Ficheiros e cartafoles dun proxecto Android: Ola Mundo. Activities, Layouts e Múltiples pantallas

Sumario

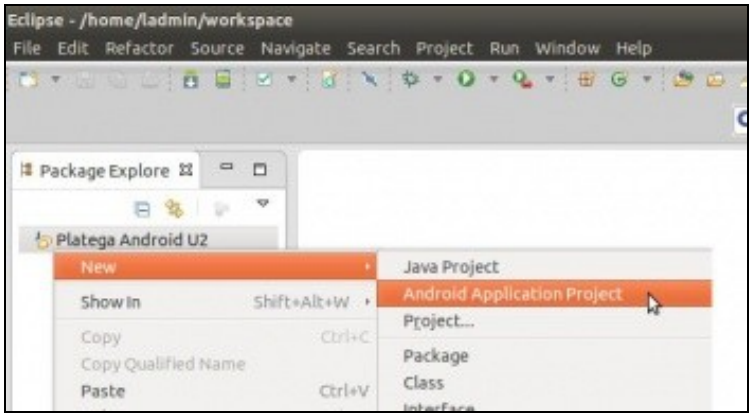
- 1 Introdución
- 2 Crear un proxecto Android: OlaMundo
- 3 Activities
- 4 Layout
- 5 Soporte múltiples pantallas
    - 5.1 Tamaños de pantalla e densidade (puntos por pulgada: dpi)
    - 5.2 Orientación e resolución
    - 5.3 Píxeles independentes da densidade (dp)
- 6 Ficheiros do proxecto Android
    - 6.1 /src
    - 6.2 /res
    - 6.3 /gen
    - 6.4 /assets
    - 6.5 /bin
    - 6.6 Ficheiro AndroidManifest.xml

Introdución

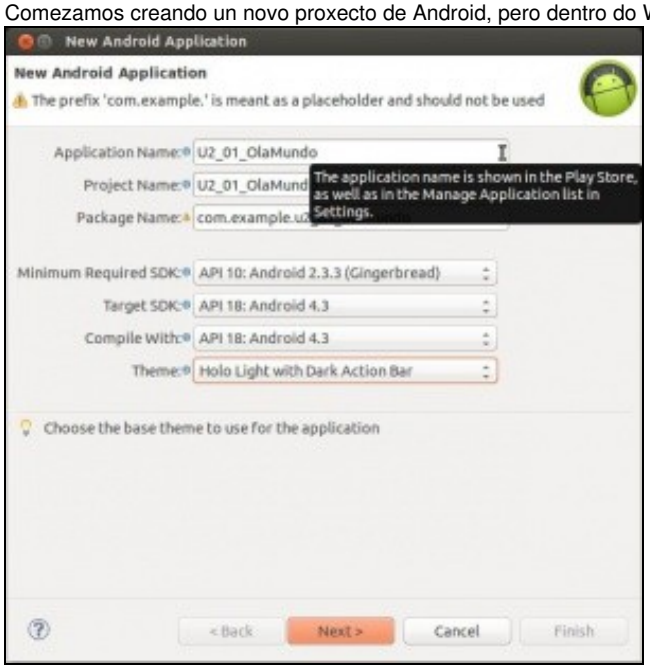
- Neste apartado vanse describir os cartafoles e ficheiros máis esenciais dos que se compón un proxecto Android.
- Comezaremos creando un proxecto: **OlaMundo**, e estudaremos as partes nas que se divide.

- Os proxectos vanse nomear e numerar da seguinte forma: **U2_XX_NomeDaAplicación**, para facilitar o seguimento. Sendo **U2** a unidade 2 e **XX** o número de proxecto dentro da Unidade.
- Estes van estar no seguinte cartafol **<ruta até o cartafol>/ProxectosEclipse/PlategaAndroid/Unidade2/<proxectos>**
    - Esa ruta pode ser a ruta ao Workspace, ou
    - Pode ser unha ruta fóra do Workspace.

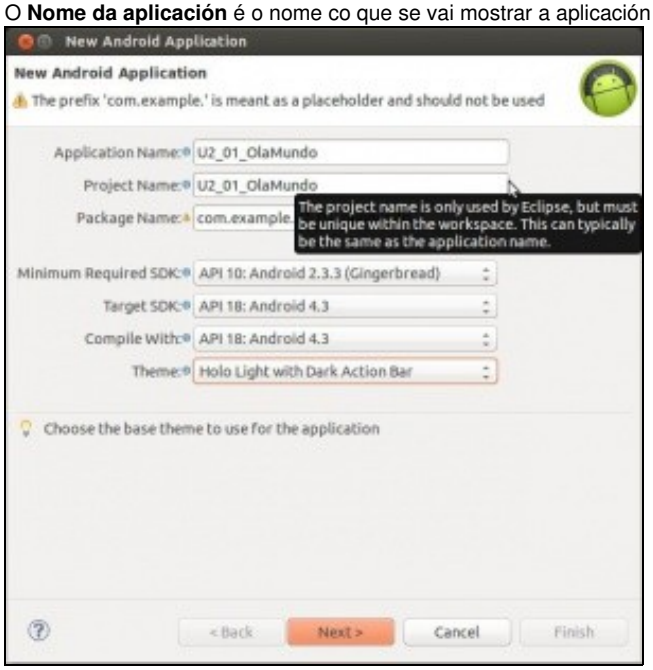
Crear un proxecto Android: OlaMundo

- Ola Mundo
- 

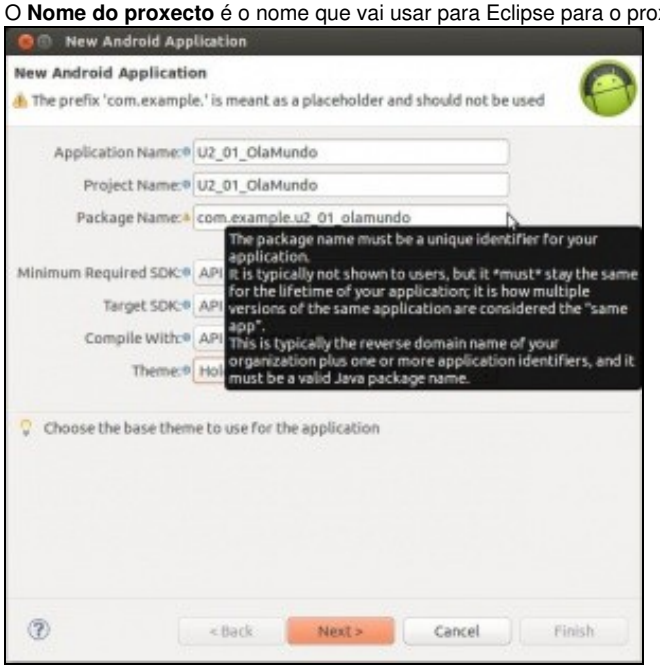
Comezamos creando un novo proxecto de Android, pero dentro do Working Set creado no punto anterior.

