**BỘ GIÁO DỤC & ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

--🙢🕮🙠--

**A picture containing text, device, gauge

Description automatically generated**

**BÁO CÁO CÁ NHÂN**

**CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI**

**ĐỀ TÀI**

**TÌM HIỂU FLUTTER**

**Nhóm 17**

3120410467 - Võ Minh Tấn

**Giảng viên phụ trách ThS. PHẠM THI VƯƠNG**

**TP.Hồ Chí Minh, tháng 5 năm 2024**

Mục lục

[1. Tìm hiểu về Widget Tooltip trong Flutter 1](#_Toc165118789)

[2. Tìm hiểu về Widget Slider trong Flutter 3](#_Toc165118790)

[3. Tìm hiểu về Widget Switch trong Flutter 5](#_Toc165118791)

[4. Tìm hiểu về Widget Charts trong Flutter 7](#_Toc165118792)

[5. Tìm hiểu về Widget Bottom Navigation Bar trong Flutter 9](#_Toc165118793)

[6. Tìm hiểu về Widget Theme trong Flutter 10](#_Toc165118794)

[7. Tìm hiểu về Widget Table trong Flutter 11](#_Toc165118795)

[8. Tìm hiểu về Widget Animation trong Flutter 11](#_Toc165118796)

[9. Xây dựng chương trình Flutter cơ bản: Ứng dụng Máy Tính (Calculator App) 12](#_Toc165118797)

1. **Tìm hiểu về Widget Tooltip trong Flutter**

Tooltip là một lớp thiết kế material design trong Flutter **cung cấp các nhãn văn bản để giải thích chức năng** của một nút hoặc hành động trên giao diện người dùng.

**Ưu điểm:** Widget Tooltip trong Flutter là một cách tiện lợi để cung cấp thông tin bổ sung cho các phần tử trong giao diện người dùng. Nó cho phép người dùng xem thông tin mô tả khi họ di chuyển con trỏ của mình qua một phần tử nhất định. Điều này giúp cải thiện trải nghiệm người dùng bằng cách cung cấp thông tin hữu ích mà không làm phiền quá nhiều không gian màn hình.

**Nhược điểm:** Tooltip có thể không phù hợp cho các ứng dụng mà không có sự tương tác trực tiếp với người dùng hoặc không có con trỏ chuột (ví dụ: ứng dụng di động). Ngoài ra, việc sử dụng tooltip quá nhiều có thể làm cho giao diện trở nên rối và gây khó chịu cho người dùng.

|  |
| --- |
| const Tooltip({  super.key,  this.message,  this.richMessage,  this.height,  this.padding,  this.margin,  this.verticalOffset,  this.preferBelow,  this.excludeFromSemantics,  this.decoration,  this.textStyle,  this.textAlign,  this.waitDuration,  this.showDuration,  this.exitDuration,  this.enableTapToDismiss = true,  this.triggerMode,  this.enableFeedback,  this.onTriggered,  this.child,  }) |

Các thuộc tính có trong Tooltip

* **message**: Đây là một thông báo chuỗi được sử dụng để hiển thị trong chú giải công cụ.
* **height** : Được sử dụng để chỉ định chiều cao của con của chú giải công cụ.
* **textStyle** : Được sử dụng để xác định kiểu cho **message**của chú giải công cụ.
* **margin** : Được sử dụng để xác định không gian trống bao quanh chú giải công cụ.
* **showDuration** : Được sử dụng để chỉ định khoảng thời gian hiển thị chú giải công cụ sau khi nhấn giữ lâu được thả ra. Theo mặc định, nó là 1,5 giây.
* **decoration**: Được sử dụng để xác định hình dạng và màu nền của chú giải công cụ. Hình dạng chú giải công cụ mặc định là một hình chữ nhật tròn có bán kính đường viền là 4,0 PX.
* **verticalOffset** : Dùng xác định khoảng cách theo chiều dọc giữa chú giải công cụ và widget con.
* **waitDuration** : Được sử dụng để chỉ định thời gian con trỏ di chuột qua widget của chú giải công cụ trước khi hiển thị chú giải công cụ. Khi con trỏ rời khỏi widget, thông báo chú giải công cụ sẽ biến mất.
* **padding** : Dùng xác định không gian để chèn con của chú giải công cụ. Theo mặc định, nó là 16.0 PX ở tất cả các hướng.
* **likesBelow** : Được sử dụng để chỉ định xem chú giải công cụ có được hiển thị bên dưới widget hay không. Theo mặc định, nó là sự thật. Chú giải công cụ sẽ được hiển thị theo hướng ngược lại nếu chúng ta không có đủ không gian để hiển thị chú giải công cụ theo hướng ưu tiên.

