# Ảnh có chứa ảnh chụp màn hình Mô tả được tạo tự độngSetup flutter in windown

# Setup flutter develop in Android studio

Thêm flutter vào android studio

<https://flutter.dev/docs/get-started/editor>

Cài đặt android SDK platform – tool

# Tổng quan về dart language

**Dart** là một ngôn ngữ lập trình web do Google phát triển.

# Tổng quan về lập trình flutter

Là một ngôn ngữ lập trình đa nền tảng. Cross-platform application development frameworks

Chứ không phải là hybrib app

(Hybrid App là ứng dụng kết hợp những ưu điểm của cả Mobile Web App và Native App. Theo đó, các phần cơ bản của ứng dụng vẫn viết bằng ngôn ngữ web, nhưng được đặt trong native container, nên vẫn có thể dưa lên kho tải.)

Và được biết trên nền ngôn ngữ dart.

## Cấu tạo của flutter

### Tầng Application

Tầng Application của Flutter cung cấp kiến trúc cho một ứng dụng viết bằng Flutter

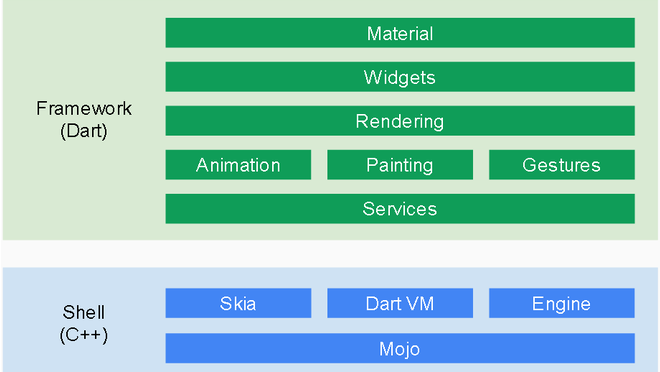
### Tầng shell

Tầng Shell cung cấp khả năng giao tiếp giữa ứng dụng viết bằng Flutter và lớp Platform

Flutter được viết chia làm hai tầng. Tầng ở trên sử dụng ngôn ngữ Dart cung cấp các đoạn mã xây dựng lên một ứng dụng Flutter. Các đoạn mã này cung cấp phương tiện để có thể thay đổi và chỉnh sửa chúng. Từ đó giúp ứng dụng của lập trình viên có thể được tùy chỉnh theo mong muốn. Tầng Application này giúp lập trình viên thay đổi mã nguồn ứng dụng ở thời điểm compile time. Tầng thứ hai của Flutter nằm ở sâu bên dưới và được viết bằng C++. Tầng Shell này chứa các công tụ trợ giúp ứng dụng Flutter trong quá trình chạy. Nổi bật cần lưu ý ở tầng này là máy ảo Dart VM. Khái niệm máy ảo là khái niệm về một ứng dụng chạy song song với mã nguồn chính như một phần của ứng dụng. Máy ảo Dart VM có ba nhiệm vụ chính bao gồm.

* Làm ứng dụng trung gian giữa mã nguồn được viết bởi Dart và thiết bị phần cứng (hoặc phần mềm nằm ngoài ứng dụng).
* Thông dịch các đoạn mã Dart theo phương thức JIT (Just in time - mã nguồn chỉ được thông dịch khi được gọi đến) hoặc AOT (Ahead of Time - Mã)[2](https://kipalog.com/posts/Chapter-1----Ngup-lan-voi-Flutter--Dive-to-Flutter#fn2)
* Thực thi các đoạn mã đã được thông dịch hoặc biên dịch cũng như cung cấp các runtime system bao gồm garbage collector, một vài các thư viện cần có của ngôn ngữ.

Máy ảo Dart VM chịu trách nhiệm lớn nhất trong việc quản lý các runtime system, hỗ trợ debugging hoặc hot reload cho các ứng dụng viết bằng Flutter



## So sánh với các Cross-platform Framework khác.

Một ứng dụng để chạy được trên các thiết bị thông thường sẽ tạm quan tâm tới hai thành phần. Thành phần gần với lập trình viên hơn là lớp Application. Lớp Application chứa các đoạn mã được viết bởi lập trình viên và các công cụ giúp ứng dụng hoạt động và quản lý hoạt động của ứng dụng. Thành phần được quan tâm thứ hai là lớp Platform ở đây dùng để chỉ các hệ điều hành hoặc các ứng dụng trung gian giữa ứng dụng của lập trình viên và thiết bị. Trong phạm vi lập trình ứng dụng mobile, Lớp Platform được chia làm ba thành phần bao gồm.

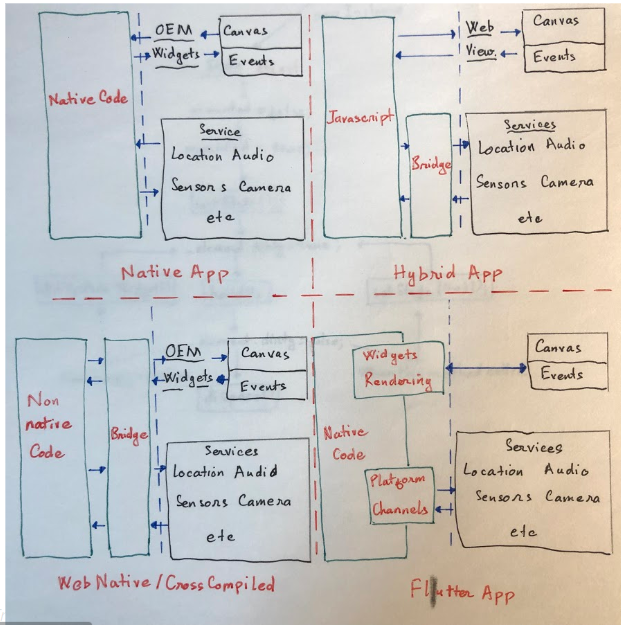
* **Services**: Bao gồm các ứng dụng được viết và cài đặt mặc định trong thiết bị có trách nhiệm quản lý giao tiếp giữa các ứng dụng trong thiết bị với nhau, cung cấp thông tin của thiết bị cho các ứng dụng hoặc thực thi các yêu cầu của ứng dụng để điều khiển thiết bị (Location, Camera, Sensors...).
* **Canvas/Events**: Là các ứng dụng giúp hiển thị giao diện lên màn hình hoặc các thiết bị trình chiếu đồng thời nhận lại các sự kiện xuất hiện trên các giao diện này (ví dụ như tương tác của người dùng hoặc các sự kiện từ các animation...).
* **OEM Widgets**: Thành phần phụ được cấu thành do sự thống nhất giữa các vendor (nhà cung cấp thiết bị (HTC), đơn vị phát hành thiết bị (NTT Docomo)) nhằm thống nhất một quy chuẩn chung cho các UI (thành phần giao diện) được sử dụng trong các Platform. Quy chuẩn được thống nhất này tạo ra một vài giới hạn cho việc viết lên UI sẽ được nói kĩ ở phần sau.

Thành phần sẽ được so sánh trong phạm vi bài viết này là kiến trúc của lớp Application cũng như cách mà lớp Application giao tiếp với lớp Platform. Giống như đã đề cập ở phần trước. Tầng Application của Flutter cung cấp kiến trúc cho một ứng dụng viết bằng Flutter và tầng Shell cung cấp khả năng giao tiếp giữa ứng dụng viết bằng Flutter và lớp Platform. Lấy ví dụ về một ứng dụng được viết bằng đoạn mã Native. Toàn bộ mã thực thi được viết ở dạng Native sẽ được biên dịch hoặc thông dịch qua dạng mà tầng Platform có thể hiểu được và thực thi trên đó.

Đối với ứng dụng viết ở dạng Hybrid, các đoạn mã giao diện của ứng dụng được viết bằng HTML/CSS và được vẽ ra với DOM qua dạng hiển thị được trên WebView. WebView sẽ trực tiếp điều khiển Canvas vẽ lên UI và nhận các event ngược lại thông qua WebView đó. Mã thực thi của các ứng dụng viết dựa trên Hybrid được sử dụng dưới dạng Javascript. JavaScript có thể dễ dàng điều khiển HTML/CSS thông qua giao tiếp với WebView tuy nhiên nhằm mục đích bảo mật, WebView không được cung cấp khả năng giao tiếp với các services hoặc điều khiển phần cứng của lớp Platform dẫn đến các ứng dụng viết dựa trên Hybrid cần một module trung gian thường gọi là **Bridge** để làm điều này. Module này phải đọc hiểu Javascript và tồn tại luôn trong mã nguồn của ứng dụng dẫn đến kích thước của ứng dụng bị tăng lên đáng kể[3](https://kipalog.com/posts/Chapter-1----Ngup-lan-voi-Flutter--Dive-to-Flutter#fn3).

Đối với các ứng dụng được viết ở dạng WebNative hoặc Cross compiled, sẽ không có phần WebView được sử dụng mà thay vào đó module **Bridge** sẽ thực hiện toàn bộ các hành vi trung gian giữa lớp Application và lớp Shell. Mã nguồn của ứng dụng cũng không bị giới hạn bởi Javascript nữa mà tùy thuộc vào nhà phát triển các nền tảng dạng này. Có thể là Native Code có thể là Non-native code. Tùy họ.

Đối với Flutter, mã nguồn được cross compile thành native code. Dựa vào máy ảo Dart VM, những thành phần ít thay đổi sẽ được compile dạng AOT còn những thành phần thường xuyên thay đổi sẽ được compile thành dạng JIT. Flutter không cam kết với các OEM Widgets. Flutter cho phép lập trình viên thay đổi và điều khiển từng pixel trên màn hình và không cần quan tâm đến những giới hạn của OEM Widgets. Tuy nhiên khác với các ứng dụng viết ở dạng Hybrid khi lập trình viên phải tự quan tâm đến các chuẩn của **OEM Widgets**, Flutter cũng cung cấp các Widget mặc định hỗ trợ hai nền tảng giao diện là Material Design (cho Android/Fuchsia) và Cupertino (cho iOS). Bạn cũng có thể quay trờ về với Flat Design thông qua một số thư viện trôi nổi trên thị trường. Sự so sánh kiến trúc của các nền tảng này được vẽ trong biểu đồ bên dưới.

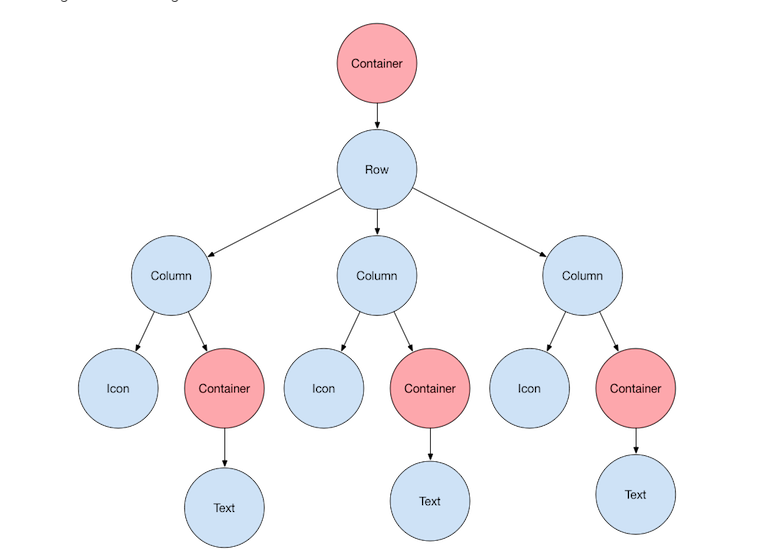


## Widget

Tất cả mọi thứ trpng giam diện của flutter đều là widget. Việc quản lý các widget được lập trình viên thực hiện.

