**ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A blue circle with text

AI-generated content may be incorrect.**

CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

Tìm hiểu về công nghệ Flutter

**Giảng viên hướng dẫn: TS. Đỗ Ngọc Như Loan**

**Thực hiện:** Lê Đức Nam – 3121411135 – DCT121C2

<MSSV> - LỚP

TP Hồ Chí Minh - tháng 4 năm 2025

Lời nói đầu

Lời cảm ơn

MỤC LỤC

DANH MỤC HÌNH ẢNH

# Chương 1: Tổng quan về Flutter

## 1.1 Nguyên nhân và lịch sử ra đời

Lập trình cross platform đang ngày càng trở nên phổ biến. Đặc biệt, với sự xuất hiện của React Native (RN) vào năm 2015 đã mở ra xu hướng xây dựng ứng dụng đa nền tảng chỉ bằng một lần code. Các công ty lớn như UberEats, Discord và Facebook đã chuyển sang React Native – như một phần tất yếu của sự dịch chuyển công nghệ.

Google cũng không chịu thua. Họ đã nhận thấy độ phủ sóng của React Native. Sau 2 năm nghe ngóng, Google cho ra mắt phiên bản alpha của Flutter. Giao diện người dùng mới này đã được khởi chạy vào năm 2017 và dự định tạo các ứng dụng gốc chất lượng cao trên các phiên bản iOS và Android trong thời gian rất ngắn.

## 1.2 Flutter là gì?

A blue arrows on a black background

AI-generated content may be incorrect.

Flutter là một bộ SDK đa nền tảng có thể hoạt động trên iOS và Android do

Google phát triển được sử dụng để tạo ra các ứng dụng dành cho di động (native app).

## 1.3 Nguyên lý hoạt động và các tính năng nổi bật

### 1.3.1 Nguyên lý hoạt động

Flutter gồm 2 thành phần quan trọng:

+ Một SDK (Software Development Ki): Một bộ sưu tập các công cụ sẽ giúp bạn phát triển các ứng dụng của mình.

+ Một Framework (UI Library based on widgets): Một tập hợp các thành phần giao diện người dùng (UI) có thể tái sử dụng (button, text inputs, slider, v.v.) giúp bạn có thể cá nhân hóa tùy theo nhu cầu của riêng mình.

Nói chung, tạo một ứng dụng di động là một công việc rất phức tạp và đầy thử thách. Có rất nhiều framework có sẵn, cung cấp các tính năng tuyệt vời để phát triển các ứng dụng di động. Để phát triển các ứng dụng dành cho thiết bị di động, Android cung cấp một framework gốc dựa trên ngôn ngữ Java và Kotlin, trong khi iOS cung cấp một framework dựa trên ngôn ngữ Objective-C/Swift.

Vì vậy, chúng ta cần hai ngôn ngữ và framework khác nhau để phát triển ứng dụng cho cả hai hệ điêu hành. Ngày nay, để khắc phục sự phức tạp này, có một số framework đã được giới thiệu hỗ trợ cả hệ điều hành cùng với các ứng dụng dành cho máy tính đề bàn. Những loại framework này được gọi là công cụ phát triển đa nền tảng.

### 1.3.2 Các tính năng của Flutter

Flutter cung cấp các phương pháp dễ dàng và đơn giản để bắt đầu xây dựng các ứng dụng dành cho thiết bị di động và máy tính để bàn đẹp mắt với một bộ thiết kế material design và widget phong phú. Ở đây, chúng ta sẽ thảo luận về các tính năng chính của nó để phát triển framework di động.

A diagram of a plant

AI-generated content may be incorrect.

**code nguồn mở(Open-Source:):** Flutter là một framework code nguồn mở và miễn phí để phát triển các ứng dụng di động.

**Đa nền tảng(Cross-platform):** Tính năng này cho phép Flutter viết code một lần, duy trì và có thể chạy trên các nền tảng khác nhau. Nó tiết kiệm thời gian, công sức và tiền bạc của các nhà phát triển.

**Tải lại nóng(Hot Reload):** Bất cứ khi nào nhà phát triển thực hiện thay đổi trong code, thì những thay đổi này có thể được nhìn thấy ngay lập tức với Tải lại nóng. Nó có nghĩa là những thay đổi hiển thị ngay lập tức trong chính ứng dụng. Đây là một tính năng rất tiện dụng, cho phép nhà phát triển sửa các lỗi ngay lập tức.

