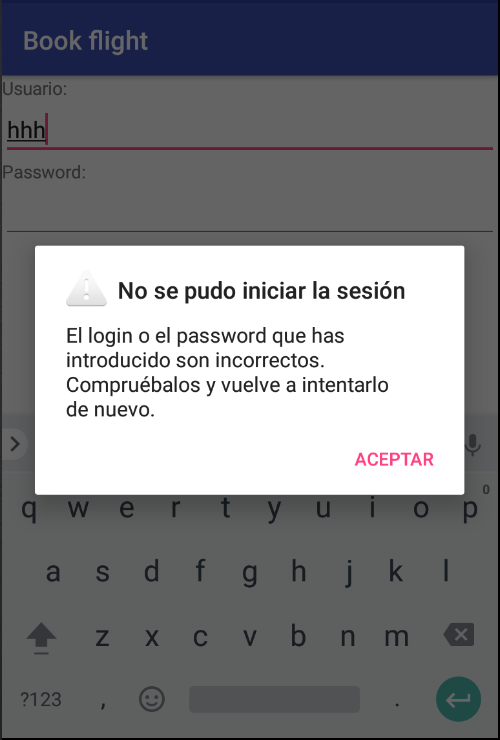
Actividad 1: Ventanas de diálogo. Intents. Logcat. Ficheros. Bases de datos. Consulta.

|  |
| --- |
| [1. Documentación. 1](#_Toc57286908)  [1.1. Ventanas de diálogo. 1](#_Toc57286909)  [1.2. Intents. 3](#_Toc57286910)  [1.3. Logcat. 4](#_Toc57286911)  [1.4. Ficheros. 4](#_Toc57286912)  [1.5. Bases de datos. 4](#_Toc57286913)  [2. Base de datos. 4](#_Toc57286914)  [2.1. SQLiteStudio. 5](#_Toc57286915)  [3. Programa. 11](#_Toc57286916)  [4. Funcionamiento. 16](#_Toc57286917)  [5. Anexos. 17](#_Toc57286918)  [Anexo 1: Ejecución de instrucciones SQL. 17](#_Toc57286919)  [Anexo 2: Depuración del programa. 19](#_Toc57286920)  [Anexo 3: Class literal *.class*. 20](#_Toc57286921) |

# 1. Documentación.

## 1.1. Ventanas de diálogo.

* Ver a partir de DialogFragment en la wiki:
  + [https://manuais.iessanclemente.net/index.php/Vent%C3%A1s\_de\_Di%C3%A1logos#DialogFragment](https://manuais.iessanclemente.net/index.php/Ventás_de_Diálogos" \l "DialogFragment)
* Por lo tanto, para crear ventanas de diálogo no se utilizará el método *showDialog* de *Activity*. Esto es debido a que esa manera de hacerlo está obsoleta a partir de API 13. En su lugar se utilizará la clase *DialogFragment*.
* Ejemplo.



* Clase *MensajeDialogo*.

|  |
| --- |
| import android.app.AlertDialog;  import android.app.Dialog;  import android.content.DialogInterface;  import android.content.DialogInterface.OnClickListener;  import android.os.Bundle;  import android.support.v4.app.DialogFragment;  import android.widget.Toast;  public class MensajeDialogo extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {  AlertDialog.Builder builder = new AlertDialog.Builder(getActivity())  .setTitle("No se pudo iniciar la sesión").setIcon(android.R.drawable.ic\_dialog\_alert)  .setMessage("El login o el password que has introducido son incorrectos. Compruébalos y vuelve a intentarlo de nuevo.")  .setPositiveButton("ACEPTAR", new OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  //Código asociado al botón ACEPTAR. Por ejemplo:  //Toast.makeText(getActivity(), "PULSADO BOTÓN ACEPTAR", Toast.LENGTH\_LONG).show();  }  });  return builder.create();  }  } |

* Con respecto al código:
  + Es posible encadenar los métodos *set* debido a que devuelven siempre un *AlertDialog.Builder*.
  + El método devuelve el resultado de llamar a *create*, el cual devuelve un *AlertDialog*, que es a su vez subclase de *Dialog*.
* Como se indica en la documentación de la Wiki, la ventana de diálogo se puede crear de dos maneras. En el ejemplo se optará por la segunda.
  + Sobreescribiendo el método *onCreateView*
    - Este método devuelve un view que será la ventana de diálogo.
    - Se deberá crear un layout con el contenido de la ventana de diálogo.
    - Se mostrará el view que conforma el diálogo mediante el layout asociado a la ventana de diálogo.
  + Sobreescribiendo el método *onCreateDialog*
    - No es necesario crear un layout asociado a la ventana de diálogo.
    - Se creará directamente el diálogo haciendo uso de la clase *AlertDialog*.
    - Es el método utilizado en el ejemplo anterior.
* Android incorpora un conjunto de drawables que pueden ser utilizados desde las apps sin necesidad de añadir imágenes como nuevos recursos drawable. Se hará uso de ellos para el icono de la ventana de diálogo, mediante el método *setIcon*.
  + Se puede encontrar información acerca de los drawables de Android en: [enlace1](http://gae.darshancomputing.com/android/1.5-drawables.html) y [enlace2](https://developer.android.com/reference/android/R.drawable).
  + Ejemplos de uso:
    - * En Java.

