

# Programación multimedia y dispositivos móviles

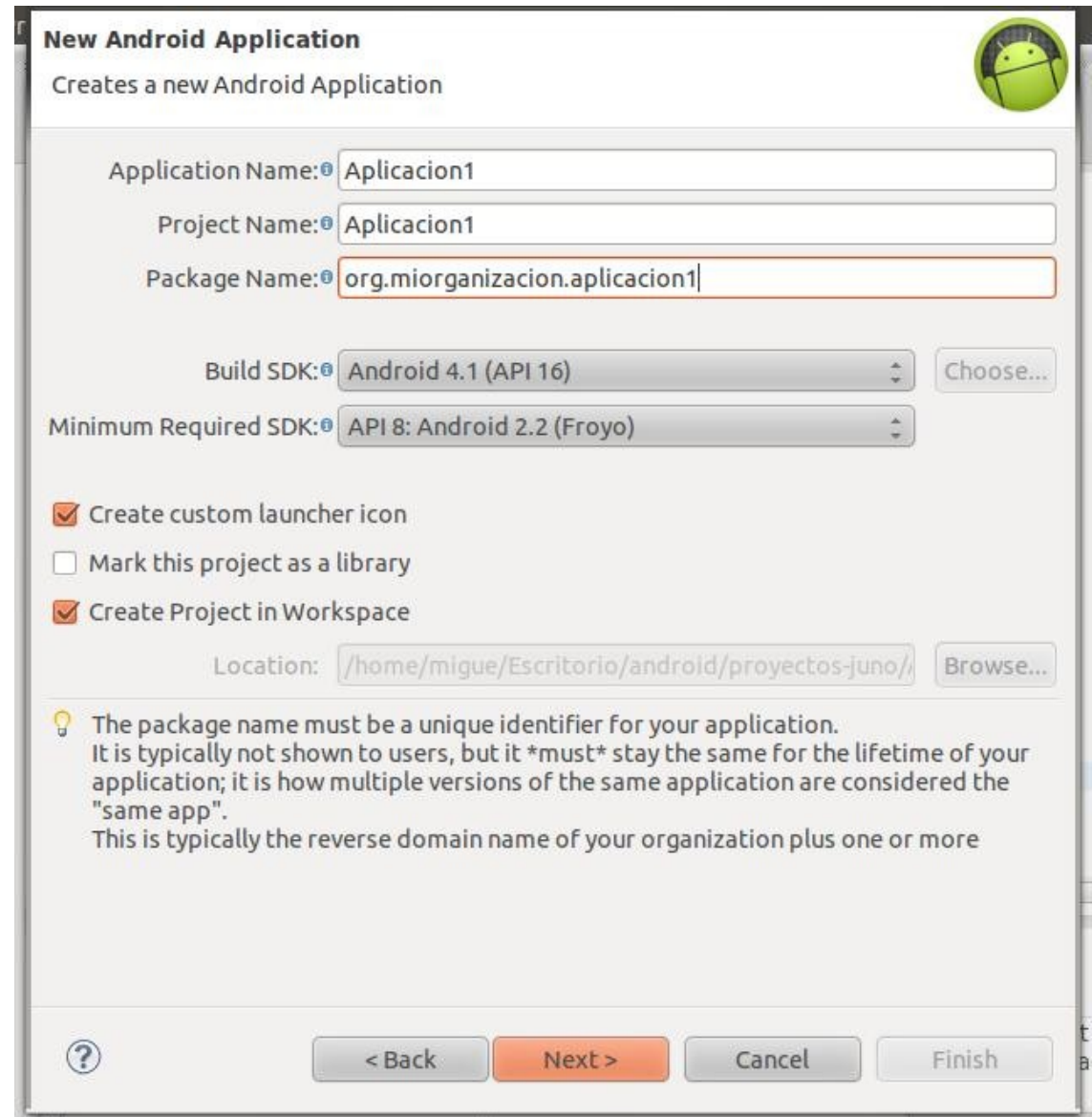
2

## Empezando a desarrollar en Android

IES Nervión  
Miguel A. Casado Alías

# Crear proyecto Android en Eclipse

- File – New – Project – Android – Android Application Project
- Rellenamos el nombre de la aplicación, el del proyecto y el paquete
- Elegimos la versión de Android SDK contra la que compilar, así como la versión de SDK más antigua en la que funcionará nuestra aplicación