**Các tính năng và SDK gốc có thể truy cập(Accessible Native Features and SDKs):** Tính năng này cho phép quá trình phát triển ứng dụng dễ dàng và thú vị thông qua code gốc của Flutter, tích hợp bên thứ ba và các API nền tảng. Do đó, chúng tôi có thể dễ dàng truy cập SDK trên cả hai nền tảng.

**Code tối thiểu(Minimal code):** Ứng dụng Flutter được phát triển bởi ngôn ngữ lập trình Dart, sử dụng biên dịch JIT và AOT để cải thiện thời gian khởi động tổng thể, hoạt động và tăng tốc hiệu suất. JIT nâng cao hệ thống phát triển và làm mới giao diện người dùng mà không cần nỗ lực thêm vào việc xây dựng hệ thống mới.

**Widget:** framework công tác Flutter cung cấp các widget có khả năng phát triển các thiết kế cụ thể có thể tùy chỉnh. Quan trọng nhất, Flutter có hai bộ widget: Material Design và các widget Cupertino giúp mang lại trải nghiệm không có trục trặc trên tất cả các nền tảng.

## 1.4 Kiến trúc Flutter

Kiến trúc Flutter chủ yếu bao gồm bốn thành phần.

Động cơ Flutter (Flutter Engine)

Flutter Engine là trái tim của framework, được viết chủ yếu bằng C++ và chịu trách nhiệm xử lý các tác vụ cấp thấp. Nó cung cấp nền tảng để chạy ứng dụng Flutter, bao gồm quản lý giao diện đồ họa, xử lý đầu vào, kết nối với hệ điều hành cơ bản (iOS, Android, web, máy tính để bàn) và điều phối các hoạt động như vẽ giao diện lên màn hình. Flutter Engine sử dụng Skia - một thư viện mã nguồn đồ họa 2D mở - để hiển thị giao diện người dùng một cách nhanh chóng và mượt mà. Đây là thành phần đảm bảo hiệu suất và khả năng nền tảng của Flutter.

Thư viện nền tảng (Foundation Library)

Thư viện này được viết bằng Dart, cung cấp các công cụ cơ sở và các khối xây dựng cần thiết để phát triển ứng dụng Flutter. Nó bao gồm các lớp và cơ sở như quản lý trạng thái, xử lý sự kiện, hoạt hình (hình động) và các tiện ích chung (tiện ích). Foundation Library đóng vai trò như một cầu nối giữa Flutter Engine và các thành phần cấp cao hơn (như Widget), giúp người lập trình dễ dàng thao tác với framework mà không cần đi sâu vào chi tiết kỹ thuật phức tạp.

Vật dụng (Widgets)

Trong Flutter, mọi thứ đều là một widget, đó là khái niệm cốt lõi của framework. Widget trong Flutter về cơ bản là một thành phần giao diện người dùng ảnh hưởng và kiểm soát chế độ xem và giao diện của ứng dụng. Nó đại diện cho một mô tả bất biến về một phần của giao diện người dùng và bao gồm đồ họa, văn bản, hình dạng và animation được tạo bằng các widget. Các widget tương tự như các thành phần React.

Trong Flutter, ứng dụng tự nó là một widget chứa nhiều widget con. Điều đó có nghĩa rằng ứng dụng là tiện ích con cấp cao nhất và giao diện người dùng của nó được xây dựng bằng cách sử dụng một hoặc nhiều tiện ích con, bao gồm các tiện ích con phụ. Tính năng này giúp bạn tạo một giao diện người dùng phức tạp rất dễ dàng.

A diagram of a software company

AI-generated content may be incorrect.

Thiết kế các widget cụ thể (Design Specific Widgets)

**Material Widgets** : Dựa trên ngôn ngữ thiết kế Material Design của Google, phù hợp với phong cách Android.

**Cupertino Widgets** : Dựa trên thiết kế ngôn ngữ của Apple (Nguyên tắc giao diện con người), phù hợp với phong cách iOS.

## 1.5 Một số loại Widgets của Flutter thường gặp

**Widget Flutter**

Widgets là nền tảng của giao diện người dùng trong Flutter. Chúng được chia thành nhiều nhóm khác nhau dựa trên chức năng, nhưng về cơ bản, mọi thứ trong Flutter đều là widget – từ các thành phần giao diện nhỏ nhất đến toàn bộ bố cục ứng dụng. Widgets hoạt động theo mô hình cây (widget tree), trong đó các widget cha chứa các widget con, tạo nên cấu trúc giao diện. Hai loại chính là:

* **StatelessWidget**: Không có trạng thái thay đổi, ví dụ: hiển thị văn bản tĩnh hoặc nút không thay đổi.
* **StatefulWidget**: Có thể thay đổi trạng thái, ví dụ: nút chuyển đổi màu sắc khi nhấn hoặc biểu mẫu nhập liệu.  
  Ví dụ cụ thể:
  + Text: Hiển thị văn bản.
  + Icon: Hiển thị biểu tượng.
  + ElevatedButton: Nút có thể nhấn với hiệu ứng nổi.

