

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM



BÁO CÁO ĐỒ ÁN 1

Đề tài:

TÌM HIỂU FLUTTER VÀ XÂY DỰNG ỨNG DỤNG

Giảng viên hướng dẫn: ThS. Nguyễn Công Hoan
Lớp: SE121.N11

Sinh viên thực hiện:
Trần Ngọc Tiến - 20520808

TP.HCM, tháng 12 năm 2022

Lời cảm ơn

Sau quá trình học tập và rèn luyện tại khoa Công nghệ phần mềm trường Đại học Công nghệ Thông tin – ĐHQG TP.HCM, em đã được trang bị các kiến thức cơ bản cùng các kỹ năng thực tế để có thể hoàn thành Đồ án 1 của mình.

Để hoàn thành đồ án này, với lòng biết ơn sâu sắc em xin gửi lời cảm ơn chân thành đến:

Ban Giám hiệu trường Đại học Công nghệ Thông tin – ĐHQG TP.HCM vì đã tạo điều kiện thuận lợi để sinh viên tìm kiếm, nghiên cứu thông tin với hệ thống thư viện hiện đại, đa dạng các loại sách và tài liệu.

Gần gửi hơn là những lời tốt đẹp nhất xin gửi đến thầy Nguyễn Công Hoan đã tận tình giúp đỡ, định hướng cách tư duy và hướng làm việc khoa học. Đó là những góp ý hết sức quý báu không chỉ trong quá trình thực hiện đồ án mà còn là hành trang tiếp bước cho em trong quá trình học tập và làm việc sau này.

Sau cùng, xin chúc quý Thầy Cô trong khoa Công nghệ Phần mềm nói riêng cũng như các giáo viên tại trường Công nghệ thông tin nói chung thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình.

Thành phố Hồ Chí Minh, 25 tháng 12 năm 2021

Sinh Viên

Trần Ngọc Tiến

Nhận xét của giảng viên

MỤC LỤC

Lời cảm ơn.....	2
Nhận xét của giảng viên	3
MỤC LỤC	4
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN	8
1. Thông tin sinh viên:	8
2. Tổng quan đề tài:	8
2.1 Giới thiệu đề tài:	8
2.2 Phạm vi nghiên cứu:.....	8
2.3 Nội dung nghiên cứu:	8
2.4 Kết quả hướng tới:	8
3. Công cụ sử dụng:	9
CHƯƠNG 2: FLUTTER	10
1. Flutter là gì?	10
2. Điều gì làm Flutter trở nên độc đáo	11
4. Kiến trúc của Flutter	13
4.1. Kiến trúc Flutter.....	13
4.2. Flutter Engine	13
4.3. Thư viện nền tảng (Foudation Library)	13
4.4. Widget.....	13
4.5. Thiết kế các Widget cụ thể.....	14
5. Giới thiệu về ngôn ngữ lập trình Dart	14
5.1. Các từ khóa	14
5.2. Kiểu dữ liệu	15
5.3. Các biến và hàm.....	16
5.4. Toán tử (Operators)	17
5.5. Câu lệnh điều kiện	19
5.6. Vòng lặp.....	21
5.7. Chú thích (Comment).....	22

5.8. Continue và Break.....	22
5.9. Từ khóa Final và Const.....	23
5.10. Lập trình hướng đối tượng (OOP)	24
6. Một số loại Widget thường gặp trong Flutter.....	31
6.1. Widget Flutter.....	31
6.2. Widget hiển thị.....	33
6.3. Widget ẩn.....	40
7. Tìm hiểu bố cục (layout) Trong giao diện Flutter	46
7.1. Bố cục.....	46
7.2. Bố cục một Widget.....	47
7.3. Các loại Widget bố cục.....	48
8. Tìm hiểu về Cử chỉ (Gestures) với giao diện Flutter	52
8.1. Con trỏ.....	52
8.2. Cử chỉ.....	53
8.3. Dò cử chỉ	54
9. Quản lý State	54
9.1. Trạng thái tức thời	55
9.2. Trạng thái ứng dụng	56
10. Quản lý thư viện và các gói trong Flutter (Lib & Package)	57
10.1. Nạp thư viện trong Dart	57
10.2. Các thư viện cung cấp sẵn	58
10.3. Tạo thư viện trong Dart.....	59
10.4. Cài đặt các gói.....	60
10.5. Các gói hay sử dụng	62
11. Tìm hiểu về Navigator và Routing trong Flutter.....	68
11.1. Tạo routes.....	68
11.2. Điều hướng sang route thứ hai bằng phương thức Navigator.push()	71
11.3. Quay lại route đầu tiên bằng phương thức Navigator.pop()	71
11.4. Điều hướng với các route được đặt tên	72
12. Tìm hiểu về Database trong Flutter	75

12.1. Cơ sở dữ liệu SQLite	76
12.2. Firebase – NoSQL lưu trữ online.....	81
12.2.1. Sơ lược về Firebase	81
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG	85
1. Tổng quan	85
1.1. Tên ứng dụng: Phần mềm quản lý chi tiêu cá nhân trên di động	85
1.2. Lý do chọn ứng dụng.....	85
1.3. Đối tượng hướng đến.....	85
1.4. Môi trường phát triển ứng dụng.....	85
1.5. Kết quả mong đợi	86
1.6. Khảo sát hiện trạng	86
1.7. Quy trình thực hiện các công việc chính	87
2. Phân tích, thiết kế hệ thống	88
2.1. Xác định và mô hình hóa các yêu cầu phần mềm	88
2.2. Thiết kế hệ thống	104
2.3. Thiết kế dữ liệu	105
2.4. Thiết kế giao diện.....	114
3. Cài đặt và thử nghiệm	138
4. Hướng dẫn cài đặt phần mềm	139
5. Hướng dẫn sử dụng phần mềm	141
5.1. Bắt đầu với hệ thống.....	141
5.2. Đăng nhập, đăng ký và đăng xuất khỏi hệ thống	141
5.3. Quên mật khẩu, Đổi mật khẩu	142
5.4. Chuyển tiếp giữa các thành phần bên trong ứng dụng	142
5.5. Thêm chi tiêu.....	143
5.6. Sửa chi tiêu	143
5.7. Thay đổi thông tin cá nhân	143
5.8. Thay đổi ngôn ngữ ứng dụng	143
5.9. Tìm kiếm chi tiêu	143
5.10. Xuất CSV	143

TÀI LIỆU THAM KHẢO	144
---------------------------------	------------

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1. Thông tin sinh viên:

MSSV	Họ tên	Email
20520808	Trần Ngọc Tiến	20520808@gm.uit.edu.vn

2. Tổng quan đề tài:

2.1 Giới thiệu đề tài:

Với sự phát triển nhanh của smartphone như hiện nay với hai hệ điều hành phổ biến nhất là Android và IOS thì nhu cầu phát triển ứng dụng trên smartphone là rất lớn và việc phát triển riêng lẻ ứng dụng trên hai hệ điều hành trên bằng hai ngôn ngữ, hai framework khác nhau sẽ làm tốn rất nhiều chi phí và thời gian. Do đó việc có một framework cross-platform có thể code một lần và chạy trên cả hai hệ điều hành sẽ giúp tiết kiệm thời gian và tiền bạc hơn.

Hiện nay có rất nhiều framework cross-platform hỗ trợ phát triển ứng dụng trên cả hai hệ điều hành một cách nhanh chóng, tiện lợi nhưng vẫn đảm bảo tính bảo mật thông tin khách hàng. Và nổi bật trong những năm gần đây đó là Flutter.

2.2 Phạm vi nghiên cứu:

Trong đồ án lần này, em chủ yếu tập trung vào nghiên cứu tổng quan cách thức hoạt động và cách sử dụng các thư viện - package cũng như cách lưu trữ dữ liệu của Flutter. Đồng thời, em áp dụng các nghiên cứu trên vào một ứng dụng thực tế để giúp em có thể hiểu sâu hơn các kiến thức đã tìm hiểu.

2.3 Nội dung nghiên cứu:

Em sẽ tiến hành nghiên cứu chi tiết về cách thức hoạt động, ưu - khuyết điểm cũng như các thư viện - tính năng - package liên quan đến Flutter.

2.4 Kết quả hướng tới:

Với đề tài này, em đề ra hai mục tiêu chính:

- Đối với các nhân: mở rộng kiến thức của mình về Flutter thông qua quá trình tìm hiểu và áp dụng vào ứng dụng thực tế. Các kiến thức tìm hiểu được thông qua đồ án thúc đẩy em phát triển thêm các ứng dụng khác bằng Flutter. Đồng thời, em học được cách nghiên cứu và sử

dụng một framework mới cần trải qua quá trình gì nhằm giúp em dễ dàng tiếp xúc với các công nghệ mới hơn trong tương lai.

- Đối với các lập trình viên dùng đồ án nghiên cứu này làm tài liệu tham khảo: thông qua tài liệu nghiên cứu và ứng dụng em đã xây dựng, các lập trình viên khác có thể dễ dàng định hướng cần phải tìm hiểu gì khi sử dụng Flutter. Đồng thời em có ghi một số khái niệm, kiến thức cơ bản và thư viện phổ biến được đề xuất bởi cộng đồng khi sử dụng Flutter, các lập trình viên khác có thể tham khảo và tìm hiểu sâu hơn.

3. Công cụ sử dụng:

Trong quá trình xây dựng phần mềm, em đã sử dụng các phần mềm sau:

- **Android Studio:** Code giao diện ứng dụng + khởi tạo máy ảo
- **Visual Studio Code:** Code giao diện ứng dụng
- **Chrome:** Tìm hiểu thông tin

CHƯƠNG 2: FLUTTER

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.

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...

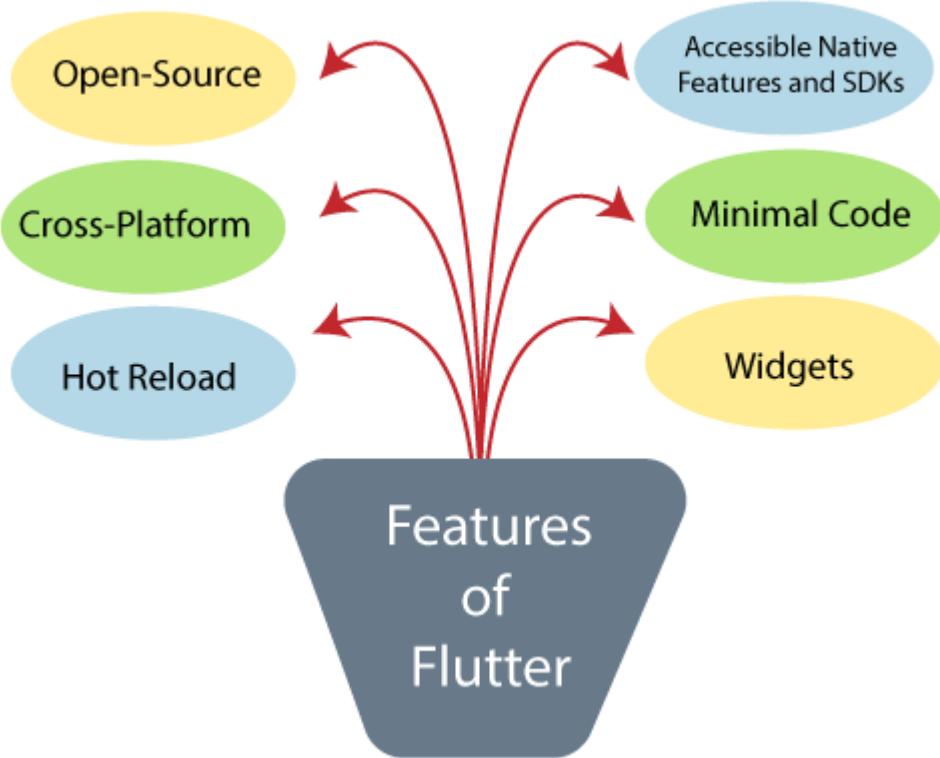
2. Điều gì làm Flutter trở nên độc đáo

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.

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 trực trắc trên tất cả các nền tảng.

4. Kiến trúc của Flutter

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)

4.2. Flutter Engine

Nó là một công 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.

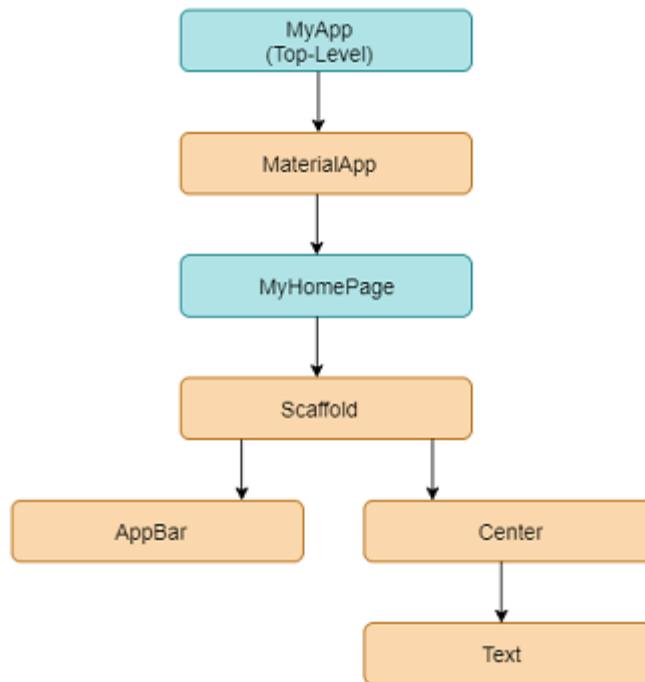
4.3. Thư viện nền tảng (Foudation Library)

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.

4.4. 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.

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.

5. Giới thiệu về ngôn ngữ lập trình Dart

5.1. Các từ khóa

abstract	else	import	show
as	enum	in	static
assert	export	interface	super

async	extends	is	switch
await	extension	late	sync
break	external	library	this
case	factory	mixin	throw
catch	false	new	true
class	final	null	try
const	finally	on	typedef
continue	for	operator	var
covariant	Function	part	void
default	get	required	while
deferred	hide	rethrow	with
do	if	return	yield
dynamic	implements	set	

5.2. 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	Ví dụ	Mô tả
String	String myName = 'ngoctienTNT';	Được dùng để biểu diễn chuỗi ký tự. Nó được nhập vào trong cặp nháy đơn '' hoặc nháy kép ". Dùng ký hiệu \, \' để biểu diễn ký tự nháy đơn (') trong. Khi bạn quyết định dấu ngoặc kép, bạn phải nhất quán với lựa chọn của mình. Nếu bạn muốn chuỗi của mình bao gồm nhiều dòng thì sử dụng "" để chứa chuỗi bên trong.

int	int age = 20;	Biểu diễn dạng số nguyên với từ khóa int. Kiểu int được biểu diễn tối đa 64-bit phụ thuộc vào phần cứng.
double	double height = 1.8;	Biểu diễn dạng số thực dấu phẩy động với từ khóa double. double đều được biểu diễn bởi 64bit.
bool	bool check = true;	Sử dụng từ khóa bool để biểu thị giá trị Boolean true và false.
List	List<int> num = [1, 2, 3, 4];	List<T> biểu diễn một nhóm các đối tượng được sắp xếp và có thiết kế giống như mảng (array) trong các ngôn ngữ khác.
Map	Map<String, dynamic> person = { "name": "Tiến", "age": 18, "height": 1.8};	Map<T, T> kiểu dữ liệu biểu diễn một tập đối tượng không được sắp xếp ở dạng <key, value>.
dynamic	dynamic name = "ngoctien"; dynamic age = 20;	dynamic trong Dart có thể hiểu là kiểu dữ liệu tùy chọn. Đây là kiểu cơ bản cho mọi kiểu dữ liệu trong Dart. Kiểu dữ liệu của một biến nếu không được chỉ định rõ ràng thì sẽ được gán với từ khóa dynamic.
class	Person person = Person();	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.

Bảng 1.1: Các kiểu dữ liệu trong ngôn ngữ Dart

5.3. Các biến và 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.

5.4. 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

5.4.1. 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 giá trị 20 thì:

Toán tử	Miêu tả	Ví dụ
+	Cộng hai toán hạng	A + B, kết quả là 30
-	Trừ toán hạng đầu cho toán hạng thứ hai	A - B, kết quả là -10
*	Nhân hai toán hạng	A * B, kết quả là 200
/	Chia toán hạng đầu cho toán hạng thứ hai	A / B, kết quả là 0.5
~/	Phép chia lấy phần nguyên	B ~/ A, kết quả là 2
%	Phép lấy số dư	B % A, kết quả là 0

Bảng 1.2: Các toán tử toán học trong Dart

5.4.2. 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.

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

Toán tử	Miêu tả	Ví dụ
<code>++</code>	Toán tử tăng <code>(++)</code> , tăng giá trị toán hạng thêm một đơn vị	<code>A++,</code> kết quả là 11 <code>++A,</code> kết quả là 11
<code>--</code>	Toán tử giảm <code>(--)</code> , giảm giá trị toán hạng đi một đơn vị	<code>B--,</code> kết quả là 19 <code>--B,</code> kết quả là 19

Bảng 1.3: Toán tử tăng và toán tử giảm trong Dart

5.4.3. 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
<code>&&</code>	Được gọi là toán tử logic AND (và). Kết quả là true nếu cả hai toán tử là true.	<code>(true && true) là true</code> <code>(true && false) là false</code>
<code> </code>	Được gọi là toán tử logic OR (hoặc). Kết quả là true nếu một trong hai toán tử là true.	<code>(true false) là true</code> <code>(false false) là false</code>
<code>!</code>	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.	<code>!(true) là false</code> <code>!(false) là true</code>

Bảng 1.4: Toán tử logic trong Dart

5.4.4. 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).

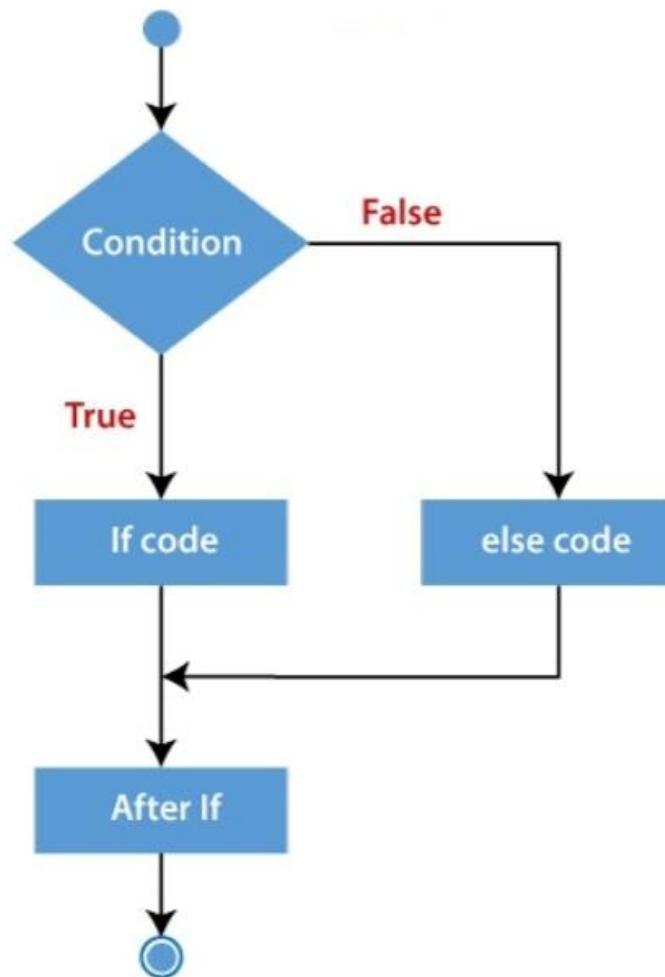
Toán tử	Miêu tả	Ví dụ
$==$	So sánh hai toán hạng nếu bằng nhau kết quả sẽ là true và ngược lại sẽ là false	(5 == 5) kết quả là true (5 == 6) kết quả là false
$!=$	So sánh hai toán hạng nếu bằng nhau kết quả sẽ là false và ngược lại sẽ là true	(5 != 5) kết quả là false (5 != 6) kết quả là true
$>$	So sánh hai toán hạng nếu toán hạng thứ nhất lớn hơn toán hạng thứ hai thì sẽ là true và ngược lại sẽ là false	(5 > 6) kết quả là false (6 > 5) kết quả là true
$<$	So sánh hai toán hạng nếu toán hạng thứ nhất bé hơn toán hạng thứ hai thì sẽ là true và ngược lại sẽ là false	(5 < 6) kết quả là true (6 < 5) kết quả là false
\geq	So sánh hai toán hạng nếu toán hạng thứ nhất lớn hơn hoặc bằng toán hạng thứ hai thì sẽ là true và ngược lại sẽ là false	(5 \geq 6) kết quả là false (6 \geq 5) kết quả là true (5 \geq 5) kết quả là true
\leq	So sánh hai toán hạng nếu toán hạng thứ nhất bé hơn hoặc bằng toán hạng thứ hai thì sẽ là true và ngược lại sẽ là false	(5 \leq 6) kết quả là true (6 \leq 5) kết quả là false (5 \leq 5) kết quả là true

Bảng 1.5: Toán tử so sánh trong Dart

5.5. Câu lệnh điều kiện

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.

```
1 void main() {  
2     var num = 10;  
3     if (num % 2 == 0) {  
4         print("$num là số chẵn");  
5     } else {  
6         print("$num là số lẻ");  
7     }  
8 }
```



Hình 1.3: Sơ đồ câu lệnh điều kiện

5.6. Vòng lặp

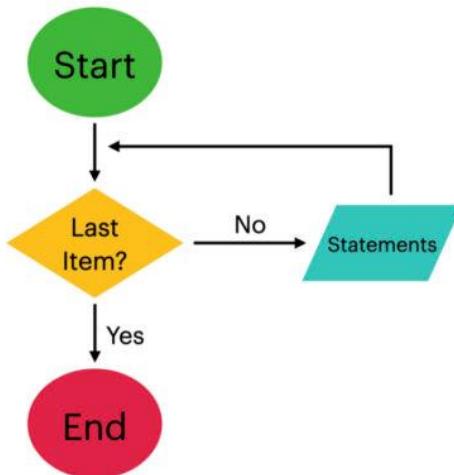
Các vòng lặp được sử dụng để thực thi một khối code lặp đi lặp lại cho đến khi một điều kiện được chỉ định trở thành đúng. Ngôn ngữ Dart hỗ trợ các loại câu lệnh lặp for, for..in, while, do..while.



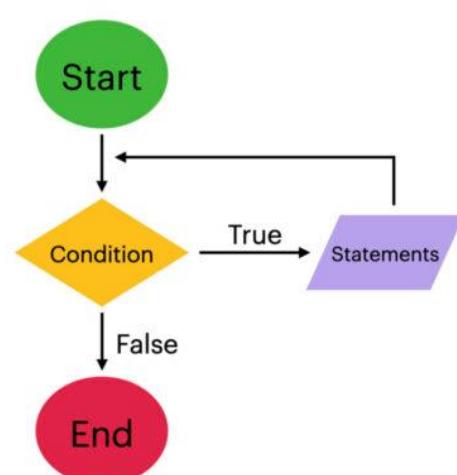
```

1 void main() {
2     var list = ["dog", "cat", "fish", "duck"];
3
4     for (var element in list) {
5         print(element);
6     }
7 }
```

For Loop



While Loop



Hình 1.4: Sơ đồ vòng lặp

5.7. Chú thích (Comment)

Việc sử dụng chú thích 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 chú thích định dạng:** Đó là một nhận xét dòng đơn (//)
- **Khối chú thích:** Đó là một nhận xét nhiều dòng /*...*/
- **Chú thích Tài liệu:** Đây là nhận xét tài liệu được sử dụng để mô tả cho thành viên và các kiểu (///)



```
1 void main() {  
2     //Một mảng chứa tên các loài động vật  
3     var list = ["dog", "cat", "fish", "duck"];  
4  
5     /*  
6         Chạy vòng lặp duyệt tất cả các phần tử có trong mảng list  
7         Sau đó in ra từng phần tử trong list  
8     */  
9     for (var element in list) {  
10         print(element);  
11     }  
12 }
```

5.8. 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.

```
1 void main() {  
2     for (int i = 1; i <= 10; i++) {  
3         if (i == 5) {  
4             print("Hello");  
5             continue; //it will skip the rest statement  
6         }  
7         print(i);  
8     }  
9 }  
10
```

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.

```
1 void main() {  
2     for (int i = 1; i <= 10; i++) {  
3         if (i == 5) {  
4             print("Hello");  
5             break; //it will terminate the rest statement  
6         }  
7         print(i);  
8     }  
9 }
```

5.9. 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.



