CAPÍTULO 7. Almacenamiento de datos

7.1. Añadiendo lista de puntuaciones en Asteroides

```
public interface AlmacenPuntuaciones {

   public void guardarPuntuacion(int puntos, String nombre,
     long fecha);

   public Vector<String> listaPuntuaciones(int cantidad);

}
```

```
public class AlmacenPuntuacionesConstante implements
AlmacenPuntuaciones{

   public void guardarPuntuacion(int puntos, String nombre,
     long fecha) {
   }

   public Vector<String> listaPuntuaciones(int cantidad) {
      Vector<String> result = new Vector<String>();
      result.add("123000 Pepito Domingez");
      result.add("111000 Pedro Martinez");
      result.add("511000 Dani Ferrito");
      return result;
   }
}
```

En la actividad `Asteroides` tendrás que declarar una variable para almacenar las puntuaciones:

```
public static AlmacenPuntuaciones almacen =
                            new AlmacenPuntuacionesConstante();
```

También hay que modificar el escuchador asociado al cuarto botón para que llame al siguiente método:

```
public void lanzarPuntuaciones() {
    Intent i = new Intent(this, Puntuaciones.class);
    startActivity(i);
}
```

El *layout* que utilizaremos para la nueva actividad se llamará `puntuaciones.xml`. Este se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/degradado">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Puntuaciones"
        android:gravity="center"
        android:layout_margin="10px"
        android:textSize="10pt""/>
    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1">
        <ListView
            android:id="@android:id/list"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:drawSelectorOnTop="false" />
        <TextView
            android:id="@android:id/empty"
            android:layout_width="fill_parent"
```

```
            android:layout_height="fill_parent"
            android:text="No hay puntuaciones" />
    </FrameLayout>
</LinearLayout>
```

Necesitamos ahora crear la actividad `Puntuaciones`. Crea una nueva clase en tu proyecto e introduce el siguiente código:

```
public class Puntuaciones extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.puntuaciones);
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1,
                Asteroides.almacen.listaPuntuaciones(10)));
        getListView().setTextFilterEnabled(true);
    }
}
```

7.2. **Comunicación entre actividades**

```
public void lanzaJuego() {
    Intent i = new Intent(this, Juego.class);
    startActivityForResult(i, 1234);
}

@Override
protected void onActivityResult (int requestCode,
                                 int resultCode, Intent data){
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==1234 & resultCode==RESULT_OK & data!=null) {
        int puntuacion = data.getExtras().getInt("puntuacion");
        if (puntuacion > 0) {
            String nombre = "Yo";
// Leerlo desde un Dialog o una nueva actividad AlertDialog.Builder
            almacen.guardarPuntuacion(puntuacion, nombre,
                                      System.currentTimeMillis());
            lanzarPuntuaciones();
        }
    }
}
```

Para realizar la respuesta de la actividad puede ser más sencillo hacerlo desde `VistaJuego`. El problema es que esta clase es un vista no una actividad. Para solucionar el problema puedes usar el siguiente truco. Introduce en `VistaJuego` el siguiente código:

```
private Activity padre;

public void setPadre(Activity padre) {
   this.padre = padre;
}
```

Cuando detectes una condición de victoria es un buen momento para almacenar la puntuación. Para ello, añade el siguiente código al final de `destruyeAsteroide()`. En caso de derrota habría que hacer lo mismo.

```
if (Asteroides.isEmpty() && padre != null) {
   Bundle bundle = new Bundle();
   bundle.putInt("puntuacion", puntuacion);
   Intent intent = new Intent();
   intent.putExtras(bundle);
   padre.setResult(Activity.RESULT_OK, intent);
}
```

En el método `onCreate` de `Juego` introduce:

```
vistaJuego.setPadre(this);
```

7.3. Accediendo a ficheros

7.3.1. Sistema interno de ficheros

```
String fichero = "fichero.txt";
String texto = "texto almacenado";
FileOutputStream fos;
try {
   fos = openFileOutput(fichero,Context.MODE_PRIVATE);
   fos.write(texto.getBytes());
   fos.close();
} catch (FileNotFoundException e) {
   Log.e("Mi Aplicación",e.getMessage(),e);
} catch (IOException e) {
   Log.e("Mi Aplicación",e.getMessage(),e);
}
```

7.3.2. Almacenando puntuaciones en el sistema de ficheros interno

El siguiente código muestra una clase que implementa la interfaz `AlmacenPuntuaciones` utilizando los métodos antes descritos.

