



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO



TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

Ingeniería en Sistemas Computacionales.

Tarea 8_ Unidad 2 Creación de la IU de una app

Asignatura: Programación Nativa para Moviles

Docente: Jorge Peralta Escobar

Hora: 14:00 – 15:00 pm

Integrantes:

Del Angel Del Angel Erika Yaneth #20071926

Villaseñor Grimaldo Alejandro #19071548

Semestre: Enero – Junio 2025.

DEJANDO HUELLA 

RUTA DE APRENDIZAJE 1 - CONCEPTOS BASICOS DE KOTLIN

GitHub - [Workspace/Programacion_movil/Tarea8U2 at master · Alejandro19071548/Workspace](#)

VIDEO 1 – RUTA DE APRENDIZAJE 1

En esta unidad introductoria sobre Kotlin, se exploran conceptos fundamentales para desarrollar aplicaciones más interactivas. El enfoque principal está en el uso de las sentencias condicionales, que permiten que una app tome decisiones basadas en diferentes condiciones. Se explica el empleo del if, else if y else para controlar el flujo de ejecución, ejemplificándolo con situaciones cotidianas como activar el modo No Molestar o Silencio en un teléfono durante una película. Para manejar múltiples condiciones de manera más legible y eficiente, se presenta la sentencia when, que evalúa un único valor contra varias condiciones posibles, mejorando la claridad del código en lugar de usar múltiples if/else.

También se introduce el concepto de expresiones condicionales, donde las sentencias if/else y when no solo ejecutan bloques de código, sino que retornan valores que pueden asignarse a variables. Esto es crucial para evitar código redundante y aumentar la eficiencia. Además, se aborda cómo Kotlin trata a las funciones como ciudadanos de primera clase, es decir, como objetos que pueden asignarse a variables, pasarse como argumentos o retornarse desde otras funciones. Se destaca el uso de expresiones lambda, que son funciones anónimas que pueden almacenarse o usarse directamente para implementar, por ejemplo, listeners en botones o campos de entrada para hacer la app interactiva. En conjunto, estos conceptos forman la base para construir aplicaciones más dinámicas y reactivas con Kotlin.

CUESTIONARIO RUTA 1

1. El siguiente código imprimirá "Divisible by 5" si `number` es igual a 25.

```
if (number % 10 == 0) {  
    println("Divisible by 10")  
} else if (number == 5) {  
    println("Divisible by 5")  
}
```

☐ true

☒ false ¡Correcto!

2. ¿Cuál de las siguientes condiciones se cumple cuando `x = 5`?

Selecciona todas las respuestas que consideres correctas.

☒ `x == 5` ¡Correcto!

☒ `x in 1..5` ¡Correcto!

☒ `x is Int` ¡Correcto!

☐ `x % 5`

3. ¿Cuáles de los siguientes no es un concepto básico de programación orientada a objetos?

☐ Abstracción

☒ Legibilidad ✓ ¡Correcto!

☐ Inheritance

☐ Polimorfismo

4. ¿Cuáles son los cuatro modificadores de visibilidad en Kotlin?

☐ `public`, `private`, `protected` y `abstract`

☐ `static`, `override`, `internal` y `external`

☒ `private`, `protected`, `public` y `internal` ✓ ¡Correcto!

☐ `public`, `protected`, `static` y `internal`

5. La palabra clave ___ se usa para llamar a un método de la clase superior.

☐ `this`

☒ `super` ✓ ¡Correcto!

6. Un(a) ___ define las propiedades o los métodos que una clase debe implementar.

☐ Delegar

☐ Tipo genérico

☒ Interfaz ✓ ¡Correcto!

☐ Subclase

7. ¿Cuál de las siguientes opciones se representa mejor con un tipo anulable?

☐ Indica la cantidad de seguidores (0 o más) en una app de redes sociales.

☒ Una foto de perfil opcional ✓ ¡Correcto!

☐ Un nombre de usuario que debe tener al menos un carácter.

☐ Es un ID único asignado a cada usuario.

8. El operador ____ permite llamar a un método solo si el objeto no es nulo.

☐ .


☐ !!

