**BÁO CÁO TUẦN 3**

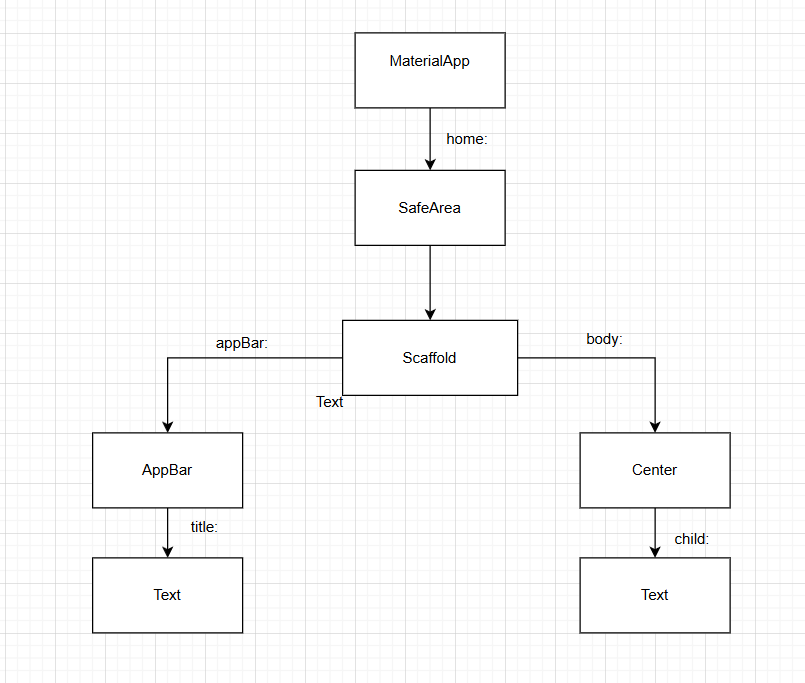
**Họ và tên:** Nguyễn Kỳ Anh  
**MSSV:** 20225793  
 **GR1**  
**Giáo viên hướng dẫn:** Nguyễn Đức Toàn

# 1. Flutter

* Flutter Engine
* Foundation Library
* Widgets
* Design Specific Widgets
* Flutter Engine : Flutter Engine là một **môi trường chạy di động linh hoạt** dành cho các ứng dụng chất lượng cao, chủ yếu được xây dựng bằng **ngôn ngữ lập trình C++**. Nó bao gồm **hệ thống hoạt ảnh và đồ họa**, **quản lý tệp và I/O mạng**, **kiến trúc plugin**, **hỗ trợ truy cập** và **một trình runtime Dart** để viết, biên dịch và thực thi các ứng dụng Flutter. Flutter Engine sử dụng **Skia**, một thư viện đồ họa mã nguồn mở của Google, để tạo ra các hình ảnh đồ họa cấp thấp.
* Foundation Library: **Thư viện Foundation** cung cấp tất cả các gói cần thiết để xây dựng các khối cơ bản của một ứng dụng Flutter. Các thư viện này được viết bằng **ngôn ngữ lập trình Dart**.
* Widgets: Mọi thứ trong Flutter đều là **widget**, đây là khái niệm cốt lõi của framework. Trong Flutter, **widget** là một **thành phần giao diện người dùng (UI)**, có ảnh hưởng và kiểm soát cách hiển thị cũng như tương tác của ứng dụng. Widget là một **mô tả bất biến** của một phần giao diện người dùng, bao gồm **đồ họa, văn bản, hình dạng và hiệu ứng hoạt ảnh**, được tạo ra bằng cách sử dụng các widget khác. Các widget trong Flutter tương tự như **các component trong React**.
* Design Specific Widgets:
* **Flutter framework** có hai loại widget tuân theo các ngôn ngữ thiết kế khác nhau:
* **Material Design** dành cho ứng dụng Android.
* **Cupertino Style** dành cho ứng dụng iOS.

# 2. Widgets

Widget tree:



Chúng ta có thể tạo một **widget** trong Flutter như sau:

Class ImageWidget extends StatelessWidget {

// Class-Stuff

}

VD: Hello Everyone

import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {

// Constructor của HomePage, nhận vào một tiêu đề (title)

HomePage({Key? key, this.title}) : super(key: key);

// Đây là widget màn hình chính của ứng dụng

final String title;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

// Hiển thị tiêu đề của trang

title: Text(this.title),

),

body: Center(

// Hiển thị văn bản "Hello Everyone" ở giữa màn hình

child: Text('Hello Everyone'),

),

);

}

}

- WIDGET TYPES: Widget trong Flutter có thể được chia thành hai loại:

* **Hiển thị (Visible)**: Gồm các widget đầu ra (Output) và đầu vào (Input).
* **Ẩn (Invisible)**: Gồm các widget bố cục (Layout) và điều khiển (Control).

