder realizar cálculos. Llamado **Web Scraping**
Ej.: *eleconomista/indice/IBEX-35*

[illegible]

Botón derecho → Ver código fuente de la página

3- CONECTAR CON LA WEB

- El problema de trabajar directamente con el código fuente es que hay ocasiones que es demasiado complejo o largo.
 - Buscar la información que necesitamos
 - Entender la estructura del dato que estamos buscando
 - Posteriormente iremos dividiendo (split) el contenido hasta llegar a la información que estamos buscando.

3- CONECTAR CON LA WEB

- Otra forma más sencilla de acceder a los datos en la Web es utilizando APIs.
- Una API es una interfaz de programación de aplicaciones (Application Programming Interface)
- Es un conjunto de rutinas que provee de acceso a funciones de un determinado software.
- Desventaja, no existen APIs para todas las Web.

3- CONECTAR CON LA WEB

- Vamos a usar la API de Yahoo para finanzas.
- Ver como funciona esta API, escribir en el buscador:
 - Yahoo API finance
- Veremos las API disponibles (seleccionar CSV API → Quotes)
- Entender como se usa la API de Yahoo para consultar las finanzas.
- Ejemplo:
 - <http://download.finance.yahoo.com/d/quotes.csv?s=%40%5EDJI,GOOG&f=ns!1op&e=.csv>

3- CONECTAR CON LA WEB

- Vamos a ver como podemos consultar (get) datos desde la web (http) en Android. Necesitamos realizar los siguientes 5 pasos:
 - Añadir permisos.
 - Crear un hilo para consultar.
 - Mostrar la consulta.
 - Añadir dependencias necesarias.
 - Corregir el «Grandle» (por defecto no funciona).
- El «Grandle» es una herramienta de Android Studio que construye los proyecto. Es como un “compilador”.

3- CONECTAR CON LA WEB

- En primer lugar necesitamos añadir permisos para realizar las consultas. Vamos a nuestro archivo «AndroidManifest.xml» y añadimos como hemos hecho otras veces:

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

3- CONECTAR CON LA WEB

- El segundo paso es crear un hilo para realizar la consulta web. El proceso de consultar es bloqueante y por lo tanto no puede ir en el hilo principal (que controla nuestra interfaz).
- Una forma fácil de crear un hilo y lanzarlo es esta:

3- CONECTAR CON LA WEB

```
Thread thread = new Thread(new Runnable(){
    @Override public void run() {
        try {
            //Nuestro código para ejecutar en
            // el hilo
            // ...
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

thread.start();
```


3- CONECTAR CON LA WEB

- Vamos a usar el escuchador de un botón para lanzar nuestro hilo con la consulta a la web. Vamos a hacer una consulta a la API de Yahoo.
- Creamos una variable global que lo utilizaremos como buffer de lectura:
private BufferedReader **in**;
- Inicializamos a null el objeto «in», dentro del método «onCreate()».
- Dentro del escuchador escribimos la llamada a la vez a su vez dentro de un hilo, de esta forma:

3- CONECTAR CON LA WEB

```
Thread thread = new Thread(new Runnable(){
    @Override
    public void run() {
        try {
            HttpClient httpclient = new DefaultHttpClient();

            HttpGet request = new HttpGet();
            URI website = new
            URI("http://download.finance.yahoo.com/d/quotes.csv?s=%40%5EDJI,GOO
            G&f=ns!1op&e=.csv");
            request.setURI(website);
            HttpResponse response = httpclient.execute(request);
            in = new BufferedReader(new InputStreamReader(
                response.getEntity().getContent()));

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
thread.start();
txt_salida.setText("Consultado...");
```

3- CONECTAR CON LA WEB

- El tercer paso es mostrar la consulta. Para ello, dentro del escuchador de otro botón vamos a leer el contenido de «in» que es de tipo «BufferedReader» y mostrarlo en un «TextView» en contenido de esta forma:

```
try {  
    String linea;  
    txt_salida.setText("Web = ");  
    while ((linea = in.readLine()) != null) {  
        txt_salida.append(linea);  
    }  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

3- CONECTAR CON LA WEB

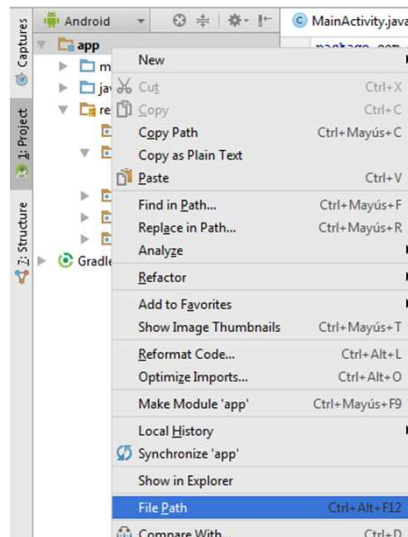
- El siguiente paso es añadir algunas dependencias que no están por defecto dentro de SDK de Android para hacer las consultas en la web.
- Realizamos las siguientes 5 acciones:

3- CONECTAR CON LA WEB

1. Vamos a **<http://hc.apache.org/downloads.cgi>** (página oficial) o directamente desde **<https://www.dropbox.com/s/ez898m3tvrppop6/httpcomponents-client-4.5.1-bin.zip>**
2. Descargamos **httpclient 4.5.1**, el primer archivo .zip
3. Descomprimos todos los archivos
4. Arrastramos dentro de nuestro proyecto **/app/build/** (ver siguiente transparencia) los archivos **httpclient-4.5.1.jar**, **httpcore-4.4.3.jar** y **httpmime-4.5.1.jar**
5. En App dentro de nuestro proyecto, botón derecho, pinchamos en “**Open Module Settings**”, **app**, **dependencies**, +, **File dependency** y añadimos los tres archivos anteriores.

3- CONECTAR CON LA WEB

- Podemos abrir nuestro directorio de trabajo pulsando botón derecho sobre «app» en nuestro proyecto y pinchando en «Show in Explorer» así:



- Aquí podemos encontrar /app/build/ para copiar nuestras dependencias.

3- CONECTAR CON LA WEB

- El último paso es corregir un error que aparece dentro del Gradle a añadir estas librerías.
- Dentro de «Gradle Scripts» abrimos «build.gradle (Module:app)» justo después de la primera línea copiamos esto:

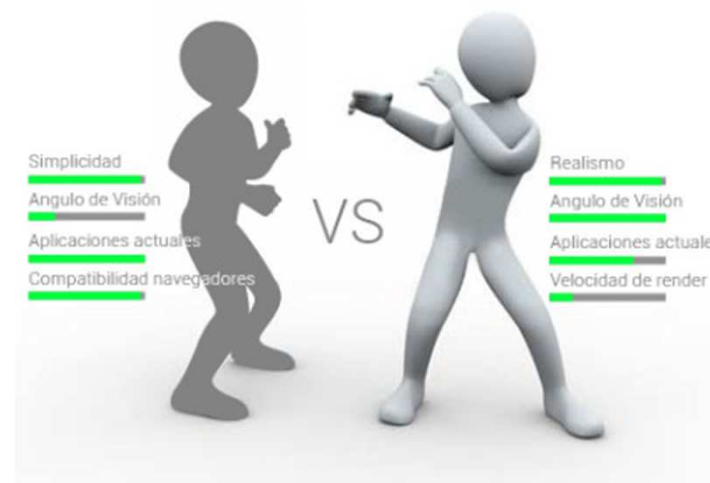
3- CONECTAR CON LA WEB