☐ ?:

☒ ?.  ¡Correcto!

9. ¿Cuál de estas afirmaciones sobre funciones no es verdadera en Kotlin?

☒ Una función se puede cambiar a otro tipo de datos, y viceversa.

 ¡Correcto!

☐ Una función se puede mostrar desde otra.


☐ Una función puede tomar otra como parámetro.

☐ Una función tiene un tipo de datos, como `(Int) -> Unit`.

10. Un literal de función es otro nombre para una ____.

☐ Tipo de función

☒ Expresión lambda

 ¡Correcto!

Results

Tu puntuación es **10 de 10**. ¡Felicitaciones! Aprobaste el cuestionario.

RUTA DE APRENDIZAJE 2 - AGREGA UN BOTON A UNA APP

VIDEO 1 – RUTA DE APRENDIZAJE 2

El video presenta una introducción divertida e informativa sobre la importancia de los dados en los juegos de mesa y propone un proyecto práctico para crear una aplicación que simule un dado digital. Explica que, generalmente, un dado tiene seis caras numeradas del uno al seis, cada una con la misma probabilidad de salir al lanzar el dado. La aplicación que se construirá incluirá una imagen del dado y un botón para “lanzarlo”, de modo que al pulsar el botón, el programa generará un número aleatorio entre uno y seis y actualizará la imagen para mostrar el resultado. El propósito principal es que los usuarios aprendan a crear interfaces que respondan a las interacciones, a modificar la UI dinámicamente y a manejar diferentes resultados. Además, se menciona que la app será desarrollada utilizando Jetpack Compose, una moderna herramienta para crear interfaces nativas en Android. Finalmente, el video anima a los espectadores a usar su aplicación casera en la próxima noche de juegos y a sorprender a sus amigos con su creación.

CUESTIONARIO RUTA 2

1. Usa un elemento ___ componible para mostrar una imagen.

☐ Botón

☐ Texto

☒ Imagen ✓ ¡Correcto!

☐ Ícono

2. `Alignment.Center` centra los componentes de la IU de manera horizontal y vertical.

☒ Verdadero ✓ ¡Correcto!

☐ Falso

3. Las funciones componibles pueden almacenar un objeto en la memoria con el elemento componible ___.

☒ recordar ✓ ¡Correcto!

☐ Columna

4. El depurador te permite inspeccionar variables cuando se suspende la ejecución del código.

☒ Verdadero ✓ ¡Correcto!

☐ Falso

5. Cuando se usan valores ____ en una función componible, se pueden hacer variables observables que programen una recomposición cuando se cambie su valor.

☐ recordar

☐ Modificador

☐ Componible

☒ mutableStateOf ✓ ¡Correcto!

6. El elemento ____ componible coloca sus elementos secundarios en una secuencia vertical.

☐ Fila

☐ Cuadro

☒ Columna ✓ ¡Correcto!

