**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**Chủ đề: Image và Icons trong Flutter**

**Sinh viên thực hiện:**

1. **Đặng Quang Lâm Lớp:22KTMT2**  **MSSV:**  106220256
2. **Nguyễn Hữu Nam** **Lớp: 22KTMT2** **MSSV:**  106220262

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

**THUYẾT MINH**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**Chủ đề: Image và Icons trong Flutter**

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | Đặng Quang Lâm | Tìm hiểu lý thuyết về tổng quan widget Image & Icon, so sánh NetworkImage vs AssetImage, cách cấu hình assets trong, tìm hiểu các thuộc tính Image properties, chuẩn bị slide và báo cáo… | 50% |
| 02 | Nguyễn Hữu Nam | Tìm hiểu lý thuyết về tổng quan widget Image & Icon, So sánh Material Icons và Cupertino Icons, tìm hiểu kỹ thuật xử lý lỗi/loading, demo code, chuẩn bị slide và báo cáo... | 50% |

 Link code github: [github](https://github.com/quanglam19082004/DaNenTang)

**Mục lục**

[**1.** **Tổng quan về Image và Icon trong Flutter**](#_Toc211378379)

[**1.1 Tầm quan trọng của hình ảnh và biểu tượng** 4](#_Toc211378380)

[**1.2 Các loại nguồn hình ảnh** 4](#_Toc211378381)

[**2.** **So sánh NetworkImage và AssetImage trong Flutter** 4](#_Toc211378382)

[**2.1 Định nghĩa** 4](#_Toc211378383)

[**2.2 Bảng so sánh NetworkImage với AssetImage** 5](#_Toc211378384)

[**2.3 Thực tiễn ứng dụng** 5](#_Toc211378385)

[**3.** **Cấu hình và quản lý assets trong pubspec.yaml** 5](#_Toc211378386)

[**3.1 Nguyên lý quản lý assets của flutter** 6](#_Toc211378387)

[**3.2 Cách bao trùm nhiều hình ảnh** 6](#_Toc211378388)

[**3.3 Quy trình cập nhật asset mới** 6](#_Toc211378389)

[**3.4 Đa dạng hoá độ phân giải hình ảnh (đáp ứng màn hình sắc nét)** 7](#_Toc211378390)

[**3.5 Demo Khai báo và sử dụng assets trong pubspec.yaml** 7](#_Toc211378391)

[**4.** **Thuộc tính quan trọng của Image widget: width, height, fit, alignment** 7](#_Toc211378392)

[**4.1 Tóm tắt các thuộc tính chính** 7](#_Toc211378393)

[**4.2 Demo thuộc tính width, height, fit, alignment** 8](#_Toc211378394)

[**5.** **Tổng quan và so sánh Icons trong Flutter: Material vs Cupertino** 8](#_Toc211378395)

[**5.1 Khái quát về hệ thống Icon Flutter** 8](#_Toc211378396)

[**5.2 Bảng so sánh** 8](#_Toc211378397)

[**6.** **Xử lý trạng thái loading và lỗi khi hiển thị Image** 10](#_Toc211378398)

[**6.1 LoadingBuilder và errorBuilder trong Image.network** 10](#_Toc211378399)

[**6.2 Một số cách xử lí lỗi nâng cao** 11](#_Toc211378400)

**NỘI DUNG**

**IMAGE VÀ ICONS TRONG FLUTTER**

**Giới thiệu**

Trong phát triển ứng dụng đa nền tảng với Flutter, Image (hình ảnh) và Icons (biểu tượng) là hai loại widget cơ bản tạo nên phần nhìn và trải nghiệm người dùng mang tính hấp dẫn và đồng nhất trên cả Android lẫn iOS. Hiểu rõ cách hoạt động, so sánh các loại ImageProvider (AssetImage vs NetworkImage), quản lý assets, cũng như phân biệt và sử dụng đúng các bộ icon (Material Icons, Cupertino Icons) là nền tảng quan trọng để xây dựng ứng dụng chất lượng cao, tối ưu hoá hiệu suất và đảm bảo giao diện nhất quán.