Note: Các widget không thể tự vẽ chính nó

Các widget sẽ đưuọc cấu trúc thành cây



## StatefullWidget , statelesswidget.

Widget tại là của cây có thể chia làm 2 loại: có thể thay đổi trạng thái, hoặc là widet tĩnh

Flutter cung cấp 2 lớp trìu tượng là StatefulWidget và StatelessWidget

### Statelesswidget:

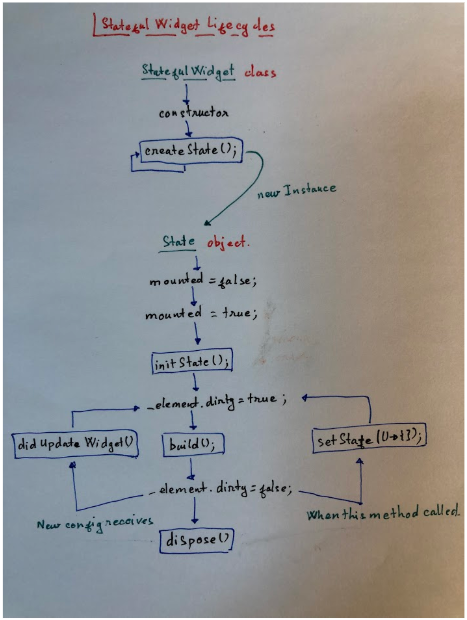
k thay đổi trong toàn bộ quá trình app chạy

### Statefullwidget:

Có thể thay đổi được và sẽ render khi state thay đổi.

Khi tạo mới một StatefulWidget, ta quan tâm đến hai class là StatefulWidget và State của Widget đó. StatefulWidget là phần không thay đổi trong Widget Tree, thay vào đó, State là phần được thay đổi. Mỗi khi Widget bị làm bẩn, state của Widget đó sẽ được khởi tạo lại thông qua việc gọi đến phương thức StatefulWidget#createState(). Generic type[5](https://kipalog.com/posts/Chapter-1----Ngup-lan-voi-Flutter--Dive-to-Flutter#fn5) (kiểu giữ chỗ) của State định nghĩa ra kiểu dữ liệu của thuộc tính State#widget chứa trong nó. Thuộc tính này chính là instance của StatefulWidget đã tạo ra State đó. Chính nhờ vậy, từ trong State hoàn toàn có thể lấy được những giá trị được truyền vào bên trong StatefulWidget thay vì phải truyền vào thông qua hàm khởi tạo.

### Vòng đười của state



#### build ()

Phương thức build () mô tả cách hiển thị một widget. Flutter gọi built() khi nó chèn một widget vào widget hierarchy, và sau đó chạy lại build () mỗi khi widget cần được update.

Trong State chứa một thể hiện của Element tương ứng trong cây thông qua State#\_element. Khi cần vẽ lại, có hai sự kiện được sử dụng để làm bẩn Element là **didUpdateWidget()** và **setState().**

#### **didUpdateWidget()**

Ở hàm thứ nhất, khi configuration bị thay đổi, Widget sẽ được thông báo về sự thay đổi này. Lập trình viên @override lại phương thức didUpdateWidget() để bắt sự kiện trước khi Flutter build lại.

#### **setState()**.

Để thực hiện yêu cầu Widget thay đổi State, lập trình viên sử dụng phương thức setState(). Trong phương thức setState()có cho phép truyền vào một hàm. Mục đích của việc này là đôi khi công việc build lại Widget sẽ tốn một khoảng thời gian trước khi mọi thứ hoàn thành. Hàm callback được truyền vào sẽ đảm bảo Flutter thực thi được trọn vẹn công việc của mình. Ví dụ sau khi thay đổi kích thước của một list view mà muốn scroll xuống cuối list.

Note: chỉ nên dùng setState để cập nhập các thay đổi trong giao diện tránh cập nhập các thay đổi logic, tính toán

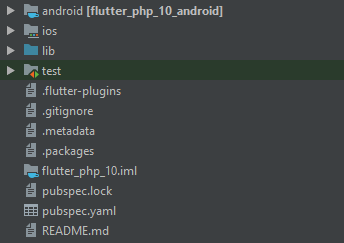
1. **createState():** When the Framework is instructed to build a StatefulWidget, it immediately calls createState()
2. **mounted is true:** When createState creates your state class, a buildContext is assigned to that state. BuildContext is, overly simplified, the place in the widget tree in which this widget is placed. Here's a longer explanation. All widgets have a bool this.mounted property. It is turned true when the buildContext is assigned. It is an error to call setState when a widget is unmounted.
3. **initState():** This is the first method called when the widget is created (after the class constructor, of course.) initState is called once and only once. It must called super.initState().
4. **didChangeDependencies():** This method is called immediately after initState on the first time the widget is built.
5. **build():** This method is called often. It is required, and it must return a Widget.
6. **didUpdateWidget(Widget oldWidget):** If the parent widget changes and has to rebuild this widget (because it needs to give it different data), but it's being rebuilt with the same runtimeType, then this method is called. This is because Flutter is re-using the state, which is long lived. In this case, you may want to initialize some data again, as you would in initState.
7. **setState():** This method is called often from the framework itself and from the developer. Its used to notify the framework that data has changed
8. **deactivate():** Deactivate is called when State is removed from the tree, but it might be reinserted before the current frame change is finished. This method exists basically because State objects can be moved from one point in a tree to another.
9. **dispose():** Dispose is called when the State object is removed, which is permanent. This method is where you should unsubscribe and cancel all animations, streams, etc.
10. **mounted is false:** The state object can never remount, and an error is thrown is setState is called.

<https://kipalog.com/posts/Chapter-1----Ngup-lan-voi-Flutter--Dive-to-Flutter>

<https://kipalog.com/posts/Chapter-1----Ngup-lan-voi-Flutter--Dive-to-Flutter>

<https://techtalk.vn/su-khac-nhau-giua-native-app-mobile-web-app-va-hybrid-app.html>

# Các thành phần trong 1 project flutter



## pubspec.yaml

import các package

## android

Mã nguồn được build thành android.

Khi cần hoàn toàn có thể chỉnh sửa code tại đây để phù hợp với nền tảng android

## IOS

Mã nguồn được build thành ios.

Khi cần hoàn toàn có thể chỉnh sửa code tại đây để phù hợp với nền tảng ios

## lib

Nơi viết mã nguồn flutter, chứa các class có đuôi .dart

## Test

Chứa file test của hệ thống

# Các thành phần widget trong fllutter

<https://www.youtube.com/watch?v=z5iw2SeFx2M&list=PLOU2XLYxmsIL0pH0zWe_ZOHgGhZ7UasUE&index=3>

## Scaffold

Là một cấu trức layout cơ bản theo kiểu material design

<https://api.flutter.dev/flutter/material/Scaffold-class.html>

<https://viblo.asia/p/tao-drawer-don-gian-trong-flutter-ORNZqa2GZ0n>

## Stack

Xếp trồng các thành phần lên nhau

Overflow: xử lý khi thằng trên to hơn thằng dưới.

Fit: Xử lý khi các child không phù hợp kích thước

## Container

Như div bao gồm các thuộc tính căn chỉnh,

Là 1 hộp có thể căn chỉnh các thuộc tính, vị trí, background.

Căn chỉnh vị trí child: alignment

Border: decoration.Boxdecoration.

Biến box thành hình tròn: Decoration

Tạo bóng cho hộp:

child: Container(  
 padding: EdgeInsets.all(kDefaultPadding / 2),  
 decoration: BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.only(  
 bottomLeft: Radius.circular(10),  
 bottomRight: Radius.circular(10),  
 ),  
 boxShadow: [  
 BoxShadow(  
 offset: Offset(0, 10),  
 blurRadius: 50,  
 color: kPrimaryColor.withOpacity(0.23),  
 ),  
 ],

Ảnh nền

decoration: BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.only(  
 bottomLeft: Radius.circular(10),  
 bottomRight: Radius.circular(10),  
 ),  
 image: DecorationImage(  
 fit: BoxFit.cover,  
 image: AssetImage(image),  
 ),

### alignment

Dinh dang vi tri cua widget con

alignment: Alignment.*centerLeft*,

## Column

### crossAxisAlignment

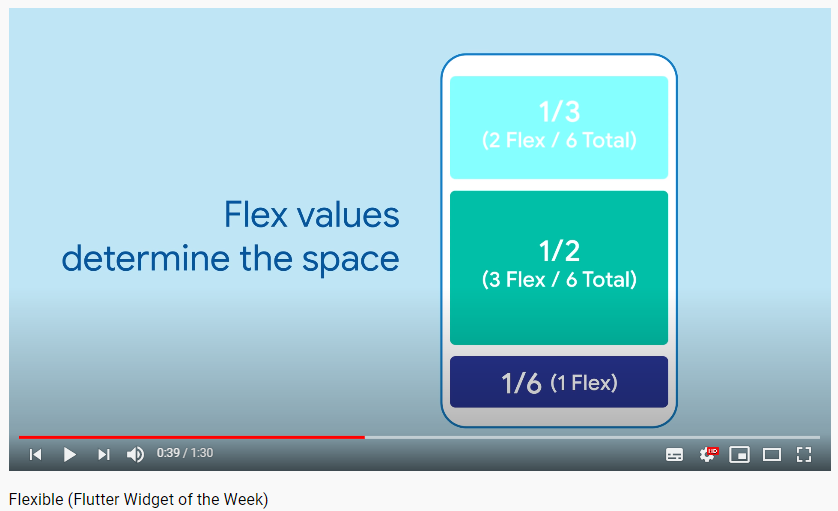
Dinh dang vi tri cua widget con

crossAxisAlignment: CrossAxisAlignment.start,

## Flexible

Điểu khiển linh hoạt vị trí các widget bằng thuộc tính flex

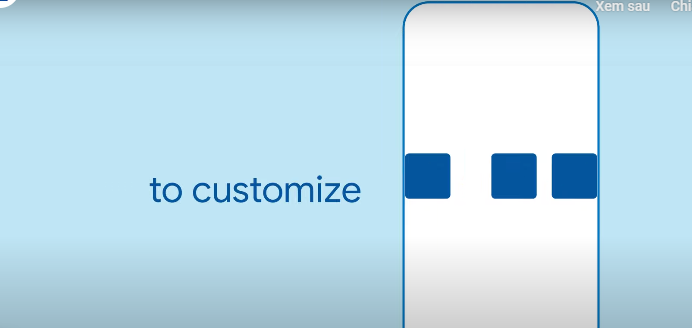
Tính kích thước phần tử cột hoặc dòng theo tỷ lệ



## Row

## Spacer

Điều chỉnh tỉ lệ khoảng cách các khối trống trong row



<https://api.flutter.dev/flutter/widgets/Spacer-class.html>

## Opacity

Xóa một thành phần widget nhưng vẫn dữ lại khoảng trống

## Divider

Giải phân cách, 1 đường chia, dòng kẻ

## RaisedButton

Bo tròn 

## TextField:

**final** textController = TextEditingController();

@override   
**void** dispose() {  
 // Clean up the controller when the widget is disposed.  
 passController.dispose();  
 **super**.dispose();  
}

Thêm controller vào textfield

child: TextField(

controller: myController,

),

Bỏ gạch chân ở text

decoration: InputDecoration(  
 prefixIcon: Icon(  
 Icons.*vpn\_key*,  
 color: Colors.*white*,  
 ),  
 border: InputBorder.*none*,

Note: You will have to provide a specific width to the TextField, simply provide width in your Container or wrap your Column in Expanded

<https://stackoverflow.com/questions/56310961/flutter-bottom-sheet-with-textfield-textformfield>

## FlatButton

Bo góc shape: RoundedRectangleBorder(  
// borderRadius: BorderRadius.circular(20),  
// ),

## ListTile

<https://api.flutter.dev/flutter/material/ListTile-class.html>

Một hàng có chiều cao cố định và có thể custom hiển thị ở trong (Thường sử dụng cho các item cùng bậc, item trong list view)

## Form

## CheckboxListTile

Checkbox với listtile

## Future

[Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) is a core Dart class for working with async operations. A Future object represents a potential value or error that will be available at some time in the future.