```
android {  
    packagingOptions {  
        exclude 'META-INF/DEPENDENCIES.txt'  
        exclude 'META-INF/DEPENDENCIES'  
        exclude 'META-INF/dependencies.txt'  
        exclude 'META-INF/LICENSE.txt'  
        exclude 'META-INF/LICENSE'  
        exclude 'META-INF/license.txt'  
        exclude 'META-INF/LGPL2.1'  
        exclude 'META-INF/NOTICE.txt'  
        exclude 'META-INF/NOTICE'  
        exclude 'META-INF/notice.txt'  
    }  
}
```


EJERCICIO PROPUESTO

- Crear una aplicación para tu dispositivo Android que permita consultar valores económicos de diferentes empresas.
 - Utilizar Yahoo Api Finance
 - Usar un TextEdit para escribir la empresa (por ejemplo AA, GOOG, APPL)
 - O usar una lista para elegir entre las empresas.
 - Usar un botón para consultar y otro para mostrar los valores.
- (Más complicado) Permitir consultar la cotización actual de las empresas del IBEX-35 directamente desde la Web.
 - Usar por ejemplo <http://www.eleconomista.es/indice/IBEX-35>
 - Puedes usar la función «split» de los Strings para dividirlos en partes. Ej: `String[] partes = linea.split(",");`

4- GRÁFICOS



4- GRÁFICOS

- En Android también podemos trabajar con gráficos 2D y 3D.
- Cuando estamos con gráficos 3D en Android tenemos una serie de herramientas para dibujar en pantalla desde un simple círculo hasta complejos escenarios 3D a través de OpenGL.
- Vamos a centrarnos en los gráficos 2D porque son más sencillos y se siguen utilizando bastante.

4- GRÁFICOS

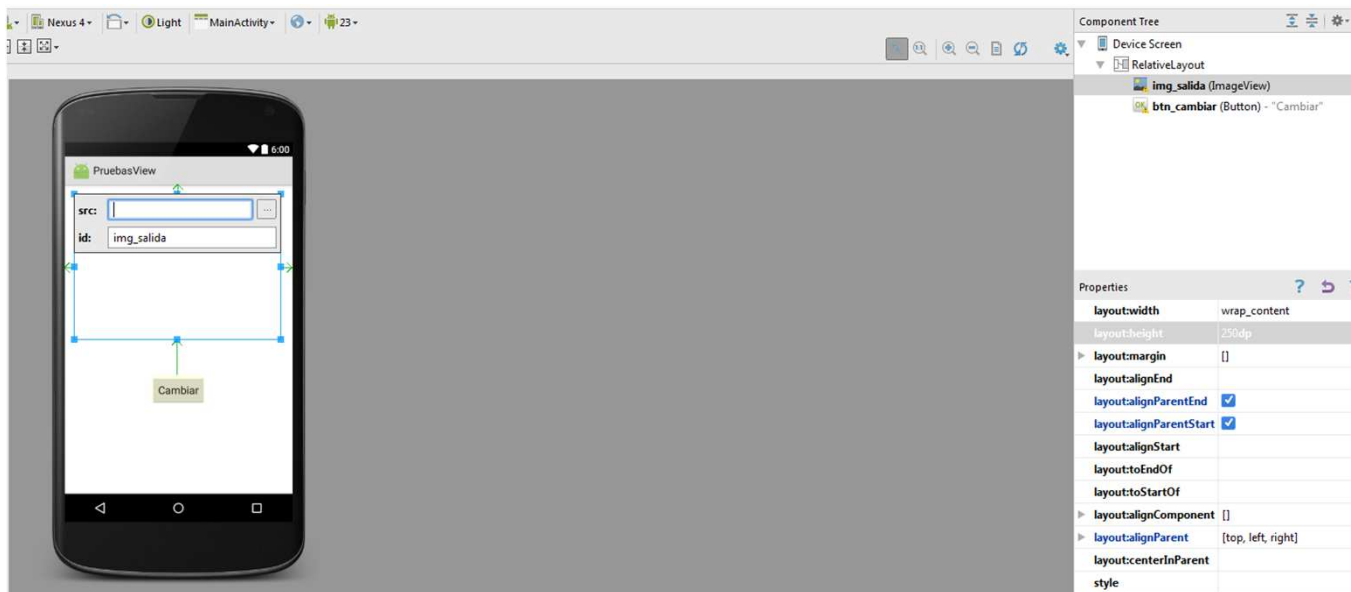
- Vamos a usar los «**Canvas**» para trabajar con gráficos.
- Un «**Canvas**» es simplemente un objeto que nos permite dibujar sobre el, o bien, modificar las vista de los «**View**» existentes.
- Podemos dibujar gráficos y animaciones dentro de un «**View**», de esta forma el sistema se encarga de dibujarlos y animarlos por nosotros.

4- GRÁFICOS

- El componente más básico que tenemos en Android para trabajar con gráficos en el «ImageView».
- Este componente nos permite mostrar imágenes, cambiar su color, una imagen, etc.
- Podemos colocar imágenes estáticas o cambiarlas dinámicamente en el código.

4- GRÁFICOS

- Vamos a arrastrar un «ImageView» dentro de nuestra aplicación y le vamos asignar un tamaño fijo.



4- GRÁFICOS

- Al igual que los componentes que hemos visto antes, el primer paso es declarar un variable y conectar con el componente de la interfaz.
- Podemos cambiar su color, su imagen, respectivamente, en el código así:

```
private ImageView img_salida;
```

```
...
```

```
img_salida = (ImageView)findViewById(R.id.img_salida);
```

```
...
```

```
img_salida.setBackgroundColor(Color.argb(255, 255, 0, 0));
```

```
img_salida.setImageDrawable(getResources().getDrawable(R.drawable.tigre));
```

Tigre es una imagen llamada «tigre.xxx» que debe estar en nuestra carpeta «Drawable».



EJERCICIO

- Crear una aplicación para nuestro dispositivo Android que tenga un «ImageView» y un botón. Cuando se pulse al botón se cambie la imagen del «ImageView».
- Descargar dos imágenes (guardar en «Drawable») y utilizarlas.

4- GRÁFICOS

- El siguiente componente importante que tenemos en Android para trabajar con gráficos en el «**Canvas**» que ya hemos explicado antes.
- Vamos a ver un ejemplo de cómo crear un «Canvas» para dibujar sobre con el dedo.
- Necesitamos crear una clase de Java para manejar el «Canvas» e implementar sus métodos. Vamos a llamarlo «CanvasVista.java».
- Vamos a ver que esta clase debe heredar de «View».

4- GRÁFICOS

```
public class CanvasVista extends View {  
    public int width;  
    public int height;  
    private Bitmap mBitmap;  
    private Canvas mCanvas;  
    private Path mPath;  
    Context context;  
    private Paint mPaint;  
    private float mX, mY;  
    private static final float TOLERANCE = 5;  
  
    public CanvasVista(Context c, AttributeSet attrs) {  
        super(c, attrs);  
        context = c;  
  
        // we set a new Path  
        mPath = new Path();  
  
        // and we set a new Paint with the desired attributes  
        mPaint = new Paint();  
        mPaint.setAntiAlias(true);  
        mPaint.setColor(Color.BLACK);  
        mPaint.setStyle(Paint.Style.STROKE);  
        mPaint.setStrokeJoin(Paint.Join.ROUND);  
        mPaint.setStrokeWidth(4f);  
    }  
}
```

4- GRÁFICOS

```
// override onSizeChanged
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    super.onSizeChanged(w, h, oldw, oldh);

    // your Canvas will draw onto the defined Bitmap
    mBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
    mCanvas = new Canvas(mBitmap);
}

// override onDraw
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    // draw the mPath with the mPaint on the canvas when onDraw
    canvas.drawPath(mPath, mPaint);
}

// when ACTION_DOWN start touch according to the x,y values
private void startTouch(float x, float y) {
    mPath.moveTo(x, y);
    mX = x;
    mY = y;
}

// when ACTION_MOVE move touch according to the x,y values
private void moveTouch(float x, float y) {
    float dx = Math.abs(x - mX);
    float dy = Math.abs(y - mY);
    if (dx >= TOLERANCE || dy >= TOLERANCE) {
        mPath.quadTo(mX, mY, (x + mX) / 2, (y + mY) / 2);
        mX = x;
        mY = y;
    }
}
```

4- GRÁFICOS

```
public void clearCanvas() {
    mPath.reset();
    invalidate();
}

// when ACTION_UP stop touch
private void upTouch() {
    mPath.lineTo(mX, mY);
}

//override the onTouchEvent
@Override
public boolean onTouchEvent(MotionEvent event) {
    float x = event.getX();
    float y = event.getY();

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            startTouch(x, y);
            invalidate();
            break;
        case MotionEvent.ACTION_MOVE:
            moveTouch(x, y);
            invalidate();
            break;
        case MotionEvent.ACTION_UP:
            upTouch();
            invalidate();
            break;
    }
    return true;
}
```

El método
«**onTouchEvent**» nos
sirve para capturar
eventos o pulsaciones que
pueda realizar el usuario.

4- GRÁFICOS

- Acabamos de crear nuestro propio «Canvas» que nos permite dibujar con el dedo sobre él.
- El siguiente paso es añadirlo a nuestra interfaz («activity_main.xml»). Además vamos a añadir un botón para borrarlo.

4- GRÁFICOS

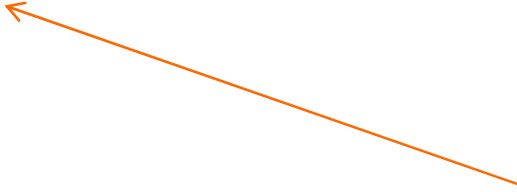
...

