

# SEMINARIO DE LENGUAJES

## OPCIÓN ANDROID

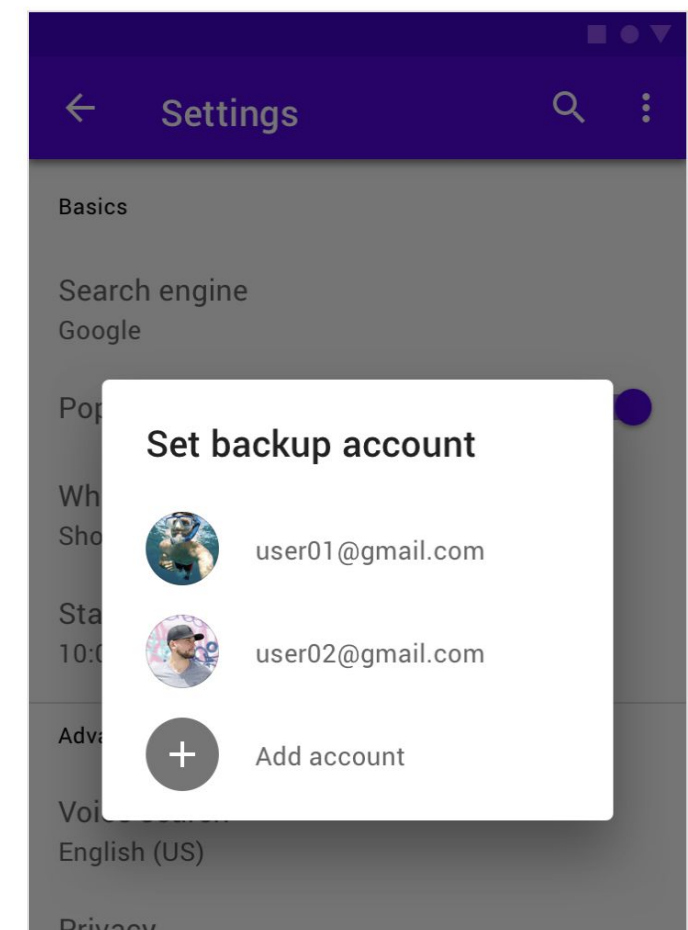
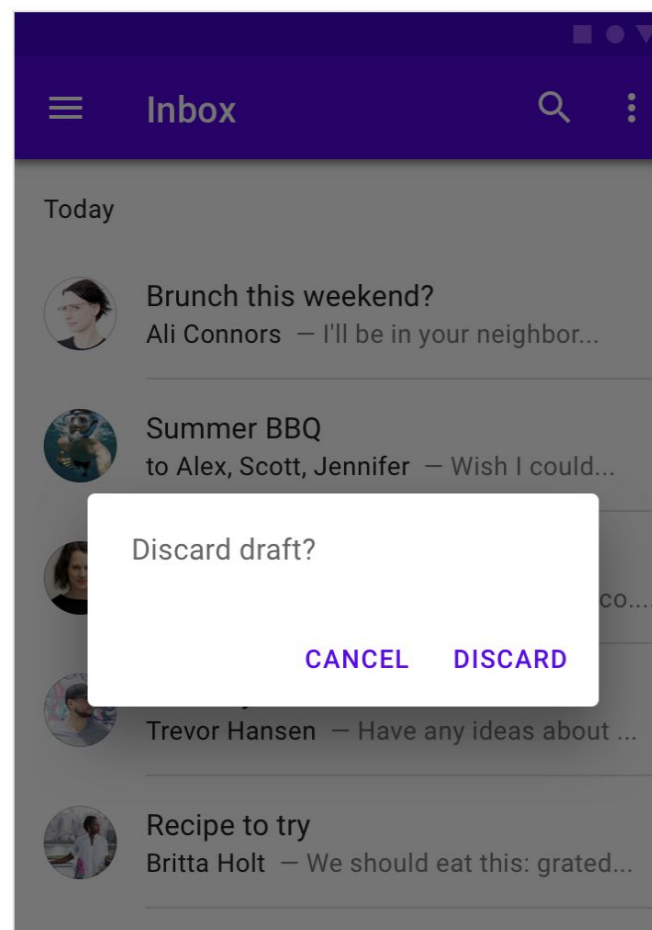


Diálogos y Notificaciones

Esp. Fernández Sosa Juan Francisco

# Diálogos

- Ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional.
- No ocupa toda la pantalla y generalmente se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar.