**Note: color phải đặt trong decoration nếu có cả 2 thược tính này**

## Material design.

Là một cách thiết kế được phát triển bởi google gồm các thành phần nội dung được xếp trồng lên nhau.

Không gian thiết kế của material design được bố trí theo trục xOyz. Các thành phần sẽ có độ dày 1dp, song song với mặt phẳng xOy và được xếp chồng lên nhau theo trục Oz

Child: Image, Text, Icon, RaisedButton

MaterialApp: khởi tạo thiết kế app

**body** đây là **primary content** của MaterialApp

<https://flutter.dev/docs/development/ui/widgets/input>

## Text

style TextStyle();

Chàn text: dùng

Flexible(  
 child: Container(

Bọc quanh text

## RichText

Áp dụng cho nhiều style trong cùng một dòng text

<https://api.flutter.dev/flutter/widgets/RichText-class.html>



## SafeArea:

Bảo vệ các thành phần widget không vướng vào các phần hiển thị của các thiết bị.

## Expanded

Chia tỉ lệ các thành phần trong row với thuộc tính flex

<https://api.flutter.dev/flutter/widgets/Expanded-class.html>

## Wrap:

Làm cho các thành phần trong row không bị tràn dòng mà sẽ tự động xuống dòng khi không còn đủ chỗ trống. Direction: Axis.vertical.

## SingleChildScrollView

Điều chỉnh màn hình scroll phù hợp với màn hình

### AlwaysScrollableScrollPhysics

K chan noi dung van cuon

child: SingleChildScrollView(  
 physics: AlwaysScrollableScrollPhysics(),

## AnimatedContainer

Điểu chỉnh mô tả của một widget

## FutureBuilder

Cập nhập dữ liệu

## FadeTransition

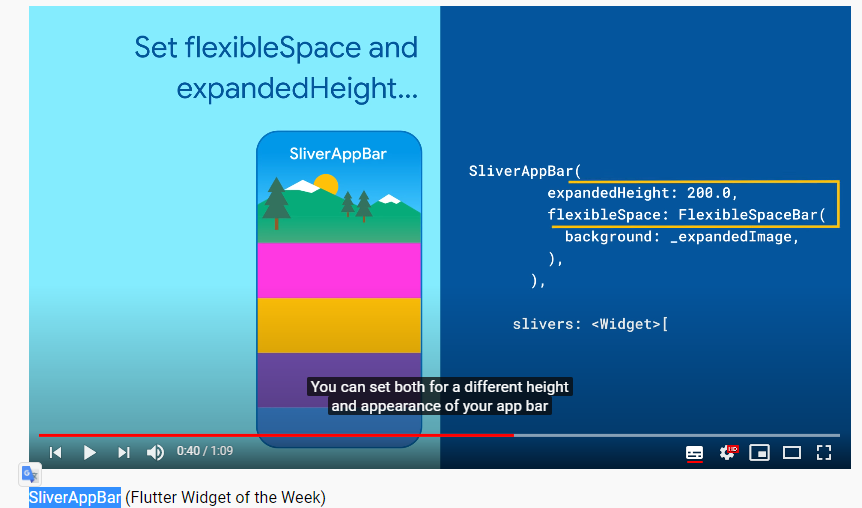
## FloatingActionButton

Nút float nôỉ

## PageView

## Table

## SliverAppBar



## SliverGrid

Sử lí scroll view lồng nhau



<https://api.flutter.dev/flutter/widgets/SliverGrid-class.html>

## FadeInImage

Công cụ load hình ảnh

## [fluttertoast](https://pub.dartlang.org/packages/fluttertoast)

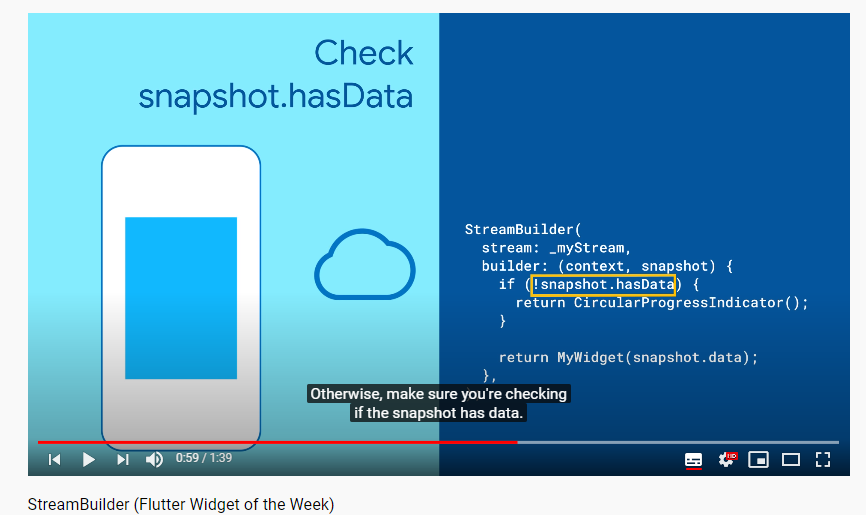
Toast trogn flutter

<https://pub.dev/packages/fluttertoast>



## StreamBuilder

Load dữ liệu và tự động đồng bộ



## InheritedModel

## ClipRRect

Tạo khung cho hình ảnh

## Hero

Di chuyển vị trí ảnh

## CustomPaint

## Tooltip

## Appbar

<https://kodestat.gitbook.io/flutter/27-flutter-appbar>

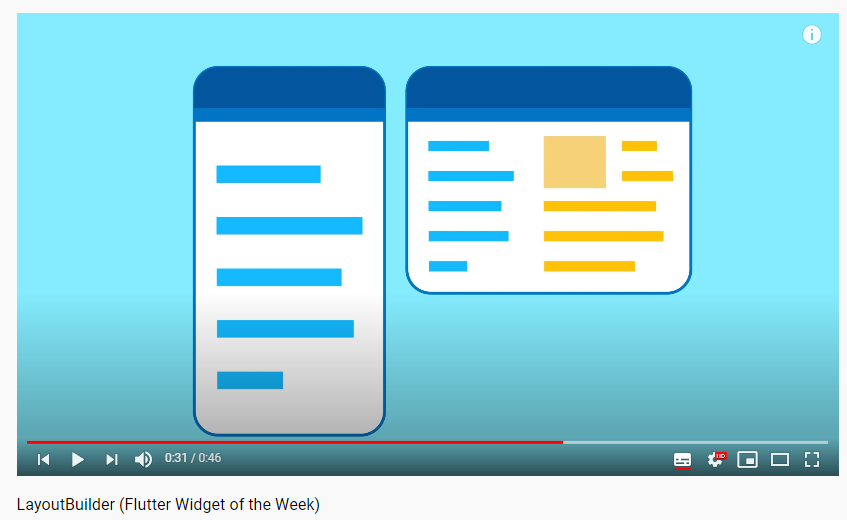
<https://flutter.dev/docs/catalog/samples/basic-app-bar>

## FittedBox

Fit các thành phần với nhau

## LayoutBuilder

Điều chỉnh layout sao cho phù hợp với màn hình

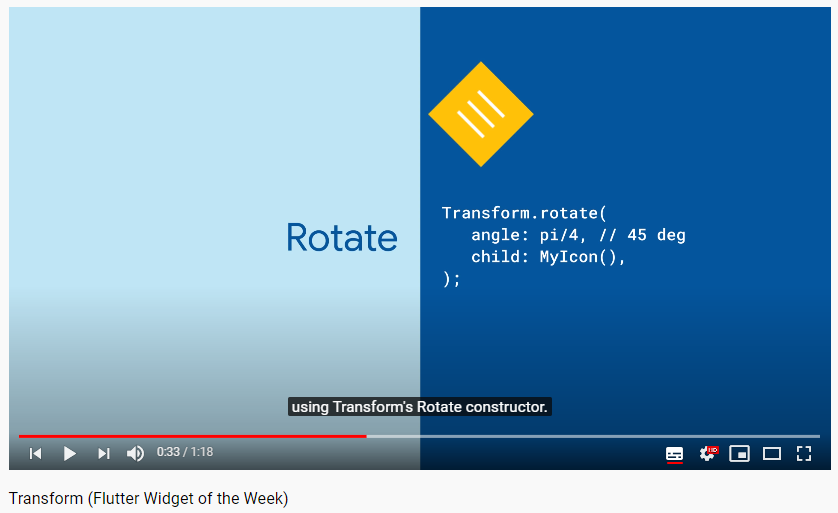


## AbsorbPointer

Điều khiển trạng thái ấn hoặc dữ màn hình

## Transform

Hiệu ứng biến đổi màn hình



## BackdropFilter

## Align

Custoom vị trí của các widget trong container và row colums

<https://www.youtube.com/watch?v=g2E7yl3MwMk>

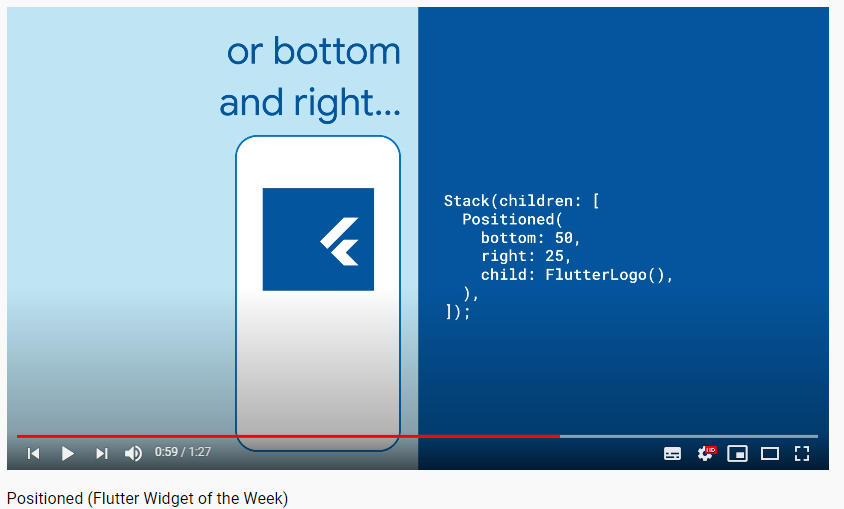
<https://api.flutter.dev/flutter/widgets/Align-class.html>



## Positioned

Điều chỉnh vị trí của các wetget trong stack

<https://www.youtube.com/watch?v=EgtPleVwxBQ>



## AnimatedBuilder

## Dismissible

Tạo các tương tác, kéo trái phải của widget

## SizeBox

Tạo một box chứa các thành phần widget, có thể tạo khoảng trống.