```
public class AlmacenPuntuacionesFichero implements AlmacenPuntuaciones {
   private static String FICHERO = "puntuaciones.txt";
   private Context context;

   public AlmacenPuntuacionesFichero(Context context) {
      this.context = context;
   }

   public void guardarPuntuacion(int puntos, String nombre, long fecha){
      try {
         FileOutputStream fos = context.openFileOutput(FICHERO,
                                    Context.MODE_APPEND);
         String texto = puntos + " " + nombre + "\n";
         fos.write(texto.getBytes());
         fos.close();
      } catch (Exception e) {
         Log.e("Asteroides", e.getMessage(), e);
      }
   }

   public Vector<String> listaPuntuaciones(int cantidad) {
      Vector<String> result = new Vector<String>();
      try {
         FileInputStream fis = context.openFileInput(FICHERO);
         String cadena = "";
         int i;
         int n = 0;
         do {
            i = fis.read();
            if (i != '\n') {
               cadena = cadena + (char) i;
            } else {
               result.add(cadena);
               cadena = "";
               n++;
            }
```

```
            } while (i > 0 & n < cantidad);
            fis.close();
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
        return result;
    }
}
```

El constructor de esta clase necesita un contexto. Tendrás que añador en `oCreate()` de `Asterodes` la línea:

```
        almacen = new AlmacenPuntuacionesFichero(this);
```

7.3.3. La tarjeta SD

7.3.4. Acceder a un fichero de los recursos

```
        InputStream is;
        try {
            is = getResources().openRawResource(R.raw.fichero);
            byte[] reader = new byte[is.available()];
            while (is.read(reader) != -1) {
                …
            }
        } catch (IOException e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
```

7.4. Trabajando con XML

```
            <?xml version="1.0" encoding="UTF-8"?>
            <lista_puntuaciones>
                <puntuacion fecha="1288122023410">
                    <nombre>Mi nombre</nombre>
                    <puntos>45000</puntos>
                </puntuacion>
                <puntuacion fecha="1288122428132">
```

```
            <nombre>Otro nombre</nombre>
            <puntos>31000</puntos>
         </puntuacion>
      </lista_puntuaciones>
```

7.4.1. Procesando XML con SAX

```
public class AlmacenPuntuacionesXML_SAX implements AlmacenPuntuaciones {
   private static String FICHERO = "puntuaciones.xml";
   private Context contexto;
   private ListaPuntuaciones lista;
   private boolean cargadaLista;

   public AlmacenPuntuacionesXML_SAX(Context contexto) {
      this.contexto = contexto;
      lista = new ListaPuntuaciones();
      cargadaLista = false;
   }

   @Override
   public void guardarPuntuacion(int puntos, String nombre,
                                                 long fecha) {
      try {
         if (!cargadaLista){
            lista.leerXML(contexto.openFileInput(FICHERO));
         }
      } catch (FileNotFoundException e) {
      } catch (Exception e) {
         Log.e("Asteroides", e.getMessage(), e);
      }
      lista.nuevo(puntos, nombre, fecha);
      try {
         lista.escribirXML(contexto.openFileOutput(FICHERO,
                                        Context.MODE_PRIVATE));
      } catch (Exception e) {
         Log.e("Asteroides", e.getMessage(), e);
      }
   }

   @Override
   public Vector<String> listaPuntuaciones(int cantidad) {
      try {
         if (!cargadaLista){
            lista.leerXML(contexto.openFileInput(FICHERO));
```

```
            }
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
        return lista.aVectorString();
    }


    private class ListaPuntuaciones {

        private class Puntuacion {
            int puntos;
            String nombre;
            long fecha;
        }

        private List<Puntuacion> listaPuntuaciones;

        public ListaPuntuaciones() {
            listaPuntuaciones = new ArrayList<Puntuacion>();
        }

        public void nuevo(int puntos, String nombre, long fecha) {
            Puntuacion puntuacion = new Puntuacion();
            puntuacion.puntos = puntos;
            puntuacion.nombre = nombre;
            puntuacion.fecha = fecha;
            listaPuntuaciones.add(puntuacion);
        }

        public Vector<String> aVectorString() {
            Vector<String> result = new Vector<String>();
            for (Puntuacion puntuacion : listaPuntuaciones) {
                result.add(puntuacion.nombre+" "+puntuacion.puntos);
            }
            return result;
        }
```