|  |
| --- |
| myMenuItem.setIcon(android.R.drawable.ic\_menu\_save); |

* + - * En XML (no olvidar la @ al comienzo).

|  |
| --- |
| android:icon="@android:drawable/ic\_menu\_save" |

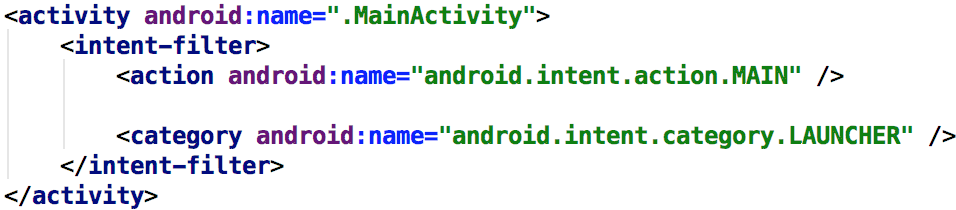
* En la activity se usará el siguiente código para mostrar la ventana de diálogo.

|  |
| --- |
| MensajeDialogo d= new MensajeDialogo();  FragmentManager fm= this.getSupportFragmentManager();  d.show(fm,"errorLogin"); |

* Para llamar llamar al *FragmentDialog* desde una *AppCompatActivity* se hará uso de uno de los dos siguientes métodos:
  + *getSupportFragmentManager()*
    - Es el caso del ejemplo anterior y del ejemplo de la wiki.
  + *getFragmentManager()*
    - En caso de que se hubiese importado la clase *android.app.DialogFragment*
* *AppCompatActivity* es subclase de *FragmentActivity*.
* El método *show(fm, tag)* de la clase *FragmentDialog* recibe una etiqueta (*tag*) que se le asignará al diálogo que se muestra. El tag permite la posibilidad de acceder a ese diálogo en otro punto del código mediante el método *findFragmentByTag(tag)* de la clase *FragmentManager*. En esta práctica no se usa el tag.

## 1.2. Intents.

* Concepto de intent.
  + <https://manuais.iessanclemente.net/index.php/Intents>
* Intents explícitos: acceder al siguiente enlace y probar ejemplo *u3\_20\_intents*.
  + [https://manuais.iessanclemente.net/index.php/Intents\_expl%C3%ADcitos](https://manuais.iessanclemente.net/index.php/Intents_explícitos)
  + Se caracterizan por llamar a una clase Java en concreto, ya sea de la propia app o de otra app distinta. La clase Java es una activity.
  + En *AndroidManifest.xml* se añade automáticamente una entrada por cada activity del app. La activity principal tiene un filtro en el que se establece como principal (MAIN) y que a través de ella se lanza el programa (LAUNCHER).



* + Ejemplo de llamada a una clase de la propia app.
    - Código en la activity que llama a otra pasándole información (nombre e id).

|  |
| --- |
| Intent intent = new Intent(this, BusquedaVuelos.class);  intent.putExtra(this.NOMBRE, usr.getNombre());  intent.putExtra(this.ID, usr.getId());  startActivity(intent); |

* + - Código en la activity que es llamada.

|  |
| --- |
| Intent intent = getIntent();  String nuevoTitulo=getResources().getText(R.string.app\_name)  +": "+intent.getExtras().getString(MainActivity.NOMBRE); |

* + Ejemplo de llamada a una clase de otra app diferente.
    - En concreto, llama a la calculadora.

|  |
| --- |
| Intent intent = new Intent();  intent.setClassName("com.android.calculator2", "com.android.calculator2.Calculator");  startActivity(intent); |

* Intents implícitos.
  + [Enlace](https://manuais.iessanclemente.net/index.php/Intents_implícitos._Permisos._Filtros.).
  + En vez de invocar una clase Java, se invoca una acción. El sistema ya elige el app y la activity destino.
  + El sistema asocia las actividades y acciones a través de los filtros definidos en *AndroidManifest.xml*.
  + Ejemplo de llamada al navegador web.

|  |
| --- |
| Intent intent = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://www.xunta.es/"));  startActivity(intent); |

## 1.3. Logcat.

* [https://manuais.iessanclemente.net/index.php/Depuraci%C3%B3n:\_LogCat](https://manuais.iessanclemente.net/index.php/Depuración:_LogCat)

## 1.4. Ficheros.

* <https://manuais.iessanclemente.net/index.php/Ficheiros>
* <https://manuais.iessanclemente.net/index.php/PDM_Avanzado_Datos_Persistentes_Arquivos>

## 1.5. Bases de datos.

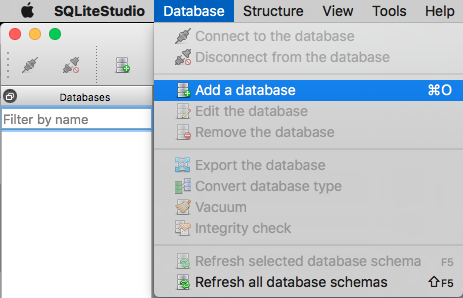
* <http://manuais.iessanclemente.net/index.php/PDM_Avanzado_Datos_Bases_de_datos>