- 

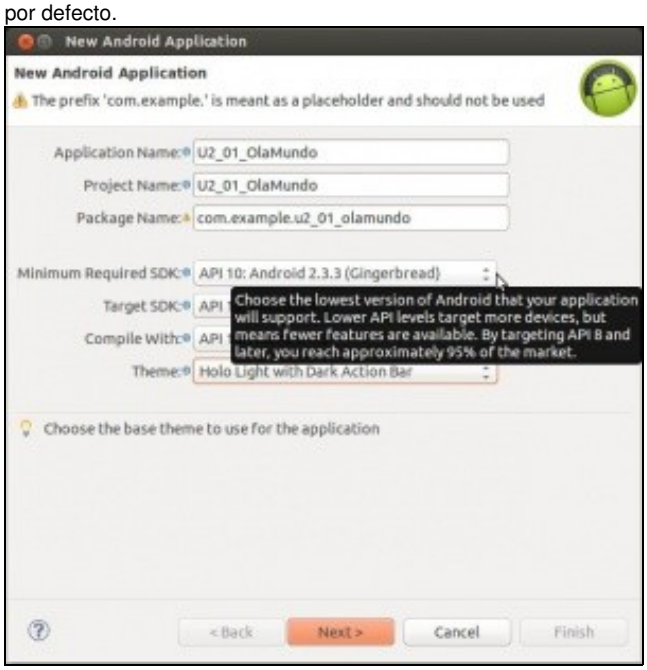
O **Nome da aplicación** é o nome co que se vai mostrar a aplicación no dispositivo: **U2_01_OlaMundo**.

- 

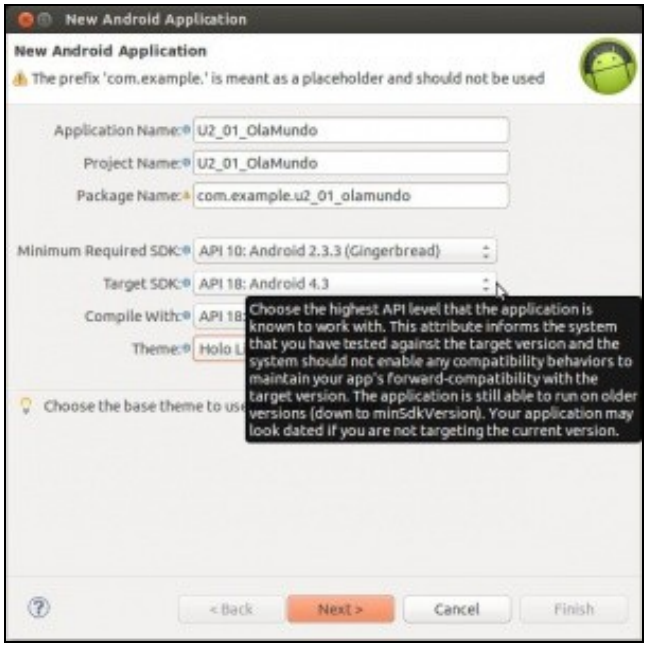
O **Nome do proxecto** é o nome que vai usar para Eclipse para o proxecto.

- 

O **Nome do paquete** debe ser único para todo aplicación. Constrúese co nome do dominio ao revés, e o identificador da aplicación. Por exemplo se aplicación fose da Consellería de Educación o nome sería: es.xunta.edu.nome_aplicación. Neste caso deixouse o nome que xera por defecto.

- 

Seleccionar a **Versión de Android máis baixa** na cal vai correr a aplicación.
OLLO edición 2015: sempre imos seleccionar a API 16. Como indica a lenda, canto máis baixa máis dispositivos soportados pero menos novas funcionalidades. E canto máis alta viceversa.

- 

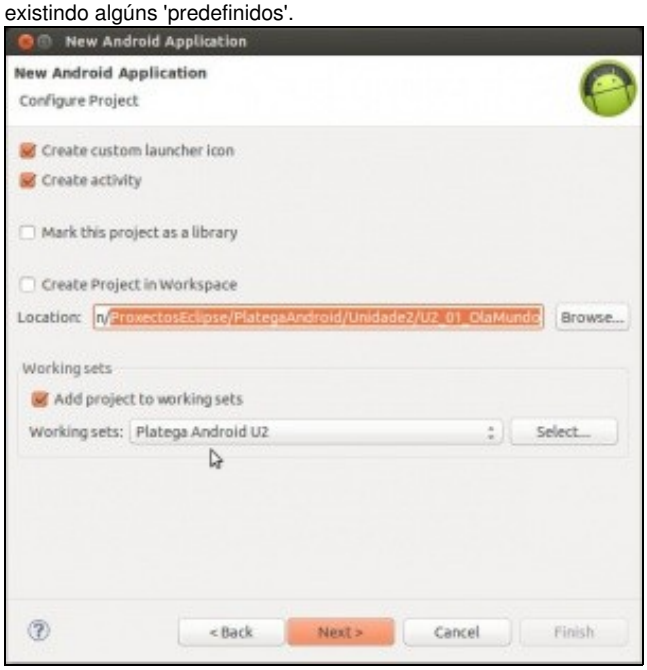
Indicar a **versión de Android máis alta** na que poderá funciona esta apliación. Isto indica ao dispositivo que esta aplicación foi testeada en dispositivos con esa versión de Android.
Edición 2015: Sempre imos seleccionar a API 21.

- 

Seleccionar con que **versión do SDK se vai compilar** a aplicación.
Edición 2015: Sempre imos seleccionar a API 21.

- 

Seleccionar o **tema** base para aplicación. Xa o veremos máis adiante pero aquí escollemos o 'aspecto' que vai ter a nosa aplicación, existindo algúns 'predefinidos'.

- 

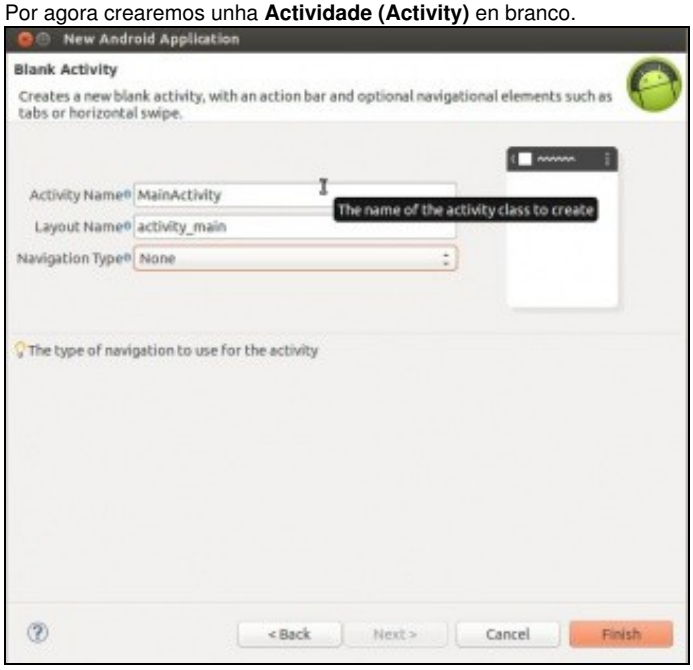
Ruta na que gravar o proxecto, nesta imaxe gárdase fóra do workspace, no cartafol creado antes. Se se desexa pódese facer que o Workspace apunte a esa mesma carpeta da Unidade 2. Observar o Working Set onde se vai crear o proxecto.

