



**TECNOLÓGICO DE ESTUDIOS SUPERIORES DE  
CHALCO**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**ASIGNATURA:**

**DESARROLLO DE APLICACIONES PARA EL COMERCIO  
ELECTRÓNICO**



**Integrantes:**

**Mendoza Colín Julio Cesar**

**Grupo: 4902**

**Profesor:**

**DANIEL GUADALUPE MORENO BECERRIL**

## Contenido

Introducción .....	3
1. Menú Textuales .....	4
2. Barra y flecha hacia atrás .....	7
3 Menú contextual .....	9
3.Toasd y Alert dialog .....	10
4. Selector de fecha y hora .....	13
Conclusiones.....	17

## Introducción

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar; los nuevos terminales ofrecen unas capacidades similares a un ordenador personal, lo que permite que puedan ser utilizados para leer el correo o navegar por Internet. Pero, a diferencia de un ordenador, un teléfono móvil siempre está en el bolsillo del usuario. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en afirmar que el nuevo ordenador personal del siglo xxi será un terminal móvil.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y está teniendo una importante aceptación tanto por parte de los usuarios como por parte de la industria. En la actualidad se ha convertido en la alternativa dominante frente a otras plataformas como iPhone o Windows Phone, a lo largo de este capítulo veremos las características de Android que lo hacen diferente de sus competidores. Se explicará también cómo instalar y trabajar con el entorno de desarrollo (Android Studio).

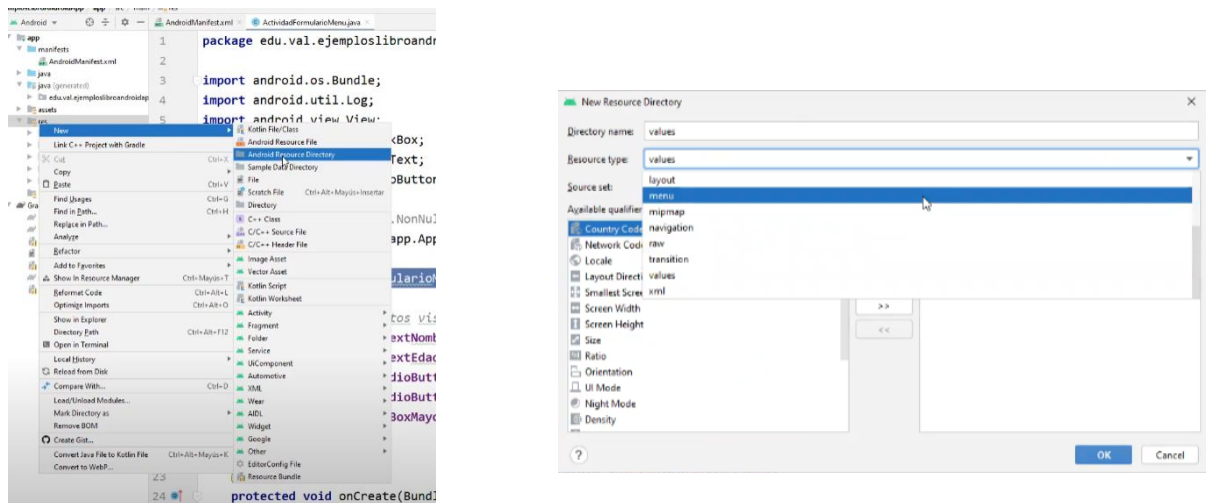
Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con C++ y NDK

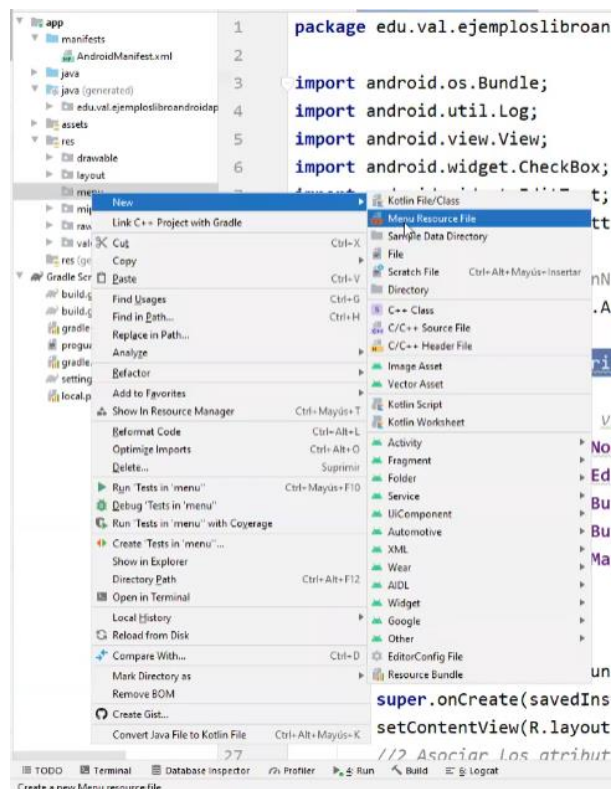
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine
- 