# 2. Base de datos.

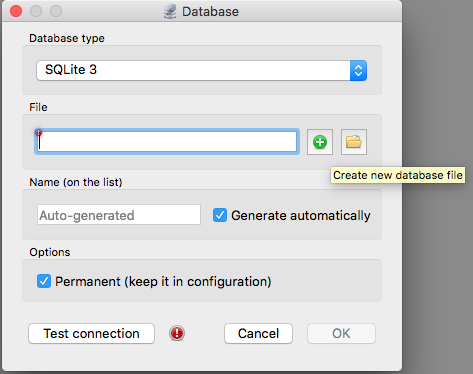
* Para saber más y ver ejemplos sobre acceso a bases de datos ir al enlace de la wiki: [enlace](https://wiki.cifprodolfoucha.es/index.php?title=PDM_Avanzado_Datos_Bases_de_datos).
* Para nuestra Actividad se va a crear la base de datos SQLite *appvuelos.db*.
* Tablas:
  + *usuario*.
    - Campos:
      * nombre
      * login
      * password
      * id

## 2.1. SQLiteStudio.

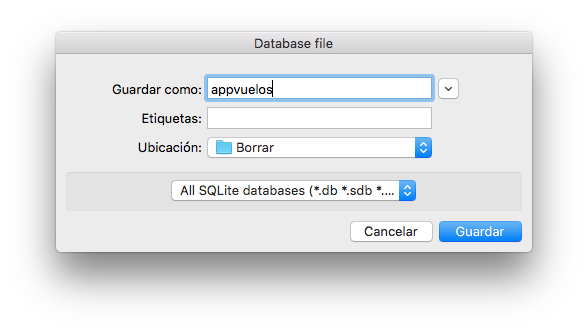
* Vamos a usar el gestor SQLiteStudio (descargar zip y descomprimir, ejecutar fichero SQLiteStudio.exe)
  + <http://sqlitestudio.pl/>
* Creación de una base de datos mediante SQLiteStudio.
  + Database → Add a database



* + Creación del archivo de la base de datos, selección de una ubicación adecuada para almacenarlo. Testear la conexión a la base de datos.

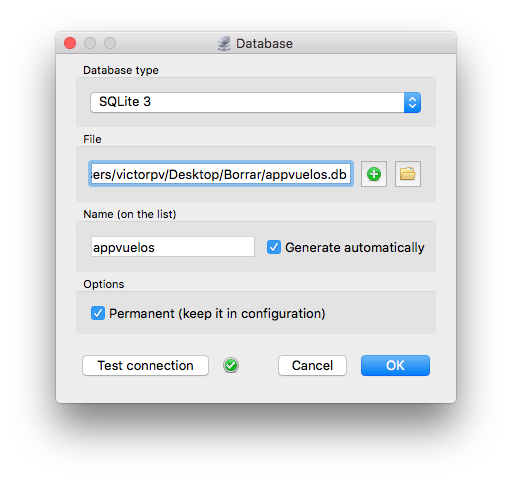


1

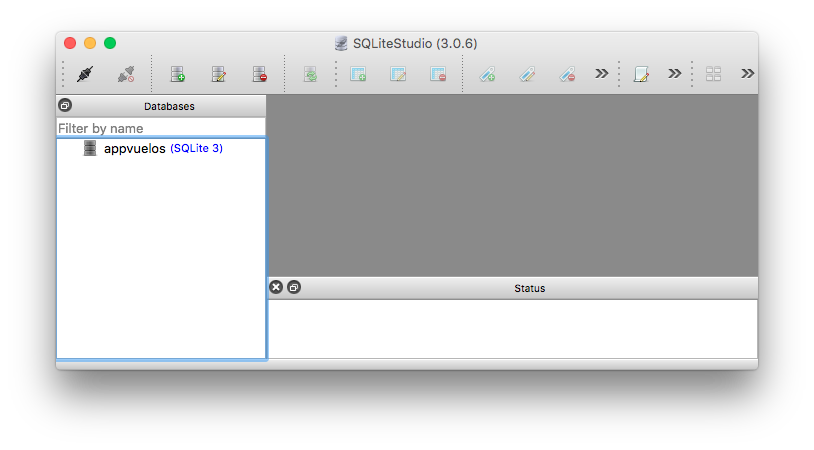


2

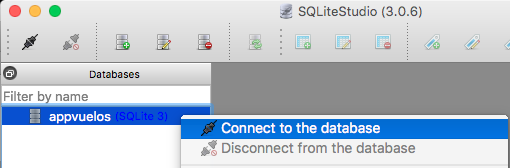
3



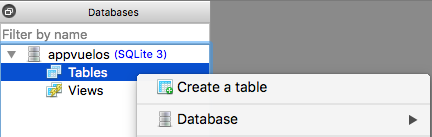
4



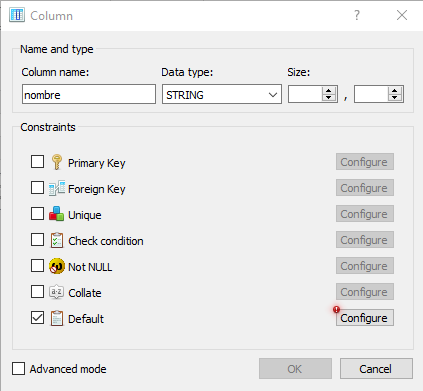
* Conectar a la base de datos.
  + Click derecho en la base de datos → Connect to the database

