

# Interfaz Simple de Usuario

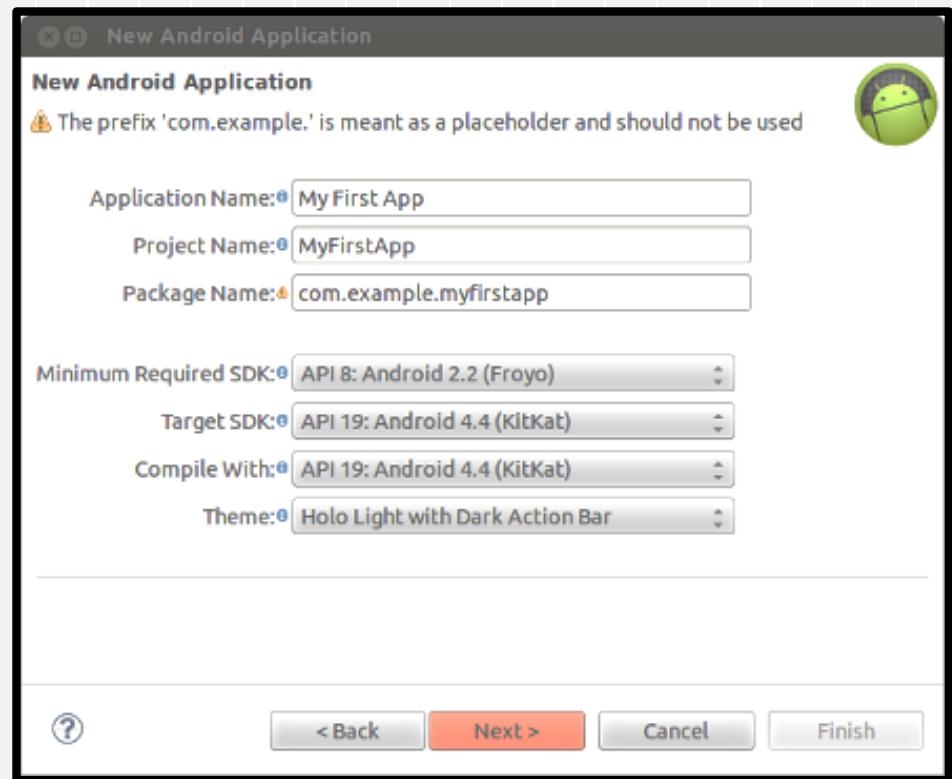
.....

## Aplicación Android

# Crear Nuevo Proyecto

## 1. Abrir el Eclipse y comenzar a crear un proyecto

- **Nombre de la aplicación** es el nombre de la aplicación que aparece a los usuarios. Para este proyecto, el uso "Mi Primera Aplicación".
- **Nombre del proyecto** es el nombre del directorio del proyecto y el nombre visible en Eclipse.