A diagram of a company

AI-generated content may be incorrect.

**Widget hiển thị**

Các widget hiền thị có liên quan đến dữ liệu đầu vào và đâu ra của người dùng.

Một số loại quan trọng của widget con này là:

Text: Text là một widget con trong Flutter cho phép chúng ta hiển thị một chuỗi Text với một dòng duy nhất trong ứng dụng của chúng ta . Tùy thuộc vào các ràng buộc về bố cục, chúng ta có thể ngắt chuỗi trên nhiều dòng hoặc tất cả có thể được hiển thị trên cùng một dòng. Nếu chúng ta không chỉ định bất kỳ kiểu nào cho widget Text, nó sẽ sử dụng kiểu lớp DefaultTextStyle gần nhất.

Button: là phần tử điều khiển đồ họa cung cấp cho người dùng kích hoạt một sự kiện như thực hiện hành động, lựa chọn, tìm kiếm mọi thứ, v.v. Chúng có thể được đặt ở bất kỳ đâu trong giao diện người dùng như hộp thoại, biểu mẫu, thẻ, thanh công cụ, v.v.

Image: Widget con này giữ hình ảnh có thể tìm nạp hình ảnh từ nhiều nguồn như từ thư mục nội dung hoặc trực tiếp từ URL. Nó cung cấp nhiều hàm tạo để tải hình ảnh, được đưa ra dưới đây:

* Image
* Asset
* File
* Memory
* Network

**Widget ẩn**

Các widget vô hình có liên quan đến cách bồ trí và kiểm soát các widget. Nó cung cấp việc kiểm soát cách các widget thực sự hoạt động và cách chúng sẽ hiển thị trên màn hình. Một số loại widget quan trọng là:

**Column** là một widget bố cục vô hình dùng để sắp xếp các widget con theo **chiều dọc** (từ trên xuống dưới). Nó không có giao diện trực quan riêng, mà chỉ cung cấp cấu trúc để tổ chức các thành phần khác.

**Ứng dụng:**

* Dùng để tạo danh sách các phần tử theo chiều dọc, như form nhập liệu, menu dọc, hoặc các đoạn văn bản tuần tự.

**Stack** là một widget bố cục vô hình khác, cho phép các widget con **chồng lên nhau** trong không gian 2D. Không giống như Column (chỉ xếp dọc) hay Row (chỉ xếp ngang), Stack linh hoạt hơn, phù hợp với các giao diện phức tạp như lớp phủ (overlay), hình ảnh nền với văn bản phía trên, hoặc các hiệu ứng xếp chồng.

**Ứng dụng:**

* Tạo giao diện lớp phủ (overlay) như hình ảnh với chú thích.
* Xây dựng các hiệu ứng giao diện phức tạp, ví dụ: biểu tượng nổi trên hình nền hoặc các thành phần chồng chéo.

## 1.6 Tìm hiểu về Database trong Flutter

Flutter là một framework phát triển giao diện người dùng (UI), nên việc quản lý dữ liệu phụ thuộc vào các thư viện Dart (ngôn ngữ lập trình của Flutter) và tích hợp với các hệ thống cơ sở dữ liệu. Có hai loại database chính được sử dụng trong Flutter:

* **Cơ sở dữ liệu cục bộ (Local Databases)**: Lưu trữ dữ liệu trực tiếp trên thiết bị, phù hợp với ứng dụng offline hoặc khi không cần đồng bộ dữ liệu qua mạng.
* **Cơ sở dữ liệu đám mây (Cloud Databases)**: Lưu trữ dữ liệu trên máy chủ từ xa, hỗ trợ đồng bộ thời gian thực và chia sẻ dữ liệu giữa các thiết bị.

Flutter hỗ trợ cả hai mô hình này thông qua các gói (package) trên pub.dev, nơi cộng đồng phát triển cung cấp nhiều giải pháp database khác nhau.