The screenshot shows the 'New Android Application' dialog box in the Eclipse IDE. The dialog has a title bar with the text 'New Android Application' and a subtitle 'Creates a new Android Application'. It features an Android robot icon in the top right corner. The main area contains several input fields and checkboxes. The 'Application Name' field is set to 'Aplicacion1'. The 'Project Name' field is also set to 'Aplicacion1'. The 'Package Name' field is set to 'org.miorganizacion.aplicacion1' and is highlighted with a red border. Below these fields are two dropdown menus for 'Build SDK' (set to 'Android 4.1 (API 16)') and 'Minimum Required SDK' (set to 'API 8: Android 2.2 (Froyo)'). There are three checkboxes: 'Create custom launcher icon' (checked), 'Mark this project as a library' (unchecked), and 'Create Project in Workspace' (checked). A 'Location' field shows the path '/home/migue/Escritorio/android/proyectos-juno/' with a 'Browse...' button next to it. At the bottom, there is a lightbulb icon and a text box explaining that the package name must be a unique identifier for the application, typically the reverse domain name of the organization plus one or more. The bottom of the dialog has four buttons: '< Back', 'Next >' (highlighted in orange), 'Cancel', and 'Finish'.

**New Android Application**  
Creates a new Android Application

Application Name:

Project Name:

Package Name:

Build SDK:


Minimum Required SDK:

☒ Create custom launcher icon

☐ Mark this project as a library

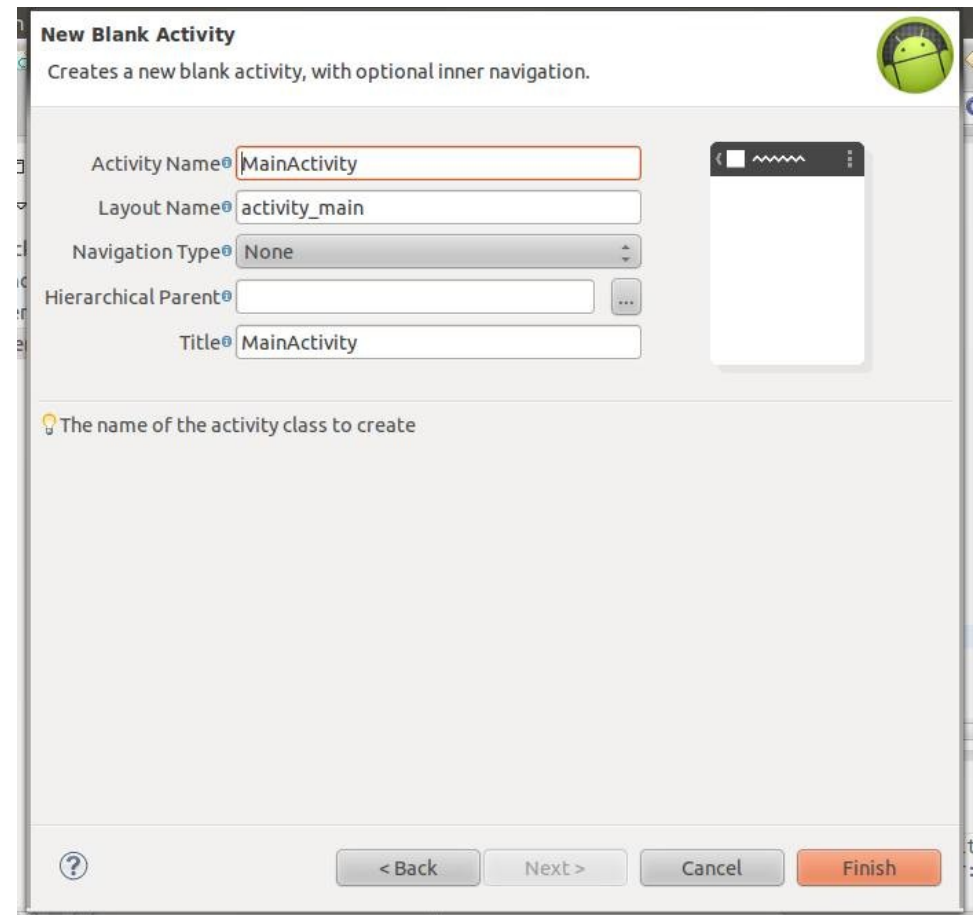
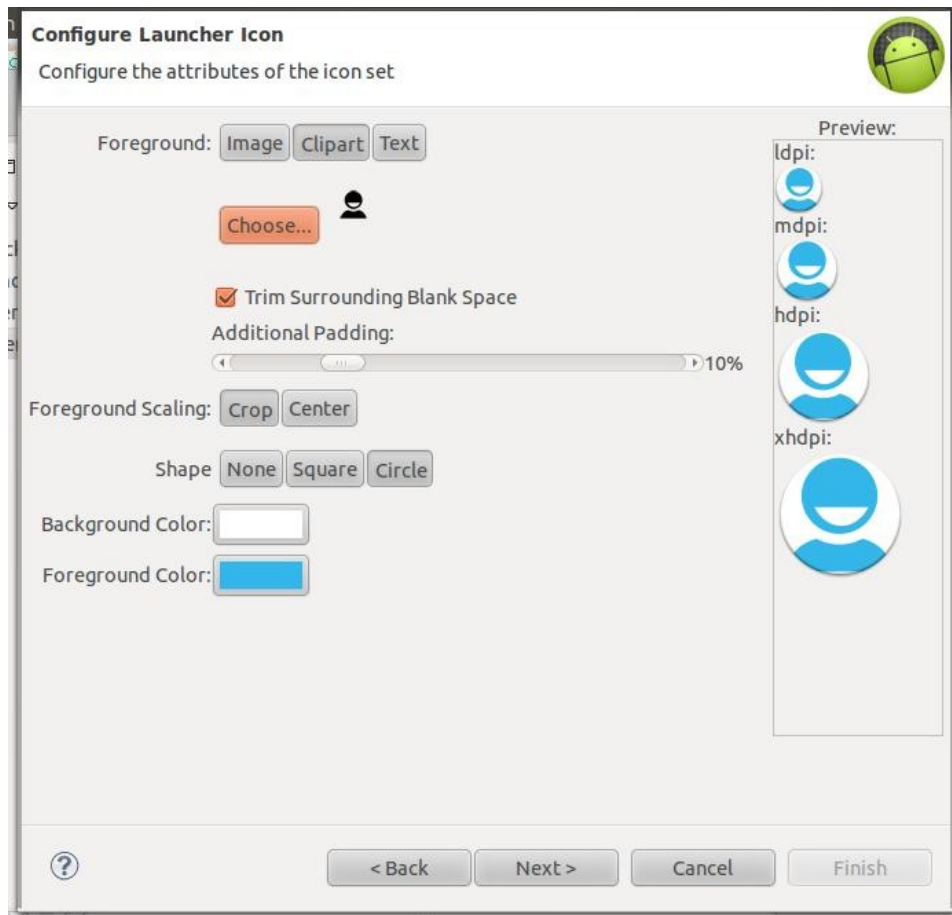
☒ Create Project in Workspace