- 

Crear a icona do lanzador (Imaxe, Clipart, Texto), escala, cor, etc. Neste caso vaise deixar a imaxe que trae por defecto

- 

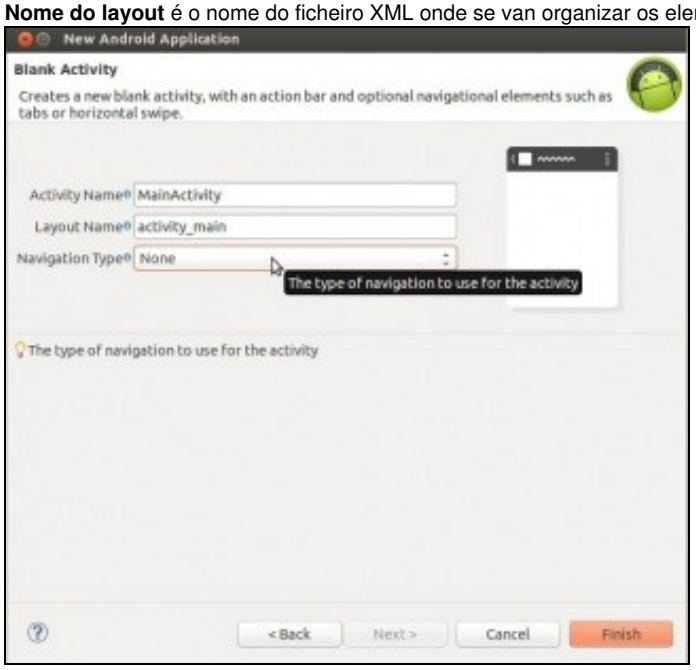
Por agora crearemos unha **Actividade (Activity)** en branco.

- 

Nome da actividade é o nome da clase Java (.java).

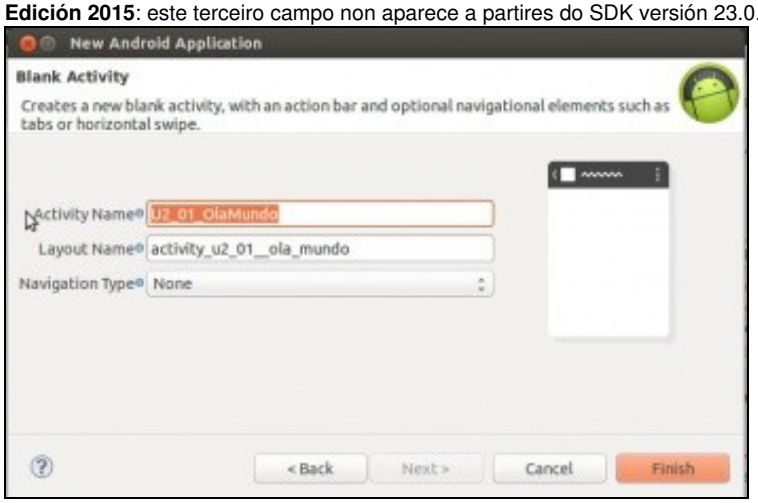
- 

Nome do layout é o nome do ficheiro XML onde se van organizar os elementos que se van presentar na pantalla.

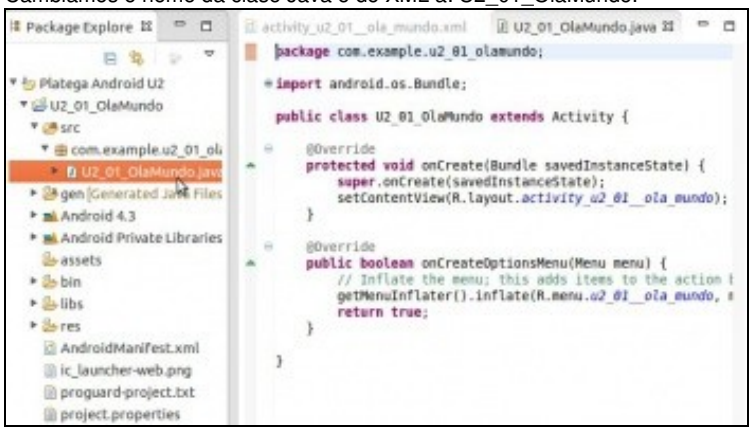
- 

Deixamos o tipo de navegación por defecto.

Edición 2015: este terceiro campo non aparece a partires do SDK versión 23.0.

- 

Cambiamos o nome da clase Java e do XML a: U2_01_OlaMundo.

- 

Observar como o nome da clase Java garda relación co indicado na imaxe anterior.

- 

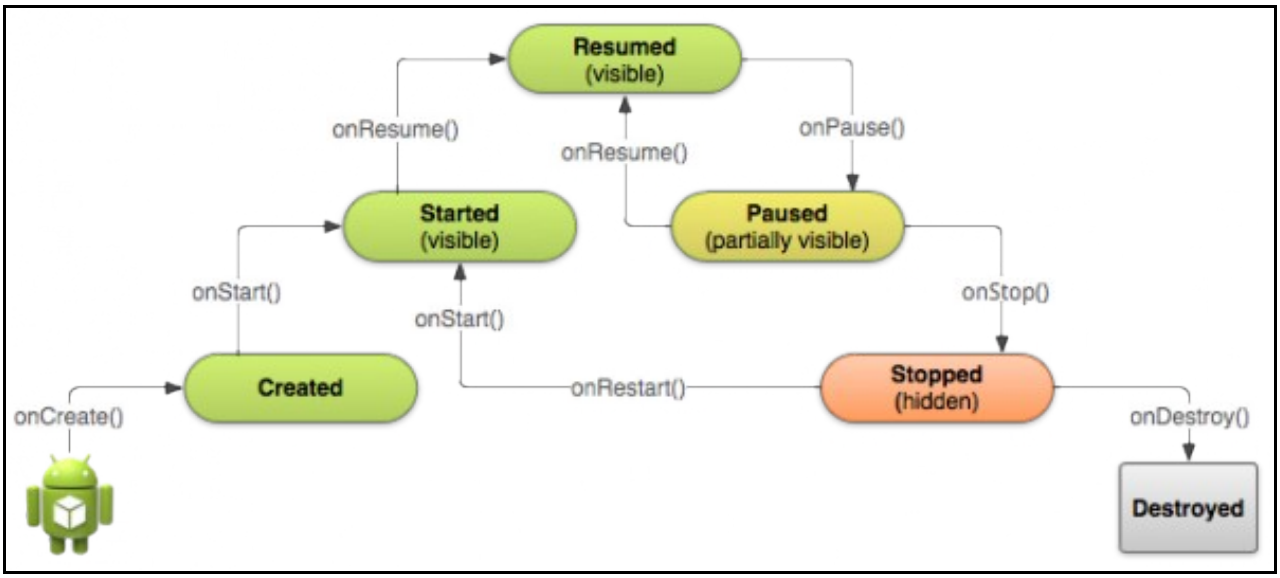
O mesmo acontece co ficheiro XML do layout.

