Curso de Desarrollo Móvil en Android

Curso Android

Bienvenido alumno

Sesiones

Completadas

4

Exámenes

3

Tareas

3

José María Cruz Parada
chema.cruz.p@gmail.com

- Resumen de curso
- Reglamento
- Temario
- Sesiones

O

Exámenes
3

Tareas
3

Sesiones

Sesión 1

Fundamentos de Java

Sesión 2

Fundamentos de Android

Sesión 3

Ciclo de vida del Activity

Sesión 4

Componentes gráficos

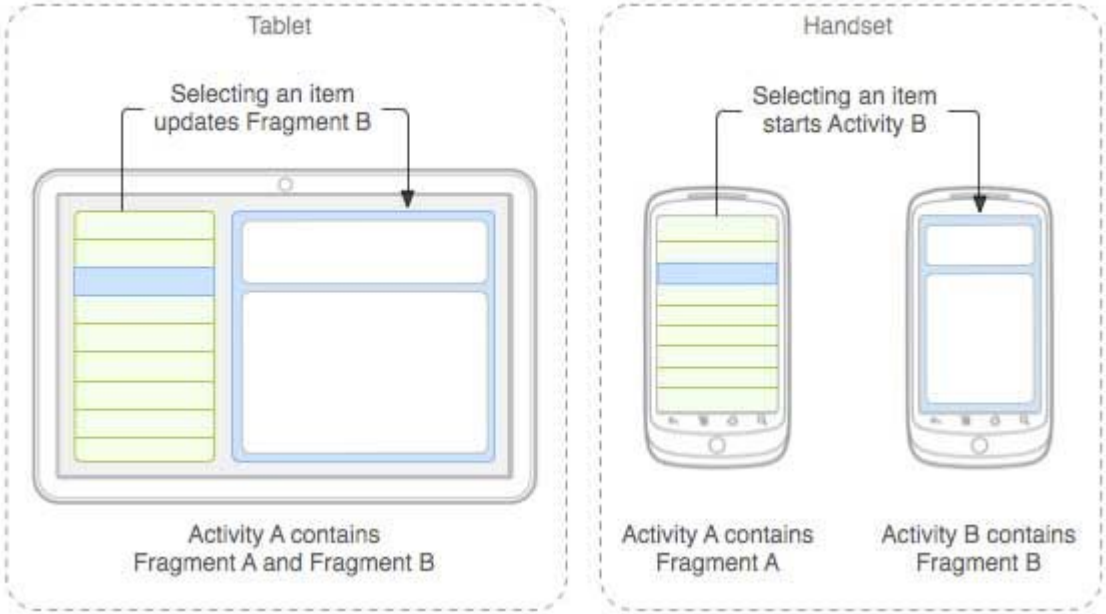
Sesión 5

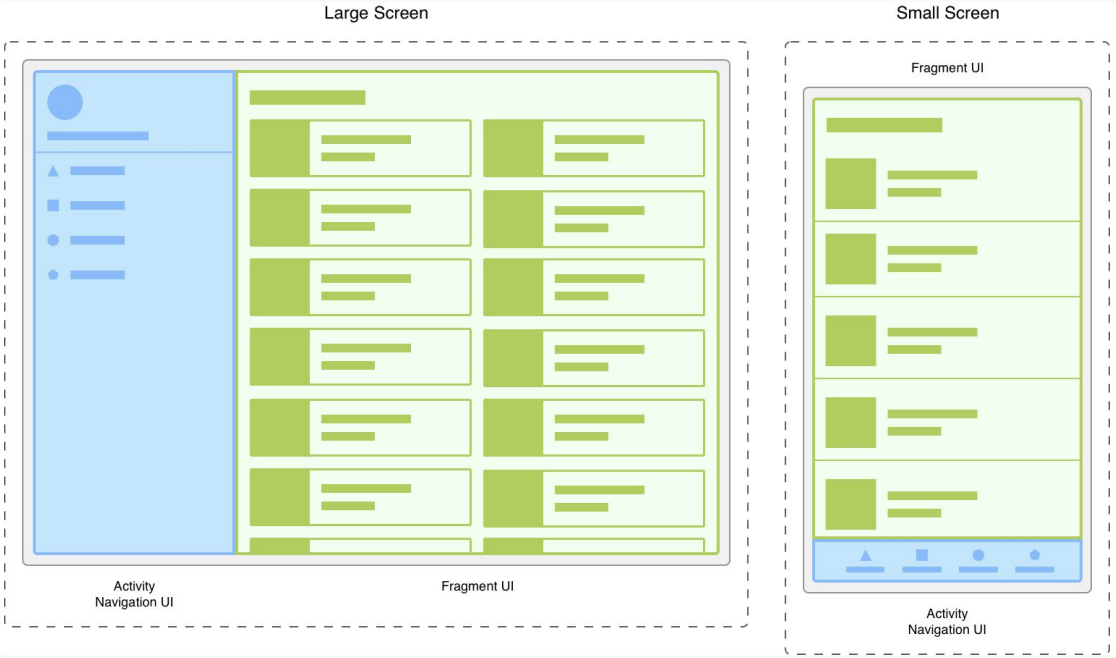
Fragments

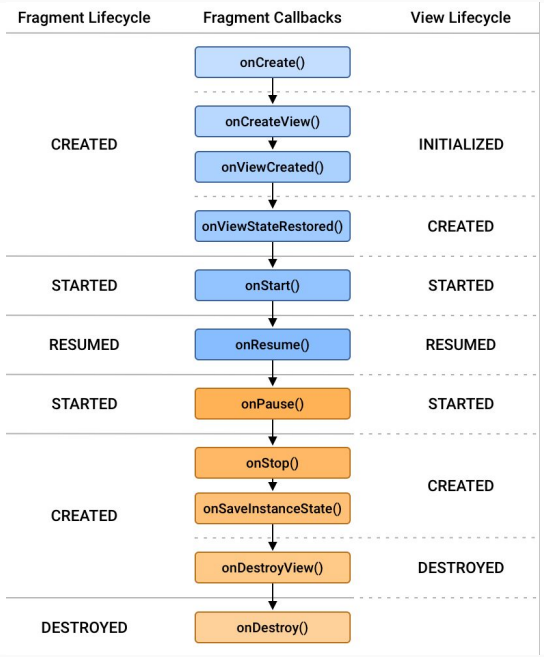
Sesión 6

Servicios Web

¿Qué es un Fragment?


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B