Location:

 The package name must be a unique identifier for your application. It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app". This is typically the reverse domain name of your organization plus one or more

# Crear proyecto Android en Eclipse (II)

- Podemos crear un icono y la primera actividad del proyecto



# Principales componentes de una aplicación

---

- Activities
- Layouts y Views
- Intents
- Services
- Content Providers
- Broadcast Receivers
- Application Context

# Activities

---

- Podríamos decir que una actividad es una pantalla con todos sus elementos y acciones asociadas
- Las aplicaciones suelen tener varias actividades
- Al igual que un sitio web está compuesto de múltiples páginas webs, se puede pensar en una aplicación como que está compuesta de varias actividades
- El usuario “navega” de actividad en actividad
- Se puede “saltar” de una actividad de una aplicación a una actividad de otra aplicación
- Suele haber una actividad principal, que es la primera que se muestra al entrar en la aplicación

# Layouts y Views

---

- Las vistas son todos los elementos como botones, listas, cuadros de texto, etc...
- Los layouts agrupan las vistas para organizarlas
- Un layout puede contener otros layouts

# Intents

---

- No se deben traducir como “intentos”, sino más bien como un propósito, como la intención de hacer algo
- Son mensajes mediante los cuales las aplicaciones intercambian información con otras aplicaciones o con el sistema. Ejemplos:
  - El sistema notifica que se ha insertado una tarjeta SD
  - El sistema notifica que hay cobertura Wi-Fi
  - Una aplicación solicita abrir una página web, en tal caso lanza un intent, y todas las aplicaciones capaces de abrir una página web (por ejemplo el navegador Firefox) se prestan para completar dicha acción
- Una aplicación lanza un intent cuando solicita algo del sistema o de otra aplicación