5.10. Lập trình hướng đối tượng (OOP)

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

5.10.1. 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.

5.10.2. 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 mô 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.
- Chỉ cho phép tạo một constructor.

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.



```
1  class Person {  
2      String name; // public  
3      int _tuoi; // private  
4      Person(this.name, this._tuoi); // hàm khởi tạo  
5      int get tuoi => _tuoi; // hàm getter  
6  
7      //private  
8      _doiTen(String name) {  
9          this.name = name;  
10     }  
11  
12     // public  
13     int layTuoi() {  
14         return this._tuoi;  
15     }  
16 }
```

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:

```
1 class Mobile {  
2     // Property Declaration  
3     String color, brandName, modelName;  
4  
5     Mobile(this.color, this.brandName, this.modelName);  
6  
7     // Method Creation  
8     String calling() {  
9         return "Mobile can do call to everyone.";  
10    }  
11  
12    String musicPlay() {  
13        return "Mobile can play all types of Music.";  
14    }  
15  
16    String clickPicture() {  
17        return "Mobile can take pictures.";  
18    }  
19}  
20  
21 void main() {  
22     // Object Creation  
23     var myMob = new Mobile("black", "SamSung", "S22 Ultra");  
24  
25     // Accessing Class's Property  
26     myMob.color = "Black";  
27     myMob.brandName = "Apple Inc.";  
28     myMob.modelName = "iPhone 11 Pro";  
29  
30     //Display Output  
31     print(myMob.color);  
32     print(myMob.modelName);  
33     print(myMob.brandName);  
34     print(myMob.calling());  
35     print(myMob.musicPlay());  
36     print(myMob.clickPicture());  
37 }
```

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.

5.10.3. Các đặc tính trong lập trình hướng đối tượng 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...

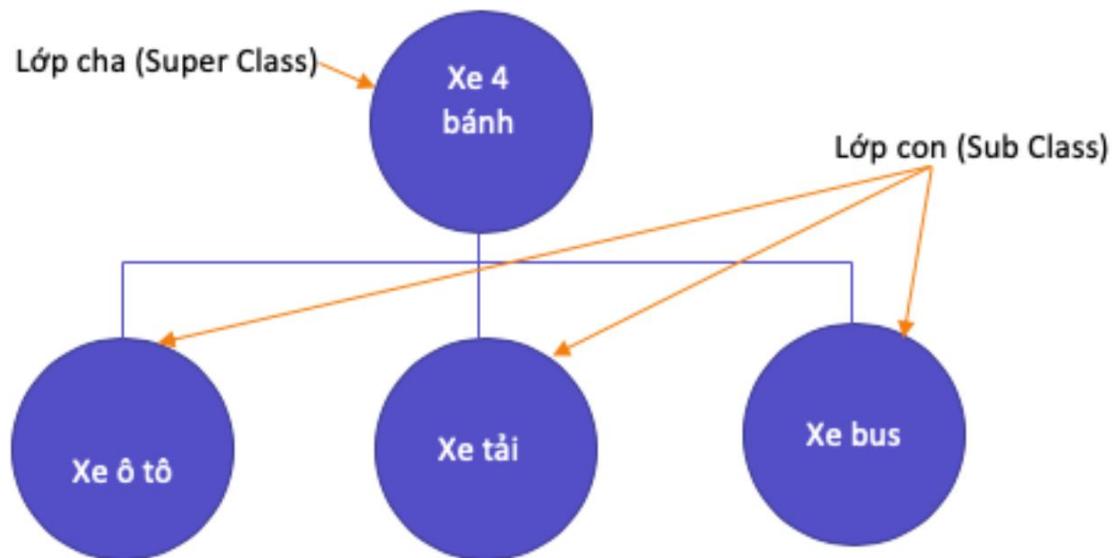
Tính đóng gói (Encapsulation)

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:

```
● ● ●  
1 int _tuoi;  
2  
3 // getter  
4 int get tuoi => _tuoi;  
5  
6 // setter  
7 void set datTuoi(int tuoi)  
8 {  
9     this._tuoi = tuoi;  
10 }
```

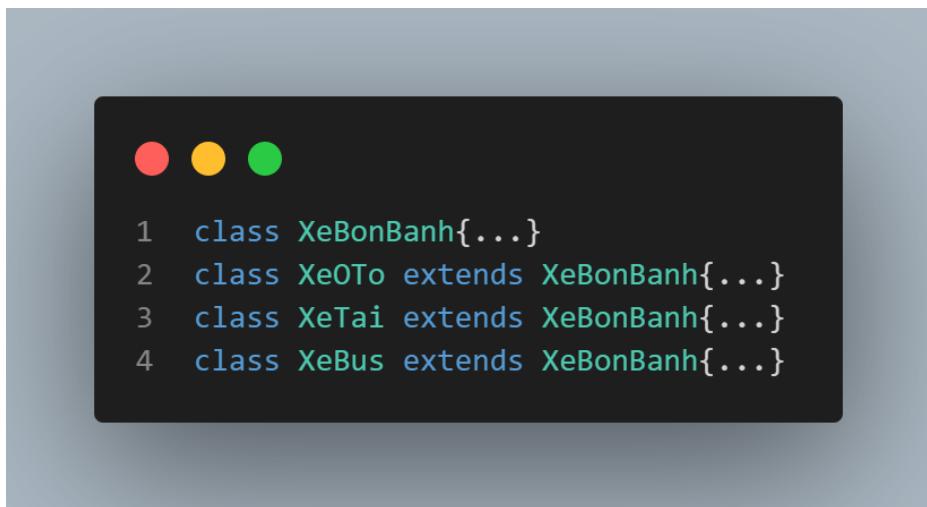
Tính kế thừa (Inheritance)

Trong một phần mềm hay chương trình, được cấu tạo bởi nhiều lớp khác nhau cùng các thành phần khác. Mỗi 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.

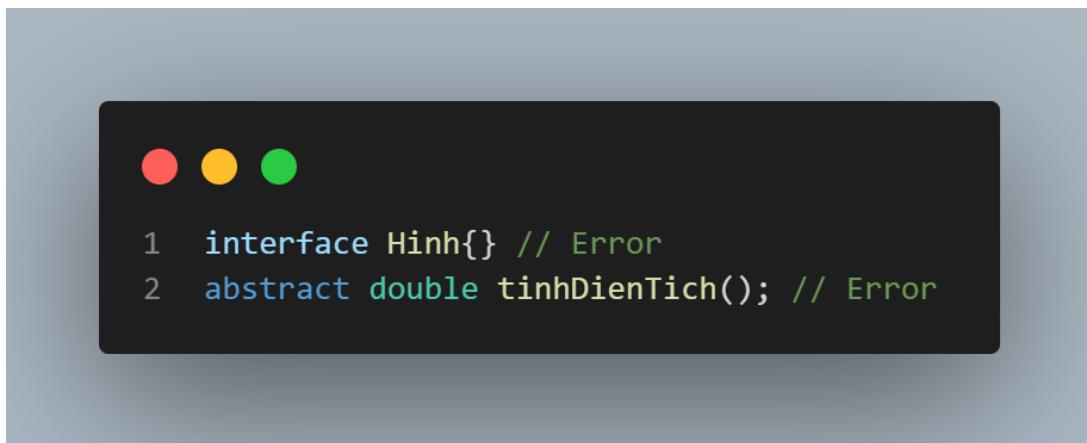


Hình 1.5: Mô tả tính kế thừa giữa các lớp

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:



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



Tính đa hình (Polymorphism)

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.

```
1 // Khai báo về tính năng của Xe
2 abstract class Xe {
3     void chuyenCho(); // Chuyên chở
4 }
5
6 // Khai báo của Xe tải
7 class XeTai extends Xe {
8     @override
9     void chuyenCho() {
10         print(" Chỉ chở hàng");
11     }
12 }
13
14 // Khai báo về Xe khách
15 class XeKhach extends Xe {
16     @override
17     void chuyenCho() {
18         print(" Chỉ chở người");
19     }
20 }
```

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.

6. Một số loại Widget thường gặp trong Flutter

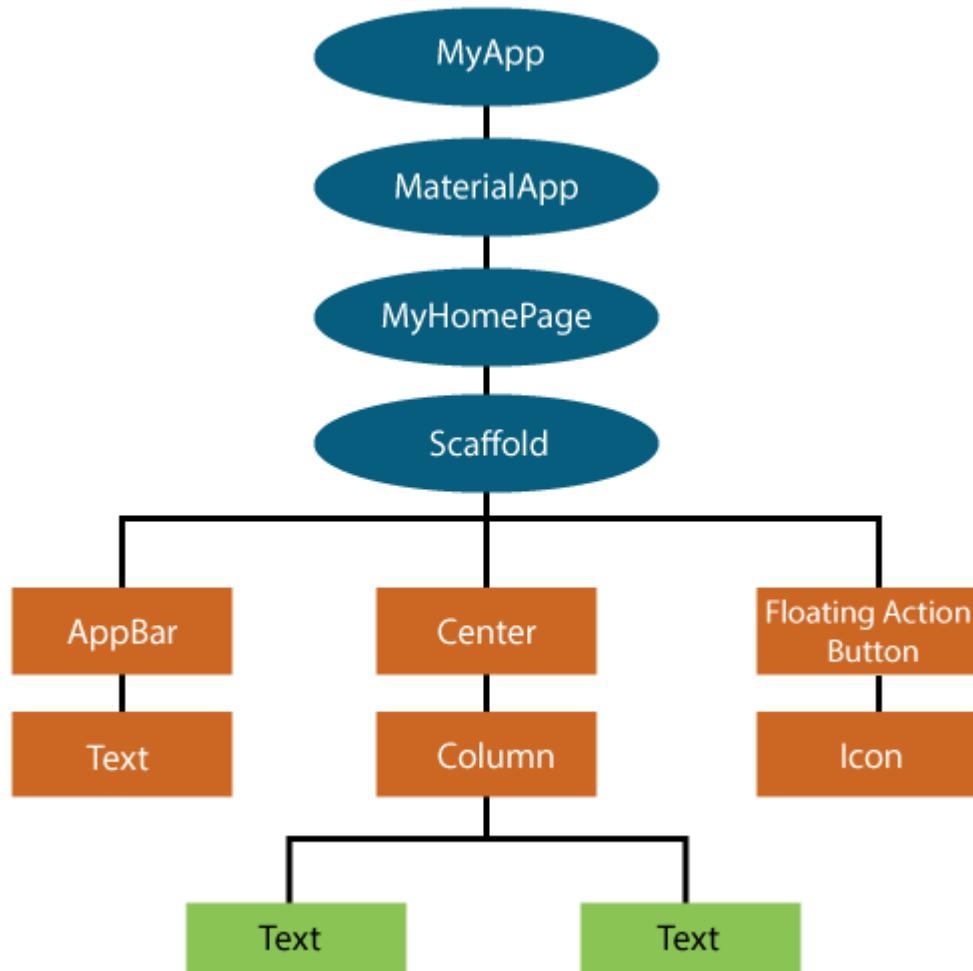
6.1. Widget Flutter

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 1.6: Cây Widget

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 TextStyle 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.

```
1 void main() {  
2     runApp(MyApp());  
3 }  
4  
5 class MyApp extends StatelessWidget {  
6     @override  
7     Widget build(BuildContext context) {  
8         return MaterialApp(  
9             theme: ThemeData(  
10                 primarySwatch: Colors.green,  
11             ),  
12             home: MyTextPage());  
13     }  
14 }  
15  
16 class MyTextPage extends StatelessWidget {  
17     @override  
18     Widget build(BuildContext context) {  
19         return Scaffold(  
20             appBar: AppBar(  
21                 title: Text("Text Widget Example"),  
22             ),  
23             body: Center(child: Text("Welcome to NgocTienTNT")),  
24         );  
25     }  
26 }
```

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 các Style Widget Text:

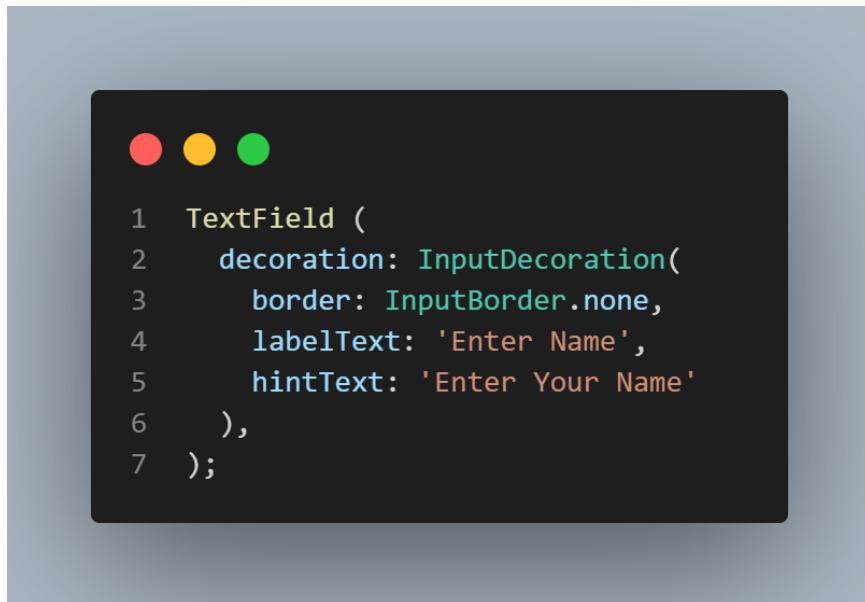
Thuộc tính	Mô tả
------------	-------

foreground	Xác định màu nền trước cho Text.
background	Xác định sơn làm nền cho Text.
fontWeight	Quyết định độ dày của Text.
fontSize	Xác định kích thước của Text.
fontFamily	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	Được sử dụng để tạo kiểu cho phông chữ ở dạng in đậm hoặc nghiêng.
color	Được sử dụng để xác định màu sắc của Text.
letterSpacing	Được sử dụng để xác định khoảng cách giữa các ký tự của Text.
wordSpacing	Được sử dụng để xác định khoảng cách giữa hai từ của Text.
shadows	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.

TextField

TextField trong Flutter là widget nhập văn bản được sử dụng phổ biến nhất cho phép người dùng thu thập dữ liệu đầu vào từ bàn phím vào một ứng dụng. Chúng ta có thể sử dụng widget TextField trong việc xây dựng biểu mẫu, gửi tin nhắn, tạo trải nghiệm tìm kiếm và nhiều hơn thế nữa. Theo mặc định, Flutter trang trí TextField bằng một gạch dưới. Chúng tôi cũng có thể thêm một số thuộc tính với TextField, chẳng hạn như nhẵn, biểu tượng, văn bản gợi ý nội tuyến và văn bản lỗi bằng cách sử dụng InputDecoration làm trang trí. Nếu chúng ta muốn loại bỏ hoàn toàn các thuộc tính trang trí, thì bắt buộc phải đặt trang trí thành null.

Ví dụ về cách sử dụng Widget TextField:



Một số thuộc tính phổ biến nhất được sử dụng với widget TextField như sau:

- **decoration:** Nó được sử dụng để hiển thị decoration xung quanh TextField.
- **border:** Nó được sử dụng để tạo một đường viền hình chữ nhật tròn mặc định xung quanh TextField.
- **labelText:** Nó được sử dụng để hiển thị văn bản nhãn trên vùng chọn TextField.
- **hintText:** Nó được sử dụng để hiển thị văn bản gợi ý bên trong TextField.
- **icon:** Nó được sử dụng để thêm các biểu tượng trực tiếp vào TextField.

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 (Text Button):** Nó là một nút nhãn văn bản không có nhiều trang trí và hiển thị mà không có bất kỳ độ cao nào . Nút phẳng có hai thuộc tính bắt buộc là: **child** và **onPressed()** . Nó chủ yếu được sử dụng trong các thanh công cụ, hộp thoại hoặc nội tuyến với các nội dung khác. Theo mặc định, nút phẳng không có màu và văn bản của nó là màu đen. Tuy nhiên, chúng ta có thể sử dụng màu cho nút và văn bản bằng cách sử dụng các thuộc tính **color** và **textColor** , tương ứng.
- **Nút nâng (Elevated Button):** Nó là một nút, dựa trên vật liệu vật liệu và có thân hình chữ nhật . Nó tương tự như một nút phẳng, nhưng nó có độ cao(elevation) sẽ tăng lên khi nút được nhấn. Nó thêm thứ nguyên cho giao diện người dùng dọc theo trục Z. Nó có một số thuộc tính như màu văn bản, hình dạng, phần đệm, màu nút, màu của nút khi bị tắt, thời gian animation, độ cao, v.v. Nút này có hai chức năng gọi lại: **onPressed ()**: Được kích hoạt khi nhấn nút. **onLongPress ()**: Được kích hoạt khi nhấn và giữ nút. Cần lưu ý rằng nút này ở trạng thái bị vô hiệu hóa nếu các lệnh gọi lại onPressed() và onLongPressed() không được chỉ định.
- **Nút viền (Outlined Button):** Nó tương tự như nút phẳng, nhưng nó có một đường viền mỏng hình chữ nhật tròn màu xám. Đường viền phác thảo của nó được xác định bởi thuộc tính shape.
- **Nút nổi (Floating Button):** Nút FAB là một nút biểu tượng hình tròn kích hoạt hành động chính trong ứng dụng của chúng ta. Nó là nút được sử dụng nhiều nhất trong các ứng dụng hiện nay. Chúng ta có thể sử dụng nút này để thêm, làm mới hoặc chia sẻ nội dung. Flutter đề xuất sử dụng nhiều nhất một nút FAB trên mỗi màn hình.

Có hai loại Nút hành động nỗi: **FloatingActionButton**: Nó tạo một nút nỗi hình tròn đơn giản với một widget con bên trong nó. Nó phải có một tham số child để hiển thị một widget.

FloatingActionButton.extended: Nó tạo ra một nút nỗi rộng cùng với một biểu tượng và nhãn bên trong nó. Thay vì một child, nó sử dụng các nhãn và các thông số biểu tượng.

- **Nút thả xuống (Drop Down Button)**: Một nút thả xuống được sử dụng để tạo một lớp phủ đẹp mắt trên màn hình cho phép người dùng chọn bất kỳ mục nào từ nhiều tùy chọn. Flutter cho phép một cách đơn giản để triển khai hộp thả xuống hoặc nút thả xuống. Nút này hiển thị mục hiện được chọn và một mũi tên mở menu để chọn một mục từ nhiều tùy chọn.
- **Nút biểu tượng (Icon Button)**: IconButton là một hình ảnh được in trên widget Material. Nó là một widget hữu ích mang lại cho giao diện người dùng Flutter một cảm giác thiết kế material design. Chúng tôi cũng có thể tùy chỉnh giao diện của nút này. Nói một cách dễ hiểu, nó là một biểu tượng phản ứng khi người dùng chạm vào nó.
- **Nút hiển thị menu (Popup Menu Button)**: Nó là một nút hiển thị menu khi nó được nhấn và sau đó gọi phương thức onPressed , menu bị loại bỏ. Đó là vì mục từ nhiều tùy chọn được chọn. Nút này chứa một văn bản và một hình ảnh. Nó sẽ chủ yếu sử dụng với menu Cài đặt để liệt kê tất cả các tùy chọn. Nó giúp tạo ra trải nghiệm người dùng tuyệt vời.

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.

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

6.3. Widget ẩn

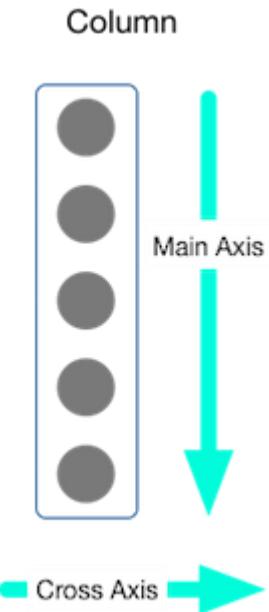
Các widget vô hình có liên quan đến cách bố trí và kiểm soát các widget. Nó cung cấp việc kiểm soát cách các widget thực sự hoạt động và cách chúng sẽ hiển thị trên màn hình. Một số loại widget quan trọng là:

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 1.7: Widget Cột

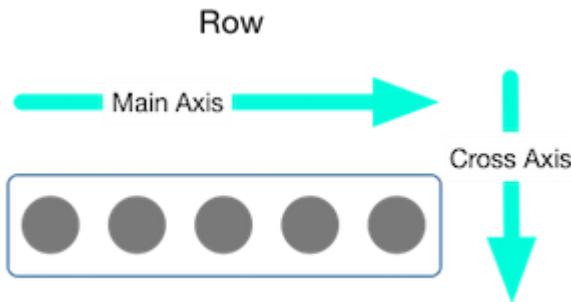
Lưu ý: widget cột cũng căn chỉnh nội dung của nó bằng cách sử dụng các thuộc tính tương tự như chúng ta đã thảo luận trong widget hàng như bắt đầu, kết thúc, giữa, khoảng trắng, khoảng trắng và khoảng trắng.

Row

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

Widget hàng có vẻ không cuộn được vì nó hiển thị các widget trong chế độ xem và hiển thị toàn bộ. Vì vậy, sẽ được coi là sai nếu chúng ta có nhiều widget cob hơn trong một hàng sẽ không phù hợp với không gian có sẵn. Nếu chúng ta muốn tạo một danh sách các widget hàng có thể cuộn được, chúng ta cần sử dụng widget ListView.

Chúng ta có thể kiểm soát cách một widget hàng căn chỉnh các con của nó dựa trên lựa chọn của chúng tôi bằng cách sử dụng thuộc tính crossAxisAlignment và mainAxisAlignment. Trục chéo của hàng sẽ chạy theo chiều dọc và trực chính sẽ chạy theo chiều ngang. Xem hình ảnh minh họa bên dưới để hiểu rõ hơn.



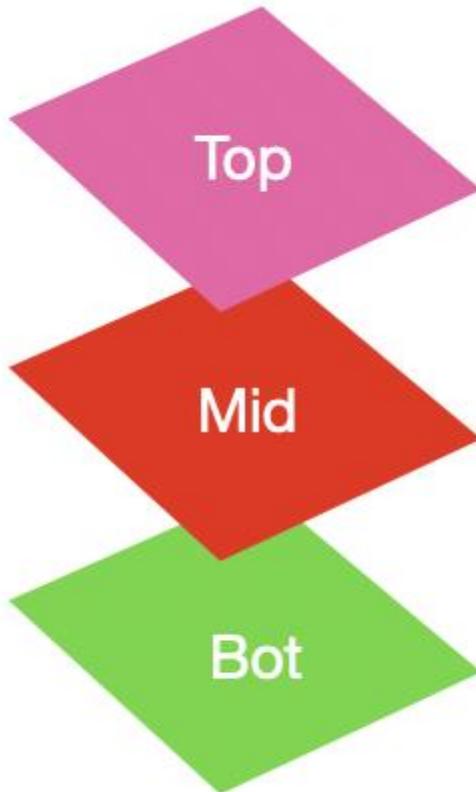
Hình 1.8: Widget Hàng

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.