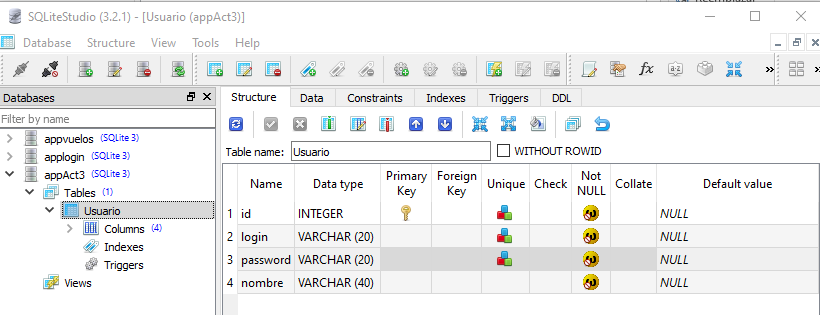


* Crear una tabla.
  + Click derecho en Tables → Create a table

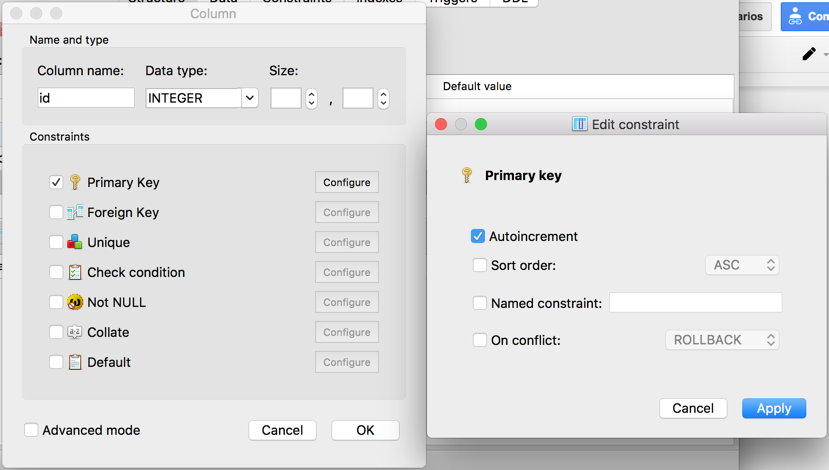


* + Doble click para añadir columnas (campos) a la tabla:

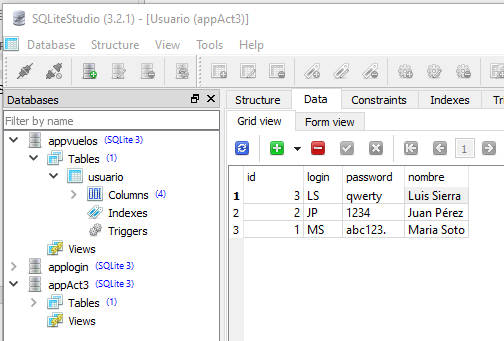




* Haciendo doble clic en el nombre del campo, se abre una ventana en la que se puede definir *id* como primary key y autoincrement.

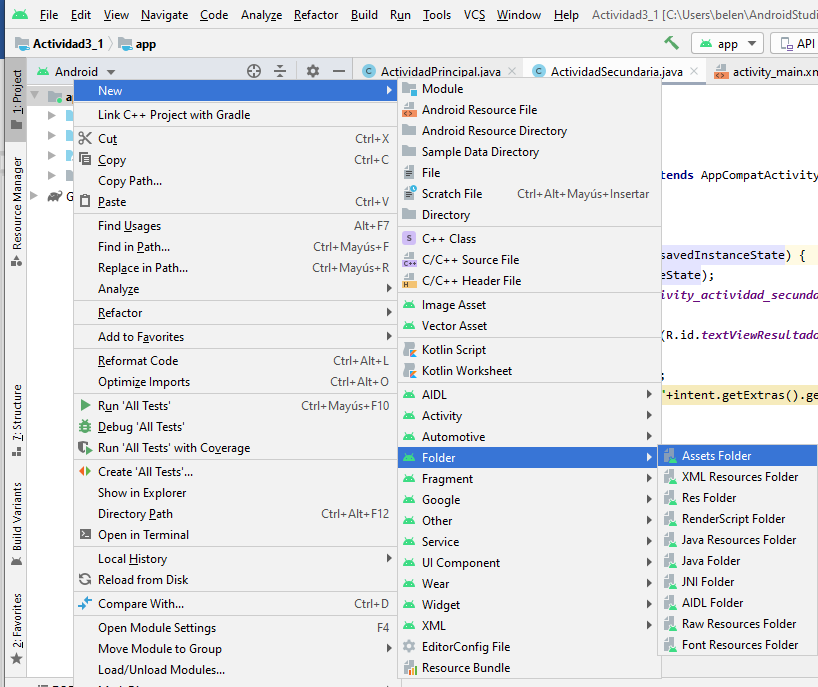


* Añadir algunas filas a la tabla.

