Curso de Desarrollo Móvil en Android

Curso Android

Bienvenido alumno

Sesiones

Completadas	Exámenes	Tareas
0	0	0

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

- Resumen de curso
- Reglamento
- Temario
- Sesiones

O

Exámenes
0

Tareas
0

Mi perfil

José María Cruz Parada

chema.cruz.p@gmail.com

@chema_cruz_p

Escolaridad

Experiencia laboral

Conocimientos

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

- Resumen de curso
- Reglamento
- Temario
- Sesiones

- Escolaridad
- Experiencia laboral
- Conocimientos

Reglamento


Secretaría de Desarrollo Económico Sustentable
Escuela Nacional de Estudios Superiores
IECAGto
Instituto Estatal de Capacitación
Reglas para las sesiones virtuales
1 Conéctate directamente desde la plataforma
2 Comunica a los docentes si tienes complicaciones de audio o video
3 Entra con video y con el micrófono apagado
4 Utiliza el chat para la sesión de preguntas y respuestas o la herramienta "Levanta la mano"
5 Empieza y termina la reunión en el tiempo establecido
6 Deja siempre tu cámara encendida para validar tu prescencia
7 Hay pase de lista
8 Las tareas se suben a la plataforma directamente
9 El Material de las presentaciones está en la plataforma
10 Solo tienes derecho a una falta para emitir tu certificado.
Desarrollo Móvil en Android
CURSO ONLINE
HORARIOS
sábados de 8:30 a.m. a 1:30 p.m.
Dentro de los horarios de clase se tendrán 2 breaks de 15 minutos

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

- Resumen de curso
- Reglamento
- Temario
- Sesiones

- Escolaridad
- Experiencia laboral
- Conocimientos

Ficha Técnica del Curso de Desarrollo Móvil en Android

Contexto

El desarrollo de Software para Dispositivos Móviles Inteligentes (APPs) es cada vez más demandado por la industria, ya que se aprovecha las características de portabilidad y facilidad de uso para generar aplicaciones altamente productivas que elevan la competitividad de las empresas.

Objetivo del curso

Adquirir los conocimientos para el Desarrollo de Aplicaciones Móviles basadas en Sistemas Operativos Android, que permitan elevar las competencias laborales de los participantes e incorporarse a un empleo como Programador Móvil en Android.

Resultado de aprendizaje

Al finalizar el curso, el participante será capaz de desarrollar aplicaciones móviles basadas en sistema operativo Android, podrá construir una interfaz gráfica y hacer uso de activities para mejorar la experiencia de navegación, de igual forma, podrá garantizar la persistencia de datos con SQLite Room y utilizar servicios web y firebase para dar funcionalidad a las aplicaciones, finalmente, aprenderá a usar herramientas de trabajo colaborativo y de control de versiones, que le permitan integrarse a un equipo de trabajo y ambiente productivo como desarrollador Android.

Estructura del curso

El curso se llevará a cabo en 8 sesiones virtuales de 5 horas cada una, con una duración total de 40 horas.

A quiénes está dirigido

El curso está dirigido a profesionistas o estudiantes que se encuentren próximos a egresas y que tengan formación y conocimientos sólidos en Tecnologías de la Información y/o Desarrollo de Software.

Requerimientos previos

- ✔ Conocimientos en Tecnologías de Información y Programación.
- ✔ Uso de computadora portátil.
- ✔ Tener una cuenta de correo electrónico para recibir mensajes.

Temario

Temas	Contenido	Duración	Sesión
1. **Fundamentos de Java**	1. Entorno de desarrollo 2. Introducción a Java 2.1. Tipos de datos 2.2. Variables 2.3. Casting 2.4. Arreglos 2.5. Funciones 2.6. Sentencias de control 2.7. Funciones 2.8. Clases 3. Programación Orientada a Objetos 3.1. Instancias 3.2. Modificadores de Acceso 3.3. Sobrecarga 3.4. Herencia y Polimorfismo 3.5. Getters y Setters 3.6. Interfaces 4. Taller: Aplicación en Java	5 horas	Sesión 1