Kích thước của box

## ValueListenableBuilder

Lắng nghe sự kiện nào đó

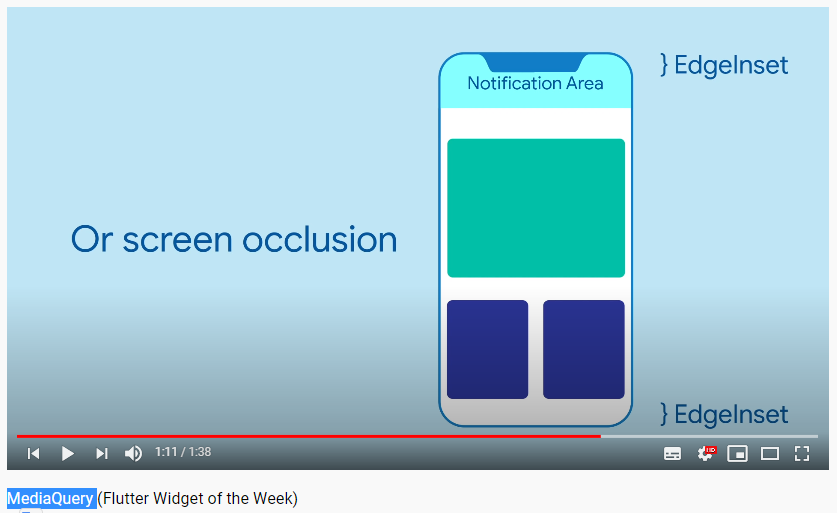
## Draggable

## AnimatedList

Các hiệu ứng liên quan tới list, thêm xóa,..

## MediaQuery

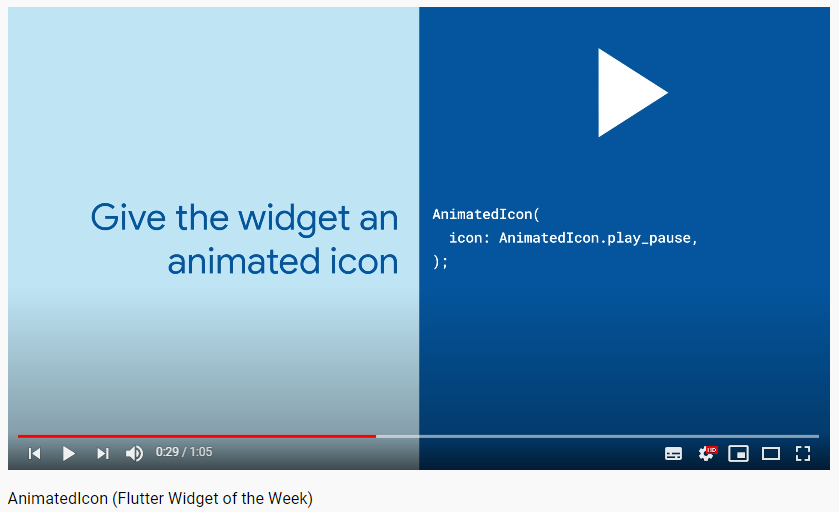
Điểu chỉnh layout theo kích thước màn hình



## InheritedWidget

## AnimatedIcon

Hiệu ứng của các icon



## AspectRatio

## LimitedBox

Đặt giá trị giới hạn của Box

## ReorderableListView

Di chuyển các item của listView



## AnimatedSwitcher

## AnimatedPositioned

## IndexedStack

## FractionallySizedBox

Điều chỉnh tỉ lệ của 1 witget so với phần tử cha



## Semantics

## SelectableText

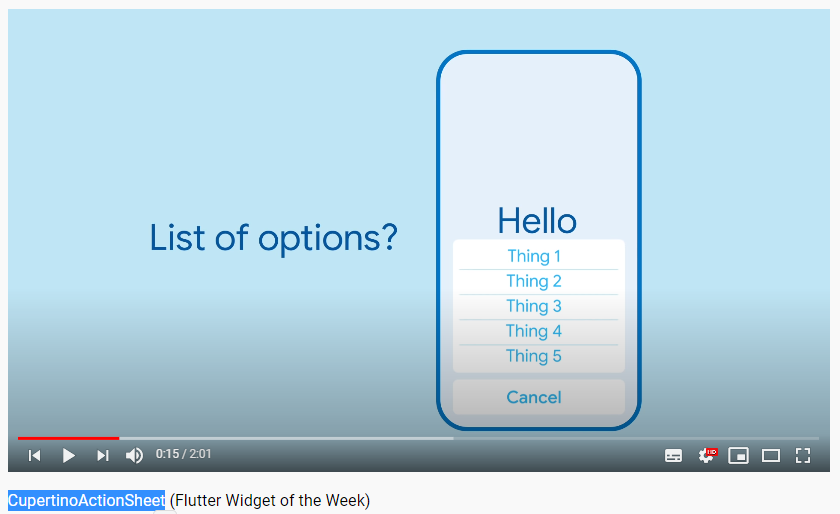
Chọn text

## DataTable

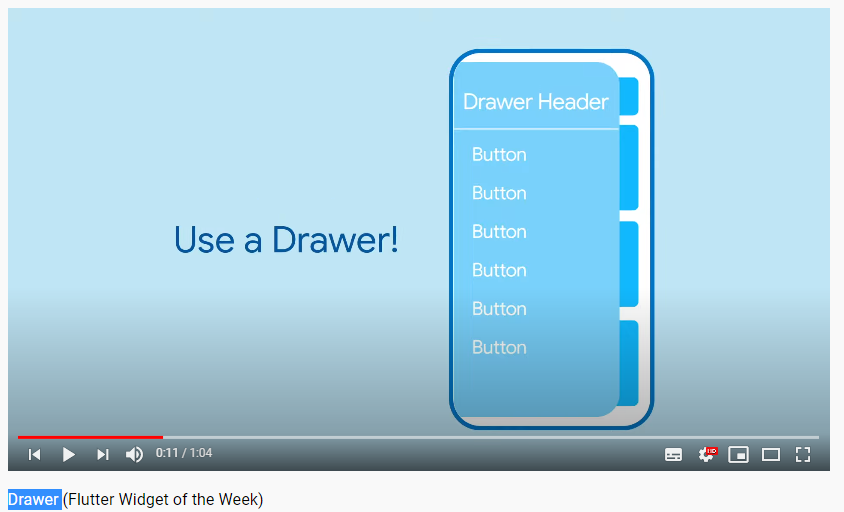
Table

## AlertDialog

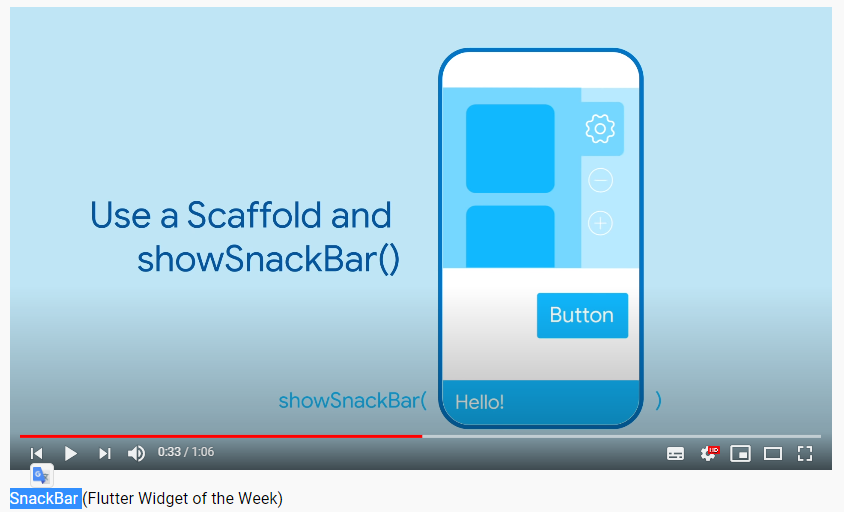
## CupertinoActionSheet



## Drawer



## SnackBar



## ListWheelScrollView

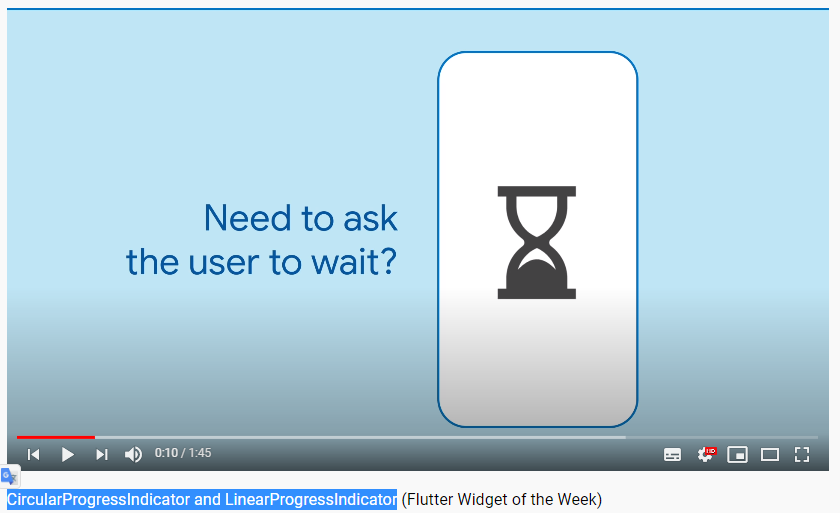


## ClipPath

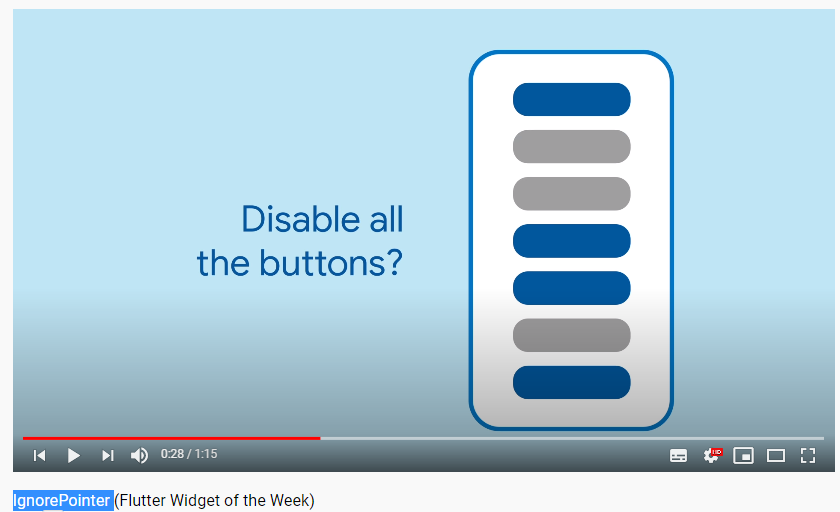
Custom box



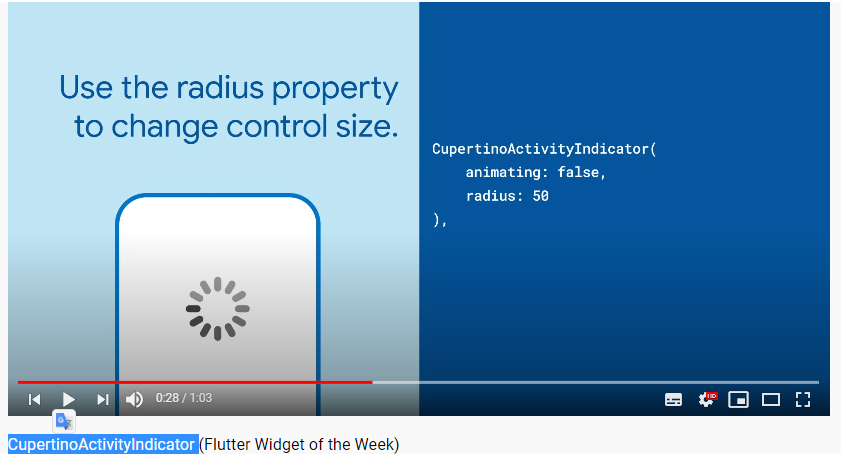
## CircularProgressIndicator and LinearProgressIndicator