1. **Tìm hiểu về Widget Slider trong Flutter**

Slider trong Flutter là một widget thiết kế material design được sử dụng để chọn một loạt các giá trị. Nó là một widget đầu vào, nơi chúng ta có thể **đặt một loạt các giá trị bằng cách kéo hoặc nhấn vào vị trí mong muốn**.

**Ưu điểm:** Widget Slider trong Flutter cung cấp một cách thuận tiện để người dùng chọn giá trị từ một phạm vi đã cho. Nó linh hoạt và dễ sử dụng, cho phép người dùng điều chỉnh giá trị bằng cách di chuyển thanh trượt. Điều này thích hợp cho các tùy chọn có giá trị liên tục như âm lượng, độ sáng, hoặc giá trị trong một khoảng thời gian.

**Nhược điểm:** Một số nhược điểm của Widget Slider là khi sử dụng trong các ứng dụng di động, đôi khi việc chính xác khi chọn giá trị có thể gây khó khăn cho người dùng, đặc biệt là trên các thiết bị có màn hình nhỏ

|  |
| --- |
| const Slider({  super.key,  required this.value,  this.secondaryTrackValue,  required this.onChanged,  this.onChangeStart,  this.onChangeEnd,  this.min = 0.0,  this.max = 1.0,  this.divisions,  this.label,  this.activeColor,  this.inactiveColor,  this.secondaryActiveColor,  this.thumbColor,  this.overlayColor,  this.mouseCursor,  this.semanticFormatterCallback,  this.focusNode,  this.autofocus = false,  this.allowedInteraction,  }) |

Các thuộc tính có trong Slider

* **value**: Là một đối số bắt buộc và được sử dụng để chỉ định giá trị hiện tại của thanh trượt.
* **onChanged:** Là một đối số bắt buộc và được gọi trong quá trình kéo khi người dùng chọn một giá trị mới cho thanh trượt. Nếu nó là null, thanh trượt sẽ bị vô hiệu hóa.
* **onChangedStart**: Là một đối số tùy chọn và được gọi khi chúng ta bắt đầu chọn một giá trị mới.
* **max**: Là một đối số tùy chọn và xác định giá trị lớn nhất có thể được sử dụng bởi người dùng. Theo mặc định, nó là 1.0. Nó phải lớn hơn hoặc bằng min.
* **min**: Là một đối số tùy chọn xác định giá trị tối thiểu có thể được sử dụng bởi người dùng. Theo mặc định, nó là 0,0. Giá trị này phải nhỏ hơn hoặc bằng giá trị tối đa.
* **divisions**: Được xác định số lần phân chia rời rạc. Nếu nó là null, thanh trượt là liên tục.
* **label**: Dùng chỉ định nhãn văn bản sẽ được hiển thị phía trên thanh trượt. Nó hiển thị giá trị của một thanh trượt rời rạc.
* **activeColor**: Dùng xác định màu của phần hoạt động của rãnh trượt.
* **inactiveColor**: Dùng xác định màu của phần không hoạt động của rãnh trượt.
* **SemanticFormatterCallback**: Là một lệnh gọi lại được sử dụng để tạo ra một giá trị context. Theo mặc định, nó là một tỷ lệ phần trăm.