```
public void leerXML(InputStream entrada) throws Exception {
    SAXParserFactory fabrica = SAXParserFactory.newInstance();
    SAXParser parser = fabrica.newSAXParser();
    XMLReader lector = parser.getXMLReader();
    ManejadorXML manejadorXML = new ManejadorXML();
    lector.setContentHandler(manejadorXML);
    lector.parse(new InputSource(entrada));
    cargadaLista = true;
}



class ManejadorXML extends DefaultHandler {
    private StringBuilder cadena;
    private Puntuacion puntuacion;

    @Override
    public void startDocument() throws SAXException {
        listaPuntuaciones = new ArrayList<Puntuacion>();
        cadena = new StringBuilder();
    }

    @Override
    public void startElement(String uri, String nombreLocal, String
              nombreCualif, Attributes atr) throws SAXException {
        if (nombreLocal.equals("puntuacion")) {
            puntuacion = new Puntuacion();
            puntuacion.fecha = Long.parseLong(atr.getValue("fecha"));
        }
    }

    @Override
    public void characters(char ch[], int comienzo, int long) {
        cadena.append(ch, comienzo, long);
    }

    @Override
    public void endElement(String uri, String nombreLocal,
           String nombreCualif) throws SAXException {
        if (nombreLocal.equals("puntos")) {
            puntuacion.puntos = Integer.parseInt(cadena.toString());
        } else if (nombreLocal.equals("nombre")) {
            puntuacion.nombre = cadena.toString();
        } else if (nombreLocal.equals("puntuacion")) {
```

```
                listaPuntuaciones.add(puntuacion);
            }
            cadena.setLength(0);
        }
    }

    @Override
    public void endDocument() throws SAXException {
    }
```

Veamos el último método de la clase `ListaPuntuaciones`, que nos permite escribir el documento XML:

```
    public void escribirXML(OutputStream salida) {
        XmlSerializer serializador = Xml.newSerializer();
        try {
            serializador.setOutput(salida, "UTF-8");
            serializador.startDocument("UTF-8", true);
            serializador.startTag("", "lista_puntuaciones");
            for (Puntuacion puntuacion : listaPuntuaciones) {
                serializador.startTag("", "puntuacion");
                serializador.attribute("", "fecha",
                                       String.valueOf(puntuacion.fecha));
                serializador.startTag("", "nombre");
                serializador.text(puntuacion.nombre);
                serializador.endTag("", "nombre");
                serializador.startTag("", "puntos");
                serializador.text(String.valueOf(puntuacion.puntos));
                serializador.endTag("", "puntos");
                serializador.endTag("", "puntuacion");
            }
            serializador.endTag("", "lista_puntuaciones");
            serializador.endDocument();
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
    }
  }
}
```

7.4.2. Procesando XML con DOM

```
public class AlmacenPuntuacionesXML_DOM implements AlmacenPuntuaciones{
    private static String FICHERO = "puntuaciones.xml";
    private Context contexto;
    private Document documento;
    private boolean cargadoDocumento;

    public AlmacenPuntuacionesXML_DOM(Context contexto) {
        this.contexto = contexto;
        cargadoDocumento = false;
    }

    @Override
    public void guardarPuntuacion(int puntos, String nombre, long fecha){
        try {
            if (!cargadoDocumento) {
                leerXML(contexto.openFileInput(FICHERO));
            }
        } catch (FileNotFoundException e) {
            crearXML();
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
        nuevo(puntos, nombre, fecha);
        try {
            escribirXML(contexto.openFileOutput(FICHERO,
                                                Context.MODE_PRIVATE));
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
    }
    @Override
    public Vector<String> listaPuntuaciones(int cantidad) {
        try {
            if (!cargadoDocumento) {
                leerXML(contexto.openFileInput(FICHERO));
            }
        } catch (FileNotFoundException e) {
            crearXML();
        } catch (Exception e) {
            Log.e("Asteroides", e.getMessage(), e);
        }
        return aVectorString();
    }
```