## 2.1 StatelessWidget

* StatelessWidget là một loại **widget không có trạng thái** trong Flutter. Nó có đặc điểm:
* **Không thay đổi trong suốt vòng đời** – Dữ liệu bên trong StatelessWidget là bất biến.
* **Chỉ được xây dựng một lần** – Khi widget được tạo, nó không thể cập nhật lại giao diện trừ khi được khởi tạo lại từ đầu.
* **Phù hợp cho giao diện tĩnh** – Các phần của UI không cần thay đổi, chẳng hạn như văn bản, hình ảnh, biểu tượng.

**Khi nào sử dụng StatelessWidget?**

* Hiển thị nội dung tĩnh như văn bản, hình ảnh, nút bấm mà không cần cập nhật lại.
* Không cần lắng nghe sự kiện thay đổi từ người dùng.
* Khi muốn tối ưu hiệu suất vì StatelessWidget không yêu cầu quản lý trạng thái.

Vd:

import 'package:flutter/material.dart';

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: HomePage(),

);

}

}

class HomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('StatelessWidget Example')),

body: Center(

child: Text('Hello, this is a StatelessWidget!'),

),

);

}

}

void main() {

runApp(MyApp());

}

## StatefulWidget

* Khái niệm: StatefulWidget là một loại **widget có trạng thái** trong Flutter. Nó có khả năng thay đổi dữ liệu trong quá trình chạy ứng dụng và cập nhật lại giao diện khi trạng thái thay đổi.
* Đặc điểm của StatefulWidget:

+ Có thể thay đổi trạng thái (state) – Dữ liệu bên trong có thể thay đổi sau khi widget được tạo.  
+ Cập nhật lại giao diện (rebuild) – Khi trạng thái thay đổi, widget sẽ tự động cập nhật lại.

+ Tạo ra hai lớp:

* Lớp chính (StatefulWidget) – Xác định widget.
* Lớp trạng thái (State) – Chứa dữ liệu thay đổi và logic cập nhật UI.

- Khi nào sử dụng StatefulWidget?

* Khi giao diện cần thay đổi theo dữ liệu hoặc sự kiện.
* Khi có **tương tác người dùng** (ví dụ: nút bấm, nhập dữ liệu).
* Khi muốn cập nhật trạng thái mà không cần khởi tạo lại widget từ đầu.

VD:

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: CounterScreen(), // Màn hình chính

);

}

}

// Lớp StatefulWidget

class CounterScreen extends StatefulWidget {

@override

\_CounterScreenState createState() => \_CounterScreenState();

}

// Lớp State chứa logic

class \_CounterScreenState extends State<CounterScreen> {

int \_counter = 0; // Biến trạng thái

// Hàm tăng giá trị bộ đếm

void \_incrementCounter() {

setState(() {

\_counter++; // Cập nhật trạng thái

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('StatefulWidget Example')),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Số lần nhấn:',

style: TextStyle(fontSize: 20),

),

Text(

'$\_counter', // Hiển thị giá trị bộ đếm

style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

),

],

),

),

floatingActionButton: FloatingActionButton(

onPressed: \_incrementCounter, // Gọi hàm tăng số

child: Icon(Icons.add),

),

);

}

}

## 2.3 Widget hiển thị (Visible Widget)

* Các widget hiển thị có liên quan đến dữ liệu do người dùng nhập vào và dữ liệu đầu ra. Các widget này có nhiều dạng khác nhau, bao gồm một số loại phổ biến sau:

### Text():

Constructor Text():

* **data**: Chuỗi văn bản hiển thị.
* **style**: Định dạng văn bản như kích thước, màu sắc, độ đậm, kiểu chữ.

VD:

Text(

'Đây là văn bản có định dạng!',

style: TextStyle(

fontSize: 24, // Cỡ chữ

fontWeight: FontWeight.bold, // Độ đậm (bold)

fontStyle: FontStyle.italic, // Kiểu chữ nghiêng

letterSpacing: 2.0, // Khoảng cách giữa các chữ cái

wordSpacing: 5.0, // Khoảng cách giữa các từ

color: Colors.blue, // Màu chữ

decoration: TextDecoration.underline, // Gạch chân

decorationColor: Colors.red, // Màu của gạch chân

decorationThickness: 2.0, // Độ dày của gạch chân

),

)

* **textAlign**: Căn chỉnh văn bản (left, center, right, justify).

