BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH : CÔNG NGHỆ THÔNG TIN

Sinh viên **:** Đỗ Thế Hiệp

Giảng viên hướng dẫn: TS. Đỗ Văn Chiểu

HẢI PHÒNG – 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

TÌM HIỂU VỀ FLUTTER VÀ ỨNG DỤNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên **:** Đỗ Thế Hiệp

Giảng viên hướng dẫn: TS. Đỗ Văn Chiểu

HẢI PHÒNG – 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Đỗ Thế Hiệp **Mã SV**: 1712111001

Lớp : CT2101C

Ngành : Công nghệ thông tin

Tên đề tài: Tìm hiểu Flutter và ứng dụng

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

- Tìm hiểu về Flutter, cài đặt, biên dịch hệ thống.
- Tìm hiểu về Website, viết ứng dụng đọc website trên di động, và biên dịch hệ thống lên các thiết bị Android và iOS

2. Các tài liệu, số liệu, thiết bị cần thiết

- Sách Flutter Fast: 65 Example Apps.
- Khóa học Flutter cơ bản của Cafedev.
- Thiết bị:
    - Máy tính chạy hệ điều hành macOS.
    - Máy tính chạy hệ điều hành Windows.
    - Điện thoại di động chạy hệ hiều hành Android.
    - Điện thoại di động chạy hệ điều hành iOS.

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : **Đỗ Văn Chiểu**

Học hàm, học vị : Tiến sĩ

Cơ quan công tác : Trường Đại học Quản lý và Công nghệ Hải Phòng

Nội dung hướng dẫn:

+ Tìm hiểu Flutter, cài đặt, biên dịch hệ thống.

+ Tìm hiểu 1 website, viết ứng dụng đọc website trên di động.

Đề tài tốt nghiệp được giao ngày 12 tháng 04 năm 2021

Yêu cầu phải hoàn thành xong trước ngày 03 tháng 07 năm 2021

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	*Giảng viên hướng dẫn*

Hải Phòng, ngày.....tháng.....năm 2021

TRƯỞNG KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: **Đỗ Văn Chiểu**

Đơn vị công tác: Khoa Công nghệ thông tin, Trường Đại học Quản lý và Công nghệ Hải Phòng

Họ và tên sinh viên: **Đỗ Thế Hiệp** Ngành: Công nghệ thông tin

Nội dung hướng dẫn:

- Tìm hiểu Flutter, cài đặt, biên dịch hệ thống.
- Tìm hiểu 1 website, viết ứng dụng đọc website trên di động, biên dịch hệ thống qua cá thiết bị iOS và Android

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

……....

……

……

2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T. T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…)

……

……

……

……

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Đạt ☐ Không đạt ☐ Điểm:……...

Hải Phòng, ngày.....tháng 07 năm 2021

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên: .....

Đơn vị công tác: .....

Họ và tên sinh viên: Đỗ Thế Hiệp Ngành: Công nghệ thông tin

Đề tài tốt nghiệp: Tìm hiểu Flutter và ứng dụng

1. Phần nhận xét của giảng viên chấm phản biện

……

……

……

……

……

……

2. Những mặt còn hạn chế

……

……

……

……

……

……

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm:…….

Hải Phòng, ngày.....tháng 07 năm 2021

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

MỤC LỤC

MỤC LỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ

Từ viết tắt	Nghĩa tiếng anh	Nghĩa tiếng việt
ECMA	ECMAScript	Một thương hiệu đặc tả ngôn ngữ kịch bản được tiêu chuẩn hóa bởi Ecma International thông qua ECMA-262 và ISO/IEC 16262
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
API	Application Programming Interface	Giao diện lập trình ứng dụng
JIT	Just In Time	Debug hàm nào chạy hàm đó thì ở đây nó sẽ viết đến đâu biên dịch ngay đến đấy
AOT	Ahead Of Time	Khi chạy chương trình, nó sẽ biên dịch từ đầu đến cuối
OOP	Object Oriented Programming	Lập trình hướng đối tượng
URL	Uniform Resource Locator	Định vị tài nguyên thống nhất. Cụ thể, URL là địa chỉ của một tài nguyên duy nhất trên Web.
IDE	Integrated Development Environment	Môi trường tích hợp dùng để viết code để phát triển ứng dụng.
Framework		một nền tảng lập trình và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows được phát triển bởi Microsoft
OEM	Original Equipment Manufacturer	Nhà sản xuất thiết bị gốc
Widgets		Tiện ích
HTML	Hypertext Markup Language	Nó giúp người dùng tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng
XHTML	Extensible HyperText Markup Language	Ngôn ngữ Đánh dấu Siêu văn bản Mở rộng

LỜI CẢM ƠN

Lời đầu tiên cho em gửi lời cảm ơn sâu sắc đến gia đình, người thân của em. Đã động viên, giúp đỡ, cổ vũ, tạo cho em thêm động lực để em có thể hoàn thành đồ án trong thời gian được giao.

Em xin chân thành cảm ơn đến các thầy cô Ban Giám Hiệu Trường Đại học Quản lý và Công Nghệ Hải Phòng, các thầy cô thuộc các Ban, Ngành của trường đã tạo mọi điều kiện để em có thể đăng kí được đồ án tốt nghiệp và hoàn thành.

Em xin chân thành cảm ơn các thầy giáo cô giáo trong Khoa Công nghệ thông tin giảng dạy cho em những kiến thức bổ ích trong vòng bốn năm qua, giúp đỡ, cung cấp cho em những kiến thức nền tảng để em có thể hoàn thành được đề tài tốt nghiệp.

Đặc biệt em xin chân thành cảm ơn thầy giáo, TS. Đỗ Văn Chiểu trong thời gian làm tốt nghiệp vừa qua, thầy đã giành nhiều thời gian và tâm huyết để hướng dẫn em thực hiện đề tài này.

Em xin cảm ơn các bạn, các anh, các chị đồng nghiệp đã giúp đỡ em có thêm những kiến thức nền tảng về lập trình, để em có thể hoàn thành tốt đề tài tốt nghiệp của em.

Dưới đây là kết quả của quá trình tìm hiểu và nghiên cứu mà em đã đạt được trong thời gian vừa qua. Mặc dù rất cố gắng và được thầy cô giúp đỡ nhưng do hiểu biết và kinh nghiệm của mình còn hạn chế nên có thể đây chưa phải là kết quả mà thầy cô mong đợi từ em. Em rất mong nhận được những lời nhận xét và đóng góp quý báu của thầy cô để bài luận văn của em được hoàn thiện hơn cũng như cho em thêm nhiều kinh nghiệm cho công việc sau này.

Em xin chân thành cảm ơn !

Hải Phòng, ngày.....tháng.....năm 2021

Sinh viên

Đỗ Thế Hiệp

LỜI NÓI ĐẦU

Với sự gia tăng đáng kinh ngạc của sự phát triển ứng dụng di động trong thập kỷ qua, iOS và Android đã trở thành những hệ điều hành hàng đầu. Điều này dẫn đến sự trỗi dậy của toàn bộ ngành công nghiệp với một số framework cho phép các nhà phát triển xây ứng dụng di động. Hầu hết các framework phát triển di động này được xây dựng để hỗ trợ phát triển ứng dụng kết hợp tương thích với cả iOS và Android. Google đã đưa ra một bộ công cụ phát triển phần mềm giao diện người dùng mã nguồn mở Flutter, được công bố lần đầu tiên vào năm 2015 và chính thức ra mắt vào năm 2017.

Để tạo ra 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 điều 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 để bàn. Những loại framework này được gọi là công cụ phát triển đa nền tảng .

Trong nội dung này, em xin trình bày về "Flutter và ứng dụng", để đáp ứng nhu cầu cần tìm hiểu về lập trình ứng dụng trên Mobile, dành cho mọi người có đam mê và sở thích về khả năng lập trình các ứng dụng tương thích trên iOS và Android. Đóng góp một phần nhỏ về kiến thức cơ bản dành cho những ai yêu thích muốn tìm hiểu đến Flutter.

Nội dung đồ án bao gồm ba chương:

Chương 1 Tìm hiểu về Flutter, giới thiệu về Flutter, các tính năng, kiến trúc, tổng quan của Flutter.

Chương 2 Tìm hiểu về Website, giới thiệu những điều cơ bản về website, cách thức hoạt động và phân loại trang web.

Chương 3 Ứng dụng thực nghiệm, xây dựng ứng dụng trên hai nền tảng là Android và iOS.

Hải Phòng, ngày.....tháng.....năm 2021

Sinh viên thực hiện

Đỗ Thế Hiệp

CHƯƠNG 1 TÌM HIỂU VỀ FLUTTER

1.1 Flutter là gì

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).

Flutter gồm 2 thành phần quan trọng:

- Một SDK (Software Development Kit): 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 điều 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 để bàn. Những loại framework này được gọi là công cụ phát triển đa nền tảng [1] .

Framework phát triển đa nền tảng có khả năng viết một code và có thể triển khai trên nhiều nền tảng khác nhau (Android, iOS và Máy tính để bàn). Nó tiết kiệm rất nhiều thời gian và nỗ lực phát triển của các nhà phát triển.

Có một số công cụ có sẵn để phát triển đa nền tảng, bao gồm các công cụ dựa trên web. Mỗi framework này có mức độ thành công khác nhau trong ngành công nghiệp di động. Gần đây, một framework công tác mới đã được giới thiệu trong họ phát triển đa nền tảng có tên là Flutter được phát triển từ Google.

Flutter là một bộ công cụ giao diện người dùng để tạo các ứng dụng nhanh, đẹp, được biên dịch nguyên bản cho thiết bị di động, web và máy tính để bàn với một ngôn ngữ lập trình và cơ sở code duy nhất. Nó là miễn phí và code nguồn mở. Ban đầu nó được phát triển từ Google và bây giờ được quản lý theo tiêu chuẩn ECMA. Ứng dụng Flutter sử dụng ngôn ngữ lập trình Dart để tạo ứng dụng.

Flutter chủ yếu được tối ưu hóa cho các ứng dụng di động 2D có thể chạy trên cả nền tảng Android và iOS. Chúng ta cũng có thể sử dụng nó để xây dựng các ứng

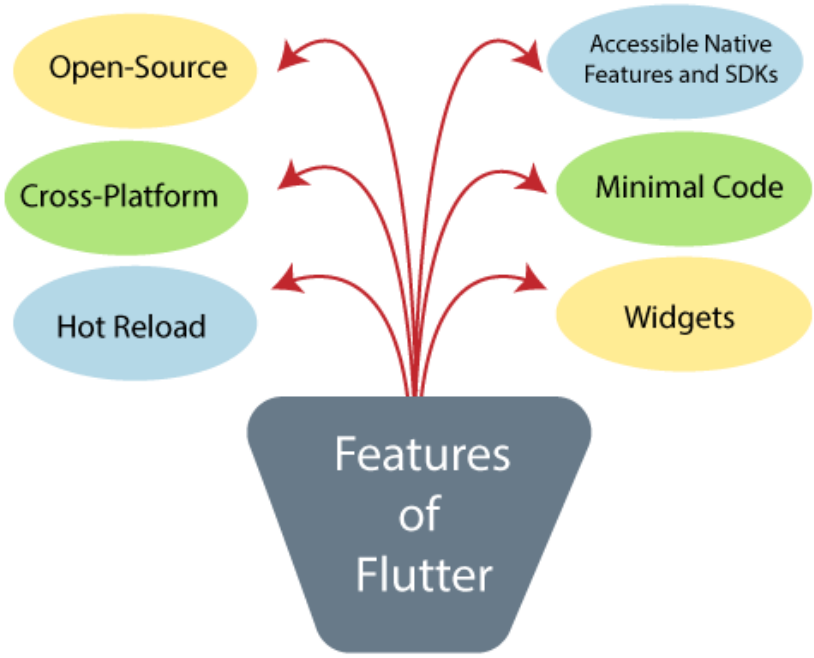
dụng đầy đủ tính năng, bao gồm máy ảnh, bộ nhớ, vị trí địa lý, mạng, SDK của bên thứ ba, v.v.

1.2 Điều làm Flutter trở lên độc đáo

Flutter khác với các framework khác vì nó không sử dụng **WebView** cũng như các widget OEM (Original Equipment Manufacturer) đi kèm với thiết bị. Thay vào đó, nó sử dụng công cụ kết xuất hiệu suất cao của riêng mình để vẽ các widget. Nó cũng triển khai hầu hết các hệ thống của nó như hoạt ảnh, cử chỉ và widget bằng ngôn ngữ lập trình Dart cho phép các nhà phát triển đọc, thay đổi, thay thế hoặc loại bỏ mọi thứ một cách dễ dàng. Nó cung cấp khả năng kiểm soát tuyệt vời cho các nhà phát triển đối với hệ thống.

1.3 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.



Hình 1.1: Các tính năng của Flutter

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 của Flutter

1.4.1 Kiến trúc Flutter

Trong phần này, chúng ta sẽ thảo luận về kiến trúc của Flutter framework. Kiến trúc Flutter chủ yếu bao gồm bốn thành phần.

- Động cơ Flutter (Flutter Engine)
- Thư viện nền tảng (Foundation Library)
- Vật dụng (Widgets)
- Thiết kế các widget cụ thể (Design Specific Widgets)

1.4.2. Flutter Engine

Nó là một cổng để giúp chạy các ứng dụng di động chất lượng cao và cơ bản dựa trên ngôn ngữ C ++. Nó triển khai các thư viện lõi Flutter bao gồm animation và đồ họa, tệp và mạng I / O, kiến trúc plugin, hỗ trợ trợ năng và thời gian chạy dart để phát triển, biên dịch và chạy các ứng dụng Flutter. Phải sử dụng thư viện đồ họa mã nguồn mở của Google, Skia, để hiển thị đồ họa cấp thấp.

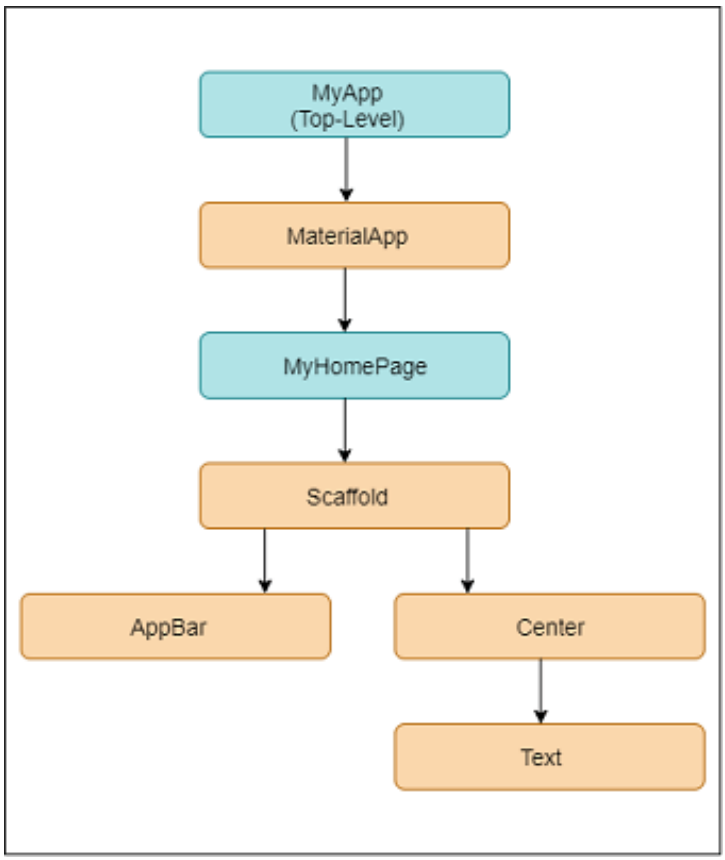
1.4.3. Thư viện nền tảng (Foundation Library)

Nó chứa tất cả các gói cần thiết cho các khối build cơ bản để viết một ứng dụng Flutter. Các thư viện này được viết bằng ngôn ngữ Dart.

1.4.4. Vật dụng (widget)

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.



Hình 1.2: Kiến trúc Flutter

Trong ví dụ trên, chúng ta có thể thấy rằng tất cả các thành phần đều là các widget có chứa các widget con. Do đó, ứng dụng Flutter tự nó là một widget.

1.4.5. Thiết kế các widget cụ thể

Framework Flutter có hai bộ widget phù hợp với các ngôn ngữ thiết kế cụ thể. Đây là Material Design cho ứng dụng Android và Cupertino Style cho ứng dụng IOS.

1.5. Giới thiệu về ngôn ngữ lập trình Dart

Dart là một ngôn ngữ lập trình hướng đối tượng mã nguồn mở, có mục đích chung với cú pháp kiểu C do Google phát triển vào năm 2011 . Mục đích của lập trình Dart là tạo giao diện người dùng frontend cho web và ứng dụng dành cho thiết bị di động. Nó đang được phát triển tích cực, được biên dịch sang mã máy gốc để xây dựng ứng dụng di động, lấy cảm hứng từ các ngôn ngữ lập trình khác như Java, JavaScript, C # và Typed mạnh. Vì Dart là một ngôn ngữ biên dịch nên bạn không thể thực thi code của mình trực tiếp; thay vào đó, trình biên dịch phân tích cú pháp nó và chuyển nó thành code máy.

Nó hỗ trợ hầu hết các khái niệm chung của ngôn ngữ lập trình như lớp, giao diện, hàm, không giống như các ngôn ngữ lập trình khác. Ngôn ngữ Dart không hỗ trợ mảng trực tiếp. Nó hỗ trợ tập hợp, được sử dụng để sao chép cấu trúc dữ liệu như mảng, generic và kiểu tùy chọn.

1.5.1 Kiểu dữ liệu

Dart là một ngôn ngữ lập trình Strongly Typed. Nó có nghĩa là, mỗi giá trị bạn sử dụng trong ngôn ngữ lập trình của mình có một kiểu là chuỗi hoặc số và phải được biết chính xác là kiểu dữ liệu gì trước khi code được biên dịch. Ở đây, chúng ta sẽ thảo luận về các kiểu dữ liệu cơ bản phổ biến nhất được sử dụng trong ngôn ngữ lập trình Dart.

Loại dữ liệu	Thí dụ	Mô tả
Chuỗi	String myName = cafedev.vn;	Nó chứa văn bản. Trong phần này, bạn có thể sử dụng dấu ngoặc kép đơn hoặc kép. Khi bạn quyết định dấu ngoặc kép, bạn phải nhất quán với lựa chọn của mình.
num, int, double	int age = 25;double = 125,50;	Kiểu dữ liệu num là viết tắt của một số. Dart có hai loại số:Số nguyên (Là một số không có chữ số thập phân.)Double (Nó là một số có chữ số thập phân.)
Boolean	bool var_name = true;Hoặcbool var_name = false;	Nó sử dụng từ khóa bool để biểu thị giá trị Boolean true và false.
vật	Person = Persion()	Nói chung, mọi thứ trong Dart là một đối tượng (ví dụ: Số nguyên, Chuỗi). Nhưng một đối tượng cũng có thể phức tạp hơn.

Hình 1.3: Bảng kiểu dữ liệu trong ngôn ngữ Dart

1.5.2 Các biến và các hàm

Các biến là không gian tên trong bộ nhớ lưu trữ các giá trị. Tên của một biến được gọi là định danh(identifiers). Chúng là nơi chứa dữ liệu, có thể lưu trữ giá trị của bất kỳ kiểu nào.

Ví dụ: Khi khai báo một biến var myAge = 50 ;

Ở đây, myAge là một biến lưu trữ giá trị số nguyên 50. Chúng ta cũng có thể cho nó là int và double. Tuy nhiên, Dart có một tính năng Type Inference, suy luận các loại giá trị. Vì vậy, nếu bạn tạo một biến với từ khóa var , Dart có thể suy ra biến đó thuộc loại số nguyên.

Bên cạnh biến, Hàm là một tính năng cốt lõi khác của bất kỳ ngôn ngữ lập trình nào. Hàm là một tập hợp các câu lệnh thực hiện một nhiệm vụ cụ thể. Chúng được tổ chức thành các khối mã logic có thể đọc được, có thể bảo trì và có thể tái sử dụng. Khai báo hàm chứa tên hàm, kiểu trả về và các tham số. Ví dụ sau giải thích hàm được sử dụng trong lập trình Dart.

1.5.3. Toán tử (Operators)

Ngôn ngữ Dart hỗ trợ tất cả các toán tử, toán tử là để xác định cách thức làm việc giữa các toán hạng. Toán hạng có thể là một hằng số, biến số hoặc một lời gọi hàm.. Tên của Operators được liệt kê dưới đây:

- Số học
- Tăng và giảm
- Logic
- So sánh

1.5.3.1 Toán tử số học

Toán tử số học là những phép toán cộng, trừ, nhân, chia cơ bản như trong toán học.

Giả sử biến A giữ giá trị 10, biến B giữ 20 thì:

Toán tử	Miêu tả	Ví dụ
+	Cộng hai toán hạng	A + B kết quả là 30
-	Trừ toán hạng thứ hai từ toán hạng đầu	A - B kết quả là -10
*	Nhân hai toán hạng	A * B kết quả là 200
/	Phép chia	B / A kết quả là 2
%	Phép lấy số dư	B % A kết quả là 0
++	Toán tử tăng (++), tăng giá trị toán hạng thêm một đơn vị	A++ kết quả là 11
--	Toán tử giảm (--), giảm giá trị toán hạng đi một đơn vị	A-- kết quả là 9

1.5.3.2 Toán tử tăng và giảm

Một toán tử tăng hay toán tử giảm được sử dụng như là một phần của biểu thức, thì sẽ có một sự khác nhau quan trọng giữa dạng tiền tố và hậu tố. Nếu bạn sử dụng dạng tiền tố thì toán tử tăng hoặc toán tử giảm được thực hiện trước biểu thức, và nếu bạn sử dụng dạng hậu tố thì toán tử tăng hoặc toán tử giảm được thực hiện sau khi biểu thức được ước lượng.