```
public void crearXML() {
    try {
        DocumentBuilderFactory fabrica =
                                DocumentBuilderFactory.newInstance();
        DocumentBuilder constructor = fabrica.newDocumentBuilder();
        documento = constructor.newDocument();
        Element raiz = documento.createElement("lista_puntuaciones");
        documento.appendChild(raiz);
        cargadoDocumento = true;
    } catch (Exception e) {
        Log.e("Asteroides", e.getMessage(), e);
    }
}

public void leerXML(InputStream entrada) throws Exception {
    DocumentBuilderFactory fabrica =
                                DocumentBuilderFactory.newInstance();
    DocumentBuilder constructor = fabrica.newDocumentBuilder();
    documento = constructor.parse(entrada);
    cargadoDocumento = true;
}



public void nuevo(int puntos, String nombre, long fecha) {
    Element puntuacion = documento.createElement("puntuacion");
    puntuacion.setAttribute("fecha", String.valueOf(fecha));
    Element e_nombre = documento.createElement("nombre");
    Text texto = documento.createTextNode(nombre);
    e_nombre.appendChild(texto);
    puntuacion.appendChild(e_nombre);
    Element e_puntos = documento.createElement("puntos");
    texto = documento.createTextNode(String.valueOf(puntos));
    e_puntos.appendChild(texto);
    puntuacion.appendChild(e_puntos);
    Element raiz = documento.getDocumentElement();
    raiz.appendChild(puntuacion);
}

public Vector<String> aVectorString() {
    Vector<String> result = new Vector<String>();
    String nombre = "", puntos = "";
    Element raiz = documento.getDocumentElement();
    NodeList puntuaciones = raiz.getElementsByTagName("puntuacion");
```

```
    for (int i = 0; i < puntuaciones.getLength(); i++) {
        Node puntuacion = puntuaciones.item(i);
        NodeList propiedades = puntuacion.getChildNodes();
        for (int j = 0; j < propiedades.getLength(); j++) {
            Node propiedad = propiedades.item(j);
            String etiqueta = propiedad.getNodeName();
            if (etiqueta.equals("nombre")) {
                nombre = propiedad.getFirstChild().getNodeValue();
            } else if (etiqueta.equals("puntos")) {
                puntos = propiedad.getFirstChild().getNodeValue();
            }
        }
        result.add(nombre + " " + puntos);
    }
    return result;
}
```

```
public void escribirXML(OutputStream salida) throws Exception {
    TransformerFactory fabrica = TransformerFactory.newInstance();
    Transformer transformador = fabrica.newTransformer();
    transformador.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,"yes");
    transformador.setOutputProperty(OutputKeys.INDENT, "yes");
    DOMSource fuente = new DOMSource(documento);
    StreamResult resultado = new StreamResult(salida);
    transformador.transform(fuente, resultado);
}
}
```

```
public void escribirXML(OutputStream salida) throws Exception {
    String s = serializa(documento.getDocumentElement());
    salida.write(s.getBytes("UTF-8"));
}

public static String serializa(Node raiz) throws IOException {
    StringBuilder resultado = new StringBuilder();
    if (raiz.getNodeType() == Node.TEXT_NODE)
        resultado.append(raiz.getNodeValue());
    else {
        if (raiz.getNodeType() != Node.DOCUMENT_NODE) {
            StringBuffer atributos = new StringBuffer();
```

```
            for (int i = 0; i < raiz.getAttributes().getLength(); ++i)
                  atributos.append(" ")
                  .append(raiz.getAttributes().item(i).getNodeName())
                  .append("=\"")
                  .append(raiz.getAttributes().item(i).getNodeValue())
                  .append("\" ");
               }
               resultado.append("<").append(raiz.getNodeName())
                  .append(" ").append(atributos).append(">");
            } else {
               Resultado
                  .append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
            }
            NodeList listaNodos = raiz.getChildNodes();
            for (int i = 0; i < listaNodos.getLength(); i++) {
               Node nodo = listaNodos.item(i);
               resultado.append(serializa(nodo));
            }
            if (raiz.getNodeType() != Node.DOCUMENT_NODE) {
               resultado.append("</").append(raiz.getNodeName())
                  .append(">");
            }
         }
         return resultado.toString();
      }
```

7.5. **Bases de datos**

7.5.1. Utilizando una base de datos para guardar puntuaciones