1. **Tổng quan về Image và Icon trong Flutter**

## **1.1 Tầm quan trọng của hình ảnh và biểu tượng**

Hình ảnh và icon là hai yếu tố trực quan tăng cường trải nghiệm người dùng (UX) một cách mạnh mẽ. Việc sử dụng đúng hình ảnh và biểu tượng không chỉ giúp truyền đạt thông tin nhanh hơn mà còn tạo dấu ấn về mặt thương hiệu, giúp giao diện thân thiện, dễ nhận diện và hiện đại.

* Image: Dùng để hiển thị hình ảnh từ nhiều nguồn (assets bên trong app hoặc từ Internet).
* Icon: Hiển thị các biểu tượng, có thể chọn phong cách Android (Material Design) hoặc iOS (Cupertino).

Trong Flutter, mọi giao diện đều xây dựng từ các widget. Image và Icon đều là widget độc lập, có thể sử dụng linh hoạt và lồng ghép trong mọi layout (Row, Column, Stack, ...), cũng như tuỳ biến với nhiều thuộc tính.

## **1.2 Các loại nguồn hình ảnh**

- AssetImage: Hình ảnh đóng gói cục bộ cùng ứng dụng (assets).

- NetworkImage: Hình ảnh tải động từ Internet qua URL.

- FileImage: Hình ảnh từ file trên thiết bị (ít dùng với demo static).

- MemoryImage: Hình ảnh từ bộ nhớ dạng bytes.

Thực tế, AssetImage và NetworkImage là phổ biến nhất khi khai báo giao diện ứng dụng.

1. **So sánh NetworkImage và AssetImage trong Flutter**

## **2.1 Định nghĩa**

**NetworkImage:** Ảnh tải động từ các URL trực tuyến. Phù hợp hiển thị dữ liệu động, avatar người dùng, ảnh sản phẩm cập nhật thường xuyên. Có thể gặp vấn đề kết nối, tốc độ mạng và cần xử lý lỗi cũng như loading cho trải nghiệm tốt.

**AssetImage:** Các file ảnh được lưu trong thư mục assets (ví dụ: assets/images/), đóng gói vào trong ứng dụng. Dùng cho các hình cố định: logo, hình nền, icon riêng của app,... AssetImage truy xuất hình ảnh nhanh chóng không cần Internet.

**2.2 Bảng so sánh NetworkImage với AssetImage**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | NetworkImage | AssetImage |
| Nguồn ảnh | Tải từ Internet qua URL | Lưu trữ cục bộ trong assets dự án |
| Cần Internet | Có | Không |
| Hiệu suất | Phụ thuộc tốc độ mạng, có thể chậm | Rất nhanh do đọc trực tiếp từ thiết bị |
| Cấu hình pubspec.yaml | Không yêu cầu cấu hình thêm | Bắt buộc khai báo đường dẫn trong pubspec.yaml |
| Xử lí lỗi | Phải xử lý lỗi mạng, URL sai hoặc timeout | Thường chỉ lỗi khi cấu hình asset sai |
| Dùng cho | Ảnh động, nội dung online, cập nhật | Ảnh tĩnh, logo, placeholder, icon app |
| Khả năng loading | Có (cần loadingBuilder/errorBuilder) | Không cần loading, rất nhanh |

**2.3 Thực tiễn ứng dụng**

Widget Image, đây là một trong những widget thường dùng nhất trong Flutter để hiển thị hình ảnh. Với Image, ta có thể lấy ảnh từ nhiều nguồn khác nhau, tuy nhiên 2 dạng phổ biến nhất là Image.asset và Image.network. AssetImage giúp giảm phụ thuộc Internet, tối ưu hiệu suất và tránh vấn đề bảo mật quyền truy cập. Về phía NetworkImage, sự linh hoạt trong thay đổi hình ảnh theo backend mở ra tiềm năng phát triển ứng dụng mang tính kết nối: mạng xã hội, thương mại điện tử, news feed,...