Hình 1.9: Widget Ngăn Xếp

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):

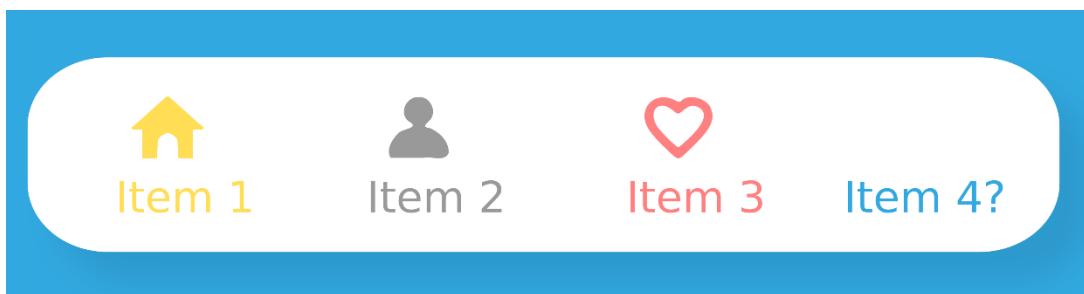


7. Tìm hiểu bố cục (layout) Trong giao diện Flutter

7.1. Bố cục

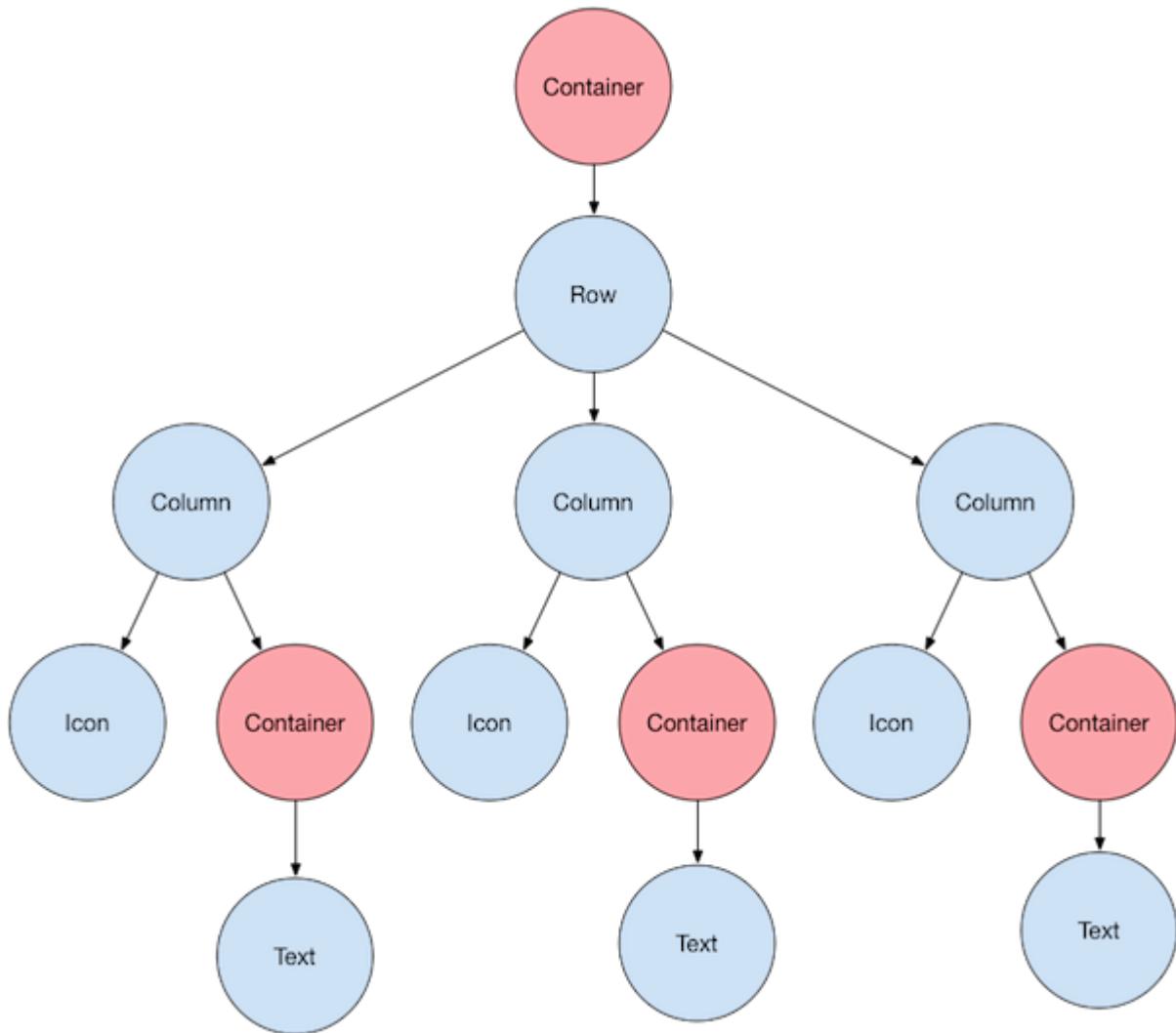
Khái niệm chính của cơ chế bố trí là widget. Chúng ta biết rằng sự flutter giả định mọi thứ như một widget Vì vậy, hình ảnh, biểu tượng, văn bản và thậm chí cả bố cục(layout) của ứng dụng của bạn đều là widget. Ở đây, một số thứ bạn không thấy trên giao diện người dùng ứng dụng của mình, chẳng hạn như 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 1.10: 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.



1.11: Nội dung của Container

Trong hình trên, vùng chúa 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 chúa để thêm lề. Toàn bộ hàng cũng được đặt trong một vùng chúa 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.

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.

7.3. Các loại Widget bố cục

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

- Widget đơn
- Widget đa

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.



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.



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.



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.



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.



```
1 class MyApp extends StatelessWidget {
2     // It is the root widget of your application.
3     @override
4     Widget build(BuildContext context) {
5         return MaterialApp(
6             title: 'Multiple Layout Widget',
7             debugShowCheckedModeBanner: false,
8             theme: ThemeData(
9                 // This is the theme of your application.
10                primarySwatch: Colors.blue,
11            ),
12            home: MyHomePage(),
13        );
14    }
15 }
16
17 class MyHomePage extends StatelessWidget {
18     @override
19     Widget build(BuildContext context) {
20         return Center(
21             child: Container(
22                 alignment: Alignment.center,
23                 color: Colors.white,
24                 child: Row(
25                     children: [
26                         Expanded(child: Text('Peter', textAlign: TextAlign.center)),
27                         Expanded(child: Text('John', textAlign: TextAlign.center)),
28                         Expanded(
29                             child: FittedBox(
30                                 fit: BoxFit.contain, // otherwise the logo will be tiny
31                                 child: const FlutterLogo(),
32                             ),
33                         ),
34                     ],
35                 ),
36             ),
37         );
38     }
39 }
```

8. Tìm hiểu về Cử chỉ (Gestures) với giao diện 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
- 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)

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.

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
 - onVerticalDragUpdate
 - onVerticalDragEnd

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.

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.

9. Quản lý State

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.

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 đổi 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)

9.1. 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.

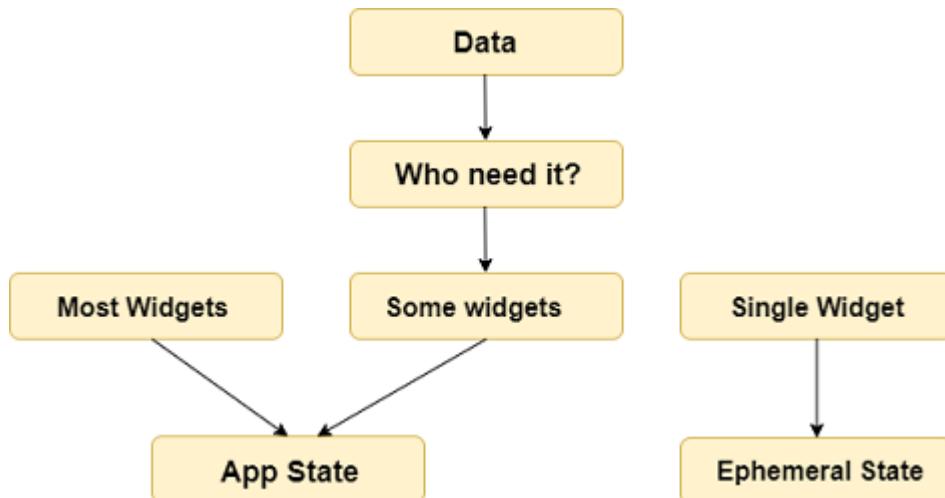
```
1 class MyHomepage extends StatefulWidget {  
2     @override  
3     MyHomepageState createState() => MyHomepageState();  
4 }  
5  
6 class MyHomepageState extends State<MyHomepage> {  
7     String _name = "Peter";  
8     @override  
9     Widget build(BuildContext context) {  
10         return ElevatedButton(  
11             child: Text(_name),  
12             onPressed: () {  
13                 setState(() {  
14                     _name = _name == "Peter" ? "John" : "Peter";  
15                 });  
16             },  
17         );  
18     }  
19 }
```

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.

9.2. 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 1.12: 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 `addListener()` 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

10. Quản lý thư viện và các gói trong Flutter (Lib & Package)

10.1. Nạp thư viện trong Dart

Một thư viện có thể chứa tập hợp các lớp, hàm, hằng ... để bạn nạp vào và sử dụng khi cần thiết.

Trong Dart để nạp thư viện bạn dùng từ khóa import ở đầu file. Với cú pháp:

```
import 'uri';
```

Nếu chỉ nạp một phần nào đó của thư viện sử dụng thêm từ khóa *show* có dạng:

```
import 'uri' show part1, part2, part3;
```

Khi nạp thư viện, để tránh xung đột về tên, bạn cũng có thể đặt lại bằng *as*

Trong đó uri như là một định danh duy nhất trỏ đến thư viện cần nạp. Nó có thể có các dạng:

Các thư viện xây dựng sẵn của Dart có cấu trúc uri `dart:tên_thư_viện` ví dụ `dart:convert`, `dart:html`, `dart:math`, `dart:js`, `dart:web_sql`. Ví dụ cần nạp thư viện toán học `dart:math` thì dùng

```
import 'dart:math';
```

Nạp từ file dự án, thì url chỉ đến đường dẫn file dart cần nạp vào. Ví dụ bạn có file `lib/mylib.dart` trong dự án thì nạp vào dùng bằng:

```
import 'lib/myLib.dart';
```

Thư viện nạp từ các gói package tải về, thì uri có dạng `package:tên_gói/thư_viện_gói`. Ví dụ gói `googleapis_auth`, có thành phần `auth_browser` cung cấp chức năng xác thực Auth với tài khoản google, thì nạp thư viện đó vào bằng:

```
import "package:googleapis_auth/auth_browser.dart";
```

10.2. Các thư viện cung cấp sẵn

Những thư viện Dart đi cùng SDK Dart, bạn có thể nạp luôn bằng import '`dart:tên_thư_viện`', danh sách thư viện như: (chi tiết tại [LIBRARIES](#))

Thư viện	Chức năng
<code>dart:core</code>	Thư viện cung cấp các hàm, lớp cơ bản, nó tự động nạp nên bạn không cần import thư viện này

dart:collection	Cung cấp các cấu trúc dữ liệu như HashSet, HashMap, Queue, ...
dart:math	Cung cấp hàm toán học
dart:convert	Tính năng mã hóa, giải mã dữ liệu, kể cả JSON, UTF-8
dart:io	Thư viện IO cung cấp các chức năng về File, Socket, HTTP

10.3. Tạo thư viện trong Dart

Giả sử dự án của bạn muốn tạo ra một thư viện có tên myfirstlib, thì bạn tạo ra file myfirstlib.dart có đường dẫn nằm trong dự án, ví dụ lib/myfirstlib.dart

Sau đó bạn code các hàm, lớp ... trong file này, ở đầu file này có đặt dòng khai báo là thư viện:

```
library myfirstlib;
```

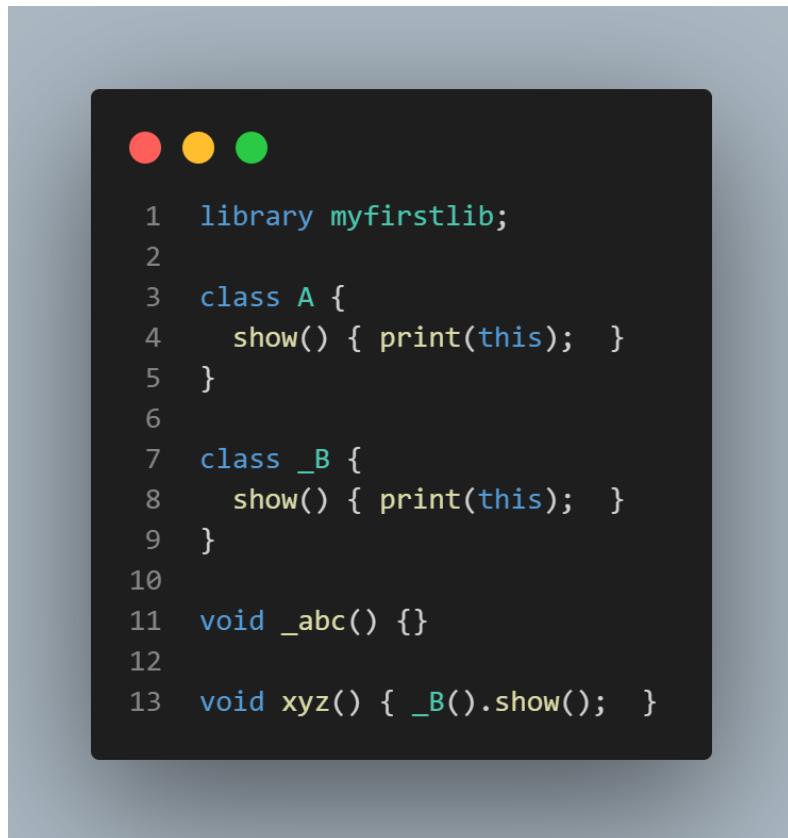
Sau đó khi nào cần sử dụng thư viện này chỉ việc gọi:

```
import 'lib/myfirstlib.dart';
```

Sau khi import là có thể dùng đến các thành phần lớp, hàm, hằng ... định nghĩa trong thư viện.

Cũng lưu ý, nếu thành phần nà (hàm, lớp ...) có tên bắt đầu bằng _ thì nó là của riêng thư viện, không thể import dùng được.

Ví dụ nội dung myfirstlib.dart



```
1 library myfirstlib;
2
3 class A {
4     show() { print(this); }
5 }
6
7 class _B {
8     show() { print(this); }
9 }
10
11 void _abc() {}
12
13 void xyz() { _B().show(); }
```

Thì lớp `_B` không thể import được, hàm `_abc` không thể import được.

Hoặc chỉ nạp hàm xyz thì gọi.

`import 'lib/myfirstlib.dart' show xyz;`

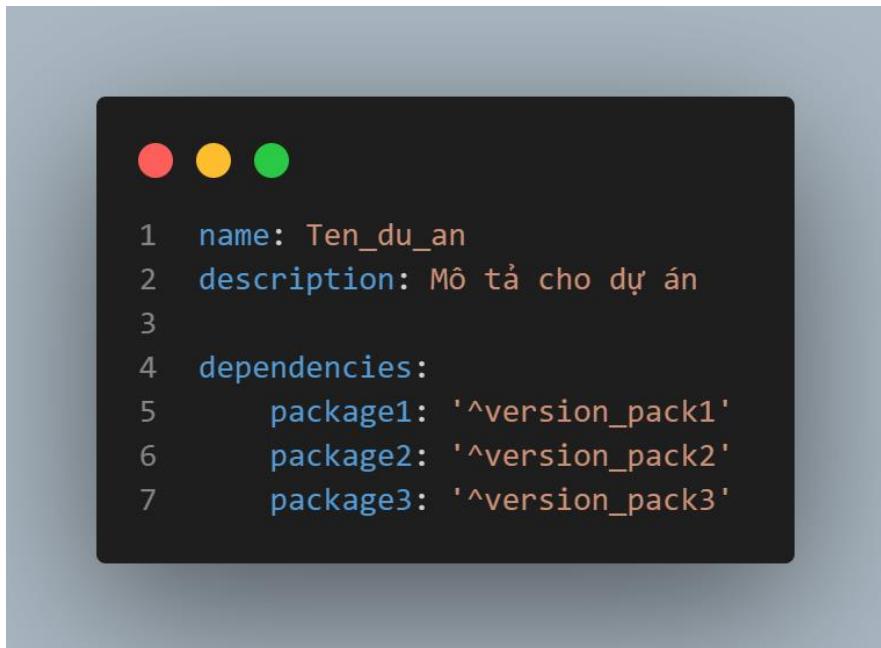
10.4. Cài đặt các gói

Dart cũng có trình quản lý gói package riêng là pub (nó giống NPM của NodeJS, giống Nuget của .Net, Gradle của Java ...)

Tìm các Package ở đâu, các gói được chia sẻ và cung cấp lưu tại DART PUB, bạn chỉ việc tìm kiếm ở đó, gói thư viện nào phù hợp thì tích hợp vào dự án!

Để tích hợp được các gói vào dự án, thì ở thư mục gốc của dự án bạn phải có file `pubspec.yaml`, nếu chưa có hãy tạo ra nó.

Sau đó liệt kê các gói cần tích hợp ở mục `dependencies`, liệt kê đúng cấu trúc yaml, môt package chỉ ra tên `tên_package: version`



```
1 name: Ten_du_an
2 description: Mô tả cho dự án
3
4 dependencies:
5   package1: '^version_pack1'
6   package2: '^version_pack2'
7   package3: '^version_pack3'
```

Ví dụ muốn dùng package dialog, thì file pubspec.yaml

Dùng thêm package google_maps thì thêm vào:



```
1 name: Ten_du_an
2 description: Mô tả cho dự án
3
4 dependencies:
5   dialog: '^0.8.0'
6   google_maps: '^6.2.0'
```

Chú ý name là tên dự án của bạn, đặt theo quy tắc đặt tên biến.

Khi có file pubspec.yaml từ terminate chỉ việc chạy các lệnh pub để cài đặt, cập nhật....

Cài đặt các package gõ lệnh *pub get*

Tương tự, để cập nhật phiên bản các package gõ lệnh *pub update*

Sau khi cài được các package thì chỉ việc dùng *import* để nạp package, và sử dụng.

10.5. Các gói hay sử dụng

10.5.1. Flutter Bloc

Bloc là một lib để quản lý state cho Flutter application. B.L.o.C nghĩa là Business Logic Component. Nhận 'Event' như là input và trả về output là 'State'. Bloc được xây dựng dựa trên RxDart.

Chúng ta có thể chia Flutter application architecture thành 3 lớp sau:

- View Layer (Presentation Layer): Có nhiệm vụ render chính nó dựa trên một hoặc nhiều state của bloc. Nó cũng xử lý các sự kiện input của user và lifecycle event của application.
- Bloc (Business Logic): Có nhiệm vụ nhận event từ lớp biểu diễn và trả về state mới. Hoạt động như một cầu nối giữa lớp data và lớp presentation.
- Data Layer: Có nhiệm vụ cung cấp data từ bất kể nguồn nào. Data Provider cung cấp data tho và repository sẽ là trình đóng gói một hoặc nhiều data providers.

Vậy có nghĩa là Bloc đứng giữa 2 lớp View và Data.

Một số concept cho Bloc:

1. **Events:** Event được truyền vào một Bloc. Nó giống như concept action trong Redux. Trong lớp presentation, event được tạo ra bởi tương tác của user như button click và truyền vào Bloc. Một event có thể bao gồm vài data được thêm vào.
2. **States:** State là một phần của application state. Nó là output của Bloc. Khi state thay đổi, thành phần UI sẽ được thông báo và dựa vào current state, nó sẽ tự render lại.
3. **Transition:** Thay đổi từ một state sang state khác gọi là transition. Nó bao gồm current state, event và state tiếp theo.

Thư viện Flutter Bloc: [tại đây](#)

10.5.2. Provider

Provider là loại package cơ bản nhất trong số các loại package của Provider. Bạn có thể sử dụng nó để cung cấp một giá trị (thường là một data model) cho bất kỳ vị trí nào trong widget tree. Tuy nhiên, nó sẽ không giúp bạn cập nhật widget tree khi mà giá trị đó thay đổi. Bạn có thể hình dung như việc nó chỉ set dữ liệu vào mà UI nó không có sự thay đổi gì, không nhận biết được sự thay đổi.

Để có thể cung cấp data model đó cho widget tree bạn cần bọc phần trên cùng của widget tree bằng một Provider. Với tham số truyền vào là data model vừa tạo. Trong widget tree để có thể tham chiếu đến data model, cần sử dụng Consumer widget để lấy ra data model vừa truyền vào ở trên.

Một số thành phần cơ bản của Provider:

- Provider
- ChangeNotifierProvider
- ValueListenableProvider
- StreamProvider
- FutureProvider
- MultiProvider
- ProxyProvider
- ChangeNotifierProxyProvider

ChangeNotifierProvider

Không giống như Provider, ChangeNotifierProvider lắng nghe các thay đổi trong data model. Khi có thay đổi, nó sẽ xây dựng lại bất kỳ widget nào trong Consumer.

Trong hàm build thay đổi Provider thành ChangeNotifierProvider. Lớp mô hình cần sử dụng extend ChangeNotifier (hoặc with ChangeNotifier). Điều này cung cấp cho bạn quyền truy cập vào notifyListeners() và bất kỳ lúc nào bạn gọi notifyListeners() thì ChangeNotifierProvider sẽ được thông báo và tất cả các widget bên trong Consumers sẽ được rebuild lại.

FutureProvider

FutureProvider về cơ bản chỉ là một wrapper với bên trong là FutureBuilder. Bạn cung cấp cho nó một số dữ liệu ban đầu để hiển thị trong giao diện người dùng và cũng có thể cung cấp cho nó một hoạt động bất đồng

bộ Future của giá trị mà bạn muốn cung cấp. FutureProvider lắng nghe khi Future hoàn thành và sau đó thông báo cho Consumers để xây dựng lại các widget của nó.

StreamProvider

StreamProvider về cơ bản là một wrapper với bên trong là một StreamBuilder. Bạn cung cấp một Stream và sau đó Consumer được xây dựng lại khi có sự kiện trong stream. Thiết lập rất giống với FutureProvider ở trên.

ValueListenableProvider

Nó giống như ChangeNotifierProvider nhưng phức tạp hơn và không có bất kỳ giá trị tăng rõ ràng nào.

MultiProvider

Nếu bạn cần cung cấp từ 2 loại model object trả lên, bạn có thể lồng các provider Tuy nhiên, có một cách khác gọn gàng hơn là sử dụng MultiProvider.

ProxyProvider

Điều gì sẽ xảy ra nếu bạn có hai Model mà bạn muốn cung cấp, nhưng một trong các Model phụ thuộc vào mô hình còn lại? Trong trường hợp đó, bạn có thể sử dụng ProxyProvider. ProxyProvider lấy giá trị từ một provider và cho phép nó được đưa vào provider khác.

ProxyProvider cơ bản có hai loại model trong dấu ngoặc nhọn. Model thứ hai thì phụ thuộc vào model đầu tiên.