```
<com.example.isma.canvasprueba.CanvasVista  
    android:id="@+id/canvas_1"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:textColor="#FFFFFF" />
```

```
<Button  
    android:id="@+id/btn_limpiar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="bottom|center"  
    android:text="Limpiar Canvas" />
```

...

Ruta y nombre
de nuestro
«Canvas»



4- GRÁFICOS

- Finalmente solo nos queda implementar el escuchador el nuestro botón y llamar al método de borrar cuando se pulse. De esta forma:

```
private CanvasVista canvas;  
private Button btn_limpiar;  
  
...  
canvas = (CanvasVista)findViewById(R.id.canvas_1);  
btn_limpiar = (Button)findViewById(R.id.btn_limpiar);  
  
...  
// Botón limpiar  
btn_limpiar.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        canvas.clearCanvas();  
    }  
});
```

EJERCICIO

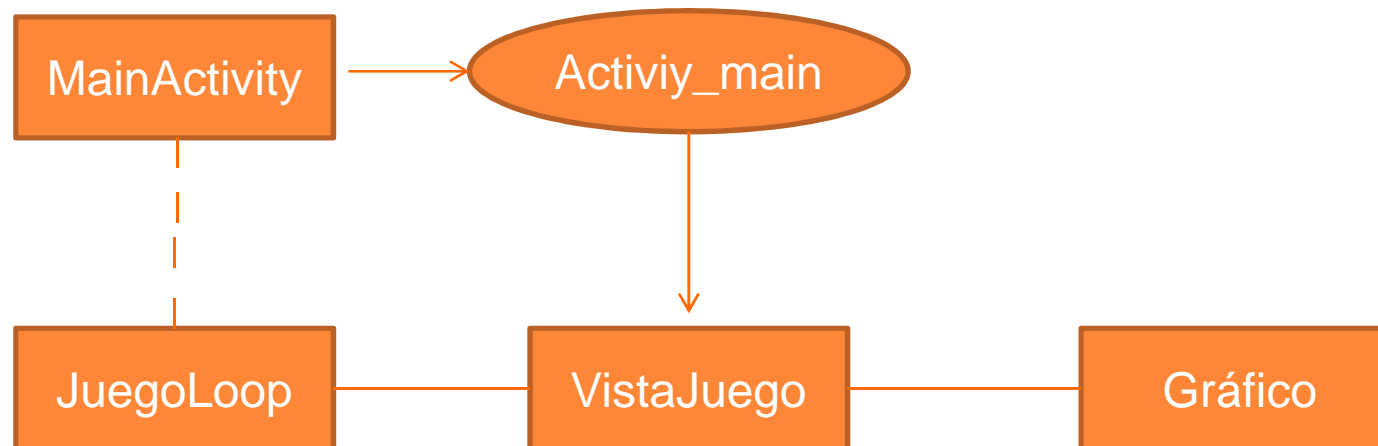
- Crear una aplicación Android como la que hemos visto antes que nos permita dibujar con el dedo. Añadir el botón de borrar.
- Crear un nuevo botón que nos permita cambiar la línea de dibujo de rojo a azul y viceversa.

4- GRÁFICOS

- La última parte que vamos a ver en gráficos son algunas nociones de cómo crear un juego usando un esquema sencillo.
- Vamos a crear un juego con pocas imágenes pero el usuario podrá interaccionar con ellas.
- La principales diferencias de un juego respecto a lo que hemos visto antes son:
 - Hay un bucle de juego donde se irá actualizando nuestra vista «View».
 - El usuario irá interaccionando con nuestra vista.

4- GRÁFICOS

- Vamos a ver el siguiente esquema, que vamos a utilizar para este tipo de juegos:



4- GRÁFICOS

- Vamos a ir viendo por partes el esquema de juego aplicado a un mini juego. Crearemos un juego donde una nave tiene que esquivar diferentes meteoritos.
- En este juego tenemos dos tipos de objetos: nave y asteroides.
- Vamos a crear un clase de java «Grafico» que nos servirá para crear tanto la nave como los asteroides, con diferentes funciones para moverlos, girarlos, etc..

4- GRÁFICOS

```
class Grafico {  
    private Drawable drawable; //Imagen que dibujaremos  
    private double posX, posY; //Posición  
    private double incX, incY; //Velocidad desplazamiento  
    private int angulo, rotacion; //Ángulo y velocidad rotación  
    private int ancho, alto; //Dimensiones de la imagen  
    private int radioColision; //Para determinar colisión  
    //Donde dibujamos el gráfico (usada en view.invalidate)  
    private View view;  
    public static final int MAX_VELOCIDAD = 25;  
  
    public Grafico(View view, Drawable drawable) {  
        this.view = view;  
        this.drawable = drawable;  
        ancho = drawable.getIntrinsicWidth();  
        alto = drawable.getIntrinsicHeight();  
        radioColision = (alto + ancho) / 4;  
    }  
  
    public double distancia(Grafico g) {  
        return Math.hypot(posX-g.posX, posY-g.posY);  
    }  
  
    public boolean verificaColision(Grafico g) {  
        return(distancia(g) < (radioColision+g.radioColision));  
    }  
}
```

4- GRÁFICOS

```
public void dibujaGrafico(Canvas canvas){
    canvas.save();
    int x=(int) (posX+ancho/2);
    int y=(int) (posY+alto/2);
    canvas.rotate((float) angulo,(float) x,(float) y);
    drawable.setBounds((int)posX, (int)posY,
        (int)posX+ancho, (int)posY+alto);

    drawable.draw(canvas);
    canvas.restore();
    int rlnval = (int) Math.hypot(ancho,alto)/2 + MAX_VELOCIDAD;
    view.invalidate(x-rlnval, y-rlnval, x+rlnval, y+rlnval);
}

public void incrementaPos(double factor){
    posX+=incX * factor;