2. Fundamentos de Android	1. Fundamentos de Android 1.1. Introducción 1.2. SDK Manager 1.3. Emuladores 1.4. AVD 1.5. DDMS 1.6. Proyecto de Android 1.6.1. Introducción 1.6.2. Arquetipo 1.6.3. Drawable 1.6.4. Layouts 1.6.5. Android Manifiest	5 horas	Sesión 2
3. Activities de Android	1. Activities 1.1. Layouts 1.2. Ciclo de vida 1.3. Intents 1.3.1. Implícitos 1.3.2. Explícitos 2. Taller: Mi aplicación en Android 3. Layouts 3.1. Linear 3.2. Relative 3.3. List View 3.4. Grid View 3.5. Frame	5 horas	Sesión 3

4. Componentes gráficos de Android	1. Componentes Gráficos 1.1. TextView 1.2. Password 1.3. Email 1.4. Phone 1.5. Number 1.6. Button 1.7. ImageButton 1.8. Checkbox 1.9. RadioGroup, RadioButton 1.10. ToggleButton 1.11. Switch 2. Widgets 2.1. ImageView 2.2. ProgressBar 2.3. Horizontal Divider 2.4. Vertical Divider 3. Taller: Aplicación con componentes Gráficos en Android	5 horas	Sesión 4

5. Fragments	1. Fragmentos 1.1. Introducción 1.2. ViewPager y Fragments 2. Permisos en Android 2.1. Uso de Permisos 2.2. Ejemplo con Cámara 3. Almacenamiento 3.1. Tipos 3.2. Shared Preferences 3.3. SQLite 3.3.1. Introducción 3.3.2. Instalación 3.3.3. Uso de Room 3.3.3.1. Concepto 3.3.3.2. Dao 3.3.3.3. Consultas 3.3.3.4. Migraciones 4. Taller: Aplicación de Android con	5 horas	Sesión 5
6. Servicios Web	1. REST 1.1. Introducción 1.2. Persistencia de datos con API REST 2. Retrofit 2.1. Instalación 2.2. Conceptos 2.3. Métodos 2.4. Manipulación de URL 2.5. Request Body 2.6. Form Encoded y Multipart 2.7. Headers 3. Taller: Aplicación con Servicios Web	5 horas	Sesión 6

7. Firebase	1. Firebase 1.1. Introducción 1.2. Instalación en Android 1.3. Colecciones 1.4. Documentos 1.5. Consulta y transformación de datos 2. Taller: Práctica Final	5 horas	Sesión 7
8. Utilidades avanzadas	1. Multi Lenguaje 1.1. Strings.xml 1.2. Traducción en proyectos 2. Cámara 2.1. Fotos 2.2. Video 2.3. Galería 3. Google Maps 3.1. Integración de mapas 3.2. Manipulación y configuración del mapa 3.3. Localización 3.4. Marcadores	5 horas	Sesión 8

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

1. Firebase 1.1. Introducción 1.2. Instalación en Android 1.3. Colecciones 1.4. Documentos 1.5. Consulta y transformación de datos 2. Taller: Práctica Final	5 horas	Sesión 7
1. Multi Lenguaje 1.1. Strings.xml 1.2. Traducción en proyectos 2. Cámara 2.1. Fotos 2.2. Video 2.3. Galería 3. Google Maps 3.1. Integración de mapas 3.2. Manipulación y configuración del mapa 3.3. Localización 3.4. Marcadores	5 horas	Sesión 8

Sesiones

Sesión 1

Fundamentos de Java

Sesión 2

Fundamentos de Android

Sesión 3

Ciclo de vida del Activity

Sesión 4

Componentes gráficos

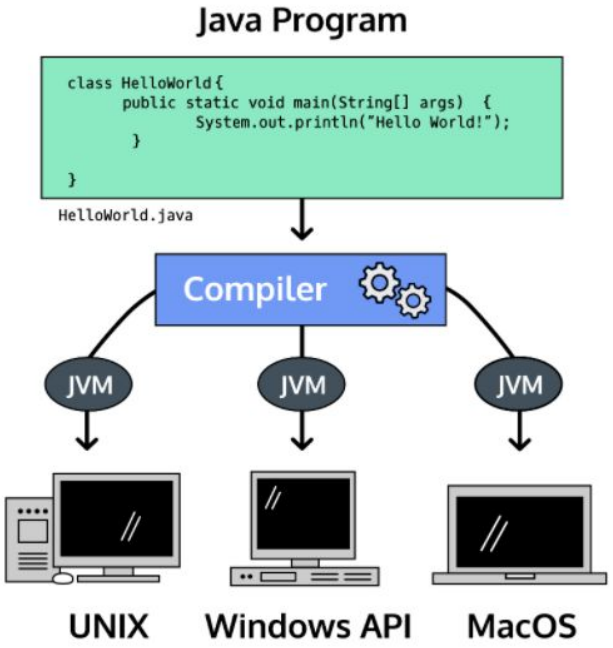
Sesión 5

Fragments

Sesión 6

Servicios Web

¿Qué es un “entorno de desarrollo” o “development environment” ?