1.5.3.3 Toán tử logic

Toán tử Logic được sử dụng để kiểm tra tính đúng đắn của một hoặc nhiều biểu thức. Giá trị trả về của các biểu thức này là một giá trị kiểu boolean, true hoặc false.

Toán tử	Miêu tả	Ví dụ
&&	Được gọi là toán tử logic AND (và). Nếu cả hai toán tử đều có giá trị khác 0 thì điều kiện trở lên true.	(A && B) là false.
||	Được gọi là toán tử logic OR (hoặc). Nếu một trong hai toán tử khác 0, thì điều kiện là true.	(A || B) là true.
!	Được gọi là toán tử NOT (phủ định). Sử dụng để đảo ngược lại trạng thái logic của toán hạng đó. Nếu điều kiện toán hạng là true thì phủ định nó sẽ là false.	!(A && B) là true.

1.5.3.4 Toán tử so sánh

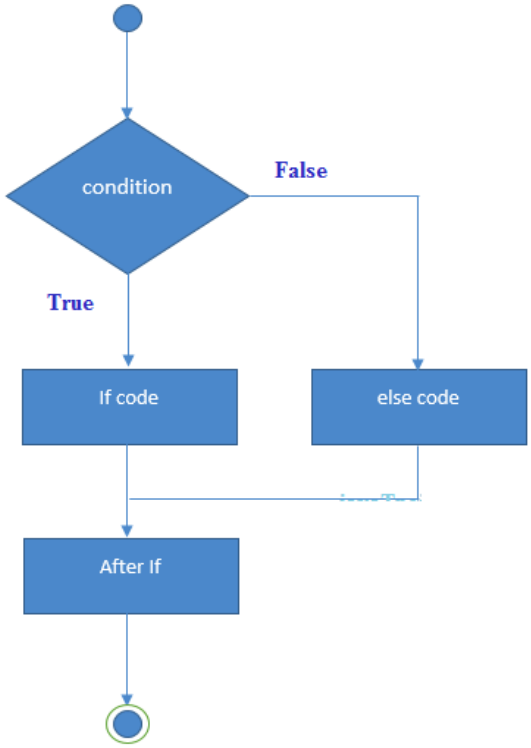
Toán tử so sánh dùng để so sánh hai toán hạng (2 biến giá trị) với nhau. Kết quả trả lại của toán tử so sánh nếu đúng thì sẽ là 1 (True), còn nếu sai thì kết quả sẽ là 0 (False).

1.5.4. Ra quyết định và các loại vòng lặp

1.5.4.1 Các câu lệnh ra quyết định

Ngôn ngữ Dart hỗ trợ các loại câu lệnh ra quyết định là câu lệnh if, câu lệnh if-else. Đoạn code minh họa các câu lệnh ra quyết định if, if-else.

```
void main() {
   var num = 12;
   if (num % 2 = = 0) {
      print("Number is Even.");
   } else {
```



Hình 1.4: Các câu lệnh quyết định if, if-else

1.5.4.2 Các vòng lặp

Các vòng lặp được sử dụng để thực thi một khối code lặp đi lặp lại cho đến khi một điều kiện được chỉ định trở thành đúng. Ngôn ngữ Dart hỗ trợ các loại câu lệnh lặp for, for..in, while, do..while.

```
void main() {
   var name = ["Peter", "Rinky Ponting", "Abhishek"];

   for (var prop in name) {
      print(prop);
   }
}
```

1.5.5 Bình luận (Comments)

Việc sử dụng bình luận là để cung cấp thông tin về dự án, biến hoặc một hoạt động, có ba bình luận được dùng ở trong lập trình Dart:

- **Thực hiện các bình luận định dạng:** Đó là một nhận xét dòng đơn (//)
- **Khối bình luận**: Đó là một nhận xét nhiều dòng (/*…*/)
- **Bình luận Tài liệu:** Đây là nhận xét tài liệu được sử dụng để mô tả cho thành viên và các kiểu (///)

1.5.6 Từ khóa Continue và Break

Dart cũng đã sử dụng từ khóa Continue và Break trong vòng lặp, và ở những nơi khác nó được yêu cầu. Câu lệnh continue cho phép bạn bỏ qua đoạn code còn lại bên trong vòng lặp và ngay lập tức chuyển đến bước lặp tiếp theo của vòng lặp.

```
void main() {
  for(int i=1;i<=10;i++){
    if(i==5){
      print("Hello");
      continue; //it will skip the rest statement
    }
    print(i);
  }
}
```

Câu lệnh break cho phép bạn kết thúc hoặc dừng dòng hiện tại của một chương trình và tiếp tục thực hiện sau phần thân của vòng lặp.

```
void main() {
  for(int i=1;i<=10;i++){
    if(i==5){
      print("Hello");
      break;//it will terminate the rest statement
    }
    print(i);
  }
}
```

1.5.7 Từ khóa Final và Const

Chúng ta có thể sử dụng một từ khóa Final để hạn chế người dùng. Nó có thể được áp dụng trong nhiều ngữ cảnh, chẳng hạn như biến, lớp và phương thức.

Từ khóa Const dùng để khai báo hằng. Chúng ta không thể thay đổi giá trị của từ khóa const sau khi gán nó.

```
void main() {
  final a = 100;
  const pi = 3.14;
  print(a);
  print(pi);
}
```

1.5.8. Lập trình hướng đối tượng

Dart là một ngôn ngữ lập trình hướng đối tượng, có nghĩa là mọi giá trị trong Dart đều là một đối tượng. Một số cũng là một đối tượng trong ngôn ngữ Dart. Lập trình Dart hỗ trợ khái niệm về các tính năng OOP như đối tượng, lớp, giao diện, v.v.

1.5.8.1 Đối tượng là gì trong lập trình

- Là các đối tượng trong thế giới thực mà có thể mô tả thông qua thuộc tính và hành vi riêng của nó.
- Ví dụ:
    - Con người (thuộc tính: tên , tuổi , giới tính.., hành vi: chạy, đá, đấm, làm việc..)
    - Căn nhà (thuộc tính: vị trí, diện tích, tên chủ nhà.. , hành vi: tránh nắng, giữ ấm…

Trong lập trình, khi bạn muốn xây dựng một đối tượng, trước tiên bạn cần phải xác định được: thuộc tính và các hành vi của nó.

1.5.8.2 Lớp (Class) trong ngôn ngữ Dart

Class là dùng để chứa các biến, và các hàm. Trong biểu diễn một đối tượng, class dùng để môt tả đối tượng, bao gồm các biến (các thuộc tính của đối tượng) các trường dữ liệu và các hàm (các phương thức của đối tượng).

Điểm khác biệt của ngôn ngữ Dart

- Không giống như các ngôn ngữ khác như Java hay C#, Dart cho phép khai báo nhiều đối tượng trong một file Dart.
- Không dùng các từ khoá như : public, private, protected.

Private được biểu diễn bằng dấu "_" trước các biến hay hàm. Ví dụ: String _name; nó được hiểu như : private String name;

Trong class có các thành phần :

Thuộc tính: Thuộc tính là những thông tin riêng của mỗi đối tượng, ta có thể thấy nó như là những biến liên quan đến đối tượng đó.

```
class Person {
 String name; // public
 int _tuoi;  // private
 Person (this.name, this._tuoi); // hàm khởi tạo
 int get tuoi => _tuoi; // hàm getter
 _doiTen (String name)  // private , trả về kiểu void
```

```
 {
   this.name = name;
 }
 int layTuoi()   // public
 {
   return this._tuoi;
```

Cách dùng "=>" trong ngôn ngữ Dart, chúng ta thấy có hàm getter:

```
int get tuoi => _tuoi;
```

Cách viết trên tương tự với:

```
int get tuoi {
   return _tuoi;
}
```

Phương thức khởi tạo: là những phương thức đặc biệt được gọi đến ngay khi khởi tạo 1 đối tượng nào đó. Đặc điểm của phương thức khởi tạo là : Có tên trùng với tên lớp, không có kiểu trả về, được tự động gọi khi 1 đối tượng thuộc lớp được khởi tạo, và có thể có nhiều Constructor. Cũng giống như các ngôn ngữ khác, Dart cũng có hàm khởi tạo. Nếu developer không tạo hàm khởi tạo riêng thì hàm khởi tạo mặc định không biến đầu vào sẽ được sử dụng.

Cách thức để viết một hàm khởi tạo trong Dart.

```
Person (this.name, this._tuoi);
```

Thay cho cách viết hàm khởi tạo của các ngôn ngữ khác:

```
Person (String name, int tuoi){
   this.name = name;
   this._tuoi = tuoi;
 }
```

Ví dụ định nghĩa một lớp trong Dart

```
class Mobile {
  // Property Declaration
  String color, brandName, modelName;
  // Method Creation
  String calling() {
```

```
    return "Mobile can do call to everyone.";
  }
  String musicPlay() {
    return "Mobile can play all types of Music.";
  }
  String clickPicture() {
    return "Mobile can take pictures.";   }  }
void main() {
  // Object Creation
  var myMob = new Mobile();
  // Accessing Class's Property
  myMob.color = "Black";
  myMob.brandName = "Apple Inc.";
  myMob.modelName = "iPhone 11 Pro";
  //Display Output
  print(myMob.color);
  print(myMob.modelName);
  print(myMob.brandName);
  print(myMob.calling());
  print(myMob.musicPlay());
  print(myMob.clickPicture());
}
```

Trong ví dụ trên, chúng ta định nghĩa một lớp Mobile , có ba biến kiểu chuỗi và ba hàm hoặc phương thức. Sau đó, chúng ta tạo một hàm main mà Dart sẽ thực thi đầu tiên khi ứng dụng của bạn khởi động. Bên trong main, chúng ta tạo một đối tượng để truy cập các thuộc tính của lớp. Cuối cùng, chúng ta in đầu ra.

1.5.8.3. Các đặc tính trong lập trình hướng đối tượng của ngôn ngữ Dart.

- **Tính trừu tượng (abstraction)**

Tính trừu tượng thể hiện ở việc lựa chọn các thuộc tính và hành vi của đối tượng mà không phải liệt kê hết tất cả các thuộc tính và hành vi của đối tượng đó.

Ví dụ: Để mô tả một người có rất nhiều thuộc tính và hành vi. Nhưng chúng ta chỉ sử dụng các thuộc tính như: tên , năm sinh, quê quán và thuộc tính như: đi, chạy mà không cần liệt kê hết tất cả các thuộc tính và hành vi khác như : tình trạng hôn nhân , lái xe, đá , đấm…

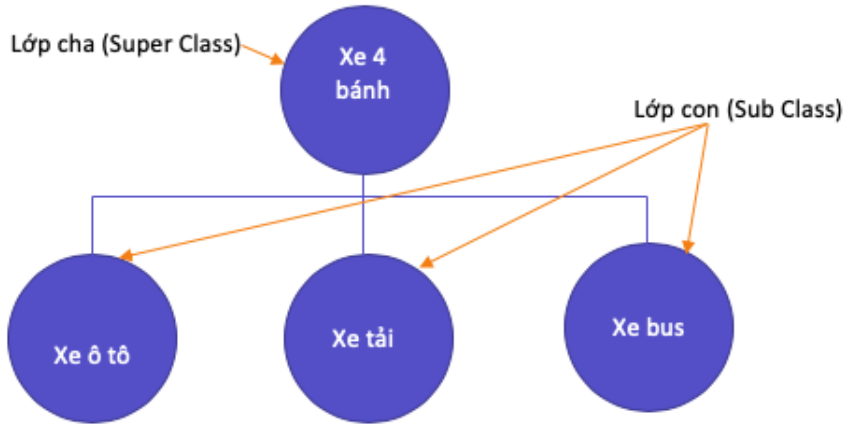
- **Tính đóng gói (Encapsulation)**

Tính đóng gói thể hiện sự che dấu trong đối tượng với mục đích bảo vệ dữ liệu và tăng khả năng mở rộng. Vì vậy khi triển khai một đối tượng, các thuộc tính nên dùng tính năng private. Ví dụ : int _tuoi; và sử dụng các phương thức để truy xuất các trường dữ liệu như getter , setter:

```
int _tuoi;
// getter
int get tuoi => _tuoi;
// setter
void set datTuoi(int tuoi)
  {
    this._tuoi = tuoi;
  }
```

- **Tính kế thừa**

Trong một phần mềm hay chương trình, được cấu tạo bởi nhiều lớp khác nhau cùng các thành phần khác. Mối quan hệ giữa các lớp, có mối quan hệ kế thừa, gồm lớp cha (super class)và các lớp con(sub class), Các lớp con đó lại có thể là lớp cha của các lớp khác.



Mô tả tính kế thừa giữa các nút

Mục đích của kế thừa là tái sử dụng. Lớp con có thể sở hữu các thuộc tính và phương thức public của lớp cha nhưng không được sở hữu các thuộc tính hay phương thức private và các hàm constructor. Biểu diễn kế thừa trong Dart cũng tương tự như trong Java : dùng extends. Cách kế thừa như sau:

```
class Car extends XeBonBanh{...}
class Truck extends XeBonBanh{...}
```

Trong ngôn ngữ Dart, không dùng từ khoá interface hay không dùng phương thức có từ khoá abstract ở phía trước

```
interface Hinh{} // Error
 abstract double tinhDienTich(); // Error
```

- **Tính đa hình**

Tính đa hình trong ngôn ngữ Dart cũng có ý nghĩa giống như trong các ngôn ngữ khác. Cùng biểu diễn một hành vi nhưng từng lớp có cách biểu diễn khác nhau. Để hiểu rõ hơn về tính đa hình của lập trình hướng đối tượng thì hãy xem một đoạn code minh họa về tính năng của xe

```
// Khai báo về tính năng của Xe
abstract class Xe {
  void chuyenCho(); // Chuyên chở}
// Khai báo của Xe tải
import 'package:dart_language/XeCo.dart';
class XeTai extends Xe{  @override
  void chuyenCho() {
    print (" Chỉ chở hàng");  }}
// Khai báo về Xe khách
import 'package:dart_language/XeCo.dart';
class XeKhach extends Xe{  @override
  void chuyenCho() {
    print (" Chỉ chở người");  }}
```

Như vậy, Cùng một hành vi là Chở, nhưng với các loại xe là các công dụng lại khác nhau – Đó là tính đa hình.

1.6. Một số loại Widgets của Flutter thường gặp

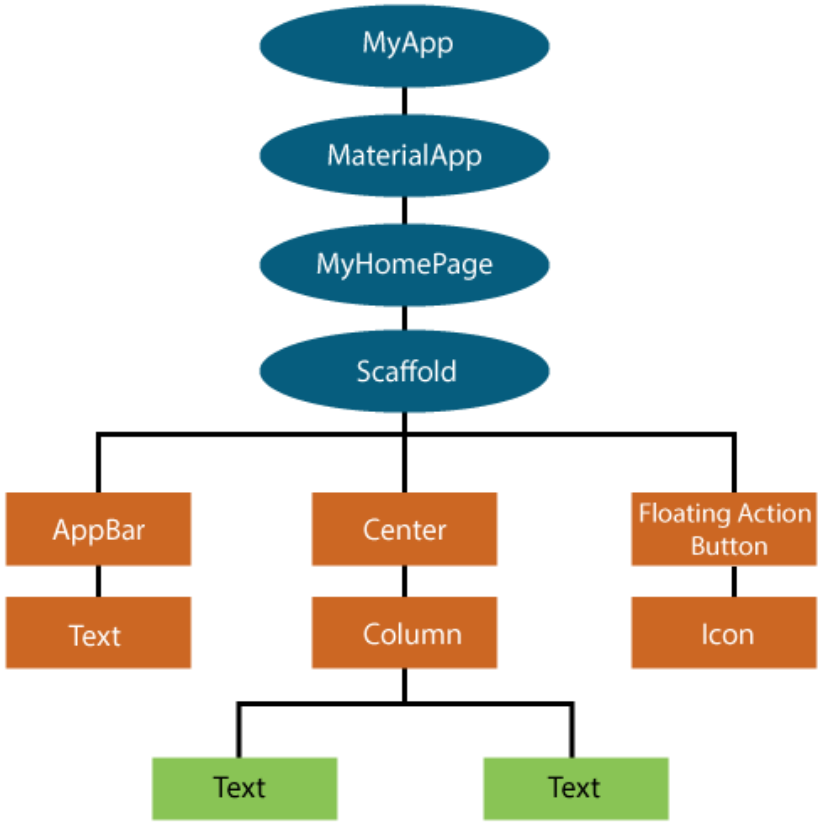
1.6.1 Widget Flutter

Trong phần này, chúng ta sẽ tìm hiểu khái niệm về một widget, cách tạo nó và các loại khác nhau của chúng có sẵn trong Flutter framework. Chúng ta đã biết trước đó rằng mọi thứ trong Flutter đều là một widget.

Bất cứ khi nào bạn định viết mã để xây dựng bất cứ thứ gì trong Flutter, nó sẽ nằm trong một widget. Mục đích chính là xây dựng ứng dụng từ các widget. Nó mô tả chế độ xem ứng dụng của bạn trông như thế nào với cấu hình và trạng thái hiện tại của chúng.

Khi bạn thực hiện bất kỳ thay đổi nào trong code, widget con sẽ xây dựng lại mô tả của nó bằng cách tính toán sự khác biệt của widget con hiện tại và trước đó để xác định những thay đổi tối thiểu đối với việc hiển thị trong giao diện người dùng của ứng dụng.

Các widget được lồng vào nhau để xây dựng ứng dụng. Nó có nghĩa là thư mục gốc của ứng dụng của bạn tự nó là một widget, và tất cả các cách nhìn xuống cũng là một widget. Ví dụ: một widget có thể hiển thị một thứ gì đó, có thể xác định thiết kế, có thể xử lý tương tác, v.v.



Hình 1 5: Cây Widget

1.6.2 widget hiển thị

Các widget hiển thị có liên quan đến dữ liệu đầu vào và đầu ra của người dùng. Một số loại quan trọng của widget con này là:

Text

Text (Văn bản) 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.

Đây là một ví dụ đơn giản để hiểu widget này. Ví dụ này hiển thị tiêu đề dự án của chúng ta trong thanh ứng dụng và một thông báo trong nội dung ứng dụng.

```
void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData(
```

```
          primarySwatch: Colors.green,   ),
        home: MyTextPage());}}
class MyTextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title:Text("Text Widget Example"),
      body: Center(
          child:Text("Welcome to Cafedev")
 ),);}}
```

Trong đoạn code trên, chúng ta đã sử dụng widget MaterialApp gọi màn hình chính bằng cách sử dụng lớp MyTextPage() . Lớp này chứa các scaffold, trong đó có AppBar và Body, nơi chúng ta đã sử dụng widget Text để hiển thị tiêu đề và cơ thể, tương ứng.

Trình tạo ra một widget con.

Công cụ tạo widget Text được sử dụng để tạo giao diện tùy chỉnh cho Text của chúng ta trong Flutter:

TextAlign: Nó được sử dụng để chỉ định cách Text của chúng ta được căn chỉnh theo chiều ngang. Nó cũng kiểm soát vị trí Text.

TextDirection: Nó được sử dụng để xác định cách các giá trị textAlign kiểm soát bố cục của Text của chúng ta. Thông thường, chúng ta viết Text từ trái sang phải, nhưng chúng ta có thể thay đổi nó bằng cách sử dụng tham số này.

Overflow: Nó được sử dụng để xác định khi nào Text sẽ không vừa với không gian có sẵn. Nó có nghĩa là chúng ta đã chỉ định nhiều Text hơn không gian có sẵn.

TextScaleFactor: Nó được sử dụng để xác định tỷ lệ của Text được hiển thị bởi widget Text. Giả sử chúng ta đã chỉ định hệ số tỷ lệ Text là 1,5, thì Text của chúng ta sẽ lớn hơn 50 phần trăm so với kích thước phông chữ được chỉ định.

SoftWrap: Nó được sử dụng để xác định có hay không hiển thị tất cả nội dung widget Text khi không còn đủ dung lượng. Nếu nó là sự thật, nó sẽ hiển thị tất cả nội dung. Nếu không, nó sẽ không hiển thị tất cả nội dung.

MaxLines: Nó được sử dụng để xác định số dòng tối đa được hiển thị trong widget Text.

TextWidthBasis: Nó được sử dụng để kiểm soát cách xác định chiều rộng Text.

TextHeightBehavior: Nó được sử dụng để kiểm soát cách đoạn văn xuất hiện giữa dòng đầu tiên và phần cuối của dòng cuối cùng.

Style: Đây là thuộc tính phổ biến nhất của widget con này cho phép các nhà phát triển tạo kiểu dáng cho Text của họ. Nó có thể tạo kiểu bằng cách chỉ định màu nền và nền trước, cỡ chữ, độ đậm của phông chữ, khoảng cách giữa các chữ và từ, ngôn ngữ, bóng, v.v.