    // Si salimos de la pantalla, corregimos posición
    if(posX<-ancho/2) {posX=view.getWidth()-ancho/2;}
    if(posX>view.getWidth()-ancho/2) {posX=-ancho/2;}
    posY+=incY * factor;
    if(posY<-alto/2) {posY=view.getHeight()-alto/2;}
    if(posY>view.getHeight()-alto/2) {posY=-alto/2;}
    angulo += rotacion * factor; //Actualizamos ángulo
}
```

4- GRÁFICOS

```
public void setIncX(double inc) { this.incX=inc; }  
public void setIncY(double inc) { this.incY=inc; }  
public void setAngulo(int ang){ this.angulo = ang; }  
public void setRotacion(int rot){ this.rotacion = rot; }  
public int getAncho(){ return this.ancho; }  
public int getAlto(){ return this.alto; }  
public void setPosX(double pos){ this.posX = pos; }  
public void setPosY(double pos){ this.posY = pos; }  
public double getPosX(){ return posX; }  
public double getPosY(){ return posY; }  
  
}
```

4- GRÁFICOS

- El siguiente paso es crear la vista del juego «**VistaJuego**» como una clase de Java que hereda de «View» donde tendremos nuestro objetos del juego (los diferentes gráficos).
- La «**VistaJuego**» que hereda de «View» se encarga de crear nuestra nave (de tipo «**Grafico**»), los asteroides (también de tipo «**Grafico**»), lanzar el bucle del juego («**JuegoLoop**»), proporciona métodos para acceder a los elementos de juego y sobrescribe los métodos para reaccionar a las pulsaciones del usuario.

4- GRÁFICOS

```
public class VistaJuego extends View {

    // /// ASTEROIDES ///
    public Vector<Grafico> Asteroides; // Vector con los Asteroides
    private int numAsteroides= 5; // Número inicial de asteroides
    private int numFragmentos= 3; // Fragmentos en que se divide
    // /// NAVE ///
    private Grafico nave; // Gráfico de la nave
    private int giroNave; // Incremento de dirección
    private float aceleracionNave; // aumento de velocidad
    // Incremento estándar de giro y aceleración
    private final int PASO_GIRO_NAVE = 5;
    private final float PASO_ACELERACION_NAVE = 0.5f;
    private JuegoLoop juegoLoop;
    private int alto, ancho;
    private Drawable drawableNave, drawableAsteroide, drawableMisil;
    private Canvas canvas;

    public VistaJuego(Context context, AttributeSet attrs) {
        super(context, attrs);

        drawableAsteroide = context.getResources().getDrawable(R.drawable.asteroide);
        drawableNave = context.getResources().getDrawable(R.drawable.nave);
        Asteroides = new Vector<Grafico>();
        nave = new Grafico(this, drawableNave);

        // Creamos los asteroides en posiciones y ángulos aleatorios
        for (int i = 0; i < numAsteroides; i++) {
            Grafico asteroide = new Grafico(this, drawableAsteroide);
            asteroide.setIncY(Math.random() * 4 - 2);
            asteroide.setIncX(Math.random() * 4 - 2);
            asteroide.setAngulo((int) (Math.random() * 360));
            asteroide.setRotacion((int) (Math.random() * 8 - 4));
            Asteroides.add(asteroide);
        }

        // Creamos el Loop para el juego
        juegoLoop = new JuegoLoop(this);
        juegoLoop.terminar = false;
        juegoLoop.start();
    }
}
```


4- GRÁFICOS

```
public int getNumAsteroides(){
    return this.numAsteroides;
}
public int getAlto(){
    return this.alto;
}
public int getAncho(){
    return this.ancho;
}
public Grafico getNave(){
    return this.nave;
}
public Vector<Grafico> getAsteroides(){
    return this.Asteroides;
}

@Override
protected void onSizeChanged(int ancho, int alto, int ancho_anter, int alto_anter) {
    super.onSizeChanged(ancho, alto, ancho_anter, alto_anter);
    this.ancho = ancho;
    this.alto = alto;
    // Una vez que conocemos nuestro ancho y alto.
    nave.setPosX(ancho/2-(nave.getAncho())/2);
    nave.setPosY(alto/2-(nave.getAlto())/2);

    for (Grafico asteroide: Asteroides) {
        do{
            asteroide.setPosX(Math.random()*(ancho-asteroide.getAncho()));
            asteroide.setPosY(Math.random()*(alto-asteroide.getAlto()));
        } while(asteroide.distancia(nave) < (ancho+alto)/5);
    }
}
```

4- GRÁFICOS

@Override

```
protected void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    this.canvas = canvas;  
  
    for (Grafico asteroide: Asteroides)  
        asteroide.dibujaGrafico(canvas);  
  
    nave.dibujaGrafico(canvas);  
}
```

@Override

```
public boolean onTouchEvent(MotionEvent event){  
    // Log.e("Info","X="+event.getX()+" Y="+event.getY());  
    if (event.getX() > nave.getPosX()) {  
        nave.setPosX(nave.getPosX() + 5);  
  
    }else {  
        nave.setPosX(nave.getPosX() - 5);  
  
    }  
  
    if (event.getY() > nave.getPosY()) {  
        nave.setPosY(nave.getPosY() + 5);  
  
    }else {  
        nave.setPosY(nave.getPosY() - 5);  
  