7. La función del depurador de ____ te permite navegar hacia atrás en la pila de llamadas.

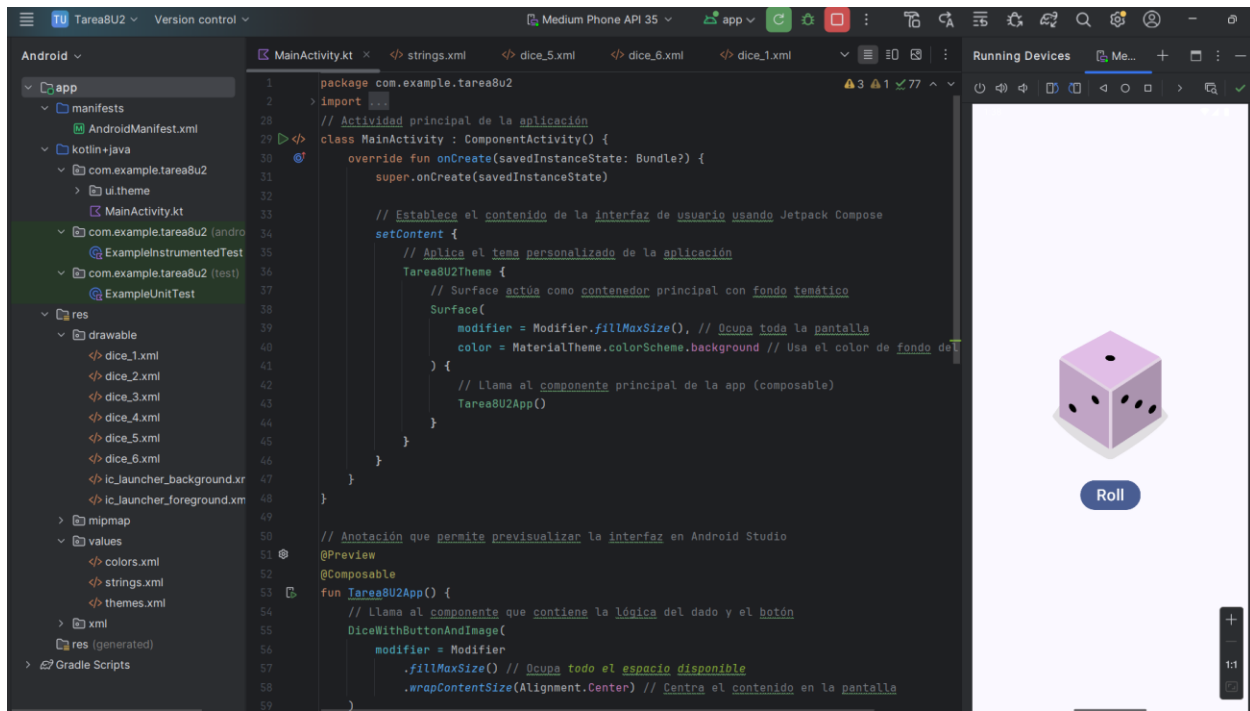
☐ Omitir

☒ Salir ✓ ¡Correcto!