Bảng mô tả các thành phần của widget text:

Thuộc tính	**Mô tả**
foreground	Nó xác định màu nền trước cho Text.
background	Nó xác định sơn làm nền cho Text.
fontWeight	Nó quyết định độ dày của Text.
fontSize	Nó xác định kích thước của Text.
fontFamily	Nó được sử dụng để chỉ định kiểu chữ cho phông chữ. Đối với điều này, chúng ta cần tải xuống file kiểu chữ trong dự án của mình, sau đó giữ tệp này vào thư mục assets/font. Cuối cùng, cấu hình file pubspec.yaml để sử dụng nó trong dự án.
fontStyle	Nó được sử dụng để tạo kiểu cho phông chữ ở dạng in đậm hoặc nghiêng.
Color	Nó được sử dụng để xác định màu sắc của Text.
letterSpacing	Nó được sử dụng để xác định khoảng cách giữa các ký tự của Text.
wordSpacing	Nó được sử dụng để xác định khoảng cách giữa hai từ của Text.
shadows	Nó được sử dụng để vẽ bên dưới Text.

decoration	Chúng ta sử dụng điều này để trang trí Text bằng cách sử dụng ba tham số: decoration, decorationColor, decorationStyle. The decoration determines the location, decorationColor specify the color, decorationStyle determine xác định hình dạng.

Button

Nút (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.

Các nút là các widget Flutter, là một phần của thư viện material design. Flutter cung cấp một số loại nút có hình dạng, kiểu dáng và tính năng khác nhau.

Tính năng của các nút

Các tính năng chuẩn của một nút trong Flutter được đưa ra dưới đây:

- Chúng ta có thể dễ dàng áp dụng các chủ đề trên các nút, hình dạng, màu sắc, hoạt ảnh và hành vi.
- Chúng tôi cũng có thể biểu tượng chủ đề và văn bản bên trong nút.
- Các nút có thể được cấu tạo từ các widget con khác nhau cho các đặc điểm khác nhau.

Các loại nút trong Flutter

- Nút phẳng (Flat Button)
- Nút nâng (Raised Button)
- Nút nổi (Floating Button)
- Nút thả xuống (Drop Down Button)
- Nút biểu tượng (Icon Button)

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:

- **Hình ảnh (Image):** Đây là một trình tải hình ảnh chung, được sử dụng bởi **ImageProvider** .
- **Tài sản (asset):** Nó tải hình ảnh từ thư mục tài sản dự án của bạn.
- **Tệp (file):** Nó tải hình ảnh từ thư mục hệ thống.
- **Bộ nhớ (memory):** Nó tải hình ảnh từ bộ nhớ.

- **Mạng (network):** Nó tải hình ảnh từ mạng.

Để thêm hình ảnh vào dự án, trước tiên bạn cần tạo một thư mục nội dung nơi bạn lưu giữ hình ảnh của mình và sau đó thêm dòng bên dưới vào tệp **pubspec.yaml** .

```
assets:

  - assets/
```

Bây giờ, thêm dòng sau vào tệp dart.

```
mage.asset('assets/computer.png')
```

1.6.3 widget ẩn

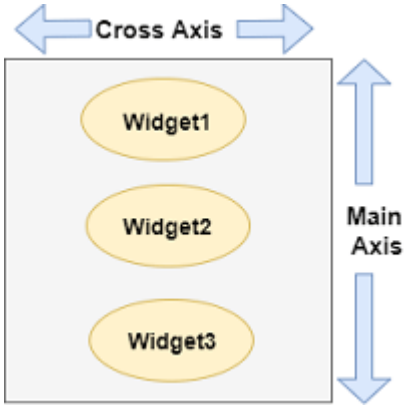
Các widget vô hình có liên quan đến cách bố trí và kiểm soát các widget. Nó cung cấp việc kiểm soát cách các widget thực sự hoạt động và cách chúng sẽ hiển thị trên màn hình. Một số loại widget quan trọng là:

Column

Widget này sắp xếp các con của nó theo hướng dọc trên màn hình . Nói cách khác, nó sẽ mong đợi một mảng dọc gồm các widget con. Nếu widget cần lấp đầy không gian dọc có sẵn, chúng ta phải bọc các widget trong widget Mở rộng.

Một widget dạng cột không thể cuộn được vì nó hiển thị các widget trong chế độ xem hiển thị đầy đủ. Vì vậy, sẽ được coi là sai nếu chúng ta có nhiều con hơn trong một cột sẽ không phù hợp với không gian có sẵn. Nếu chúng ta muốn tạo một danh sách các widget cột có thể cuộn được, chúng ta cần sử dụng ListView Widget.

Chúng ta cũng có thể kiểm soát cách một widget cột sắp xếp các con của nó bằng cách sử dụng thuộc tính mainAxisAlignment và crossAxisAlignment. Trục chéo của cột sẽ chạy theo chiều ngang và trục chính sẽ chạy theo chiều dọc . Hình ảnh minh họa dưới đây giải thích rõ ràng hơn.



Hình 1. 6 Widget Cột

Lưu ý: widget cột cũng căn chỉnh nội dung của nó bằng cách sử dụng các thuộc tính tương tự như chúng ta đã thảo luận trong widget hàng như bắt đầu, kết thúc, giữa, khoảng trắng, khoảng trắng và khoảng trắng.**Center**

Widget con này được sử dụng để căn giữa widget con, nằm bên trong nó. Tất cả các ví dụ trước đều chứa bên trong widget Center.

Stack

Ngăn xếp (Stack) là một widget trong Flutter chứa danh sách các widget và đặt chúng trên đầu các widget khác. Nói cách khác, ngăn xếp cho phép các nhà phát triển chồng nhiều widget vào một màn hình duy nhất và hiển thị chúng từ dưới lên trên. Do đó, widget đầu tiên là mục dưới cùng và widget cuối cùng là mục trên cùng

Các điểm chính liên quan đến Stack Widget

Sau đây là những điểm chính của widget ngăn xếp Flutter:

- widget trong ngăn xếp có thể được định vị hoặc không được định vị.
- Các mục được định vị được bao bọc trong widget được Định vị và phải có một thuộc tính không rỗng
- Các widget con không định vị được tự căn chỉnh. Nó hiển thị trên màn hình dựa trên sự liên kết của ngăn xếp. Vị trí mặc định của các con là ở góc trên cùng bên trái.
- Chúng ta có thể sử dụng thuộc tính alignment để thay đổi sự liên kết của các widget.
- Stack đặt các widget con theo thứ tự với con đầu tiên ở dưới cùng và con cuối cùng ở trên cùng. Nếu chúng ta muốn sắp xếp lại widget dành cho trẻ em, thì bắt buộc phải xây dựng lại ngăn xếp theo thứ tự mới. Theo mặc định, widget đầu tiên của mỗi ngăn xếp có kích thước tối đa so với các widget khác.

Cách sử dụng một ngăn xếp (stack):

```
child: Stack(
            fit: StackFit.passthrough,
            overflow: Overflow.visible,
            children: <Widget>[
              // Max Size Widget
              Container(
                height: 300,
                width: 400,
                color: Colors.green,
                child: Center(
```

```
                  child: Text(
                    'Top Widget: Green',
                    style: TextStyle(color: Colors.white, fontSize:
20), ),   ), ),
```

1.7. Tìm hiểu về bố cụ(layout) giao diện trong Flutter

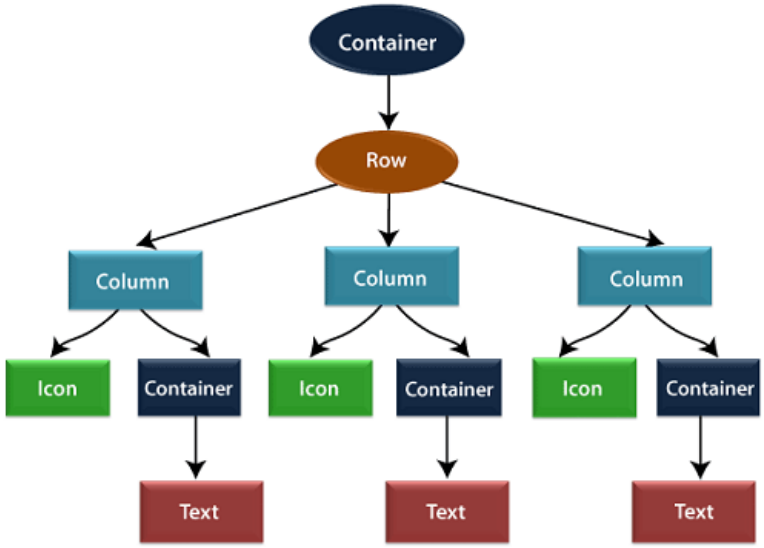
1.7.1 Bố cục

Khái niệm chính của cơ chế bố trí là widget. Chúng ta biết rằng sự flutter giả định mọi thứ như một widget Vì vậy, hình ảnh, biểu tượng, văn bản và thậm chí cả bố cục(layout) của ứng dụng của bạn đều là widget. Ở đây, một số thứ bạn không thấy trên giao diện người dùng ứng dụng của mình, chẳng hạn như các hàng, cột và lưới sắp xếp, ràng buộc và căn chỉnh các widget hiển thị cũng là các widget.

Flutter cho phép chúng ta tạo bố cục bằng cách soạn nhiều widget để xây dựng các widget phức tạp hơn. Ví dụ , chúng ta có thể thấy hình ảnh dưới đây hiển thị ba biểu tượng với nhãn bên dưới mỗi biểu tượng.

Hình 1.7: BIểu tượng và nhãn dán trong flutter

Trong hình ảnh thứ hai, chúng ta có thể thấy bố cục trực quan của hình ảnh trên. Hình ảnh này hiển thị một hàng gồm ba cột và các cột này chứa một biểu tượng và nhãn.



Hình 1.8: Nội dung của Container

Trong hình trên, vùng chứa là một lớp widget cho phép chúng ta tùy chỉnh widget con. Nó chủ yếu được sử dụng để thêm đường viền, đệm, lề, màu nền và

nhiều thứ khác. Tại đây, widget văn bản nằm dưới vùng chứa để thêm lề. Toàn bộ hàng cũng được đặt trong một vùng chứa để thêm lề và phần đệm xung quanh hàng. Ngoài ra, phần còn lại của giao diện người dùng được kiểm soát bởi các thuộc tính như màu sắc, kiểu văn bản, v.v.

1.7.2 Bố cục một widget

Hãy để chúng ta tìm hiểu cách chúng ta có thể tạo và hiển thị một widget đơn giản. Các bước sau đây cho biết cách bố trí widget:

Bước 1: Đầu tiên, bạn cần chọn một Bố cục widget.

Bước 2: Tiếp theo, tạo một widget hiển thị.

Bước 3: Sau đó, thêm widget hiển thị vào widget layout.

Bước 4: Cuối cùng, thêm widget bố cục vào trang mà bạn muốn hiển thị.

1.7.3. Các loại widget bố cục

Chúng tôi có thể phân loại widget bố cục thành hai loại:

- widget đơn
- widget đa

1.7.3.1 Các widget đơn

Widget bố cục con duy nhất là một loại widget, có thể chỉ có một widget bên trong widget bố cục mẹ. Các widget này cũng có thể chứa chức năng bố cục đặc biệt. Flutter cung cấp cho chúng ta nhiều widget con để làm cho giao diện người dùng của ứng dụng trở nên hấp dẫn. Nếu chúng ta sử dụng các widget này một cách thích hợp, nó có thể tiết kiệm thời gian của chúng ta và làm cho code ứng dụng dễ đọc hơn. Danh sách các loại widget đơn lẻ khác nhau là:

Container: Đây là widget bố cục phổ biến nhất cung cấp các tùy chọn có thể tùy chỉnh để đặt màu, định vị và định cỡ các widget.

```
Center(
  child: Container(
    margin: const EdgeInsets.all(15.0),
    color: Colors.blue,
    width: 42.0,
    height: 42.0,
  ),
)
```

Padding: Nó là một widget được sử dụng để sắp xếp widget con của nó theo khoảng đệm đã cho. Nó chứa EdgeInsets và EdgeInsets.fromLTRB cho phía mong muốn mà bạn muốn cung cấp đệm.

```
const Greetings(
  child: Padding(
    padding: EdgeInsets.all(14.0),
    child: Text('Hello Cafedev!'),
  ),
)
```

Center: widget này cho phép bạn căn giữa widget trong chính nó.

SizedBox: widget này cho phép bạn cung cấp kích thước được chỉ định cho widget thông qua tất cả các màn hình.

```
SizedBox(
  width: 300.0,
  height: 450.0,
  child: const Card(child: Text('Hello Cafedev!')),
)
```

AspectRatio: widget này cho phép bạn giữ kích thước của widget theo một tỷ lệ khung hình được chỉ định.

ConstrainedBox: Đây là một widget cho phép bạn buộc các ràng buộc bổ sung lên widget con của nó. Nó có nghĩa là bạn có thể buộc widget con có một ràng buộc cụ thể mà không làm thay đổi các thuộc tính của widget con.

1.7.3.2 Đa widget

Đa widget là một loại widget chứa nhiều hơn một widget con bên trong và cách bố trí của các widget này là duy nhất . Ví dụ: widget Hàng bố trí widget theo hướng ngang và widget Cột bố trí widget theo hướng dọc. Nếu chúng ta kết hợp widget Hàng và Cột, thì nó có thể xây dựng bất kỳ cấp độ nào của widget phức tạp.

```
class MyApp extends StatelessWidget {
  // It is the root widget of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Multiple Layout Widget',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        // This is the theme of your application.
        primarySwatch: Colors.blue,   ),
      home: MyHomePage(),
    );   }  }
```

```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        alignment: Alignment.center,
        color: Colors.white,
        child: Row(
          children: <Widget>[
            Expanded(
              child: Text('Peter', textAlign: TextAlign.center),     ),
            Expanded(
              child: Text('John', textAlign: TextAlign.center ),
),
            Expanded(
              child: FittedBox(
                fit: BoxFit.contain, // otherwise the logo will be tiny
                child: const FlutterLogo(),
              ),
          ], ),),); } }
```

1.8. Tìm hiểu về Cử chỉ(Gestures) với giao diện trong Flutter

Cử chỉ (Gestures) là một tính năng thú vị trong Flutter cho phép chúng ta tương tác với ứng dụng di động (hoặc bất kỳ thiết bị dựa trên cảm ứng). Nói chung, cử chỉ xác định bất kỳ hành động hoặc chuyển động vật lý nào của người dùng nhằm mục đích kiểm soát thiết bị di động. Một số ví dụ về cử chỉ là:

- Khi màn hình di động bị khóa, bạn trượt ngón tay trên màn hình để mở khóa.
- Nhấn vào một nút trên màn hình điện thoại di động của bạn và
- Nhấn và giữ biểu tượng ứng dụng trên thiết bị dựa trên cảm ứng để kéo biểu tượng đó qua các màn hình.

Chúng ta sử dụng tất cả những cử chỉ này trong cuộc sống hàng ngày để tương tác với điện thoại hoặc thiết bị dựa trên cảm ứng của bạn.

Flutter chia hệ thống cử chỉ thành hai lớp khác nhau, được đưa ra dưới đây:

- Con trỏ(Pointers)
- Cử chỉ(Gestures)

1.8.1 Con trỏ

Con trỏ(Pointers) là lớp đầu tiên đại diện cho dữ liệu thô về tương tác của người dùng. Nó có các sự kiện, mô tả vị trí và chuyển động của các con trỏ như chạm, chuột và kiểu trên màn hình. Flutter không cung cấp bất kỳ cơ chế nào để hủy hoặc dừng các sự kiện con trỏ được gửi đi thêm. Flutter cung cấp một widget Listener để lắng nghe các sự kiện con trỏ trực tiếp từ lớp widget. Con trỏ-sự kiện được phân loại thành bốn loại chủ yếu:

- PointerDownEvents
- PointerMoveEvents
- PointerUpEvents
- PointerCancelEvents

PointerDownEvents: Nó cho phép con trỏ tiếp xúc với màn hình tại một vị trí cụ thể.

PointerMoveEvents: Nó cho phép con trỏ di chuyển từ vị trí này đến vị trí khác trên màn hình.

PointerUpEvents: Nó cho phép con trỏ dừng tiếp xúc với màn hình.

PointerCancelEvents: Sự kiện này được gửi khi tương tác với con trỏ bị hủy.

1.8.2 Cử chỉ

Đây là lớp thứ hai đại diện cho các hành động ngữ nghĩa như chạm, kéo và chia tỷ lệ, được nhận dạng từ nhiều sự kiện con trỏ riêng lẻ. Nó cũng có thể gửi nhiều sự kiện tương ứng với vòng đời cử chỉ như kéo bắt đầu, kéo cập nhật và kéo kết thúc. Một số cử chỉ được sử dụng phổ biến được liệt kê dưới đây:

Chạm (Tap): Có nghĩa là chạm vào bề mặt màn hình từ đầu ngón tay trong một thời gian ngắn rồi thả chúng ra. Cử chỉ này chứa các sự kiện sau:

- onTapDown
- onTapUp
- onTap
- onTapCancel

Nhấn đúp (Double Tap): Nó tương tự như cử chỉ Nhấn, nhưng bạn cần nhấn hai lần trong thời gian ngắn. Cử chỉ này chứa các sự kiện sau:

- onDoubleTap

Kéo (Drag): Nó cho phép chúng ta chạm vào bề mặt của màn hình bằng đầu ngón tay và di chuyển nó từ vị trí này sang vị trí khác rồi thả chúng ra. Flutter phân loại kéo thành hai loại:

- Kéo ngang: Cử chỉ này cho phép con trỏ di chuyển theo hướng ngang. Nó chứa các sự kiện sau:
    - onHorizontalDragStart
    - onHorizontalDragUpdate
    - onHorizontalDragEnd
- Kéo dọc: Cử chỉ này cho phép con trỏ di chuyển theo hướng thẳng đứng. Nó chứa các sự kiện sau:
    - onVerticalDragStart
    - onVerticalDragStart
    - onVerticalDragStart

Nhấn lâu (Long Press): Có nghĩa là chạm vào bề mặt của màn hình tại một vị trí cụ thể trong một thời gian dài. Cử chỉ này chứa các sự kiện sau:

- onLongPress

Di chuyển (Pan): Có nghĩa là chạm vào bề mặt của màn hình bằng đầu ngón tay, có thể di chuyển theo bất kỳ hướng nào mà không cần nhả đầu ngón tay. Cử chỉ này chứa các sự kiện sau:

- onPanStart
- onPanUpdate
- onPanEnd

Chụm (Pinch): Có nghĩa là chụm (di chuyển ngón tay và ngón cái của một người hoặc đưa chúng lại gần nhau trên màn hình cảm ứng) bề mặt của màn hình bằng cách sử dụng hai ngón tay để phóng to hoặc thu nhỏ màn hình.

1.8.3 Dò cử chỉ

Flutter cung cấp một tiện ích hỗ trợ tuyệt vời cho tất cả các loại cử chỉ bằng cách sử dụng tiện ích GestureDetector. GestureWidget là các widget không trực quan, chủ yếu được sử dụng để phát hiện cử chỉ của người dùng. Ý tưởng cơ bản của bộ phát hiện cử chỉ là một tiện ích không trạng thái có chứa các tham số trong hàm tạo của nó cho các sự kiện chạm khác nhau.

Trong một số tình huống, có thể có nhiều bộ phát hiện cử chỉ tại một vị trí cụ thể trên màn hình và sau đó, khung sẽ xác định cử chỉ nào sẽ được gọi. Widget GestureDetector quyết định cử chỉ nào sẽ nhận ra dựa trên lệnh gọi lại nào của nó là không rỗng.

1.9. Quản lý State

Như đã tìm hiểu ở phần widget, mọi thứ trong ứng dụng flutter đều là một widget. Có thể phân loại widget này thành hai loại, một là widget Không trạng thái

(Stateless widget) và một là widget có Trạng thái(Stateful widget). widget không trạng thái không có bất kỳ trạng thái bên trong nào. Nó có nghĩa là một khi nó được xây dựng, chúng ta không thể thay đổi hoặc sửa đổi nó cho đến khi chúng được khởi tạo lại. Mặt khác, widget Stateful là động và có trạng thái. Nó có nghĩa là chúng ta có thể sửa đổi nó một cách dễ dàng trong suốt vòng đời của nó mà không cần khởi động lại nó một lần nữa.

1.9.1. State (Trạng thái)

Trạng thái là thông tin có thể đọc được khi widget được tạo và có thể thay đổi hoặc sửa đổi trong suốt thời gian tồn tại của ứng dụng. Nếu bạn muốn thay đổi widget của mình, bạn cần cập nhật đối tượng trạng thái, có thể được thực hiện bằng cách sử dụng hàm setState() có sẵn cho các widget Stateful. Hàm setState() cho phép chúng ta thiết lập các thuộc tính của đối tượng trạng thái kích hoạt vẽ lại giao diện người dùng.

Quản lý state (trạng thái) là một trong những quy trình phổ biến và cần thiết nhất trong vòng đời của một ứng dụng. Theo tài liệu chính thức, Flutter mang tính chất khai báo. Điều đó có nghĩa là Flutter xây dựng giao diện người dùng của mình bằng cách phản ánh trạng thái hiện tại của ứng dụng của bạn. Hình sau giải thích rõ hơn về nơi bạn có thể xây dựng giao diện người dùng từ trạng thái ứng dụng.