Java Program
class HelloWorld{
public static void main(String[] args) {
System.out.println("Hello World!");
}
}
HelloWorld.java
Compiler
JVM
JVM
JVM
UNIX
Windows API
MacOS


IJ
Apache
NetBeans IDE
IDE
eclipse

¿Qué se de Java o Kotlin?

Java

```java
public class Run {
    public static void main(String args[]) {
      test();
    }

    static void test(){
        int x = 10;
        int y = 20;
        int z = sumar(x,y);

        System.out.println(
            "The sum of x (" +
            x + ") + y(" + y +
            ") = z(" + z + ")"
        );
    }

    static int sumar(int a, int b) {
        return a + b;
    }
}
```

Termial

```
The sum of x (10) + y(20) = z(30)

The sum of x (10) + y(20) = zFormal(30)
The sum of x (10) + y(20) = zInformal(30)
The sum of x (10) + y(20) = zFuncionFlecha(30)
```

Kotlin

```kotlin
fun main(args: Array<String>) {
    test()
}

fun test() {
    val x = 10
    val y = 20
    val zFormal = sumarFormal(x, y)
    println("The sum of x ($x) + y($y) = zFormal($zFormal)")

    val zInformal = sumarInformal(x, y)
    println("The sum of x ($x) + y($y) = zInformal($zInformal)")

    val sumarFuncionFlecha: (Int, Int) -> Int = { a, b -> a + b }
    val zFuncionFlecha = sumarFuncionFlecha(10, 20)
    println("The sum of x ($x) + y($y) = zFuncionFlecha($zFuncionFlecha)")
}

fun sumarFormal(a: Int, b: Int): Int {
    return a + b
}

fun sumarInformal(a: Int , b: Int) = a + b
```

1.2.1. Tipos de datos

byte: 8 bits [-128 a 127].
short: 16 bits [-32,768 a 32,767].
int: 32 bits [-2,147,483,648 a 2,147,483,647].
long: 64 bits.
float: 32 decimales.
double: 64 bits decimales con mayor precisión.
char: 16 bits un único carácter Unicode.
boolean: 8 bits verdadero (true) o falso (false).

1.2.2. Variable

Tipos de datos primitivos Java

```java
byte edad = 25;
short codigo = 12345;
int cantidad = 1000;
long poblacion = 7000000000L;
float precio = 19.99f;
double salario = 2500.50;
char letra = 'A';
boolean esActivo = true;
```

Tipos de datos primitivos Kotlin

```kotlin
val edad: Byte = 25
val codigo: Short = 12345
val cantidad: Int = 1000
val poblacion: Long = 7000000000L
val precio: Float = 19.99f
val salario: Double = 2500.50
val letra: Char = 'A'
val esActivo: Boolean = true
```

Variables no inicializadas Java

```java
int numero; // su valor predeterminado es 0
boolean esActivo; // su valor predeterminado es false
String texto; // su valor predeterminado es null
```

Variables no inicializadas Kotlin

```kotlin
var numero: Int
var esActivo: Boolean
var texto: String
```

Inferencia de tipos Kotlin

```kotlin
// Inferencia de tipos
val nombre = "Juan"
val edad = 25
val salario = 2500.50

var contador = 0
var esActivo = true

// Declaración explícita
val promedio: Double = 8.5
var codigo: String = "ABC123"
```

1.2.3. Casting

Casting Java

```java
// Implicito
int numero = 10;
double decimal = numero; // decimal = 10.0

// Explicito
double decimal = 3.14;
int numero = (int) decimal; // numero = 3
```

Casting Kotlin

```kotlin
// Seguro
val texto: String? = "Hola"
val numero1: Int? = texto as? Int // null

// Explicito
val decimal: Double = 3.14
val numero: Int = decimal.toInt() // numero = 3
```

1.2.4. Arreglos

Arreglos Java

```java
int[] numeros = new int[5];
String[] nombres = { "José", "Pedro", "Juan" };
```