# Diálogos

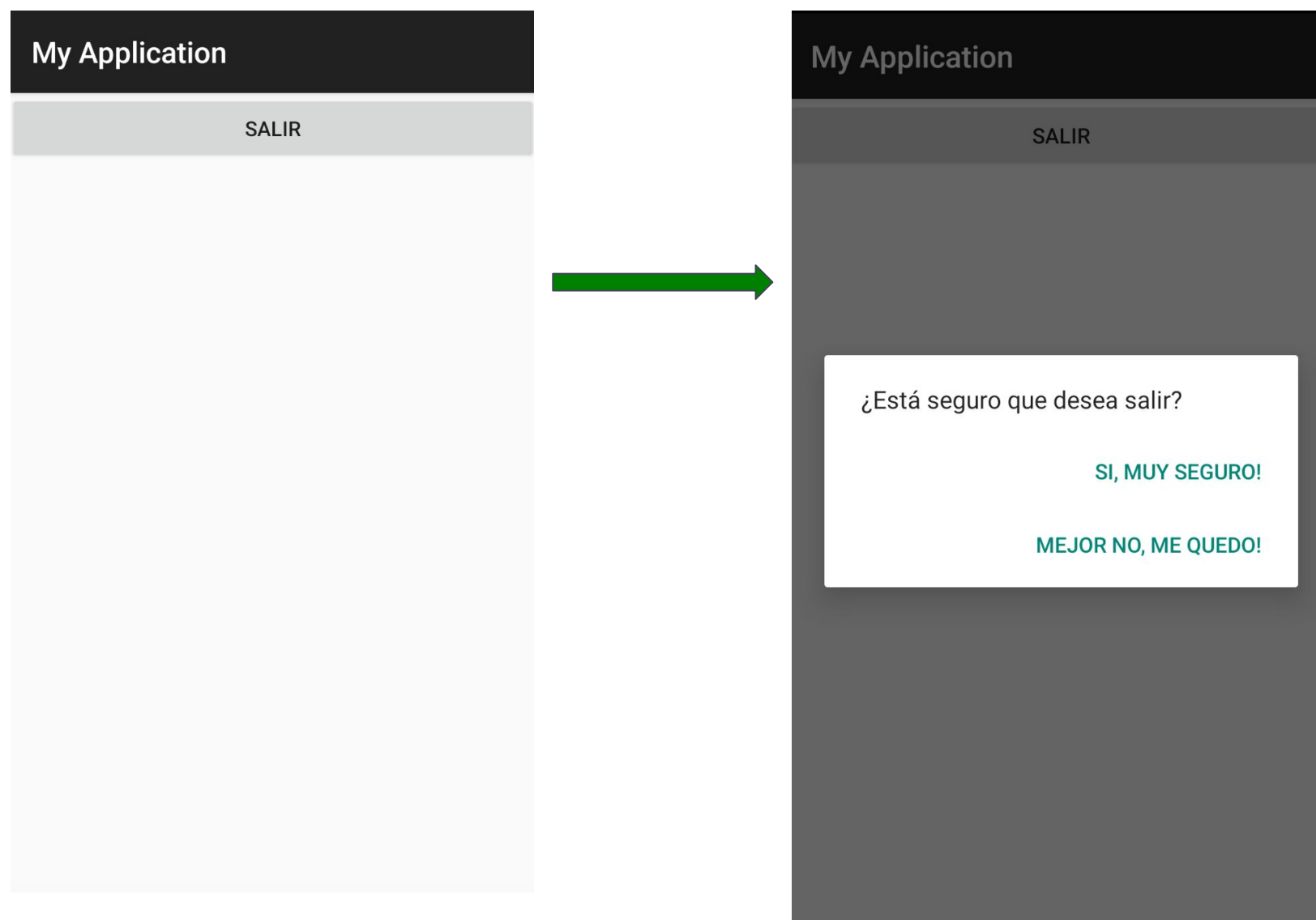
- Es posible diseñar nuestros propios diseños de diálogo.
  - Lectura adicional:  
<https://material.io/design/components/dialogs.html?hl=es-419>
- Existen diálogos predefinidos:
  - **AlertDialog**
    - Un diálogo que muestra un título, hasta tres botones, una lista de ítems para seleccionar o un layout específico.
  - **DatePickerDialog**
    - Diálogo para seleccionar una fecha
  - **TimePickerDialog**
    - Diálogo para seleccionar una hora.

# Diálogos

- Los diálogos se crean extendiendo la clase **DialogFragment**
- En primer lugar se debe **crear una clase** que extienda de **DialogFragment**. Dicha clase será la encargada de generar el diálogo, sobrescribiendo el método **onCreateDialog()**
- En segundo lugar, se debe mostrar en pantalla el diálogo creado.

# Diálogos - Actividad guiada

- Objetivo: Abrir un alerta de confirmación (diálogo) al presionar un botón.
- Crear un nuevo proyecto en Android Studio



# Diálogos - Actividad guiada

- Definir los siguientes strings en el archivo de recursos. Serán utilizados en la Alerta de Confirmación

```
<resources>
```

```
...
```

```
    <string name="salir">Salir</string>
```

```
    <string name="confirmacion">¿Está seguro que desea  
salir?</string>
```