Trong Flutter, quản lý state (trạng thái) phân thành hai loại khái niệm, được đưa ra dưới đây:

- Trạng thái tức thời (Ephemeral State)
- Trạng thái ứng dụng (App State)

1.9.1.1 Trạng thái tức thời

Trạng thái này còn được gọi là Trạng thái giao diện người dùng hoặc trạng thái địa phương. Nó là một loại trạng thái có liên quan đến widget cụ thể, hoặc bạn có thể nói rằng nó là một trạng thái chứa trong một widget duy nhất. Trong loại trạng thái này, bạn không cần sử dụng các kỹ thuật quản lý state (trạng thái). Ví dụ phổ biến của trạng thái này là Text Field.

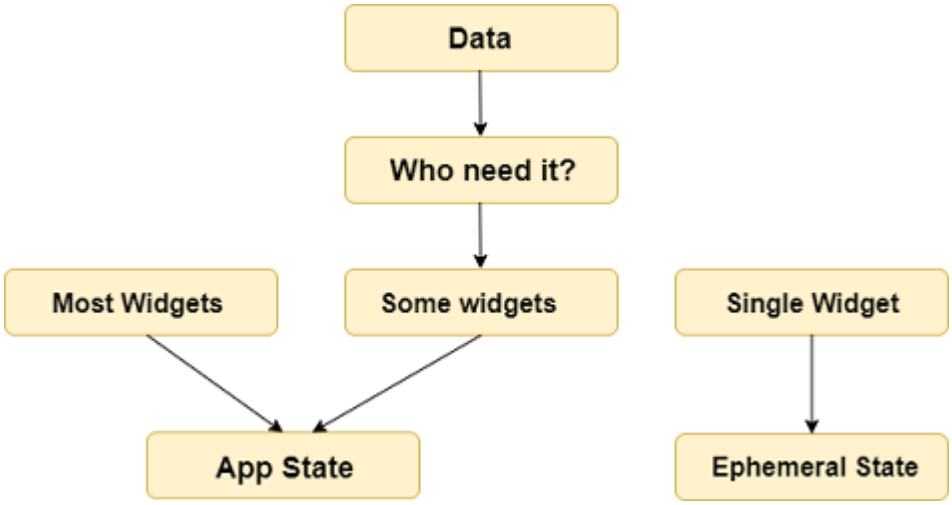
```
class MyHomepage extends StatefulWidget {
  @override
  MyHomepageState createState() => MyHomepageState();
}

class MyHomepageState extends State<MyHomepage> {
  String _name = "Peter";

  @override
```

```
Widget build(BuildContext context) {
  return RaisedButton(
      child: Text(_name),
      onPressed: () {
         setState(() {
            _name = _name == "Peter" ? "John" : "Peter";
         });
       },
    );
}
```

Trong ví dụ trên, _name là một trạng thái tạm thời. Ở đây, chỉ có hàm setState () bên trong lớp của StatefulWidget mới có thể truy cập vào _name. Phương thức xây dựng gọi một hàm setState (), hàm này thực hiện sửa đổi các biến trạng thái. Khi phương thức này được thực thi, đối tượng widget sẽ được thay thế bằng đối tượng mới, mang lại giá trị biến được sửa đổi.

1.9.1.2 Trạng thái ứng dụng

Nó khác với trạng thái tức thời. Đó là một loại trạng thái mà chúng ta muốn chia sẻ trên các phần khác nhau của ứng dụng và muốn giữ lại giữa các phiên của người dùng. Do đó, loại trạng thái này có thể được sử dụng trên toàn cầu. Đôi khi nó còn được gọi là trạng thái ứng dụng hoặc trạng thái chia sẻ. Một số ví dụ về trạng thái này là Tùy chọn người dùng, Thông tin đăng nhập, thông báo trong ứng dụng mạng xã hội, giỏ hàng trong ứng dụng thương mại điện tử, trạng thái đã đọc / chưa đọc của các bài báo trong ứng dụng tin tức, v.v.



Hình 1. 9 Sự khác biệt giữa trạng thái ứng dụng và trạng thái tức thời

Ví dụ đơn giản nhất về quản lý trạng thái ứng dụng có thể được học bằng cách sử dụng gói nhà cung cấp. Việc quản lý state (trạng thái) với nhà cung cấp rất dễ

hiểu và ít phải viết mã. Nhà cung cấp là thư viện của bên thứ ba. Ở đây, chúng ta cần hiểu ba khái niệm chính để sử dụng thư viện này.

1. ChangeNotifier: là một lớp đơn giản, cung cấp thông báo thay đổi cho người nghe của nó. Nó dễ hiểu, dễ thực hiện và được tối ưu hóa cho một số lượng nhỏ người nghe. Nó được sử dụng để người nghe quan sát một mô hình để thay đổi. Trong điều này, chúng ta chỉ sử dụng phương thức tifyListener () để thông báo cho người nghe.

2. ChangeNotifierProvider: là widget cung cấp một phiên bản của Change Notifier cho con của nó. Nó đến từ gói nhà cung cấp.

3. Khách hàng (Consumer): Nó là một loại nhà cung cấp không làm bất kỳ công việc cầu kỳ. Nó chỉ gọi nhà cung cấp trong một widget con mới và ủy quyền triển khai bản dựng của nó cho người xây dựng.

1.10 Tìm hiểu về Navigator và Routing trong Flutter

Điều hướng và định tuyến (Navigation and Routing) là một số khái niệm cốt lõi của tất cả các ứng dụng di động, cho phép người dùng di chuyển giữa các trang khác nhau. Chúng ta biết rằng mọi ứng dụng di động đều chứa một số màn hình để hiển thị các loại thông tin khác nhau. Ví dụ**:** một ứng dụng có thể có màn hình chứa nhiều sản phẩm khác nhau. Khi người dùng chạm vào sản phẩm đó, ngay lập tức nó sẽ hiển thị thông tin chi tiết về sản phẩm đó.

Trong Flutter, các màn hình và trang được gọi là các tuyến và các tuyến này chỉ là một widget. Trong Android, một tuyến tương tự như Activity, trong khi trong iOS, nó tương đương với ViewController.

Trong bất kỳ ứng dụng di động nào, điều hướng đến các trang khác nhau xác định quy trình làm việc của ứng dụng và cách xử lý điều hướng được gọi là định tuyến. Flutter cung cấp một lớp định tuyến cơ bản MaterialPageRoute và hai phương thức Navigator.push () và Navigator.pop () cho biết cách điều hướng giữa hai tuyến đường. Các bước sau là bắt buộc để bắt đầu điều hướng trong ứng dụng của bạn.

Bước 1: Đầu tiên, bạn cần tạo hai tuyến đường.

Bước 2: Sau đó, điều hướng đến một tuyến đường từ một tuyến đường khác bằng cách sử dụng phương thức Navigator.push().

Bước 3: Cuối cùng, điều hướng đến tuyến đường đầu tiên bằng cách sử dụng phương thức Navigator.pop ().

1.10.1 Tạo routes

Ở đây, chúng ta sẽ tạo hai tuyến đường để điều hướng. Trong cả hai tuyến đường, chúng ta chỉ tạo một nút duy nhất. Khi chúng ta nhấn vào nút trên trang đầu

tiên, nó sẽ điều hướng đến trang thứ hai. Một lần nữa, khi chúng ta nhấn vào nút trên trang thứ hai, nó sẽ trở lại trang đầu tiên. Đoạn mã dưới đây tạo ra hai tuyến đường trong ứng dụng Flutter.

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
 ),
      body: Center(
        child: RaisedButton(
          child: Text('Open route'),
          onPressed: () {
// Navigate to second route when tapped.
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
          },
          child: Text('Go back!'),
  ),
```

1.10.2 Điều hướng đến route thứ hai bằng phương thức Navigator.push()

Phương thức Navigator.push() được sử dụng để điều hướng / chuyển sang một tuyến đường / trang / màn hình mới. Ở đây, phương thức push () thêm một trang / tuyến đường trên ngăn xếp và sau đó quản lý nó bằng cách sử dụng Bộ điều hướng. Một lần nữa, chúng ta sử dụng lớp MaterialPageRoute cho phép chuyển đổi giữa các tuyến bằng cách sử dụng hoạt ảnh dành riêng cho nền tảng. Đoạn mã dưới đây giải thích việc sử dụng phương thức Navigator.push().

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondRoute()),
  );
}
```

1.10.3 Quay lại route đầu tiên bằng phương thức Navigator.pop ()

Bây giờ, chúng ta cần sử dụng phương thức Navigator.pop () để đóng tuyến thứ hai và quay lại tuyến đầu tiên. Phương thức pop() cho phép chúng ta loại bỏ tuyến đường hiện tại khỏi ngăn xếp, được quản lý bởi Bộ điều hướng.

Để triển khai quay lại tuyến ban đầu, chúng ta cần cập nhật phương thức gọi lại onPressed () trong tiện ích con SecondRoute như đoạn mã bên dưới:

```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);  }
```

1.11. Tìm hiểu về Database trong Flutter

Cơ sở dữ liệu (Database) là một tập hợp dữ liệu có tổ chức, hỗ trợ việc lưu trữ và thao tác dữ liệu và được truy cập điện tử từ hệ thống máy tính. Chúng ta có thể tổ chức dữ liệu thành các hàng, cột, bảng và chỉ mục. Nó giúp cho việc quản lý dữ liệu trở nên dễ dàng. Chúng tôi có thể lưu trữ nhiều thứ trong cơ sở dữ liệu, như tên, tuổi, hình ảnh, hình ảnh, tệp, pdf, v.v.

Flutter cung cấp nhiều gói để làm việc với cơ sở dữ liệu. Các gói phổ biến và được sử dụng nhiều nhất là:

1. **Cơ sở dữ liệu sqflite:** Nó cho phép truy cập và thao tác với cơ sở dữ liệu SQLite.
2. **Cơ sở dữ liệu Firebase:** Nó sẽ cho phép bạn truy cập và thao tác với cơ sở dữ liệu đám mây.

1.11.1 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. Sqflite cung cấp hầu hết các chức năng cốt lõi liên quan đến 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.

Các bước lưu trữ, tìm và lưu dữ liệu

Bước 1: Đầu tiên, tạo một dự án mới trong Android Studio và thêm các phần phụ thuộc vào tệp pubspec.yaml.

```
dependencies:

  flutter:

    sdk: flutter

  sqflite: any

  path_provider: any
```

Gói sqflite cung cấp các lớp và chức năng để tương tác với cơ sở dữ liệu SQLite.

Các path_provider gói cung cấp các chức năng để xác định vị trí của cơ sở dữ liệu của bạn trên hệ thống địa phương, chẳng hạn như TemporaryDirectory và ApplicationDocumentsDirectory .

Bước 2: Tạo một lớp mô hình. Ở bước này, chúng ta phải xác định dữ liệu cần lưu trữ trước khi tạo bảng để lưu trữ thông tin. Đoạn mã sau đây giải thích nó một cách dễ dàng.

```
class Book {
  final int id;
  final String title;
  final int price;

  Book({this.id, this.title, this.price});
}
```

Bước 3: Mở cơ sở dữ liệu. Tại đây, chúng ta cần mở kết nối với cơ sở dữ liệu. Nó yêu cầu hai bước:

- Đặt đường dẫn đến cơ sở dữ liệu bằng cách sử dụng phương thức getDtabasePath () và kết hợp nó với gói đường dẫn.
- Sử dụng hàm openDatabase () để mở cơ sở dữ liệu.

```
final Future<Database> database = openDatabase(
  join(await getDatabasesPath(), 'book_database.db'),
);
```

Bước 4: Tạo bảng. Trong bước này, chúng ta phải tạo một bảng lưu trữ thông tin về các cuốn book. Ở đây, chúng ta sẽ tạo một bảng có tên book, trong đó có id, tên book và giá của book. Chúng được thể hiện dưới dạng ba cột trong bảng book.

```
final Future<Database> database = openDatabase(
  join(await getDatabasesPath(), 'book_database.db'),
  // When you create a database, it also needs to create a table to
store books.
  onCreate: (db, version) {
    return db.execute(
      "CREATE TABLE books(id INTEGER PRIMARY KEY, title TEXT, price
INTEGER)",
    );
  },    version: 1,
);
```

Bước 5: Chèn book vào cơ sở dữ liệu. Ở đây, bạn phải lưu trữ thông tin trên bảng về các cuốn book khác nhau. Chèn một cuốn book vào bảng bao gồm hai bước:

- Chuyển book thành map
- Sử dụng phương thức insert()

```
class Book{
  final int id,
  final String title;
  final int price;

  Book({this.id, this.title, this.price});

  // It converts a Book into a Map. The keys correspond to the names
of the columns in the database.
  Map<String, dynamic> toMap() {
    return {
      'id': id,
```

```
      'title': title,
      'price': price,
Future<void> insertBook(Book book) async {
  final Database db = await database;
  await db.insert(   'books',
    book.toMap(),
    conflictAlgorithm: ConflictAlgorithm.replace,
  );}
```

Bước 6: Lấy danh book book. Bây giờ, chúng ta đã lưu trữ book vào cơ sở dữ liệu và bạn có thể sử dụng truy vấn để truy xuất một cuốn book cụ thể hoặc danh book tất cả các cuốn book. Nó bao gồm hai bước:

Chạy một truy vấn trả về List<Map>.

Chuyển List<Map> thành List<book>.

```
Future<List<Book>> books() async {
  final Database db = await database;

  // Use query for all Books.
  final List<Map<String, dynamic>> maps = await db.query('maps');

  return List.generate(maps.length, (i) {
    return Book(
      id: maps[i]['id'],
      title: maps[i]['title'],
      price: maps[i]['price'],
    );
  });
}

// It prints all the books.
print(await books());
```

Bước 7: Cập nhật book trong cơ sở dữ liệu. Bạn có thể sử dụng phương thức update() để cập nhật book mà bạn muốn. Nó bao gồm hai bước:

- Chuyển đổi book thành Bản đồ.
- Sau đó, sử dụng mệnh đề where để cập nhật book.

Bước 8: Xóa book khỏi cơ sở dữ liệu. Bạn có thể sử dụng phương thức delete () để xóa cơ sở dữ liệu. Đối với điều này, bạn cần tạo một hàm lấy id và xóa cơ sở dữ liệu của id phù hợp.

```
Future<void> deleteBook(int id) async {
  final db = await database;
  // This function removes books from the database.
  await db.delete(
    'books',
    where: "id = ?",
    whereArgs: [id],
  );  }
```

Thông qua các bước làm việc với cơ sở dữ liệu SQLite thì chúng ta có thể thấy được các bước thêm mới xóa sửa cơ sở dữ liệu trong Flutter.

1.11.2 Firebase – NoSQL lưu trữ online

FireBase là một dạng database lưu trức 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 kiểu 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 tại đây. Nhược điểm duy nhất của phương pháp này là bạn không nhận được lượng dữ liệu mạnh giống như cách bạn làm với Moor hoặc Hive. Bạn sẽ phải tự làm việc này bằng tay.

Ưu điểm :

- Đồng bộ hóa với Firebase trực tuyến theo thời gian thực.
- Hỗ trợ công cụ tuyệt vời.
- Dễ dàng duyệt dữ liệu trực tuyến thông qua bảng điều khiển Firebase.

Nhược điểm :

- Thiết lập Firebase có thể phức tạp nếu bạn đã thêm nó vào ứng dụng của mình.
- Vì cơ sở dữ liệu đang trực tuyến, bạn cần chú ý nhiều hơn về cơ sở dữ liệu trên thiết bị (ví dụ như quyền truy cập).
- FireBase chưa sẵn sàng để hỗ trợ cho Flutter.

CHƯƠNG 2 TÌM HIỂU VỀ WEBSITE

2.1 Website là gì

Website còn gọi là trang mạng (có thể nhầm lẫn với "web page"), là một tập hợp trang web con, bao gồm hình ảnh, văn bản, hình ảnh, video,flash,...vv thường chỉ nằm trong một tên miền hoặc tên miền phụ trên World Wide Web của Internet. Một trang web là tập tin HTML hoặc XHTML có thể truy nhập dùng giao thức HTTP.

Trang mạng có thể được xây dựng từ các tệp tin HTML (trang mạng tĩnh) hoặc vận hành bằng các CMS chạy trên máy chủ (trang mạng động). Trang mạng có thể được xây dựng bằng nhiều ngôn ngữ lập trình khác nhau (PHP, ASP.NET, Java, Ruby on Rails, Perl,...)[2].

Trang web được giao tiếp và hiển thị cho người dùng truy cập bằng các phần mềm được gọi là trình duyệt web. Một số trình duyệt website được cài đặt mặc định vào mỗi máy tính cài hệ điều hành windows và được thay thế bởi Edge trên Windows 10 hay Chrome được phát triển bởi Google và Firefox được phát triển bởi Mozilla.

2.2 Cấu tạo của trang web

Các thành phần của website:

Web server (riêng) thường dành cho những website lớn, các website nhỏ và trung bình thì thường sử dụng một phần nhỏ tài nguyên của web server. Web server máy chủ cài đặt các chương trình phục vụ các ứng dụng web. Webserver có khả năng tiếp nhận request từ các trình duyệt web và gửi phản hồi đến client thông qua giao thức HTTP hoặc các giao thức khác.

Tên miền (domain):

Có thể hiểu tên miền là tên thay thế cho địa chỉ IP của máy chủ web, bởi địa chỉ IP là một dãy số rất khó nhớ. Do đó người ta gán (định danh) địa chỉ IP thành một chuỗi ký tự và nó dễ nhớ hơn (ví dụ: thay vì ghi nhớ địa chỉ 210.211.113.135 ta sẽ nhớ chuỗi "hocban.vn" và tương tự, điều này được thực hiện với hàng triệu website khác). Việc gán (ánh xạ) tên miền cho địa chỉ IP được thực hiện bởi hệ thống phân giải tên miền (DNS);

Dữ liệu:

Dữ liệu người dùng – hay cơ sở dữ liệu: tạm hiểu là những thông tin được lưu trữ về người dùng như: Tên đăng nhập, mật khẩu, nhật ký hoạt động (viết, chỉnh sửa bài viết, thiết đặt website,..vv..); Dữ liệu website: gọi chung cho tất cả các

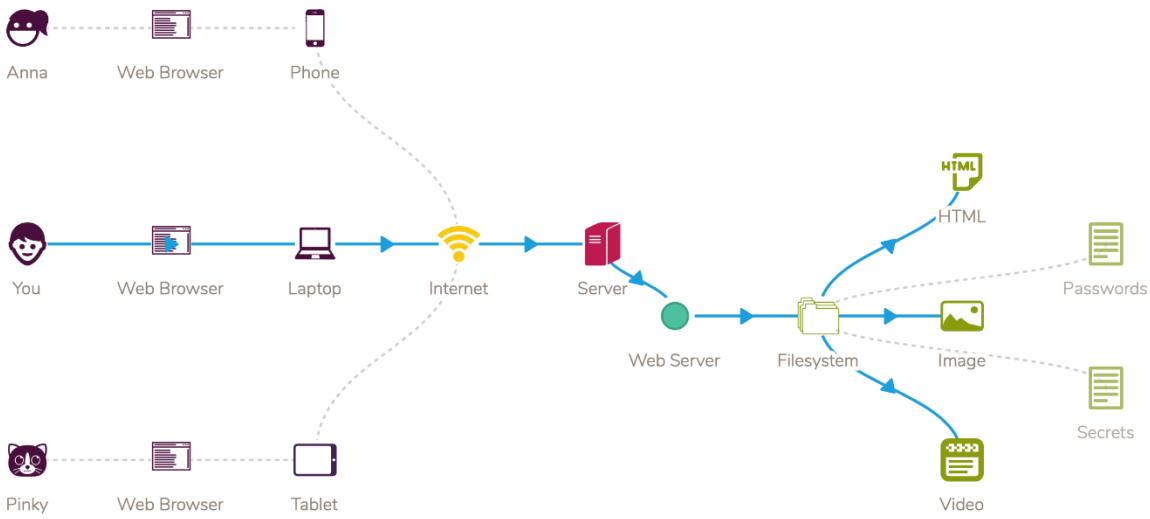
tập tin đa phương tiện như văn bản, âm thanh, hình ảnh, video,.. được lưu trữ trên máy chủ web.

Mã nguồn:

Để dễ hình dung, ở đây xem như mã nguồn là một phần mềm hoàn chỉnh nó được cài đặt lên web server/hosting của bạn, nó như công cụ để tạo lập và quản lý nội dung website. Hiện nay mã nguồn web thường là các phần mềm quản trị nội dung như Joomla, WordPress, Drupal,…

Giao diện người dùng:

Giao diện người dùng là tất cả sự bố trí, trình bày nội dung trên website, bao gồm: bố cục, màu sắc, font chữ, hiệu ứng,… mà người truy cập website có thể thấy và tương tác; Giao diện người dùng là những gì mà người dùng nhìn thấy sau khi các đoạn mã đằng sau nó được dịch (đằng sau một website "đẹp lung linh – sinh động " mà chúng ta nhìn thấy là những đoạn chương trình, dưới dạng ngôn ngữ HTML xen lẫn với ngôn ngữ tự nhiên).