Ví dụ dùng Image.asset:



Ví dụ dùng Image.network:

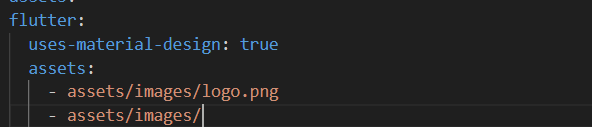


1. **Cấu hình và quản lý assets trong pubspec.yaml**

**3.1 Nguyên lý quản lý assets của flutter**

Assets (bao gồm hình ảnh, JSON, audio...) được quản lý bởi hệ pubspec.yaml. Tất cả tệp assets muốn dùng phải khai báo đúng và đúng thụt lề trong file này. Nếu sai cấu trúc hoặc thiếu thụt lề dễ gặp lỗi khi build app.

Ví dụ:

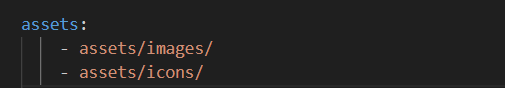


* Dùng dấu / để bao toàn bộ thư mục.
* Đảm bảo khai báo đúng vị trí, đúng mức thụt lề.

Lưu ý: Tên thư mục hoàn toàn tự đặt, thường dùng assets/images/ rõ ràng và quản lý dễ hơn. Việc này cực kỳ quan trọng khi làm project nhóm hoặc dự án lớn.

**3.2 Cách bao trùm nhiều hình ảnh**

Chỉ định tên thư mục với dấu / cuối để lấy toàn bộ file trong thư mục (không lấy file trong subfolder con, muốn lấy phải khai báo tiếp):



## **3.3 Quy trình cập nhật asset mới**

1. Tạo/coppy file ảnh vào thư mục assets.

2. Khai báo (hoặc đảm bảo đã khai báo) đúng đường dẫn asset trong pubspec.yaml.

3. Chạy lệnh flutter pub get để update asset bundle cho dự án.

Lưu ý: Chạy lại lệnh flutter pub get sau khi thêm/sửa đổi asset.

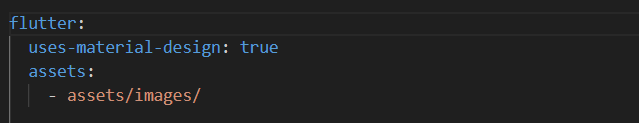
**3.4 Đa dạng hoá độ phân giải hình ảnh (đáp ứng màn hình sắc nét)**

Flutter hỗ trợ auto load ảnh phù hợp density của thiết bị qua các thư mục: 1.0x, 2.0x, 3.0x,... Ví dụ:

* assets/images/logo.png (phiên bản gốc - mdpi)
* assets/images/2.0x/logo.png (xhdpi)
* assets/images/3.0x/logo.png (xxhdpi)

Khai báo asset chỉ cần tên asset gốc, Flutter tự động phát hiện các phiên bản density khác.

**3.5 Demo Khai báo và sử dụng assets trong pubspec.yaml**



Trong dart:



1. **Thuộc tính quan trọng của Image widget: width, height, fit, alignment**

**4.1 Tóm tắt các thuộc tính chính**

- Width: Chiều rộng hình (double, pixel logic)

- Height: Chiều cao hình (double, pixel logic)

- Fit: Cách hiển thị/cắt/căn chỉnh hình trong khung (BoxFit)

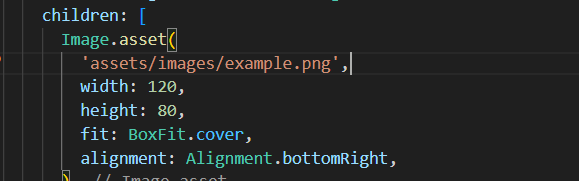
- Alignment: Căn lề hình ảnh trong khung cha (Alignment.center, .topLeft,...)

