Guía de Laboratorio 4

Animaciones Implícitas en Flutter

1. Objetivo General

Aplicar el uso de **animaciones implícitas en Flutter** para mejorar la **interactividad y la experiencia visual** de las aplicaciones, utilizando widgets que permiten transiciones suaves sin necesidad de controladores explícitos.

2. Objetivos Específicos

- Comprender la diferencia entre **animaciones implícitas y explícitas**.
- Identificar y utilizar los principales **widgets de animación implícita** disponibles en Flutter.
- Implementar ejemplos prácticos de animaciones que modifiquen **propiedades visuales** (tamaño, color, opacidad, posición, rotación, escala, etc.).
- Desarrollar una pequeña aplicación demostrativa que integre varios tipos de animaciones implícitas de forma coordinada.

3. Marco Teórico

3.1. Concepto de Animaciones Implícitas

Las **animaciones implícitas** en Flutter son aquellas en las que el propio framework se encarga de interpolar los valores entre un estado inicial y uno final cuando cambia una propiedad. No se necesita un AnimationController; basta con especificar:

- una **duración** (duration)
- una **curva de animación** (curve)
- y los **valores iniciales y finales** de las propiedades.

El sistema calcula automáticamente la transición.

3.2. Principales Widgets de Animación Implícita

Flutter ofrece una variedad de widgets Animated... que permiten animar cambios de propiedades sin controladores:

Widget	Propiedad animada principal
AnimatedContainer	tamaño, color, bordes
AnimatedOpacity	transparencia
AnimatedAlign	alineación
AnimatedPadding	espacio interno
AnimatedPositioned	posición en un Stack
AnimatedRotation	rotación
AnimatedScale	tamaño (zoom)
AnimatedSize	cambios de tamaño
AnimatedCrossFade	transición entre dos widgets
AnimatedSwitcher	transición al cambiar hijo
AnimatedDefaultTextStyle	estilo de texto

3.3. Ventajas

- No requieren configuración manual de controladores.
- Fáciles de implementar.
- Reducen la complejidad del código.
- Permiten mejorar la **fluidez visual** y la **percepción de calidad** en la interfaz.

4. Desarrollo del Laboratorio

4.1. Preparación del Proyecto

1. Crear un nuevo proyecto Flutter (animation_app):
2. Abrir el proyecto en VS Code.
3. Ejecutar el proyecto para verificar que funciona correctamente.

4.2. Estructura Base

Edita el archivo lib/main.dart con el siguiente código base:

```
import 'package:flutter/material.dart';

void main() => runApp(const AnimacionesApp());

class AnimacionesApp extends StatelessWidget {
  const AnimacionesApp({super.key});
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Laboratorio de Animaciones Implícitas',
      theme: ThemeData(useMaterial3: true, colorSchemeSeed: Colors.blue),
      home: const AnimacionesScreen(),
    );
  }
}

class AnimacionesScreen extends StatefulWidget {
  const AnimacionesScreen({super.key});

  @override
  State<AnimacionesScreen> createState() => _AnimacionesScreenState();
}

class _AnimacionesScreenState extends State<AnimacionesScreen> {
  bool _cambio = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Animaciones Implícitas')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            AnimatedContainer(
              duration: const Duration(milliseconds: 600),
              width: _cambio ? 200 : 100,
              height: _cambio ? 100 : 200,
              color: _cambio ? Colors.teal : Colors.orange,
              curve: Curves.easeInOut,
            ),
            const SizedBox(height: 20),
            AnimatedOpacity(
              duration: const Duration(milliseconds: 600),
              opacity: _cambio ? 0.3 : 1.0,
              child: const Text(
                'Flutter Animations',
                style: TextStyle(fontSize: 24),
              ),
            ),
            const SizedBox(height: 20),
```

```
            AnimatedScale(
              duration: const Duration(milliseconds: 600),
              scale: _cambio ? 1.5 : 1.0,
              child: const Icon(Icons.star, size: 50),
            ),
            const SizedBox(height: 40),
            ElevatedButton(
              onPressed: () => setState(() => _cambio = !_cambio),
              child: const Text('Animar'),
            ),
          ],
        ),
      ),
    );
  }
}
```

4.3. Actividades Prácticas

1. **Ejercicio 1:** Modifica el AnimatedContainer para que también cambie el **radio de borde** y el **sombra (BoxShadow)**.

```
              decoration: BoxDecoration(
                color: _cambio ? Colors.teal : Colors.orange,
                borderRadius: BorderRadius.circular(_cambio ? 30 : 5),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withValues(alpha: _cambio ? 0.5 : 0.0),
                    blurRadius: _cambio ? 20 : 0,
                    offset: const Offset(0, 8),
                  ),
                ],
              ),
```

2. **Ejercicio 2:** Agrega un AnimatedAlign para mover el texto “`Flutter Animations`” a la izquierda y al centro. Manteniendo la animación de la opacidad.

```
              AnimatedAlign(
              duration: const Duration(milliseconds: 600),
              alignment: _cambio ? Alignment.center : Alignment.topLeft,
              curve: Curves.easeInOut,
              child: AnimatedOpacity(
                duration: const Duration(milliseconds: 600),
                opacity: _cambio ? 0.3 : 1.0,
                child: const Text(
```

```
            'Flutter Animations',
            style: TextStyle(fontSize: 22),
          ),
        ),
      ),
```

3. **Ejercicio 3:** Implementa AnimatedRotation considerando el el icono “Icons.refresh”.

```
      AnimatedRotation(
        duration: const Duration(milliseconds: 600),
        curve: Curves.easeInOut,
        turns: _cambio ? 0.5 : 0.0, // 0.5 = 180°
        child: const Icon(Icons.refresh, size: 48),
      ),
```

4. **Ejercicio 4:** Implementa AnimatedCrossFade para observar una animación de cambio de estados.

```
       AnimatedCrossFade(
        duration: const Duration(milliseconds: 600),
        firstChild: const Text(
          'Estado A',
          style: TextStyle(fontSize: 20),
        ),
        secondChild: Row(
          mainAxisSize: MainAxisSize.min,
          children: const [
            Icon(Icons.check_circle, size: 22),
            SizedBox(width: 8),
            Text('Estado B', style: TextStyle(fontSize: 20)),
          ],
        ),
        crossFadeState: _cambio
            ? CrossFadeState.showSecond
            : CrossFadeState.showFirst,
        // Opcional para ajustar layout:
        alignment: Alignment.center,
      ),
```