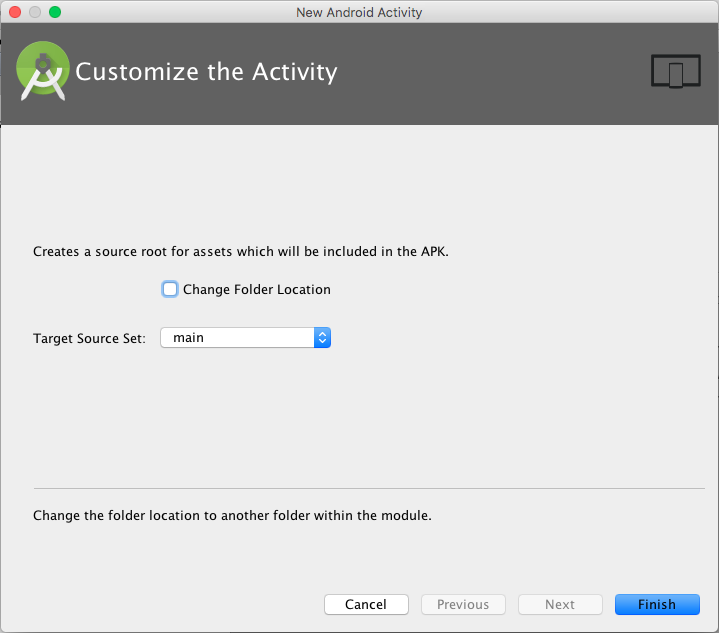


2.2. Añadir la base de datos a la app.

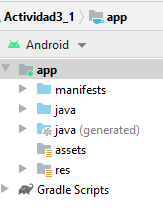
* Guardar el archivo de la base de datos en la carpeta assets del proyecto Android Studio.
  + En la carpeta assets se guardan recursos. A diferencia de los recursos de la carpeta res, para los assets no se generan IDs, no existen las restricciones de nombre de archivo y se pueden crear subcarpetas.
  + Creación de la carpeta assets en Android Studio.
    - Clic derecho en *app* tal como muestra la imagen:



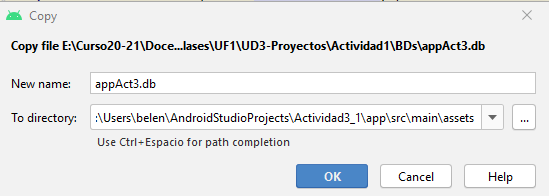
* + - No modificar la ventana del asistente y pulsar finish.



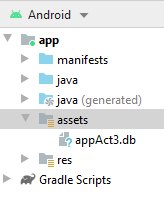
* + - Resultado.



* Copiar y pegar el archivo de la base de datos en la carpeta assets recién creada.



* Resultado.



# 3. Programa.

* Clase *Usuario: el órden de los parámetros del constructor debe coincidir con el orden de los campos de la tabla Usuario*

|  |
| --- |
| public class Usuario {  String nombre, login, password;  int id;  public Usuario(int id, String login, String password, String nombre) {  this.nombre = nombre;  this.login = login;  this.password = password;  this.id = id;  }  public String getLogin() {  return login;  }  public void setLogin(String login) {  this.login = login;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getNombre() {  return nombre;  }  public void setNombre(String nombre) {  this.nombre = nombre;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  } |

* Clase *APPACT3*
  + Representa la Base de Datos. Un objeto *SQLiteOpenHelper* ([enlace](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "getWritableDatabase())) permite crear, abrir y gestionar una base de datos ([enlace](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html" \l "SQLiteOpenHelper(android.content.Context, java.lang.String, android.database.sqlite.SQLiteDatabase.CursorFactory, int))). También permite gestionar versiones de la base de datos.

|  |
| --- |
| import android.content.Context;  import android.database.sqlite.SQLiteDatabase;  import android.database.sqlite.SQLiteOpenHelper;  public class APPACT3 extends SQLiteOpenHelper {  public final static String NOME\_BD = "appAct3.db";  public final static int VERSION\_BD = 1;  public APPACT3 (Context context) {  super(context, NOME\_BD, null, VERSION\_BD);  }  @Override  public void onCreate(SQLiteDatabase db) {  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  }  } |

* Interfaz *UsuarioDAO* (*DAO: Database Access Object*) tiene un único método getUsuario al que se le pasa un login y un password y devuelve un objeto de tipo Usuario.

|  |
| --- |
| public interface UsuarioDAO {  Usuario getUsuario(String login, String password);  } |

* Clase *UsuarioDAOSQLite*
  + La práctica habitual es escribir la consulta SQL directamente en la clase DAO. Por lo tanto, no se crearán recursos string en *strings.xml* conteniendo las consultas SQL del programa.

|  |
| --- |
| import android.content.Context;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.util.Log;  public class UsuarioDAOSQLite implements UsuarioDAO {  private APPACT3 appbd;  private Context context;  UsuarioDAOSQLite(Context context){  this.context=context;  this.appbd= new APPACT3(this.context);  }  public Usuario getUsuario(String login, String password) {  Usuario resultado = null;  SQLiteDatabase sqlLiteDB = appbd.getWritableDatabase();  String[] param = {login, password};  String consulta = "SELECT \* FROM Usuario WHERE login=? AND password=?";  Cursor cursor = sqlLiteDB.rawQuery(consulta, param);  this.depuracion(consulta, param);  Log.d("DEPURACIÓN", "Nº filas: " + cursor.getCount());  if (cursor.moveToFirst()) {  resultado = new Usuario(cursor.getString(0), cursor.getString(1), cursor.getString(2), cursor.getInt(3));  }  return resultado;  }  void depuracion(String consulta, String[] param) {  String texto = "Consulta: " + consulta + " Valores: ";  for (String p : param) {  texto += p + " ";  }  Log.d("DEPURACIÓN", texto);  }  } |