Large Screen
Activity
Navigation UI
Fragment UI
Small Screen
Fragment UI
Activity
Navigation UI

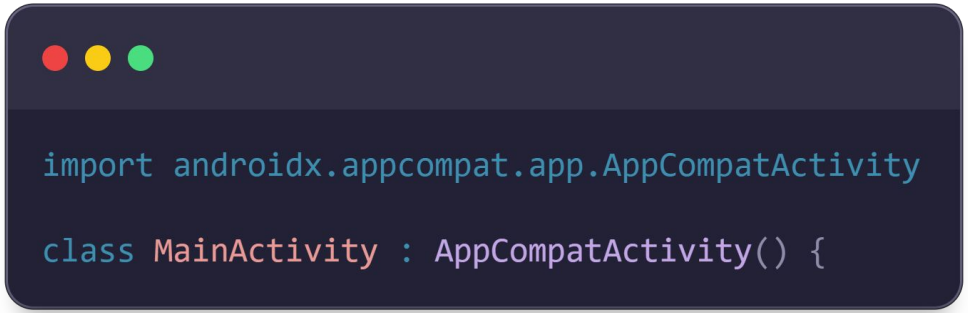

Fragment Lifecycle
Fragment Callbacks
View Lifecycle
onCreate()
onCreateView()
onViewCreated()
CREATED
INITIALIZED
onViewStateRestored()
CREATED
STARTED
onStart()
STARTED
RESUMED
onResume()
RESUMED
STARTED
onPause()
STARTED
onStop()
onSaveInstanceState()
CREATED
CREATED
onDestroyView()
DESTROYED
DESTROYED
onDestroy()

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fragment Example" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me" />