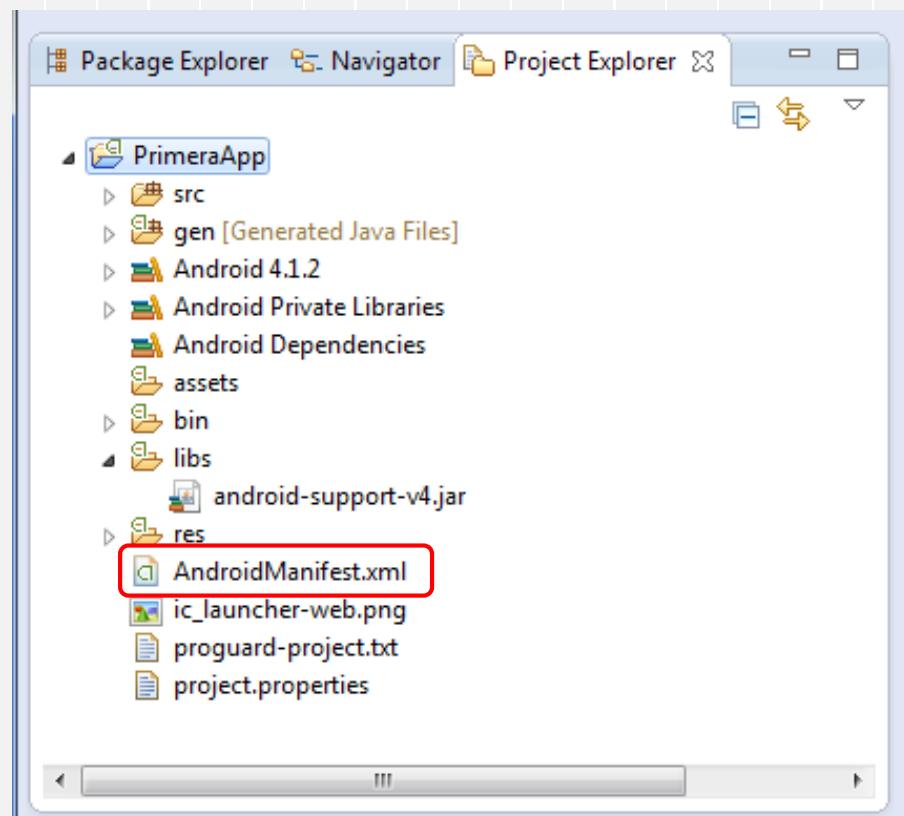
# Crear Nuevo Proyecto

2. **Haga clic en Siguiente.**
3. En la siguiente pantalla para configurar el proyecto, deje las selecciones predeterminadas y **haga clic en Siguiente.**
4. La siguiente pantalla puede ayudarle a crear un icono de lanzador para su aplicación. Puede personalizar un icono de varias maneras y la herramienta genera un ícono para todas las densidades de pantalla. Antes de publicar su aplicación, usted debe asegurarse de que su ícono cumple con las especificaciones definidas en la guía de diseño Iconografía. **Haga clic en Siguiente.**

# Crear Nuevo Proyecto

5. Para este proyecto, **seleccione BlankActivity y haga clic en Siguiente.** En este paso se selecciona la plantilla para la Actividad a crear. Empezaremos con **BlankActivity**.
  
6. Dejar todos los detalles de la actividad en su estado predeterminado y **haga clic en Finalizar.**

# Explorador de Proyectos



## AndroidManifest.xml:

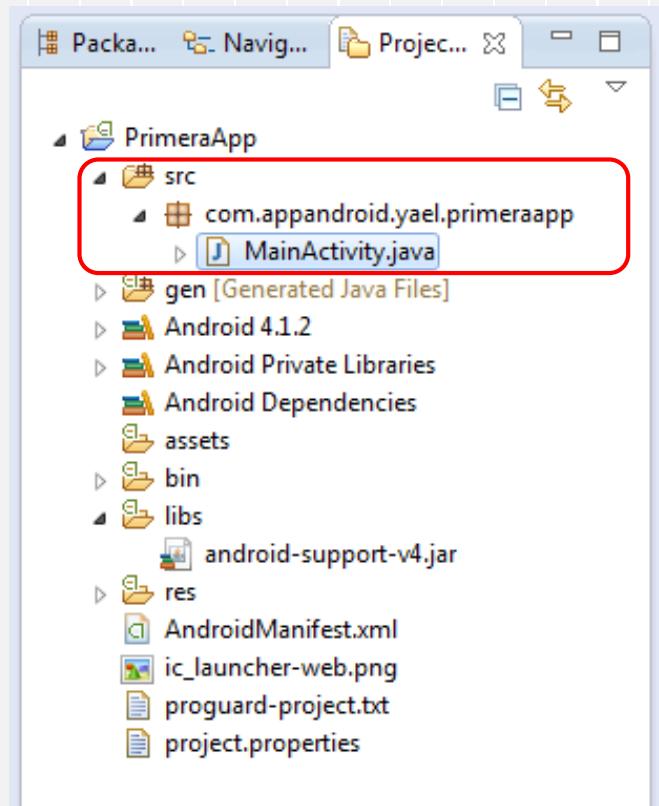
Describe las características fundamentales de la aplicación de sus componentes.

Todos los valores que se ingresan al crear un Proyecto se ven reflejados en ese archivo.

### Tarea:

Abrir el archivo AndroidManifest.xml en el editor.