## 1. Menú Textuales

Comenzamos creando un nuevo directorio



Creamos un nuevo menú



Vemos y ya aparece el menú



Y ahora pasamos a hacer las opciones

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado una opción del menú");
    switch (item.getItemId())
    {
        case R.id.opcionborrar:
            Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado una opción del menú");
            break;
        case R.id.opcionsalir:
            Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado una opción del menú");
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Probamos de nuevo y esta vez tocamos los menú y vemos por consola si funciona

```
17:49:40.696 7508-7508/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado una opción del menú
17:49:40.697 7508-7508/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado la opción limpiar
17:49:43.953 7508-7508/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado una opción del menú
17:49:43.953 7508-7508/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado la opción salir
```

Por último creamos los eventos de los botones

```
private void limpiarFormulario ()
{
    this.editTextNombre.setText("");
    this.editTextEdad.setText("");
    this.radioButtonHombre.setChecked(false);
    this.radioButtonMujer.setChecked(false);
    this.checkBoxMayorEdad.setChecked(false);
}

private void salir ()
{
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Saliendo ...");
    this.finish();
}
```

Probamos y estos ya deben de funcionar

The image displays two screenshots of an Android application interface. The left screenshot shows a form with a green Android logo at the top. Below the logo are two text input fields: the first contains 'Vale' and the second contains '37'. Below these fields are two radio buttons: 'Hombre' (selected) and 'Mujer'. Below the radio buttons is a checked checkbox labeled 'Soy mayor de Edad'. At the bottom of the form is a purple button labeled 'ACEPTAR'. The right screenshot shows a 'Form Menu' with a purple header bar containing the text 'Form Menu' and a 'Salir' button. Below the header bar is a green Android logo. Below the logo are two text input fields: the first is labeled 'Intro nombre' and the second is labeled 'Intro edad'. Below these fields are two radio buttons: 'Hombre' and 'Mujer'.

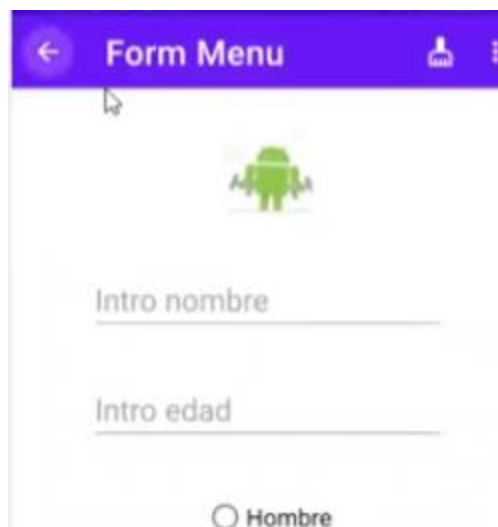
## 2. Barra y flecha hacia atrás

En el método en create lo definimos

```
private void initActivity ()
{
    this.editTextNombre = findViewById(R.id.editTextTextPersonName);
    this.editTextEdad = findViewById(R.id.editTextNumber);
    this.radioButtonHombre = findViewById(R.id.radioButtonHombre);
    this.radioButtonMujer = findViewById(R.id.radioButtonMujer);
    this.checkBoxMayorEdad = findViewById(R.id.checkBox);

    //ocultarBarra
    getSupportActionBar().hide();
    //mostarBarrar
    getSupportActionBar().show();
    //dibujar la flecha hacia atrás
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
}
```

Ejecutamos y vemos la implementación



Para darle función

Esto se declara en la función de opciones

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado una opción del menú");
    switch (item.getItemId())
    {
        case R.id.opcionborrar:
            Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado la opción limpiar");
            limpiarFormulario ();
            break;
        case R.id.opcionsalir:
            Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado la opción salir");
            salir ();
            break;
        case android.R.id.home:
            Log.d(MainActivity.ETIQUETA_LOG, msg: "Se ha tocado la opción salir con la flecha");
            salir ();
            break;
    }
}
```

Corremos y vemos en la consola