BoxFit có các kiểu phổ biến:

* cover: co giãn che phủ toàn bộ khung, có thể cắt hình.
* contain: vừa khung, không bị cắt, có thể có viền trắng.
* fill: bóp méo để điền đầy khung.
* fitWidth/fitHeight: căn theo chiều rộng/chiều cao.
* scaleDown: giống contain nhưng không phóng lớn hơn gốc.

Alignment mặc định là center, thay đổi bằng các giá trị như .topLeft, .bottomRight tùy layout.

## **4.2 Demo thuộc tính width, height, fit, alignment**



Tùy layout, đặt width/height giúp đảm bảo hình hiển thị đồng nhất trên các thiết bị, tránh bị vượt biên hoặc méo hình. Khi cần bố cục linh hoạt (ví dụ bố trí nhiều ảnh trong Row/Column), chú ý dùng Expanded, Flexible kết hợp layout tốt.

1. **Tổng quan và so sánh Icons trong Flutter: Material vs Cupertino**

**5.1 Khái quát về hệ thống Icon Flutter**

Flutter hỗ trợ hai nhóm icon chính, tương ứng với hai hệ thống thiết kế:

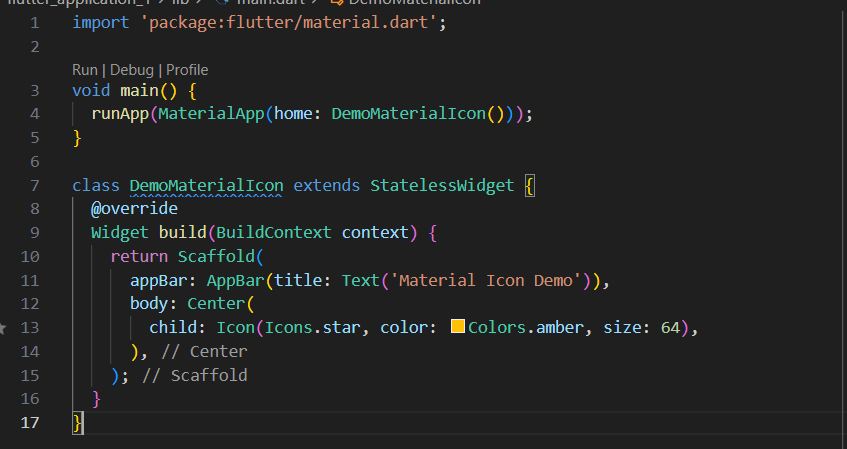
* **Material Icons**: Chuẩn Material Design của Google, đa dạng cho Android/web/cross-platform.
* **Cupertino Icons**: Theo tiêu chuẩn Human Interface iOS của Apple, tối ưu cho iOS hoặc giao diện look-like iOS.

Ngoài ra, các icon ngoài (FontAwesome, LineAwesome, CryptoFont,...) hoặc icon tự tạo dựa vào ImageIcon, SVG cũng có thể dễ dàng tích hợp.