- 

Ficheiro XML que define o Layout e os elementos que o compoñen. Por agora ten unha etiqueta.

Activities

- Imos dar unha breve descrición do que é un **Activity**. Estas veranse máis adiante.
- Unha **Activity** é unha pantalla única coa cal o usuario poderá interactuar a través dos elementos que nela se dispoñan.
- Unha **aplicación** estará composta dunha ou varias Activities.
- Incluso se pode saltar da Activity dunha aplicación á unha Activity calquera de outra aplicación. Por exemplo, cando se usa WhatsApp e se accede a consultar os datos dun contacto.
- Unha Activity pode ser chamada cando se inicia unha aplicación, dende outra aplicación, ou cando se recupera a aplicación.
- Unha Activity pode ser destruída pola propia aplicación, cando se preme o botón **Back** do dispositivo ou polo sistema porque está na pila de aplicacións en segundo plano e se necesitan os recursos que está consumindo.
- As Activities teñen un ciclo de vida como se expón dun xeito simple na seguinte imaxe:



Nun punto posterior afondarase sobre as Activities.

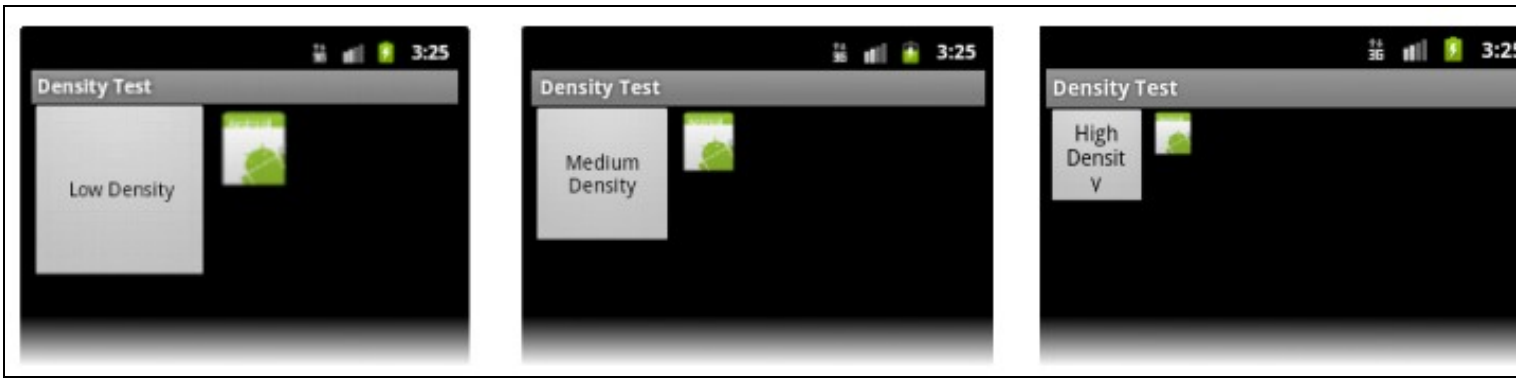
- Referencias:
  - http://developer.android.com/reference/android/app/Activity.html
  - http://developer.android.com/guide/components/activities.html
  - http://developer.android.com/training/basics/activity-lifecycle/index.html

Layout

- Un **layout** é un panel onde se definen os elementos gráficos que compoñen unha Activity: botóns, listas depregables, etiquetas, etc.
- Un Layout pode declararse en ficheiros XML ou en tempo de execución. Comezaremos polos ficheiros XML.
- Un Layout poden ser de varios tipos: lineais, táboa, grid, etc. Como se verá no seguinte apartado.
- Referencias:
  - http://developer.android.com/guide/topics/ui/declaring-layout.html

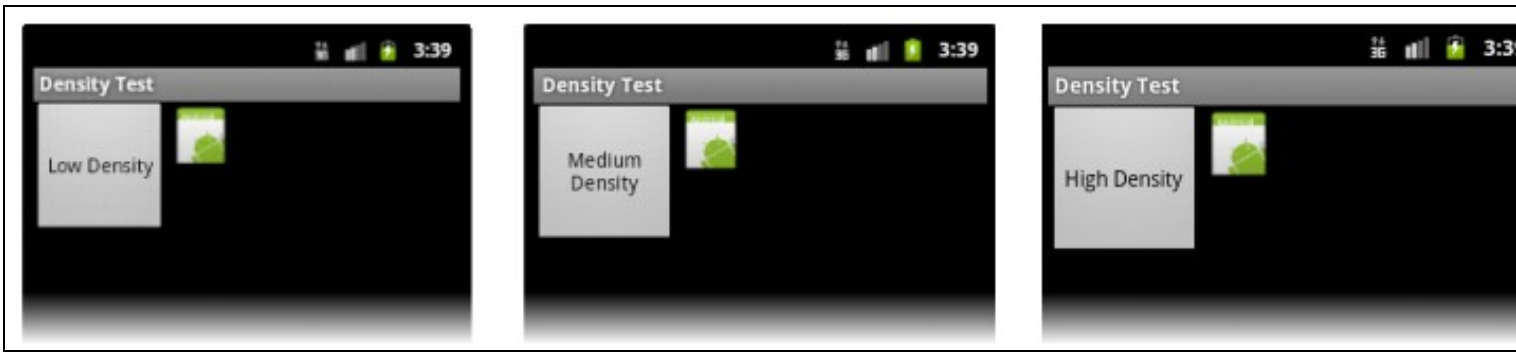
Soporte múltiples pantallas

- Como sabemos no mercado hai distintos dispositivos: teléfonos, *tablets*, etc. e agora ... coa nova versión **Lollipop**: *reloxos*, *pulseiras*, *TVs*, etc.
- Cada un ten os seus tamaños e resolucións.
- Android proporciona ferramentas para adaptarse a toda esa variedade. Por exemplo, poderíamos deseñar unha aplicación cuxa Interface Gráfica fose distinta dun teléfono a unha *tablet*, o se está en apaisado o teléfono ou en vertical.
- O sistema tamén trata de escalar a aplicación para adaptala aos distintos tipos (Tamaños/resolucións) de pantalla, pero se usamos **pixels** para definición dos elementos que compoñen unha pantalla pode pasar o seguinte:



- Os dispositivos teñen o mesmo tamaño de pantalla,
- A resolución dos dispositivos vai de menor a maior (de esquerda á dereita).
- Observar como ao definir que un obxecto ocupa X*Y **píxeles** estes obxectos son máis pequenos a medida que a resolución da pantalla é maior.
- Observar que se ven afectados os tipos de recursos gráficos:
    - Os creados polo usuario na aplicación: por exemplo o botón.
    - Os engadidos polo usuario á aplicación (debuxos/imaxes): por exemplo a imaxe de Android.

- O obxectivo sería, que independentemente da resolución da pantalla se obtivera o seguinte resultado: os obxectos vense do mesmo tamaño aínda tendo resolucións distintas.