</LinearLayout>
```

```kotlin
class ExampleFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflar el diseño del fragmento
        return inflater.inflate(R.layout.fragment_example, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // Configurar la lógica del fragmento
        val textView = view.findViewById<TextView>(R.id.textView)
        val button = view.findViewById<Button>(R.id.button)

        button.setOnClickListener {
            textView.text = "Button Clicked"
        }
    }
}
```

```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="vertical"
    android:id="@+id/activityLayout"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragmentContainer"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

</LinearLayout>
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Agregar el fragmento al contenedor
    val fragment = ExampleFragment()
    supportFragmentManager.beginTransaction()
        .add(R.id.fragmentContainer, fragment)
        .commit()
}
```



```
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
```

¿Qué es ViewPager?

```
<androidx.viewpager.widget.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Página 1" />

</LinearLayout>
```

```
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentPagerAdapter

class ViewPagerAdapter(fm: FragmentManager) : FragmentPagerAdapter(fm) {

    override fun getCount(): Int {
        return 3 // Número total de páginas del ViewPager
    }

    override fun getItem(position: Int): Fragment {
        return PageFragment.newInstance(position + 1)
    }
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val viewPager: ViewPager = findViewById(R.id.viewPager)
    val adapter = ViewPagerAdapter(supportFragmentManager)
    viewPager.adapter = adapter
}
```

```
class PageFragment : Fragment() {

    companion object {
        private const val ARG_PAGE_NUMBER = "page_number"

        fun newInstance(pageNumber: Int): PageFragment {
            val fragment = PageFragment()
            val args = Bundle()
            args.putInt(ARG_PAGE_NUMBER, pageNumber)
            fragment.arguments = args
            return fragment
        }
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.fragment_page, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val textView : TextView

        val pageNumber = arguments?.getInt(ARG_PAGE_NUMBER)
        textView = view.findViewById(R.id.textView)
        textView.text = "Página $pageNumber"
    }
}
```

Permisos

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.CAMERA"
        tools:ignore="PermissionImpliesUnsupportedChromeOsHardware" />
```

```
private fun requestCameraPermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
        // Permiso de cámara otorgado
    } else {
        // Permiso de cámara no otorgado, solicitarlo
        ActivityCompat.requestPermissions(this, arrayOf(Manifest.permission.CAMERA), CAMERA_PERMISSION_CODE)
    }
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    requestCameraPermission()
}
```

```kotlin
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)

    when (requestCode) {
        CAMERA_PERMISSION_CODE -> {
            if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permiso de cámara otorgado
            } else {
                // Permiso de cámara denegado, volver a solicitarlo
            }
        }
    }
}
```

¿Qué entiendo por almacenamiento?

Tipos de almacenamiento

```
// Almacenamiento Interno
val nombreArchivoAI = "miarchivo.txt"
val datosAI = "Contenido del archivo en almacenamiento interno."
```

```kotlin
// Almacenamiento Interno
fun escribirDatosAlmacenamientoInterno(nombreArchivo: String, datos: String) {
    val archivo = File(this.filesDir, nombreArchivo)
    archivo.writeText(datos)
}

fun leerDatosAlmacenamientoInterno(nombreArchivo: String): String {
    val archivo = File(this.filesDir, nombreArchivo)
    return archivo.readText()
}
```

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Almacenamiento Interno
    escribirDatosAlmacenamientoInterno(nombreArchivoAI, datosAI)
}
```

```
override fun onResume() {
    super.onResume()

    // Almacenamiento Interno
    val contenido = leerDatosAlmacenamientoInterno(nombreArchivoAI)
    Toast.makeText(this,contenido,Toast.LENGTH_LONG).show()
}
```