```
public class AlmacenPuntuacionesSQLite extends SQLiteOpenHelper
                                    implements AlmacenPuntuaciones{
   //Métodos de SQLiteOpenHelper
   public AlmacenPuntuacionesSQLite(Context context) {
      super(context, "puntuaciones", null, 1);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE puntuaciones ("+
            "_id INTEGER PRIMARY KEY AUTOINCREMENT, "+
            "puntos INTEGER, nombre TEXT, fecha LONG)");
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
   // En caso de una nueva versión habría que actualizar las tablas
   }

   //Métodos de AlmacenPuntuaciones
   public void guardarPuntuacion(int puntos, String nombre,
                                                      long fecha) {
      SQLiteDatabase db = getWritableDatabase();
      db.execSQL("INSERT INTO puntuaciones VALUES ( null, "+
            puntos+", '"+nombre+"', "+fecha+")");
   }

   public Vector<String> listaPuntuaciones(int cantidad) {
      String[] CAMPOS = {"puntos", "nombre"};
      Vector<String> result = new Vector<String>();
      SQLiteDatabase db = getReadableDatabase();
      Cursor cursor=db.query("puntuaciones", CAMPOS, null, null,
               null, null, "puntos", Integer.toString(cantidad));
       while (cursor.moveToNext()){
            result.add(cursor.getInt(0)+" " +cursor.getString(1));
      }
      return result;
   }
}
```

7.6. Utilizando la clase ContentProvider

7.6.1. Conceptos básicos

7.6.1.1. El modelo de datos

7.6.1.2. Las URI

7.6.2. Acceder a la información de un ContentProvider

7.6.2.1. Leer información de un ContentProvider

Veamos un ejemplo que permite leer el registro de llamadas del teléfono. Crea una nueva aplicación con los siguientes datos:

```
Project name: ContentCallLog
Build Target: Android 1.5
Application name: ContentCallLog
Package name: org.example.puntuacionesprovider
Create Activity: ContentCallLog
Min SDK Version: 3
```

Añade en el fichero `res/layout/main.xml` la línea subrayada:

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

xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:id="@+id/salida"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

```
...
String[] TIPO_LLAMADA = {"","entrante","saliente","perdida"};
TextView salida = (TextView) findViewById(R.id.salida);

Uri llamadas = Uri.parse("content://call_log/calls");
Cursor c = managedQuery(llamadas, null, null, null, null);
while (c.moveToNext()) {
   salida.append("\n"
      + DateFormat.format("dd/MM/yy k:mm (",
               c.getLong(c.getColumnIndex(Calls.DATE)))
      + c.getString(c.getColumnIndex(Calls.DURATION)) + ") "
      + c.getString(c.getColumnIndex(Calls.NUMBER)) + ", "
      + TIPO_LLAMADA[Integer.parseInt(c.getString(c
                              .getColumnIndex(Calls.TYPE)))]);
   }
```

Para ilustrar el uso de estos parámetros reemplaza en el ejemplo anterior la línea,

```
Cursor c = managedQuery(llamadas, null, null, null, null);
```

por,

```
String[] proyeccion = new String[] {
      Calls.DATE, Calls.DURATION, Calls.NUMBER, Calls.TYPE };
   String[] argsSelecc = new String[] {"1"};
   Cursor c = managedQuery(
      llamadas,      // Uri del ContentProvider
      proyeccion,    // Columnas que nos interesan
      "type = ?",    // consulta WHERE
      argsSelecc,    // parámetros de la consulta anterior
      "date DESC");  // Ordenado por fecha, orden ascenciente
```

7.6.2.2. Escribir información en un ContentProvider

Añadir un nuevo elemento en un *ContentProvider* resulta muy sencillo. Para ilustrar como se hace escribe el siguiente código al principio del ejemplo anterior:

```
ContentValues valores = new ContentValues();
valores.put(Calls.DATE, new Date().getTime() );
valores.put(Calls.NUMBER, "555555555");
```