Thư viện Provider: [tại đây](#)

10.5.3. Get

Get làm được những gì:

- State Manager : Quản lý state trong Flutter
- Navigation Manager: Quản lý việc điều hướng
- Dependencies Manager: Cung cấp giải pháp dependencies injection tuyệt vời
- Utils function: Các hàm tiện ích cực kỳ hữu ích trong lập trình Flutter

Get - State manager

Những ưu điểm của Get - State manager

- Chỉ update những widget cần thiết
- Sử dụng ít bộ nhớ hơn so với các kiểu quản lý state khác
- Quên đi StatefulWidget. Với Get các bạn không phải suy nghĩ sử dụng StatefulWidget hay StateLessWidget nữa. Bây giờ bạn chỉ việc một component duy nhất là GetWidget
- Việc tổ chức cấu trúc project sẽ cực kỳ clear, phần code logic được tách hẳn hoàn toàn so với UI (mình sẽ demo structure dự án ở bài cuối của seri)
- Update widgets without spending ram for that
- Tối ưu hóa bộ nhớ, bạn sẽ không phải lo lắng việc Out Memory nữa, Get sẽ tự động thu gọn những component không cần thiết

Get Route manager

Để sử dụng được Get Route manager bạn cũng cần phải sử dụng GetMaterialApp thay vì MaterialApp như thông thường

Thư viện Get: [tại đây](#)

10.5.4. Rx Dart

RxDart là một thư viện cung cấp một lượng class và function rất đồ sộ cho Stream. Ta sẽ chia ra làm 3 nhóm nhỏ:

- Nhóm 1 - Những class Stream do RxDart cung cấp
- Nhóm 2 - Những extension function dành cho lớp Stream
- Nhóm 3 – Subjects

Những class Stream của RxDart:

- **MergeStream:** ta có 2 stream gọi là Stream 1 và Stream 2 chạy song song nhau và 2 stream đó được merge lại thành 1 MergeStream, các event trong MergeStream được emit theo thứ tự đúng y hệt dòng thời gian của Stream 1 và Stream 2. Thực tế bạn thích truyền bao nhiêu Stream vào MergeStream để merge lại cũng được.
- **ZipStream:** ta có 2 Stream là Stream 1 và Stream 2 chạy song song, 2 Stream này cũng được merge lại thành 1 ZipStream.
- **TimerStream:** Truyền vào 1 value, và 1 duration. Sau 1 khoảng thời gian duration đó Stream mới emit value được truyền vào.

- **RangeStream:** Truyền vào 2 biến: startInclusive và endInclusive để tạo thành 1 range. RangeStream sẽ emit một chuỗi các số nguyên liên tiếp trong range đó.
- **RetryStream:** Truyền vào 1 stream. Khi stream được truyền vào gặp lỗi RetryStream sẽ cho retry lại nó. Chúng ta có thể truyền thêm 1 biến count chính là số lần được phép retry. Nếu không truyền biến count thì nó sẽ retry mãi mãi.

Những extension function của RxDart:

- **debounceTime:** Hàm này cho phép chúng ta truyền vào 1 Duration, cụ thể trong hình là 10s. Nếu Source Stream emit ra 1 event mà sau 10s nó không emit tiếp 1 event khác thì Output Stream mới emit event đó ra. Hàm này cực kỳ hữu ích để làm tính năng live search cho app (real time search) bởi vì app sẽ tránh spam server do call API quá nhiều.
- **onErrorResumeNext:** Hàm này truyền vào 1 Stream gọi là recoveryStream đi ha. Khi Source Stream gặp lỗi thay vì nó emit cái event lỗi đó thì nó lại vào emit các phần tử trong recoveryStream
- **interval:** Hàm này truyền vào 1 duration. Cứ sau khoảng thời gian duration đó thì nó mới emit ra 1 event.
- **concatWith:** Merge 2 stream lại thành 1 ConcatStream và emit tất cả event trong lần lượt từng stream một. Sau khi emit xong tất cả event của Stream 1, thì ConcatStream vẫn chờ x giây (trong hình) mới emit phần tử đầu tiên của Stream 2 chứ không emit ngay lập tức. Chính vì vậy tổng thời gian của t1 (thời gian stream 1 dừng) + t2 (thời gian stream 2 dừng) = t3 (thời gian ConcatStream dừng).
- **distinctUnique:** hàm distinctUnique của RxDart dùng để emit các phần tử không trùng nhau trong Source Stream.

Subjects:

- **BehaviorSubject:** Đây cũng là một Broadcast StreamController. Đặc điểm của thằng này là mỗi khi có một listener mới subscribe thì nó lập tức emit phần tử mới nhất vừa được add vào controller (the latest item that has been added to the controller)

- **ReplaySubject:** Đây cũng là một Broadcast StreamController. Thằng này còn bá đạo hơn thằng BehaviorSubject là vừa chào đời không những nó chụp 1 data mới nhất mà nó chụp tất cả data đã được add vào controller luôn.
- **PublishSubject:** nó rất giống một Broadcast StreamController thông thường. Có nghĩa là lúc subscription thứ 2 được tạo ra nó sẽ không chôm chĩa bất cứ data nào trước đó của controller mà nó chờ có data mới được add vào controller nó mới nhận.

Thư viện RxDart: [tại đây](#)

10.5.5. SQFLite

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.

Thư viện SQFLite: [tại đây](#)

10.5.6. Shared Preferences

Shared Preferences là nơi bạn có thể lưu trữ các thông tin dưới dạng key-value được xây dựng sẵn trong hệ điều hành Android.

Thư viện Shared Preferences: [tại đây](#)

10.5.7. Http

Flutter cung cấp cho ta package http để sử dụng các phương thức HTTP. Http là một thư viện dựa trên Future và sử dụng các chức năng await và async.

Nó cung cấp nhiều phương thức cấp cao và đơn giản hóa việc phát triển của ứng dụng di động dựa trên REST.

Thư viện Http: [tại đây](#)

10.5.8. Url Launcher

Url Launcher được sử dụng để mở trang web, tạo email, mở bản đồ,... Url Launcher sẽ giúp bạn tương tác với các ứng dụng bên ngoài.

Thư viện Url Launcher: [tại đây](#)

10.5.9. Font Awesome Flutter

Thư viện này cung cấp cho bạn rất nhiều icon đẹp từ [fontawesome.com](#)

Thư viện Font Awesome Flutter: [tại đây](#)

11.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().

11.1. 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.

```
1  class FirstRoute extends StatelessWidget {
2      @override
3      Widget build(BuildContext context) {
4          return Scaffold(
5              appBar: AppBar(
6                  title: Text('First Route'),
7              ),
8              body: Center(
9                  child: ElevatedButton(
10                     child: Text('Open route'),
11                     onPressed: () {
12                         // Navigate to second route when tapped.
13                     },
14                     ),
15                     ),
16                     );
17                 }
18             }
19
20             class SecondRoute extends StatelessWidget {
21                 @override
22                 Widget build(BuildContext context) {
23                     return Scaffold(
24                         appBar: AppBar(
25                             title: Text("Second Route"),
26                         ),
27                         body: Center(
28                             child: ElevatedButton(
29                                 onPressed: () {
30                                     // Navigate back to first route when tapped.
31                                 },
32                                     child: Text('Go back! '),
33                                     ),
34                                     ),
35                                     );
36                 }
37             }
```

11.2. Điều hướng sang route thứ hai bằng phương thức Navigator.push()

Fương 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().



11.3. 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:



11.4. Điều hướng với các route được đặt tên

Chúng ta đã biết cách điều hướng đến một màn hình mới bằng cách tạo một tuyến đường mới và quản lý nó bằng cách sử dụng Bộ điều hướng. Bộ điều hướng duy trì lịch sử dựa trên ngăn xếp của các tuyến đường. Nếu có nhu cầu điều hướng đến cùng một màn hình trong nhiều phần của ứng dụng, thì cách làm này không có lợi vì nó dẫn đến trùng lặp mã. Giải pháp cho vấn đề này có thể được loại bỏ bằng cách xác định các tuyến đường được đặt tên và có thể sử dụng các tuyến đường được đặt tên để điều hướng.

Chúng ta có thể làm việc với các tuyến đường được đặt tên bằng cách sử dụng hàm `Navigator.pushNamed()`. Hàm này nhận hai đối số bắt buộc (xây dựng ngữ cảnh và chuỗi) và một đối số tùy chọn. Ngoài ra, chúng ta biết về `MaterialPageRoute`, chịu trách nhiệm chuyển đổi trang. Nếu chúng ta không sử dụng điều này, thì rất khó để thay đổi trang.

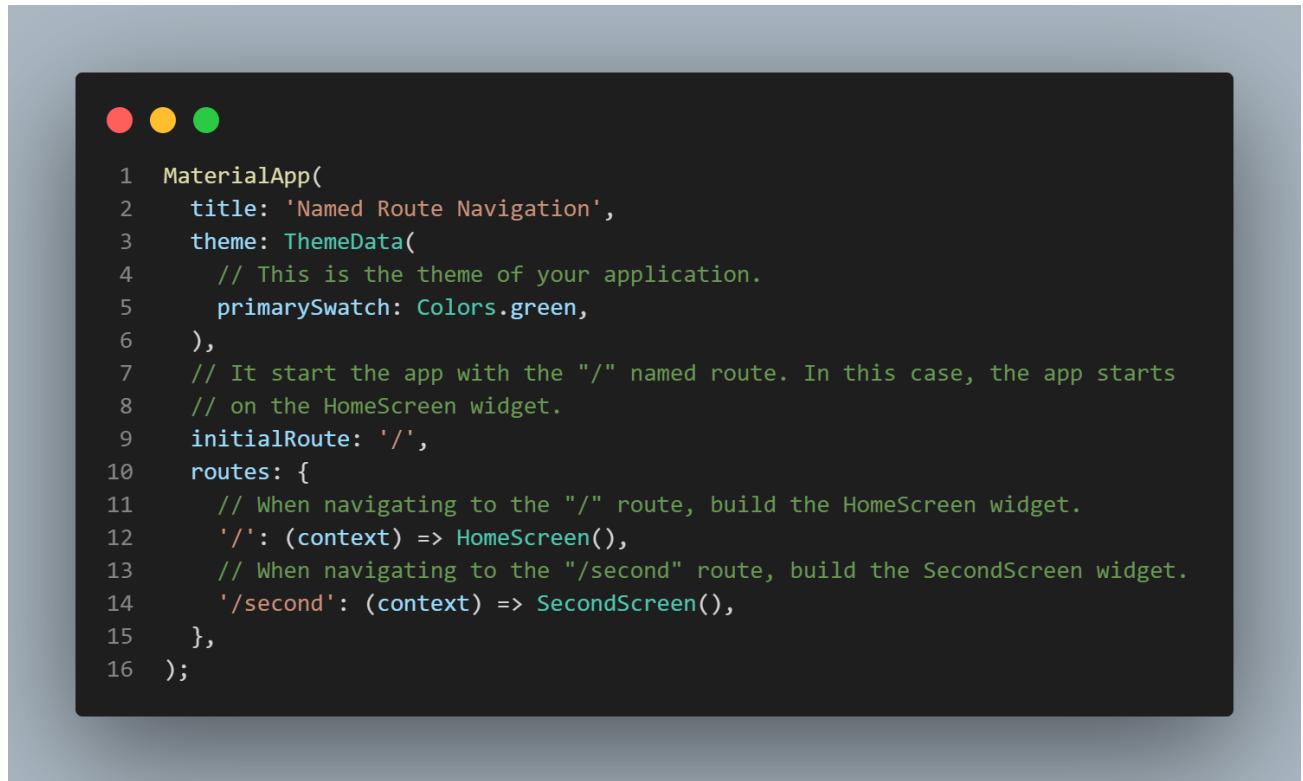
Các bước sau là cần thiết, trình bày cách sử dụng các tuyến đường được đặt tên.

Bước 1: Đầu tiên, chúng ta cần tạo hai màn hình. Đoạn mã sau tạo hai màn hình trong ứng dụng của chúng ta.

```
1 class HomeScreen extends StatelessWidget {
2     @override
3     Widget build(BuildContext context) {
4         return Scaffold(
5             appBar: AppBar(
6                 title: Text('Home Screen'),
7             ),
8             body: Center(
9                 child: ElevatedButton(
10                     child: Text('Click Here'),
11                     onPressed: () {
12                         // ...
13                     },
14                     ),
15                     ),
16                 );
17             }
18         }
19
20 class SecondScreen extends StatelessWidget {
21     @override
22     Widget build(BuildContext context) {
23         return Scaffold(
24             appBar: AppBar(
25                 title: Text("Second Screen"),
26             ),
27             body: Center(
28                 child: ElevatedButton(
29                     onPressed: () {
30                         // ...
31                     },
32                     child: Text('Go back!'),
33                     ),
34                     ),
35                 );
36             }
37 }
```

Bước 2: Xác định các tuyến đường(routes).

Trong bước này, chúng ta phải xác định các tuyến đường. Phương thức khởi tạo MaterialApp chịu trách nhiệm xác định tuyến đường ban đầu và các tuyến đường khác. Tại đây, tuyến đường ban đầu cho biết phần bắt đầu của trang và thuộc tính tuyến đường xác định các tuyến đường và tiện ích con được đặt tên có sẵn. Đoạn mã sau đây giải thích rõ ràng hơn.



```
1  MaterialApp(
2      title: 'Named Route Navigation',
3      theme: ThemeData(
4          // This is the theme of your application.
5          primarySwatch: Colors.green,
6      ),
7      // It start the app with the "/" named route. In this case, the app starts
8      // on the HomeScreen widget.
9      initialRoute: '/',
10     routes: {
11         // When navigating to the "/" route, build the HomeScreen widget.
12         '/': (context) => HomeScreen(),
13         // When navigating to the "/second" route, build the SecondScreen widget.
14         '/second': (context) => SecondScreen(),
15     },
16 );
```

Bước 3: Điều hướng đến màn hình thứ hai bằng chức năng Navigator.pushNamed().

Trong bước này, chúng ta cần gọi phương thức Navigator.pushNamed() để điều hướng. Đối với điều này, chúng ta cần cập nhật một lệnh gọi lại onPressed() trong phương thức xây dựng của Màn hình chính như các đoạn mã bên dưới.



Bước 4: Sử dụng hàm Navigator.pop() để quay lại màn hình đầu tiên.

Đây là bước cuối cùng, nơi chúng ta sẽ sử dụng phương thức Navigator.pop() để quay lại màn hình đầu tiên.



12.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, file, 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 SQLite:** 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

12.1. 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.



```
1 dependencies:
2   flutter:
3     sdk: flutter
4
5   sqflite: any
6   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.



```
1 class Book {
2   final int id;
3   final String title;
4   final int price;
5   Book({required this.id, required this.title, required this.price});
6 }
```

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 `getDatabasePath()` và kết hợp nó với gói đường dẫn.
- Sử dụng hàm `openDatabase()` để mở cơ sở dữ liệu.



```

1 final Future<Database> database = openDatabase(
2     join(await getDatabasesPath(), 'book_database.db'),
3 );

```

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.



```

1 final Future<Database> database = openDatabase(
2     join(await getDatabasesPath(), 'book_database.db'),
3     // When you create a database, it also needs to create a table to store books.
4     onCreate: (db, version) {
5         return db.execute(
6             "CREATE TABLE books(id INTEGER PRIMARY KEY, title TEXT, price INTEGER)",
7         );
8     },
9     version: 1,
10 );

```

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()`

```
1 class Book {  
2     final int id;  
3     final String title;  
4     final int price;  
5     Book({required this.id, required this.title, required this.price});  
6  
7     /* It converts a Book into a Map. The keys correspond to  
8      the names of the columns in the database.*/  
9     Map<String, dynamic> toMap() {  
10        return {'id': id, 'title': title, 'price': price};  
11    }  
12  
13    Future<void> insertBook(Book book) async {  
14        final Database db = await database;  
15        await db.insert(  
16            'books',  
17            book.toMap(),  
18            conflictAlgorithm: ConflictAlgorithm.replace,  
19            );  
20        }  
21    }  
}
```

Bước 6: Lấy danh sách 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 sách book tất cả các cuốn book. Nó bao gồm hai bước:

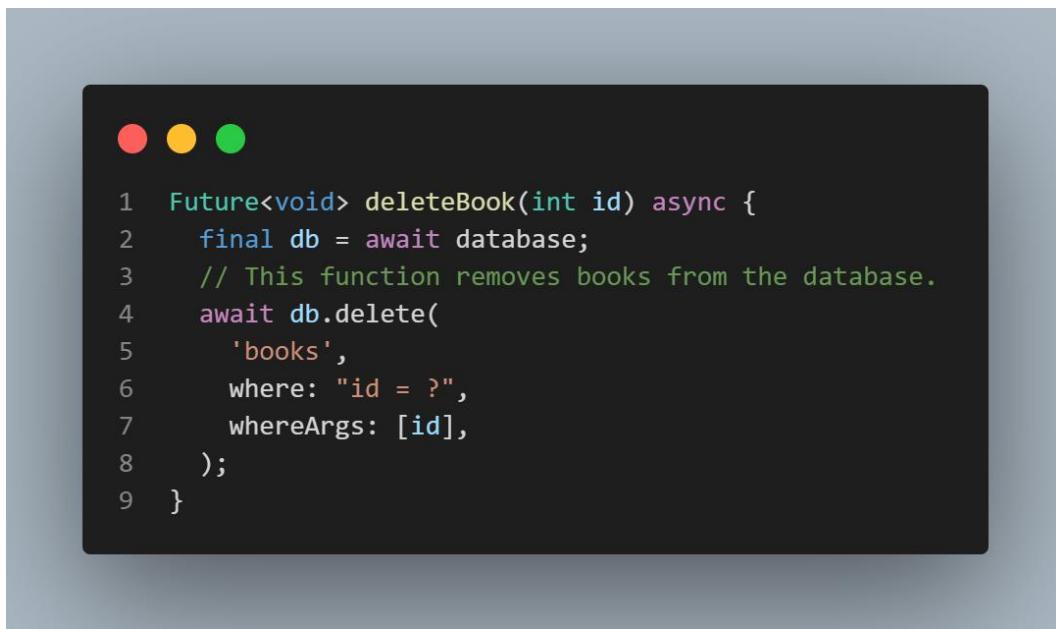
- Chạy một truy vấn trả về List<Map>.
- Chuyển List<Map> thành List<book>.

```
1 Future<List<Book>> books() async {
2     final Database db = await database;
3     // Use query for all Books.
4     final List<Map<String, dynamic>> maps = await db.query('maps');
5     return List.generate(maps.length, (i) {
6         return Book(
7             id: maps[i]['id'],
8             title: maps[i]['title'],
9             price: maps[i]['price'],
10        );
11    });
12 }
13
14 // It prints all the books.
15 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.



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.

12.2. Firebase – NoSQL lưu trữ online

12.2.1. Sơ lược về Firebase

FireBase là một dạng database lưu trữ theo cách truyền thống. Mọi lưu trữ data trong bộ sưu tập giống như bảng trong database truyền thống. Tài liệu được lưu trữ trong các bộ sưu tập này. Các 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).

12.2.2. Thêm Firebase vào ứng dụng Flutter của bạn

Điều kiện tiên quyết

Cài đặt trình soạn thảo hoặc IDE ưa thích của bạn.

Thiết lập thiết bị hoặc trình mô phỏng để chạy ứng dụng của bạn. Trình giả lập phải sử dụng hình ảnh giả lập với Google Play.

Đảm bảo rằng ứng dụng của bạn đáp ứng các yêu cầu sau:

- API mục tiêu cấp 19 (KitKat) trở lên
- Sử dụng Android 4.4 trở lên

Cài đặt Flutter cho hệ điều hành cụ thể của bạn, bao gồm:

- SDK Flutter
- Thư viện hỗ trợ
- SDK và phần mềm dành riêng cho nền tảng
- Đăng nhập vào Firebase bằng tài khoản Google của bạn.

Nếu bạn chưa có ứng dụng Flutter, bạn có thể hoàn thành Get Started: Test Drive để tạo ứng dụng Flutter mới bằng trình chỉnh sửa hoặc IDE ưa thích của bạn.

Bước 1: Cài đặt các công cụ dòng lệnh cần thiết

Nếu bạn chưa cài đặt Firebase CLI .

Đăng nhập vào Firebase bằng tài khoản Google của bạn bằng cách chạy lệnh sau:

```
firebase login
```

Cài đặt FlutterFire CLI bằng cách chạy lệnh sau từ bất kỳ thư mục nào:

```
dart pub global activate flutterfire_cli
```

Bước 2: Định cấu hình ứng dụng của bạn để sử dụng Firebase

Sử dụng FlutterFire CLI để định cấu hình các ứng dụng Flutter của bạn để kết nối với Firebase.

Từ thư mục dự án Flutter của bạn, hãy chạy lệnh sau để bắt đầu quy trình cấu hình ứng dụng:

```
flutterfire configure
```

Bước 3: Khởi tạo Firebase trong ứng dụng của bạn

Từ thư mục dự án Flutter của bạn, hãy chạy lệnh sau để cài đặt plugin cốt lõi:

```
flutter pub add firebase_core
```

Từ thư mục dự án Flutter của bạn, hãy chạy lệnh sau để đảm bảo rằng cấu hình Firebase của ứng dụng Flutter của bạn được cập nhật:

```
flutterfire configure
```

Trong tệp lib/main.dart của bạn, hãy nhập plugin lõi Firebase và tệp cấu hình bạn đã tạo trước đó:

```
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
```

Cũng trong tệp lib/main.dart của bạn, hãy khởi tạo Firebase bằng đối tượng DefaultFirebaseOptions được xuất bởi tệp cấu hình:

```
await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
```

Xây dựng lại ứng dụng Flutter của bạn:

```
flutter run
```

Bước 4: Thêm plugin Firebase

Bạn truy cập Firebase trong ứng dụng Flutter của mình thông qua các plugin Firebase Flutter khác nhau, một plugin cho mỗi sản phẩm Firebase (ví dụ: Cloud Firestore, Authentication, Analytics, v.v.).

Vì Flutter là một khuôn khổ đa nền tảng, nên mỗi plugin Firebase đều có thể áp dụng cho Apple, Android và các nền tảng web. Vì vậy, nếu bạn thêm bất kỳ plugin Firebase nào vào ứng dụng Flutter của mình, plugin đó sẽ được Apple, Android và các phiên bản web của ứng dụng của bạn sử dụng.

CHƯƠNG 3: XÂY DỰNG ỦNG DỤNG

1. Tổng quan

1.1. Tên ứng dụng: Phần mềm quản lý chi tiêu cá nhân trên di động

1.2. Lý do chọn ứng dụng

Hiện nay thiết bị di động – smart phone ngày càng phổ biến. Việc đưa một ứng dụng lên các kho lưu trữ tương đối dễ dàng. Hơn nữa, người sử dụng cũng sẵn sàng chi trả một khoảng tiền phù hợp cho những ứng dụng mà họ cảm thấy cần thiết hơn so với trước đây.

Vấn đề cân đối giữa thu-cho, khoản vay nợ và kỳ hạn phải trả luôn là một vấn đề khó khăn với đa phần tầng lớp trong xã hội. Bởi vậy, một ứng dụng chạy trên thiết bị di động và giúp đỡ người sử dụng trong việc quản lý chi tiêu là rất cần thiết.

Một số phần mềm có tính năng tương tự đã được phát triển trên các nền tảng Android, IOS, ... như: Money lover, Sổ thu chi, Sổ thu chi misa, ..., tuy nhiên các phần mềm này còn nhiều bất lợi cho người dùng như: thu phí các tính năng nâng cao, không đồng bộ dữ liệu giữa các thiết bị, không hỗ trợ đầy đủ chức năng, ...

Xuất phát từ những nhận định đó, tôi đã chọn đề tài “Phần mềm quản lý chi tiêu cá nhân trên di động” nhằm hỗ trợ người dùng giải quyết các vấn đề trên.

Phần mềm tạm gọi với tên “**Spending Management**”

1.3. Đối tượng hướng đến

Với sự bùng nổ của Smartphone, nhóm đối tượng chủ yếu của ứng dụng là các tổ chức/cá nhân có và biết sử dụng smart phone ở mọi độ tuổi, cụ thể:

- Những cá nhân không có nhiều kinh nghiệm cũng như kiến thức về quản lý chi tiêu muốn tìm hiểu và trang bị thêm cho mình kiến thức.
- Những cá nhân/tổ chức quan tâm về quản lý chi tiêu, mong muốn hạn chế số tiền chi tiêu không hợp lý.
- Những cá nhân/tổ chức muốn có một công cụ giúp dễ dàng quản lý mọi nguồn tiền chi/thu của mình.
- Những cá nhân/tổ chức chưa hài lòng về các công cụ hỗ trợ quản lý chi tiêu cá nhân đang sử dụng.