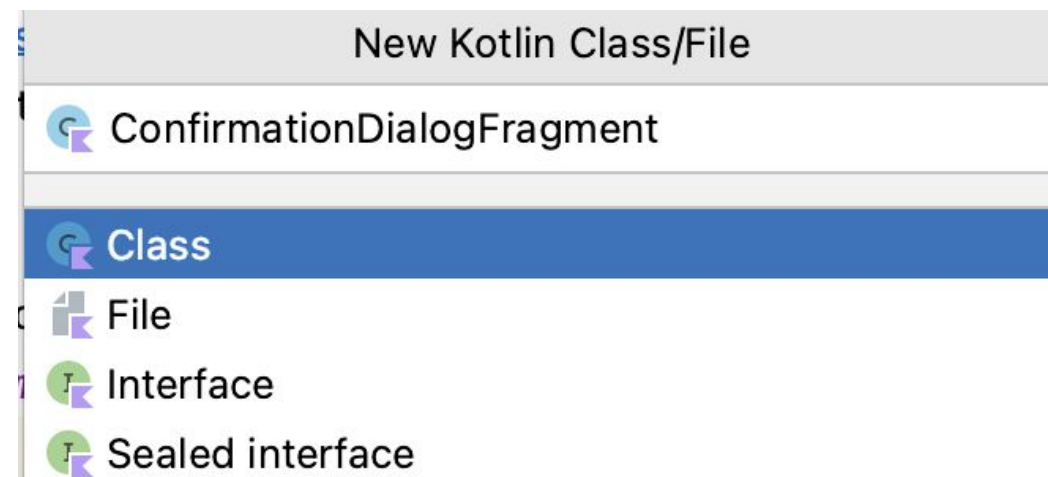
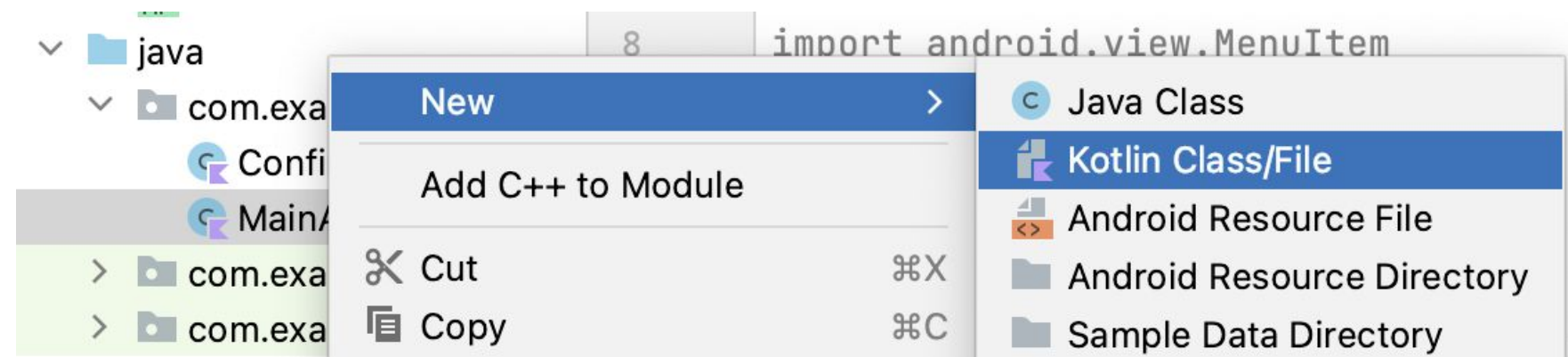
```
    <string name="aceptar">Si, muy seguro!</string>
```

```
    <string name="cancelar">Mejor no, me quedo!</string>
```

```
</resources>
```

# Diálogos - Actividad guiada

- Crear una nueva clase Kotlin, a la altura de la clase MainActivity. Lllamarla **ConfirmationDialogFragment**
- Extender a la superclase **DialogFragment**



# Diálogos - Actividad guiada

- Extender la clase **ConfirmationDialogFragment** de la superclase **DialogFragment()**
- Declarar y codificar el método **onCreateDialog()**. Se explicará en detalle en la siguiente filmi

```
class ConfirmationDialogFragment : DialogFragment() {
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        return AlertDialog.Builder(requireContext())
            .setMessage(getString(R.string.confirmacion))
            .setPositiveButton(getString(R.string.aceptar)) { dialog, which ->
                //acciones a realizar cuando se presiona Confirmar
            }
            .setNegativeButton(getString(R.string.cancelar)) { dialog, which
->
                //acciones a realizar cuando se presiona Cancelar
            }
            .create()
    }
}
```



# Diálogos - Actividad guiada

Se crea una instancia de la clase  
AlertDialog.Builder

Contexto actual de la aplicación

```
class ConfirmationDialogFragment : DialogFragment() {
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        return AlertDialog.Builder(requireContext())
            .setMessage(getString(R.string.confirmacion))
            .setPositiveButton(getString(R.string.aceptar)) { dialog, which ->
                //acciones a realizar cuando se presiona Confirmar
            }
            .setNegativeButton(getString(R.string.cancelar)) { dialog, which ->
                //acciones a realizar cuando se presiona Cancelar
            }
            .create()
    }
}
```


Mensaje que se mostrará en el  
diálogo. Se puede configurar un  
título con el método setTitle()

Botón de opción positiva y negativa, junto con  
las acciones que se ejecutarán cuando se haga  
clic sobre ellos

# Diálogos - Actividad guiada

- Editar el archivo activity\_main.xml
- Se invocará al diálogo al hacer clic en el botón.

```
<LinearLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/salir"
        android:onClick="salir" />
</LinearLayout>
```



Se debe implementar el manejador del clic, para lanzar el diálogo

# Diálogos - Actividad guiada

- Agregar a la clase MainActivity el método salir

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
    fun salir(v: View) {  
        val alert = ConfirmationDialogFragment()  
        alert.show(supportFragmentManager, "ConfirmationDialog")  
    }  
}
```

Se crea una instancia del Diálogo configurado anteriormente

El método `show()` de la clase **DialogFragment** muestra en pantalla el

Verificar  
comportamiento en  
el Emulador

# Manejando la respuesta del usuario - Actividad guiada

- Agregue las sentencias sombreadas en el método **onCreateDialog** de la clase **ConfirmationDialogFragment**