Arreglos Kotlin

```kotlin
val numeros = Array<Int>(5) { 0 }
val nombres = arrayOf("José", "Pedro", "Juan")
```

1.2.5. Funciones

Función Java

```java
public static int sumar(int a, int b) {
      return a + b;
  }
```

Función Kotlin

```kotlin
fun sumar(a: Int, b: Int): Int {
    return a + b
}

fun sumar(a: Int , b: Int) = a + b

val sumar: (Int, Int) -> Int = { a, b -> a + b }
```

1.2.6. Sentencia de control

Sentencia de Control if

```
if (condicion1) {
    // Código a ejecutar si la condición1 es verdadera
} else if (condicion2) {
    // Código a ejecutar si la condición2 es verdadera
} else if (condicion3) {
    // Código a ejecutar si la condición3 es verdadera
} else {
    // Si ninguna de las condiciones anteriores es verdadera
}
```

Sentencia de Controls switchJava

```java
switch (expresion) {
    case valor1:
        // si la expresión es igual a valor1
        break;
    case valor2:
        // si la expresión es igual a valor2
        break;
    default:
        // si la expresión no coincide con ninguno
        break;
}
```

Sentencia de Control when Kotlin

```kotlin
when (expresion) {
    valor1 -> {
        // si expresión es igual a valor1
    }
    valor2 -> {
        // si la expresión es igual a valor2
    }
    else -> {
        // si la expresión no coincide con ninguno
    }
}
```

Sentencia de Control for y forEach Java

```java
for (int i = 0; i < 5; i++) {
    System.out.println("Iteración: " + i);
}

int[] numeros = {1, 2, 3, 4, 5};
for (int numero : numeros) {
    System.out.println(numero);
}
```

Sentencia de Control for y forEach Kotlin

```kotlin
for (i in 0 until 5) {
    println("Iteración: $i")
}

val numeros = arrayOf(1, 2, 3, 4, 5)
for (numero in numeros) {
    println(numero)
}
```

Sentencia de Controls when

```
while (condicion) {
    // Código a ejecutar mientras se cumpla la condición
}
```

1.2.7. Clases

Estrucutra de una Clase

```
class MiClase {
    // Atributos

    // Constructor(es)

    // Métodos
}
```

Estrucutra de una Clase Persona Java

```java
public static class Persona {
      // Atributos
      private String nombre;
      private int edad;

      // Constructor por defecto
      public Persona() {
      }

      // Métodos
      public void setNombre(String nombre) {
          this.nombre = nombre;
      }

      public void setEdad(int edad) {
          this.edad = edad;
      }

      public void saludar() {
          System.out.println("Hola, mi nombre es " +
          nombre + " y tengo " + edad + " años.");
      }
}
```

Estrucutra de una Clase Persona Kotlin

```kotlin
class Persona {
    var nombre: String = ""
    var edad: Int = 0

    constructor() {
        // Constructor primario
    }

    fun saludar() {
        println("Hola, mi nombre es" +
        nombre + " y tengo $edad años.")
    }
}
```

Uso de Clase Persona Java

```java
public static void main(String[] args) {
    Persona persona = new Persona();
    persona.setNombre("Juan");
    persona.setEdad(25);
    persona.saludar();
}
```

Uso de Clase Persona Kotlin

```kotlin
fun main() {
    val persona = Persona()
    persona.nombre = "Juan"
    persona.edad = 25
    persona.saludar()
}
```

```
Hola, mi nombre es Juan y tengo 25 años.
```

¿Qué se de POO?

1.3.1. Instancias

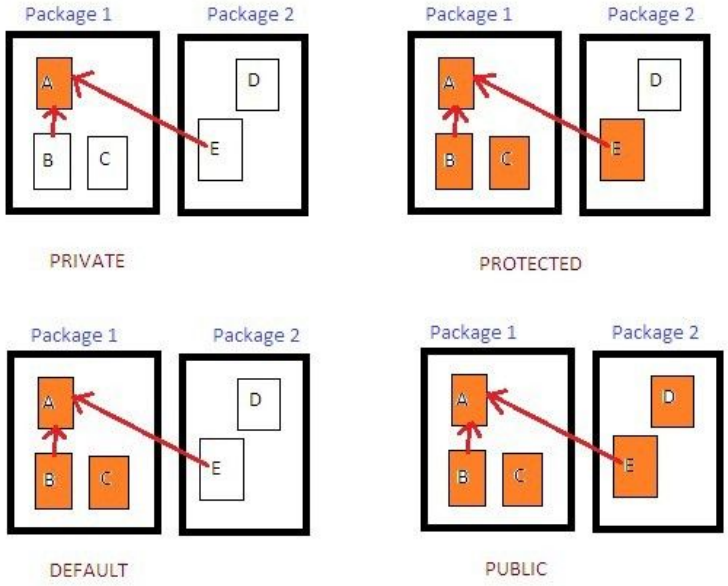
Instanciar Clase Persona Java

```java
public static void main(String[] args) {
    Persona persona1 = new Persona("Juan", 25);
    Persona persona2 = new Persona("María", 30);
}
```