## IgnorePointer



## CupertinoActivityIndicator



## Google Map

<https://medium.com/flutter/google-maps-and-flutter-cfb330f9a245>

<https://pub.dev/packages/google_maps_flutter>

* **onMapCreated**: method that is called on map creation and takes a MapController as a parameter.
* **initialCameraPosition**: required parameter that sets the starting camera position. Camera position describes which part of the world you want the map to point at.
* **mapController**: manages camera function (position, animation, zoom). This pattern is similar to other controllers available in Flutter, for example [TextEditingController](https://docs.flutter.io/flutter/widgets/TextEditingController-class.html).

|  |  |
| --- | --- |
| Completer<GoogleMapController> \_controller = Completer();  static const LatLng \_center = const LatLng(45.521563, -122.677433); | |
|  | |  |
|  | |  |
| body: GoogleMap( |
|  | onMapCreated: \_onMapCreated, | |
|  | initialCameraPosition: CameraPosition( | |
|  | target: \_center, | |
|  | zoom: 11.0, | |
|  | ), | |
|  | ), | |

### Vị trí hiện tại

Thư viện location

<https://medium.com/swlh/working-with-geolocation-and-geocoding-in-flutter-and-integration-with-maps-16fb0bc35ede>

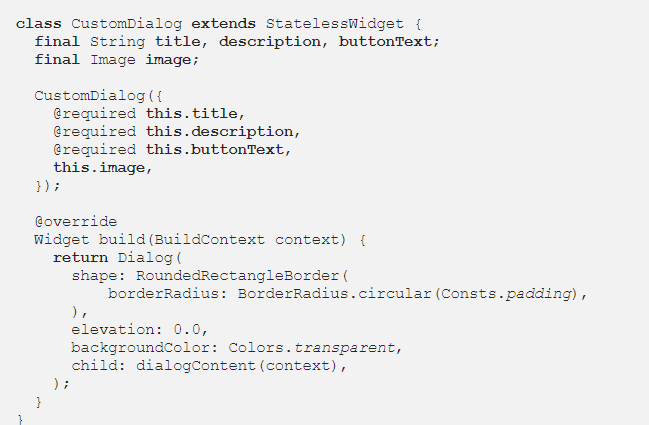
<https://pub.dev/packages/location>

## Dialog

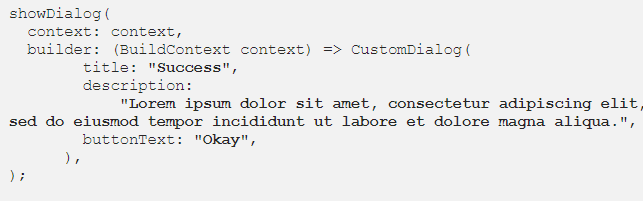
### Custom dialog

<https://medium.com/@excogitatr/custom-dialog-in-flutter-d00e0441f1d5>

tạo một class



Gọi đến hàm tạo dialog



### Progress\_dialog

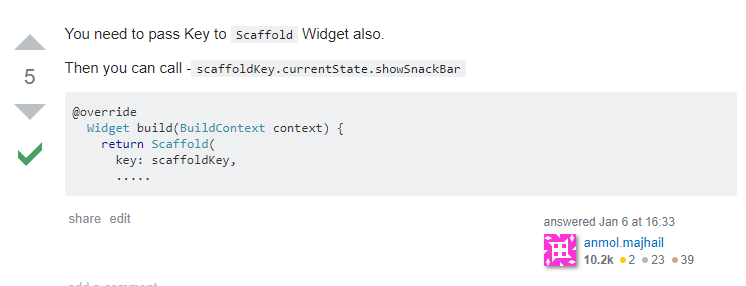
<https://www.youtube.com/watch?v=wgw49985teE>

### Pop\_up dialog(snackbar)

<https://www.youtube.com/watch?v=FGfhnS6skMQ>

#### Lỗi: [Flutter showSnackBar called on null](https://stackoverflow.com/questions/54063473/flutter-showsnackbar-called-on-null)

<https://stackoverflow.com/questions/54063473/flutter-showsnackbar-called-on-null>



## ProgressHUD

Progress gọi ngay trogn initState dùng để yêu cầu dữ liệu

<https://pub.dev/packages/progress_hud>

## Http

Khi truyền theo {“”:””,””:””} thì phải lấy bằng phương thức $\_post.

### Dạng 1

Khi gửi body: {"status":status,"body":jsonEncode({'status': status, 'admin': admin})});

Dữ liệu vẫn vào php://input raw:status=status&body=%7B%22status%22%3A%22status%22%2C%22admin%22%3A%22admin%22%7D

Vẫn lấy được dữ liệu dạng $\_POST

Status:status,body:{"status":"status","admin":"admin"}

### Dạng 2

Khi gửi body: jsonEncode({'status': status, 'admin': admin}));

Dữ liệu vẫn vào php://input

raw:{"status":"status","admin":"admin"}

Không lấy được dữ liệu dạng $\_POST

## DateTime

<https://kodestat.gitbook.io/flutter/26-flutter-datepicker>

<https://api.flutter.dev/flutter/material/showTimePicker.html>

<https://api.flutter.dev/flutter/material/showDatePicker.html>

## Explain

<https://medium.com/flutter-community/flutter-statemanagement-with-provider-ee251bbc5ac1>

## Set<> get

<https://api.flutter.dev/flutter/dart-core/Set-class.html>

# Các class điều chỉnh thuốc tính của widget

<https://api.flutter.dev/flutter/painting/TextStyle-class.html>

## EdgeInsets

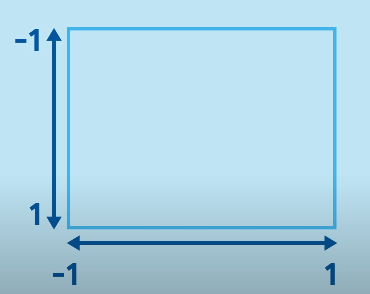
Kiểu dữ liệu căn chỉnh theo 4 hướng

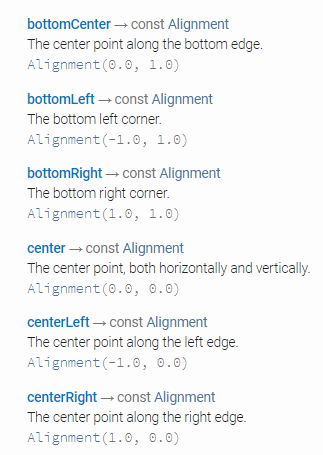
## TextStyle

Điều chỉnh style của text

## Alignment

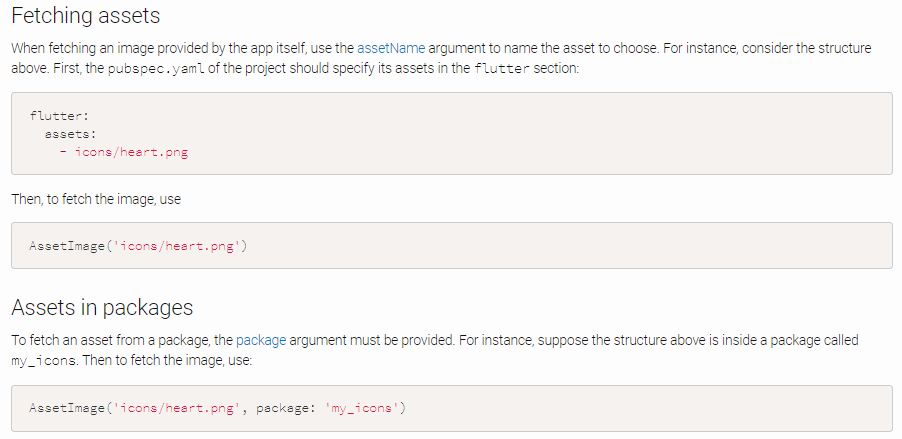
Biến căn chỉnh theo hình chữ nhật

 Alignment(0.0, 0.0) Alignment(-1.0, -1.0)



## AssetImage

Thư viện dùng lấy ảnh



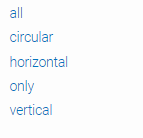
## Border

Border của 1 phần nào đó định nghĩa theo 4 chiều, trên, dưới, trái, phải

<https://api.flutter.dev/flutter/painting/Border-class.html>

## BorderRadius

Bo góc cho border



## BorderSide

Có thể tọa màu cho border

Container(

padding: EdgeInsets.all(8.0),

decoration: BoxDecoration(

border: Border(

top: BorderSide(width: 16.0, color: Colors.lightBlue.shade50),

bottom: BorderSide(width: 16.0, color: Colors.lightBlue.shade900),

),

),

child: Text('Flutter in the sky', textAlign: TextAlign.center),

)

## BoxDecoration

<https://api.flutter.dev/flutter/painting/BoxDecoration-class.html>

Quan trọng để trang trí 1 box

Container(

decoration: BoxDecoration(

color: **const** Color(0xff7c94b6),

image: **const** DecorationImage(

image: NetworkImage('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl-2.jpg'),

fit: BoxFit.cover,

),

border: Border.all(

color: Colors.black,

width: 8,

),

borderRadius: BorderRadius.circular(12),

),

)

## BoxShadow

1 list các bóng của hộp

## Decoration

Trang trí hộp

## Gradient

Độ dốc 2d

Đổ màu từ đâu đến đâu và có gradient như nào

## Shadow

A single shadow

# Internation­alizing Flutter apps (multi language)

<https://flutter.dev/docs/development/accessibility-and-localization/internationalization>