1. **Tìm hiểu về Widget Switch trong Flutter**

Switch là một phần tử giao diện người dùng hai trạng thái được sử dụng để chuyển đổi giữa các trạng thái **BẬT (Đã kiểm tra) hoặc TẮT (Bỏ chọn).**

**Ưu điểm:** Widget Switch trong Flutter là một cách thuận tiện để người dùng chuyển đổi giữa hai trạng thái - Bật hoặc Tắt. Nó cung cấp một cách đơn giản và trực quan để thực hiện các tùy chọn hoặc cài đặt trong ứng dụng.

**Nhược điểm:** Switch có thể không phù hợp cho các tùy chọn có nhiều trạng thái hơn hai, vì nó chỉ hỗ trợ hai trạng thái. Ngoài ra, trên một số thiết bị hoặc giao diện người dùng, Switch có thể không phù hợp về mặt thẩm mỹ hoặc không thích hợp cho không gian màn hình hạn chế.

Code mẫu để tạo một Switch trên màn hình

|  |
| --- |
| class MyApp extends StatelessWidget {  @override  Widget build(BuildContext context) {  return MaterialApp(  home: Scaffold(  appBar: AppBar(  backgroundColor: Colors.blue,  title: Text("Custom Switch Example"),  ),  body: Center(  child: SwitchScreen()  ),  )  );  }  }    class SwitchScreen extends StatefulWidget {  @override  SwitchClass createState() => new SwitchClass();  }    class SwitchClass extends State {  bool isSwitched = false;  @override  Widget build(BuildContext context) {  return Column(  mainAxisAlignment: MainAxisAlignment.center,  children:<Widget>[  CustomSwitch(  value: isSwitched,  activeColor: Colors.blue,  onChanged: (value) {  print("VALUE : $value");  setState(() {  isSwitched = value;  });  },  ),  SizedBox(height: 15.0,),  Text('Value : $isSwitched', style: TextStyle(color: Colors.red,  fontSize: 25.0),)  ]);  }  } |

Các thuộc tính có trong Switch

* **onChanged**: Được gọi bất cứ khi nào người dùng chạm vào công tắc.
* **value**: Chứa giá trị Boolean true hoặc false để kiểm soát xem chức năng của công tắc đang BẬT hay TẮT.
* **activeColor**: Được sử dụng để chỉ định màu của công tắc bóng tròn khi nó BẬT.
* **activeTrackColor**: Chỉ định màu thanh chuyển hướng.
* **inactiveThubmColor**: Được sử dụng để chỉ định màu của công tắc bóng tròn khi nó TẮT.
* **inactiveTrackColor**: Chỉ định màu thanh công tắc khi nó TẮT.
* **dragStartBehavior**: Nó đã xử lý hành vi bắt đầu kéo. Nếu chúng ta đặt nó là DragStartBehavior.start, thì thao tác kéo sẽ di chuyển công tắc từ bật sang tắt.

1. **Tìm hiểu về Widget Charts trong Flutter**

Charts trong Flutter được dùng để**biểu diễn dữ liệu theo cách đồ họa** cho phép người dùng hiểu chúng một cách đơn giản.

**Ưu điểm**: Widget Charts giúp hiển thị dữ liệu một cách trực quan, dễ hiểu. Nó cung cấp một loạt các loại biểu đồ như đường, cột, tròn, v.v., để phù hợp với nhu cầu hiển thị dữ liệu khác nhau. Widget này hỗ trợ tùy chỉnh cao, cho phép bạn điều chỉnh các yếu tố như màu sắc, kích thước, và hiệu ứng của biểu đồ.