1.4. Môi trường phát triển ứng dụng

- Hệ điều hành phát triển: Microsoft Windows
- Hệ điều hành cài đặt: Android

- Cơ sở dữ liệu đám mây: Firebase
- Công cụ phân tích thiết kế: figma.com
- Công cụ phân chia nhiệm vụ: Notion + Messenger
- Công cụ xây dựng ứng dụng: Android Studio, Visual Studio Code
- Ngôn ngữ sử dụng: Dart
- Framework: Flutter
- Công cụ quản lý source code: Github

1.5. Kết quả mong đợi

- Học hỏi cách phát triển một ứng dụng trên di động qua quá trình làm đề tài.
- Triển khai được một sản phẩm hoàn thiện, có ích cho người dùng.
- Hiểu được và áp dụng quy trình phát triển phần mềm.

1.6. Khảo sát hiện trạng

Thực tế hiện nay, Việt Nam đã có nhiều ứng dụng quản lý chi tiêu quen thuộc có thể kể đến như: Money Lover, Sổ thu chi, Spendee, MISA Money Keeper, . . . Ta cùng đi qua một số thông tin chung về các ứng dụng này.

Ứng dụng Chức năng	Money lover	Sổ thu chi	Sổ thu chi misa	Nhóm
Thêm, chỉnh sửa thu chi	X	X	X	Khả quan
Tra cứu thu chi	X	X	X	Khả quan
Báo cáo theo tuần , tháng,năm	X	X	X	Khả quan
Liên kết theo dõi tài khoản ngân hàng	X			Không khả quan
Sử dụng trên nhiều thiết bị	X		X	Khả quan
Nhắc nhở các khoản thu chi định ki	X	X	X	Khả quan

Chuyển đổi tiền tệ	X		X	Khả quan
Quét hóa đơn	X			Không khả quan
Lập hạn mức chi tiêu	X		X	Khả quan
Thiết lập số dư ban đầu	X	X	X	Khả quan
Số tiết kiệm	X	X	X	Không khả quan
Số nợ	X			Không khả quan
Xuất dữ liệu	X	X	X	Khả quan
Danh sách mua sắm, du lịch			X	Không khả quan
Tính lãi vay, thuế TNCN			X	Không khả quan

1.7. Quy trình thực hiện các công việc chính

Hiện nay có rất nhiều quy trình phát triển phần mềm khác nhau. Tuy nhiên theo yêu cầu của đề tài, nhóm đã sử dụng mô hình thác nước cải tiến. Mô hình trên bao gồm các trình tự: xác định yêu cầu, phân tích, thiết kế, cài đặt, kiểm thử, bảo trì. Trong đó, kết quả của giai đoạn trước là cơ sở đầu vào của giai đoạn sau. Vì vậy, nếu như có lỗi xảy ra, nhóm có thể quay lui để sửa lỗi và tối ưu phần mềm trong khi tiến độ hiện tại vẫn được duy trì.

Cụ thể các trình tự phát triển phần mềm của nhóm như sau:

- Xác định yêu cầu: Khảo sát yêu cầu người dùng, lập ra bảng các yêu cầu và quy định cụ thể cho phần mềm.
- Phân tích: Phân loại các yêu cầu và lập sơ đồ Use-case

- Thiết kế: Mô tả các thành phần của phần mềm một cách rõ ràng, gồm các bước:
 - Thiết kế hệ thống, kiến trúc, các đối tượng.
 - Thiết kế cơ sở dữ liệu.
 - Thiết kế giao diện.
 - Cài đặt: Dựa theo những thiết kế và phân tích, tiến hành xây dựng ứng dụng thực tế.
 - Kiểm thử: Chạy thực nghiệm và đánh giá, tìm và sửa lỗi.

2. Phân tích, thiết kế hệ thống

2.1. Xác định và mô hình hóa các yêu cầu phần mềm

2.1.1. Xác định yêu cầu

a. Một số yêu cầu phần mềm phải có

- Đăng ký, đăng nhập.
- Xác thực email khi đăng ký tài khoản.
- Người sử dụng có thể lưu lại các thông tin chi tiêu hàng ngày. Các thông tin lưu trữ cho một phần chi tiêu bao gồm: thời gian, lý do, mục chi tiêu, hình ảnh liên quan....
- Thêm, xem, chỉnh sửa, xóa chi tiêu.
- Có phần thống kê, đánh giá theo các khoảng thời gian nhất định.
- Các danh mục chi tiêu có thể được tạo ra bởi một người dùng một cách linh hoạt.
- Có chức năng tìm kiếm theo tên, ngày tháng, mô tả....
- Có chức năng thay đổi thông tin người dùng.
- Có chức năng quên mật khẩu, thay đổi mật khẩu.
- Có thể thay đổi ngôn ngữ.
- Đồng bộ dữ liệu tài khoản trên mọi thiết bị đăng nhập.

b. Ràng buộc logic ban đầu

- Người dùng mới sẽ nhập vào số dư ban đầu mỗi tháng khi tạo tài khoản
- Tự động khởi tạo số dư ban đầu tháng bằng số tiền người dùng đã nhập vào ban đầu.
- Mọi giao dịch trong hệ thống chỉ thực hiện đến đơn vị hàng triệu tỉ (tức 15 chữ số).
- Mọi loại hình chi tiêu hay thu nhập chỉ thực hiện với số tiền dương

c. Tính khả dụng

Tương thích với hệ điều hành Android 9 trở lên.

d. Tính ổn định

Hệ thống phải hoạt động liên tục 24/7.

e. Hiệu suất

Xét trong điều kiện mạng ổn định

STT	Nghiệp vụ	Tốc độ xử lý
1	Thêm, xóa, sửa chi tiêu	Không quá 10s
2	Tra cứu chi tiêu	Không quá 5s
3	Lập báo cáo chi tiêu	Không quá 15s
4	Thay đổi thông tin cá nhân	Không quá 15s
5	Xuất dữ liệu chi tiêu	Không quá 10s

f. Bảo mật

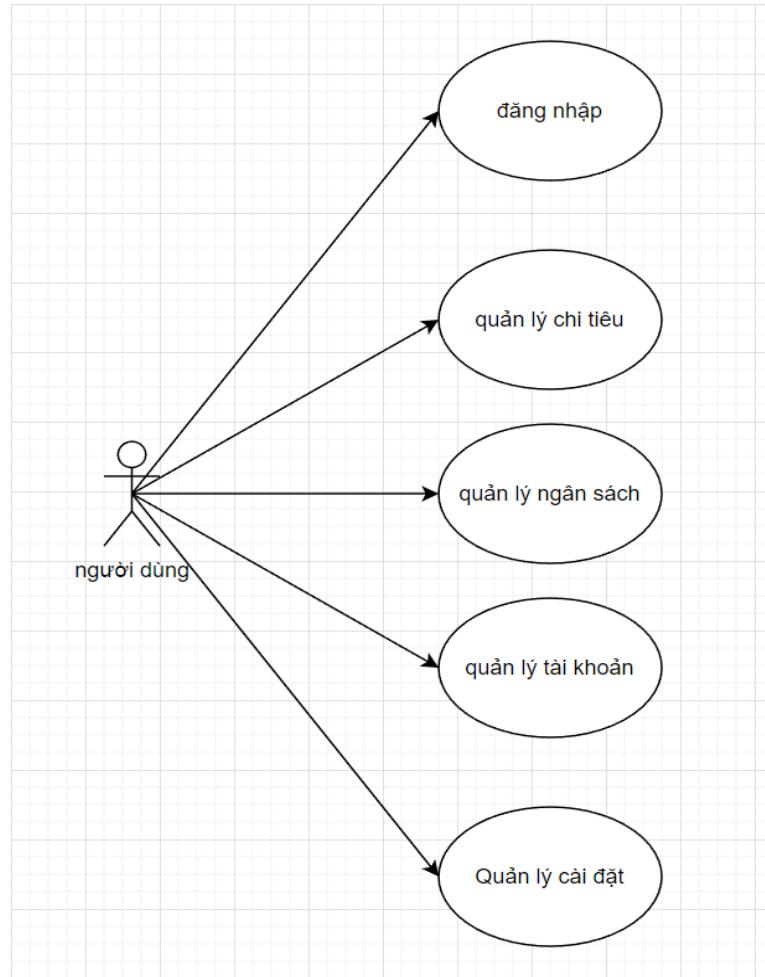
- Hệ thống phải có cơ chế toàn vẹn dữ liệu
- Đảm bảo người dùng không có quyền chỉnh sửa thông tin của người dùng khác hay truy cập vào những dữ liệu chi tiêu không thuộc quyền sở hữu của mình.

2.1.2. Mô hình hóa yêu cầu

a. Danh sách chức năng

- Đăng ký, đăng nhập
- Quên mật khẩu, thay đổi mật khẩu
- Thêm, sửa, xóa chi tiêu
- Xem thống kê, báo cáo theo các khoảng thời gian nhất định
- Thay đổi thông tin cá nhân
- Thay đổi ngôn ngữ
- Thay đổi số tiền quản lý
- Tìm kiếm chi tiêu

- Xuất dữ liệu chi tiêu
 - b. Lược đồ Use-case



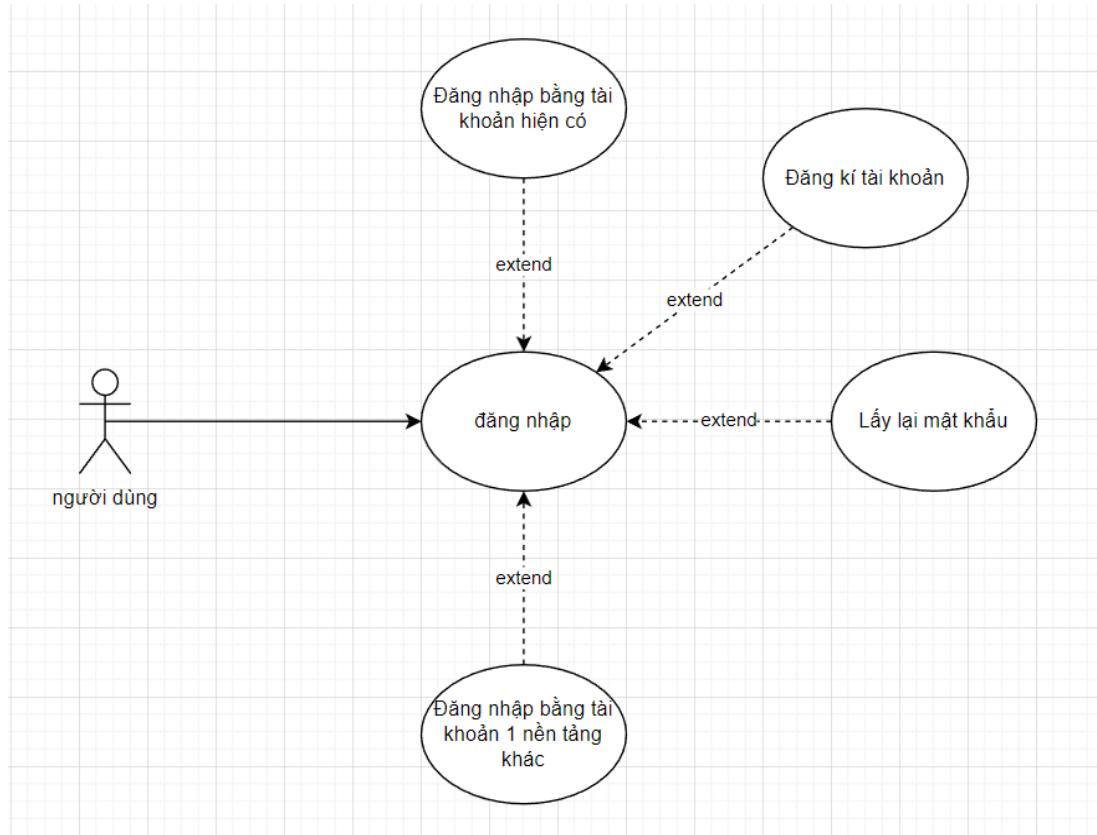
Lược đồ Use-case

Tên use case	Mô tả
Đăng nhập	Đăng nhập vào hệ thống và sử dụng chức năng quản lý tương ứng
Quản lý chi tiêu	Người dùng có thể thêm, xóa, sửa chi tiêu
Quản lý ngân sách	Người dùng có thể sửa số dư trong ví
Quản lý tài khoản	Người dùng có thể Sửa thông tin tài khoản

Quản lý cài đặt	Người dùng có thể chỉnh sửa app theo chức năng sẵn có (ngôn ngữ, giao diện...)
-----------------	--

Đặc tả use-case

Đặc tả use case đăng nhập



Đăng nhập bằng tài khoản hiện có

Tên use case	Đăng nhập bằng tài khoản hiện có
Tác nhân sử dụng	Người dùng
Mục đích	Đăng nhập vào hệ thống và sử dụng chức năng quản lý tương ứng
Mô tả	Nhập thông tin tài khoản và mật khẩu, hệ thống ghi nhận, xử lý và trả về kết quả

Tiền điều kiện	Người dùng có tài khoản và mật khẩu đăng nhập vào hệ thống
Kích hoạt	Mở phần mềm
Diễn biến chính	<ol style="list-style-type: none"> 1. Mở phần mềm 2. Màn hình đăng nhập được hiển thị 3. Nhập thông tin tài khoản và mật khẩu 4. Xử lý thông tin và trả về kết quả 5. Màn hình chính được hiển thị với các chức năng tương ứng
Ngoại lệ	Nếu kết quả kiểm tra thông tin đăng nhập là không hợp lệ, chương trình sẽ yêu cầu đăng nhập lại

Đăng ký tài khoản

Tên use case	Đăng ký tài khoản
Tác nhân sử dụng	Người dùng
Mục đích	Đăng ký tài khoản mới sau đó đăng nhập vào hệ thống và sử dụng chức năng quản lý tương ứng
Mô tả	Nhập thông tin tài khoản và mật khẩu, hệ thống ghi nhận, xử lý và trả về kết quả
Tiền điều kiện	Không có
Kích hoạt	Mở phần mềm
Diễn biến chính	<ol style="list-style-type: none"> 1. Mở phần mềm 2. Màn hình đăng nhập được hiển thị 3. Chọn chức năng đăng ký tài khoản

	4. Nhập thông tin tài khoản và mật khẩu 5. Xử lý thông tin và trả về kết quả 6. Màn hình chuyển sang nhập mã đã gửi về mail(tên tài khoản đã cung cấp) 7. Hệ thống nhận thông tin và trả kết quả 8. Màn hình chính được hiển thị với các chức năng tương ứng
Ngoại lệ	Nếu kết quả kiểm tra mật khẩu nhập lần thứ 2 không giống với mật khẩu nhập ban đầu , chương trình sẽ yêu cầu nhập lại mật khẩu

Lấy lại mật khẩu

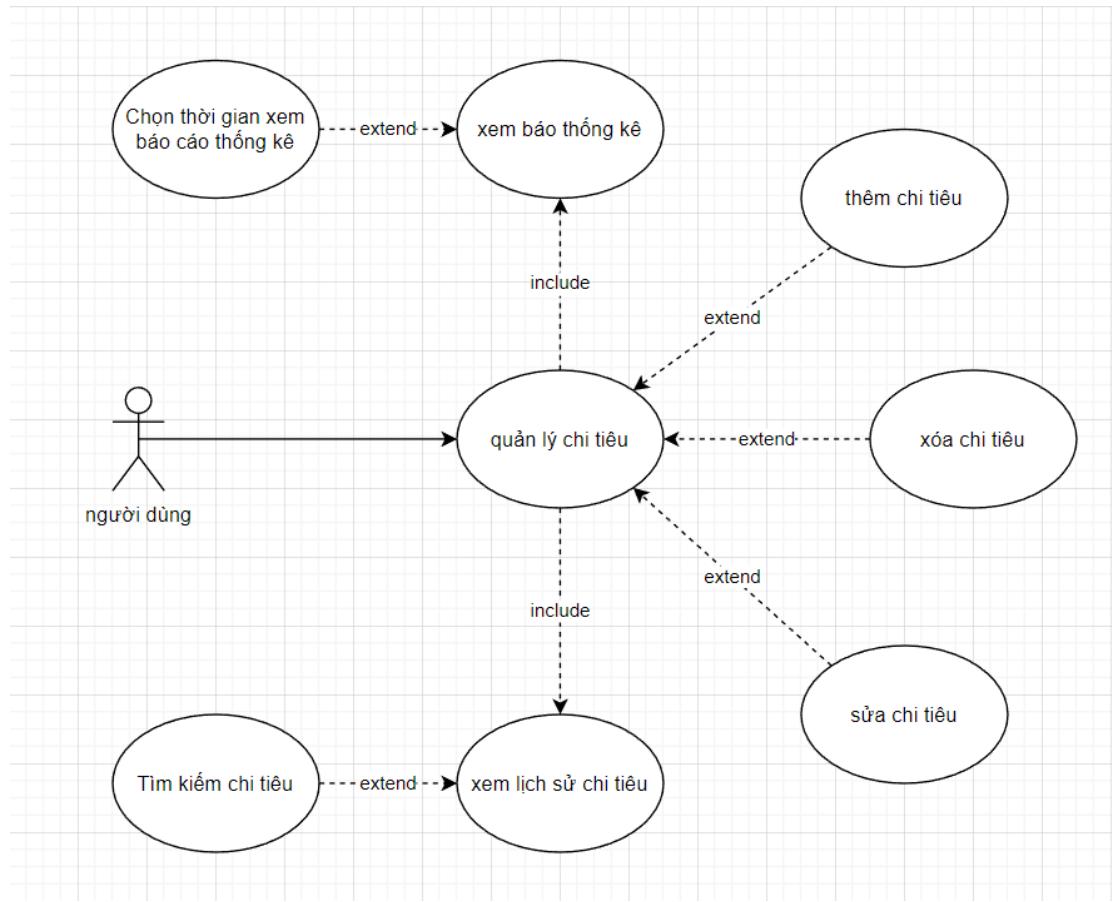
Tên use case	Lấy lại mật khẩu
Tác nhân sử dụng	Người dùng
Mục đích	Lấy lại mật khẩu sau đó đăng nhập vào hệ thống và sử dụng chức năng quản lý tương ứng
Mô tả	Nhập tên đăng nhập, hệ thống ghi nhận, xử lý và trả về kết quả
Tiền điều kiện	Không có
Kích hoạt	Mở phần mềm
Diễn biến chính	1. Mở phần mềm 2. Màn hình đăng nhập được hiển thị 3. Chọn chức năng quên mật khẩu 4. Nhập thông tin tài khoản 5. Xử lý thông tin và trả về kết quả 6. Màn hình chuyển sang nhập mã đã gửi về mail(tên tài khoản đã cung cấp) 7. Hệ thống nhận thông tin và trả kết quả

	8. Màn hình sẽ quay lại giao diện đăng nhập
Ngoại lệ	Nếu kết quả kiểm tra mã đã nhập là không chính xác hệ thống sẽ yêu cầu nhập lại mã

Đăng nhập bằng tài khoản 1 nền tảng khác

Tên use case	Đăng nhập bằng tài khoản 1 nền tảng khác
Tác nhân sử dụng	Người dùng
Mục đích	Đăng nhập vào hệ thống và sử dụng chức năng quản lý tương ứng
Mô tả	Chọn 1 nền tảng mà bạn có tài khoản để đăng nhập, hệ thống ghi nhận, xử lý và trả về kết quả
Tiền điều kiện	Người dùng có tài khoản ở 1 nền tảng khác
Kích hoạt	Mở phần mềm
Diễn biến chính	<ol style="list-style-type: none"> Mở phần mềm Màn hình đăng nhập được hiển thị Chọn 1 nền tảng mà bạn có tài khoản để đăng nhập Xử lý thông tin và trả về kết quả Màn hình chính được hiển thị với các chức năng tương ứng
Ngoại lệ	Nếu kết quả kiểm tra thông tin đăng nhập là không hợp lệ, chương trình sẽ yêu cầu đăng nhập lại

Đặc tả use case quản lý chi tiêu



Thêm chi tiêu

Tên use case	Thêm chi tiêu
Tác nhân sử dụng	Người dùng
Mục đích	Thêm thông tin chi tiêu mới vào hệ thống
Mô tả	Nhập thông tin về chi tiêu mới vào hệ thống dựa trên nhu cầu của người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng thêm chi tiêu ở màn hình
Diễn biến chính	1. Yêu cầu tạo chi tiêu mới

	<ol style="list-style-type: none"> 2. Hiện form nhập thông tin chi tiêu 3. Nhập thông tin chi tiêu 4. Xử lý thông tin và trả kết quả
Diễn biến phụ	
Ngoại lệ	Nếu thông tin về số tiền chi tiêu là không hợp lệ, chương trình sẽ yêu cầu nhập lại

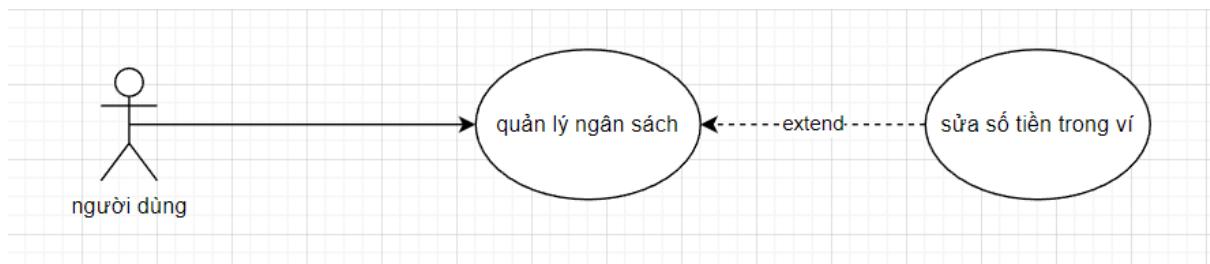
Sửa chi tiêu

Tên use case	Sửa chi tiêu
Tác nhân sử dụng	Người dùng
Mục đích	Sửa thông tin chi tiêu trong hệ thống
Mô tả	Nhập thông tin về chi tiêu mới vào hệ thống dựa trên nhu cầu của người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng thêm chi tiêu ở màn hình
Diễn biến chính	<ol style="list-style-type: none"> 1. Yêu cầu sửa chi tiêu 2. Hiện form sửa thông tin chi tiêu 3. Nhập lại thông tin chi tiêu 4. Xử lý thông tin và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Nếu thông tin về số tiền chi tiêu là không hợp lệ, chương trình sẽ yêu cầu nhập lại

Xóa chi tiêu

Tên use case	Xóa chi tiêu
Tác nhân sử dụng	Người dùng
Mục đích	Xóa thông tin chi tiêu có trong hệ thống
Mô tả	Tìm đến chi tiêu cần xóa và tiến hành xóa tất cả thông tin liên quan đến chi tiêu đó
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng trang chủ/ lịch ở màn hình
Diễn biến chính	<ol style="list-style-type: none"> 1. tìm kiếm chi tiêu cần xóa 2. Chọn chức năng xóa chi tiêu 3. Sử lý thông tin và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