# Explorador de Proyectos



## Directorio **/src** :

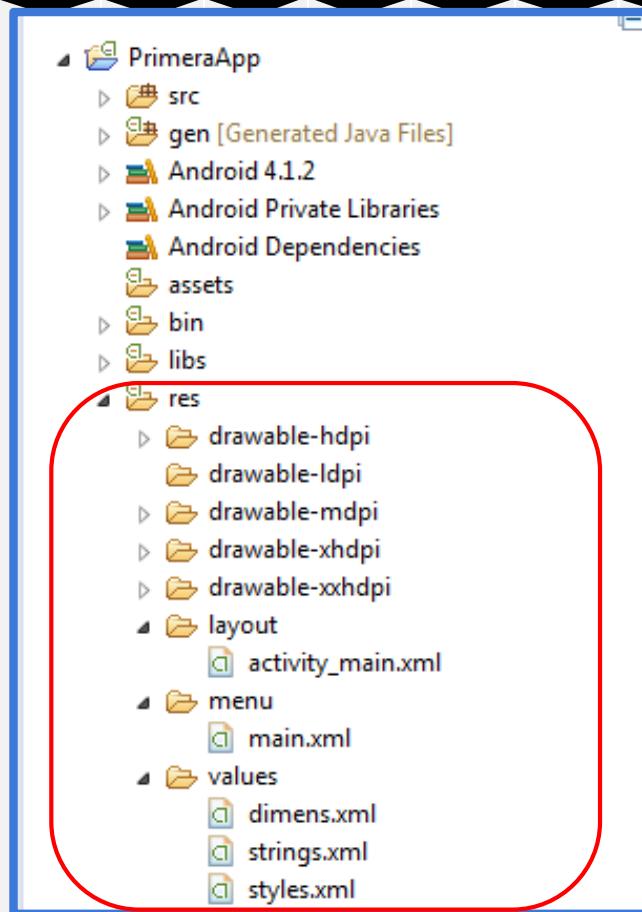
Directorio donde se alojan los principales archivos de código fuente de su aplicación.

Se incluye por defecto, un archivo de `MainActivity.java`, de tipo de actividad, que se ejecuta cuando se inicia su aplicación cuando se clickea el ícono de la aplicación.

## Tarea:

Abrir el archivo `MainActivity.java` en el editor.

# Explorador de Proyectos



## Directorio **res/** :

Carpeta donde se los recursos de su aplicación:

### **drawable-hdpi/**

Directorio para objetos estirables (tales como mapas de bits) que están diseñados para pantallas de alta densidad (HDPI).

Otros directorios de los objetos estirables contienen los diseños para otras densidades de pantalla.

### **layout/**

Diseño. Directorio de archivos que definen la interfaz de usuario de la aplicación.

### **values/**

Directorio para otros archivos XML que contienen las definiciones de cadenas y de color.

## Tarea:

Abrir el archivo los diferentes archivos.

# Temas

.....

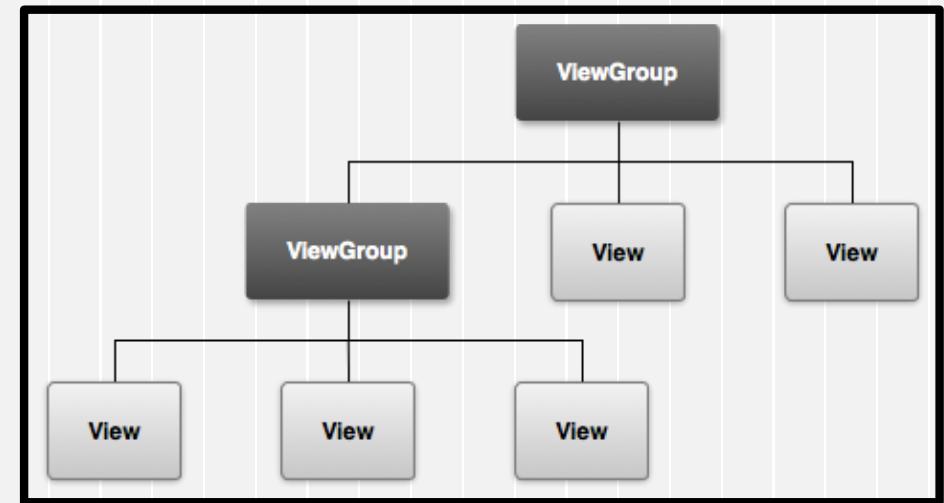
1. Crear un diseño lineal
2. Añadir un campo de texto
3. Añadir recursos de Cadena (String)
4. Añadir un Botón
5. Hacer que el Cuadro de entrada se extienda al ancho de pantalla

# Interfaz Simple de Usuario

Las interfaz gráfica para crear aplicaciones trabaja con el concepto

- Vista (View) y
- Vista de Grupo (ViewGroup).

Los objetos **View** son por lo general los “widgets” de interfaz de usuario, como botones o campos de texto y los objetos **ViewGroup** son contenedores invisibles que definen cómo los objetos View se disponen, como en una rejilla o una lista vertical.



# Crear un diseño lineal

Abrir `activity_main.xml` de la carpeta `res/layout/`.