```

override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
    return AlertDialog.Builder(requireContext())
        .setMessage(getString(R.string.confirmacion))
        .setPositiveButton(getString(R.string.aceptar)) { dialog, which ->
            //acciones a realizar cuando se presiona Confirmar
            activity?.finish();
        }
        .setNegativeButton(getString(R.string.cancelar)) { dialog, which ->
            //acciones a realizar cuando se presiona Cancelar
            Toast.makeText(getActivity(), "El usuario decidió quedarse",
                Toast.LENGTH_SHORT).show();
        }
        .create()
}

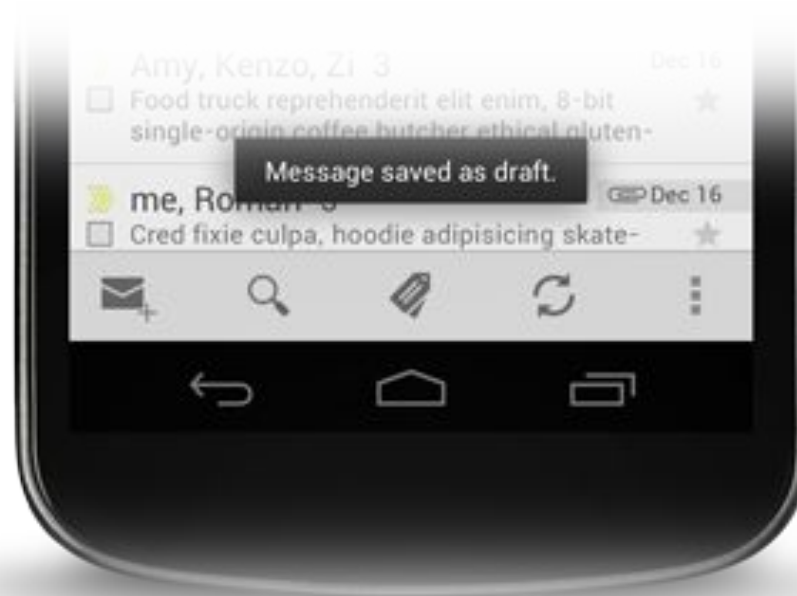
```

# Notificaciones

- Se conocerán tres tipos de notificaciones
  - **Toast**
  - **Barra de estados**
  - **Snackbar**

# Toast

- Mensaje que se muestra en pantalla durante unos segundos.
- No requiere intervención por parte del usuario.
- No interfiere con las acciones que está llevando a cabo el usuario
- Desaparece automáticamente.
- No requiere de una actividad en foreground. Puede ser lanzado desde un servicio por ejemplo.



# Toast

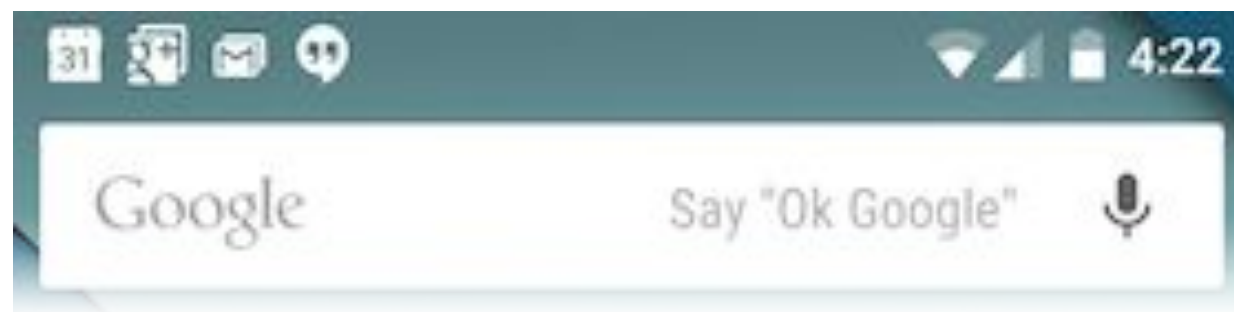
- Son ideales para mostrar mensajes rápidos y sencillos al usuario.
- Al no requerir confirmación, no se debería utilizar para hacer avisos importantes
- Invocación:

```
Toast.makeText(this, "¡Hola Mundo!", Toast.LENGTH_SHORT).show()
```

- La duración puede ser:
  - Toast.LENGTH\_LONG
  - Toast.LENGTH\_SHORT

# Notificaciones en la barra de estado

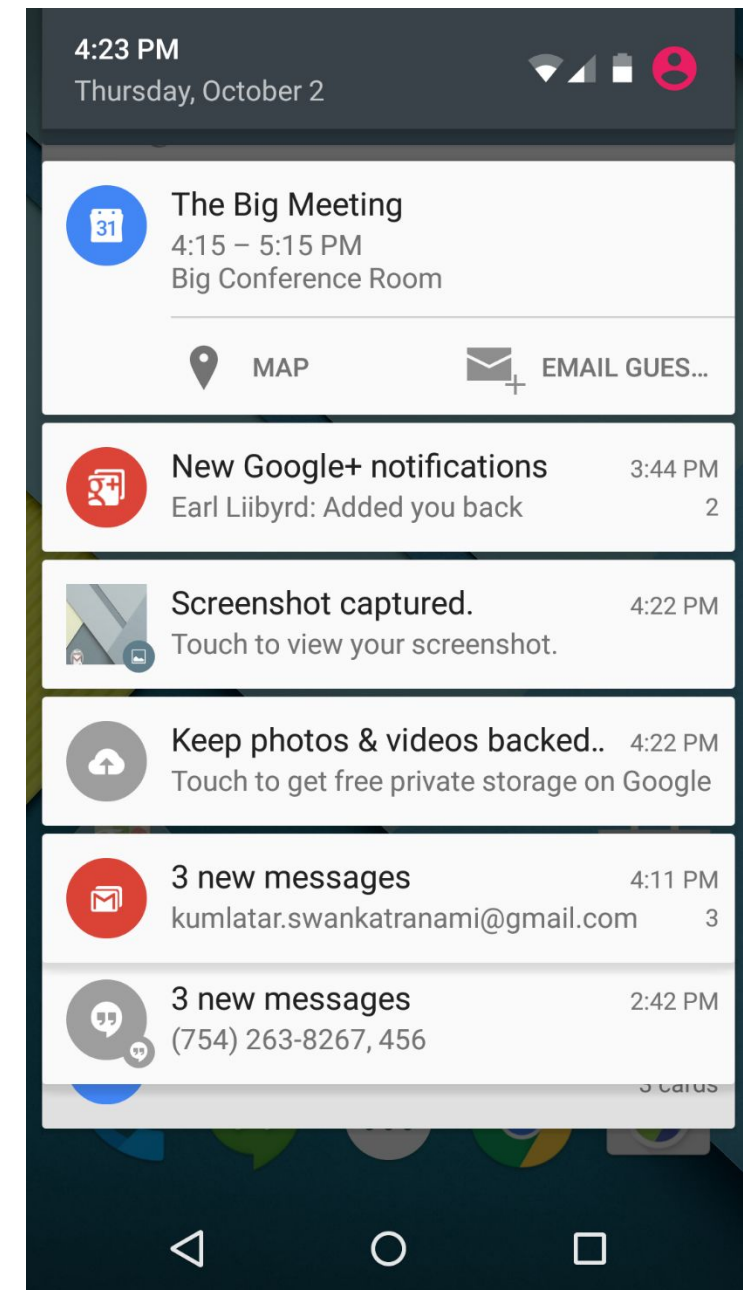
- Permite mostrar información al usuario, de un modo más persistente que mediante Toast.
- La notificación se muestra fuera de la interfaz de usuario de la aplicación, en la barra de estados.
- Inicialmente se muestra el aviso mediante un ícono, en el área de notificaciones.



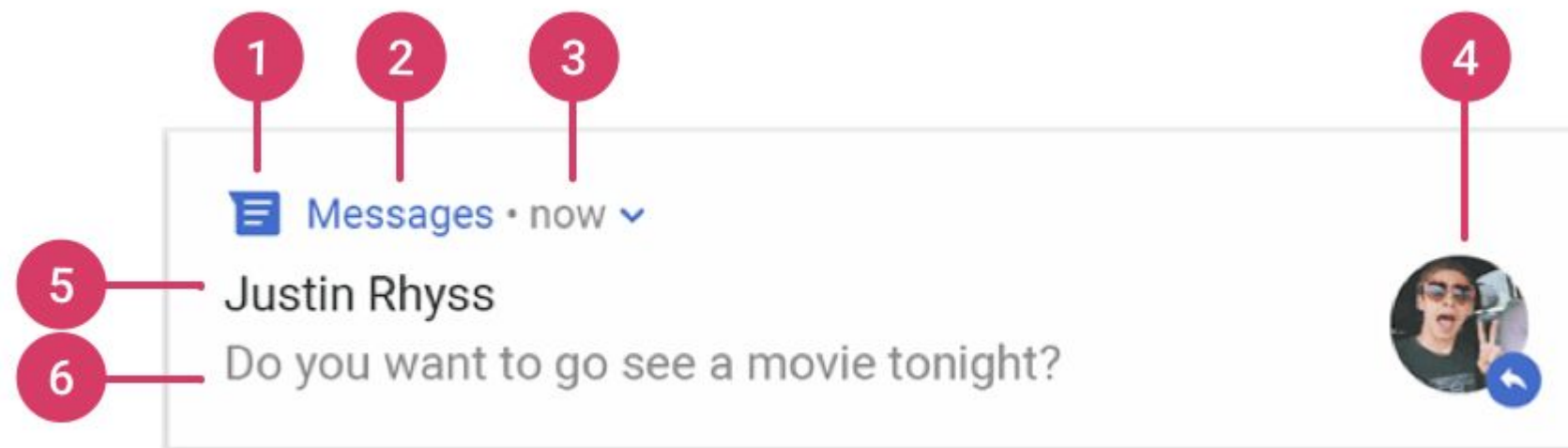


# Notificaciones en la barra de estado

- Las notificaciones constan de un icono, un título, una fecha y una descripción.
- El usuario puede expandir el área de notificaciones y ver el detalle de la notificación.
- Las notificaciones pueden tener acciones asociadas.



# Notificaciones. Anatomía



- ❶ Ícono pequeño (*small Icon*). Obligatorio
- ❷ Nombre de la App. Proporcionada por el sistema
- ❸ Marca de tiempo. Proporcionada por el sistema
- ❹ Ícono grande (*large Icon*). Opcional
- ❺ Título. Opcional
- ❻ Texto. Opcional

# Notificaciones en la barra de estado

- A partir de Android 8, API 26, las notificaciones se tienen que enviar a través de **canales** o *channels* específicos.
- Dichos canales pueden tener diferentes **niveles de importancia** o **prioridad**. Determinan el nivel de interrupción (visual y auditiva) de cada notificación.
- A partir de Android 13, API 33, se debe agregar en el manifiesto el permiso "android.permission.**POST\_NOTIFICATIONS**". Es un permiso de tiempo de ejecución.

# StatusBar - Actividad guiada

- Crear un nuevo proyecto desde Android Studio
- Mostrar una notificación en el StatusBar al presionar un botón
- Para crear una notificación:
  - a. Definir el contenido de la notificación
  - b. Crear un canal y definir la importancia
  - c. Mostrar la notificación

# StatusBar - Actividad guiada

- Definir un botón en el layout de la actividad y asociarle un manejador al click