**Nhược điểm**: Một số biểu đồ có thể phức tạp và khó hiểu nếu không được trình bày một cách rõ ràng. Điều này có thể làm giảm khả năng tiếp cận của người dùng đối với dữ liệu. Ngoài ra, việc tùy chỉnh cao có thể đòi hỏi kiến thức kỹ thuật và thời gian..

Flutter chủ yếu hỗ trợ ba loại biểu đồ và mỗi biểu đồ đi kèm với một số tùy chọn cấu hình. Sau đây là biểu đồ được sử dụng trong ứng dụng Flutter:

**LineChart**:

* + **LineChartData**: Lớp này chứa dữ liệu và cấu hình cho biểu đồ dạng đường.
    - **lineBarsData**: Danh sách các dữ liệu biểu đồ, mỗi dữ liệu biểu đồ là một đường trên biểu đồ.
    - **titlesData**: Thông tin về các tiêu đề của biểu đồ như tiêu đề trục x, trục y, và các chú thích.
    - **borderData**: Dữ liệu liên quan đến đường biên của biểu đồ như viền và màu sắc.
  + **LineChartBarData**: Dữ liệu của một đường trên biểu đồ dạng đường.
    - **spots**: Danh sách các điểm trên đường biểu đồ, mỗi điểm có tọa độ x và y.
    - **isCurved**: Xác định xem đường có được vẽ cong hay không.
    - **colors**: Màu sắc của đường biểu đồ.
    - **barWidth**: Độ dày của đường biểu đồ.

**BarChart**:

* + **BarChartData**: Lớp này chứa dữ liệu và cấu hình cho biểu đồ cột.
    - **barGroups**: Danh sách các nhóm cột trên biểu đồ.
    - **titlesData**: Thông tin về các tiêu đề của biểu đồ như tiêu đề trục x, trục y, và các chú thích.
    - **borderData**: Dữ liệu liên quan đến đường biên của biểu đồ như viền và màu sắc.
  + **BarChartGroupData**: Dữ liệu của một nhóm cột trên biểu đồ cột.
    - **x**: Vị trí trên trục x của nhóm cột.
    - **barRods**: Danh sách các cột trong nhóm.
    - **showingTooltipIndicators**: Danh sách các chỉ số của các cột hiển thị tooltip khi được chạm vào.

**PieChart**:

* + **PieChartData**: Lớp này chứa dữ liệu và cấu hình cho biểu đồ hình tròn.
    - **sections**: Danh sách các phần của biểu đồ hình tròn.
    - **centerSpaceRadius**: Bán kính của phần trống ở giữa biểu đồ.
    - **borderData**: Dữ liệu liên quan đến đường biên của biểu đồ như viền và màu sắc.
  + **PieChartSectionData**: Dữ liệu của một phần trên biểu đồ hình tròn.
    - **value**: Giá trị của phần trên biểu đồ.
    - **color**: Màu sắc của phần trên biểu đồ.
    - **radius**: Bán kính của phần trên biểu đồ.

1. **Tìm hiểu về Widget Bottom Navigation Bar trong Flutter**

Bottom Navigation Bar trong Flutter **có thể chứa nhiều mục** như nhãn văn bản, biểu tượng hoặc cả hai. Dùng để diều hướng ứng dụng.

**Ưu điểm**: Widget Bottom Navigation Bar cung cấp một cách tiện lợi để điều hướng giữa các màn hình chính trong ứng dụng. Nó cho phép người dùng truy cập nhanh chóng đến các phần chính của ứng dụng mà không cần phải thực hiện nhiều thao tác. Bottom Navigation Bar thường được sử dụng trong các ứng dụng di động để tạo ra một trải nghiệm điều hướng dễ sử dụng và trực quan.

**Nhược điểm**: Bottom Navigation Bar không phù hợp cho các ứng dụng có số lượng màn hình chính lớn hoặc cần tích hợp nhiều tính năng phụ. Nếu không được thiết kế cẩn thận, nó có thể gây ra sự cồng kềnh hoặc khó sử dụng cho người dùng.

