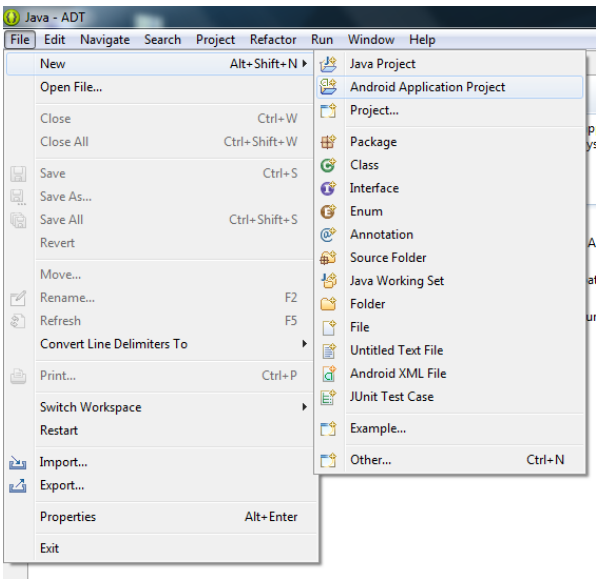
Desarrollo paso a paso de una aplicación Android con Eclipse

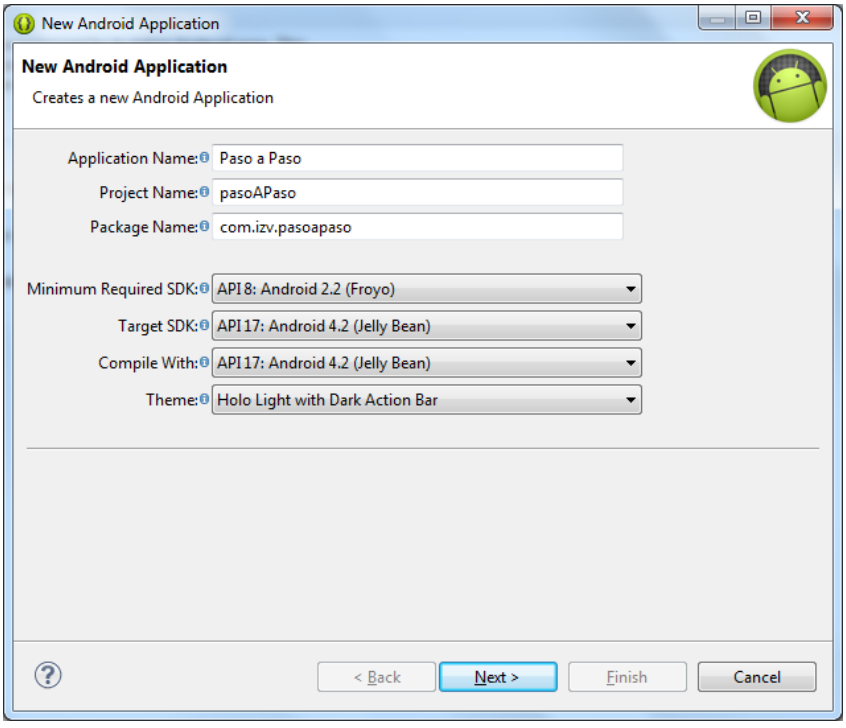
http://developer.android.com/training/basics/firstapp/creating-project.html

Para crear un proyecto Android en Eclipse debemos seguir los siguientes pasos.

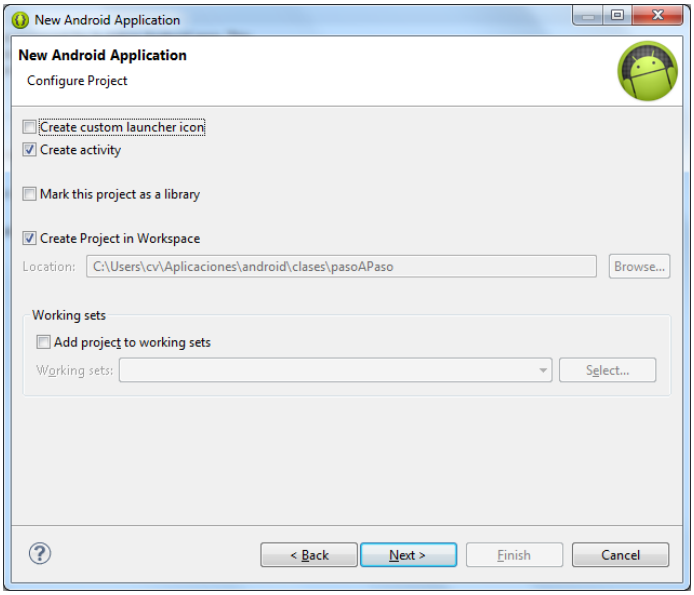
File > New > Android Application Project