**Giá trị có thể sử dụng**:

* TextAlign.left – Căn trái (mặc định).
* TextAlign.right – Căn phải.
* TextAlign.center – Căn giữa.
* TextAlign.justify – Căn đều hai bên.

VD:

Text(

'Chào mừng bạn đến với Flutter!',

textAlign: TextAlign.center, // Căn giữa văn bản

)

* **textDirection**: Hướng văn bản (LTR - trái sang phải, RTL - phải sang trái).

**Có hai giá trị**:

* TextDirection.ltr – Từ trái sang phải (mặc định, dùng cho tiếng Anh, Việt Nam…).
* TextDirection.rtl – Từ phải sang trái (dùng cho tiếng Ả Rập, Do Thái…).

Vd:

Text(

'مرحبا بك في Flutter!',

textDirection: TextDirection.rtl, // Văn bản từ phải sang trái (dùng cho tiếng Ả Rập, Hebrew)

)

* **softWrap**: Cho phép hoặc không cho phép xuống dòng tự động.

**Giá trị có thể sử dụng**:

* true – Văn bản sẽ tự động xuống dòng khi chạm mép màn hình.
* false – Văn bản sẽ không xuống dòng, có thể bị cắt hoặc tràn khỏi màn hình.

Vd:

Text(

'Flutter là một framework mạnh mẽ giúp lập trình viên phát triển ứng dụng đa nền tảng.',

softWrap: false, // Không tự động xuống dòng

)

* **overflow**: Xử lý khi văn bản quá dài (có thể cắt bỏ hoặc hiển thị dấu ...).

**Các giá trị của TextOverflow**:

* TextOverflow.clip – Cắt bỏ phần dư mà không có dấu hiệu gì.
* TextOverflow.fade – Làm mờ dần phần cuối của văn bản.
* TextOverflow.ellipsis – Hiển thị dấu ... khi văn bản bị cắt.

Vd:

Text(

'Flutter là một framework tuyệt vời để phát triển ứng dụng di động và web.',

maxLines: 1, // Giới hạn số dòng hiển thị

overflow: TextOverflow.ellipsis, // Hiển thị "..." khi văn bản quá dài

)

* **maxLines**: Giới hạn số dòng hiển thị.

**Giá trị mặc định**: null – Không giới hạn số dòng.

Vd:

Text(

'Dart là ngôn ngữ lập trình mạnh mẽ giúp phát triển ứng dụng với Flutter.',

maxLines: 2, // Chỉ hiển thị tối đa 2 dòng

)

* **textScaler** – Điều chỉnh tỷ lệ văn bản.

**Các cách sử dụng textScaler**:

* TextScaler.linear(2.0) – Tăng gấp đôi kích thước chữ.
* TextScaler.noScaling – Không thay đổi kích thước chữ.

Vd:

Text(

'Đây là văn bản có kích thước điều chỉnh!',

textScaler: TextScaler.linear(1.5), // Phóng to văn bản 1.5 lần

)

* **semanticsLabel** – Gán mô tả cho công cụ đọc màn hình

Vd:

Text(

'📌 Đây là nội dung quan trọng!',

semanticsLabel: 'Đây là nội dung quan trọng không chứa emoji', // Trợ năng

)

Tổng hợp:

|  |  |  |
| --- | --- | --- |
| | **Thuộc tính** | **Công dụng** | | --- | --- | |
| |  |  | | --- | --- | | textAlign | Căn chỉnh văn bản (trái, phải, giữa, justify). | |
| |  |  | | --- | --- | | style | Định dạng chữ (màu, cỡ chữ, độ đậm, kiểu chữ, khoảng cách…). | |
| |  |  | | --- | --- | | textDirection | Xác định hướng văn bản (LTR hoặc RTL). | |
| |  |  | | --- | --- | | overflow | Xử lý khi văn bản quá dài (cắt, thu gọn, hiển thị "..."). | |
| |  |  | | --- | --- | | softWrap | Cho phép văn bản tự động xuống dòng hay không. | |
| |  |  | | --- | --- | | maxLines | Giới hạn số dòng hiển thị. | |
| |  |  | | --- | --- | | textScaler | Điều chỉnh tỷ lệ kích thước văn bản. | |
| |  |  | | --- | --- | | semanticsLabel | Hỗ trợ mô tả cho công cụ đọc màn hình. | |