<https://github.com/flutter/flutter/tree/master/packages/flutter_localizations/lib/src/l10n>

ví dụ

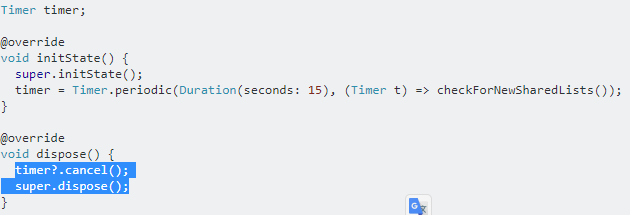
<https://github.com/flutter/website/blob/master/examples/internationalization/minimal/lib/main.dart>

note: Các biến cần gọi sau khi hàm build đã thực hiện

# Gọi function mỗi 1 khoảng thời gian

<https://fluttermaster.com/tips-to-use-timer-in-dart-and-flutter/>

dừng gọi <https://stackoverflow.com/questions/52569602/flutter-run-function-every-x-amount-of-seconds>



# DropdownButton

<https://api.flutter.dev/flutter/material/DropdownButton-class.html>

Note: giá trị lựa chọn ban đầu của dropdown phải trùng với 1 trong các giá trị của mảng truyền vào

Trong mảng chuyển vào không được có 2 giá trị bằng nhau

Khuyến cáo truyền dữ liệu đầu vào bằng list object

<https://stackoverflow.com/questions/49720450/flutter-dropdownbutton-using-keyvaluepair?fbclid=IwAR2_T5JM1bCgnUdLc5Lm83pIH_M7YIMP6wwel1rddrAoGYvIaO4tscmTB_g>

# Chuyển giữa các màn hình flutter

Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => SecondRoute()),  
);

***Chú ý 1:*** Với Android, việc tạo constructor có chứa tham số cho Activity, và tương tự với iOS ViewController, là điều hoàn toàn “cấm kị”, thì với Flutter, đây là con đường dễ nhất để truyền tham số sang màn hình sau. Sở dĩ có thể thực hiện, và phải nên thực hiện, điều này là vì, như tôi luôn nhắc tới nhắc lui với các bạn, là Flutter app không hoạt động theo kiểu Android Activity hay iOS ViewController mà thay vào đó, khi bạn Navigator.push, thì màn hình mới – thực chất là Widget – sẽ nằm chồng lên Widget gọi tới nó và chiếm focus, tạo cảm giác startActivity hay present. Với Android, bạn phải gửi gián tham số gián tiếp qua Intent, với iOS là segue, còn với Flutter chỉ đơn giản là dùng luôn constructor.

***Chú ý 2:*** Ở đây tôi chỉ sử dụng trường hợp đơn giản nhất là chỉ gửi một tham số. Nếu bạn muốn gửi nhiều tham số cùng lúc, bạn có thể đặt thêm biến trong constructor của Widget nhận tham số – tức Widget sẽ xuất hiện. Tuy nhiên, vì Navigator.pop(BuildContext, T result) – bạn sắp gặp tới nó ở phần ngay bên dưới – chỉ chứa một result duy nhất, nên tốt nhất, nếu muốn truyền nhiều tham số một lúc, bạn nên đặt chúng trong một Map<K,V> duy nhất và là tham số duy nhất trong constructor của các màn hình.

***Chú ý 3:*** Bản thân Navigator.push là một Future. Trong trường hợp các bạn quên, thì Future<T> sẽ trả về một T trong result nếu bạn sử dụng nó với .then((T result) => …) hoặc trực tiếp T nếu sử dụng với await. Do đó, nó có tác dụng bắt kết quả được trả về từ màn hình được trỏ tới.

Quay về màn hình trước

Navigator.*pop*(context);

<http://eitguide.net/flutter-bai-11-di-chuyen-giua-cac-man-hinh-va-dialog/>

# Quản lý route

Hướng dẫn: <https://flutter.dev/docs/cookbook/navigation/named-routes>

## Quản lý route với thư viện Fluro

Hướng dẫn: https://youtube.com/watch?v=w3s\_M5BFOWA

Tài liệu: <https://pub.dev/packages/fluro>

Code mẫu: <https://github.com/thanhniencung/demo-route-fluro>

# Lập trình hướng đối tượng

## Đôi từ json sang object và ngược lại

<https://flutter.dev/docs/development/data-and-backend/json>

Một class trong flutter(dart) chỉ cho phép định nghĩa 1 hàm constructor duy nhất

# Lưu biến

* Cách đầu tiên là đặt kiểu dữ liệu trước tên biến. Biến sẽ chỉ được khai báo một lần và tất cả mọi lần thay đổi dữ liệu biến đều phải giữ nguyên kiểu dữ liệu cũ.
* Sử dụng var giúp cho biến dù được khai báo một lần và không thể thay đổi kiểu dữ liệu. Tuy nhiên sau lần nhận dữ liệu đầu tiên, biến mới được xác định kiểu dữ liệu của bản thân.
* Sử dụng dynamic giúp cho biến dù được khai báo một lần nhưng muốn thay đổi dữ liệu và kiểu dữ liệu bao nhiêu lần cũng được.
* final và const là hai từ khóa dễ nhầm cần được nói riêng.

# Up ảnh từ thiết bị

Import thư viện **image\_picker\_modern: ^0.4.12+2**

<https://medium.com/@viveky259259/image-picker-in-flutter-android-ios-c2d96ae6a360>

<https://www.developerlibs.com/2018/08/flutter-capture-image-from-camera-or.html>

Thư viện **image\_picker**

# Cấp quyền cho thiết bị

# Click Text

Sử dụng GestureDetector

<https://stackoverflow.com/questions/49067624/how-can-i-implement-onpressed-callback-for-text-widget-flutter>

# Thao tác với image

# Auto place picker google map

## Thư viện

Thêm thư viện flutter\_google\_places  <https://pub.dev/packages/flutter_google_places#-readme-tab->

Thêm thư viên <https://pub.dev/packages/geocoder> để tìm kiếm địa chỉ theo tên

## Import các thư viện

**import** 'package:flutter\_google\_places/flutter\_google\_places.dart';  
**import** 'package:geocoder/geocoder.dart';  
**import** 'package:google\_maps\_webservice/places.dart';

## Khai báo biến

**const** kGoogleApiKey = "AIzaSyBus9xYNKFJ0Xyq8QYLRqLmlXMYRRS85a4";  
GoogleMapsPlaces \_places = GoogleMapsPlaces(apiKey: kGoogleApiKey);

## Sử dụng

body: Container(  
 alignment: Alignment.*center*,  
 child: RaisedButton(  
 onPressed: () **async** {  
 // show input autocomplete with selected mode  
 // then get the Prediction selected  
 Prediction p = **await** PlacesAutocomplete.*show*(  
 context: context, apiKey: kGoogleApiKey);  
// displayPrediction(p);  
 },  
 child: Text('Find address'),  
  
 )  
 )  
 );  
 }  
  
 Future<Null> displayPrediction(Prediction p) **async** {  
 **if** (p != **null**) {  
 PlacesDetailsResponse detail =  
 **await** \_places.getDetailsByPlaceId(p.placeId);  
  
 **var** placeId = p.placeId;  
 double lat = detail.result.geometry.location.lat;  
 double lng = detail.result.geometry.location.lng;  
 String name = detail.result.name;  
 **var** address = **await** Geocoder.*local*.findAddressesFromQuery(p.description);  
  
 print(lat);  
 print(lng);  
// print("Address:"+address);  
 print("Name:"+name);  
 }  
 }

## Ví dụ

<https://www.youtube.com/watch?v=sco5RsYgpwc>

<https://stackoverflow.com/questions/55870508/how-to-create-a-simple-google-maps-address-search-with-autocomplete-in-flutter-a>

<https://medium.com/@rajesh.muthyala/flutter-with-google-maps-and-google-place-85ccee3f0371>

## Note:

Prediction

Và

GoogleMapsPlaces

Nằm trong thư viện

**import** 'package:google\_maps\_webservice/places.dart';

# Lấy thời gian

Thư viện format time <https://pub.dev/packages/intl>

Ví dụ

<https://stackoverflow.com/questions/51579546/how-to-format-datetime-in-flutter>

<https://fluttermaster.com/tips-to-use-timer-in-dart-and-flutter/>

# Nén ảnh

<https://github.com/btastic/flutter_native_image>

# stream

dữ liệu đầu vào và ra, vào sink ra stream

# Notification onesignal

## 1: connect flutter with firebase

<https://firebase.google.com/docs/flutter/setup>

## 2: create onesignal project

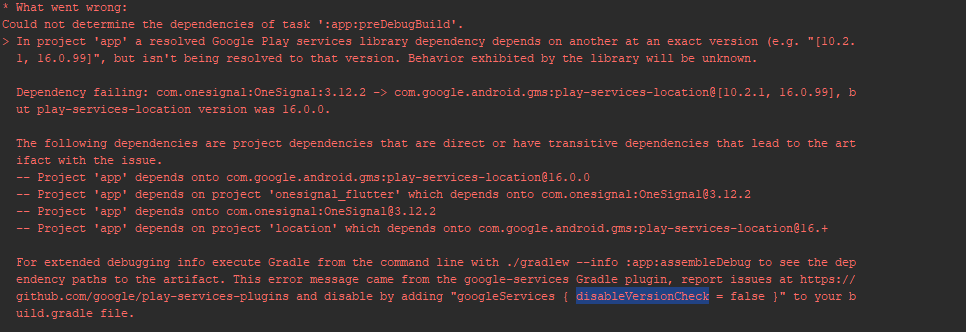
Add firebasekey for onesignal

firebase -> project -> setting->cloudMessaging

## 3: config flutter project

<https://documentation.onesignal.com/docs/generate-a-google-server-api-key>

Add key onesignal for flutter project and run ap to connect



Add

apply plugin: 'com.google.gms.google-services'  
com.google.gms.googleservices.GoogleServicesPlugin.config.disableVersionCheck = true

trong app/buildgradle

Thêm

buildTypes {  
 release {  
 // *TODO: Add your own signing config for the release build.* // Signing with the debug keys for now, so `flutter run --release` works.  
 signingConfig signingConfigs.debug  
 }  
 }  
 configurations.all {  
 resolutionStrategy {  
 force 'androidx.media:media:1.0.0'  
 }  
 }  
}  
  
flutter {  
 source '../..'  
}

vào /src/build.gradle

configurations.all {  
 resolutionStrategy {  
 force 'androidx.media:media:1.0.0'  
 }  
}