Instanciar Clase Persona Kotlin

```kotlin
fun main() {
    val persona1 = Persona("Juan", 25)
    val persona2 = Persona("María", 30)
}
```

1.3.2. Modificadores de acceso


Package 1
Package 2
A
B
C
D
E
PRIVATE
Package 1
Package 2
A
B
C
D
E
PROTECTED
Package 1
Package 2
A
B
C
D
E
DEFAULT
Package 1
Package 2
A
B
C
D
E
PUBLIC

1.3.3. Sobrecarga

sobrecarga de métodos Java

```java
public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }

    public double sumar(double a, double b) {
        return a + b;
    }

    public int sumar(int a, int b, int c) {
        return a + b + c;
    }
}
```

sobrecarga de métodos Kotlin

```kotlin
class Calculadora {
    fun sumar(a: Int, b: Int): Int {
        return a + b
    }

    fun sumar(a: Double, b: Double): Double {
        return a + b
    }

    fun sumar(a: Int, b: Int, c: Int): Int {
        return a + b + c
    }
}
```

1.3.4. Herencia y Polimorfismo

Herencia y poliforfismo Java

```java
class Animal {
    public void hacerSonido() {
        System.out.println("Haciendo sonido genérico");
    }
}

class Perro extends Animal {
    public void hacerSonido() {
        System.out.println("Ladrando");
    }
}

class Gato extends Animal {
    public void hacerSonido() {
        System.out.println("Maullando");
    }
}
```

Herencia y poliforfismo Kotlin

```kotlin
open class Animal {
    open fun hacerSonido() {
        println("Haciendo sonido genérico")
    }
}

class Perro : Animal() {
    override fun hacerSonido() {
        println("Ladrando")
    }
}

class Gato : Animal() {
    override fun hacerSonido() {
        println("Maullando")
    }
}
```

1.3.5. Getters y Setters

Getters y setter Java

```java
public class Persona {
    private String nombre;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

Getters y setter Kotlin

```kotlin
class Persona {
    var nombre: String = ""
        get() = field
        set(value) {
            field = value
        }

    var edad: Int = 0
        get() = field
        set(value) {
            field = value
        }
}
```

1.3.6. Interfaces

Crear interfaz Java

```java
public interface Volador {
    void volar();

    default void aterrizar() {
        System.out.println("Aterrizando...");
    }

    static void despegar() {
        System.out.println("Despegando...");
    }
}
```

creación de interfaz Kotlin

```kotlin
interface Volador {
    fun volar()

    fun aterrizar() {
        println("Aterrizando...")
    }

    companion object {
        fun despegar() {
            println("Despegando...")
        }
    }
}
```

uso de interfaz Java

```java
public static class Ave implements Volador {
    @Override
    public void volar() {
        System.out.println("Volando como un ave");
    }
}

public static void main(String[] args) {
    Ave ave = new Ave();
    ave.volar();
    ave.aterrizar();
    Volador.despegar();
}
```

```
Volando como un ave
Aterrizando...
Despegando...
```

uso de interfaz Kotlin

```kotlin
class Pato : Volador {
    override fun volar() {
        println("Volando como un pato")
    }
}

fun main() {
    val pato = Pato()
    pato.volar()
    pato.aterrizar()
    Volador.despegar()
}
```

```
Volando como un pato
Aterrizando...
Despegando...
```

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

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

- Resumen de curso
- Reglamento
- Temario
- Sesiones

Sesión 2

Fundamentos de Android

Sesión 4

Componentes gráficos

Sesión 6

Servicios Web

Curso Android

Bienvenido alumno

Sesiones

Completadas	Exámenes	Tareas
1	0	0

Curso de Desarrollo Móvil en Android