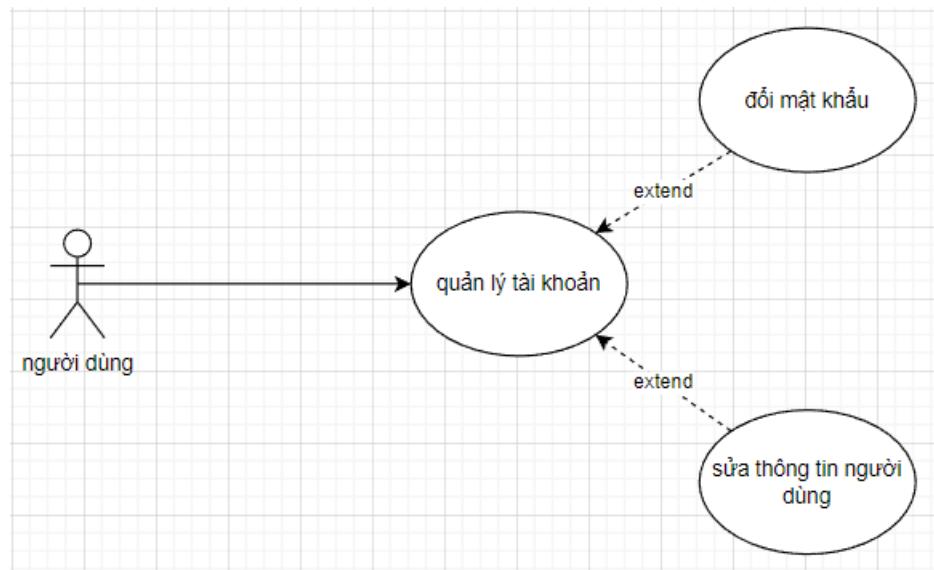
Đặc tả use case quản lý ví



Tên use case	Sửa số tiền trong ví
Tác nhân sử dụng	Người dùng

Mục đích	Chỉnh sửa số tiền (ví) trong hệ thống
Mô tả	Nhập số mới vào hệ thống dựa trên nhu cầu của người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng điều chỉnh số dư ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Yêu cầu sửa chi tiêu 2. Hiện form sửa thông tin số dư ví 3. Nhập lại số dư trong ví 4. Xử lý thông tin và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Nếu thông tin về số dư là không hợp lệ, chương trình sẽ yêu cầu nhập lại

Đặc tả use case quản lý tài khoản



Đổi mật khẩu

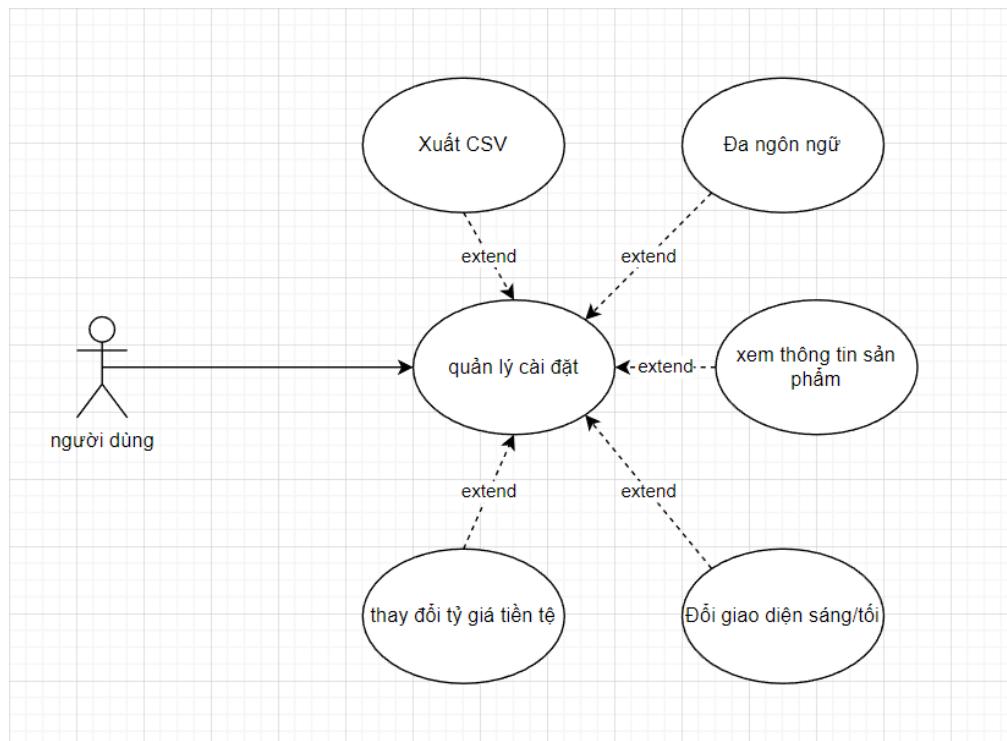
Tên use case	Sửa đổi mật khẩu
Tác nhân sử dụng	Người dùng
Mục đích	Chỉnh sửa mật khẩu người dùng trong hệ thống
Mô tả	Nhập mật khẩu mới vào hệ thống dựa trên nhu cầu của người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng đổi mật khẩu ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Yêu cầu đổi mật khẩu 2. Hiện form đổi mật khẩu 3. Nhập mật khẩu cũ 4. Nhập lại mật khẩu mới 5. Nhập lại mật khẩu mới lần 2 6. Xử lý thông tin và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

Sửa thông tin người dùng

Tên use case	Sửa thông tin người dùng
Tác nhân sử dụng	Người dùng
Mục đích	Chỉnh lại thông tin người dùng trong hệ thống

Mô tả	Nhập thông tin mới vào hệ thống dựa trên nhu cầu của người dùng
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng sửa thông tin người dùng ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Yêu cầu sửa thông tin 2. Hiện form sửa thông tin người dùng 3. Nhập lại thông tin người dùng 4. Xử lý thông tin và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

Đặc tả use case quản lý cài đặt



Xuất CSV

Tên use case	Xuất CSV
Tác nhân sử dụng	Người dùng
Mục đích	Xuất ra chi tiết các chi tiêu của người dùng
Mô tả	Chọn chức năng xuất CSV trong cài đặt
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng sửa thông tin người dùng ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Đăng nhập vào hệ thống 2. chọn chức năng cài đặt 3. chọn chức năng xuất CSV 4. hệ thống xử lý và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

Đa ngôn ngữ

Tên use case	Đa ngôn ngữ
Tác nhân sử dụng	Người dùng
Mục đích	chuyển đổi các ngôn ngữ sẵn có trong hệ thống
Mô tả	Chọn chức năng ngôn ngữ trong cài đặt
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng sửa thông tin người dùng ở màn hình cài đặt

Diễn biến chính	<ol style="list-style-type: none"> 1. Đăng nhập vào hệ thống 2. chọn chức năng cài đặt 3. chọn chức năng ngôn ngữ và chọn ngôn ngữ 4. hệ thống xử lý và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

Xem thông tin sản phẩm

Tên use case	Xem thông tin sản phẩm
Tác nhân sử dụng	Người dùng
Mục đích	xem thông tin nhà sáng tạo
Mô tả	Chọn chức năng ngôn ngữ trong cài đặt
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng Thông tin ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Đăng nhập vào hệ thống 2. chọn chức năng cài đặt 3. chọn chức năng thông tin 4. hệ thống xử lý và truyền đến màn hình cần xem
Diễn biến phụ	Không có
Ngoại lệ	Không có

Đổi giao diện

Tên use case	Đổi giao diện(sáng/tối)
--------------	-------------------------

Tác nhân sử dụng	Người dùng
Mục đích	chuyển đổi giao diện
Mô tả	Thay đổi giao diện trong cài đặt
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng chế độ tối ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 1. Đăng nhập vào hệ thống 2. chọn chức năng cài đặt 3. thay đổi giao diện bằng cách nhấp nút on/off 4. hệ thống xử lý và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

Thay đổi tỷ giá tiền tệ

Tên use case	Thay đổi tỷ giá tiền tệ
Tác nhân sử dụng	Người dùng
Mục đích	chuyển đổi các tỷ giá tiền sảnh có trong hệ thống
Mô tả	Chọn chức năng ngôn ngữ trong cài đặt
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Kích hoạt	Người dùng chọn chức năng tỷ giá tiền tệ ở màn hình cài đặt
Diễn biến chính	<ol style="list-style-type: none"> 5. Đăng nhập vào hệ thống 6. chọn chức năng cài đặt 7. chọn chức năng tỷ giá tiền tệ và chọn tỷ giá

	8. hệ thống xử lý và trả kết quả
Diễn biến phụ	Không có
Ngoại lệ	Không có

2.2. Thiết kế hệ thống

2.2.1. Kiến trúc hệ thống

Phần mềm sử dụng kiến trúc bloc:

Bloc giúp dễ dàng tách biệt phần UI và Business logic, giúp code nhanh, dễ test và dễ sử dụng. Mục đích của Pattern này là tách code business logic ra khỏi UI thay vì code gộp chung cả logic và UI vào cùng 1 file, để sau này spec mới có yêu cầu sửa code business logic hay sửa UI sẽ dễ dàng sửa hơn. Code business logic được tách ra đó người ta đặt tên là Bloc (Business Logic Component). Bên cạnh đó, nó còn giúp chúng ta quản lý state của 1 màn hình tốt hơn vì các state sẽ được quản ở Bloc tách biệt với UI. Chính vì vậy, mỗi màn hình trong app flutter chúng ta nên tạo ra 1 bloc để xử lý logic của màn hình đó và quản lý state của cả màn hình đó.

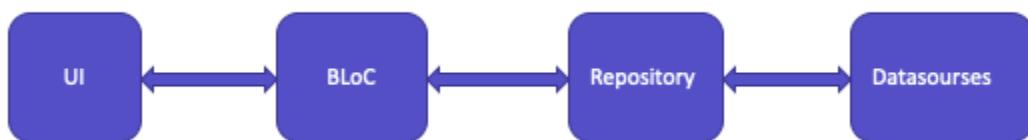
Bloc được coi như bộ não với nhiệm vụ chuyển đổi (convert) các đầu vào là các stream Event thành các stream State ở đầu ra.

Kiến trúc Bloc

Về kiến trúc Bloc trong Flutter, có hai dạng mà bạn thường gặp đó là :

- Xây dựng Bloc với RxDart
- Xây dựng BLoC với Event – State

Nhưng dù dùng kiểu nào đi nữa, thì cấu trúc cũng theo một mô hình như bên dưới:

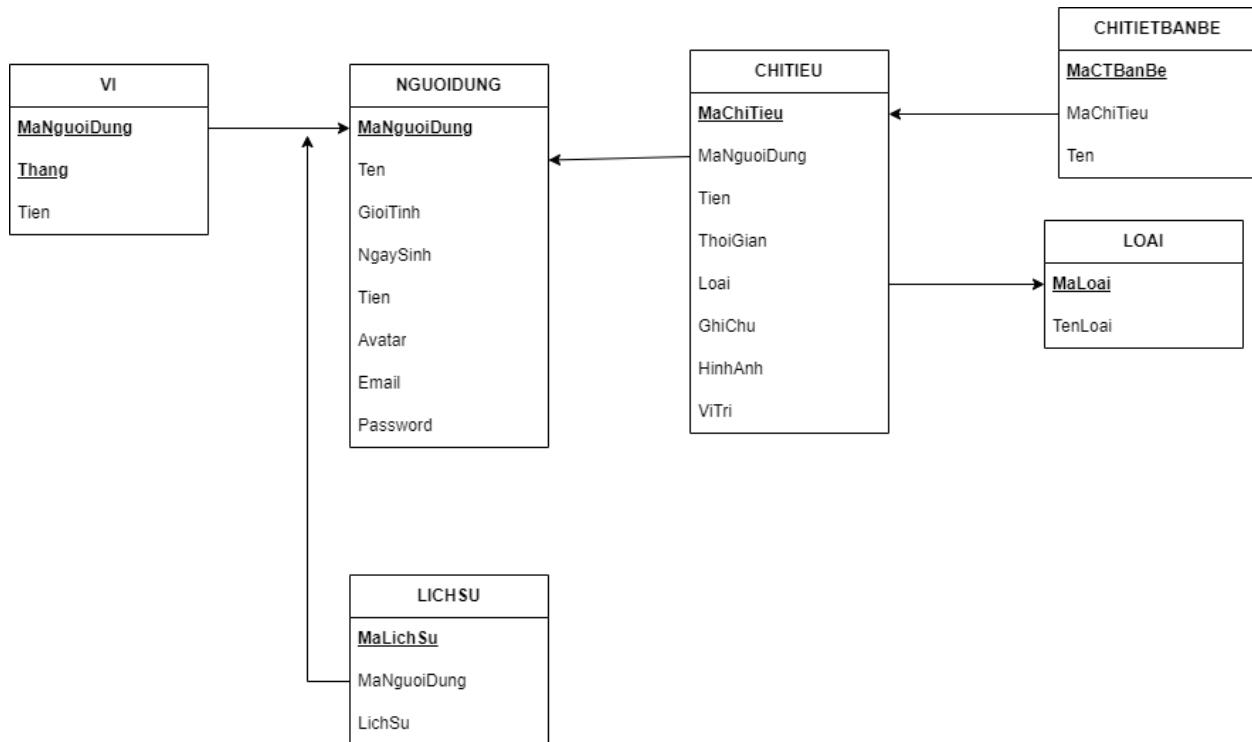


2.2.2. Mô tả các thành phần trong hệ thống

STT	Thành phần	Diễn giải
1	UI (User Interface)	Là những phần của Ứng dụng để hiển thị với người dùng.
2	BloC	Luôn lắng nghe các sự kiện pass qua nó, ví dụ luôn lắng nghe data pass qua stream.
3	Repository	Dùng để fetching data từ Data sources. Đầu ra của lớp này sẽ là đầu vào của khối Bloc, khi đó data sẽ được đặt trong các Stream
4	Data Sources	Là khối cung cấp data cho ứng dụng như network, sqflite, shared_preferences

2.3. Thiết kế dữ liệu

2.3.1. Thiết kế dữ liệu lưu trữ



Danh sách chi tiết các table trong dữ liệu

STT	Tên quan hệ	Ý nghĩa
1	NGUOIDUNG	Thông tin người dùng
2	CHITIEU	Thông tin chi tiêu của người dùng
3	CHITIETBANBE	Thông tin bạn bè trong chi tiêu
4	LOAI	Thông tin loại chi tiêu
5	VI	Thông tin số tiền hiện có mỗi tháng của người dùng
6	LICHSU	Thông tin lịch sử tìm kiếm chi tiêu của người dùng

NGUOIDUNG

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaNguoiDung	String	PK	Mã người dùng
2	Ten	String		Tên người dùng
3	GioiTinh	String		Giới tính
4	NgaySinh	DateTime		Ngày sinh
5	Tien	int		Số tiền
6	Avatar	String		Ảnh đại diện
7	Email	String		Email
8	Password	String		Mật khẩu

CHITIEU

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaChiTieu	String	PK	Mã chi tiêu
2	MaNguoiDung	String	FK	Mã người dùng
3	Tien	Int		Số tiền chi tiêu
4	ThoiGian	DateTime		Thời gian
5	Loai	String	FK	Loại chi tiêu
6	GhiChu	String		Ghi chú
7	HinhAnh	String		Hình ảnh

8	VịTri	String		Vị trí
---	-------	--------	--	--------

CHITIETBANBE

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaCTBanBe	String	PK	Mã chi tiết bạn bè
2	MaChiTieu	String	FK	Mã chi tiêu
3	Ten	String		Tên

LOAI

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaLoai	String	PK	Mã loại chi tiêu
2	TenLoai	String		Tên loại chi tiêu

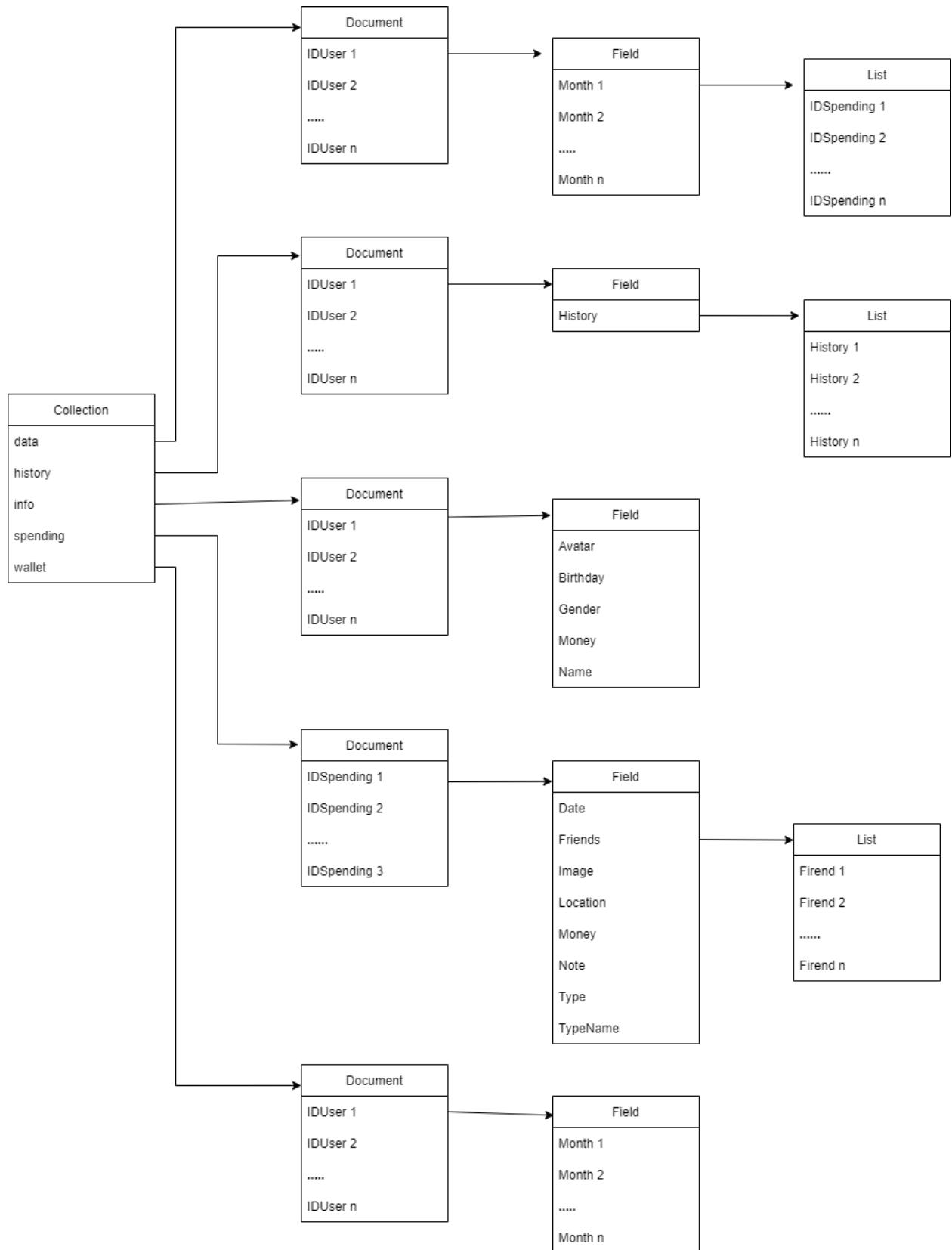
VI

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaNguoiDung	String	PK, FK	Mã người dùng
2	Thang	String	PK	Tháng
3	Tien	int		Số tiền

LICHSU

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	MaLichSu	String	PK	Mã lịch sử
2	MaNguoiDung	String	FK	Mã người dùng
3	LichSu	String		Lịch sử tìm kiếm

2.3.2. Thiết kế dữ liệu thực tế trên Firebase



Mô tả chi tiết dữ liệu trên Firebase

Collection “data”

Chứa các Document là id của người dùng! Mỗi Document sẽ là id của người dùng và có các trường chi tiết như sau:

STT	Tên thuộc tính	Kiểu	Ý nghĩa
1	Month 1	Array	Danh sách id chi tiêu tháng thứ 1
2	Month 2	Array	Danh sách id chi tiêu tháng thứ 2
...
n	Month n	Array	Danh sách id chi tiêu tháng thứ n

Collection “history”

Chứa các Document là id của người dùng! Mỗi Document sẽ là id của người dùng và có các trường chi tiết như sau:

STT	Tên thuộc tính	Kiểu	Ý nghĩa
1	history	Array	Danh sách tìm kiếm

Collection “info”

Chứa các Document là id của người dùng! Mỗi Document sẽ là id của người dùng và có các trường chi tiết như sau:

STT	Tên thuộc tính	Kiểu	Ý nghĩa
1	Avatar	String	Đường dẫn đến ảnh đại diện
2	Birthday	String	Ngày sinh
3	Gender	Bool	Giới tính
4	Money	Int	Tiền

5	Name	String	Tên
---	------	--------	-----

Collection “spending”

Chứa các Document là id của người dùng! Mỗi Document sẽ có các trường chi tiết như sau:

STT	Tên thuộc tính	Kiểu	Ý nghĩa
1	Date	TimeStamp	Thời gian chi tiêu
2	Friends	Array	Bạn bè
3	Image	String	Hình ảnh
4	Location	String	Vị trí
5	Money	Int	Số tiền
6	Note	String	Ghi chú
7	Type	Int	Loại ghi chú
8	TypeName	String	Tên loại do người dùng tạo

Collection “wallet”

Chứa các Document là id của người dùng! Mỗi Document sẽ là id của người dùng và có các trường chi tiết như sau:

STT	Tên thuộc tính	Kiểu	Ý nghĩa
1	Month 1	Int	Số tiền chi tiêu tháng thứ 1
2	Month 2	Int	Số tiền chi tiêu tháng thứ 2
...
n	Month n	Int	Số tiền chi tiêu tháng thứ n

Danh sách các lớp

STT	Tên lớp	Loại	Ý nghĩa
1	User	Lớp	Chứa thông tin người dùng
2	Spending	Lớp	Chứa thông tin chi tiêu

Mô tả chi tiết từng lớp

User

Danh sách thuộc tính

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	name	String	Không rỗng	Tên người dùng
2	birthday	DateTime		Ngày sinh
3	avatar	String		Ảnh đại diện
4	gender	Bool		Giới tính
5	money	Int	Số dương	Số tiền

Danh sách phương thức

STT	Tên phương thức	Kiểu	Ràng buộc	Ý nghĩa
1	toMap	Map		Chuyển lớp sang dạng Map
2	User.fromFirebase	User		Phân tích data từ Firebase và tạo dữ liệu cho User
3	copyWith	User		Sao chép và tạo một biến User mới dựa trên một số thuộc tính thay đổi

Spending

Danh sách thuộc tính

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa

1	id	String	Không rỗng	ID chi tiêu
2	money	Int	Khác 0	Số tiền chi tiêu
3	type	Int		Loại chi tiêu
4	note	String		Ghi chú
5	dateTime	DateTime	Không null	Thời gian
6	image	String		Đường dẫn hình ảnh
7	location	String		Vị trí
8	friends	List<String>		Bạn bè
9	typeName	String		Tên loại chi tiêu do người dùng thêm mới

Danh sách phương thức

STT	Tên phương thức	Kiểu	Ràng buộc	Ý nghĩa
1	toMap	Map		Chuyển lớp sang dạng Map
2	User.fromFirebase	User		Phân tích data từ Firebase và tạo dữ liệu cho Spending
3	copyWith	User		Sao chép và tạo một biến Spending mới dựa trên một số thuộc tính thay đổi

2.3.3. Danh sách các lớp đối tượng

Danh sách các lớp

STT	Tên lớp	Loại	Ý nghĩa

1	User	Lớp	Chứa thông tin người dùng
2	Spending	Lớp	Chứa thông tin chi tiêu

Mô tả chi tiết từng lớp

User

Danh sách thuộc tính

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa
1	name	String	Không rỗng	Tên người dùng
2	birthday	DateTime		Ngày sinh
3	avatar	String		Ảnh đại diện
4	gender	Bool		Giới tính
5	money	Int	Số dương	Số tiền

Danh sách phương thức

STT	Tên phương thức	Kiểu	Ràng buộc	Ý nghĩa
1	toMap	Map		Chuyển lớp sang dạng Map
2	User.fromFirebase	User		Phân tích data từ Firebase và tạo dữ liệu cho User
3	copyWith	User		Sao chép và tạo một biến User mới dựa trên một số thuộc tính thay đổi