La plantilla `BlankActivity` que se escogió cuando se creó este proyecto incluye el archivo `activity_main.xml` que tiene un objeto root de `RelativeLayout` y una vista hija de `TextView`.

En primer lugar, eliminar tag `<TextView>` y cambiar el `<RelativeLayout>` para `<LinearLayout>`.

Luego agregar el atributo `android:orientation` y ponerlo en "horizontal". (`android:orientation="horizontal"`)

El resultado se ve así:

`<LinearLayout`

```
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal" >
```

`</LinearLayout>`

# Agregar un Campo Texto

Para crear un campo de texto editable por el usuario, agregue un elemento <EditText> dentro de la estructura <LinearLayout>.

Como todo objeto, View, debe definir ciertos atributos XML para especificar las propiedades del objeto EditText.

He aquí cómo usted debe declararlo dentro del elemento <LinearLayout>:

## <LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="horizontal" >  
  
    <EditText android:id="@+id/edit_message"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:hint="@string/edit_message" />  
  
    </LinearLayout>
```

# Agregar un Campo Texto

Como @string no está definido en nuestro archivo de valores de cadena nos informará un error al escribir el código.

error: Error: No resource found that matches the given name (at 'hint' with value '@string/edit\_message').

## <LinearLayout>

```
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal" >  
  
        <EditText android:id="@+id/edit_message"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:hint="@string/edit_message" />  
  
    </LinearLayout>
```

# Agregar un Campo Texto

## Atributos definidos:

### android:id

Esto proporciona un identificador único para el objeto view, que se puede utilizar para hacer referencia al objeto en el código de aplicación, tales como leer (get) y manipular (set) el objeto.

El signo de arroba (@) es necesario siempre que se refiere a cualquier objeto de los recursos de XML. Es seguido por el tipo de recurso (id en este caso), una barra /, entonces el nombre del recurso (edit\_message). El signo más (+), va sólo cuando se está definiendo un ID de recurso por primera vez. Al compilar la aplicación, las herr. del SDK utilizan el nombre de ID para crear un nuevo ID de recurso en el archivo de gen / R.java de su proyecto que se refiere al elemento EditarTexto.

### <LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="horizontal" >  
  
    <EditText android:id="@+id/edit_message"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:hint="@string/edit_message" />  
  
</LinearLayout>
```

# Agregar un Campo Texto

## Atributos definidos:

android:layout\_width  
android:layout\_height

En lugar de utilizar tamaños específicos para el ancho y la altura, el valor "wrap\_content" especifica que la vista debe tan grande como sea necesario para ajustarse a los contenidos de la vista. Si se va a usar en su lugar "match\_parent", entonces el elemento EditText llenaría la pantalla, ya que podría coincidir con el tamaño de la LinearLayout padre.

## <LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="horizontal" >  
  
    <EditText android:id="@+id/edit_message"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:hint="@string/edit_message" />  
  
</LinearLayout>
```

# Agregar un Campo Texto

## Atributos definidos:

### android:hint

Es el String (cadena) predeterminado que se mostrará cuando el campo de texto esté vacío.

- En lugar de utilizar una cadena fija, utiliza el valor de la **"@string/edit\_message"** que refiere a un recurso String (cadena) que se definen en un archivo separado.
- Se refiere a un recurso concreto (no sólo un identificador), y por eso no necesita necesitar el signo más (+).

Es por eso que verá un error de compilación en un primer momento porque no se ha definido el recurso de cadena todavía. ( Vamos a arreglar esto en la siguiente paso mediante la definición de la cadena.)

### <LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="horizontal" >  
  
    <EditText android:id="@+id/edit_message"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:hint="@string/edit_message" />  
  
</LinearLayout>
```

# Agregar un Recurso String

- Cuando se necesita agregar texto en la interfaz de usuario, siempre se debe especificar cada cadena como un recurso.
- Los recursos de cadena permiten gestionar todos los textos de interfaz de usuario en un solo lugar, lo que hace que sea más fácil de encontrar y el texto de actualización.
- Externalizar las cadenas también le permite localizar su aplicación a diferentes idiomas, proporcionando definiciones alternativas para cada recurso de cadena.
- Por defecto, el proyecto Android incluye un archivo de recurso de cadena en `res/values/strings.xml`.

Ahora vamos a añadir una nueva cadena denominada "edit\_message" y establecer el valor a "Introduzca un mensaje." (Puede eliminar la cadena "hello\_world".)

Añada también una cadena "Enviar" para un botón que se va a agregar "button\_send".