- Imos ver antes de nada unha serie de conceptos:

Tamaños de pantalla e densidade (puntos por pulgada: dpi)

- **Tamaño da pantalla**: tamaño da diagonal da pantalla física do dispositivo.
    - Por simplicidade Android divide as pantallas, até a versión Android 3.2 en : **pequenas (small)**, **normales**, **grandes (large)** e **extra largas (extra large).**
- **Densidade de pantalla (Screen density)**: puntos por pulgada (**ppp** / **(dot per inch (dpi))**). Unha pantalla de baixa resolución terá poucos dpis comparada cunha pantalla normal ou de alta resolución.
    - Por simplicidade Android agrupaba as densidades :
    - **ldpi:** baixa (low, 120 dpi)
    - **mdpi:** media (medium, 160 dpi)
    - **hdpi:** alta (high, 240 dpi)
    - **xhdpi:** extra alta (extra high, 320 dpi)
    - **xxhdpi:** extra extra alta (extra-extra high, 480 dpi)
    - **xxxhdpi:** extra extra extra alta (extra-extra-extra high, 640 dpi)

Orientación e resolución

- **Orientación**: dende o punto de vista do usuario indica se o dispositivo está en apaisado ou en vertical. Ao rotar a pantalla cambia a relación de altura e ancho.
- **Resolución**: indica o número total de píxeles que ten a pantalla. Cando se desenvolven aplicacións para distintas pantallas débese traballar coa densidade e co tamaño da pantalla e non coa resolución total.

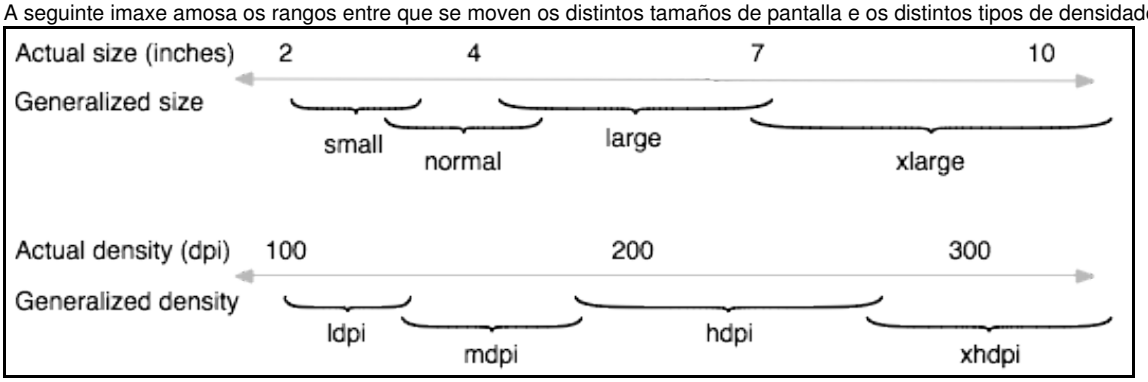
Píxeles independentes da densidade (dp)

- **Píxel independente da densidade (Density-independent pixel (dp))**: é un píxel virtual que debe ser usado cando se definen elementos gráficos nun layout, tanto para expresar tamaño dun obxecto, como a súa posición.
    - Unha pantalla media(medium) ten 160 dpi.
    - Nunha pantalla media 1dp=1dpi. Isto é nunha pantalla media o píxel virtual coincide co píxel real.
    - En tempo de execución é cando se fai a conversión de unidades dp a píxeles físicos (px).

- ♦ O factor de conversión é: **px= dp (dpi / 160)**.
- ♦ Por exemplo 1 dp nunha pantalla de densidade:
- ♦ 120dpi: sería igual a 0,75 px (píxeles físicos).
    - ◊ 160dpi: sería igual a 1 px (píxeles físicos).
    - ◊ 240dpi: sería igual a 1,5 px (píxeles físicos).
    - ◊ 320dpi: sería igual a 2 px (píxeles físicos).
    - ◊ 480dpi: sería igual a 3 px (píxeles físicos).
    - ◊ 640dpi: sería igual a 4 px (píxeles físicos).
- ♦ Deste xeito se defininos sempre as dimensións e posicións en dp sempre se vai realizar unha conversión en tempo de execución ao número de píxeles físicos equivalentes.

• **Scale-independent Pixels (sp)**: esta unidade é igual a dp, pero é usada para tamaños de texto. Estes tamaños poden se axustados pola densidade da pantalla e polas preferencias do usuario.

A seguinte imaxe amosa os rangos entre que se moven os distintos tamaños de pantalla e os distintos tipos de densidade:

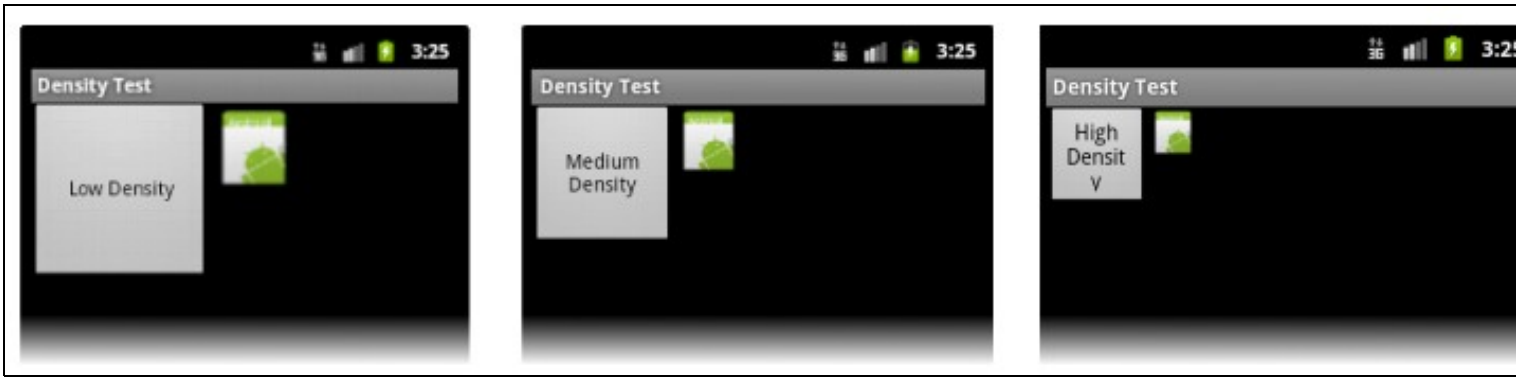


• **AS IMAXES**

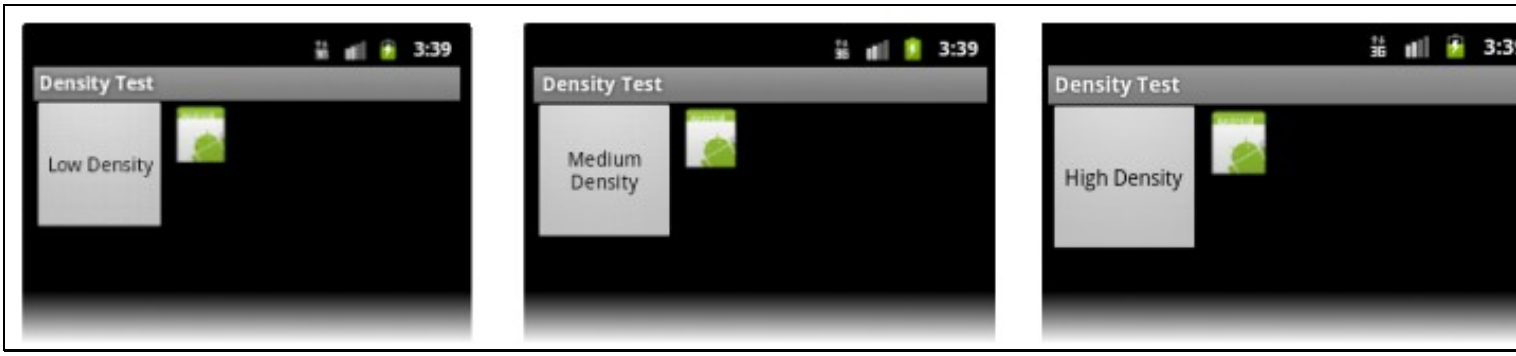
• Pero aínda nos falta saber que facer coas **imaxes que pode incorporar un usuario a unha aplicación**.

- ♦ Pódese prover para cada imaxe un ficheiro distinto adaptado ás distintas densidades.
- ♦ Se non se provén ficheiros para as distintas densidades o sistema escalará o ficheiro proporcionado para a densidade media (mdpi) e adaptarao a densidade da pantalla. Pero sempre será mellor proporcionar distintas versións da imaxe.
- ♦ Para proporcionar distintas versións de densidade dunha imaxe debemos seguir a regra: 3(ldpi):4(mdpi):6x(hdpi):8(xhdpi). Isto é, se collemos como referencia unha imaxe densidade media (mdpi) de tamaño X*Y píxeles, debemos crear novas imaxes que garden a seguinte relación en tamaño (número de píxeles): ldpi(x0.75), mdpi(x1), hdpi(x1,5) e xhdpi(x2).
- ♦ Exemplo se temos unha imaxe de 48x48 píxeles (tamaño dunha icona de lanzamento dunha aplicación) e esta é para unha densidade mdpi, debemos crear versións cos seguintes tamaños en píxeles:
    - ◊ ldpi: 36x36
    - ◊ mdpi: 48x48
    - ◊ hdpi: 72x72
    - ◊ xhdpi: 96x96

• Visto todo o anterior agora volvamos a ver de novo as imaxes onde se usan os píxeles para os tamaños e as posicións:



- E a que usa **dp** e distintas versións dunha mesma imaxe:



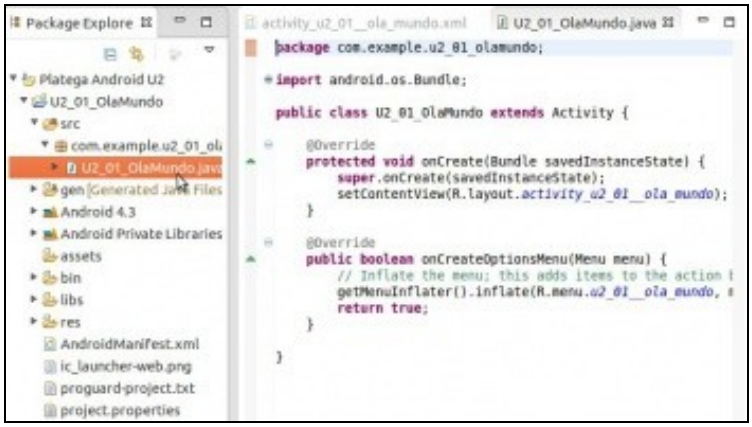
- Observar como coinciden para os dous casos na densidade media.

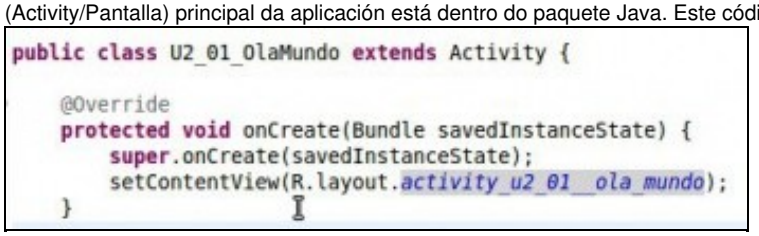
- Referencias:
  - http://developer.android.com/guide/practices/screens_support.html
  - http://developer.android.com/design/style/iconography.html
  - Para as dimensións: http://developer.android.com/guide/topics/resources/more-resources.html#Dimension

Ficheiros do proxecto Android

A continuación imos ver os contidos dos distintos cartafoles que compoñen un proxecto.

/src

- /src
- 

Cartafol src: contén o código fonte da aplicación, do interfaz gráfico, clases auxiliares. O código fonte asociado á actividade (Activity/Pantalla) principal da aplicación está dentro do paquete Java. Este código lanza a Pantalla Principal (Activity) e o Menú.
- 
```
public class U2_01_OlaMundo extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u2_01__ola_mundo);
    }
```
A clase **U2_01_OlaMundo** herda da clase **Activity** os seus atributos e métodos.
Neste caso sobrescríbese o método **onCreate**, que é o primeiro que se executa cando se lanza unha Activity (ver imaxe superior dos estados dunha Activity). Este método recibe un parámetro de tipo **Bundle**, no que recibe o estado da activity:
(-Null: porque acabamos de iniciala
-ou valores do estado anterior se vén de estar parada. Mirar imaxe superior do ciclo de vida).
Chama ao método **onCreate** da clase pai.
A continuación mostra en pantalla o contido do layout: **activity_u2_01__ola_mundo**. Para iso fai uso da clase R, como veremos a continuación.
http://developer.android.com/reference/android/os/Bundle.html
http://developer.android.com/reference/android/app/Activity.html#setContentView%28int%29

/res

Cartafol res (Resources/Recursos): almacena todos os ficheiros de recursos necesarios para a aplicación: imaxes, vídeos, ficheiros xml, animacións, ficheiros para a internacionalización, etc.

- /res
- 

/**res**/**drawable-X**: almacenan as imaxes da aplicación en función da densidade (dpi) do dispositivo de destino.
- 

/res/layout: almacena os ficheiros xml onde se definen as distintas pantallas (cos seus elementos) que compoñen a aplicación. Se creamos o cartafol **/res/layout-land** e para definir pantallas distintas en función da orientación do dispositivo.

- 

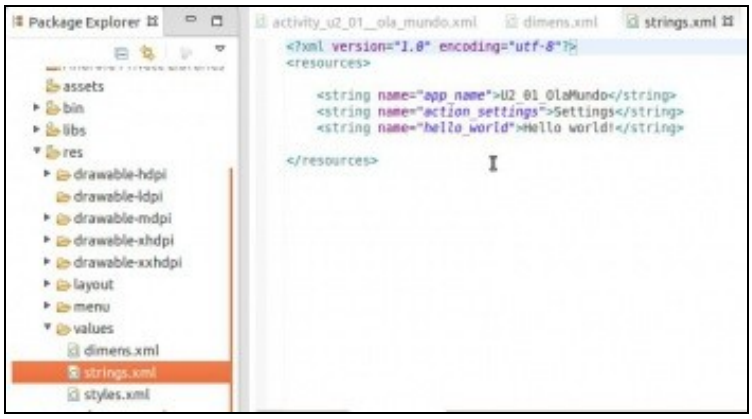
Este é o contido do ficheiro xml do layout. Observar que hai unha serie de referencias a outros recursos (as que comezan por @: @dim/... @string/...) (máis adiante veremos isto). Pero onde están definidas esas referencias ? ...