# Services

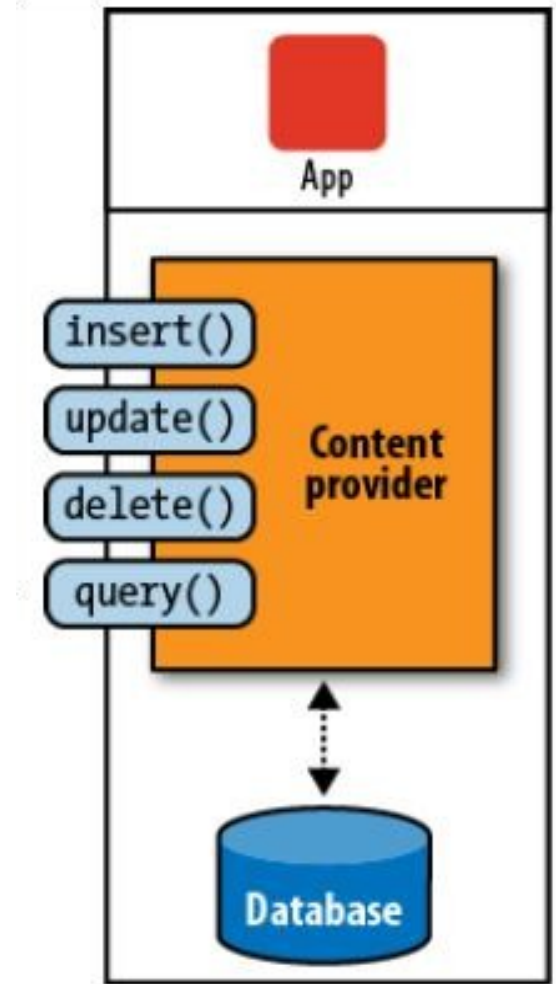
---

- Se ejecutan en segundo plano
- No tiene interfaz de usuario
- Son útiles para llevar a cabo acciones durante un buen tiempo independientemente de lo que se esté mostrando en la pantalla. Por ejemplo:
  - Queremos descargar un fichero de gran tamaño
  - Queremos hacer sonar música y que ésta no cese cuando estemos usando otras aplicaciones