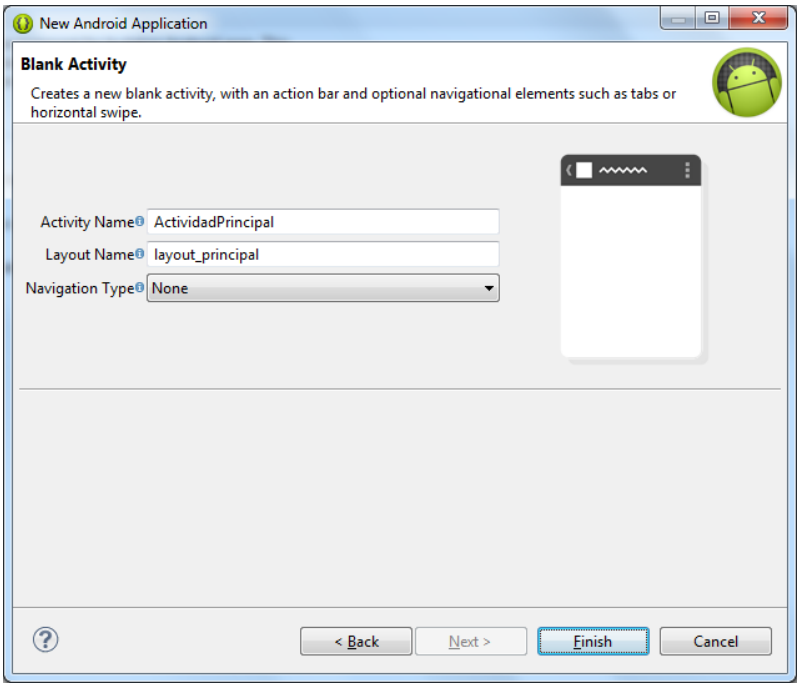
Introducimos el nombre de la aplicación, que es el nombre con el que se identifica en el dispositivo Android y en Google Market, el nombre del proyecto, que es el nombre con el que identificamos esta aplicación en Eclipse, y el nombre del paquete, que debe ser un nombre de paquete único.



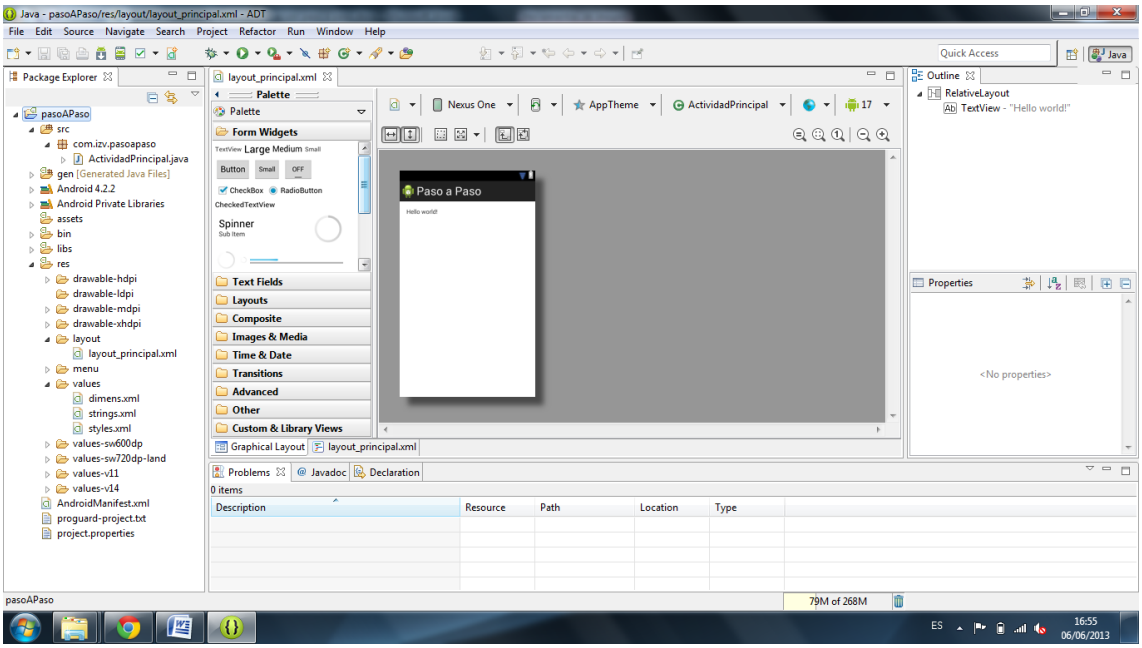
En la siguiente ventana del asistente de creación, quitamos la marca de **Create custum launcher icon**, para no tener que componer el icono de la aplicación.