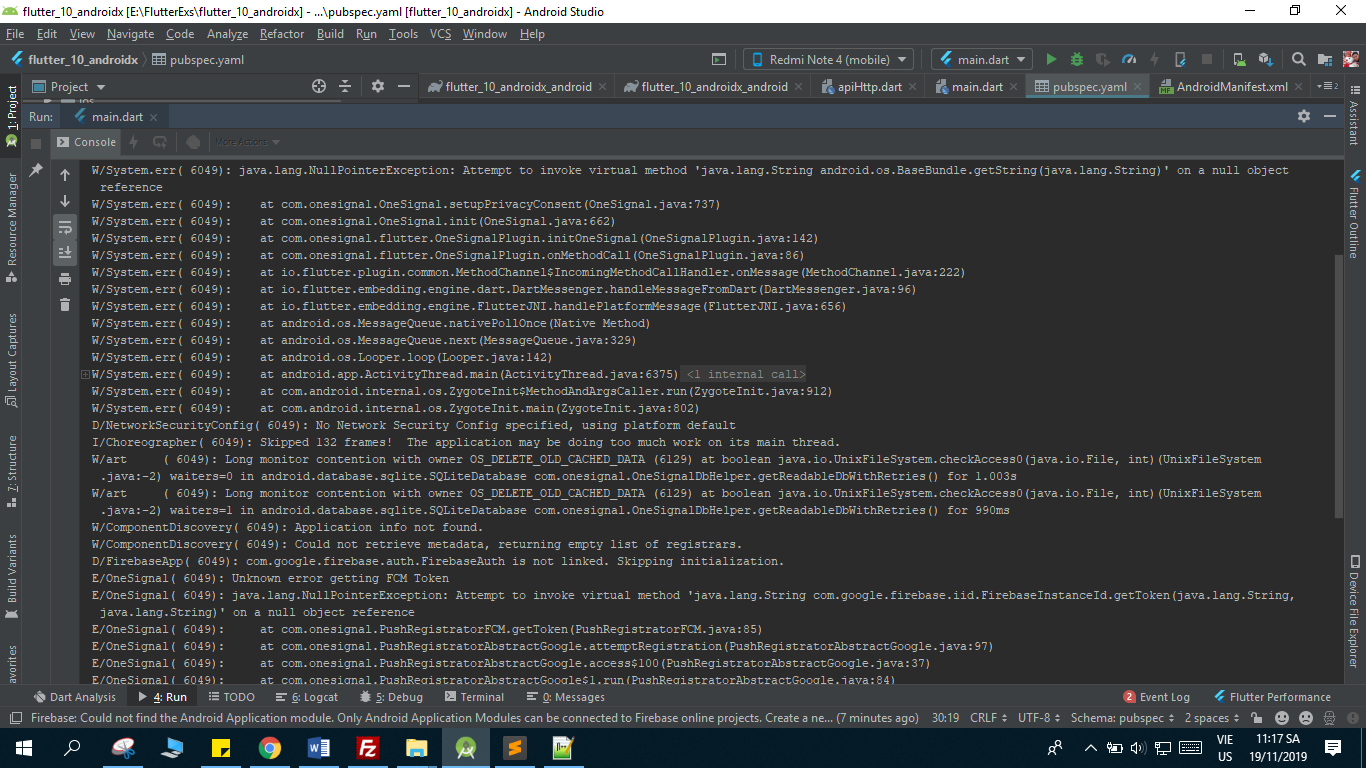
add pluin onesinag trước pluin application

apply plugin: 'com.onesignal.androidsdk.onesignal-gradle-plugin'  
apply plugin: 'com.android.application'

buildscript {  
 ext.kotlin\_version = '1.3.21'  
 repositories {  
 google()  
 jcenter()  
 }  
  
 dependencies {  
 classpath 'com.android.tools.build:gradle:3.3.0'  
 classpath 'com.google.gms:google-services:4.2.0'  
 classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin\_version"  
 }  
}

4. setup flutter onesignal

<https://documentation.onesignal.com/docs/flutter-sdk-setup>



# QR code

<https://medium.com/flutter-community/building-flutter-qr-code-generator-scanner-and-sharing-app-703e73b228d3>

<https://pub.dev/packages/qr_flutter>

<https://pub.dev/packages/barcode_scan#-readme-tab->

Hàm gọi tới scanscreen

Future scan() **async** {  
 **try** {  
 String barcode = **await** BarcodeScanner.*scan*();  
 setState(() {  
 **this**.barcode = barcode;  
 showAlertConfirmQRcode(context,barcode);  
 });  
 } **on** PlatformException **catch** (e) {  
 **if** (e.code == BarcodeScanner.*CameraAccessDenied*) {  
 setState(() {  
 **this**.barcode = 'The user did not grant the camera permission!';  
 });  
 } **else** {  
 setState(() => **this**.barcode = 'Unknown error: $e');  
 }  
 } **on** FormatException {  
 setState(() => **this**.barcode =  
 'null (User returned using the "back"-button before scanning anything. Result)');  
 } **catch** (e) {  
 setState(() => **this**.barcode = 'Unknown error: $e');  
 }  
}

Genarage QR image

QrImage(  
 data: \_content,  
 size: 0.5 \* bodyHeight,  
),

# Tạo API

Retrofit: <https://www.youtube.com/watch?v=zFXK5EsrUF0>

Thư viện mặc định của fluter

<https://flutter.dev/docs/cookbook/networking/fetch-data>

Example

<https://www.javacodegeeks.com/2019/09/how-to-make-an-api-call-in-flutter-rest-api.html?fbclid=IwAR1EtFukyts9ho04ujeKIhN9HNb7VE0xaFBHNytuPvAHNTMtShP0Tbiu7B0>

Khi gọi Api sẽ mất một thời gian để đợi phản hồi.

Đây là lập trình không đồng bộ

<https://www.youtube.com/watch?v=uR-9VG7Pfcc>

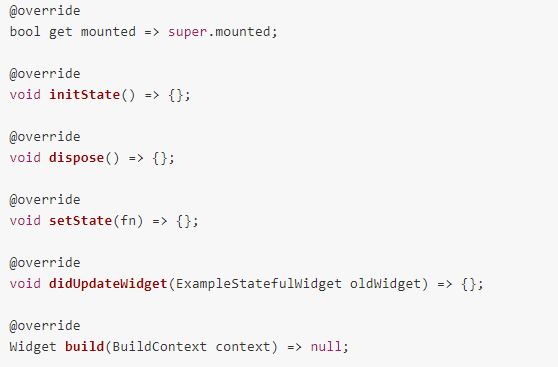
## Giải mã JWT

<https://pub.dev/packages/jaguar_jwt>

<https://stackoverflow.com/questions/52017389/how-to-get-the-claims-from-a-jwt-in-my-flutter-application?rq=1>

# Example

<https://kodestat.gitbook.io/flutter/flutter-using-onchanged-to-show-input-text>

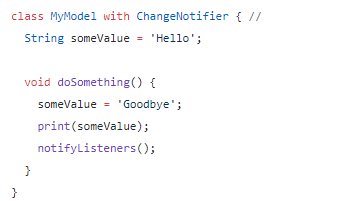


# Bloc Pattent

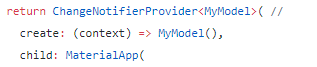
# Provider

Với mỗi cây widget luôn có 1 cây dữ liệu gán vào. Mục đích của provider là blind dữ liệu đó vào widget. Nếu dữ liệu thay đổi thì UI sẽ thay đổi theo.

Để làm điều đó thì thường dữ liệu sẽ kế thừa lớp **Changenotify**. Sau đó có các hàm để đổi dữ liệu, và sau đó gọi tới **notifyListeners()** để thay báo hiệu UI có thay đổi dữ liệu.



Để sử dụng **Provider** này ta cần khai báo bằng **ChangeNotifierProvider**



Hoặc 

Khi sử dụng cụ thể thì dùng **Consumer**



Các dữ liệu model sẽ có trong **cart,**

## Doccument

<https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple>

<https://pub.dev/packages/provider>

<https://viblo.asia/p/flutter-tim-hieu-ve-state-management-voi-provider-LzD5d1wOKjY#_changenotifierprovider-2>

<https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple#providerof>

## Nâng cao

<https://medium.com/flutter-community/making-sense-all-of-those-flutter-providers-e842e18f45dd>

## Changenotify

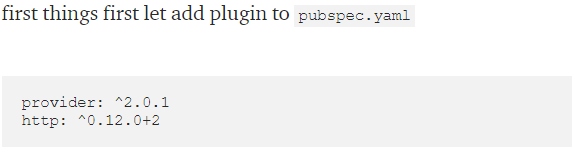
<https://medium.com/flutter-community/flutter-statemanagement-with-provider-ee251bbc5ac1>

<https://api.flutter.dev/flutter/foundation/ChangeNotifier/notifyListeners.html>

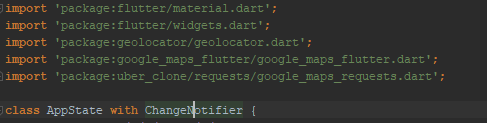
Có thể hiểu đơn giản rằng khi bạn muốn sử lý dữ liệu từ một class và setState lên một màn hình thì bạn k thể sử dụng setState mà bạn sẽ phải sử dụng **notifylistener** của **ChangeNotifier**

## Cách sử dụng:

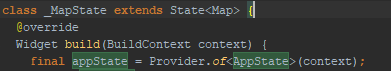
1



2. Tạo lớp xử lý



3. Khai báo tại lớp màn hình



4. Sử dụng:

Gọi đến hàm khi sử lý xong



Viết lớp xử lý

ChangeNotifier is a simple class included in the Flutter SDK which provides change notification to its listeners. In other words, if something is a ChangeNotifier, you can subscribe to its changes. (It is a form of Observable, for those familiar with the term.)

In provider, ChangeNotifier is one way to encapsulate your application state. For very simple apps, you get by with a single ChangeNotifier. In complex ones, you’ll have several models, and therefore several ChangeNotifiers. (You don’t need to use ChangeNotifier with provider at all, but it’s an easy class to work with.)

In our shopping app example, we want to manage the state of the cart in a ChangeNotifier. We create a new class that extends it, like so:

*content\_copy*

class CartModel extends ChangeNotifier {

/// Internal, private state of the cart.

final List<Item> \_items = [];

/// An unmodifiable view of the items in the cart.

UnmodifiableListView<Item> get items => UnmodifiableListView(\_items);

/// The current total price of all items (assuming all items cost $42).

int get totalPrice => \_items.length \* 42;

/// Adds [item] to cart. This and [removeAll] are the only ways to modify the

/// cart from the outside.

void add(Item item) {

\_items.add(item);

// This call tells the widgets that are listening to this model to rebuild.

notifyListeners();

}

/// Removes all items from the cart.

void removeAll() {

\_items.clear();

// This call tells the widgets that are listening to this model to rebuild.

notifyListeners();

}

}

The only code that is specific to ChangeNotifier is the call to notifyListeners(). Call this method any time the model changes in a way that might change your app’s UI. Everything else in CartModel is the model itself and its business logic.