```
valores.put(Calls.DURATION, "55");
valores.put(Calls.TYPE, Calls.INCOMING_TYPE);

Uri nuevoElemento = getContentResolver().insert(
                                    Calls.CONTENT_URI, valores);
```

7.6.2.3. Borrar y modificar elementos de un ContentProvider

```
getContentResolver().delete(Calls.CONTENT_URI,
                                "number='555555555'", null);
```

También puedes utilizar el método `update()` para modificar elementos de un *ContentProvider*:

```
int ContentProvider.update(Uri uri, ContentValues valores,
                           String seleccion, String[] argsSelecc)
```

Por ejemplo, si quisiéramos modificar los registros con número 555555555, por el número 444444444, escribiríamos:

```
ContentValues valores2 = new ContentValues();
valores2.put(Calls.NUMBER, "444444444");
getContentResolver().update(Calls.CONTENT_URI, valores2,
                                    "number='555555555'", null);
```

7.6.3. Creación de un ContentProvider

Crea una nueva aplicación con los siguientes datos:

```
Project name: PuntuacionesProvider
Build Target: Android 1.5
Application name: PuntuacionesProvider
Package name: org.example.puntuacionesprovider
Create Activity: ActividadPrincipal
Min SDK Version: 3
```

7.6.3.1. Definir la estructura de almacenamiento del ContentProvider

Para crear la base de datos de nuestro *ContentProvider* añade una nueva clase con nombre `PuntuacionesSQLiteHelper` e introduce el siguiente código:

```
public class PuntuacionesSQLiteHelper extends SQLiteOpenHelper {

   public PuntuacionesSQLiteHelper(Context context) {
      super(context, "puntuaciones", null, 1);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL("CREATE TABLE puntuaciones ("
            + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "puntos INTEGER, "
            + "nombre TEXT, "
            + "fecha LONG)");
   }

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion,
                                                   int newVersion) {
   // En caso de una nueva versión habría que actualizar las tablas
   }
}
```

7.6.3.2. Extendiendo la clase ContentProvider

```
public class PuntuacionesProvider extends ContentProvider {

   public static final String AUTORIDAD =
                     "org.example.puntuacionesprovider";
   public static final Uri CONTENT_URI = Uri.parse("content://"
                     + AUTORIDAD + "/puntuaciones");
   public static final int TODOS_LOS_ELEMENTOS = 1;
   public static final int UN_ELEMENTO = 2;
   private static UriMatcher URI_MATCHER = null;
   static {
      URI_MATCHER = new UriMatcher(UriMatcher.NO_MATCH);
      URI_MATCHER.addURI(AUTORIDAD, "puntuaciones",
                                                TODOS_LOS_ELEMENTOS);
      URI_MATCHER.addURI(AUTORIDAD, "puntuaciones/#", UN_ELEMENTO);
   }
   public static final String TABLA = "puntuaciones";
   private SQLiteDatabase baseDeDatos;
```

```java
@Override
public boolean onCreate() {
    PuntuacionesSQLiteHelper dbHelper = new
                    PuntuacionesSQLiteHelper(getContext());
    baseDeDatos = dbHelper.getWritableDatabase();
    return baseDeDatos != null && baseDeDatos.isOpen();
}
```

```java
@Override
public String getType(final Uri uri) {
    switch (URI_MATCHER.match(uri)) {
    case TODOS_LOS_ELEMENTOS:
        return "vnd.android.cursor.dir/vnd.org.example.puntuacion";
    case UN_ELEMENTO:
        return "vnd.android.cursor.item/vnd.org.example.puntuacion";
    default:
        throw new IllegalArgumentException("URI incorrecta: " + uri);
    }
}
```

```java
@Override
public Cursor query(Uri uri, String[] projection, String
seleccion,
          String[] argSeleccion, String orden) {
    SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
    queryBuilder.setTables(TABLA);
    switch (URI_MATCHER.match(uri)) {
    case TODOS_LOS_ELEMENTOS:
        break;
    case UN_ELEMENTO:
        String id = uri.getPathSegments().get(1);
        queryBuilder.appendWhere("_id = " + id);
        break;
    default:
        throw new IllegalArgumentException("URI incorrecta "+uri);
    }
    return queryBuilder.query(baseDeDatos, proyeccion, seleccion,
          argSeleccion, null, null, orden);
}
```