Các thuộc tính có trong Bottom Navigation Bar

* **items:** Nó xác định danh sách để hiển thị trong thanh điều hướng dưới cùng. Nó sử dụng đối số BottomNavigationBarItem có chứa các thuộc tính sup được đưa ra bên dưới:

|  |
| --- |
| const BottomNavigationBarItem({  @required this.icon,  this.title,  Widget activeIcon,  this.backgroundColor,  }) |

* **currentIndex:** Nó xác định mục thanh điều hướng dưới cùng đang hoạt động hiện tại trên màn hình.
* **onTap:** Nó được gọi khi chúng ta chạm vào một trong các mục trên màn hình.
* **iconSize:** Nó được sử dụng để chỉ định kích thước của tất cả các biểu tượng mục điều hướng phía dưới.
* **fixedColor:** Nó được sử dụng để đặt màu của mục đã chọn. Nếu chúng ta chưa đặt màu cho biểu tượng hoặc tiêu đề, nó sẽ được hiển thị.
* **type:** Nó xác định bố cục và hành vi của thanh điều hướng dưới cùng. Nó hoạt động theo hai cách khác nhau, đó là: **fixed**và shifting. Nếu nó là null, nó sẽ sử dụng fixed. Nếu không, nó sẽ sử dụng tính năng chuyển đổi nơi chúng ta có thể xem hoạt ảnh khi chúng ta nhấp vào một nút.

1. **Tìm hiểu về Widget Theme trong Flutter**

Theme là các gói đặt trước chứa các**giao diện đồ họa trên** trang web hoặc màn hình ứng dụng dành cho thiết bị di động của chúng ta.

**Ưu điểm**: Widget Theme cho phép bạn tùy chỉnh giao diện của ứng dụng một cách dễ dàng và linh hoạt. Bạn có thể định nghĩa các thuộc tính giao diện chung một lần và áp dụng chúng cho nhiều widget trong toàn bộ ứng dụng. Điều này giúp giảm thiểu việc lặp lại code và giữ cho code của bạn dễ bảo trì.

**Nhược điểm**: Mặc dù Widget Theme là một công cụ mạnh mẽ, nhưng đôi khi việc tùy chỉnh giao diện có thể phức tạp và đòi hỏi hiểu biết sâu rộng về Flutter và cách làm việc với chủ đề.

**Cấu trúc:** Widget Theme trong Flutter được sử dụng bên trong widget tree để áp dụng các thuộc tính giao diện nhất định cho một phần hoặc toàn bộ ứng dụng. Một Widget Theme bao gồm một hoặc nhiều thuộc tính giao diện được định nghĩa bởi ThemeData.

1. **Tìm hiểu về Widget Table trong Flutter**

Widget Table trong Flutter là một cách để hiển thị dữ liệu dưới dạng bảng, với hàng và cột.

**Ưu điểm**: Widget Table cung cấp một cách tiện lợi để hiển thị dữ liệu dưới dạng bảng trong ứng dụng Flutter. Nó cho phép bạn sắp xếp và hiển thị dữ liệu một cách cấu trúc, giúp người dùng dễ dàng đọc và tìm kiếm thông tin.

**Nhược điểm**: Widget Table có thể không phù hợp cho việc hiển thị dữ liệu lớn hoặc phức tạp, vì nó có thể gây ra sự cồng kềnh và tốn nhiều không gian trên màn hình.

Widget Table trong Flutter có cấu trúc đơn giản, bao gồm một danh sách các TableRow, mỗi TableRow chứa một danh sách các TableCell. Mỗi TableCell có thể chứa một widget, thường là Text hoặc một widget khác để hiển thị dữ liệu. Ví dụ đoạn code như sau:

|  |
| --- |
| TableRow(children: [  TableCell(child: Text('javatpoint')),  TableCell(  child: Text('Flutter'),  ),  TableCell(child: Text('Android')),  TableCell(child: Text('MySQL')),  ]), |

1. **Tìm hiểu về Widget Animation trong Flutter**

Animation trong Flutter là một cách mạnh mẽ để tạo ra các hiệu ứng chuyển động và thay đổi trong giao diện người dùng của ứng dụng.