Hình 2.1: Mô hình cơ bản về hoạt động của một website trên Internet

Các bước cơ bản để truy cập một trang web trên Internet:

Bước 1: Sau khi bạn mở trình duyệt, nhập địa chỉ một trang web (ví dụ https://nth.vn) và bấm Enter, sau khoảng vài giây nội dung website sẽ được hiện ra trên trình duyệt với nhiều đoạn văn bản, có thể gồm cả hình ảnh, âm thanh hay video,… Quá trình đã xảy ra từ khi bạn Enter có thể được mô tả tóm tắt như sau:

Bước 2: Trình duyệt Web sẽ thực hiện một truy vấn dựa vào tên miền (domain) để tìm ra địa chỉ IP thực sự của web server tương ứng chứa website có tên miền đó bằng một giao thức đặc biệt gọi là DNS.

Bước 3:Sau khi đã tìm được địa chỉ IP, trình duyệt sẽ gửi gói tin yêu cầu – HTTP request đến địa chỉ của web server, yêu cầu trả về nội dung trang web. Gói tin yêu cầu đó cũng như tất cả các gói tin, dữ liệu khác trao đổi giữa máy chủ với máy chúng ta (gọi là máy khách) được thực hiện qua một bộ giao thức TCP/IP.

Bước 4: Khi nhận được các yêu cầu từ máy khách, máy chủ web sẽ tiến hành kiểm tra và nếu có thể đáp ứng các yêu cầu đó thì nó sẽ gửi lại cho máy khách các tập tin được yêu cầu. Thông thường máy chủ sẽ trả về tập tin HTML để hiển thị trên trình duyệt, có liên kết đến những tập tin hình ảnh, âm thanh,… khác. Các tập tin này có thể được chia thành nhiều gói tin (packets) nhỏ và gửi về cho trình duyệt của người dùng đang ở máy khách.

Bước 5: Khi nhận được, trình duyệt sẽ ghép những gói tin nhỏ nhận được thành những tập tin hoàn chỉnh và hiển thị lên màn hình. Như thế là chúng ta có một trang web hoàn chỉnh để xem.

2.3 Các loại website và phân loại Website

Ở thời điểm ngành công nghệ thông tin và mạng Internet phát triển mạnh mẽ như hiện nay, website được xem là công cụ đơn giản, phổ biến và rất quan trọng đối với hoạt động quảng cáo, kinh doanh, bán hàng của các tổ chức, doanh nghiệp, cá nhân. Giống như những mặt hàng khác trên thị trường, website hay trang web cũng bao gồm rất nhiều loại nhằm đáp ứng đầy đủ nhất cho nhu cầu sử dụng đa dạng của người dùng.

Để phân loại một cách tổng quan, chính xác, đầy đủ về website, chúng ta có thể dựa vào 4 yếu tố bao gồm: cấu trúc website, quyền sở hữu, chức năng và lĩnh vực hoạt động.

2.3.1 Phân loại website theo cấu trúc website

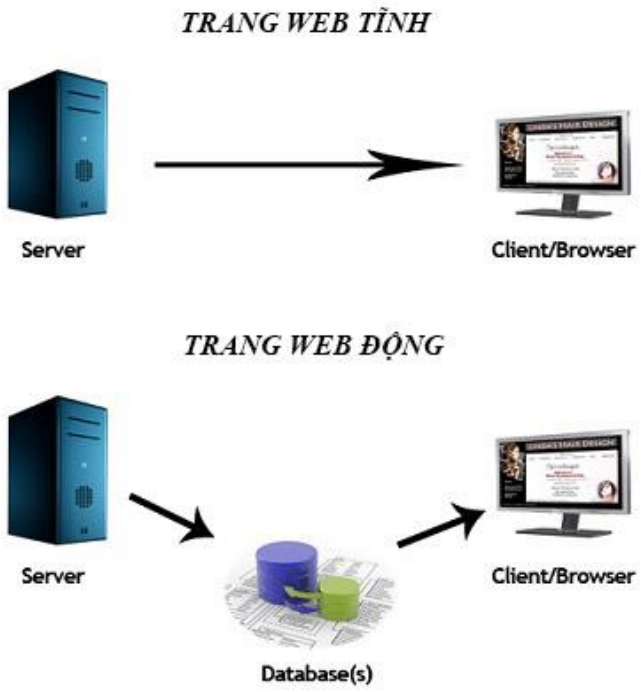
Phân loại website theo cấu trúc cũng có thể hiểu là dạng dữ liệu, cách vận hành của trang web. Hiện nay có 02 loại cấu trúc web bao gồm:

2.3.1.1 Website tĩnh (static website)

Web tĩnh ở đây được hiểu theo nghĩa là dữ liệu web không được thay đổi thường xuyên. Loại web này được lập trình dựa trên nền tảng HTML, CSS và Javascript. Nếu muốn thay đổi nội dụng trên web, quản trị viên phải sửa đổi trực tiếp trên mã lệnh và chỉ những người am hiểu về ngôn ngữ lập trình mới có thể thực hiện thao tác này. Chính vì thế mà trang web tĩnh hiện nay không được sử dụng phổ biến.

2.3.1.2 Website động (dynamic website)

Hầu hết các trang web hiện nay đều thuộc cấu trúc website động. Loại web này sử dụng nền tảng HTML, CSS, Javascript, PHP hoặc ASP.NET….Quản trị viên của trang web động có thể thoải mái cập nhật thông tin, thêm bớt module,…cho trang một cách đơn giản và nhanh chóng.



Hình 2.2: Các hoạt động của trang web động và trang web tĩnh

2.3.2 Phân loại website theo quyền sở hữu

2.3.2.1 Website doanh nghiệp

Đây là cổng thông tin điện tử chính thức của doanh nghiệp, tạo ra với mục đích giới thiệu công ty, cập nhật thông tin hoạt động, quảng bá sản phẩm dịch vụ và còn rất nhiều chức năng khác. Tất cả các đơn vị kinh doanh hiện nay đều có nhu cầu quảng bá thương hiệu, hình ảnh của mình nên website doanh nghiệp được xem là một phần không thể thiếu đối với các công ty hiện nay.

2.3.2.2 Website cá nhân

Nếu như trang web công ty thuộc sở hữu chung của một doanh nghiệp thì trang web cá nhân chỉ thuộc quyền sở hữu của một người nào đó. Bất cứ ai cũng có thể tạo một web cá nhân để phục vụ cho bất kỳ mục đích nào của mình. Tuy nhiên, loại web này chỉ phổ biến với người của công chúng, những người cần quảng bá hình ảnh cá nhân để phục vụ cho mục đích thương mại hay công việc. Chẳng hạn như: chính trị gia, ca sĩ, nhà thiết kế, nhà văn, nhà thuyết giảng….

2.3.3 Phân loại website theo chức năng

Xét về chức năng, trang web có thể được chia làm rất nhiều loại trang wed sau đây:

2.3.3.1 Trang web bán hàng

Loại web này bao gồm trang web thương mại điện tử tổng hợp hoặc trang bán hàng của một đơn vị, cá nhân cụ thể. Website thương mại bán hàng cho phép người dùng đặt hàng và lựa chọn các hình thức thanh toán trực tiếp trên web. Chủ sở hữu trang web này trực tiếp quản lý việc buôn bán của mình hoặc có thể cho các cửa hàng khác thuê lại.

2.3.3.2 Trang web tin tức

Đây là loại website cung cấp thông tin văn hóa, chính trị, xã hội, sức khỏe, giáo dục…cho độc giả, được phát triển từ nền tảng báo giấy truyền thống. Chủ sở hữu của dạng web này thường là các cơ quan Nhà nước, hiệp hội, tổ chức….

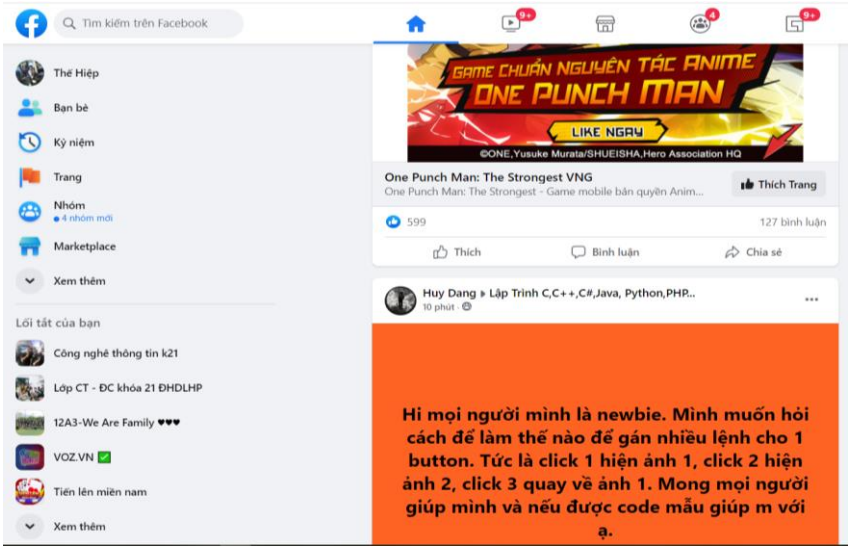
2.3.3.3 Mạng xã hội

Mạng xã hội hay blog là một dạng web cho phép mỗi cá nhân và doanh nghiệp tạo ra một không gian riêng cho chính mình. Bạn sẽ có thể đăng tải thông tin, viết nhật ký, chia sẻ bất cứ điều gì mà mình quan tâm trên trang cá nhân. Phần lớn người dùng mạng xã hội là để phục vụ cho nhu cầu giải trí, kết nối, liên lạc với bạn bè. Những trang mạng xã hội phổ biến nhất hiện nay như: Facebook, Twitter, Instagram, Youtube, Zalo….

Đặc điểm của mạng xã hội là ứng dúng được sử dụng trên nền tảng internet, tất cả nội dung đều do người dùng tạo ra và chia sẻ. Mỗi người dùng mạng xã hội cần phải tạo tài khoản và hồ sơ cá nhân riêng, và Mạng xã hội sẽ kết nối tài khoản người dùng đến các tài khoản cá nhân, tổ chức khác thông qua các tài khoản ảo do người dùng tạo ra.

Mục tiêu của mạng xã hội là tạo ra một hệ thống cho phép người dùng có thể kết nối, giao lưu, chia sẻ những thông hữu ích trên nền tảng Internet. Ngoài ra mạng xã hội còn có mục tiêu là xây dựng lên một cộng đồng có giá trị, nâng cao vai trò của mỗi người dùng trong việc xây dựng các mối quan hệ.

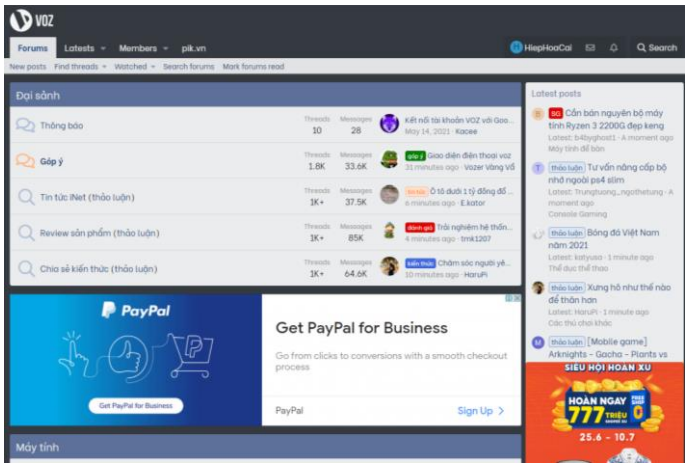
Hiện nay, mạng xã hội đã đem lại cho chúng ta rất nhiều lợi ích và không ít những tác hại có thể gây nghiện hoặc là bạo lực trên mạng. Nhưng chúng ta không thể phủ nhận được lợi ích mà nó mang lại là rất to lớn.



Hình 2.3: Mạng xã hội Facebook

2.3.3.4 Diễn đàn / Forum

Diễn đàn là nơi để tất cả những người có cùng một sở thích hay sự quan tâm, điểm chung nào đó cùng tham gia chia sẻ, trao đổi kiến thức, kinh nghiệm… với nhau. Bất cứ ai cũng có thể tạo tài khoản và tham gia vào diễn đàn và dĩ nhiên sẽ có một hoặc nhiều quản trị viên của trang kiểm duyệt nội dung trên diễn đàn.



Hình 2.4: Diễn đàn công nghệ Voz

2.3.3.5. Trang web giải trí

Đây là dạng web phục vụ cho một nhu cầu giải trí cụ thể nào đó, chẳng hạn như: nghe nhạc, xem phim hay chơi game online…. Ngày càng có nhiều trang web giải trí khác nhau ra đời, nhằm phục vụ cho nhu cầu thư giãn, giải trí của người dùng và mục đích kinh doanh của chủ sở hữu.

2.3.3.6. Trang web rao vặt

Trang web rao vặt có chức năng tương tự như một khu chợ online. Tại đây, bất cứ ai cũng có thể giới thiệu, đăng tải thông tin mặt hàng mình muốn rao bán và người mua cũng sẽ dễ dàng tìm kiếm được sản phẩm mà mình cần sở hữu. Các

website rao vặt bao gồm rất nhiều loại hình nhưng phổ biến nhất là rao vặt bất động sản và trang rao vặt.

2.3.4 Phân loại trang web theo lĩnh vực hoạt động

Cách phân loại web này phổ biến hơn xuất phát từ mục đích sử dụng của người dùng Internet. Có vô số loại website được phân chia theo ngành nghề và lĩnh vực hoạt động như: web xây dựng, web du lịch, web giáo dục, web thời trang, web mỹ phẩm, web tin tức….

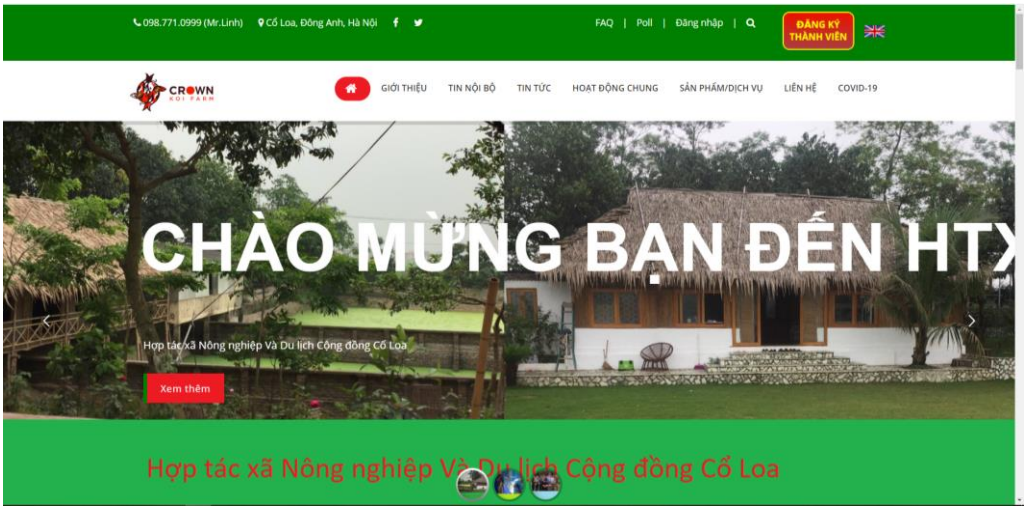
2.4 Tìm hiểu về trang Web Hợp Tác Xã Nông nghiệp và Du lịch Cộng đồng Cổ Loa

- Địa chỉ của website: https://coloagroup.com/

- Giới thiệu sơ bộ: Crown Koi Farm là công ty con thuộc Hợp tác xã Nông nghiệp và Du lịch Cộng đồng Cổ Loa phụ trách nhập khẩu, bán buôn bán lẻ mặt hàng cá Koi giống F0 trực tiếp từ trại Matsuda thuộc tỉnh Niigata, Nhật Bản.

Công ty hiện đặt trụ sở tại Cổ Loa, Đông Anh, Hà Nội. Lĩnh vực kinh doanh chủ yếu là cá Koi. Cá Koi là 1 giống cá cảnh bắt nguồn từ Nhật Bản, rất được ưa chuộng ở Việt Nam trong lĩnh vực cá cảnh, ngoài việc nuôi để trang trí tại sân vườn hay là phong thủy, thì còn rất nhiều người nuôi cá Koi vì đam mê. Bởi cá Koi là một loài cá đẹp thuộc họ cá chép , quý nhưng không hiếm, có rất nhiều màu sắc khác nhau, cực kì dễ nuôi do bản tính hiền lành, rất thân thiện với con người, nên có rất nhiều người nuôi để trang trí cũng như là giải trí sau những giờ làm việc căng thẳng. Chúng có thể sống trung bình từ 25 năm đến 35 năm, tuổi thọ rất cao nên được nhiều người săn đón, trong đó có cả các hội nuôi cá cảnh của Việt Nam. Ngoài ra, loài cá này được săn đón nhờ màu sắc sặc sỡ, những con cá Koi có màu sắc đặc biệt thường có giá rất cao, đem lại rất nhiều giá trị về kinh tế từ việc bán cá.

Hiện nay có rất nhiều người muốn tìm hiểu về loài cá Koi, mà họ lại lười truy cập lên các website tìm hiểu về giống cá này, nên em phát triển ra một ứng dụng Webview (hiển thị nội dung trang web ngay trên ứng dụng, không cần phải mở trình duyệt để xem các nội dung này) cho phép người dùng tải xuống ứng dụng, sử dụng để tìm hiểu về cá Koi mà không cần tốn quá nhiều thời gian để truy cập lên các trang web.



Hình 2.5: Trang web Hợp tác xã Nông nghiệp và Du lịch Cộng đồng Cổ Loa

CHƯƠNG 3 BẢN HƯỚNG DẪN SỬ DỤNG VÀ ỨNG DỤNG THỰC NGHIỆM

3.1. Bản hướng dẫn sử dụng

3.1.1 Bản hướng dẫn sử dụng trên hệ điều hành Windows

Flutter hiện là SDK khá phổ biến dành cho lập trình viên về thiết kế giao diện người dùng trên điện thoại. Ngoài flutter ra, vẫn còn framework khác phát triển phần mềm di động, tiêu biểu là React Native, do Facebook phát triển. Tuy nhiên, Flurer vẫn là lựa chọn hàng đầu, vì nó được hỗ trợ bởi Google và có một cộng đồng vô cùng đông đảo trên toàn thế giới.

Yêu cầu hệ thống

- Để cài đặt Flutter thì máy tính cần đáp ứng những yêu cầu sau đây:
- Windows 7 SP1 hoặc mới hơn (64-bit).

- Dung lượng trống 400MB (không bao gồm dung lượng cho phần IDE/tools).
- Các công cụ: Flutter phụ thuộc vào các công cụ có sẵn trong máy tính của bạn:
- PowerShell 5.0 hoặc mới hơn (cái này đã được tích hợp sẵn vào Windows 10).
- Git cho Windows 2.x, với tùy chọn chạy câu lệnh git từ cửa sổ lệnh Windows Command Prompt. (Nếu Git đã được cài, hãy chắc chắn rằng bạn có thể chạy câu lệnh git từ cửa sổ lệnh Windows Command Prompt).

Tiến hành cài đặt

Bạn duyệt tìm đến phần Get the Flutter SDK, bấm vào flutter_window _vxxxx -stable.zip (với xxxx là phiên bản flutter [4]), để tải flutter SDK về. Bạn có thể lưu ở đâu tùy thích sau khi giải nén mình có thể di chuyển nó sau.