- 

/res/values: contén constantes que define: cadeas de texto, estilos, cores, dimensións, etc.

- 

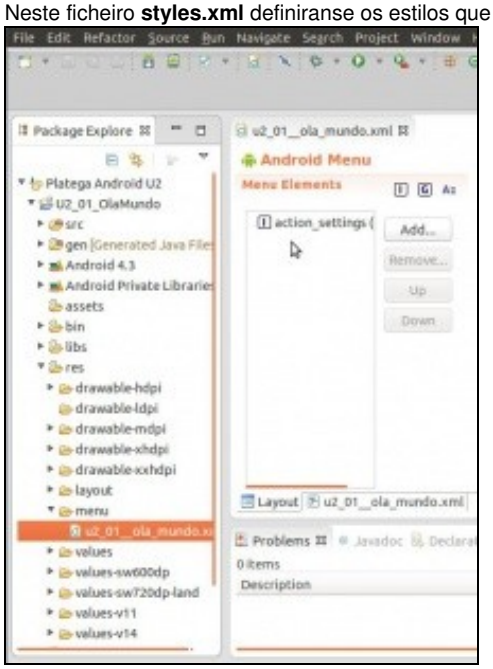
No ficheiro **dimens.xml** están declaradas as dimensións que se usaban no layout.

- 

No ficheiro **strings.xml** están declaradas cadeas de texto. A última é usada no layout, logo veremos onde se usan as demais.

- 

Neste ficheiro **styles.xml** definiranse os estilos que se usarán para homoxeneizar os elementos gráficos.

- 

/**res**/**menu** contén a definición dos menús da aplicación.

- 

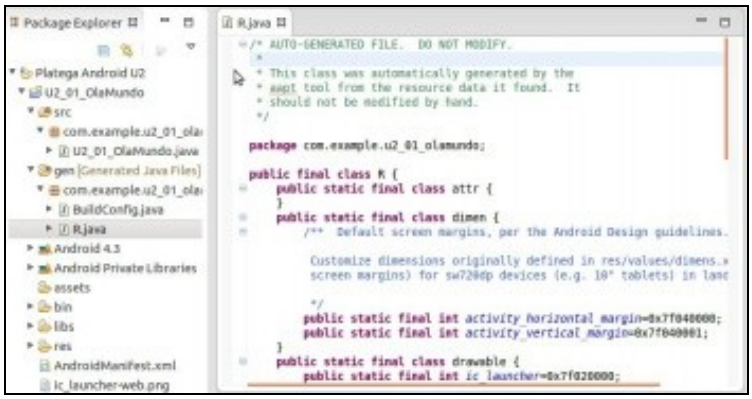
Aquí vemos no dispositivo como se acceden a eses menús.

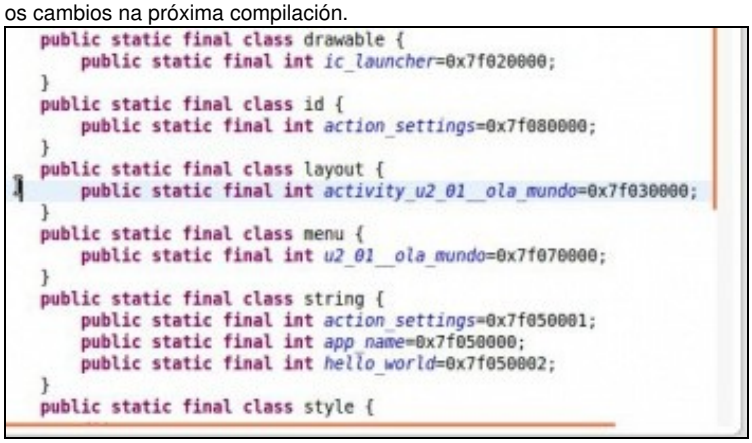
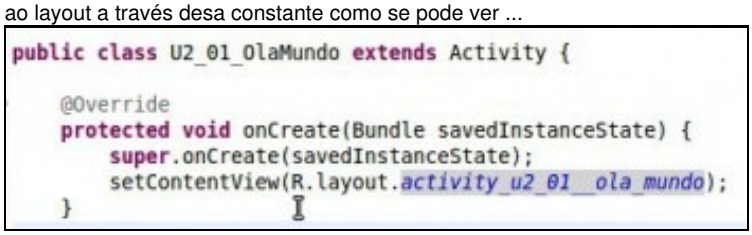
- 

Estes catro cartafoles serven para definir valores específicos para dispositivos que teñen versións de Android iguais ou superiores ás asociadas á API11 ou API14. Sobre os outros dous cartafoles seguro que atopas información en internet.

/gen

O cartafol /**gen** contén codigo Java creado automaticamente ao compilar a aplicación. O ficheiro máis importante é **R.java**.

- /res
- 

O ficheiro **R.java** define a clase R que serve de enlace entre os recursos creados en /res e o código Java da aplicación. Para iso, a clase R contén constantes cos **IDs** dos recursos de /**res**. Este ficheiro é xerado cada vez que se compila co cal se modifica manualmente perderanse os cambios na próxima compilación.
- 
```
public static final class drawable {
    public static final int ic_launcher=0x7f020000;
}
public static final class id {
    public static final int action_settings=0x7f080000;
}
public static final class layout {
    public static final int activity_u2_01__ola_mundo=0x7f030000;
}
public static final class menu {
    public static final int u2_01__ola_mundo=0x7f070000;
}
public static final class string {
    public static final int action_settings=0x7f050001;
    public static final int app_name=0x7f050000;
    public static final int hello_world=0x7f050002;
}
public static final class style {
```
Por exemplo, podemos ver que para o layout definido en /res, nun ficheiro xml, existe unha constante que nos permite acceder dende Java ao layout a través desa constante como se pode ver ...
- 
```
public class U2_01_OlaMundo extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u2_01__ola_mundo);
    }
```
... no código fonte asociado á Actividade que lanza a aplicación.