    }  
    return true;  
}
```

4- GRÁFICOS

- El siguiente paso es crear una clase de Java para el bucle del juego «JuegoLoop» que hereda de la clase hilos («Thread»).
- Aquí tendremos un bucle del juego donde moveremos los asteroides. Vamos a sobrescribir el método «run» y usando un bucle vamos mover los asteroides haciendo una pequeña espera para no bloquear nuestro dispositivo.
- Esta parte se ejecutará en un hilo paralelamente.

4- GRÁFICOS

```
public class JuegoLoop extends Thread {
    private VistaJuego vistaJuego;
    static boolean terminar = false;

    public JuegoLoop(VistaJuego vistaJuego){
        this.vistaJuego = vistaJuego;
    }

    @Override
    public void run(){
        while (!terminar){
            // Log.e("Info", "Ejecutando");

            for (int i=0;i<vistaJuego.getNumAteroides();i++) {
                vistaJuego.Asteroides.get(i).incrementaPos(1);

                if (vistaJuego.Asteroides.get(i).verificaColision(vistaJuego.getNave())){
                    Log.e("Info", "Colisión");
                }
            }

            try {
                sleep(1000 / Grafico.MAX_VELOCIDAD); // 1000 / 25
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

4- GRÁFICOS

- A continuación dentro de nuestro «activity_main.xml» vamos a añadir la vista del juego que hemos creado, de esta forma:

...

```
<com.example.isma.asteroides2.VistaJuego  
    android:id="@+id/VistaJuego"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:focusable="true"  
    android:background="@drawable/espacio"  
/>
```

...

Debemos poner la ruta y el nombre de nuestra vista «VistaJuego»

4- GRÁFICOS

- Por último dentro de nuestro «MainActivity» a parte de cargar nuestro «activity_main» como hacemos siempre, vamos a sobrescribir el método «onKeyDown» para detectar cuando el usuario pulsa volver (tecla de volver de nuestro dispositivo) y poder parar correctamente el juego.
- Nuestro «MainActivity» quedaría así:

4- GRÁFICOS

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    @Override  
    public boolean onKeyDown(int keyCode, KeyEvent event){  
        Log.e("Info", "Se ha pulsado la tecla " + keyCode);  
  