Nếu không muốn cài đặt phiên bản cố định của gói cài đặt, bạn có thể tải xuống bằng cách khác. Sử dụng cmd(Command Prompt) để lấy mã nguồn từ kho lưu trữ Flutter trên GitHub và thay đổi các branches hoặc tags nếu cần.

```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

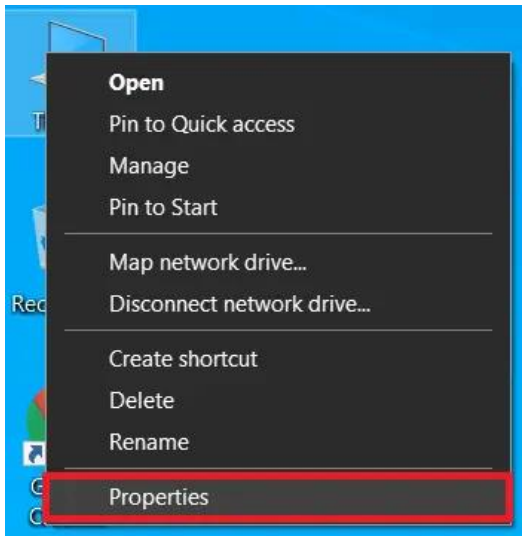
Sau khi tải về xong, các bạn giải nén file vừa tải về (tùy chọn Extract here) ta sẽ được một thư mục tên là flutter. Các bạn hãy để thư mục này vào một nơi nào đó mà bạn muốn (lưu ý không được đặt vào thư mục "C:\Program Files\" vì nó yêu cầu quyền riêng tư). Ví dụ mình sẽ làm y như trong docs của Google là tạo một thư mục mới tên là src đặt trong ổ đĩa "C:\" và copy thư mục flutter vào thư mục "C:\src\".

Bây giờ bạn đã sẵn sàng để chạy lệnh Flutter Console. Nhưng để có thể chạy lệnh flutter từ Command Prompt, bạn nên cập nhật đường dẫn(Path).

Thêm đường dẫn này là để khi truy cập từ Command Prompt, có thể kiểm tra được các công cụ của flutter đã cài đầy đủ chưa. Cách lấy thêm đường dẫn của flutter trên windows rất dễ dàng.

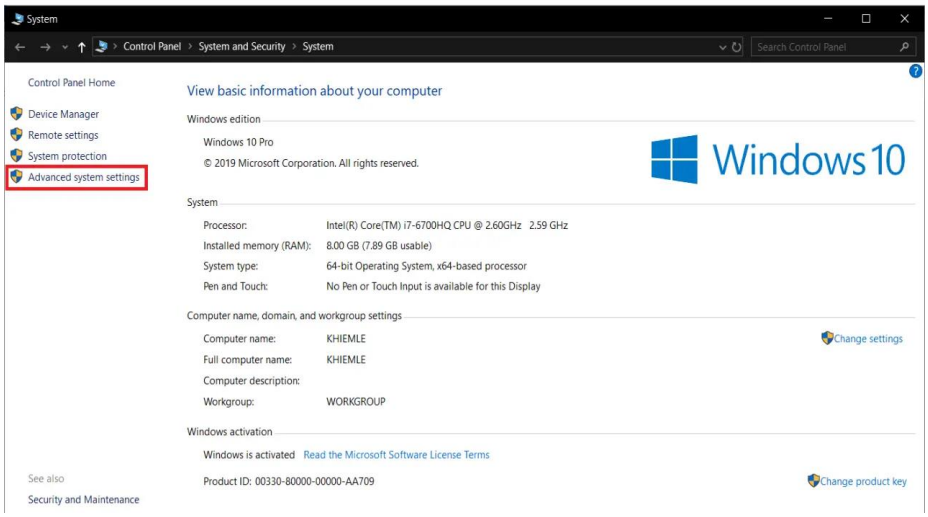
Các bước thực hiện như sau:

Bước 1: Click phải chuột vào This PC, chọn Properties



Hình 3.1: Nhấp chuột phải vào biểu tượng Proprerties

Bước 2: Chọn Advanced system settings (gần phía trên bên trái) trong cửa sổ System.

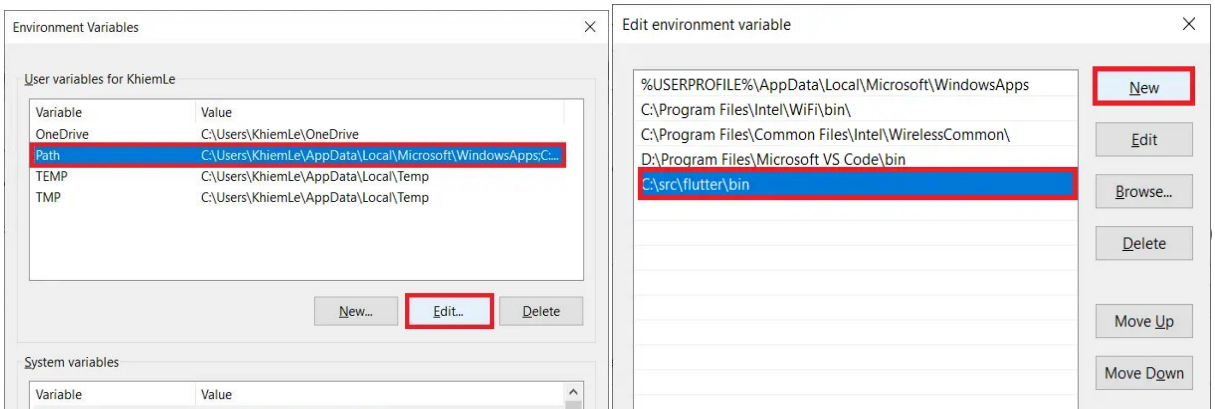


Hình 3.2: Truy cập Advanced system settings

Bước 3.Chọn Enviroment Variables… trong cửa sổ System Properties.

Bước 4.Trong phần User variables, các bạn tìm Variable là PATH và nhấn Edit, nhấn New và thêm đường dẫn đến thư mục “flutter\bin”. Ví dụ của mình là “C:\src\flutter\bin”. Nếu như bạn không tìm thấy Variable PATH bạn nhấn New và đặt tên Variable là PATH và đường dẫn đến thư mục flutter\bin của bạn.

Bước 5: Nhấn OK để lưu tất cả các thay đổi lại.



Hình 3.3: Sửa biến môi trường Path

Hình 3 4: Thêm đường dẫn thư mục bin của Flutter vào Edit environment variable

Vậy là đã có thể chạy lệnh flutter từ Command Prompt rôi. Giờ hãy check thử. Mở Command Prompt lên và chạy lệnh "flutter –version". Nếu như nó hiện lên thông tin Flutter, framwork, dart version… thì bạn đã cài thành công. Tiếp theo, sẽ phải cài đặt Plugins cho IDE để code với Flutter.

Việc thêm Path (đường dẫn) khá quan trọng trong việc kiểm tra về môi trường lập trình cho flutter, kiểm tra tool cho flutter, kiểm tra máy ảo, và các package. Nhưng không thêm đường dẫn cũng không sao nếu, đi mượn thiết bị của người khác mà chỉ cần chạy ra kết quả. Thì không bắt buộc phải thêm path cho flutter.

Cài đặt Plugins Flutter

Google hỗ trợ hai IDE đó là Android Studio và Visual Studio Code. Đương nhiên là mọi người vẫn có thể sử dụng các IDE khác, và nếu có thì nó cũng là do cá nhân tự phát triển, không đầy đủ bằng. Ở trong đồ án lần này, sẽ chỉ hướng dẫn cài đặt trên Android Studio trên cả hệ điều hành windows và hệ điều hành macOs

Đối với Android Studio:

Bước 1: Khởi động Android Studio

Ở màn hình Welcome to Android Studio, mở menu Configure (phía dưới góc phải), chọn Plugins để thêm các plugins cần thiết để giúp bạn lập trình.

Ở tab Marketplace, các bạn search "flutter", bạn nhấn install vào plugin Flutter ngay kết quả đầu tiên. Plugin này yêu cầu các bạn cài đặt thêm một plugin nữa là Dart, chọn Yes để cho phép cài đặt Dart nữa. (Hoặc có thể cài đặt 2 plugin này sau khi đã cài được cái thiết bị máy ảo cũng như là hoàn thiện đủ các tool, Bằng cách truy cập vào settings của android studio để cài đặt.)

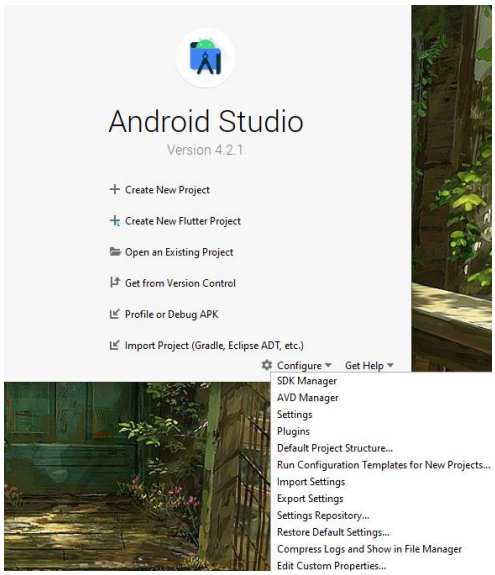
Cuối cùng các bạn RESTART IDE để áp dụng thay đổi, ấn khởi động lại IDE nhằm mục đích để IDE chạy được mượt mà.

Tạo máy ảo Android Studio

Để tạo một máy ảo Android Studio, các bạn làm theo các bước sau:

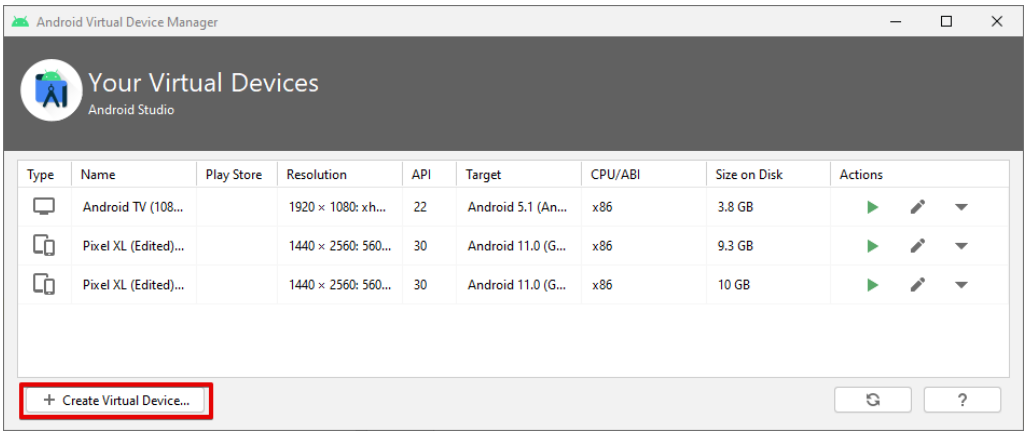
Bước 1: Khởi động Android Studio

Vào menu Configure và chọn AVD Manager, để truy cập vào mục các thiết bị của bạn.



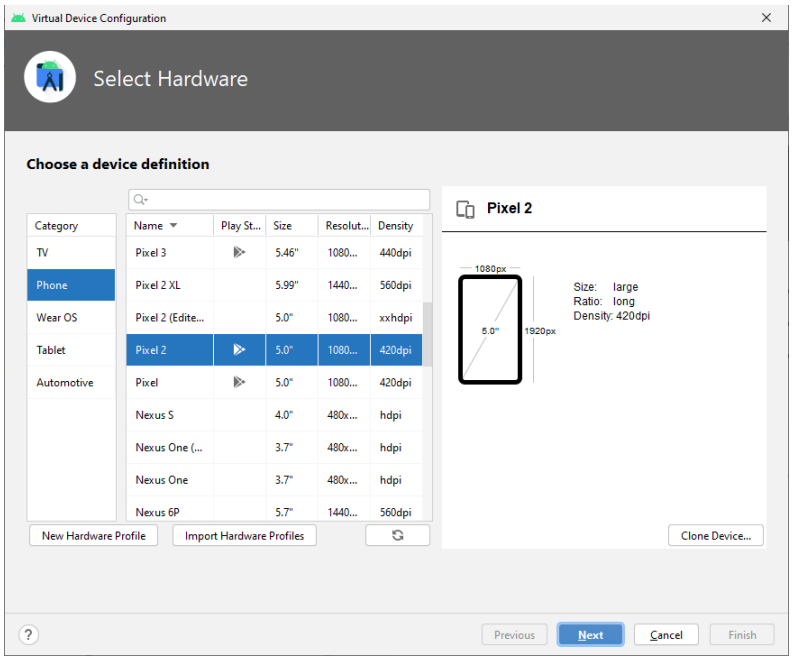
Hình 3.5: Mở AVD Manager

Bước 2: Chọn Create Virtual Device



Hình 3.6: Chọn tạo mới một thiết bị máy ảo

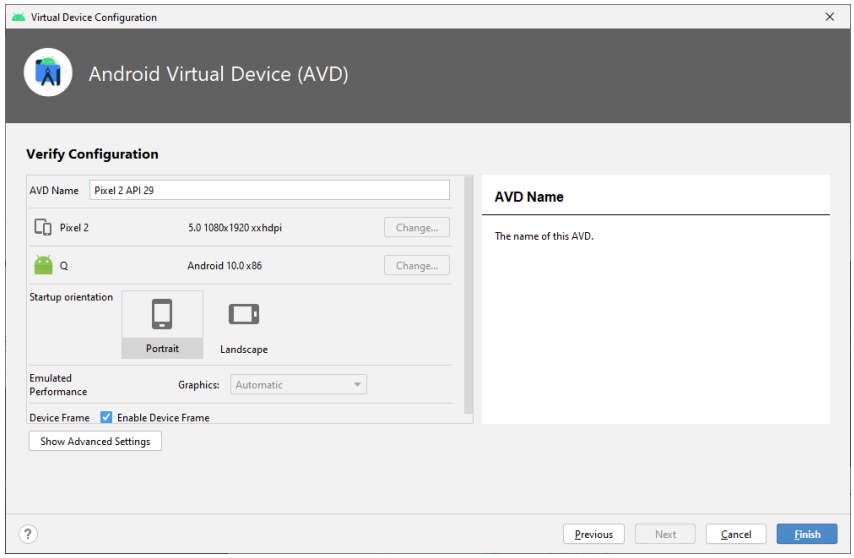
Bước 3: Chọn một thiết bị và nhấn Next



Hình 3.7: Thiết lập chọn cấu hình phần cứng

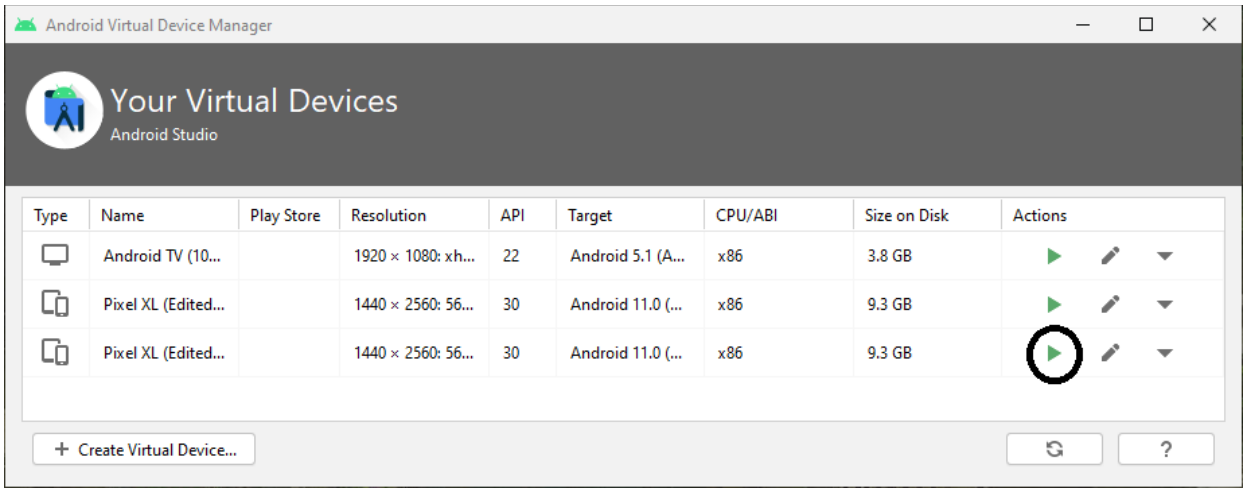
Cách chọn cấu hình cũng tùy thuộc vào sự lựa chọn của người dùng. Các bạn có thể tùy chọn size của điện thoại độ phân giải và thiết bị có Play Store để tạo, ngoài ra còn việc chọn hệ điều hành cho thiết bị android. Hệ điều hành thiết bị android sẽ từ 5.0 (Lolipop) trở lên. Sẽ rất phù hợp cho kiểm thử, để kiểm tra ứng đụng đó có tương thích phiên bản thấp hay không.

Trong phần Graphics, bạn chọn Hardware – GLES 2.0. Bạn có sửa các tùy chọn khác nếu muốn và nhấn Finish để tạo máy ảo. Đây là phần cuối cùng để tạo ra máy ảo, có các thông số từ RAM, bộ nhớ trong, và cấu hình phần cứng thiết bị bạn vừa chọn



Hình 3.8: Cấu hình máy ảo trong android studio

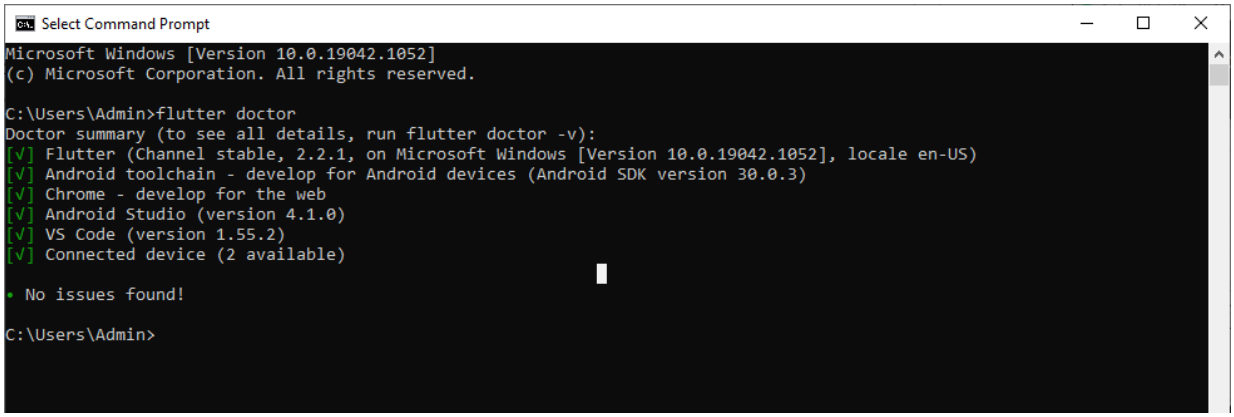
Bước 4: Nhấn nút ▶ để chạy máy ảo vừa tạo



Hình 3.9: Bảng AVD nơi chứa thông tin các máy ảo đã tạo

Kiểm tra

Kiểm tra lại trước khi hoàn tất bằng cách mở Command Prompt, gõ lệnh "flutter doctor" và nhấn enter, sau khi câu lệnh chạy xong, kiểm tra xem các mục Flutter, Android toolchain, Android Studio, Connected device. Nếu các thiết bị chưa cài đặt thành công nó sẽ hiện chữ "[X]".



Hình 3.10: Chạy lệnh flutterdoctor trong cmd để kiểm tra môi trường lập trình, thiết bị, và các gói

3.1.2 Bản hướng dẫn sử dụng trên hệ điều hành Mac OS

Tương tự như viết ứng dụng trên windows, cái khác biệt khi sử dụng trên macOs chính là cách cài đặt.

Để lấy Flutter SDK, chọn bản releases mới nhất bằng cách truy cập trang chủ của Flutter để lấy file Unzip file flutter_macos_v1.2.1-stable.zip[3].

Kiểm tra biến môi trường: echo $PATH

Thêm 1 PATH mới: export PATH="$PATH:'pwd'/flutter/bin"

Lưu ý: PATH="$PATH:'pwd'/flutter/bin" đây chỉ là cách thêm path tạm thời sau khi khởi động lại máy tính, đường dẫn sẽ mật.

Để thêm 1 PATH luôn tồn tại trong cách phiên đăng nhập

```
cd ~/
touch .bash_profile
open .bash_profile
```

Điều đó sẽ mở .bash_profile trong trình chỉnh sửa, viết bên trong phần sau đây sau khi thêm những gì bạn muốn vào đường dẫn phân tách từng giá trị theo cột.

```
export PATH=/usr/local/git/bin:/usr/local/bin:
```

Lưu, thoát, khởi động lại thiết bị đầu cuối.

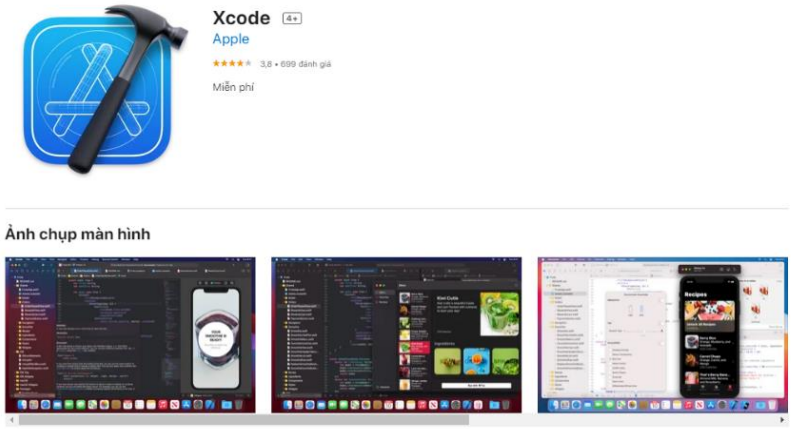
Cài đặt Xcode:

XCode là bộ phát triển phần mềm tích hợp được Apple phát triển chạy trên hệ điều hành Mac để các lập trình viên có thể phát triển phần mềm chạy trên hệ điều hành Mac và iOS. Ứng dụng này cực kì quan trọng đối với việc lập trình trên mobile trên máy Mac, Xcode hỗ trợ việc tạo máy ảo, hỗ trợ run trên thiết bị iOS thật, và hỗ trợ việc tạo ra file để có thể đưa lên AppStore.

Cách tải 1: mở terminal và gõ

```
$ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
$ sudo xcodebuild -runFirstLaunch
```

Cách 2 : truy cập AppStore của máy mac tìm kiếm và tải thẳng từ trên AppStore về



Hình 3.11: Ứng dụng Xcode trên App Store

Để chuẩn bị chạy và kiểm tra ứng dụng Flutter của bạn trên simulator iOS, tìm simulator qua Spotlight hoặc bằng cách sử dụng lệnh sau:

Set up the iOS simulator: open -a Simulator

Hoặc truy cập Xcode tool để mở 1 thiết bị hệ điều hành iOS ảo.

Phát triển cho các thiết bị iOS:

Run flutter doctor: flutter doctor để flutter tự động kiểm tra các gói cài đặt trên máy bạn đã phù hợp và đã đầy đủ hay chưa.

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Mac OS X 10.14.4 18E226, locale vi-VN)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from:
      https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK
      components.
      (or visit https://flutter.io/setup/#android-setup for detailed
      instructions).
      If Android SDK has been installed to a custom location, set ANDROID_HOME
      to that location.
      You may also want to add it to your PATH environment variable.

[!] iOS toolchain - develop for iOS devices (Xcode 10.2.1)
    ✗ libimobiledevice and ideviceinstaller are not installed. To install with
      Brew, run:
        brew update
        brew install --HEAD usbmuxd
        brew link usbmuxd
        brew install --HEAD libimobiledevice
        brew install ideviceinstaller
    ✗ ios-deploy not installed. To install:
        brew install ios-deploy
[!] Android Studio (not installed)
[!] VS Code (version 1.32.3)
    ✗ Flutter extension not installed; install from
      https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter
[✓] Connected device (1 available)
```

Hình 3.12: chạy lệnh flutter doctor trên terminal(iOS)

Ở đây chúng ta sẽ bỏ qua Android toolchain, mà sẽ dùng đến iOS toolchain Install homebrew [5], thằng này quản lý package của macOS Homebrew giúp cài thêm các phần mềm, thư viện có trong Linux, Unix nhưng lại không sẵn cài trong MacOSX. chưa kể sẽ phải cấu hình môi trường thì phần mềm mới có thể chạy được.