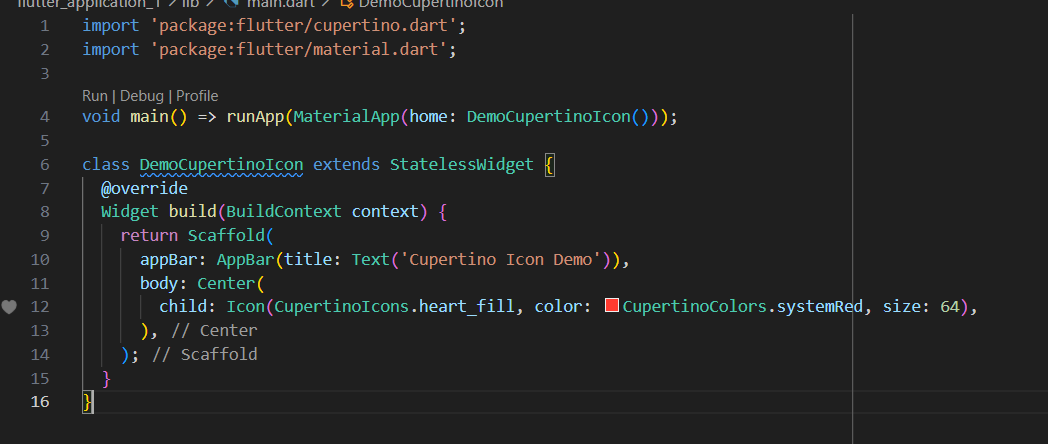
## **5.2 Bảng so sánh**

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Material Ikons | Cupertino Icons |
| Phong cách | Material Design (Android, Google) | iOS Design (Apple, tối giản, flat) |
| Widget sử dụng | Icons | CupertinoIcons |
| Thư viện | Material.dart | Cupertino.dart  Cupertino\_icons |
| Tương thích | Android, Web, iOS | Chủ yếu iOS, trên Android hơi khác font |
| Số lượng và đa dạng | Rất nhiều, nhiều biến thể | Ít hơn, chủ yếu biểu tượng thuần túy iOS |
| Tùy biến | Đổi màu, cỡ, semantic label, blend mode | Đổi màu, cỡ, semantic label |
| Ví dụ sử dụng | Icon(Icons.favorite) | Icon(cupertino.heart) |
| Cập nhật | Thường xuyên | Ít cập nhật hơn |

Ví dụ Meterial icons:



Ví dụ Cupertino icons:

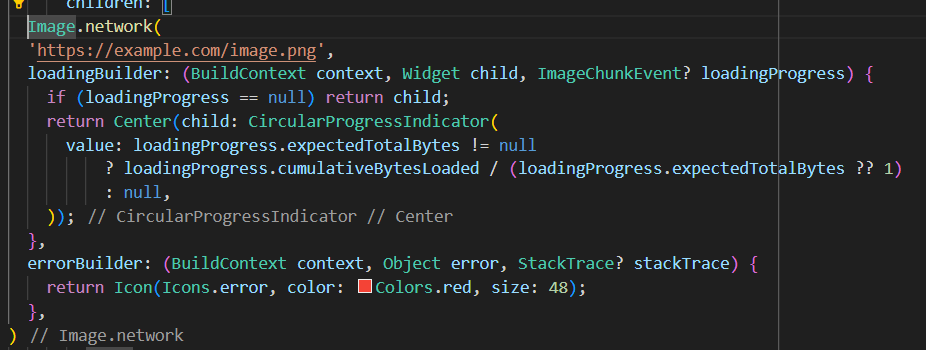


1. **Xử lý trạng thái loading và lỗi khi hiển thị Image**

## **6.1 LoadingBuilder và errorBuilder trong Image.network**

Image.network hỗ trợ hai thuộc tính nổi bật:

* **loadingBuilder**: Widget hiển thị trong quá trình ảnh đang tải về (thường dùng CircularProgressIndicator).
* **errorBuilder**: Widget hiển thị khi tải ảnh thất bại (ví dụ icon lỗi, Text thông báo).



Ý nghĩa:

* loadingProgress trả về null nếu ảnh tải xong, nếu không sẽ cập nhật tiến độ tải.
* errorBuilder rất quan trọng để tránh crash app khi ảnh sai URL hoặc mất mạng.

Tính năng nổi bật: Chỉ có Image.network mới có loadingBuilder, Image.asset không cần vì luôn load gần như tức thì.

## **6.2 Một số cách xử lí lỗi nâng cao**

- Sử dụng CachedNetworkImage để caching ảnh, cung cấp placeholder/error widget tiện lợi.

- Có thể tận dụng FadeInImage để vừa loading ảnh vừa thể hiện hiệu ứng mượt mà khi hiển thị.

**TÀI LIỆU THAM KHẢO**

[1] [VNCODER](https://vncoder.vn/bai-hoc/widget-trong-flutter-223)

[2] Vu Van Hanh, “Flutter : Project structure - localization, dependencies and assets”, VIBLO, 2018

[3] Tien Tran, “Flutter icon: Định nghĩa, Phân loại và Chi tiết cách áp dụng A-Z”, ITviec blog, 2024