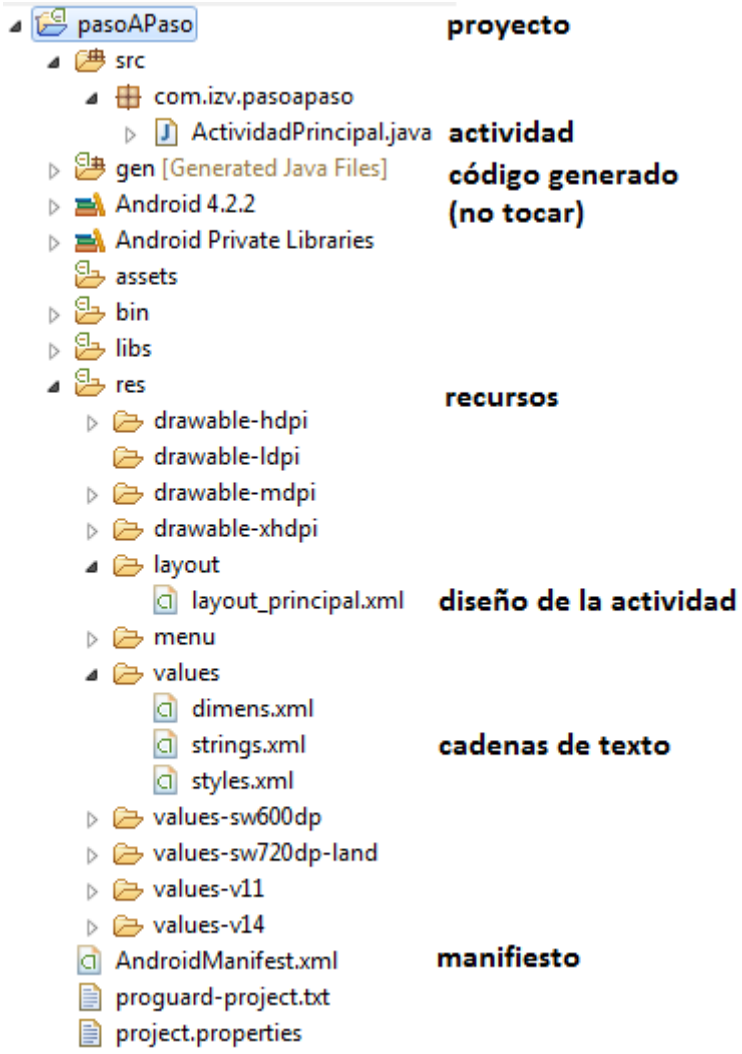


En la siguiente ventana marcamos **Create Activity** y seleccionamos **Blank Activity**. En la última ventana introducimos el nombre de la actividad, que es el nombre de la clase en Java y el nombre de su diseño, que es el nombre del archivo XML en el que se diseña la pantalla asociada a la actividad.