```
    @Override
    public Uri insert(Uri uri, ContentValues valores) {
        long IdFila = baseDeDatos.insert(TABLA, null, valores);
        if (IdFila > 0) {
            return ContentUris.withAppendedId(CONTENT_URI, IdFila);
        } else {
            throw new SQLException("Error al insertar registro en "+uri);
        }
    }

    @Override
    public int delete(Uri uri, String seleccion, String[] argSeleccion) {
        switch (URI_MATCHER.match(uri)) {
        case TODOS_LOS_ELEMENTOS:
            break;
        case UN_ELEMENTO:
            String id = uri.getPathSegments().get(1);
            if (TextUtils.isEmpty(seleccion)) {
                seleccion = "_id = " + id;
            } else {
                seleccion = "_id = " + id + " AND (" + seleccion + ")";
            }
            break;
        default:
            throw new IllegalArgumentException("URI incorrecta: " + uri);
        }
        return baseDeDatos.delete(TABLA, seleccion, argSeleccion);
    }


@Override
public int update(Uri uri, ContentValues valores, String seleccion,
        String[] ArgumentosSeleccion) {
    switch (URI_MATCHER.match(uri)) {
    case TODOS_LOS_ELEMENTOS:
        break;
    case UN_ELEMENTO:
        String id = uri.getPathSegments().get(1);
        if (TextUtils.isEmpty(seleccion)) {
            seleccion = "_id = " + id;
        } else {
            seleccion = "_id = " + id + " AND (" + seleccion + ")";
        }
        break;
```

```
    default:
        throw new IllegalArgumentException("URI incorrecta: " + uri);
    }
    return baseDeDatos.update(TABLA, valores, seleccion,
                                                    ArgumentosSeleccion);
}
}
```

7.6.3.3. Declarar el ContentProvider en AndroidManifest.xml

```
<provider
  android:authorities="org.example.puntuacionesprovider"
  android:name="org.example.puntuacionesprovider.PuntuacionesProvider"
/>
```

7.6.4. Acceso a PuntuacionesProvider desde Asteroides

Crea una nueva clase en la aplicación `Asteroides` con el nombre `AlmacenPuntuacionesProvider`. Introduce el siguiente código:

```
public class AlmacenPuntuacionesProvider
                                    implements AlmacenPuntuaciones {
private Activity activity;

public AlmacenPuntuacionesProvider(Activity activity) {
    this.activity = activity;
}

public void guardarPuntuacion(int puntos, String nombre, long fecha)
{
    Uri uri = Uri.parse(
        "content://org.example.puntuacionesprovider/puntuaciones");
    ContentValues valores = new ContentValues();
    valores.put("nombre", nombre);
    valores.put("puntos", puntos);
    valores.put("fecha", fecha);
    try {
        activity.getContentResolver().insert(uri, valores);
    } catch (Exception e) {
        Toast.makeText(
              activity,
              "Verificar que está instalado ”+
                        “org.example.puntuacionesprovider",
```

```
            Toast.LENGTH_LONG).show();
        Log.e("Asteroides", "Error: " + e.toString(), e);
    }
}

public Vector<String> listaPuntuaciones(int cantidad) {
    Vector<String> result = new Vector<String>();
    Uri uri = Uri.parse(
        "content://org.example.puntuacionesprovider/puntuaciones");
    Cursor cursor = activity.managedQuery(uri, null, null,
                                          null, "fecha DESC");
    if (cursor != null) {
        while (cursor.moveToNext()) {
            String nombre = cursor.getString(
                            cursor.getColumnIndex("nombre"));
            int puntos = cursor.getInt(
                            cursor.getColumnIndex("puntos"));
            result.add(puntos + " " + nombre);
        }
        cursor.close();
    }
    return result;
}
}
```

Para utilizar esta nueva clase, modifica en el principio de la clase `Asteroides` y de la clase `Puntuaciones` la declaración de la variable `almacen` por:

```
AlmacenPuntuaciones almacen = new AlmacenPuntuacionesProvider(this);
```