* Clase *ActividadPrincipal*,
  + Ver [Anexo 3](#_f31rthnwprph) sobre *.class*.

|  |
| --- |
| import android.content.Intent;  import android.os.Bundle;  import android.support.v4.app.FragmentManager;  import android.support.v7.app.AppCompatActivity;  import android.util.Log;  import android.view.View;  import android.widget.Button;  import android.widget.EditText;  import android.widget.Toast;  import java.io.File;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  public class MainActivity extends AppCompatActivity {  private APPACT3 appbd;  public final static String NOMBRE = "nombre";  public final static String ID = "id";  public final static String PASSWORD = “password”;  public final static String LOGIN = “login”;  UsuarioDAOSQLite usrDAO;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.Activity);  copiarBD();  this.usrDAO = new UsuarioDAOSQLite(this);  xestionarEventos();  }  private void copiarBD() {  String bddestino = "/data/data/" + getPackageName() + "/databases/"  + APPACT3.NOME\_BD;  File file = new File(bddestino);  Log.d("DEPURACIÓN", "Ruta archivo BD: " + bddestino);  if (file.exists()) {  Toast.makeText(getApplicationContext(), "La BD no se va a copiar. Ya existe", Toast.LENGTH\_LONG).show();  return; // Ya existe la BD, salimos del método  }  String pathbd = "/data/data/" + getPackageName()  + "/databases/";  File filepathdb = new File(pathbd);  filepathdb.mkdirs();  InputStream inputstream;  try {  inputstream = getAssets().open(APPACT3.NOME\_BD);  OutputStream outputstream = new FileOutputStream(bddestino);  int tamread;  byte[] buffer = new byte[2048];  while ((tamread = inputstream.read(buffer)) > 0) {  outputstream.write(buffer, 0, tamread);  }  inputstream.close();  outputstream.flush();  outputstream.close();  Toast.makeText(getApplicationContext(), "BASE DE DATOS COPIADA", Toast.LENGTH\_LONG).show();  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  private void xestionarEventos() {  Button btnLogin = (Button) findViewById(R.id.button);  btnLogin.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  // TODO Auto-generated method stub  iniciarSesion();  }  });  }  void iniciarSesion() {  String login = ((EditText) findViewById(R.id.editTextLogin)).getText().toString();  String password = ((EditText) findViewById(R.id.editTextPassword)).getText().toString();  Usuario usr =usrDAO.getUsuario(login, password);  if (usr != null) {  Log.d("DEPURACIÓN", "Nombre usr: "+ usr.getNombre());  Toast.makeText(getApplicationContext(), "Iniciando sesión.", Toast.LENGTH\_LONG).show();  Intent intent = new Intent(this, ActividadSecundaria.class);  intent.putExtra(this.NOMBRE, usr.getNombre());  intent.putExtra(this.PASSWORD, usr.getPassword());  intent.putExtra(this.LOGIN, usr.getLogin());  intent.putExtra(this.ID, usr.getId());  startActivity(intent);  finish();  } else {  //Toast.makeText(getApplicationContext(), "Error de autentificación.", Toast.LENGTH\_LONG).show();  MensajeDialogo d= new MensajeDialogo();  d.setCancelable(false);  FragmentManager fm= this.getSupportFragmentManager();  d.show(fm,"errorLogin");  }  }  } |

* Clase *ActividadSecundaria: abre otra pantalla mostrando en el título el nombre del usuario que introdujo su login y password y un informe con todos los campos del usuario.*

|  |
| --- |
| public class ActividadSecundaria extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_actividad\_secundaria);  Intent intent = getIntent();  String nuevoTitulo="Book flight: " + intent.getExtras().getString(ActividadPrincipal.NOMBRE);  setTitle(nuevoTitulo);  **et\_resultado** = findViewById(R.id.***textViewResultado***); **et\_resultado**.setText(**"ID: "**+intent.getExtras().getInt(ActividadPrincipal.***ID***)+**"\n"** +**"Nombre: "**+intent.getExtras().getString(ActividadPrincipal.***NOMBRE***)+**"\n"** +**"Login: "**+intent.getExtras().getString(ActividadPrincipal.***LOGIN***)+**"\n"** +**"Password: "**+intent.getExtras().getString(ActividadPrincipal.***PASSWORD***)+**"\n"**);  }  } |

* Clase *MensajeDialogo*.

|  |
| --- |
| import android.app.AlertDialog;  import android.app.Dialog;  import android.content.DialogInterface;  import android.content.DialogInterface.OnClickListener;  import android.os.Bundle;  import android.support.v4.app.DialogFragment;  import android.widget.Toast;  public class MensajeDialogo extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {  AlertDialog.Builder builder = new AlertDialog.Builder(getActivity())  .setTitle("No se pudo iniciar la sesión").setIcon(android.R.drawable.ic\_dialog\_alert)  .setMessage("El login o el password que has introducido son incorrectos. Compruébalos y vuelve a intentarlo de nuevo.")  .setPositiveButton("ACEPTAR", new OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  //Código asociado al botón Aceptar. Por ejemplo:  //Toast.makeText(getActivity(), "PULSADA OPCION BOA", Toast.LENGTH\_LONG).show();  }  });  return builder.create();  }  } |

* Para la depuración del programa seguir las indicaciones del [Anexo II](#_9uoanv6snux1).