Al pulsar sobre **Finish** se crea el proyecto con la siguiente estructura:





La estructura de todas las aplicaciones Android es similar a esta, los archivos que vamos a tocar inicialmente son:

- ActividadPrincipal.java //actividad, activity
- layout_principal.xml //diseño, layout
- strings.xml //cadenas de texto
- AndroidManifest.xml //manifiesto

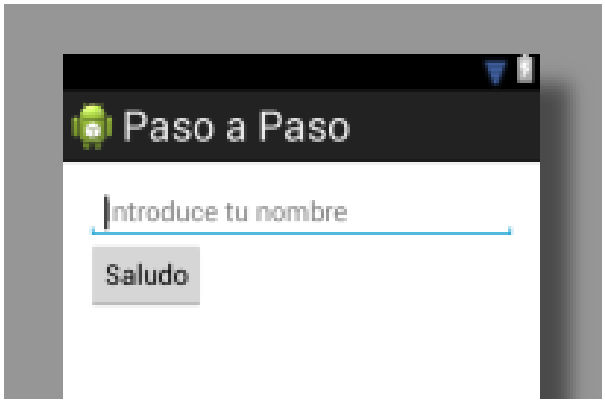
Para ejecutar la aplicación pulsamos Ctrl-F11 e indicamos que es una aplicación Android.

Para modificar el diseño de la actividad se puede usar el asistente o editar directamente el contenido del archivo **layout_principal.xml**.

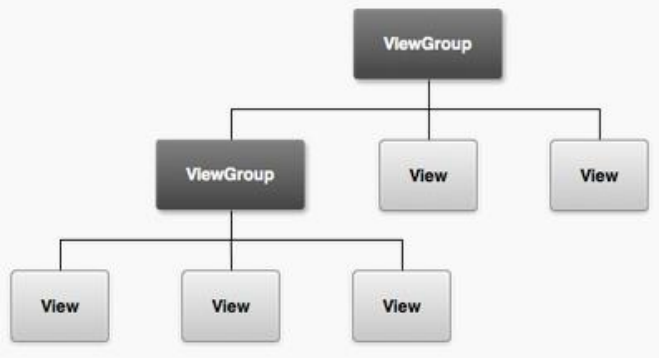
Para crear y editar las cadenas de texto necesarias se abre el archivo **strings.xml**, que se podrá editar con el asistente o directamente editando el código XML.

Para modificar el comportamiento de la actividad editaremos directamente el archivo **ActividadPrincipal.java**.

Ejercicio: Modifica los archivos strings.xml y principal.xml hasta dejarlos del siguiente modo.



La interfaz de usuario se construye mediante dos tipos de controles, los de tipo ViewGroup y los de tipo View. Los grupos de vistas son invisibles y sirven para agrupar otros controles. Las vistas son los controles básicos.



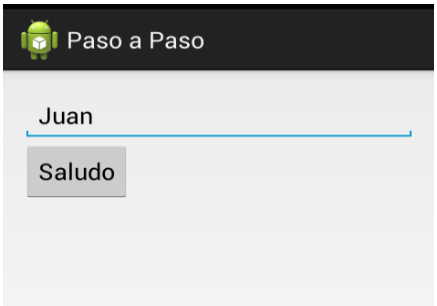
En este caso usamos un ViewGroup: LinearLayout y tres Views: EditText, Button y TextView (este inicialmente no es visible puesto que contiene la cadena vacía).

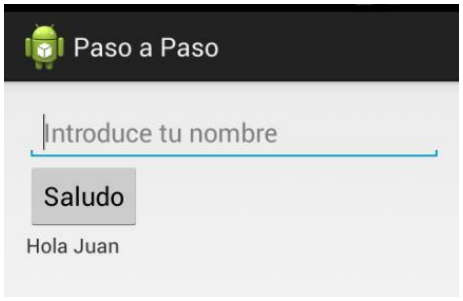
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".ActividadPrincipal" >
    <EditText
        android:id="@+id/etNombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/eth_saludo" >
        <requestFocus />
    </EditText>
    <Button
        android:id="@+id/btSaludo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/bt_saludo" />
    <TextView
        android:id="@+id/tvSaludo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />
</LinearLayout>
```

La interfaz se compone sin utilizar cadenas de caracteres, cada uno de los mensajes que se vaya a mostrar se deben definir en el archivo strings.xml. Para hacer referencia desde el layout a las cadenas de caracteres se utiliza **@string/nombreCadena**. Para poner identificadores a los elementos del layout se utiliza **@+id/identificador**.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Paso a Paso</string>
    <string name="bt_saludo">Saludo</string>
    <string name="eth_saludo">Introduce tu nombre</string>
    <string name="tv_saludo">Hola</string>
</resources>
```