ChangeNotifier is part of flutter:foundation and doesn’t depend on any higher-level classes in Flutter. It’s easily testable (you don’t even need to use [widget testing](https://flutter.dev/docs/testing#widget-tests) for it). For example, here’s a simple unit test of CartModel:

*content\_copy*

test('adding item increases total cost', () {

final cart = CartModel();

final startingPrice = cart.totalPrice;

cart.addListener(() {

expect(cart.totalPrice, greaterThan(startingPrice));

});

cart.add(Item('Dash'));

});

<https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple#providerof>

# Rxdart

Quản lý stream

<https://viblo.asia/p/flutter-rxdart-phan-biet-phan-biet-behaviorsubject-publishsubjectreplaysubject-Qpmlew17Krd>

# Firebase message

# Local notification

<https://www.google.com/search?q=local+notification+flutter&oq=Local+notification+&aqs=chrome.0.0j69i57j0l6.1260j0j4&sourceid=chrome&ie=UTF-8>

# Dio

<https://pub.dev/packages/dio>

Thư viện http

# Isolate

Thư sử dụng đa luồng cho flutter

Thư viện: <https://pub.dev/packages/flutter_isolate>

Hướng dẫn: <https://www.youtube.com/watch?v=A9d2UxjObcU&list=PLVDJsRQrTUz6l30vQ30Dyb0RQHhYqqixt&index=8>

## Isolate and event loops

<https://medium.com/dartlang/dart-asynchronous-programming-isolates-and-event-loops-bffc3e296a6a>

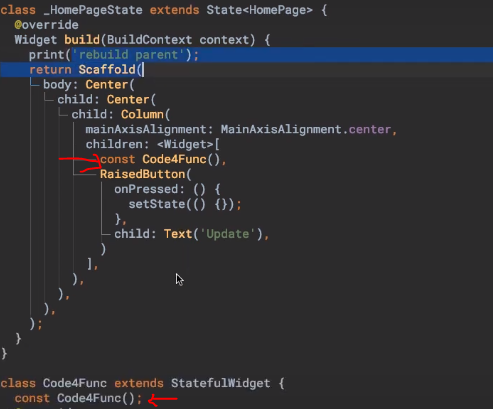
# Note

## With

Đa kế thừa

## Ngăn chặn 1 child không bị rebuild

Class child khai báo contructor với **const,**  khi sử dụng class đó cũng sử dụng **const**



## Lưu dữ liệu với những tác vụ bất đồng bộ

Hướng dấn: <https://www.youtube.com/watch?v=QsjW96ewRQg&list=PLVDJsRQrTUz5ZiT0z_SGvOkrHllexhz72&index=20>

Thư viện: <https://pub.dev/packages/async>

## Out memory khi chụp ảnh nhiều lần

Chụp ảnh trực tiếp từ thư viện camera

<https://flutter.dev/docs/cookbook/plugins/picture-using-camera?fbclid=IwAR26TywsU1fvJgTON7xTyr1iA-VcheuJk7zpulW41LIGXzutpLvtjqXuPzg>

thư viện yêu cần minSDK 21

licycle của state

example

<https://medium.com/flutter-community/flutter-lifecycle-for-android-and-ios-developers-8f532307e0c7>

<https://stackoverflow.com/questions/44331725/onresume-and-onpause-for-widgets-on-flutter>

chưa thể onresume được state trên flutter

## Sau khi chụp ảnh bị quay

<https://stackoverflow.com/questions/14066038/why-does-an-image-captured-using-camera-intent-gets-rotated-on-some-devices-on-a?fbclid=IwAR26TywsU1fvJgTON7xTyr1iA-VcheuJk7zpulW41LIGXzutpLvtjqXuPzg>

## dito Flutter app(kéo xuống reload)

<https://medium.com/flutterpub/adding-swipe-to-refresh-to-flutter-app-b234534f39a7>

### Bước 1:Thêm biến

final GlobalKey<RefreshIndicatorState> \_refreshIndicatorKey =

new GlobalKey<RefreshIndicatorState>();

### Bước 2: Gọi hàm refresh

body: RefreshIndicator(  
key: \_refreshIndicatorKey,  
onRefresh: \_refresh,  
child: ListView(children: [  
// body of above  
])),

### Bước 3: Tạo hàm \_refresh

Future<List<GroupEventRP>> \_refresh() {

return getListEvent();

}

Hàm này có tác dụng lấy lại dữ liệu và setState. Hàm này nên trả về dữ liệu cần reload.

getLisstEvent là hàm lấy dữ liệu đầu vào, trả lại Future<List<GroupEventRP>>

## Chuyển các màn hình

<https://medium.com/flutter-community/flutter-push-pop-push-1bb718b13c31>

## Làm tròn double

double doubleFormat(double n, int i) {  
 String a = n.toStringAsFixed(n.truncateToDouble() == n ? 0 : i);  
 **return** double.*parse*(a);  
}

## Lấy imei máy

<https://stackoverflow.com/questions/45031499/how-to-get-unique-device-id-in-flutter?fbclid=IwAR30-SvR_PtG9gjiGAUBwPPae-ph9Q8bw0PFJBWB0z4cRcuOurU5n-2iwhE>

## Đo khoảng cách từ 2 điểm

Đo thủ công

double calculateDistance(lat1, lon1, lat2, lon2){  
 **var** p = 0.017453292519943295;  
 **var** c = cos;  
 **var** a = 0.5 - c((lat2 - lat1) \* p)/2 +  
 c(lat1 \* p) \* c(lat2 \* p) \*  
 (1 - c((lon2 - lon1) \* p))/2;  
 **return** 12742 \* asin(sqrt(a));  
}

Hàm sẵn

<https://pub.dev/packages/geolocator>

**import** 'package:geolocator/geolocator.dart';

double distanceInMeters = **await** Geolocator().distanceBetween(52.2165157, 6.9437819, 52.3546274, 4.8285838);

lưu ý: <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

## Không lồng được 2 scroll vào với nhau

Xử lý <https://medium.com/flutterpub/flutter-listview-gridview-inside-scrollview-68b722ae89d4>

Có thể sử dụng Stack để chúng đè lên nhau sau đó dùg Align để setup vị trí của các widget

## Lỗi không build được con khi cha đang build

Thay đổi cách gọi ontap: function() thành ontap: ()=> function()

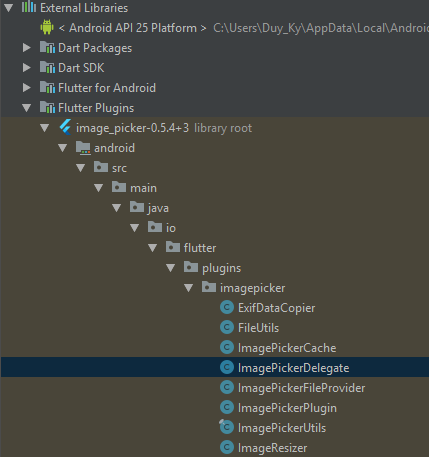
<https://stackoverflow.com/questions/44485760/flutter-cannot-build-because-the-frawework-is-already-building>

## Lỗi không lấy được ảnh qua camera trên thiết bị android

Thay đổi thư viện image\_picker trên

## [Show/hide widgets in Flutter programmatically](https://stackoverflow.com/questions/44489804/show-hide-widgets-in-flutter-programmatically)

<https://stackoverflow.com/questions/44489804/show-hide-widgets-in-flutter-programmatically/53044562#53044562>



<https://github.com/jason-simmons/flutter_plugins/commit/b13d9f633679f062e7fc6dc0e4b58aa9fa4de139#diff-732868cbc0c9d243da485cc537dc6a17>

không hiện ảnh

<https://github.com/flutter/flutter/issues/41346>

Image\_picker không load được asset

Unable to load asset:

I/flutter (14743): /storage/emulated/0/Android/data/com.example.flutter\_php\_100\_noandroidx/files/Pictures/0c5107da-3479-46b1-b91e-82d389356e951409381529.jpg

## Index trong gridview bắt đầu từ 1

## Khó lấy labelText từ textfield

<http://www.bytebau.com/flutter-textfield-how-to/>

## Lỗi [The getter 'length' was called on null](https://stackoverflow.com/questions/51008506/nosuchmethoderror-the-getter-length-was-called-on-null)

Có 2 khả năng xảy ra nếu bạn có sử dụng widget.usernames.length.

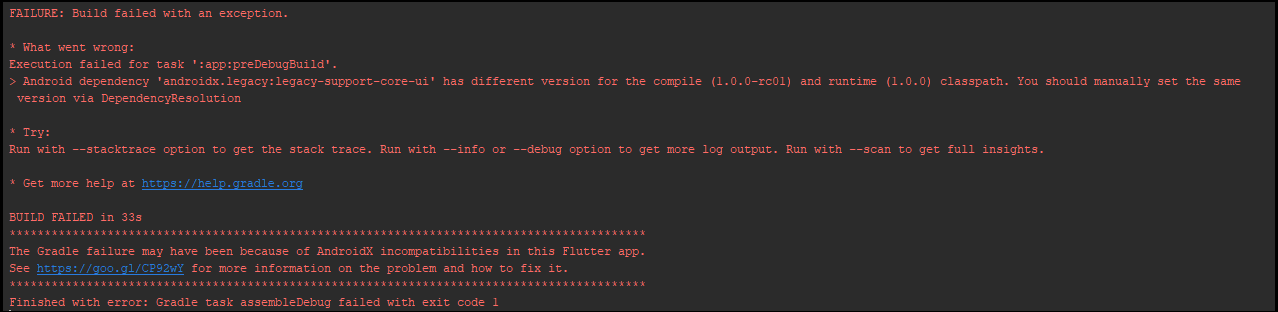
Hoặc là widget = null

Hoặc là widget.usernames = null

Vì thế bạn cần check null từ widget đến widget.usernames

Mọi kiểu biến đều cần check null kể cả k phải list hay mảng

## Lỗi khi dùng thư viện yêu cầu AndroidX

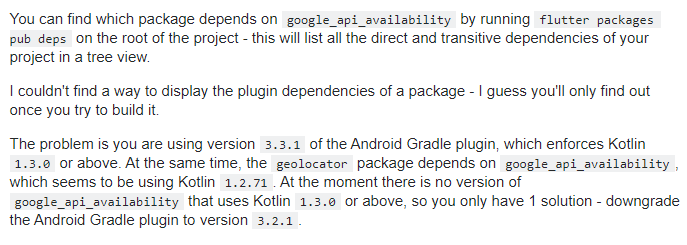


Làm đúng theo hướng dẫn

<https://pub.dev/packages/location>

Xung đột phiên bản gradle và phiên bản kotlin

<https://stackoverflow.com/questions/54693595/flutter-the-android-gradle-plugin-supports-only-kotlin-gradle-plugin-version-1>



## Khi lỗi flutter the getter length was called on null http receiver

là do tham số chuyển vào http bị null

## Áp dụng các quy tắc lập trình với lint

Hướng dẫn: <https://www.youtube.com/watch?v=fJwwxPBhFqI&list=PLVDJsRQrTUz5ZiT0z_SGvOkrHllexhz72&index=23>

## Chinh noi dung vao giua man hinh

padding: EdgeInsets.symmetric(  
 vertical: 20,  
 horizontal: 20  
),

# Các thư viện thông dụng

## Keyboard Dismisser

ẩn bàn phím khi forcus ngoài textfield

Thư viện: <https://pub.dev/packages/keyboard_dismisser>

## flutter\_libphonenumber

Kiểm tra số điện thoại

<https://pub.dev/packages/flutter_libphonenumber>

# Flutter CLI

# Mixins

# Core design principles

## Dry

<https://medium.com/flutter-community/keeping-flutter-widgets-dry-with-dart-mixins-252cada41217>

## Kiss

## SOLID

# Simple persistence

## Serializer

<https://pub.dev/packages/localstorage>

# Architecture

## Mô hình MVC

## MVP

## MVVP

## BLOC Partern

## Redux

# Dart

## Concurrent

## Reactive

## Functional

## Freezed

<https://pub.dev/packages/freezed>

# Package manager

<https://flutter.dev/docs/development/packages-and-plugins/developing-packages>

## maven

<https://maven.apache.org/>

## Pub

## Popular plugins

# Memory leaks

<https://flutter.dev/docs/perf/rendering/ui-performance>

<https://www.youtube.com/watch?v=vVg9It7cOfY&feature=youtu.be>