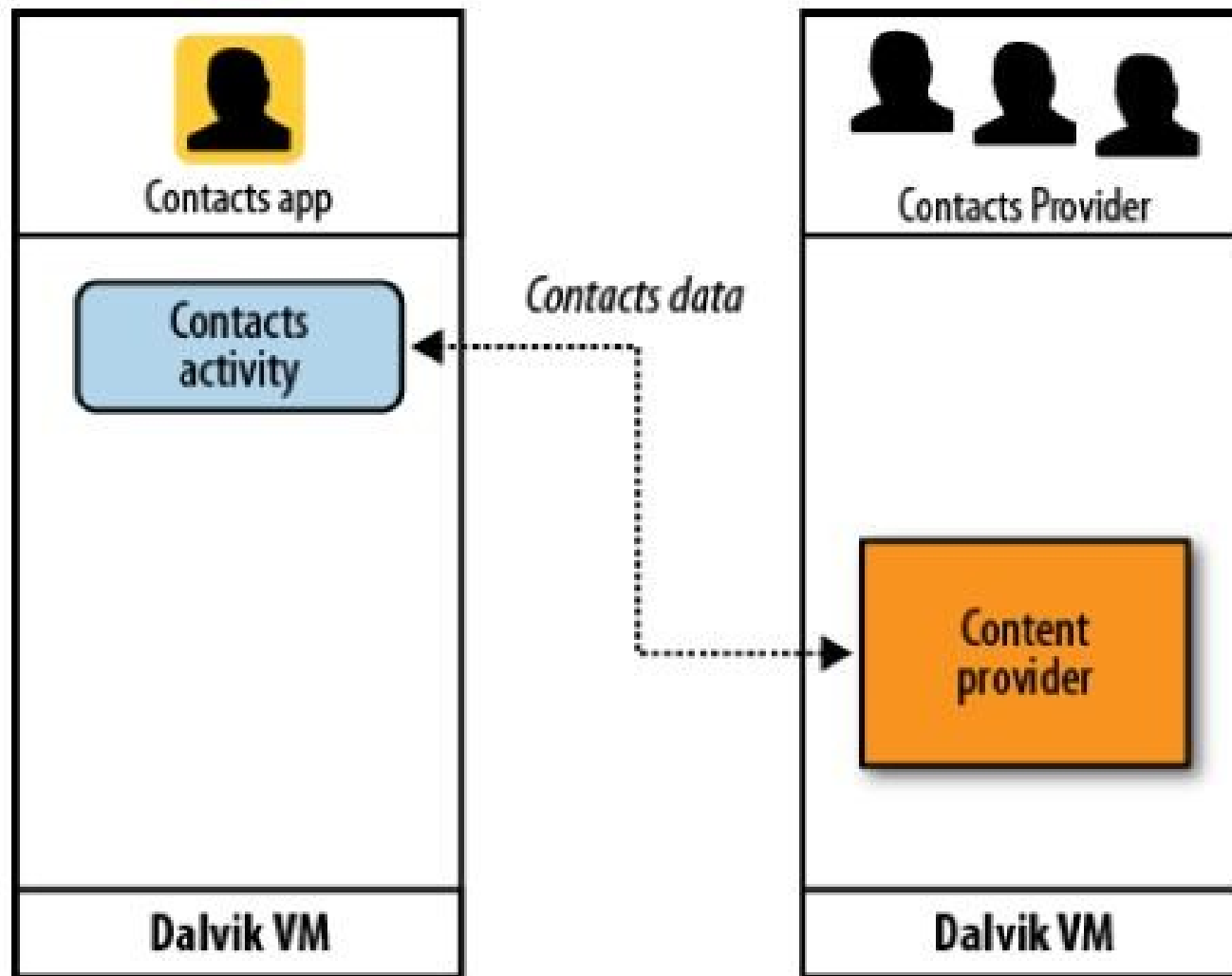


# Content Providers

- Son interfaces para compartir datos entre aplicaciones
- Por defecto Android ejecuta cada aplicación en su propio “contenedor”, aislando a cada aplicación y sus datos de las demás aplicaciones
- Ejemplos:
  - “Contacts Provider” proporciona información sobre los contactos a las aplicaciones que la necesiten
  - “Settings Provider” comparte los ajustes del sistema con las aplicaciones



## Content Providers (II)



# Broadcast Receivers

---

- Código destinado a llevar a cabo una acción cuando ocurra un evento determinado
- El sistema lanza mensajes broadcast continuamente. Por ejemplo: cuando llega una llamada, cuando se recibe un SMS, cuando la batería está cargada, etc...
- Si quisiéramos hacer una aplicación que baje el brillo de la pantalla al mínimo cuando la batería esté apunto de agotarse, deberíamos hacerlo mediante un “broadcast receiver”

# Application Context

---

- Actividades, Servicios, Proveedores de contenidos, etc... todos juntos forman una aplicación. Podríamos decir que todos ellos habitan en el mismo “**Contexto**”
- El contexto de la aplicación hace referencia al entorno de la misma y al proceso dentro del cual todos los componentes de la aplicación se ejecutan
- El contexto permite a los diversos “bloques de construcción” de la aplicación intercambiar información y recursos
- Las actividades y los servicios son subclases de contexto

# Explorando el proyecto

---

- `AndroidManifest.xml`: describe la aplicación y sus componentes
- `bin/` : alberga la aplicación una vez compilada
- `libs/` : librerías de terceros usadas por nuestra aplicación
- `res/` : recursos como imágenes, layouts, etc...
- `src/` : código fuente de nuestra aplicación
- `assets/` : más recursos, no accesibles a través de clase `R`
- `gen/` : código generado por las herramientas de desarrollo de Android

# La clase R.java

---

- La primera vez que se compila un proyecto se genera automáticamente R.class
- Contiene una serie de constantes enlazadas a los recursos disponibles en el directorio res/
- En nuestro código podemos hacer referencia a los recursos escribiendo algo como R.drawable.mi\_imagen
- NO SE DEBE MODIFICAR

# Layouts XML vs Layouts Java

---

- Layouts Java
  - Es mejor usarlos sólo cuando necesitamos instanciación dinámica de las vistas
  - Ejemplo: ver proyecto Skeleton/Now
- Layouts XML
  - Recomendado. Separamos visualización del contenido
  - Los creamos en res/layout
  - Los enlazamos con el código fuente java a través del método *setContentView* y valiéndonos de la clase R.java
  - Accedemos a los elementos mediante *findViewById*
  - Ejemplo: ver proyecto Layouts/NowRedux

## @+id

---

- Necesitamos poner un identificador a aquellos elementos de nuestro Layout basado en XML a los que queramos acceder desde el código java
  - Ejemplo: **android:id="@+id/boton1"**
- La convención es anteceder el nombre con **@+id/** **la primera vez que aparezca** ese elemento en nuestro fichero XML
- Con el "+" indicamos al parser que estamos creando un nuevo identificador, y que debe ser añadido a nuestra clase R.java
- Tras esa primera vez, nos referiremos al identificador con **@id/nombre\_del\_identificador**
  - Ejemplo: **android:id="@id/boton1"**