**Ưu điểm**: Animation trong Flutter cho phép bạn tạo ra các hiệu ứng chuyển động mượt mà và hấp dẫn trong ứng dụng của bạn. Nó giúp cải thiện trải nghiệm người dùng và tạo ra giao diện động hơn, từ việc di chuyển đối tượng đến việc thay đổi màu sắc và kích thước của chúng.

**Nhược điểm**: Việc sử dụng Animation trong Flutter có thể phức tạp và đòi hỏi hiểu biết về cách làm việc với các loại AnimationController, Tween, và AnimatedBuilder. Nếu không được xử lý cẩn thận, việc sử dụng animation có thể dẫn đến code khó hiểu và khó bảo trì.

**Cấu trúc:** Animation trong Flutter thường được xây dựng từ ba phần chính:

* AnimationController: Quản lý quá trình chạy của animation, bao gồm việc điều khiển thời gian và tốc độ của animation.
* Tween: Xác định các giá trị đầu vào và đầu ra của animation, cho phép bạn xác định các thay đổi trong animation.
* AnimatedBuilder hoặc các widget tương tự: Sử dụng để tạo ra các hiệu ứng dựa trên animation, bằng cách cập nhật giao diện của widget theo giá trị animation được cung cấp.

1. **Xây dựng chương trình Flutter cơ bản: Ứng dụng Máy Tính (Calculator App)**

Máy tính (Calculator) - Một trong những ứng dụng cơ bản giúp cho những lập trình viên có thể làm quen với cấu trúc giao diện khi mới bắt đầu xây dựng những dự án mới, flutter cũng không phải ngoại lệ. Sau khi xây dựng xong chương trình này những gì mà ta đạt được là cách cấu trúc thư mục, tổ chức code, sử dụng chức năng, thư viện, framework một cách hợp lý.

**Cấu trúc chương trình**

Do là một chương trình đơn giản và chỉ thao tác trên một màn hình duy nhất. Cho nên cấu trúc thư mục của chương trình cũng đơn giản và dễ hiễu, cây thư mục của chương trình được biểu thị như sau:

A screenshot of a computer

Description automatically generated

Hình 3. . Cây thư mục chương trình Máy tính (Calculator App)

Trong đây, ta có thể thấy cây thư mục được chia ra làm 3 folder chính được mô tả như sau:

* **screens:** Là thư mục chứa các màn hình của chương trình. Ở đây ta chỉ thực hiện trên một màn hình duy nhất nên chỉ có một màn hình *home\_screen.dart*.
  + **home\_screen.dart:** File này chứa code của màn hình chính của app.
* **utils:** Là thư mục chứa các hàm và lớp tiện ích của ứng dụng. Có nghĩa là mỗi lần cần sử dụng các hàm build-in sẵn ta chỉ cần gọi chúng lại thay vì phải code lại như từ đầu gây mất thời gian và quá nhiều code trong chương trình.
  + **calc.dart:** File này chứa các hàm tính toán.
  + **const.dart:** File này chứa các hằng số của ứng dụng và một số hàm getter nhanh
* **widgets:** Thư mục này chứa các widget của ứng dụng. Mỗi widget được biểu hiện thành 1 File.
  + **button\_number.dart:** File này chứa code biển thị các nút bấm số của máy tính.
  + **show\_result\_area.dart:** File này chứa code hiển thị khu vực kết quả sau quá trình nhập và tính toán

Việc phân chia và tách code ra thành các thư mục sẽ giúp cho chương trình dễ quản lý code cũng như góp phần tái sử dụng lại code cho những phần code sau.