```
<LinearLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Mostrar notificación"
        android:onClick="mostrarNotificacion" />
</LinearLayout>
```

# Actividad guiada - Definir el contenido de la notificación

- Agregar un ícono como recurso al proyecto.
  - *File > New > Image Asset*
- Definir
  - **Icon Type**: “ActionBar and TabIcons”
  - **Name**: “notificacion”
  - **AssertType**: Clip Art
  - **ClipArt**: seleccionar uno de los disponibles
- Next y finish

# Actividad guiada - Definir el contenido de la notificación

- La clase **NotificationCompat.Builder** permite configurar el contenido y el canal de una notificación.
- Agregar al método **mostrarNotificacion()** de la activity.

```
val builder = NotificationCompat.Builder(this,  
"CHANNEL_SEMINARIO_ANDROID")  
    .setSmallIcon(R.drawable.notificacion)  
    .setContentTitle("Mi Notificación")  
    .setContentText("¡Hola Mundo!")
```

Nombre del canal

small Icon creado

Título

Texto

# Actividad guiada - Crear un canal y definir la importancia

- Los canales se crean utilizando la clase **NotificationChannel**. Se especifica el ID, nombre y se le puede agregar una descripción para visualizarse desde el sistema

```
private fun createNotificationChannel() {  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
        val name = "Nombre del canal"  
        val descriptionText = "Canal para el envio de notificaciones en el  
Seminario"  
        val importance = NotificationManager.IMPORTANCE_DEFAULT  
        val channel = NotificationChannel("CHANNEL_SEMINARIO_ANDROID", name,  
importance).apply {  
            description = descriptionText  
        }  
  
        // Register the channel with the system  
        val notificationManager: NotificationManager =  
            getSystemService(Context.NOTIFICATION_SERVICE) as  
NotificationManager  
        notificationManager.createNotificationChannel(channel)  
    }  
}
```

Configuración del canal

Registrar el canal en el sistema

Invocar esta función en el método onCreate de la actividad



# Actividad guiada - Mostrar la notificación

- Se debe utilizar el método **NotificationManagerCompat.notify()**
- La notificación debe tener un ID asociado
- Agregar el siguiente código luego de la configuración de la notificación en el método **mostrarNotificacion()**