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Introduzca un mensaje</string>
    <string name="button_send">Send</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

# Agregar un Botón

- Abrir el activity\_main.xml
- Agregue un botón para el diseño con un tag <Button>, inmediatamente después del elemento <EditText>:

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_send" />
```

- La altura y el ancho se ajustan a "wrap\_content" por lo que el botón será tan grande como sea necesario para ajustarlo al texto del botón.
- Este botón no necesita el atributo android:id, ya que no se hace referencia desde el código de la activity.

# Ancho del InputBox

- Abrir el activity\_main.xml
- Queremos lograr que el campo para ingresar texto tenga el mismo ancho que su contenido.
- Esto funciona bien para el objeto botón, pero no así para el campo de texto, ya que el usuario puede escribir algo más largo. Por lo tanto, sería bueno para llenar el ancho de la pantalla sin usar con el campo de texto.
- Se puede hacer esto dentro de un LinearLayout con la propiedad de weight (peso), que se puede especificar mediante el atributo android: layout\_weight.

Para llenar el espacio restante en su diseño con el elemento EditText, darle un peso de 1 y dejar el botón sin peso.

```
<EditText  
    android:layout_weight="1"  
    android:layout_width="0dp"  
    ... />
```

# Ancho del InputBox

El valor de peso es un número que especifica la cantidad de espacio que cada View debe consumir, en función del espacio que queda disponible en relación con la cantidad consumida por otras View de "hermanos".

Esto funciona algo así como la cantidad de los ingredientes en una receta de la bebida: "2 partes de vodka, 1 parte de licor de café" significa dos tercios de la bebida es el vodka.

Por ejemplo, si se le da a una View un peso de 2 y otro un peso de 1, la suma es 3, por lo que la primera vista se llena 2/3 del espacio restante y el segundo punto de vista se llena el resto.



Si añade una tercer View y le da un peso de 1, entonces la primera vista (con peso de 2) ahora recibe 1/2 el espacio restante, mientras que los dos restantes cada uno recibe 1/4.



El peso predeterminado para todos los puntos de vista es 0, por lo que si se especifica ningún valor de peso superior a 0 a un solo punto de vista, entonces esa View llena cualquier espacio que queda después de que a todas las View se les da el espacio que necesitan.

# activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

# Agregar Acción a un Botón

- Vamos a Cambiar el valor del EditText cuando se presiona el Botón.
- **Paso 1: Modificar propiedad en Layout**
  - Abrir el layout activity\_main.xml
  - Agregue al botón dentro del tag <Button>, el atributo android:id para poder identificarlo dentro de la clase R de Recursos del Proyecto. La sentencia es:

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_send"  
    android:id="@+id(btn_modificar_texto" />
```

# Agregar Acción a un Botón

- **Paso 2: Modificar propiedad en Layout**
  - Agregue al EditText dentro del tag <EditText>, el atributo android:id para poder identificarlo dentro de la clase R de Recursos del Proyecto. La sentencia es la misma que la anterior para View Button:

```
<EditText android:id="@+id/edit_message"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message"
    android:id="@+id/txt_edit_message" />
```

# Agregar Acción a un Botón

- **Paso 2: Agregar el código en la clase java**
  - Abrir clase java **MainActivity.java**
  - Definir las variables de la clase que vamos a usar en función de View del layout (Button y EditText)
  - Asignar a las variables las View del layout.
  - Dentro del método **OnCreate** escribir las siguientes sentencias:

```
private EditText TxtTexto;  
private Button BtnModificarTexto;
```

```
TxtMensaje = (EditText)findViewById(R.id.txt_mensaje);  
btnBoton1 = (Button)findViewById(R.id.BtnBoton1);
```

**cambiar los nombres de las variables. no hacer Copy Paste**

# Agregar Acción a un Botón

- **Paso 2: Agregar el código en la clase java**
  - Programar las acciones para la acción del Click del Botón
  - Dentro del método **OnCreate**, debajo de lo anterior, escribir lo siguiente:

```
btnBoton1.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View arg0)  
    {  
        TxtMensaje.setText("Botón pulsado!");  
    }  
});
```



Escribir el texto cambiando los nombres de las variables. no hacer Copy Paste

# Más Información

.....

**developer.android.com**

Botón ( Button )

<http://developer.android.com/guide/topics/ui/controls/button.html>

<http://developer.android.com/reference/android/widget/Button.html>

Campo Texto ( EditText )

<http://developer.android.com/guide/topics/ui/controls/text.html>

<http://developer.android.com/reference/android/widget/EditText.html>