Homebrew là một hệ thống quản lý gói phần mềm mã nguồn mở và miễn phí. Giúp đơn giản hóa việc cài đặt phần mềm trên hệ điều hành macOS của Apple cũng như Linux.

Homebrew được viết bằng Ruby ngôn ngữ kịch bản, được bổ sung thêm nhiều lệnh để đơn giản tối đa việc cấu hình, biên dịch, cài đặt, thiết lập môi trường cho một ứng dụng chạy trơn chu. Cài rồi thì update mới nhất: brew update

Truy cập terminal của Mac tương tự như cmd của windows gõ lệnh:

```
brew install --HEAD usbmuxd

brew link usbmuxd
```

```
brew install --HEAD libimobiledevice

brew install ideviceinstaller ios-deploy cocoapods

pod setup
```

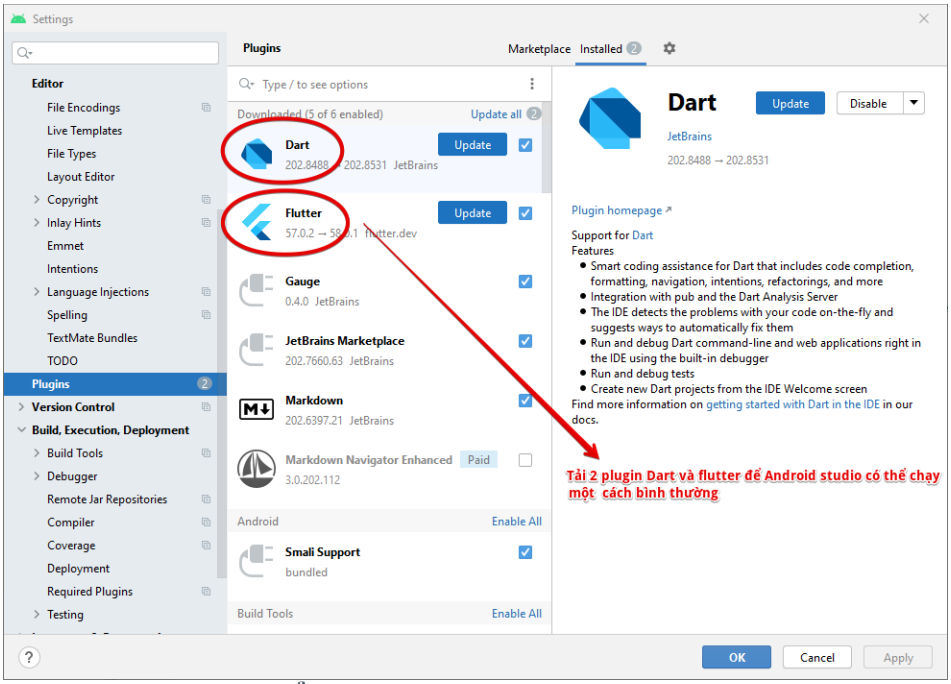
Sau khi cài xong libmobiledevice và cocoapods thì run lệnh flutter doctor, để kiểm tra đã đủ các gói đã cài đầy đủ hay chưa.
Thêm plugin Dart và Flutter :

Bước 1: Open Android Studio chọn File.

Bước 2: Mở File chọn settings

Bước 3: Trong mục settings chọn Plugin

Bước 4: Tìm plugin Flutter và plugin Dart tải về và kích hoạt



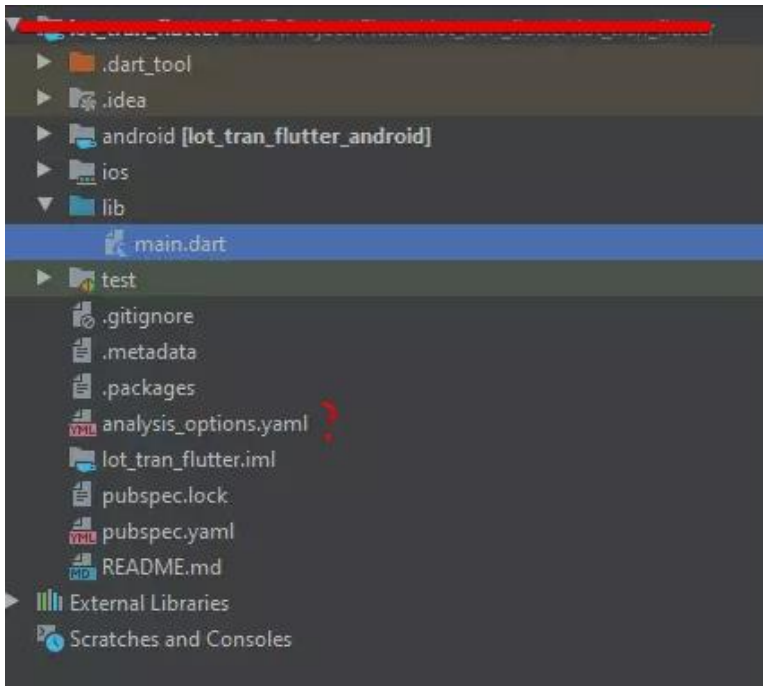
Hình 3.13: Ảnh minh họa tải 2 plugin dart vs flutter

Bắt đầu truy cập vào android studio:

Do android studio có thể chạy được trên cả hai nền tảng là Windows và macOs. Nên vì thế, sẽ cực kì thuận tiện trong việc xây dựng những ứng dụng trên các thiết bị android hay iOS. Ngôn ngữ sử dụng để viết ở đây là ngôn ngữ Dart, Dart là ngôn ngữ lập trình đa mục đích ban đầu được phát triển bởi Google và sau đó được Ecma (ECMA-408) phê chuẩn làm tiêu chuẩn. Nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động. Dart là một

ngôn ngữ hướng đối tượng, được xác định theo lớp, với cơ chế garbage-collected, sử dụng cú pháp kiểu C để dịch mã tùy ý sang JavaScript.

Ngay sau khi tạo project đầu tiên, các bạn sẽ thấy cấu trúc của một project Flutter như sau:



Hình 3.14: Cấu trúc của một project flutter

Bên trên là sườn của một project flutter được tạo:

- **Folder lib**: Folder quan trọng nhất là folder lib, trong đó Flutter có tạo sẵn cho chúng ta 1 file source code là main.dart. Đây cũng chính là nơi chúng ta tạo các file .dart và viết code.
- **Hai folder là android và ios:** Đây là thư mục source của từng platform android và ios. Đa số là bạn không cần dùng đến, tuy nhiên cũng sẽ có những tính năng mà flutter không support được mà buộc bạn phải vào đây viết native code, android thì sử dụng Java hoặc Kotlin còn ios sử dụng Objective-C hoặc Swift. Ơ thế là phải học code native nữa à. Tất nhiên nhưng đừng lo lắng quá vì hầu hết đã được Flutter support rồi, Flutter không support được thì cũng có cộng đồng ngoài kia viết cả đống thư viện support nên hầu như rất ít khi phải vào đây code.
- **Folder test:** là nơi viết Unit Test cho dự án
- **File pubspec.yaml** đây là nơi khai báo tên dự án, mô tả dự án, các thư viện sử dụng trong dự án, các asset như icon, ảnh hoặc font được sử dụng trong dự án

- **Hai file.gitignore và README.md** thì quá quen thuộc với những bạn sài git, github rồi. Nó ko liên quan gì đến source dự án nên ko cần quan tâm cũng được.
- **Hai file. metadata và .packages** là những file config. Bạn sẽ không cần đến chúng nhưng Flutter cần chúng.
- **File analysis_options.yaml** được mình đánh dấu hỏi đó thì có thể lúc bạn tạo dự án sẽ không thấy đâu. Đây là file ko bắt buộc phải có trong dự án nhưng đây là file giúp cái code của mình được tốt hơn, tránh code thối bằng các rule do chính ta tự định nghĩa trong file này.
- Ngoài ra chúng ta cũng có thể thêm một số folder vào trong bằng cách là click chuột phải > chọn Directory. Vì Flutter khá chú trọng vào giao diện người dùng thường, người lập trình sẽ tạo thêm một file để lưu trữ ảnh và icons, để khi gọi ra tiện lợi hơn, và nhanh chóng hơn.

Cách tạo một chương trình đơn giản trong main.dart:

```
// Bước 1: import thư viện material vào, thư viện này sẽ cung cấp widget để code
import 'package:flutter/material.dart';

// Bước 2: khai báo hàm main, đây là nơi mà code sẽ thực thi đầu tiên
void main() {
  // Bước 3: gọi hàm runApp truyền vào 1 object MaterialApp
  // MaterialApp chính là widget root, tổ tiên của 1 cây widget sau này
  runApp(MaterialApp(
//Bước 4:Trong constructor của MaterialAppb có propertylà `home`
 // ta sử dụng property `home` này để code nội dung trong app
      // ở đây mình truyền vào widget Text truyền vào 1 String
      home: SafeArea( child: Scaffold(appBar:

 AppBar(

  backgroundColor: Colors.pink, // set màu background cho app bar
     title: Text('Flutter Demo Home Page') // title của app bar),
          body: Center(
              child: Text('Hi, chào mừng đến với flutter minh họa!')   )
      ),
    ),
 ));
}
```

Lần đầu chạy ứng dụng trên thiết bị ảo, sẽ tốn 1 khoảng thời gian lớn, do flutter cần khởi tạo đầy đủ các chức năng và các gói thư viện.

Kết quả :

Hình 3.15: Ứng dụng minh họa về Flutter

3.2 Xây dựng ứng dụng thực nghiệm

Ứng dụng thực nghiệm Webview

Xây dựng ứng dụng dùng webview để đưa 1 website về ứng dụng

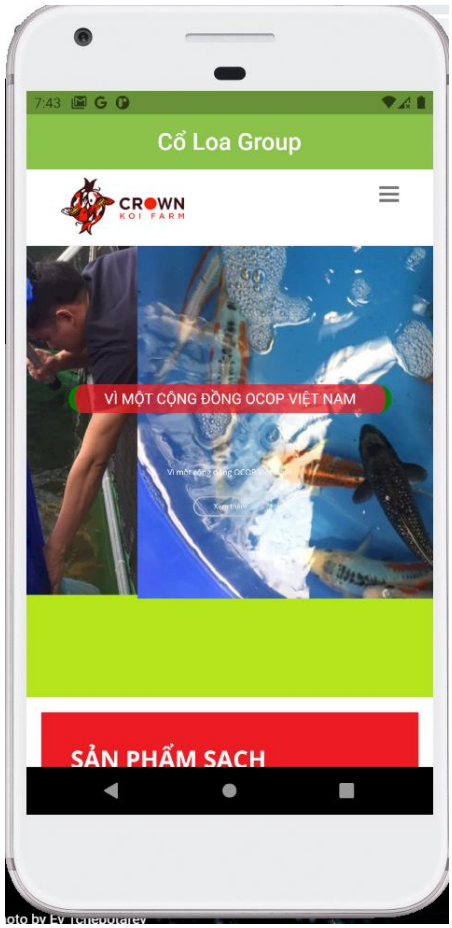
Webview là một ứng dụng hệ thống trên các hệ điều hành đã cài về điện thoại, cho phép điện thoại của bạn có thể hiển thị nội dung trang web ngay trên ứng dụng, không cần phải mở trình duyệt để xem các nội dung này.

Như đã giới thiệu ở phần 2.4, mục đích để tạo ra webview này để giúp người dùng có thể tiện lợi hơn. Mỗi lần bạn cần tìm hiểu về một vấn đề gì đó, bạn cần phải truy cập lên trên các trang mạng để tìm kiếm vấn đề đó, mỗi lần bạn như vậy sẽ rất mất thời gian để tìm kiếm hoặc truy cập mạng.

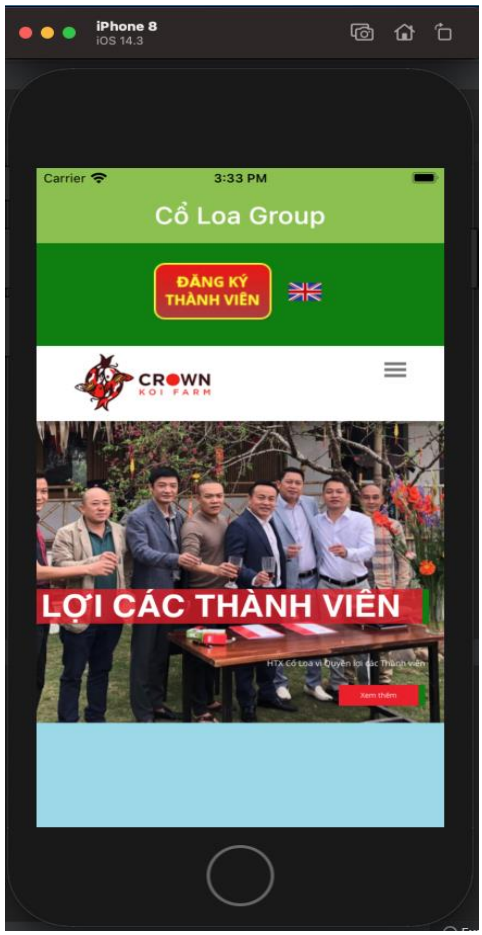
Vì thế, em đã xây dựng ra một ứng dụng Webview thay vì lên mạng tìm kiếm, thì mình chỉ cần tải ứng dụng này về máy, mỗi lần cần thì truy cập bằng app,

rất tiện lợi và tiết kiệm thời gian cho người dùng. Do trên thế giới hiện nay các thiết bị di động đã trở lên phổ biến, chủ yếu là phân làm 2 cực. Một bên là Iphone do Apple phát triển, bên còn lại là các thiết bị có sử dụng hệ điều hành Android(có thể kể tên là samsung, xiaomi, oppo,....). Nên ứng dụng này em sẽ là xây dựng để có thể chạy được trên cả 2 thiết bị trên.

Hình ảnh minh họa :



Hình 3.16: Ứng dụng webview chạy trên thiết bị android



Hình 3.17: Ứng dụng webview chạy trên thiết bị iOS

Những chức năng để xây dựng lên hệ thống:

-Webview

- Checking internet

- Splash screen

3.2.1 Webview

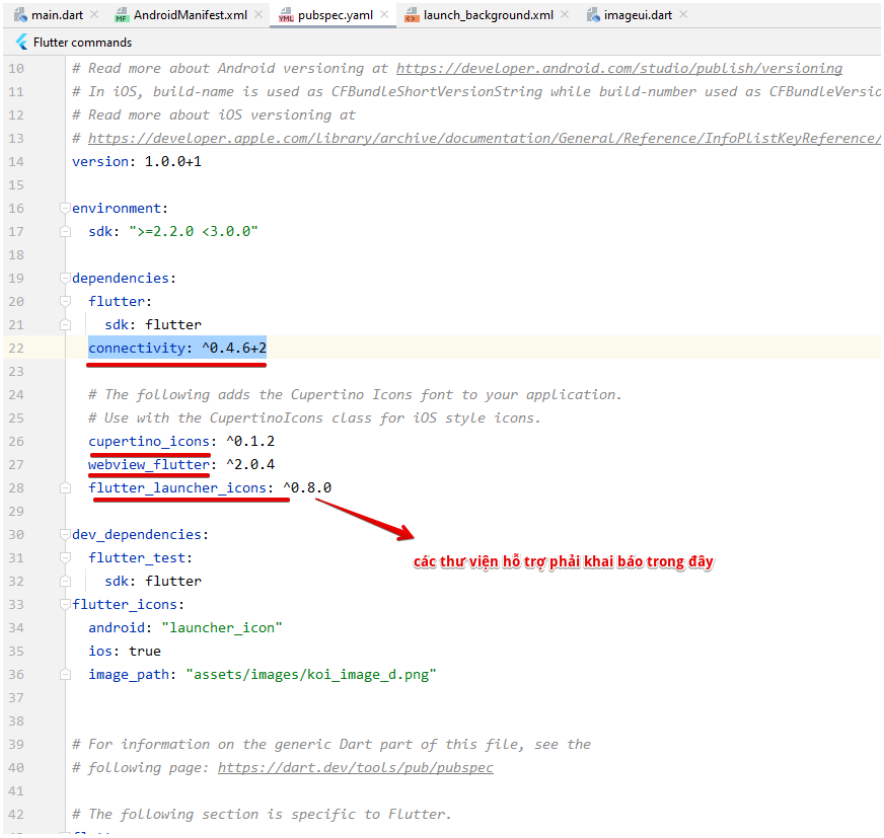
3.2.1.1 Khái niệm

Webview flutter là gì : là 1 plugin của flutter cho phép người dùng gọi url của 1 website trên mạng về app của mình mà không cần phải truy cập vào trang web đó[6], rất tiện lợi cho người sử dụng.

3.2.1.2 Cách dùng

Sử dụng gói này như là một thư viên bằng cách thêm dòng lệnh

webview_flutter: + (phiên bản) vào thư mục "pubspec.yaml ", thường thì phiên bản sẽ chọn phiên bản mới nhất để thư viện này đầy đủ. Sau đó thì import thư viện vào trong code Dart :" importpackage: webview_flutter / webview_flutter.dart';"



Hình 3.18: Thư mục pubspec.yaml nơi thêm cách gói(package)

Thêm plugin vào thư mục pubspec.yaml. và Pub Get để lấy dữ liệu của thư viện về

Bước tiếp theo : import thư viện vào codeDart là hoàn thành quá trình import thư viện của webview_flutter

Để chạy 1 webview thì chúng ta thực hiện như sau :

Step 1: Tạo HomeScreen

Tạo một flutter project bằng command line nhưng thông thường : **flutter create flutter_demo_webviews**

Mở project vừa tạo tìm đến file main.dart và sửa đổi như sau

```
// main.dart
```

```
import 'package:flutter/material.dart';
import 'app.dart';
void main() => runApp(App());
```

Tạo file app.dart với các Material Design styling cơ bản. Đồng thời set home navigation route là home screen.

```
// app.dart
import 'package:flutter/material.dart'; // thư viện này sẽ cung cấp widget để code
import 'screens/home.dart';
class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Web Views',// tiêu đề sẽ chạy ra màn hình ứng dụng
      theme: ThemeData(
          primarySwatch: Colors.blue,
          fontFamily: "Arial",
          textTheme: TextTheme(
              button: TextStyle(color: Colors.white, fontSize: 18.0),
              title: TextStyle(color: Colors.red))),
      home: Home(),
    );
  }
```

Step 2: Tạo home.dart

Màn hình home của chúng ta sẽ chỉ show 1 button có nhiệm vụ mở một URL đã được chỉ định sẵn. Trong ví dụ này chúng ta sẽ dùng Material Design widgets button (FlatButton) để đơn giản hóa. Màn hình có thể chứa một vài widgets bạn không quen

thuộc, nhưng bạn chỉ cần hiểu hàm handleURLButtonPress bên dưới chịu nhiệm vụ navigate đến web view được triển khai trong widget WebViewContainer.

```
// screens/home.dart
import 'package:flutter/material.dart';
import 'web_view_container.dart';
class Home extends StatelessWidget {
  final _links = ['https://google.com'];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: SafeArea(
            child: SingleChildScrollView(
                child: Column(
      mainAxisAlignment: MainAxisAlignment.center,

      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: _links.map((link) => _urlButton(context, link)).toList(),
    ))));
  }
  Widget _urlButton(BuildContext context, String url) {
    return Container(
        padding: EdgeInsets.all(20.0),
        child: FlatButton(
          color: Theme.of(context).primaryColor,
          padding: const EdgeInsets.symmetric(horizontal: 50.0, vertical: 15.0),
          child: Text(url),
```

```
        onPressed: () => _handleURLButtonPress(context, url),
      ));
  }

  void _handleURLButtonPress(BuildContext context, String url) {

    Navigator.push(context,
      MaterialPageRoute(builder: (context) => WebViewContainer(url)));
  }
}
```

Step 3: Tạo WebViewContainer

WebView của chúng ta sẽ hiển thị full screen và được implement bởi một widget mới là WebViewContainer. Mỗi widget chính là một màn hình đơn giản Thành phần quan trọng nhất của màn hình này là Webview widget, chỉ việc import webview_flutterpackage

```
WebView(
   key: _key,
   javascriptMode: JavascriptMode.unrestricted,
   initialUrl: _url)
```

Đầu tiên, 'key' parameter cho phép Flutter widget tree dễ dàng refer đến widget này thông qua một unique key tạo qua Flutter's UniqueKey() method. Tiếp theo, javascriptMode chỉ đơn gianr là cho chép chúng ta kiểm soát lại những Javascript nào có thể chạy trong webview. Cuối cùng,initialUrl là URL web page chúng ta muốn hiển thị trong webview.