```
with(NotificationManagerCompat.from(this@MainActivity)) {
    if (ActivityCompat.checkSelfPermission(
        this@MainActivity,
        Manifest.permission.POST_NOTIFICATIONS
    ) != PackageManager.PERMISSION_GRANTED
    ) {
        return
    }
    notify(265, builder.build())
}
```

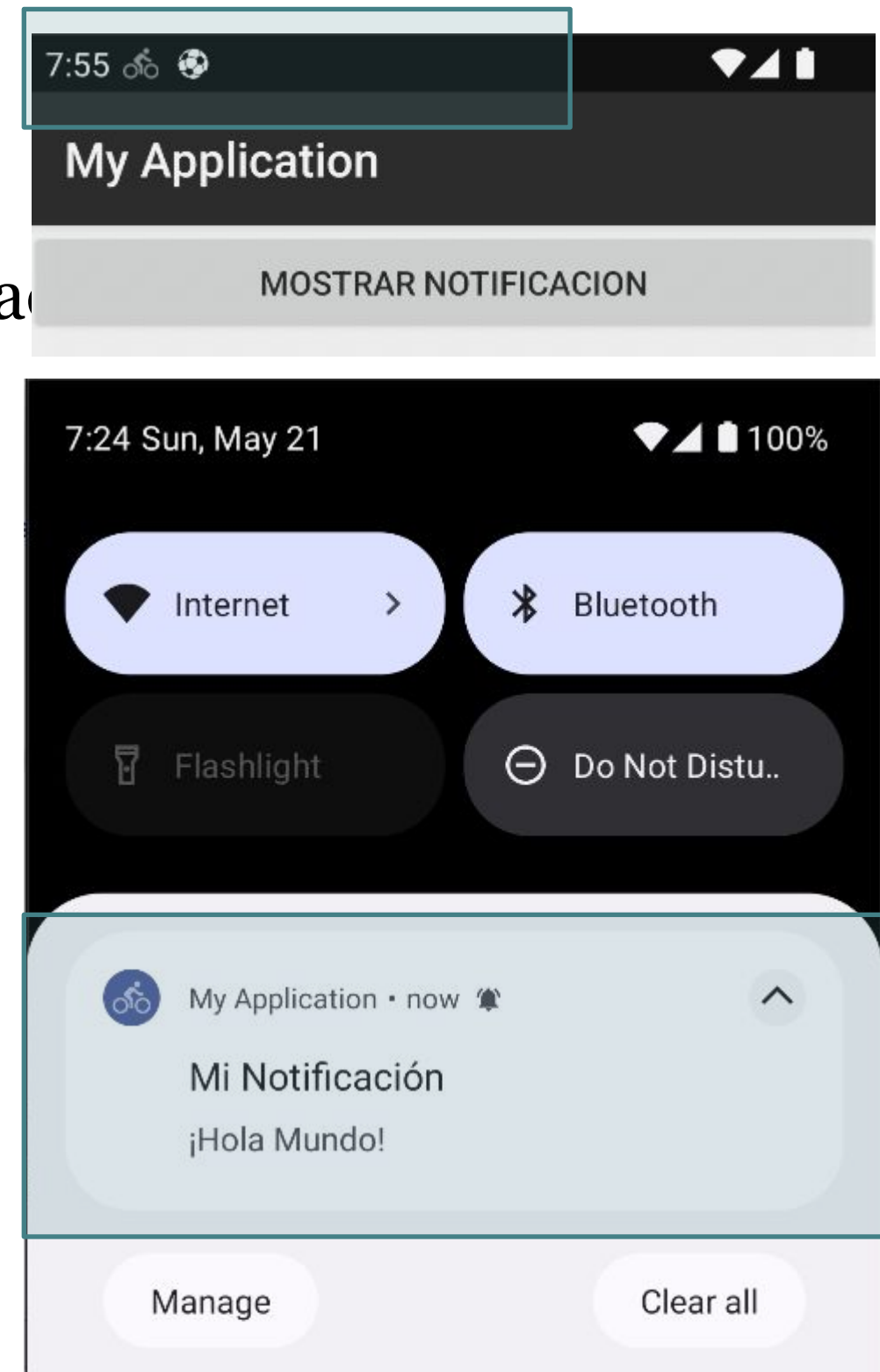
ID de la notificación

# Actividad guiada - Mostrar la notificación

- Verificar comportamiento en el emulador

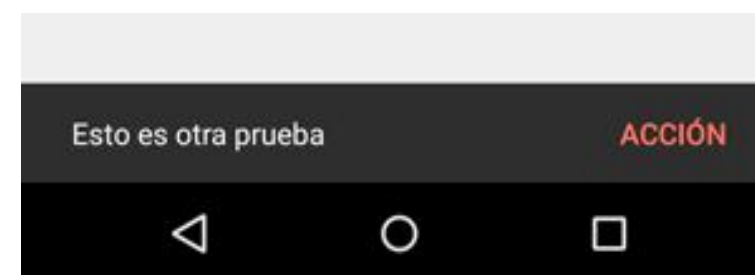
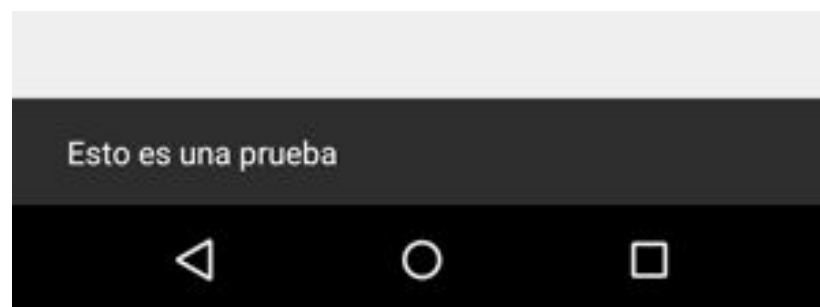
```
fun mostrarNotificacion(v: View) {
    val builder = NotificationCompat.Builder(this,
        "CHANNEL_SEMINARIO_ANDROID")
        .setSmallIcon(R.drawable.notification)
        .setContentTitle("Mi notificacion")
        .setContentText("¡Hola Mundo!")

    with(NotificationManagerCompat.from(this@MainActivity)) {
        if (ActivityCompat.checkSelfPermission(
            this@MainActivity,
            Manifest.permission.POST_NOTIFICATIONS
        ) !=
        PackageManager.PERMISSION_GRANTED
        ) {
            return
        }
        notify(265, builder.build())
    }
}
```



# Snackbar

- Permite mostrar información al usuario, de forma similar a un Toast.
- La notificación desaparece luego de un período de tiempo, similar al Toast.
- Requiere ser mostrada dentro de una actividad.
- Da la posibilidad de asociarle una acción
- Puede ser descartada haciendo swipe.



# Snackbar - Actividad guiada

- Agregar un botón en el layout de la actividad y asociarle un manejador al clic llamado **mostrarSnackbar**

```
fun mostrarSnackbar(v: View) {  
    Snackbar.make(v, "Esto es una prueba", Snackbar.LENGTH_LONG).show()  
}
```

- Probar en el emulador

# Snackbar - Actividad guiada

- Incorporar un acción al Snackbar

```
fun mostrarSnackbar(v: View) {  
    Snackbar.make(v, "Esto es una prueba", Snackbar.LENGTH_LONG)  
        .setAction("Acción") {  
            Log.i("Snackbar", "Se invocó la acción")  
        }  
    .show() ;  
}
```

# Snackbar - Actividad guiada

- Incorporar un acción al Snackbar