        if (keyCode == KeyEvent.KEYCODE_BACK) {  
            JuegoLoop.terminar = true;  
        }  
        // return super.onKeyDown(keyCode, event);  
        return super.onKeyDown(keyCode,event);  
    }  
}
```

EJERCICIO

- Crear el juego de las naves y asteroides para nuestro dispositivo Android.
- (Un poco más avanzado). Controlar cuando nos choquemos con un asteroide se cambie la imagen de nuestra nave por una explosión para saber que se ha destruido la nave.

5- PUBLICAR APLICACIONES. GOOGLE PLAY



5- PUBLICAR APLICACIONES. GOOGLE PLAY

- **Google Play** Store (anteriormente Android Market) es una plataforma de distribución digital de aplicaciones móviles para todos los dispositivos con sistema operativo Android. Es una tienda en línea desarrollada y operada por Google.
- Esta plataforma permite a los usuarios navegar y descargar aplicaciones (apk), música, libros, revistas y películas.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- También se puede adquirir dispositivos de Google (como los Nexus).
- Las aplicaciones se encuentran disponibles de forma gratuita, así como también con costo.
- En julio de 2013 se anunció que Google Play había sobrepasado 1 millón de aplicaciones publicadas y se habían registrado más de 50 mil millones de descargas.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- La gran novedad que incorpora Google Play hace referencia a los desarrolladores que podrán crear sus propias aplicaciones y publicirlas.
- Ofrece una retroalimentación y un sistema de calificación de aplicaciones similar a las calificaciones de vídeos de YouTube.
- Tres acciones para subir aplicaciones al mercado:
 - Necesitamos una cuenta de desarrollador de Google
 - Subir y describir el contenido
 - Publicar el contenido

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- La cuenta de desarrollador de Google tiene un coste de 25\$ (actualmente de por vida) que debe pagarse con tarjeta de crédito.
- Los desarrolladores obtienen el 70% de los beneficios que produzcan sus aplicaciones. El 30% restante es para Google.
- Google Play paga a los desarrolladores a través de Google Checkout.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- ¿Cómo publicar una aplicación que hemos creado?
- Tecleamos en el buscador **Google Play Developer**, es la primera opción.

The screenshot shows the Google Play Developer Console registration process. At the top, there's a progress bar with four steps: 'Sign-in with your Google account', 'Accept Developer Agreement' (highlighted in blue), 'Pay Registration Fee', and 'Complete your Account details'. Below this, it says 'YOU ARE SIGNED IN AS...' followed by a profile card for 'Ismael Serrano Gracia' with email 'ismael.serrano.gracia@gmail.com'. To the right, it states 'This is the Google account that will be associated with your Developer Console.' and provides instructions for choosing an account. Below that, it says 'BEFORE YOU CONTINUE...' and lists three items: 'Read and agree to the Google Play Developer distribution agreement' (with a checkbox checked), 'Review the distribution countries where you can distribute and sell applications.', and 'Make sure you have your credit card handy to pay the \$25 registration fee in the next step.' At the bottom, there is a 'Continue to payment' button.

Google play | Developer Console

Sign-in with your Google account | **Accept Developer Agreement** | Pay Registration Fee | Complete your Account details

YOU ARE SIGNED IN AS...

Ismael Serrano Gracia
ismael.serrano.gracia@gmail.com

This is the Google account that will be associated with your Developer Console.

If you would like to use a different account, you can choose from the following options below. If you are an organization, consider registering a new Google account rather than using a personal account.

[Sign in with a different account](#) [Create a new Google account](#)

BEFORE YOU CONTINUE...

Read and agree to the [Google Play Developer distribution agreement](#).

☒ I agree and I am willing to associate my account registration with the Google Play Developer distribution agreement.

Review the distribution countries where you can distribute and sell applications.

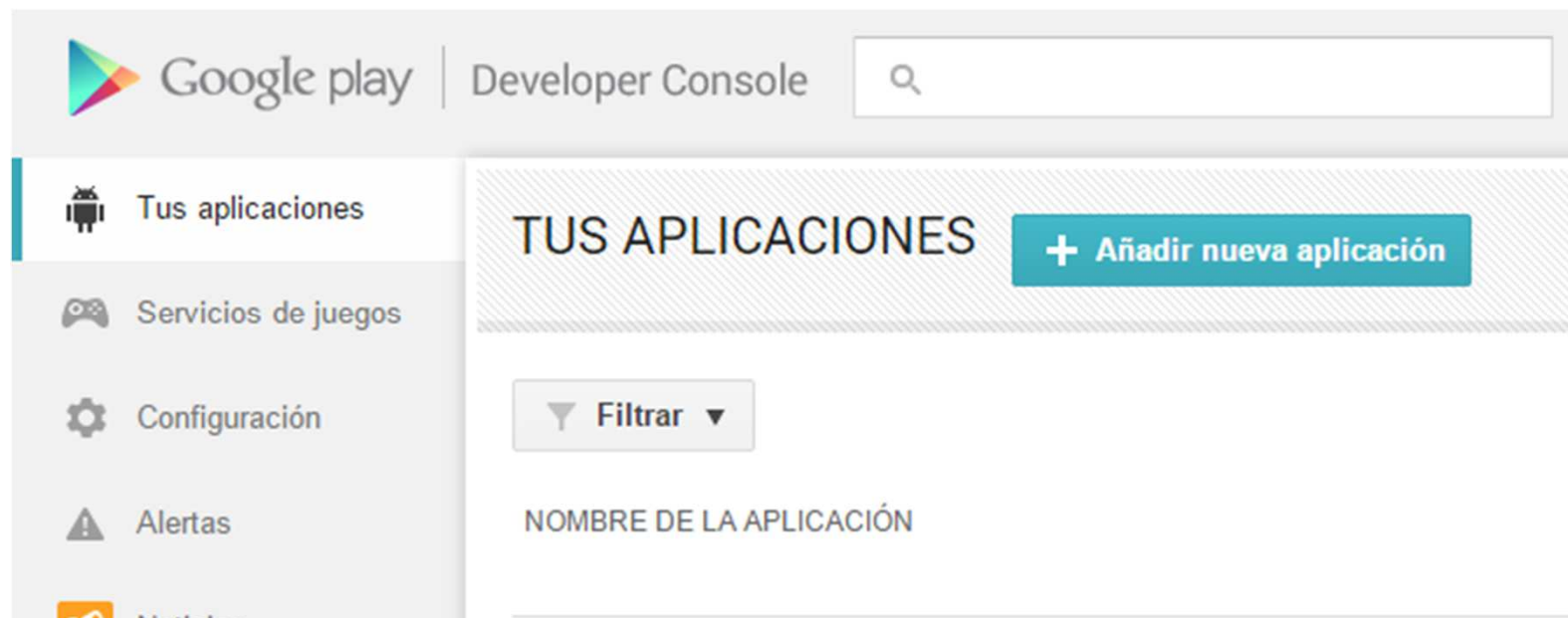
If you are planning to sell apps or in-app products, check if you can have a merchant account in your country.

\$25 Make sure you have your credit card handy to pay the \$25 registration fee in the next step.

[Continue to payment](#)

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Una vez nos registremos veremos lo siguiente:



5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Pinchamos en +Añadir nueva aplicación. Nos saldrá este mensaje:



AÑADIR NUEVA APLICACIÓN

Idioma predeterminado *

Español (Latinoamérica) – es-419 ▼

Nombre *

0 de 30 caracteres

¿Cómo te gustaría empezar?

Subir APK Preparar ficha de Play Store Cancelar

- Elegimos el idioma, ponemos un Nombre a nuestra aplicación y pulsamos sobre Prepara ficha Play Store.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Debemos completar la información sobre nuestra aplicación que nos va pidiendo...

LECTOR DE CÓDIGOS ISMAEL

FICHA DE PLAY STORE Guardar

INFORMACIÓN DEL PRODUCTO Para publicar la aplicación, debes rellenar los campos marcados con *.

Español (España) – es-ES

Añadir traducciones

Título*

Español (España) – es-ES

Lector de Códigos Ismael