# 4. Funcionamiento.

|  |  |
| --- | --- |
|  |  |

# 5. Anexos.

## Anexo 1: Ejecución de instrucciones SQL.

* Mediante la siguiente implementación de la clase *UsuarioDAOSQLite* se puede realizar inyección SQL.

|  |
| --- |
| public class UsuarioDAOSQLite implements UsuarioDAO {  private APPVUELOS appv;  private Context context;  UsuarioDAOSQLite(Context context){  this.context=context;  this.appv = new APPVUELOS(this.context);  }  public Usuario getUsuario(String login, String password){  Usuario resultado=null;  SQLiteDatabase sqlLiteDB = appv.getWritableDatabase();  String consulta= "SELECT \* FROM usuario WHERE login='"+login  +"' AND password='"+password+"'";  Log.d("DEPURACIÓN", "Consulta: "+ consulta);  Cursor cursor = sqlLiteDB.rawQuery(consulta, null);  Log.d("DEPURACIÓN", "Nº filas: "+ cursor.getCount());  if (cursor.moveToFirst()){  resultado= new Usuario(cursor.getString(0), cursor.getString(1), cursor.getString(2), cursor.getInt(3));  }  return resultado;  }  } |

* Ejecución introduciendo login y/o password erróneos.

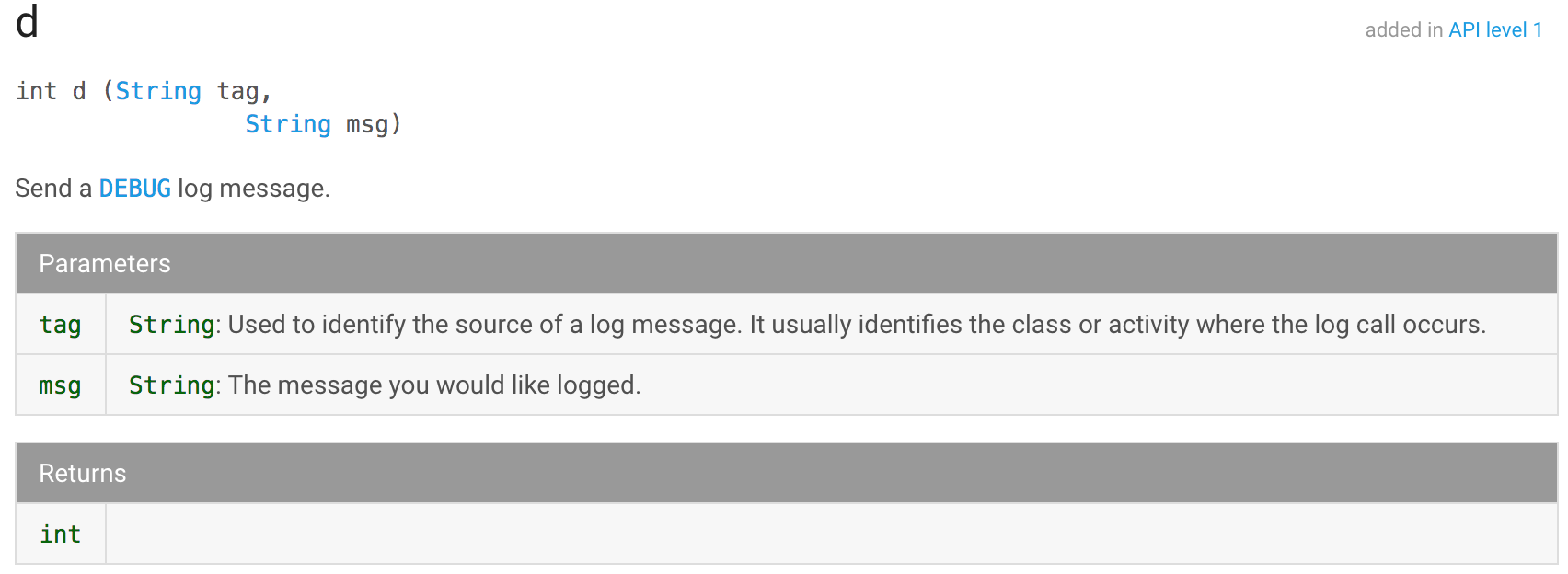
|  |  |
| --- | --- |
|  |  |

* Ejecución realizando inyección SQL.
  + En el campo login se introducirá la cadena *‘ OR 1=1 --*
  + Se introducirá cualquier cosa en el campo password.
  + Se consigue iniciar sesión sin haber introducido un login o password registrados en la BD.

|  |  |
| --- | --- |
|  |  |

## Anexo 2: Depuración del programa.

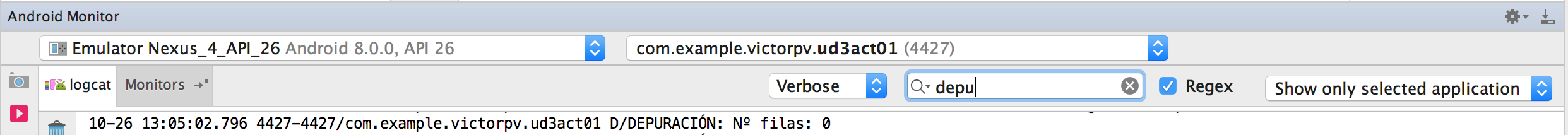
* Añadir mensajes de depuración en Logcat.
  + Se utiliza la clase Log ([enlace](https://developer.android.com/reference/android/util/Log.html)) para enviar mensajes a logcat.
  + En concreto, se usa el método *d*.