**Cơ sở dữ liệu SQLite**

SQLite là một thư viện phần mềm cơ sở dữ liệu phổ biến cung cấp hệ thống quản lý cơ sở dữ liệu quan hệ để lưu trữ cục bộ / máy khách. Nó là một công cụ cơ sở dữ liệu nhẹ và được kiểm tra theo thời gian và chứa các tính năng như công cụ cơ sở dữ liệu SQL giao dịch độc lập, không cần máy chủ, không cấu hình.

Flutter SDK không hỗ trợ SQLite trực tiếp. Thay vào đó, nó cung cấp một plugin sqflite, thực hiện tất cả các hoạt động trên cơ sở dữ liệu tương tự như thư viện SQLite. SQLite cung cấp hầu hết các chức năng cốt lõi liên quan đền cơ sở dữ liệu như sau:

- Nó tạo hoặc mở cơ sở dữ liệu SQLite.

- Nó có thể thực thi các câu lệnh SQL một cách dễ dàng.

- Nó cũng cung cấp một phương pháp truy vấn nâng cao để lấy thông tin từ cơ sở dữ liệu SQLite.

**Fire – NoSQL lưu trữ online**

FireBase là một dạng database lưu trữ theo cách truyền thống. Mọi lưu trữ data trong bộ sưu tập giống như bảng trong database truyền thống. Tài liệu được lưu trữ trong các bộ sưu tập này. Các kiều lưu trữ data như string, int...Chúng cũng có thể được liên kết đến những tài liệu khác. Mặc dù FireBase không liên kết hoàn toàn, nhưng bạn vẫn có thể tạo được sự liên kết với tài liệu của mình.

Quy trình thiết lập cho Firebase khá liên quan so với các tùy chọn trên thiết bị khác, như Moor hoặc Hive, nhưng bạn vẫn có được sự đồng bộ hóa dữ liệu giữa máy khách và máy chủ. Điều này có nghĩa là nếu bạn có nhiều khách hàng với một ứng dụng và tất cả họ đều tương tác với cùng một dữ liệu, thì dữ liệu này có thể được giữ đồng bộ giữa các khách hàng này. Ngoài ra, thiết lập này được trình bày khá tốt trong Google Codelab.

## 1.7 Ưu nhược điểm

Ưu điểm:

* Nó làm cho quá trình phát triển ứng dụng cực kỳ nhanh chóng vì tính năng tải lại nóng. Tính năng này cho phép chúng tôi thay đổi hoặc cập nhật code được phản ánh ngay sau khi các thay đổi được thực hiện.
* Nó cung cấp trải nghiệm cuộn mượt mà và liền mạch khi sử dụng ứng dụng mà không bị treo hoặc cắt nhiều, giúp chạy ứng dụng nhanh hơn so với các framework phát triển ứng dụng dành cho thiết bị di động khác.
* Flutter làm giảm thời gian và nỗ lực kiểm tra. Như chúng ta đã biết, các ứng dụng là đa nền tảng, do đó người thử nghiệm không cần phải luôn chạy cùng một nhóm thử nghiệm trên các nền tảng khác nhau cho cùng một ứng dụng.
* Nó có giao diện người dùng tuyệt vời vì nó sử dụng tiện ích tập trung vào thiết kế, các công cụ phát triển cao, API nâng cao và nhiều tính năng khác.
* Nó tương tự như một framework phản ứng trong đó các nhà phát triển không cần cập nhật nội dung giao diện người dùng theo cách thủ công.
* Nó phù hợp với các ứng dụng MVP (Sản phẩm khả thi tối thiểu) vì quá trình phát triển nhanh chóng và tính chất đa nền tảng của nó.

Nhược điểm:

* Flutter là một ngôn ngữ tương đối mới cần được hỗ trợ tích hợp liên tục thông qua việc duy trì các tập lệnh.
* Nó cung cấp quyền truy cập rất hạn chế vào các thư viện SDK. Nó có nghĩa là một nhà phát triển không có nhiều chức năng để tạo một ứng dụng di động. Các loại chức năng như vậy cần được phát triển bởi chính nhà phát triển Flutter.
* Các ứng dụng Flutter không hỗ trợ trình duyệt. Nó chỉ hỗ trợ nền tảng Android và iOS.
* Nó sử dụng lập trình Dart để viết code, vì vậy một nhà phát triển cần phải học các công nghệ mới. Tuy nhiên, nó rất dễ học đối với các nhà phát triển.