```
18-08 18:23:04.124 11225-11225/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado una opción del menú
18-08 18:23:04.124 11225-11225/edu.val.ejemploslibroandroidapp D/AppEjemplos: Se ha tocado la opción salir con la flecha
18-08 18:23:04.124 11225-11225/edu.val.ejemploslibroandroidapp D/AppEjemplos: Saliendo ...
```



### 3 Menú contextual

Creamos un objeto con el logo para que este despliegue el menú que vamos a realizar

```
private void initActivity ()
{
    this.editTextNombre = findViewById(R.id.editTextTextPersonName);
    this.editTextEdad = findViewById(R.id.editTextNumber);
    this.radioButtonHombre = findViewById(R.id.radioButtonHombre);
    this.radioButtonMujer = findViewById(R.id.radioButtonMujer);
    this.checkBoxMayorEdad = findViewById(R.id.checkBox);

    //ocultarBarra
    getSupportActionBar().hide();
    //mostarBarrar
    getSupportActionBar().show();
    //dibujar la flecha hacia atrás
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    //dibujamos el menú contextual
    this.imagenAndroid = findViewById(R.id.imageView);
    registerForContextMenu(this.imagenAndroid);
}
```

Creamos el menú en una función

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(Menu.NONE, itemId: 3, order: 3, title: "BORRAR");
}
```

Y por último invocamos la opción para que desaparezca la imagen

```
@Override
public boolean onContextItemSelected(@NonNull MenuItem item) {
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Opcion tocada = " + item.getItemId());
    this.imagenAndroid.setVisibility(View.INVISIBLE);

    return super.onContextItemSelected(item);
}
```




### 3. Toasd y Alert dialog

En la función de limpiar formulario simplemente colocamos el toasd

```
private void limpiarFormulario ()
{
    this.editTextNombre.setText("");
    this.editTextEdad.setText("");
    this.radioButtonHombre.setChecked(false);
    this.radioButtonMujer.setChecked(false);
    this.checkBoxMayorEdad.setChecked(false);
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Formulario limpio");

    Toast toast = Toast.makeText(context: this, text: "Formulario limpiado", Toast.LENGTH_
    toast.show();
}
```

Cargamos y damos limpiar

 **Form Menu**  



☐ Hombre

☐ Mujer

☐ Soy mayor de Edad

**ACEPTAR**

Formulario limpiado

Pasamos al alert dialog, este lo colocamos en la función de salir

```
AlertDialog ad = new AlertDialog.Builder( context: this).create();  
//this es el contexto  
ad.setTitle("Salir");  
ad.setMessage("¿Desea salir?");  
//y los botones con sus respectivas opciones ya programadas  
ad.setButton(AlertDialog.BUTTON_NEGATIVE, text: "NO",  
    new DialogInterface.OnClickListener(){  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            dialog.cancel();  
        }  
    });  
ad.setButton(AlertDialog.BUTTON_POSITIVE, text: "SÍ",  
    new DialogInterface.OnClickListener(){  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            ActividadFormularioMenu.this.finish();  
        }  
    });  
ad.show();
```

Probamos y damos salir



## 4. Selector de fecha y hora

Creamos dos edit text



Programamos evento para el llamado foco para cada caja pero antes creamos las actividades

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_actividad_seleccion_fecha_y_hora);

    this.caja_fecha = findViewById(R.id.cajaFecha);
    this.caja_hora = findViewById(R.id.cajaHora);

    this.caja_fecha.setOnFocusChangeListener(this);
    this.caja_hora.setOnFocusChangeListener(this);
}
```

Pasamos a la función del foco

```
@Override
public void onFocusChange(View v, boolean hasFocus) {
    if (hasFocus)
    {
        switch (v.getId())
        {
            case R.id.cajaFecha:
                Log.d(MainActivity.ETIQUETA_LOG, msg: "FECHA Toca lanzar el Calendario");
                break;
            case R.id.cajaHora:
                Log.d(MainActivity.ETIQUETA_LOG, msg: "HORA Toca lanzar el Reloj");
                break;
        }
    }
}
```

Para el calendario creamos una nueva clase e importamos la librería y creamos con el asistente la función datePickerDialog