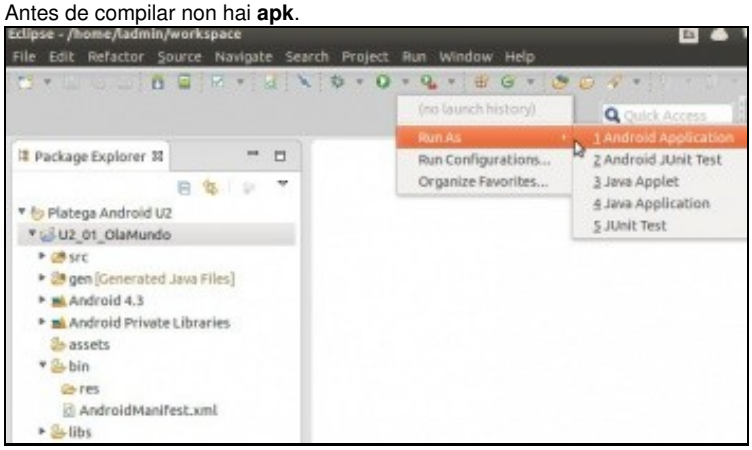
/assets

- Contén o resto de ficheiros que se precisen para a aplicación: ficheiros de configuración, de datos, etc. Para acceder a estes recursos non se fai a a través da clase R senón que a través da súa ruta no sistema

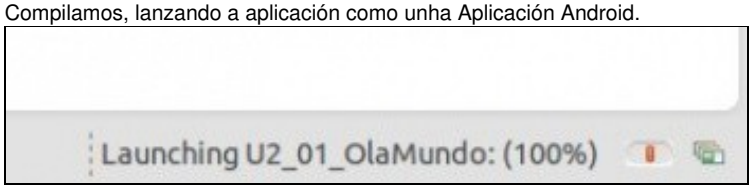
/bin

- Unha vez que se compila a aplicación xerase o **apk** da aplicación para poder ser instalada nun dispositivo.
- /bin
-

Antes de compilar non hai **apk**.

- 

Compilamos, lanzando a aplicación como unha Aplicación Android.

- 

Lanzando a aplicación

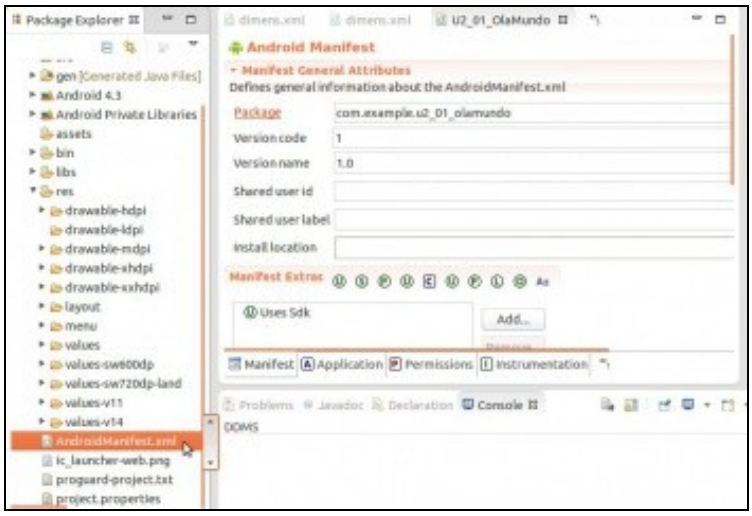
- 

O ficheiro **apk** que permite instalar a aplicación nun dispositivo e que Eclipse o fai por nós.

Ficheiro AndroidManifest.xml

- Contén a definición das características máis importantes da aplicación: nome, versión, icono, activities, servizos, instrumentación, permisos de acceso aos recursos, etc.

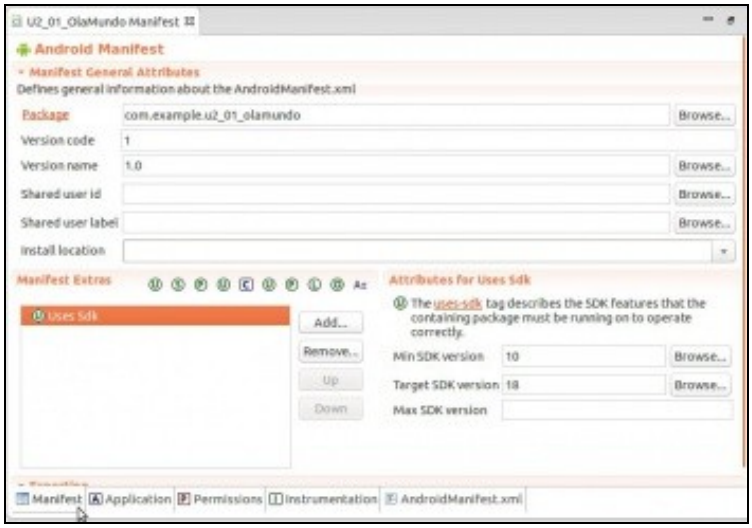
- AndroidManifest.xml

- 

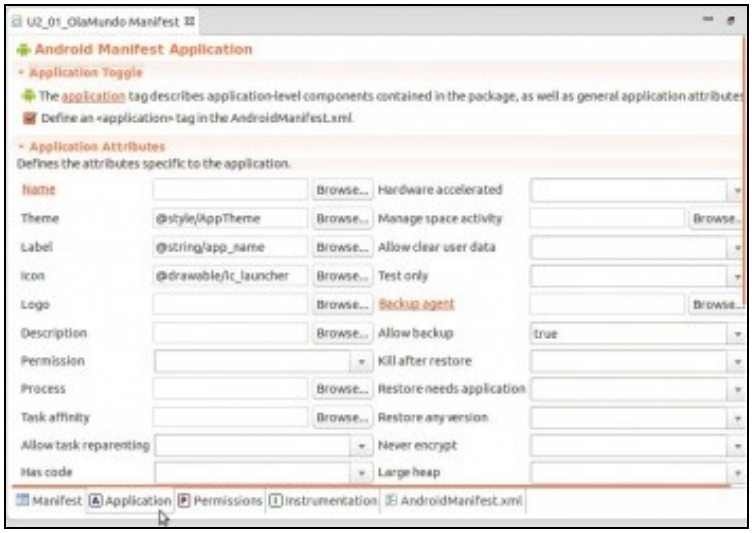
Facemos dobre click nel e vemos a lapela **Manifest**

- 

Maximizámola ...

- 

Vemos os SDK mínimo e obxectivo. Se o desexamos agora podemos cambiar eses valores. **Edición 2015**: lembrar que o SDK mínimo é o 16.

- 

Na lapela **Application** indicamos o tema, o nome da aplicación e a icona a usar. Observar que son referencias a outros recursos (@.../...). Poderíamos ir aos ficheiros ou cartafoles correspondentes de /res e mirar cal vai ser o seu valor.

- 

Non hai ningún permiso definido por defecto.

- 

Esta aplicación tampouco ten definida ningunha instrumentación do dispositivo para ser usada por ela.

- 

Todo o anterior, pero en formato xml editable.

- 

```
    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.u2_01_olamundo.U2_01_OlaMundo"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Podemos ver os SDKs mínimo e obxectivo (**Edición 2015:** lembrar SDK mínimo, como mínimo o 16), a icona que se vai usar para a aplicación, (foi o que escollemos cando a creamos, está accesible a través dunha referencia a un recurso), o nome da aplicación e o tema a usar. Revisar o ficheiro strings.xml de /res/values. Finalmente está a lista de activities que ten a aplicación, neste caso unha soa, a principal.

- Referencias:
  - http://developer.android.com/guide/topics/manifest/manifest-intro.html

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).