```
// Almacenamiento Externo
val nombreArchivoAE = "miarchivoAE.txt"
val datosAE = "Contenido del archivo en almacenamiento externo."
```

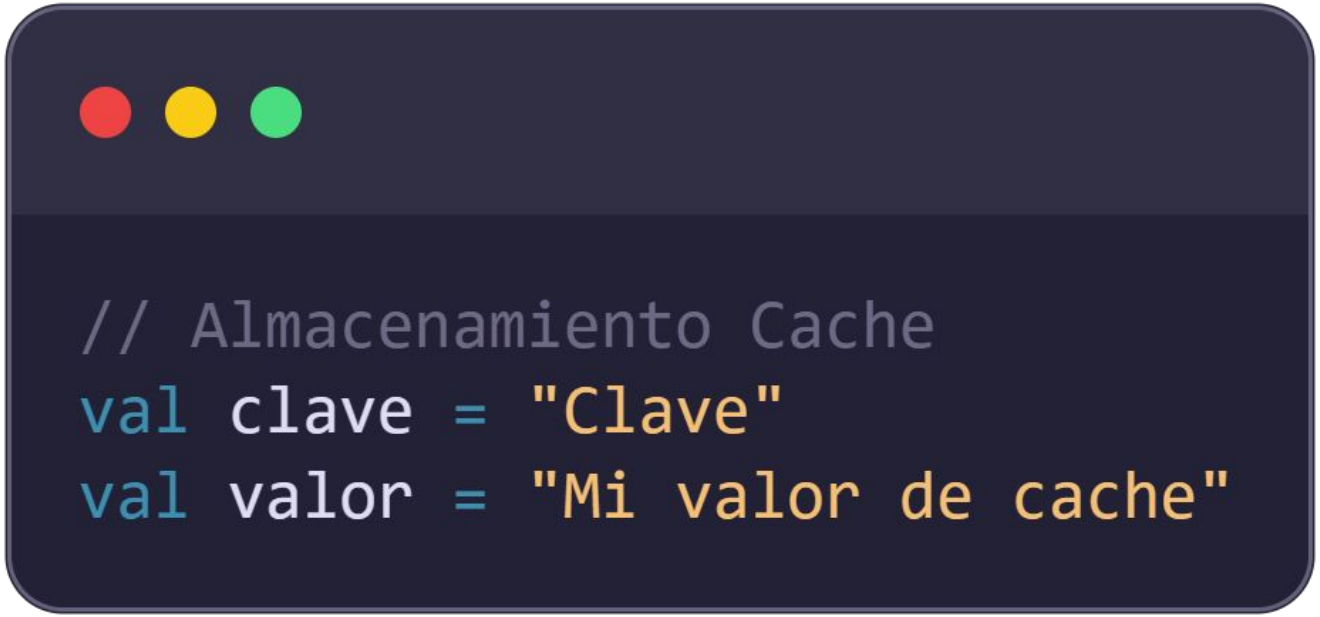
```kotlin
// Almacenamiento Externo
private fun escribirDatosAlmacenamientoExterno() {
    val estado = isExternalStorageWritable()
    if (estado) {
        val directorio = getExternalFilesDir(null)
        val archivo = File(directorio, nombreArchivoAE)
        try {
            FileOutputStream(archivo).use {
                it.write(datosAE.toByteArray())
            }
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}
```

```kotlin
// Almacenamiento Externo
private fun leerDatosAlmacenamientoExterno(): String {
    val estado = isExternalStorageReadable()
    if (estado) {
        val directorio = getExternalFilesDir(null)
        val archivo = File(directorio, nombreArchivoAE)
        var fileInputStream = FileInputStream(archivo)
        var inputStreamReader: InputStreamReader = InputStreamReader(fileInputStream)
        val bufferedReader: BufferedReader = BufferedReader(inputStreamReader)
        val stringBuilder: StringBuilder = StringBuilder()
        var text: String? = null
        while ({ text = bufferedReader.readLine(); text }() != null) {
            stringBuilder.append(text)
        }
        fileInputStream.close()

        return stringBuilder.toString()
    }

    return ""
}
```

```kotlin
// Almacenamiento Externo
private fun isExternalStorageWritable(): Boolean {
    val estado = Environment.getExternalStorageState()
    return Environment.MEDIA_MOUNTED == estado
}

private fun isExternalStorageReadable(): Boolean {
    val estado = Environment.getExternalStorageState()
    return Environment.MEDIA_MOUNTED == estado || Environment.MEDIA_MOUNTED_READ_ONLY == estado
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Almacenamiento Externo
    escribirDatosAlmacenamientoExterno()
}
```