A continuación modificaremos el archivo ActividadPrincipal.java para que al pulsar el botón muestre un mensaje de saludo.





```
public class ActividadPrincipal extends Activity {
    private Button bt;
    private EditText et;
    private TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
        bt = (Button) findViewById(R.id.btSaludo);
        et = (EditText) findViewById(R.id.etNombre);
        tv = (TextView) findViewById(R.id.tvSaludo);
        OyenteBoton ob = new OyenteBoton();
        bt.setOnClickListener(ob);
    }
    private class OyenteBoton implements OnClickListener{
        @Override
        public void onClick(View v) {
            String saludo = getResources().getString(R.string.tv_saludo);
            tv.setText(saludo+" "+et.getText().toString());
            et.setText("");
        }
    }
}
```

Al crearse la actividad es llamado el método onCreate(), en él realizaremos todas las operaciones de inicialización necesarias. La primera instrucción será necesariamente la llamada a super.onCreate(). En la llamada al método setContentView() establecemos la vista o el layout que se debe dibujar. A continuación asignamos a las tres variables de instancia los tres objetos del layout que vamos a referenciar, el botón, el cuadro de texto y la etiqueta. Por último, lo que hacemos es crear el objeto oyente que se encarga de procesar los eventos de tipo click que se vayan a producir sobre el botón.

Todo esto se suele escribir de forma más abreviada en Java:

```
public class ActividadPrincipal extends Activity {
    private Button bt;
    private EditText et;
    private TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
        bt = (Button) findViewById(R.id.btSaludo);
        et = (EditText) findViewById(R.id.etNombre);
        tv = (TextView) findViewById(R.id.tvSaludo);
        bt.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
               String saludo = getResources().getString(R.string.tv_saludo);
               tv.setText(saludo+" "+et.getText().toString());
               et.setText("");
            }
        });
    }
}
```

Observaciones:

- Los identificadores en los archivos XML se indican con @+id/identificador.
- Para referenciar una cadena de texto desde el archivo layout de XML se usa @string/cadena.
- Para referenciar a los objetos XML desde Java se usa el método findViewById().
- Para obtener las cadenas de texto se usa el método getResources().getString().
- Para obtener el contenido de los cuadros de texto se usa getText().toString().
- La forma de referenciar los elementos creados en XML desde Java es a través de la clase R, los layouts se referencian R.layout.nombrearchivo, los elementos del layout se referencian R.id.identificadorelemento y las cadenas de caracteres se referencian R.string.nombrecadena.

Existe la posibilidad de programar el evento onclick de los botones de otro modo. Esta forma puede resultar más cómoda, pero hemos de considerar que no siempre va a funcionar –dependerá de cómo se construya el layout–, por lo que es conveniente conocer ambas formas de hacerlo.

En la etiqueta del botón definiremos el atributo **android:onClick**.

```
<Button
    android:id="@+id/btSaludo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="saludar"
    android:text="@string/bt_saludo" />
```

En la actividad implementaremos un método que debe ser de tipo public void con el mismo nombre que el valor asignado al atributo android:onClick y que recibe un parámetro de tipo View, que hace referencia al control sobre el que se ha producido el evento.

```
public class ActividadPrincipal extends Activity {
private EditText et;
private TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
        et = (EditText) findViewById(R.id.etNombre);
        tv = (TextView) findViewById(R.id.tvSaludo);
    }
    public void saludar(View v){
        String saludo = getResources().getString(R.string.tv_saludo);
        tv.setText(saludo+" "+et.getText().toString());
        et.setText("");
    }
}
```

Consideracionesfinales:

Para definir qué actividad es la que se debe ejecutar al lanzar la aplicación se debe indicar en el manifiesto. La actividad principal debe tener un filtro con la acción MAIN y la categoría LAUNCHER.

```
<application …>
    <activity android:name="com.izv.pasoapaso.ActividadPrincipal" …>
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
</application>
```