24 de 30 caracteres

Descripción breve*

Español (España) – es-ES

Lector de Códigos Ismael

24 de 80 caracteres

Descripción completa*

Español (España) – es-ES


Un lector de códigos QR creado por Ismael

41 de 4000 caracteres

Consulta estas sugerencias sobre cómo crear descripciones de aplicaciones acordes a la política para evitar que la aplicación se suspenda por motivos comunes.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Ahora pulsamos en guardar y vamos a la siguiente ventana. Precio y distribución.



APK ☒

Ficha de Play Store ☒

Precio y distribución ☒

Productos integrados en la aplicación

Servicios y APIs

Sugerencias de optimización 1

LECTOR DE CÓDIGOS ISMAEL

FICHA DE PLAY STORE [Guardar](#)

INFORMACIÓN DEL PRODUCTO

Español (España) - es-ES [Añadir traducción](#)

Título*
Español (España) - es-ES [2]

Descripción breve*
Español (España) - es-ES []

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Aquí podremos seleccionar el precio de venta de la nuestra aplicación en euros. Podemos convertir automáticamente los precios a monedas diferentes de otros países. O elegir que sea gratuita.
- Además debemos seleccionar en que países se va a distribuir la aplicación.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Finalmente podremos subir nuestra aplicación (archivo .apk).

The screenshot shows the Google Play Console interface for the application 'LECTOR DE CÓDIGOS ISMAEL'. On the left is a sidebar menu with the following items: 'APK' (selected), 'Ficha de Play Store', 'Precio y distribución', 'Productos integrados en la aplicación', 'Servicios y APIs', and 'Sugerencias de optimización' (with a '1' badge). The main content area has a header 'LECTOR DE CÓDIGOS ISMAEL' and a sub-header 'APK'. Below this, there are three tabs: 'PRODUCCIÓN' (with the description 'Publicar tu aplicación en Google Play'), 'BETA TESTING' (with the description 'Configurar el betatesting de tu aplicación'), and 'ALPHA TESTING' (with the description 'Configurar el testing alpha de tu aplicación'). Below the tabs, there is a message box with an Android icon and a 'NEW' badge, stating: 'Ahora las claves de licencia se administran de forma individual para cada aplicación. Si tu aplicación utiliza servicios de licencias (por ejemplo, si se trata de una aplicación de pago o utiliza la facturación integrada en aplicaciones o archivos de expansión APK), puedes obtener la nueva clave de licencia en la página [Servicios y APIs](#).' At the bottom right of the main content area, there is a blue button that says 'Subir tu primer archivo APK en fase de producción'.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Podremos ver la hora a la que se ha subido la aplicación, los dispositivos compatibles, la versión, etc.

Ficha de Play Store

Precio y distribución

Productos integrados en la aplicación

Servicios y APIs

Sugerencias de optimización

PRODUCCIÓN

Versión
1

BETA TESTING
Configurar el betatesting de tu aplicación

ALPHA TESTING
Configurar el testing alpha de tu aplicación

CONFIGURACIÓN DE PRODUCCIÓN [Subir nuevo APK de producción](#)

APK ACTUAL subido el 27/11/2014 08:46:54

Dispositivos compatibles
7480
[Ver lista](#)

Dispositivos excluidos
0
[Administrar dispositivos excluidos](#)

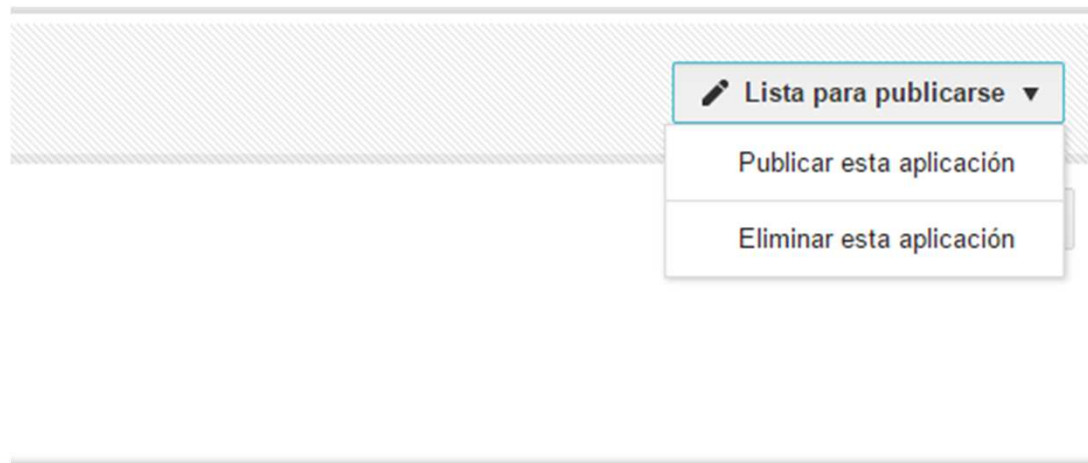
| ▼ VERSIÓN | FECHA DE SUBIDA | ESTADO |
|-----------|-----------------|------------------------|
| 1 (1.0) | 27/11/2014 | borrador en producción |

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Si en ESTADO aparece borrador en producción, esto significa que quedan información o imágenes por subir.
- Pulsamos en la derecha, en Borrador → ¿Por qué no puedo publicar la aplicación?
- Aquí nos dirá que nos falta.

5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Una vez añadida la información que faltaba podremos publicar nuestra aplicación.
- Botón arriba a la derecha Lista para publicarse → Publicar esta aplicación.



5- PUBLICAR APLICACIONES. GOOGLE PLAY

- Una vez publicada nuestra aplicación estará disponible en el Google Play en unas horas.

6- OTROS

- Vamos a ver algunas cuestiones de Android que no hemos visto antes, que no están dentro de ninguna de la categorías anterior.
- Vamos a ver:
 - Cambiar el icono de nuestra App.
 - Ocultar el teclado cuando queramos.

6- OTROS

- Vamos a ver como cambiar el icono de nuestra App. Para ello necesitamos una imagen cuadrada de formato png o jpg, de los siguiente tamaños:
 - Densidad Media (mdpi) → 48x48 px
 - Alta Densidad (hdpi) → 72x72 px
 - Extra Alta Densidad (xhdpi) → 96x96 px
 - Extra Extra Alta Densidad (xxhdpi) → 144x144 px
- Dentro del directorio «drawable».
(.\miProyecto\app\src\main\res\)

6- OTROS

- Otra opción es colocar en «mipmap-mdpi», «mipmap-hdpi», ... Una copia del icono del tamaño adecuado.
- De esta forma dependiendo de la resolución del dispositivo Android seleccionará la mejor.
- El última paso es dentro de archivo «AndroidManifest.xml», cambiamos la línea:
 - **android:icon="@drawable/mi_imagen"**

6- OTROS

- También vamos a ver como ocultar el teclado cuando queramos. Para ello debemos ejecutar esto:

```
InputMethodManager inputManager = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
```

```
inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), InputMethodManager.HIDE_NOT_ALWAYS);
```

- Podemos añadir esto dentro de un escuchador o donde queramos.