So sánh với công nghệ khác

A white and blue logo

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
|  | Flutter | React Native |
| Khả năng dùng lại code | Flutter cho phép overwriting code. Nếu bạn có kế hoạch cho việc sử dụng lại code thì Flutter là tùy chọn tốt nhất để có thể tối ưu được việc lựa sử dụng lại code. | React Native cho phép bạn sử dụng lại code, nhưng điều này lại bị giới hạn trong một vài components cơ bản. Để có thể định nghĩa các style cho nền tảng React Native sẽ mất khá nhiều thời gian. |
| Các thư viện Third Party | Có nhiều các third-party packages đang được sử dụng và đang ngày càng được phát triển, và chúng thực sự rất hữu dụng. | Từ khi React Native trở lên phổ biến, đã có rất nhiều các third-party packages được phát triển và được sử dụng rất nhiều trong ứng dụng, chúng có thể được thêm bớt một cách linh động trong ứng dụng của bạn. |
| Độ phổ biến | Với khoảng 30k Github stars, Flutter đã trở thành trends trong việc phát triển. Như Google đã giới thiệu về framework này. Nó chắc chắn muốn trở thành một nền tảng phổ biến trong sự lựa chọn của các developer. | Với 65k Github stars, React Native có lượng developer sử dụng đang nhiều hơn bởi vì lượng developer sử dụng JavaScript rất dễ dàng để sử dụng với các thư viện của React. |
| Cộng đồng phát triển | Từ khi Flutter mới được giới thiệu, đã có một lượng lớn các bài viết hướng dẫn được đưa lên online, trong đó có rất nhiều các tài nguyên hữu ích để có thể bắt đầu viết ứng dụng đầu tiên trên mobile app. | React Native đã được phổ biến từ lâu, vì thế cho nên nó cũng có rất nhiều các cộng đồng để hỗ trợ online. Những kỹ năng của các lập trình viên có kinh nghiệm trong JavaScript chắc chắn sẽ đưa ra giải pháp cho bất kỳ vấn đề nào trong quá trình phát triển của bạn. |

Ứng dụng thực tế

Flutter, với khả năng phát triển ứng dụng đa nền tảng (cross-platform) nhanh chóng và hiệu quả, đã được sử dụng trong nhiều ứng dụng thực tế trên toàn cầu. Dưới đây là một số ví dụ nổi bật về các ứng dụng thực tế được xây dựng bằng Flutter, cùng với mô tả ngắn gọn về cách chúng tận dụng framework này:

**Google Ads**

* **Mô tả**: Ứng dụng Google Ads cho phép người dùng quản lý chiến dịch quảng cáo, theo dõi hiệu suất và điều chỉnh ngân sách ngay trên thiết bị di động.
* **Ứng dụng Flutter**: Flutter giúp Google Ads duy trì giao diện nhất quán trên iOS và Android, đồng thời đảm bảo hiệu suất mượt mà khi hiển thị dữ liệu thời gian thực.

**Grab (GrabFood)**

* **Mô tả**: Grab, một siêu ứng dụng Đông Nam Á, sử dụng Flutter trong một số module như GrabFood để đặt món ăn.
* **Ứng dụng Flutter**: Flutter cho phép GrabFood tối ưu hóa giao diện đặt hàng, hiển thị danh sách nhà hàng và theo dõi đơn hàng một cách mượt mà.

**Tencent (WeSure)**

* **Mô tả**: WeSure là ứng dụng bảo hiểm của Tencent, cung cấp các dịch vụ bảo hiểm cá nhân hóa.
* **Ứng dụng Flutter**: Flutter giúp WeSure triển khai nhanh ứng dụng trên cả iOS và Android, với giao diện thân thiện và khả năng xử lý dữ liệu thời gian thực.

Xu hướng phát triển

Flutter đang phát triển mạnh mẽ với xu hướng mở rộng hỗ trợ đa nền tảng (desktop, web, nhúng), tối ưu hiệu suất (giảm kích thước, tăng tốc render), tích hợp AI/ML, và nâng cao hệ sinh thái cộng đồng. Nó cũng hướng tới tích hợp sâu hơn với Firebase/Google Cloud, hỗ trợ low-code cho người không chuyên, cạnh tranh vượt trội với các framework như React Native, và ứng dụng vào các lĩnh vực mới như y tế, giáo dục, game nhẹ. Flutter hứa hẹn trở thành tiêu chuẩn cho phát triển đa nền tảng trong tương lai.

Chương 2: Xây dựng ứng dụng minh họa

Chương 3: Tổng kết

Tài liệu tham khảo