☐ Entrar

☐ Reanudar programa

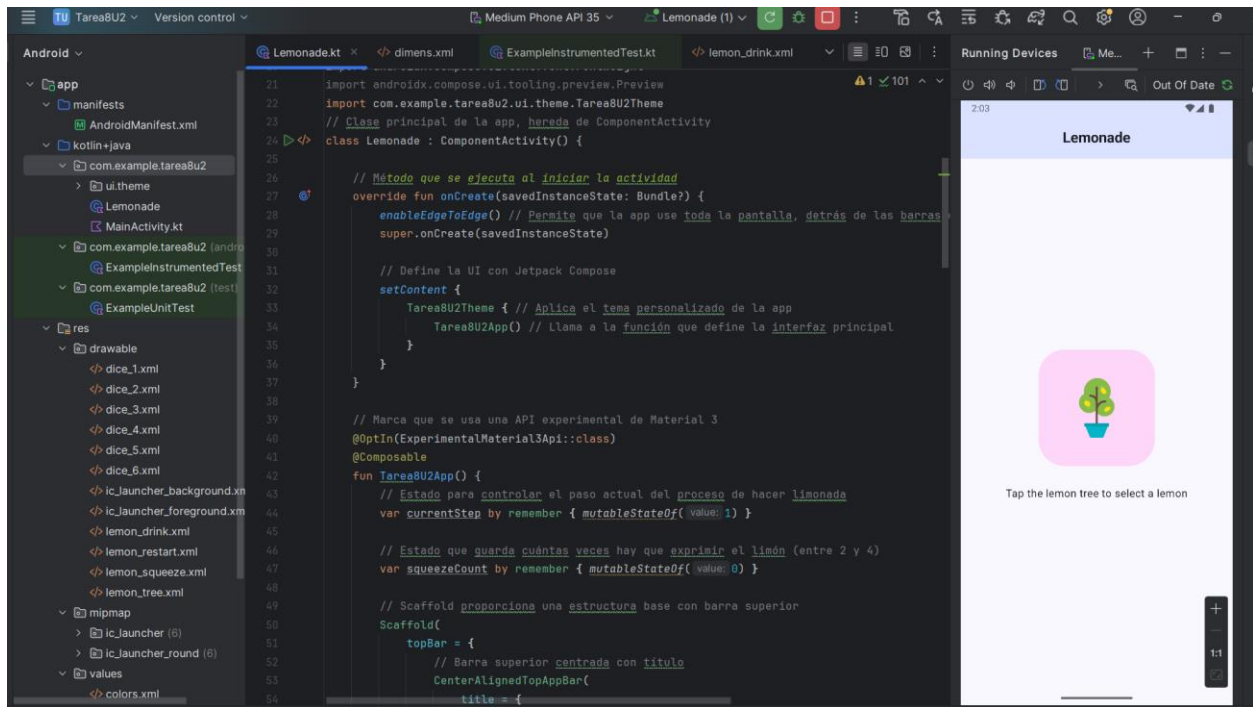
Dice Roller



Este código crea una app sencilla de Android usando Jetpack Compose que muestra:

- La imagen de un dado (de 1 a 6).
- Un botón que, al presionarlo, lanza el dado generando un número aleatorio del 1 al 6.
- La imagen del dado se actualiza automáticamente según el número generado.

LEMONADE



Esta app de Android simula el proceso de hacer limonada en 4 pasos interactivos:

1. Seleccionar un limón del árbol.
2. Exprimir el limón varias veces (de 2 a 4 toques).
3. Beber la limonada.
4. Reiniciar el proceso.

Cada paso muestra una imagen y un texto. Al tocar la imagen, se avanza al siguiente paso. Está hecha con Jetpack Compose y usa Button, Image, Text y Scaffold para construir la interfaz.

RUTA DE APRENDIZAJE 3 - INTERACTUA CON LA IU Y EL ESTADO