```
fun mostrarSnackbar(v: View) {  
    Snackbar.make(v, "Esto es una prueba", Snackbar.LENGTH_LONG)  
        .setAction("Acción") {  
            Log.i("Snackbar", "Se invocó la acción")  
        }  
    .show();  
}
```

Define una acción en el Snakbar

Texto a mostrar con el nombre de la acción

Código a ejecutar cuando se presiona sobre la acción

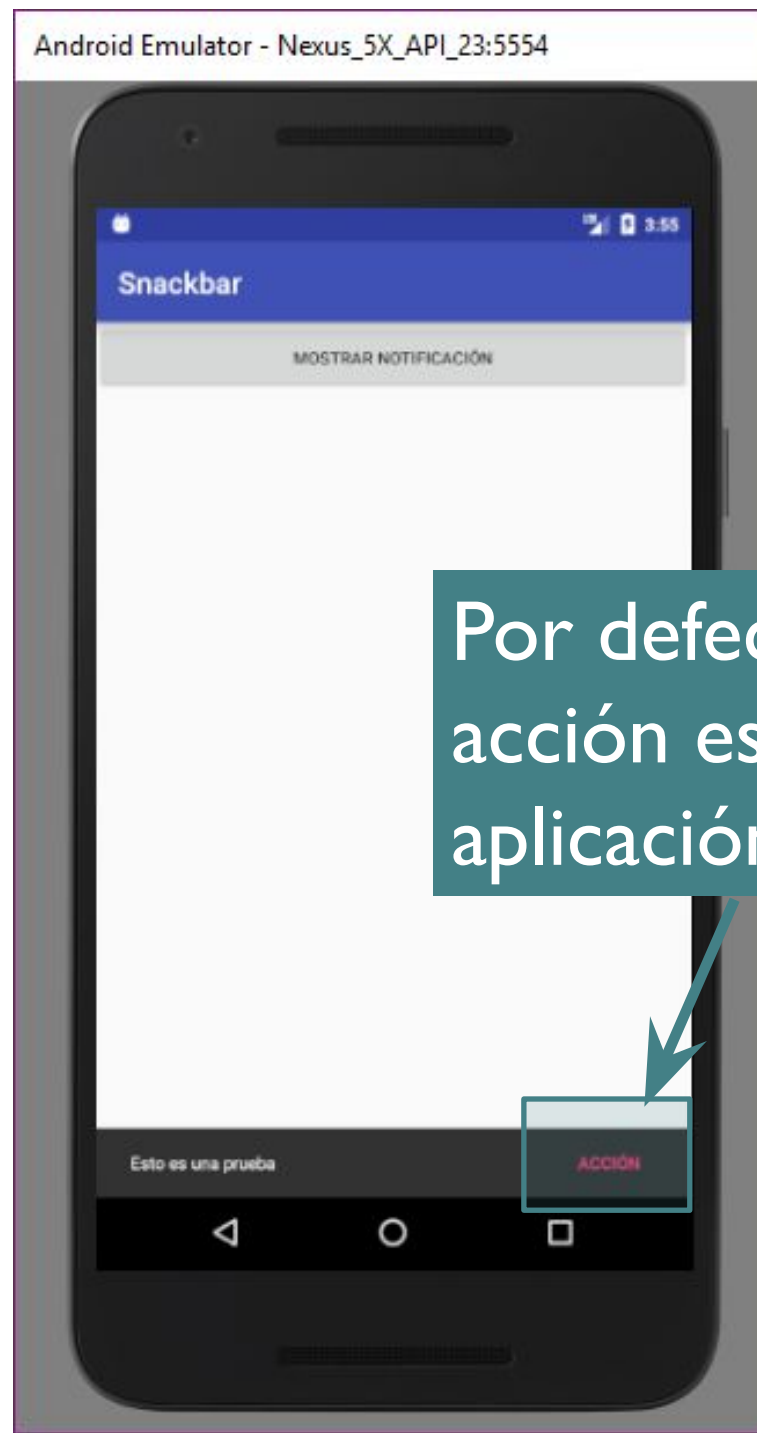
# Snackbar - Actividad guiada

- Probar la aplicación en el emulador



# Snackbar - Actividad guiada

- Probar la aplicación en el emulador



Por defecto el color de la acción es del tema de la aplicación



# Snackbar - Actividad guiada

- Cambiar el color de la acción

```
fun mostrarSnackbar(v: View) {  
    Snackbar.make(v, "Esto es una prueba", Snackbar.LENGTH_LONG)  
        .setAction("Acción") {  
            Log.i("Snackbar", "Se invocó la acción")  
        }  
        .setActionTextColor(Color.GREEN)  
        .show();  
}
```

# Snackbar - Actividad guiada

- Es posible incorporar la animación de descartar la notificación arrastrándola hacia la derecha
- Solo hay que definir un nuevo elemento raíz en el layout de la actividad

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_height="match_parent"
        android:layout_width="match_parent">
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Mostrar notificación"
            android:onClick="mostrarNotificacion" />
        </LinearLayout>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

# Snackbar - Actividad guiada

- Probar la aplicación en el emulador

Android Emulator - Nexus\_5X\_API\_23:5554