Spending

Danh sách các thuộc tính

STT	Tên thuộc tính	Kiểu	Ràng buộc	Ý nghĩa

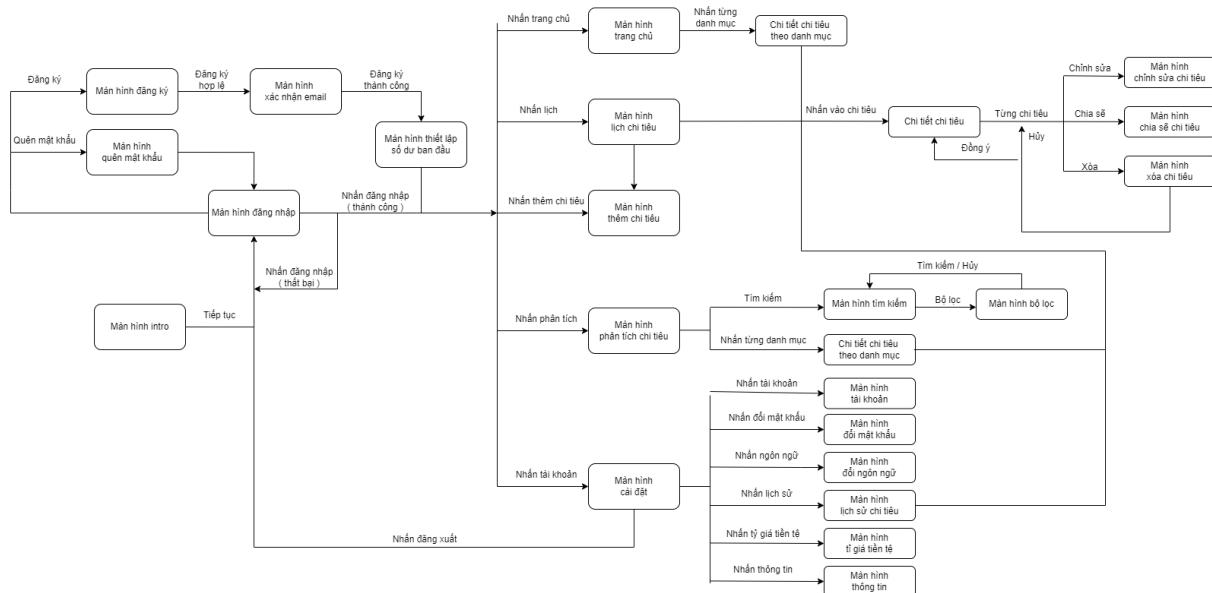
1	id	String	Không rỗng	ID chi tiêu
2	money	Int	Khác 0	Số tiền chi tiêu
3	type	Int		Loại chi tiêu
4	note	String		Ghi chú
5	dateTime	DateTime	Không null	Thời gian
6	image	String		Đường dẫn hình ảnh
7	location	String		Vị trí
8	friends	List<String>		Bạn bè
9	typeName	String		Tên loại chi tiêu do người dùng thêm mới

Danh sách phương thức

STT	Tên phương thức	Kiểu	Ràng buộc	Ý nghĩa
1	toMap	Map		Chuyển lớp sang dạng Map
2	User.fromFirebase	User		Phân tích data từ Firebase và tạo dữ liệu cho Spending
3	copyWith	User		Sao chép và tạo một biến Spending mới dựa trên một số thuộc tính thay đổi

2.4. Thiết kế giao diện

2.4.1. Sơ đồ liên kết màn hình



2.4.2. Danh sách các màn hình

- Màn hình giới thiệu
- Màn hình đăng nhập
- Màn hình đăng ký
- Màn hình quên mật khẩu
- Màn hình xác thực email
- Màn hình thiết lập số dư ban đầu
- Màn hình trang chủ
- Màn hình lịch chi tiêu
- Màn hình thêm chi tiêu
- Màn hình chỉnh sửa chi tiêu
- Màn hình xem chi tiêu
- Màn hình báo cáo
- Màn hình cài đặt
- Màn hình tìm kiếm
- Màn hình đổi thông tin cá nhân
- Màn hình đổi mật khẩu
- Màn hình thống kê chi tiêu

2.4.3. Mô tả các màn hình

2.4.3.1. Màn hình giới thiệu

- Giao diện

The screenshot displays a mobile application interface for financial management, likely a budgeting or expense tracking app. It includes several sections:

- Trang Chủ (Home):** Shows a summary of spending for the current month (Tháng này) with a total expenditure of -1.152.000 VND. Below this is a list of recent expenses categorized by type (e.g., Ăn uống, Di chuyển, Tiền nước).
- Lịch (Calendar):** Displays a monthly calendar for December 2022, highlighting specific dates with green boxes. Below the calendar is a list of transactions corresponding to those dates.
- Thêm Chi Tiêu (Add Expense):** A form for entering new expenses, including fields for amount (100.000 VND), date (02/12/2022), time (09:24), category (Ghi Chú), location (Vị Trí), and notes (Bạn bè).
- Tìm kiếm (Search):** Allows users to search for specific transactions using keywords.
- Thông kê (Statistics):** Provides a visual summary of spending and income for the period from November 28, 2022, to December 4, 2022. It shows a pie chart with categories like Chi Tiêu and Thu Nhập.
- Tài Khoản (Account):** Manages personal information and settings, including account details (Trần Ngọc Tiến), bank account (Tài Khoản), password (Đổi Mật Khẩu), language (Ngôn Ngữ), dark mode (Chế Độ Tối), history (Lịch sử), CSV export (Xuất CSV), and exchange rate (Tỷ Giá Tiền Tệ).

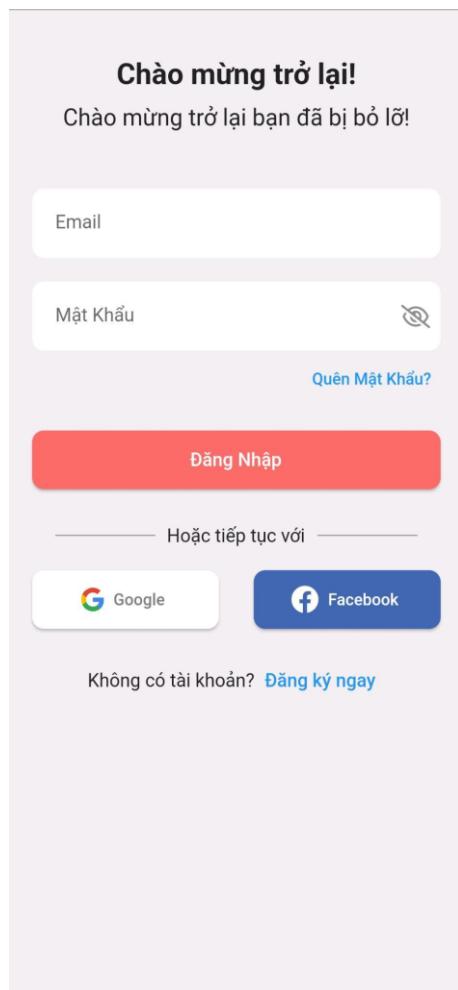
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
-----	-----------	---------	---------------	---------

1	Giới thiệu tiếp theo	Hiển thị thông tin giới thiệu ứng dụng tiếp theo	Khi người dùng nhấn vào button tiếp	
2	BỎ qua các thông tin giới thiệu	BỎ qua các thông tin giới thiệu và trực tiếp đi đến thông tin giới thiệu cuối cùng	Khi người dùng nhấn vào button bỏ qua	

2.4.3.2. Màn hình đăng nhập

a. Giao diện

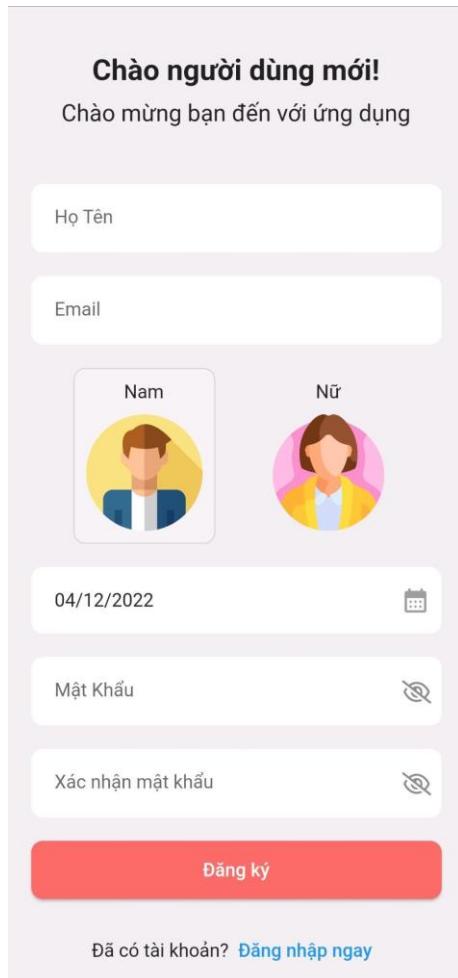


b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Đăng Nhập	Đăng nhập vào hệ thống bằng email + password	Khi người dùng nhập đầy đủ Email + password và nhấn vào button đăng nhập	
2	Đăng nhập bằng Google	Đăng nhập vào hệ thống bằng tài khoản Google	Khi người dùng nhấn vào button Google	
3	Đăng nhập bằng Facebook	Đăng nhập vào hệ thống bằng tài khoản Facebook	Khi người dùng nhấn vào button Facebook	
4	Quên mật khẩu	Lấy lại mật khẩu khi người dùng quên	Khi người dùng nhấn vào button quên mật khẩu	
5	Đến màn hình đăng ký	Đăng ký một tài khoản mới	Khi người dùng nhấn vào button Đăng ký ngay	

2.4.3.3. Màn hình đăng ký

a. Giao diện

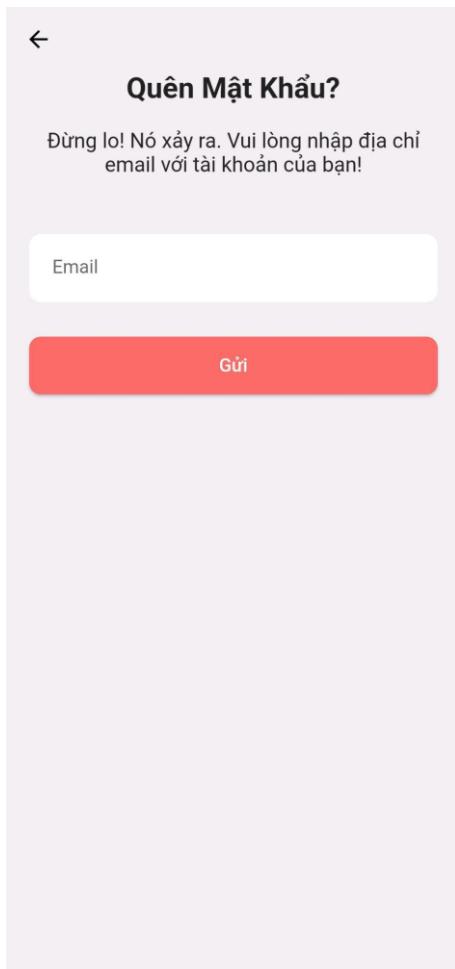


b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Đăng ký	Đăng ký tài khoản mới bằng Email + password mới trên hệ thống	Khi người dùng nhập đầy đủ các thông tin và nhấn vào button đăng ký	
2	Về màn hình đăng nhập	Về lại màn hình đăng nhập	Khi người dùng nhấn vào đăng nhập ngay	

2.4.3.4. Màn hình quên mật khẩu

a. Giao diện

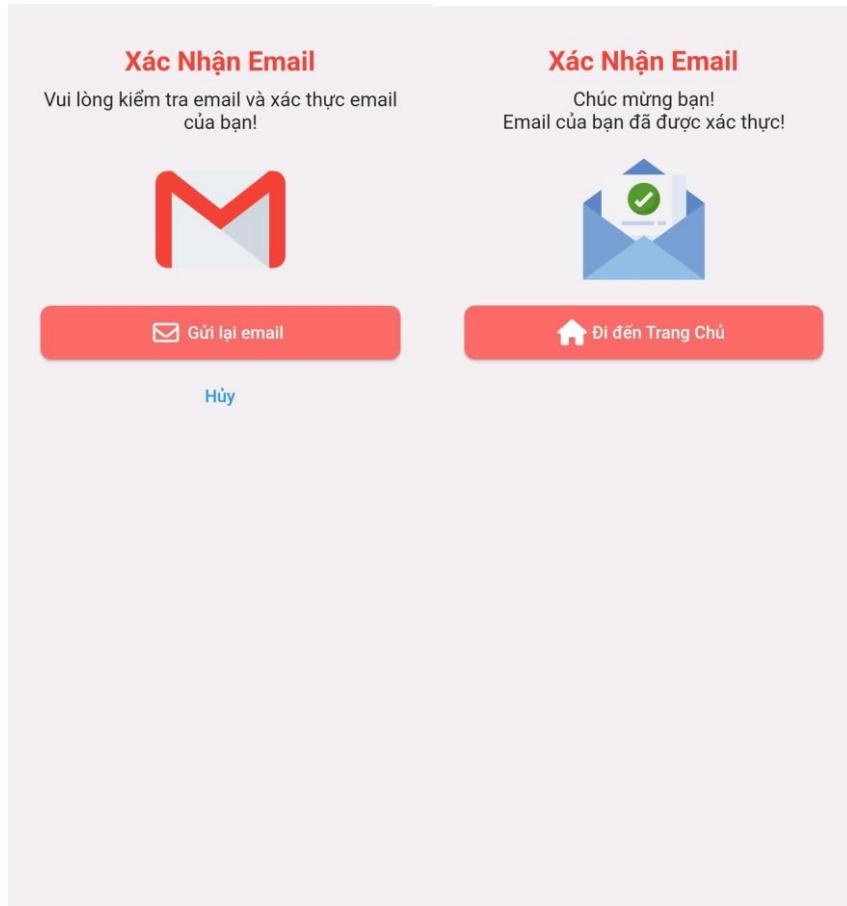


b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Gửi yêu cầu đặt lại mật khẩu	Gửi đi yêu cầu đặt lại mật khẩu lên hệ thống	Khi người dùng nhập vào email đúng và nhấn vào button Gửi	

2.4.3.5. Màn hình xác thực email

a. Giao diện

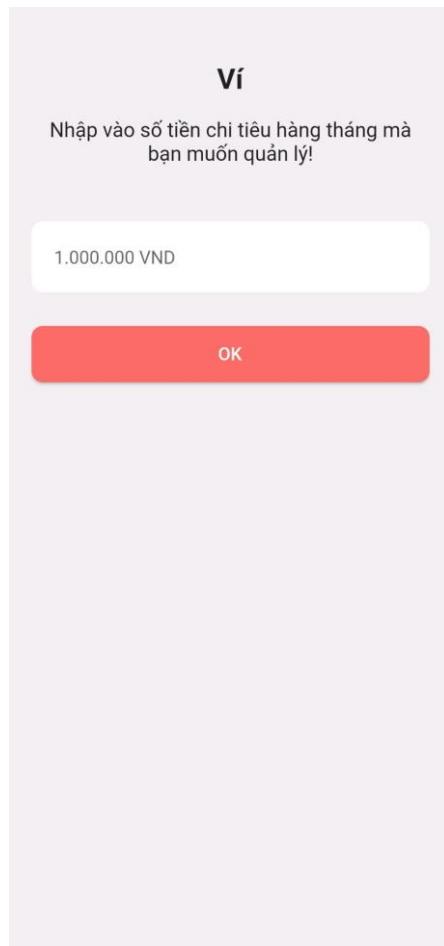


b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Gửi lại email xác thực	Hệ thống sẽ gửi một email xác thực đến tài khoản email của dùng	Nhấn vào button gửi lại email	
2	Đến trang chủ	Đưa người dùng đến trang chủ	Khi người dùng xác thực email và nhấn vào button đi đến trang chủ	

2.4.3.6. Màn hình thiết lập số dư ban đầu

a. Giao diện



b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Lưu lại số tiền hàng tháng của người dùng	Lưu lại số tiền hàng tháng của người dùng vào hệ thống	Khi người dùng nhập vào số tiền và nhấn button ok	

2.4.3.7. Màn hình trang chủ

a. Giao diện



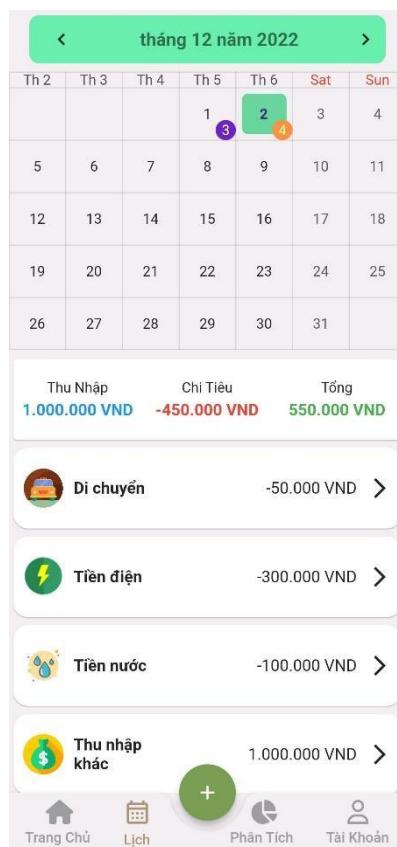
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Thanh cuộn tháng	Chọn để hiển thị danh sách chi tiêu tháng đó	Chọn tháng cần hiển thị trên màn hình	
2	Text lable số dư đầu số dư cuối	hiển thị số dư đầu số dư cuối và tổng số tiền đã chi tiêu trong tháng	mở page home/trang chủ trên màn hình	
3	Danh sách chi tiêu tháng	Hiển thị danh sách chi tiêu	không	

		của tháng đã chọn		
4	Chi tiết chi tiêu	xem chi tiết về chi tiêu mình đã chọn trong danh sách chi tiêu	chọn 1 chi tiêu bất kì trong danh sách chi tiêu	

2.4.3.8. Màn hình lịch chi tiêu

a. Giao diện



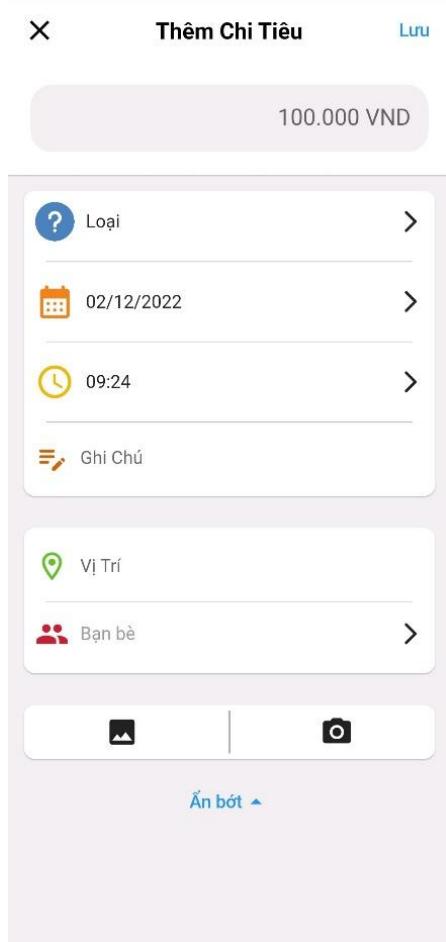
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Lịch	Chọn thời gian mong muốn, truy	Chọn ngày tháng năm cần xem chi tiêu	

		tìm chi tiêu theo ngày nhanh chóng		
2	Text lable thu nhập	Hiển thị thu nhập trong ngày	Mở trang lịch	
3	Text lable chi tiêu	Hiển thị chi tiêu trong ngày	Mở trang lịch	
4	Text lable tổng	Hiển thị tổng thu nhập và chi tiêu trong ngày	Mở trang lịch	
5	Danh sách chi tiêu trong ngày	Hiển thị danh sách chi tiêu trong ngày đã chọn	Chọn ngày mong muốn	
6	Chi tiết chi tiêu	xem chi tiết về chi tiêu mình đã chọn trong danh sách chi tiêu	chọn 1 chi tiêu bất kì trong danh sách chi tiêu	

2.4.3.9. Màn hình thêm chi tiêu

a. Giao diện



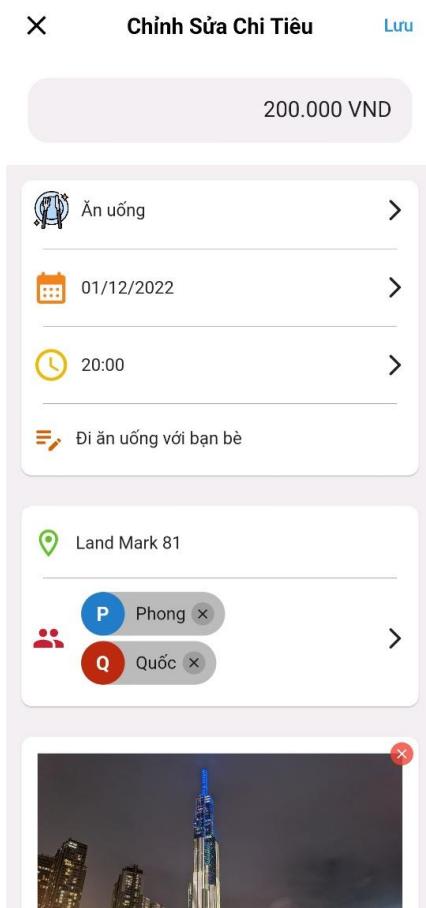
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Button thoát	Hủy thêm chi tiêu	Khi người dùng nhấn vào button X trên màn hình	
2	Texbutton Lưu	Lưu thông tin chi tiêu người dùng đã nhập	Khi người dùng nhấn vào texbutton Lưu trên màn hình	
3	Số tiền	Nhập số tiền chi tiêu	Nhấn vào textbox	Bắt buộc

4	Ngày tháng năm	Ngày tháng năm chi tiêu	Nhấn vào label ngày tháng năm	Mặc định là hiện tại
5	Giờ	Thời gian chi tiêu trong ngày	Nhấn vào label giờ	Mặc định là hiện tại
6	Ghi chú	Nhập ghi chú	Nhấn vào textbox	Không bắt buộc
7	Loại	Chọn loại chi tiêu	Nhấn vào label loại	Bắt buộc
8	Vị trí	Chọn vị trí chi tiêu	Nhấn vào label loại	Không bắt buộc
9	Bạn bè	Thêm bạn bè cùng chi tiêu	Nhấn vào label bạn bè	Không bắt buộc
10	Ảnh	Ảnh chụp chi tiêu	Nhấn vào label ảnh	Không bắt buộc

2.4.3.10. Màn hình chỉnh sửa chi tiêu

a. Giao diện



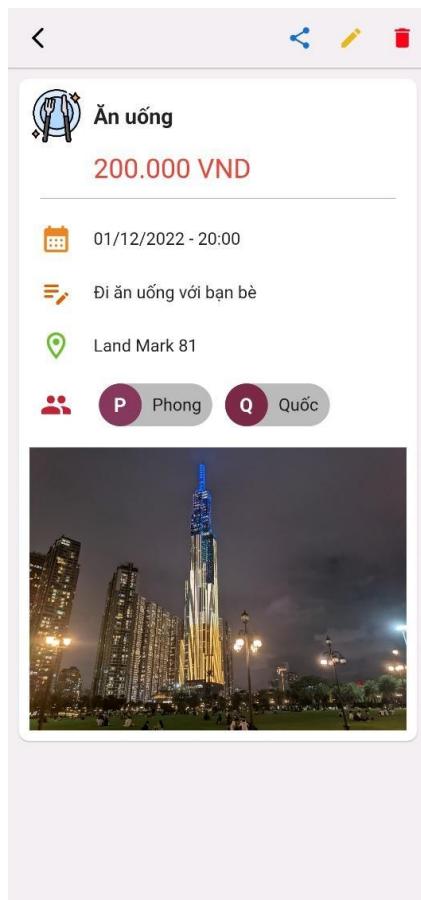
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
	Button thoát	hủy chỉnh sửa chi tiêu	Khi người dùng nhấn vào button X trên màn hình	
	Texbutton Lưu	lưu chi tiêu thông tin người dùng đã sửa trên màn hình	Khi người dùng nhấn vào texbutton Lưu trên màn hình	
	Sửa thông tin chi tiêu	Người dùng có thể sửa thông tin chi tiêu với thông tin	người dùng có thể sửa thông tin chi tiêu khi người dùng	

		hiện có trên màn hình	click vào thông tin tương ứng	
--	--	-----------------------	-------------------------------	--