VIDEO 1 – RUTA DE APRENDIZAJE 3

Murat Yener, experto en desarrollo de Android con un enfoque en marcos de IU modernos, lidera una exploración integral de la administración de estado dentro de Jetpack Compose. Con una amplia experiencia en la creación de aplicaciones interactivas, Yener enfatiza la importancia del

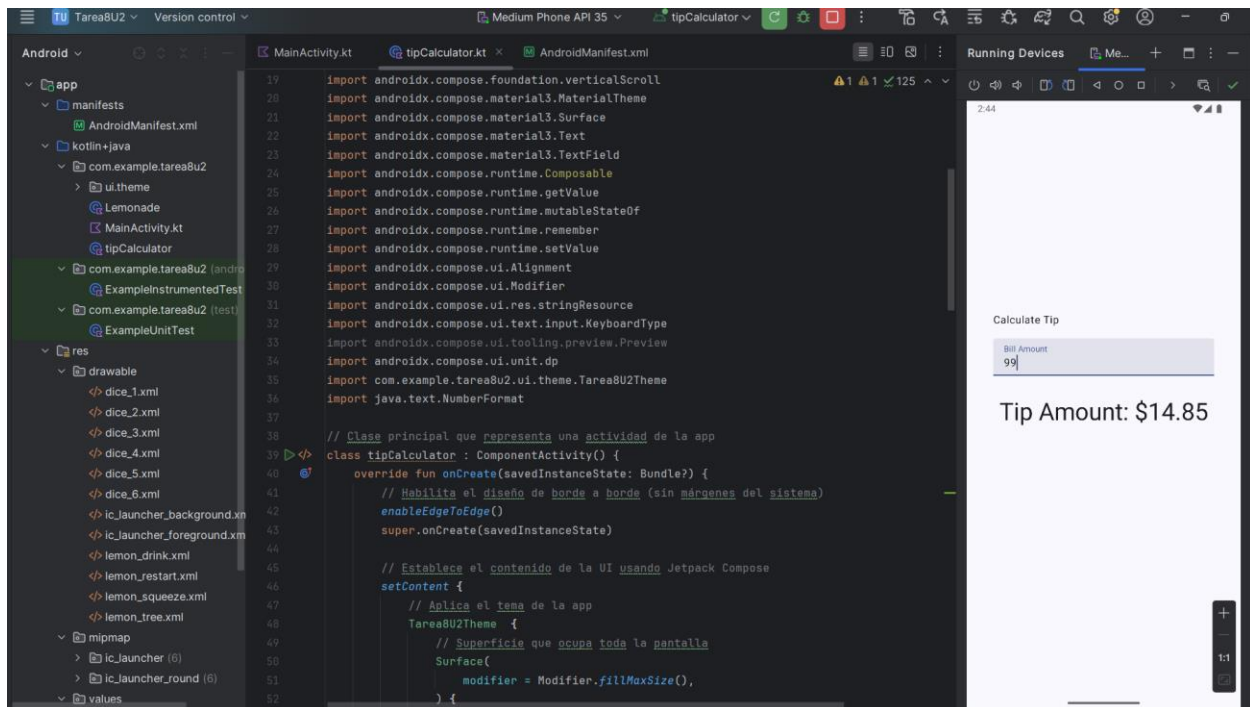
estado en el desarrollo de aplicaciones, ilustrando cómo puede ser dinámico y evolucionar con el tiempo. Aclara las metodologías básicas para administrar el estado, incluidos los conceptos de recordar el estado y elevar el estado, que facilitan la creación de elementos componibles reutilizables y sin estado. Su enfoque se basa en ejemplos prácticos y explicaciones claras, lo que proporciona información útil para los desarrolladores que buscan mejorar su comprensión de la arquitectura de Compose.