```
override fun onResume() {
    super.onResume()

    // Almacenamiento Externo
    val contenido = leerDatosAlmacenamientoExterno()
    Toast.makeText(this,contenido,Toast.LENGTH_LONG).show()
}
```



```
// Almacenamiento Cache
val clave = "Clave"
val valor = "Mi valor de cache"
```

```kotlin
// Almacenamiento Cache
fun escribirDatosAlmacenamientoCache(context: Context, clave: String, valor: String) {
    val sharedPreferences = context.getSharedPreferences("cache", Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()
    editor.putString(clave, valor)
    editor.apply()
}

fun leerDatosAlmacenamientoCache(context: Context, clave: String): String? {
    val sharedPreferences = context.getSharedPreferences("cache", Context.MODE_PRIVATE)
    return sharedPreferences.getString(clave, null)
}
```

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Almacenamiento Cache
    escribirDatosAlmacenamientoCache(this, clave, valor)
}
```

```
override fun onResume() {
    super.onResume()

    // Almacenamiento Cache
    val contenido = leerDatosAlmacenamientoCache(this, clave)
    Toast.makeText(this,contenido,Toast.LENGTH_LONG).show()
}
```

```
// Almacenamiento SQLite
val databaseHelper = DatabaseHelper(this)
```

```
package com.example.storageexample

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "mydatabase.db"
        private const val DATABASE_VERSION = 1

        // Define el nombre de la tabla y las columnas
        private const val TABLE_NAME = "contacts"
        private const val COLUMN_ID = "_id"
        private const val COLUMN_NAME = "name"
        private const val COLUMN_PHONE = "phone"
    }
```

```
override fun onCreate(db: SQLiteDatabase) {
    // Crea la tabla cuando se crea la base de datos
    val createTable = "CREATE TABLE $TABLE_NAME " +
            "($COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "$COLUMN_NAME TEXT, " +
            "$COLUMN_PHONE TEXT)"
    db.execSQL(createTable)
}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
    // Maneja la actualización de la base de datos si es necesario
    db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
    onCreate(db)
}
```

```kotlin
// Agrega un nuevo contacto a la base de datos
fun addContact(name: String, phone: String) {
    val db = writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_NAME, name)
        put(COLUMN_PHONE, phone)
    }
    db.insert(TABLE_NAME, null, values)
    db.close()
}

// Obtiene todos los contactos de la base de datos
fun getAllContacts(): List<Contact> {
    val contactList = mutableListOf<Contact>()
    val db = readableDatabase
    val selectQuery = "SELECT * FROM $TABLE_NAME"
    val cursor = db.rawQuery(selectQuery, null)

    if (cursor.moveToFirst()) {
        do {
            val id = cursor?.getColumnIndex(COLUMN_ID)?.let { cursor.getInt(it) }
            val name = cursor?.getColumnIndex(COLUMN_NAME)?.let { cursor.getString(it) }
            val phone = cursor?.getColumnIndex(COLUMN_PHONE)?.let { cursor.getString(it) }
            val contact = Contact(id!!, name!!, phone!!)
            contactList.add(contact)
        } while (cursor.moveToNext())
    }

    cursor.close()
    //db.close()
    return contactList
}
```

```
data class Contact(val id: Int, val name: String, val phone: String)
```

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Almacenamiento SQLite
    databaseHelper.addContact("Jose", "123456789")
}
```

```
override fun onResume() {
    super.onResume()

    // Almacenamiento SQLite
    val contenido = databaseHelper.getAllContacts()
    for (contact in contenido) {
        Toast.makeText(this,"ID: ${contact.id}, Name: ${contact.name}, Phone: ${contact.phone}",Toast.LENGTH_LONG).show()
    }
}
```

Sesiones

Sesión 1

Fundamentos de Java

Sesión 2

Fundamentos de Android

Sesión 3

Ciclo de vida del Activity

Sesión 4

Componentes gráficos

Sesión 5

Fragments

Sesión 6

Servicios Web

Curso de Desarrollo Móvil en Android