2.4.3.11. Màn hình xem chi tiêu

a. Giao diện



b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Button quay lại	Quay lại màn hình trước đó	Khi người dùng nhấn vào button < trên màn hình	
2	Button chia sẻ	chia sẻ chi tiêu lên mạng xã hội	Khi người dùng nhấn vào button	

			chia sẻ trên màn hình	
3	Button xóa	xóa chi tiêu đang hiển thị trên màn hình	Khi người dùng nhấn vào button xóa trên màn hình	
4	Thông tin chi tiêu	Hiển thị chi tiết chi tiêu mà người dùng đã lưu	Khi người dùng nhấn vào chi tiêu ở page home trước đó	

2.4.3.12. Màn hình báo cáo

a. Giao diện

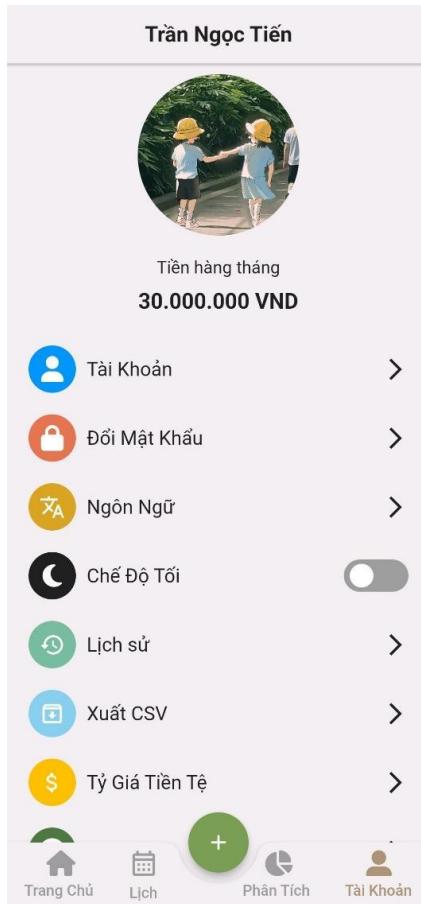


b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Báo cáo Tuần	Hiển thị báo cáo tuần	Khi người dùng nhấn vào nút tuần	
2	Báo cáo Tháng	Hiển thị báo cáo tháng	Khi người dùng nhấn vào nút tháng	
3	Báo cáo Năm	Hiển thị báo cáo năm	Khi người dùng nhấn vào nút năm	
4	Tìm kiếm	Chuyển sang trang tìm kiếm	Khi người dùng nhấn vào nút tìm kiếm	
5	Báo cáo Chi tiêu	Chuyển sang báo cáo cho chi tiêu	Khi người dùng nhấn vào nút chi tiêu	
6	Báo cáo Thu nhập	Chuyển sang báo cáo cho thu nhập	Khi người dùng nhấn vào nút thu nhập	
7	Hiển thị biểu đồ tròn	Chuyển sang biểu đồ tròn	Khi người dùng nhấn vào biểu tượng biểu đồ tròn	
8	Hiển thị biểu đồ cột	Chuyển sang biểu đồ cột	Khi người dùng nhấn vào biểu tượng biểu đồ cột	

2.4.3.13. Màn hình cài đặt

a. Giao diện



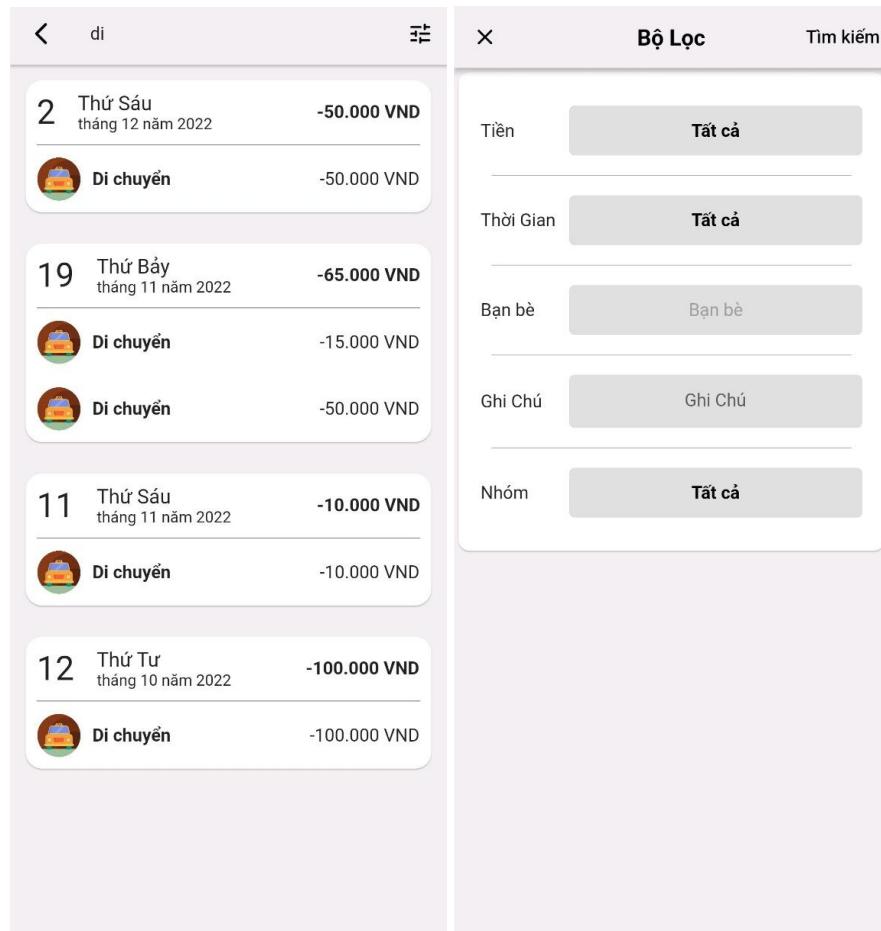
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Cài đặt tài khoản	Thay đổi thông tin cá nhân	Khi người dùng nhấn vào Button tài khoản trên màn hình	
2	Đổi mật khẩu	Thay đổi mật khẩu hiện tại	Khi người dùng nhấn vào Button đổi mật khẩu trên màn hình	

3	Ngôn ngữ	Chuyển đổi ngôn ngữ sẵn có trong ứng dụng	Khi người dùng nhấn vào Button ngôn ngữ trên màn hình	
4	Chế độ tối	Thay đổi giao diện màn hình	Khi người dùng nhấn vào Button chế độ tối trên màn hình	
5	Lịch sử	Xem lịch sử giao dịch từ mới nhất đến cũ nhất của người dùng	Khi người dùng nhấn vào Button lịch sử trên màn hình	
6	Xuất CSV	Xuất toàn bộ dữ liệu chi tiêu của người dùng thành file CSV	Khi người dùng nhấn vào Button xuất CSV trên màn hình	
7	Tỷ giá tiền tệ	Xem tỷ giá tiền tệ hiện tại	Khi người dùng nhấn vào Button Tỷ giá tiền tệ trên màn hình	
8	Thông tin	Xem thông tin ứng dụng	Khi người dùng nhấn vào Button thông tin trên màn hình	
9	Đăng xuất	Đăng xuất tài khoản ra khỏi ứng dụng	Khi người dùng nhấn vào Button Đăng xuất trên màn hình	

2.4.3.14. Màn hình tìm kiếm

a. Giao diện



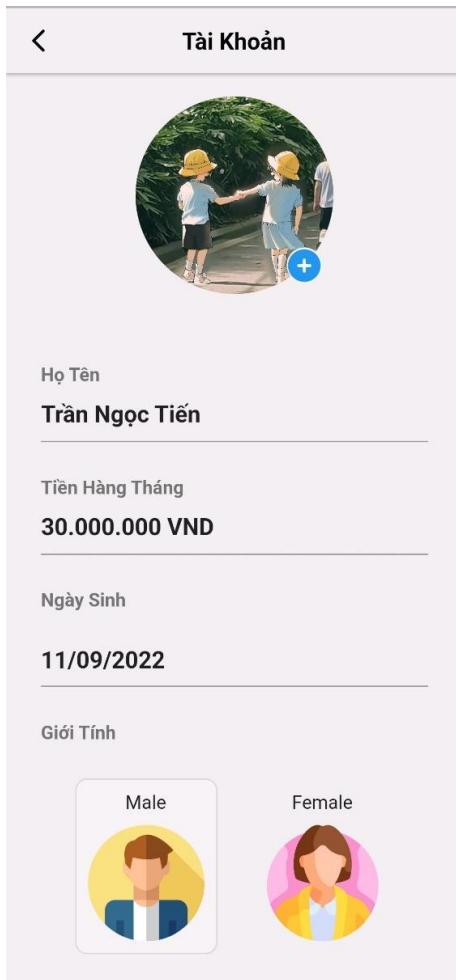
b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Tìm kiếm chi tiêu	Tìm kiếm chi tiêu theo yêu cầu của người dùng	Khi người dùng nhấn vào button kính lúp trên bàn phím	
2	Hiển thị bộ lọc	Hiển thị bộ lọc tìm kiếm lên màn hình	Khi người dùng nhấn vào button có biểu tượng bộ lọc	
3	Lọc tìm kiếm theo số tiền	Lọc các tìm kiếm theo số tiền mà người	Khi người dùng chọn những tùy	

		dùng chọn trong bộ lọc	chọn trong bộ lọc số tiền	
4	Lọc tìm kiếm theo thời gian	Lọc các tìm kiếm theo thời gian mà người dùng chọn trong bộ lọc	Khi người dùng chọn những tùy chọn trong bộ lọc thời gian	
5	Lọc tìm kiếm theo bạn bè	Lọc các tìm kiếm theo bạn bè mà người dùng chọn trong bộ lọc	Khi người dùng chọn những tùy chọn trong bộ lọc bạn bè	
6	Lọc tìm kiếm theo ghi chú	Lọc các tìm kiếm theo ghi chú mà người dùng chọn trong bộ lọc	Khi người dùng chọn những tùy chọn trong bộ lọc ghi chú	
7	Lọc tìm kiếm theo nhóm	Lọc các tìm kiếm theo nhóm mà người dùng chọn trong bộ lọc	Khi người dùng chọn những tùy chọn trong bộ lọc nhóm	
8	Áp dụng bộ lọc vào tìm kiếm	Áp dụng bộ lọc sau khi được người dùng tùy chỉnh vào trong tìm kiếm	Khi người dùng nhấn vào nút tìm kiếm trong bộ lọc	

2.4.3.15. Màn hình thay đổi thông tin cá nhân

a. Giao diện



b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Thêm ảnh người dùng	Người dùng có thể thêm ảnh cá nhân vào tài khoản	Khi người dùng nhấn vào button dấu + trên màn hình	
2	Họ và tên người dùng	Người dùng có thể thêm tên hiển thị trong tài khoản	Người dùng click vào thanh textbox họ tên để sửa	
3	Số tiền hàng tháng	Người dùng có thể sửa tổng số	Người dùng click vào thanh	

		tiền chi tiêu hàng tháng	textbox tiền hàng tháng sửa	
4	Ngày sinh	Người dùng có thể sửa ngày sinh của mình trong tài khoản	Người dùng click vào thanh textbox ngày sinh để sửa	
5	Giới tính	Người dùng có thể sửa giới tính của mình trong tài khoản	Khi người dùng chọn button male/female	

2.4.3.16. Màn hình đổi mật khẩu

a. Giao diện

The image shows a mobile application interface for changing a password. It consists of two screens:

- Screen 1: Bạn muốn đổi mật khẩu?**
Title: Bạn muốn đổi mật khẩu?
Text: Vui lòng nhập mật khẩu hiện tại của bạn
Fields: Password (with eye icon) and Xác nhận mật khẩu (with eye icon).
Buttons: A red "Gửi" (Send) button.
- Screen 2: Nhập mật khẩu mới**
Title: Nhập mật khẩu mới
Text: Vui lòng nhập vào mật khẩu mới của bạn
Fields: Mật Khẩu (with eye icon) and Xác nhận mật khẩu (with eye icon).
Buttons: A red "Gửi" (Send) button.

b. Mô tả các đối tượng trên màn hình

STT	Tên xử lý	Ý nghĩa	Điều kiện gọi	Ghi chú
1	Textbox Nhập mật khẩu cũ	Nhập mật khẩu cũ để có thể sang trang tiếp theo để đổi mật khẩu	Người dùng click và textbox trên màn hình để nhập mật khẩu cũ	
2	Button gửi	chuyển trang tiếp khi người dùng nhập đúng mật khẩu cũ	Khi người dùng click vào button gửi trên màn hình	
3	Textbox nhập mật khẩu mới	Nhập mật khẩu mới của người dùng	Người dùng click và textbox mật khẩu để nhập mật khẩu mới	
4	Textbox nhập lại mật khẩu mới	Nhập lại mật khẩu mới của người dùng	Người dùng click và textbox nhập lại mật khẩu để nhập lại mật khẩu mới	
5	Button gửi	kiểm tra và nhận thay đổi mật khẩu của người dùng	Khi người dùng click vào button gửi trên màn hình	

3. Cài đặt và thử nghiệm

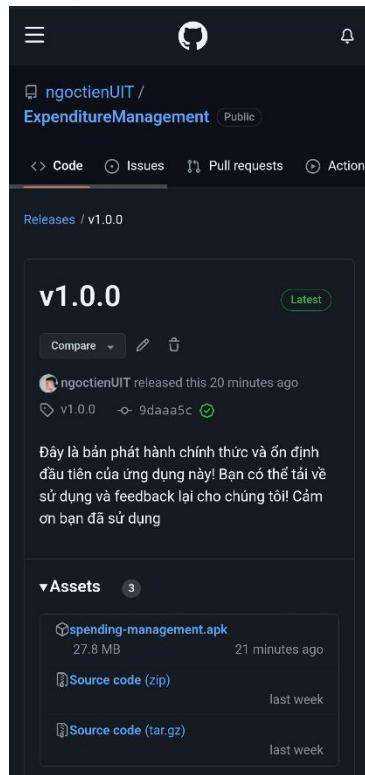
Source code: <https://github.com/ngoctienUIT/ExpenditureManagement>

STT	Chức năng	Mức độ hoàn thành	Ghi chú

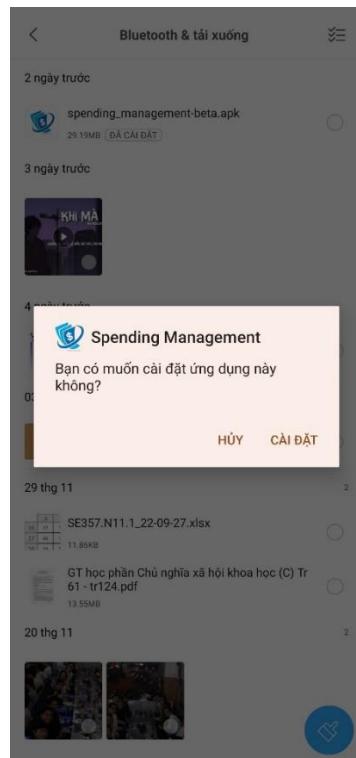
1	Đăng ký	100%	Đăng ký tài khoản mới để truy cập vào hệ thống
2	Đăng nhập	100%	Đăng nhập tài khoản vào hệ thống để quản lý chi tiêu
3	Đổi mật khẩu	100%	Thay đổi mật khẩu tài khoản của người dùng
4	Quên mật khẩu	100%	Đặt lại mật khẩu người dùng sau khi người dùng xác thực tài khoản
5	Thêm, sửa, xóa chi tiêu	100%	Thêm, xóa, sửa chi tiêu
6	Thông kê, báo cáo theo các khoảng thời gian nhất định	100%	Lập báo cáo theo tuần, tháng năm theo 2 loại biểu đồ cột và tròn
7	Thay đổi thông tin cá nhân	100%	Thay đổi các thông tin cá nhân của người dùng
8	Thay đổi ngôn ngữ	100%	Thay đổi ngôn ngữ ứng dụng (Anh – Việt)
9	Tìm kiếm chi tiêu	100%	Tra cứu chi tiêu dựa trên bộ lọc tra cứu có sẵn
10	Xuất dữ liệu chi tiêu	100%	Xuất dữ liệu chi tiêu ra file CSV

4. Hướng dẫn cài đặt phần mềm

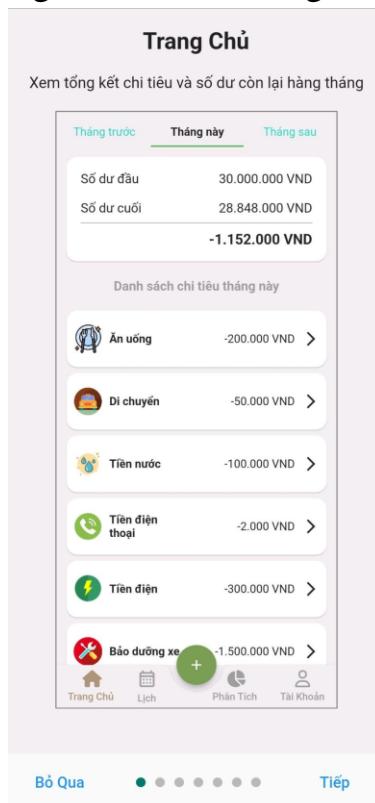
Bước 1: các bạn tải file cài đặt [tại đây](#)



Bước 2: Truy cập vào quản lý file và tìm kiếm đến file apk mà bạn vừa tải về và mở lên để tiến hành cài đặt



Bước 3: Sau khi cài đặt xong tiến hành mở ứng dụng lên để sử dụng



5. Hướng dẫn sử dụng phần mềm

5.1. Bắt đầu với hệ thống

Sau khi đọc tập tài liệu này, bạn có khả năng:

- Truy cập vào hệ thống;
- Đăng nhập hoặc đăng xuất khỏi hệ thống;
- Truy cập những tính năng của hệ thống;

5.2. Đăng nhập, đăng ký và đăng xuất khỏi hệ thống

Khi bạn thấy màn hình đăng nhập của ứng dụng, bạn hãy nhập **tên đăng nhập (email)** và **mật khẩu (password)** để đăng nhập vào ứng dụng.

Nếu đăng nhập thất bại, bạn sẽ thấy thông báo tương ứng từ hệ thống. Bạn hãy thử lại.

Tại màn hình đăng nhập bạn nhấn vào **Đăng ký ngay** để di chuyển đến màn hình đăng ký. Bạn cần điền đầy đủ thông tin cần thiết và nhấn **Đăng ký** để tiến hành

tạo tài khoản trong hệ thống. Sau khi đăng ký thành công bạn vui lòng kiểm tra email và xác thực tài khoản để có thể truy cập vào hệ thống.

Nếu đăng ký thất bại, bạn sẽ thấy thông báo tương ứng từ hệ thống. Bạn hãy thử đăng ký lại.

Nếu bạn muốn đổi tài khoản quản lý bạn có thể vào **Tài khoản** và chọn **Đăng xuất**

5.3. Quên mật khẩu, Đổi mật khẩu

Hệ thống có cung cấp chức năng **lấy lại mật khẩu** cho người dùng khi người dùng **quên mật khẩu**. Ở màn hình đăng nhập nhấn vào nút **quên mật khẩu**, màn hình quên mật khẩu sẽ hiển thị. Ở đây người dùng cần nhập vào địa chỉ email cần lấy lại mật khẩu. Sau khi nhập xong 1 email sẽ được gửi đến địa chỉ email vừa nhập. Người dùng tiến hành kiểm tra email và đổi mật khẩu.

Để bảo mật tài khoản thì hệ thống có cung cấp chức năng đổi mật khẩu. Để đổi mật khẩu, người dùng cần truy cập vào hệ thống và chọn **tài khoản**, chọn **đổi mật khẩu**. Sau đó màn hình đổi mật khẩu sẽ hiển thị. Người dùng nhập mật khẩu hiện tại. Nếu mật khẩu chính xác sẽ được chuyển đến màn hình nhập mật khẩu mới. Người dùng nhập mật khẩu mới và xác nhận đổi mật khẩu.

5.4. Chuyển tiếp giữa các thành phần bên trong ứng dụng

Khi vừa truy cập vào ứng dụng bạn sẽ thấy có màn hình chính và phía cuối màn hình sẽ có một thanh bottom tab. Bottom tab này dùng để di chuyển giữa các tính năng bên trong ứng dụng.

Trang chủ là nơi sẽ hiển thị số dư hàng tháng của bạn. Bạn có thể xem chi tiết mỗi tháng bằng cách chọn các tháng trên thanh tab bar phí trên.

Trang lịch là nơi bạn có thể xem được chi tiết chi tiêu từng ngày của bạn bằng cách chọn ngày mà bạn muốn xem trên lịch

Thêm chi tiêu là nơi bạn có thể thêm vào chi tiêu của mình

Báo cáo là nơi bạn có thể thống kê chi tiêu theo tuần, tháng, năm của mình theo biểu đồ cột hoặc tròn.

Tài khoản là nơi bạn có thể quản lý thông tin cá nhân và các cài đặt khác của ứng dụng

5.5. Thêm chi tiêu

Trong màn hình chính bạn nhấn vào biểu tượng dấu +. Màn hình thêm chi tiêu sẽ hiển thị. Bạn tiến hành thêm những thông tin cần thiết vào bên trong và tiến hành lưu chi tiêu. Chi tiêu sẽ được lưu trên hệ thống.

5.6. Sửa chi tiêu

Chọn chi tiêu màn bạn muốn sửa. Tiếp theo chọn biểu tượng cây bút để tiến hành sửa chi tiêu. Nhập vào nội dung mà bạn muốn thay đổi sau đó nhấn **Lưu** để lưu lại thay đổi.

5.7. Thay đổi thông tin cá nhân

Trong màn hình chính bạn chọn chức năng **Tài khoản**. Sau đó chọn **tài khoản** và tiến hành thay đổi thông tin mà bạn muốn.

5.8. Thay đổi ngôn ngữ ứng dụng

Trong màn hình chính bạn chọn chức năng **Tài khoản**. Sau đó chọn **Ngôn ngữ** và tiến hành thay đổi ngôn ngữ mà bạn muốn.

5.9. Tìm kiếm chi tiêu

Trong màn hình chính bạn chọn chức năng **Báo cáo**. Sau đó chọn biểu tượng **kính lúp** và tiến hành nhập vào chi tiết mà bạn muốn tìm kiếm. Có thể chọn bộ lọc mà tùy chọn những thông tin cần thiết mà bạn muốn lọc.

5.10. Xuất CSV

Trong màn hình chính bạn chọn chức năng **Tài khoản**. Sau đó chọn **Xuất CSV** và đợi vài giây để hệ thống kết xuất và tạo file báo cáo.

TÀI LIỆU THAM KHẢO

Em có dùng các tài liệu tham khảo từ trang chủ Flutter. Bên cạnh đó em còn tham khảo nhiều chủ đề thảo luận, ví dụ minh họa về Dart, Flutter, Firebase trên các trang web như Stackoverflow, Viblo, Github, ...

Flutter Document: <https://flutter.dev/docs>

StackOverFlow: <https://stackoverflow.com/>

Youtube: <https://youtube.com/>

Firebase Document: <https://firebase.google.com/docs/android/setup>

Firebase: <https://firebase.google.com/docs>

FlutterFire Document: <https://firebase.flutter.dev/docs/overview/>

Github Flutter: <https://github.com/flutter/samples>

Dart Packages: <https://pub.dev/>

Medium: <https://medium.com/>

Viblo: <https://viblo.asia/>