VIDEO 2 – RUTA DE APRENDIZAJE 3

Descripción general de la aplicación Calculadora de propinas

- Este proyecto consiste en el desarrollo de una aplicación de calculadora de propinas que ayuda a los usuarios a calcular la cantidad de propina adecuada cuando sea necesario.
- El desarrollo de la app se llevará a cabo en dos codelabs separados, lo que permitirá una experiencia de aprendizaje estructurada.

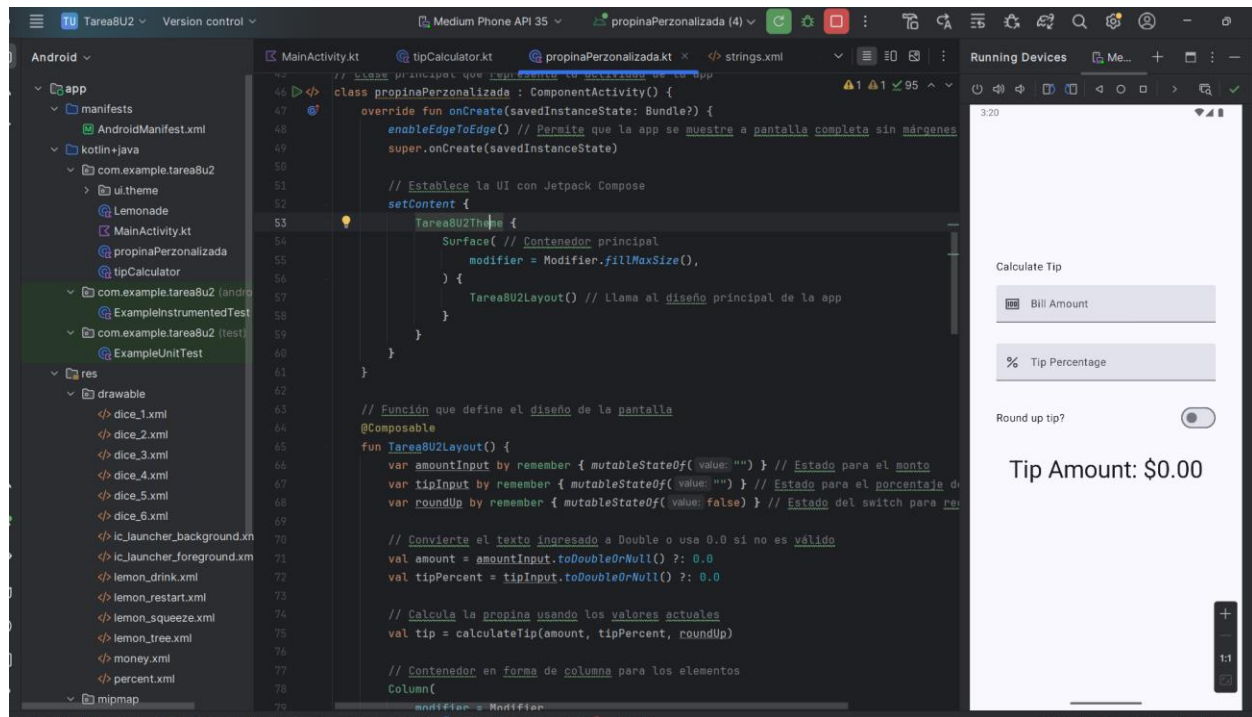
TIP CALCULADORA



Esta aplicación en Android desarrollada con Jetpack Compose permite calcular la propina basada en el monto ingresado por el usuario. Al iniciar, muestra un campo donde se puede escribir la

cantidad total de una cuenta y, al mismo tiempo, calcula automáticamente el 15 por ciento de ese monto como propina. El resultado se muestra en pantalla en formato de moneda. La interfaz está compuesta por un diseño vertical centrado con un título, un campo de entrada y un texto que muestra la propina calculada. Todo el contenido se adapta a la pantalla, incluye scroll por si el espacio no es suficiente y respeta las zonas seguras del sistema.

CALCULADORA PERSONALIZADA



Esta aplicación calcula una propina personalizada en Android usando Jetpack Compose. El usuario puede ingresar el monto de la cuenta y el porcentaje de propina que desea aplicar, además puede activar una opción para redondear el resultado. La app toma esos datos, realiza el cálculo y muestra el total de la propina en formato de moneda. La interfaz incluye dos campos de entrada con íconos, un interruptor para redondear, y un texto con el resultado. Todo está alineado verticalmente y adaptado para ocupar toda la pantalla con scroll en caso necesario.

CUESTIONARIO RUTA 3

1. Jetpack Compose ejecuta tus elementos componibles por primera vez durante ____, y realizará un seguimiento de esos elementos que llamas para describir la IU.

☒ Composición inicial

✓ ¡Correcto!

☐ Recomposición

☐ Cambio de estado

☐ Cierre inesperado de la app

2. La única forma de modificar un objeto Composition es a través de la recomposición.

☒ Verdadero

✓ ¡Correcto!

☐ Falso

3. El/la ____ se genera cuando Jetpack Compose vuelve a ejecutar los elementos componibles que pueden haberse modificado en respuesta a los cambios de datos.

☐ Composición inicial

☒ Recomposición

✓ ¡Correcto!

☐ Cambio de estado

4. ___ en una aplicación es cualquier valor que puede cambiar con el tiempo.

☒ Estado ✓ ¡Correcto!

☐ valor

☐ valueChange

☐ StateValue

5. ___ es un patrón en el que el estado se mueve hacia arriba para dejar a un componente sin estado.

☐ Cambio de estado

☒ Elevación de estado ✓ ¡Correcto!

☐ Composición elevada

☐ Recomposición

6. ¿Qué propiedad `KeyboardAction` se usa para mover el foco al siguiente elemento que admite composición?

☐ `onDone`

☒ `onNext` ✓ ¡Correcto!

☐ `onGo`

☐ `onSend`

7. ¿Cuál de las siguientes funciones de Kotlin se usa para redondear un double o float?

☐ `kotlin.math.ceilUp()`

☒ `kotlin.math.ceil()` ✓ ¡Correcto!

☐ `kotlin.math.roundDown()`

☐ `kotlin.math.roundUp()`

8. El Inspector de diseño es una herramienta de Jetpack Compose que te permite inspeccionar un diseño de Compose dentro de una app en ejecución en un emulador o dispositivo físico.

☒ Verdadero ✓ ¡Correcto!

9. Las pruebas de IU se almacenan en el directorio ____.

☐ principal

☒ androidTest

✓ ¡Correcto!

☐ prueba

☐ res

10. Las pruebas locales y de la IU deben anotarse con la anotación ____.

☐ @VisibleForTesting

☐ @Preview

☒ @Test

✓ ¡Correcto!

☐ @Composable

Results

Tu puntuación es **10 de 10**. ¡Felicitaciones! Aprobaste el cuestionario.