```
import java.util.Calendar;

public class SeleccionCalendario extends DialogFragment implements DatePickerDialog.On
```

En

```
@NonNull
@Override
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    //personalizar el calendario
    Dialog calendario = null;
    int anio, mes, dia;
    Calendar calendario actual:

@Override
public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Fecha seleccionada " + dayOfMonth + " / " + month + " / "+ year);
}

mes = calendario_actual.get(Calendar.MONTH);
dia = calendario_actual.get(Calendar.DATE);

calendario = new DatePickerDialog(getActivity(), listener: this, anio, mes, dia
```

Y (

```
return calendario;
}

switch (v.getId())
{
    case R.id.cajaFecha:
        Log.d(MainActivity.ETIQUETA_LOG, msg: "FECHA Toca lanzar el Calendario");
        DialogFragment fragmento_calendario = new SeleccionCalendario();
        fragmento_calendario.show(getSupportFragmentManager(), tag: "CALENDARIO");
        break;
}
```



Pasamos a la parte de la hora de igual manera creamos un clase y colocamos la función

```
public class SeleccionHora extends DialogFragment implements TimePickerDialog.OnTimeSetListener {

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        Dialog reloj= null;
        int hora, minutos;
        Calendar calendar;

        calendar = Calendar.getInstance();
        hora = calendar.get (Calendar.HOUR_OF_DAY);
        minutos = calendar.get (Calendar.MINUTE);

        reloj = new TimePickerDialog(getActivity(), listener: this, hora, minutos, is24HourView: true);

        return reloj;
    }
}
```

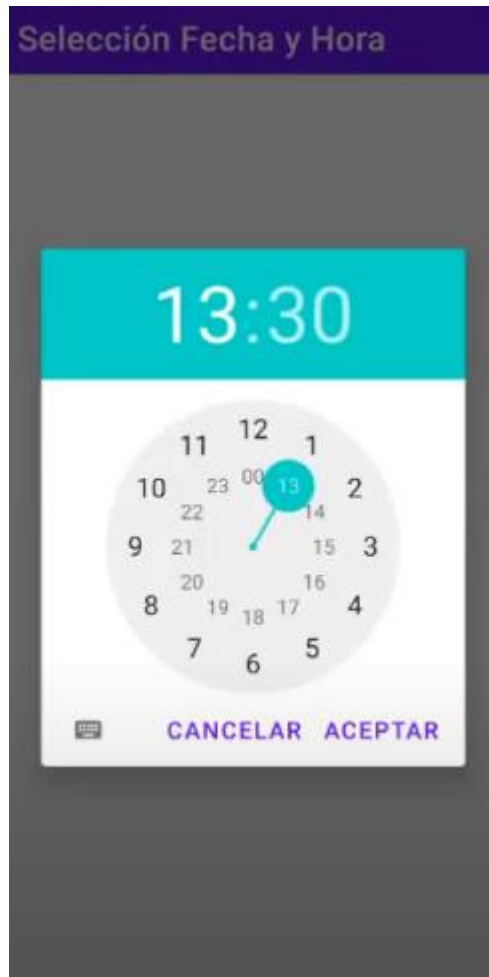
Y con el asiente creamos la función TimePicker con su mensaje

```
@Override
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    Log.d(MainActivity.ETIQUETA_LOG, msg: "Hora elegida= " + hourOfDay+"":"+minute);
}
```

Y lo invocamos en la caja hora

```
case R.id.cajaHora:
    Log.d(MainActivity.ETIQUETA_LOG, msg: "HORA Toca lanzar el Reloj");
    DialogFragment fragmento_reloj = new SeleccionHora();
    fragmento_reloj.show(getSupportFragmentManager(), tag: "RELOJ");
    break;
```

Seleccionamos y vemos que sale la hora





## **Conclusiones**

El programar aplicaciones Android es muy importante, es una plataforma que actualmente domina el mercado de los dispositivos móviles, debido a su carácter libre y su desarrollo abierto, no controlado por una única empresa. Android ofrece un entorno de desarrollo que facilita la implementación de aplicaciones de manera ágil y eficiente, aprovechando al máximo las características de cada dispositivo móvil. Con esta guía se puede adquirir un conocimiento profundo y detallado de la estructura y del funcionamiento de Android, sirviendo así, como herramienta para fomentar el desarrollo de aplicaciones que aprovechan al máximo el Sistema Operativo.