Triển khai WebViewContainer

```
import 'package:flutter/material.dart';
import 'package:webview_flutter/webview_flutter.dart';
class WebViewContainer extends StatefulWidget {
  final url;
  WebViewContainer(this.url);
  @override
  createState() => _WebViewContainerState(this.url);}
class _WebViewContainerState extends State<WebViewContainer> {
  var _url;
  final _key = UniqueKey();
  _WebViewContainerState(this._url);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        body: Column(
          children: [
            Expanded(
                child: WebView(
                    key: _key,
                    javascriptMode: JavascriptMode.unrestricted,
                    initialUrl: _url))
          ],   ));  }}
```

Điều quan trọng ở đây là bất kỳ Widget nào có chứa WebView widget đều phải sử dụng StatefulWidget. Nếu bạn sử dụng StatelessWidget, webview sẽ không load đúng cách. Ở thây chúng ta sẽ truyền tham số Url vào Widget. Tham số này sẽ được sử dụng trong state của StatefulWidget. Có rất nhiều tính năng của WebView

widget được cung cấp. Để xem chi tiết chỉ việc bấm Ctrl/Command + Click vào WebView widget để đọc source code.

3.2.2 Splash screen

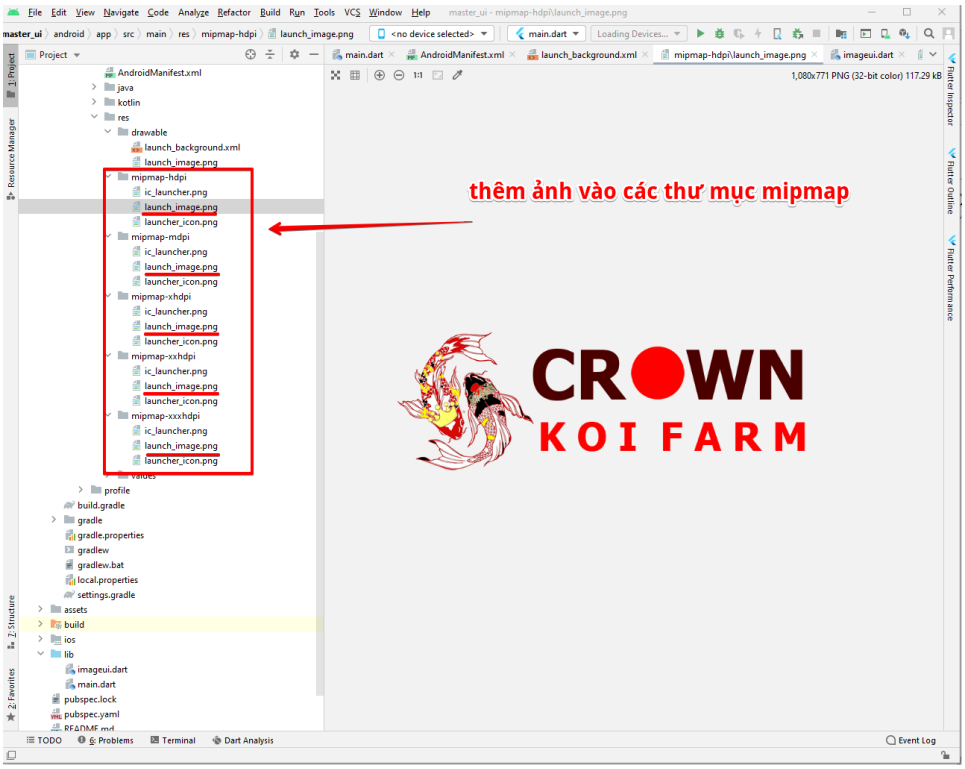
Splash screen là gì : là trải nghiệm và là thứ đầu tiên người dùng nhìn thấy đối với mỗi ứng dụng. Nó thường được sử dụng để hiển thị có thể là Progress hay là một ảnh nền, hay là một icon[7].

Để thêm Splash screen vào trong app

Bước 1 : Truy cập android/app/src/main/res/drawable/ lanch_background Sửa android src : thành địa chỉ của ảnh. Android garvity: để đặt ảnh khi chạy splash screen xuất hiện ở vị tri nào. Thường thì Android garivity=" center " để cho bức anh có thể xuất hiện ở giữa màn hình.

Bước 2: Truy cập android/app/src/main/res/

Thêm ảnh đã chuẩn bị là "launch_iamge.png" vào các thư mục 4 mipmap.



Hình 3.19: Thêm ảnh vào các thư mục mipmap

Kết quả khi bắt đầu chạy ứng dụng:

Hình 3.20: Splash Screen được chạy khi khởi động ứng dụng

3.2.3 Checking Internet

Kiểm tra kết nối mạng là bước rất quan trọng. Nó có thể in thông báo đến người dùng là mất mạng khi wi-fi hoặc dữ liệu di động của ngươi dùng có vấn đề không kết nối được mạng, tránh xảy ra thường hợp không mất mát dữ liệu.

Cách thức hoạt động :

❖ Lúc bắt đầu truy cập phần mềm:

Khi truy cập ứng dụng phần mềm sẽ tự động kiểm tra kết nối mạng từ wifi và dữ liệu di động, nếu không có kết nối mạng, phần mềm sẽ in ra 1 thông báo trên màn hình để người dùng có thể kiểm tra lại kết nối mạng của mình, khi kiểm tra và kết nối mạng thành công, phần mềm sẽ tự ngắt thông báo là chuyển đến nội dung của app,

❖ Khi đang sử dụng phần mềm :

Khi đang sử dụng, tương tự như lúc bắt đầu truy cập vào phần mềm. Kiểm tra Internet khi bắt đầu mở ứng dụng

```
void initState(){
  super.initState();
  try {
    InternetAddress.lookup('google.com').then((result){
```

```
      if(result.isNotEmpty && result[0].rawAddress.isNotEmpty){
        // internet conn available

Navigator.of(context).pushReplacement(MaterialPageRoute(builder:
(context) =>
          imageui(),
        ));
      }else{
        // no conn
        _showdialog();
      }
    }).catchError((error){
      // no conn
      _showdialog();
    }

);
  } on SocketException catch (_){
    // no internet
    _showdialog(); }

  Connectivity().onConnectivityChanged.listen((ConnectivityResult
connnresult){
    if(connnresult == ConnectivityResult.none){

    }else if(previous == ConnectivityResult.none){
      // internet conn

Navigator.of(context).pushReplacement(MaterialPageRoute(builder:
(context) =>
          imageui(),
        ));
    }

    previous = connnresult;
  });
```

InternetAddress để lấy địa chỉ kiểm tra, ở đây em chọn google.com để kiểm tra:

Nếu có truy cập được đến google.com(tức là có mạng) thì tiếp tục truy cập đến imageui(), ImageUI là thân của chương trình.

Ngược lại nếu không truy cập được thì sẽ show ra 1 dialog thông báo đến người dùng, không thể kết nối Internet

Ở đây show ra 1 dialog() như sau :

```
void _showdialog(){
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text('Lỗi Kết Nối'),
      content: Text("Không thể kết nối Internet."),
      actions: <Widget>[
        // ignore: deprecated_member_use
        FlatButton(
          // method to exit application programitacally
          onPressed: () =>
SystemChannels.platform.invokeMethod('Systemnavigator.pop'),
          child: Text(" Thoát "),),
       ],
      ),
    );
}
```

Dialog này bao gồm có title(tiêu đề), content(nội dung), action(hành động), title và content biểu thị thông tin là lỗi kết nối và không thể truy cập mạng, Action là 1 widget FlatButton(cho phép người dùng tạo 1 nút ấn phẳng), ấn ra thoát người dùng sẽ thoát khỏi ứng dụng. Ngoài ra, nếu người dùng chưa muốn thoát thì có thể ấn bào bất kì đâu trên màn hình ngoại trừ dialog để tắt thông báo, màn hình trờ lúc này sẽ quay lại Widget Build. Widget build này bao gồm padding:EdgeInset.only là 1 biểu tưởng quay tròn và 1 đoạn text kiểm tra kết nối

Cần thêm icon và text rằng đang kiểm tra kết nối mạng

```
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          CircularProgressIndicator(),
          Padding(
            padding: EdgeInsets.only(
              top: 20.0 ),
            child: Text(
              " Kiểm tra kết nối Mạng !"
            ),    ),   ],), ), );
```

Kiểm tra Internet khi đang chạy ứng dụng

```
Future<bool> checkinternet() async {
  try {
    final result = await InternetAddress.lookup('google.com')[7];
    if (result.isNotEmpty && result[0].rawAddress.isNotEmpty) {
      return Future.value(true);
    }
```

```
  } on SocketException catch (_) {
    return Future.value(false);
  }
}
@override
void initState() {
  super.initState();
  connectivitySubscription = Connectivity()
      .onConnectivityChanged
      .listen((ConnectivityResult connresult) {
    if (connresult == ConnectivityResult.none) {
      dialogshown = true;
      showDialog(context: context,
        barrierDismissible: false,
        builder: (content) => AlertDialog(
          title: Text("Lỗi",),
          content: Text(" Không có kết nối dữ liệu.",),
          actions: <Widget>[
            // ignore: deprecated_member_use
            FlatButton(
              onPressed: () => {

SystemChannels.platform.invokeMethod('SystemNavigator.pop'),
              },
              child: Text("Thoát."),  ),], ),);
    } else if (_previousResult == ConnectivityResult.none) {
      checkinternet().then((result) {
        if (result == true) {
          if (dialogshown == true) {
            dialogshown = false;
            Navigator.pop(context); }       }})
       ;}
    _previousResult = connresult;  });}
@override
void dispose() {
  super.dispose();

  connectivitySubscription.cancel();}
```

Ở đây sử dụng hàm Future<bool> kiểu true false để kiểm tra kết nối.

Dùng hàm kiểm tra true false để bắt luôn sự kiện nếu đúng thì chạy tiếp, sai thì phải dừng lại, tránh những sự cố như đang thực hiện 1 bước mua hàng mà mất mạng, khi có mạng thì bước đó chưa thực hiện, sẽ ảnh hưởng đến trải nghiệm của người dùng.

Khi kết nối mạng bị mất, phần mềm sẽ hiện lên thông báo: "không thể kết nối mạng", khi đó hàm kiểm tra kết nối mạng hoạt động, dùng hàm await là để ứng dụng thực hiện xong việc kiểm tra mạng, rồi sau đó mới làm khối lệnh tiếp theo.

Tương tự như phần kiểm tra mạng lúc bắt đầu khởi động ứng dụng, thân trương trình vẫn có initState, nhưng có thêm cả dispose(), dispose ở đây dùng để thoát khỏi ứng dụng, khi mất mạng giữa chừng sẽ ảnh hưởng đến người dùng, nếu không thể kết nối mạng lại, người dùng sẽ thoát khỏi ứng dụng bằng chức năng này.

Cách đổi tên ứng dụng :

- Truy cập tới android>app>src>Androidmanifest.xml

- Thay đổi tên ứng dụng android:label

```
android:label="Cổ Loa Group"
```

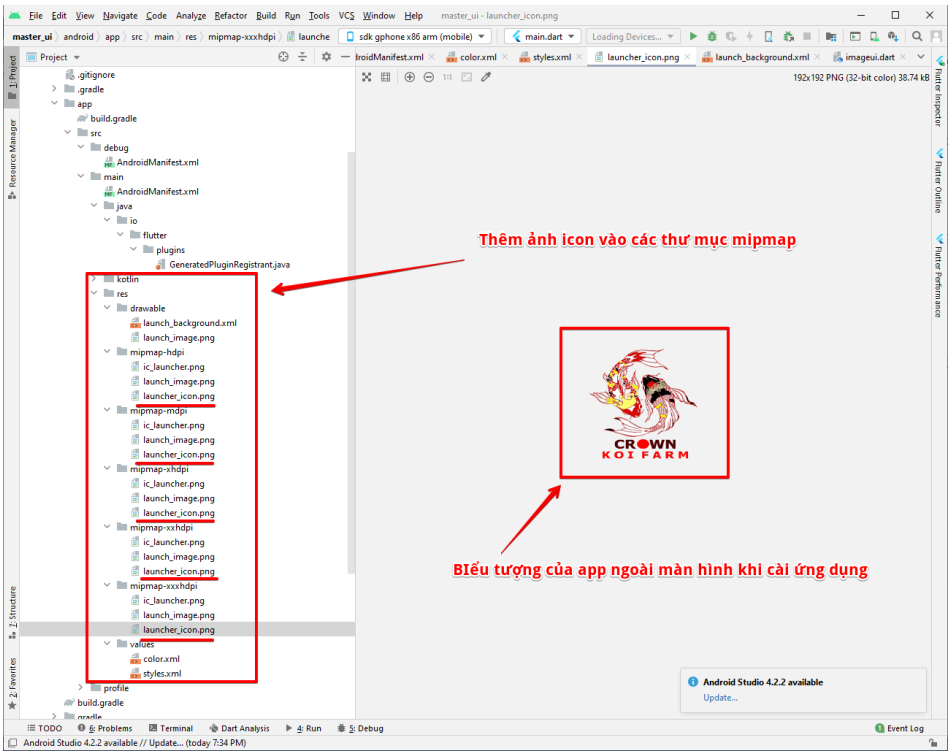
Hình 3.21: Tên của ứng dụng sau khi đã đổi

Cách đổi hình nền của ứng dụng thay vì là hình Flutter mặc định :

- Truy cập android/app/src/main/res/drawable/ như đặt biểu tượng cho splash screen

- Thêm ảnh lần lượt vào các thư mục mipmap.

- Trong android > app > src > Androidmanifest.xml thêm đường dẫn thư mục mipmap của icon vào android:icon= "`@mipmap/launcher_icon`", mipmap là thư mục chứa biểu tượng, còn launcher_icon là tên của biểu tượng



Hình 3.22: Thêm biểu tượng icon vào các thư mục mitmap

Chạy ứng dụng trong trình giả lập để xác nhận rằng biểu tượng trình khởi chạy đã được tạo thành công.

Hình 3.23: Biểu tượng của ứng dụng sau khi đã đổi

3.2.4 Cấu hình và một số lỗi

Trước khi build đối với cả iOS và Android hãy kiểm tra đầy đủ plugin, tránh tình trạng build ra ứng dụng mà plugin không có, không chạy được ứng dụng.

Đặc biệt chú ý, khi build ra phần mềm có sử dụng đến nết nối mạng (online), cần phải truy cập đến android/app/src/main/AndroidManifest.xml để kiểm tra đã có câu lệnh.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Câu lệnh này giúp ứng dụng cần truy cập mạng, khi build xong sẽ chạy và kết nối mạng bình thường, nếu không thêm ứng dụng hoàn toàn bị mất mạng[8].

Một số lỗi thường gặp :

- Webview_flutter:

```
H:\flutterapplication\flutter_webviewwhehe\android\app\src\debug\AndroidManifest.xml Error:
    uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:webview_flutter]
```

Use-sdk: minSdkVersion 16 không được nhỏ hơn phiên bản 19 được khai báo trong thư viện.

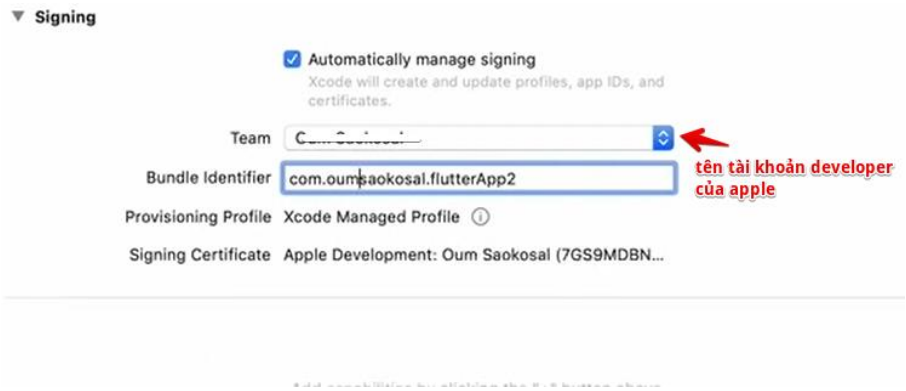
Cách sửa lỗi: truy cập vào build > webview_flutter > intermediates > library_manifest >debug > AndroidManifest.XMl thay đổi `<uses-sdk android:minSdkVersion="19" />` thành `<uses-sdk android:minSdkVersion="16" />` thì ứng dụng sẽ hoạt động bình thường.

Cấu hình, cách xây dựng ứng dụng trên Android:

- Hệ điều hành: android lolipop 5.0 trở lên
- Bộ nhớ: trống khoảng 50M
- Cách build ứng dụng :
    - Bước 1: truy cập Build >Flutter>chọn build apk
    - Sau đó truy cập đến vị trí của app-release.apk:
    - Địa chỉ của app-release.apk : build > app > outputs > release > app-release.apk
    - Tải ứng dụng về thiết bị thật và cài đặt ứng dụng như bình thường.
    - Run được trên các thiết bị của android.

Cấu hình, cách xây dựng ứng dụng trên iOS:

- Yêu cầu : phiên bản Xcode là phiên bản mới nhất để có thể hỗ trợ chạy trên các thiết bị Iphone từ Iphone 6 trở lên, có tài khoản developer của apple, và có kết nối với thiết bị iOS thật
- Cách build ứng dụng: ở trong Android Studio, truy cập ios > runner.xcodeproj> project. Để khởi động Xcode và tiến hành build ứng dụng trên Xcode.
- Đăng nhập tài khoản developer Apple



Hình 3.24: đăng nhập tài khoản developer của apple

Ấn build ứng dụng sẽ được build trên máy thật. Khi chạy thì codesign sẽ yêu cầu xác thực bằng cách yêu cầu nhập mật khẩu tài khoản User có quyền admin, nhập và ấn vào alway allows, vài lần để xác thực.

Lưu ý: khi cài cần truy cập vào device management để thiết lập sự tin tưởng để thiết bị Iphone có thể cài đặt ứng dụng bình thường.

Cấu hình: Khi build ứng dụng trên xcode phụ thuộc vào phiên bản Xcode, nếu sử dụng phiên bản thấp, khi build trên thiết bị thật sẽ xảy ra hiện tượng là không hỗ trợ (Unsuppoted OS version).

Hình 3.25: lỗi không nhận các thiết bị có OS version từ 14.5 trở lên

KẾT LUẬN

Trong thời gian làm đồ án, bằng kiến thức đã được học trong trường cùng sự giúp đỡ hướng dẫn tận tình của thầy cô, bạn bè. Đã giúp em vận dụng, hoàn thành đề tài đồ tốt nghiệp trong thời gian quy định. Qua quá trình hoàn thiện đồ án tốt nghiệp em đã học hỏi được rất nhiều kiến thức để có thể tìm hiểu được các kiến thức về kiến trúc, các tính năng, tổng quan của Flutter. Em cũng đã tìm hiểu được những thành phần cơ bản của một website và cách phân loại. Ngoài ra, em đã biên dịch được ứng dụng chạy được trên cả hai nền tảng là Android và iOS.

Thời gian tới, em dự định sẽ sử dụng những kiến thức trên để xây dựng một ứng dụng bán hàng trên di động chạy trên cả hai nền tảng Android và iOS, có sử dụng các tính năng đã được vận dụng để làm đồ án như là kiểm tra kết nối mạng, màn hình chờ, đặc biệt sử dụng chế độ xem web để có thể hiển thị được các trang web mà không cần phải mở trình duyệt để xem các nội dung này.

Trong phạm vi của đồ án này, em đã giải đáp được những nét khái quát nhất về Flutter, một lượng kiến thức cơ bản về một website, ứng dụng của Webview Flutter, và kiến thức chạy ứng dụng được trên cả hai nền tảng Android và iOS. Do kiến thức còn hạn hẹp, nên đồ án tốt nghiệp của em không thể tránh khỏi những thiếu sót, em mong sẽ nhận được những lời đóng góp của các thầy cô trong khoa để đồ án của em trở lên hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày.....tháng.....năm 2021

Sinh viên thực hiện

Đỗ Thế Hiệp

TÀI LIỆU THAM KHẢO

[1] Sách Learn Google Flutter Fast: 65 Example Apps tác giả Mark Clow

[2] https://bizfly.vn/techblog/website

[3] https://flutter.dev/docs/development/tools/sdk/releases?tab=macos

[4] https://flutter.dev/docs/get-started/install/windows

[5] https://brew.sh/

[6] https://stackoverflow.com/questions/ 8614553 /can- someone- give-o ne-exact-example-of-webview-implementation-in-android.

[7] https://medium. com/ viet-flutter-developers /flutter-t%E1% BA%A1o-splash-screen-cho-android-ios-b8b21068032a.

[8] https://stackoverflow .com/questions /55603979/why-cant-a- flutter-application-connect-to-the-internet-when-installing-app-release.