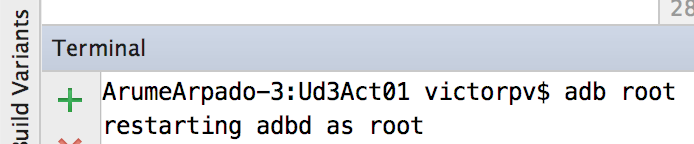
* + Ejemplo: mostramos el número de filas que devuelve una consulta a la BD.

|  |
| --- |
| Log.d("DEPURACIÓN", "Nº filas: " + cursor.getCount()); |

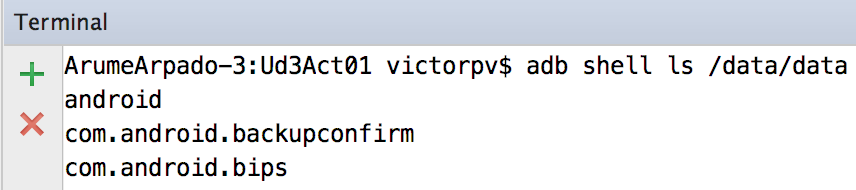
* + En la ventana Android Monitor, filtrar por *depu* (no es case sensitive)



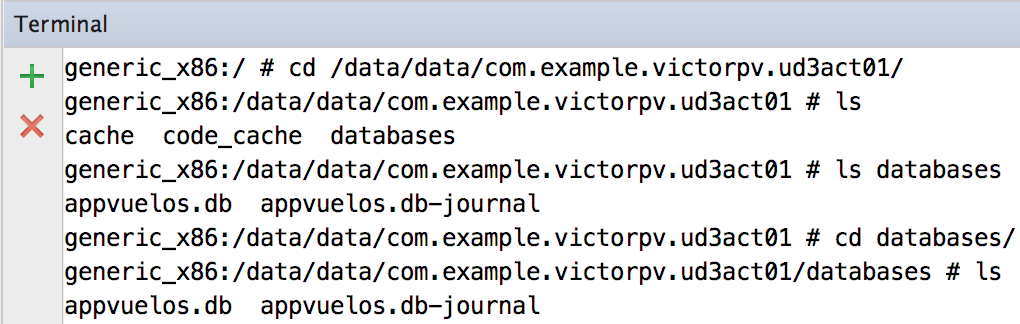
* Acceder al AVD en modo comando.
  + Se puede usar el terminal incluido en Android Studio.
  + A partir del API 24 es necesario ejecutar adb como root.



* + Esto permite acceder a todos los archivos del AVD.



* + Situarse en el directorio de la base de datos de la aplicación.

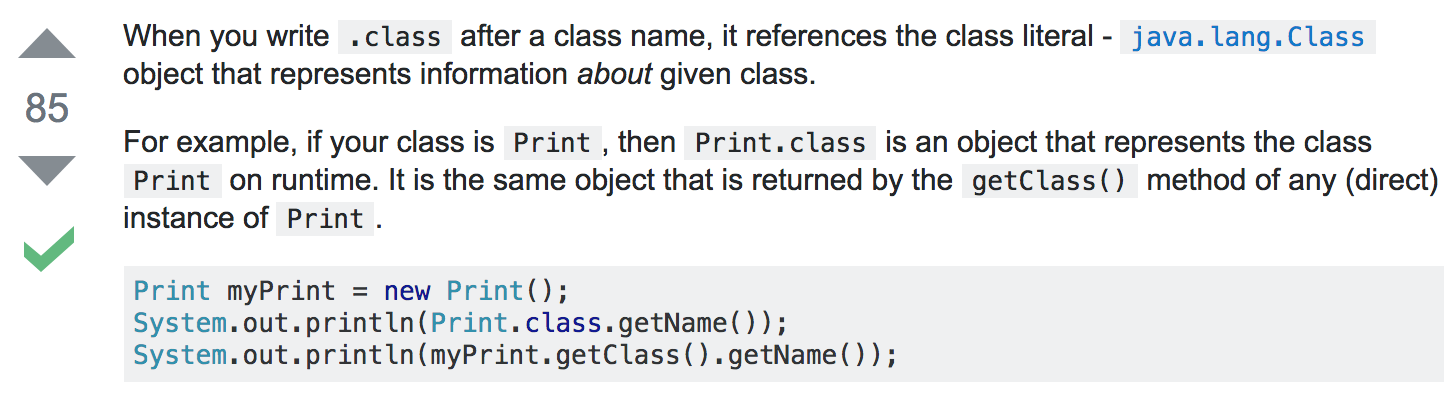


* + Realizar consultas sobre las tablas.



## Anexo 3: Class literal *.class*.

* Un class literal (literal de clase) es un expresión que consiste en un nombre de clase (o también interfaz, tipo primitivo o el pseudo-tipo void) seguido de punto y del token *class* ([enlace](https://docs.oracle.com/javase/specs/jls/se9/html/jls-15.html" \l "jls-15.8.2)). Esta expresión devuelve el nombre de la clase.



* Fuente: [enlace](https://stackoverflow.com/questions/15078935/what-does-class-mean-in-java).