**CÔNG TY CÔNG NGHỆ & KỸ THUẬT STECH**

**VIỆN NGHIÊN CỨU & SÁNG TẠO KHOA HỌC MÁY TÍNH – IRICS**

**A logo with orange letters

Description automatically generated**

**BÁO CÁO NGHIÊN CỨU FLUTTER**

**TASK 7 – NGHIÊN CỨU FLUTTER**

Người hướng dẫn**: MR. NGUYỄN TẤN HÙNG ANH**

**MR. VÕ THÀNH LUÂN**

Người thực hiện**: ĐINH NGỌC AN THƠ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**TÓM TẮT**

Flutter là một framework mã nguồn mở do Google phát triển, dùng để xây dựng các ứng dụng đa nền tảng (iOS, Android, Web, và Desktop) từ một cơ sở mã nguồn duy nhất. Ra mắt lần đầu vào tháng 5 năm 2017, Flutter nhanh chóng trở thành một công cụ phổ biến nhờ khả năng giúp lập trình viên tiết kiệm thời gian và công sức khi phát triển ứng dụng cho nhiều nền tảng. Flutter sử dụng ngôn ngữ lập trình Dart, cũng do Google phát triển, nổi bật với khả năng biên dịch nhanh và hiệu suất cao.

Một trong những điểm mạnh của Flutter là khả năng tạo ra giao diện người dùng đẹp và linh hoạt. Nó cung cấp một bộ widget phong phú và có thể tùy chỉnh, giúp các nhà phát triển dễ dàng xây dựng giao diện ứng dụng theo ý muốn. Flutter cũng hỗ trợ tính năng "hot reload", cho phép lập trình viên xem ngay lập tức các thay đổi trong mã nguồn mà không cần khởi động lại toàn bộ ứng dụng, từ đó tăng hiệu quả phát triển.

Flutter còn nổi bật với hiệu suất cao nhờ việc sử dụng đồ họa Skia, giúp ứng dụng chạy mượt mà và hiển thị đồ họa phức tạp một cách dễ dàng. Hệ sinh thái của Flutter cũng rất phong phú với nhiều thư viện và plugin hỗ trợ, giúp mở rộng chức năng của ứng dụng một cách linh hoạt. Nhờ các ưu điểm này, Flutter đã được nhiều công ty và lập trình viên trên toàn thế giới tin dùng để xây dựng các ứng dụng di động và đa nền tảng.

**MỤC LỤC**

[CHƯƠNG 1: SERVICES SOURCE WEATHERAPP 5](#_Toc169792934)

[I. location.dart 5](#_Toc169792935)

[II. networking.dart 6](#_Toc169792936)

[III. weather.dart 7](#_Toc169792937)

[CHƯƠNG 2: SERVICES SOURCE FLUTTER\_TASK\_7 9](#_Toc169792938)

[I. screens 9](#_Toc169792939)

[II. loading\_screen.dart 10](#_Toc169792940)

[III. location\_screen.dart 12](#_Toc169792941)

[IV. ultilities 14](#_Toc169792942)

[CHƯƠNG 3: TỔNG HỢP 17](#_Toc169792943)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

[Hình 1 : file location.dart 5](#_Toc169793116)

[Hình 2 : file networking.dart 6](#_Toc169793117)

[Hình 3: file location\_screen.dart 7](#_Toc169793118)

[Hình 4: cityscreen 10](#_Toc169793119)

[Hình 5: loading\_screen.dart 11](#_Toc169793120)

[Hình 6: location\_screen.dart 13](#_Toc169793121)

[Hình 7: constanst.dart của file ultilities 15](#_Toc169793122)

[Hình 8: main của project task 7 15](#_Toc169793123)

CHƯƠNG 1: SERVICES SOURCE WEATHERAPP

1. location.dart

**A screen shot of a computer

Description automatically generated**

Hình 1 : file location.dart

-import package ‘geolocator’ : lệnh import package ‘geolocator’ cho phép truy cập vào các chức năng liên quan đến vị trí của thiết bị

- class ‘Location’ : một class có hai thuộc tính ‘latitude*’* và ‘longtitude’ để lưu trữ thông tin vị trí

- Method ‘getCurrentLocation()’ : asynchronous method (phương thức không đồng bộ) dùng để lấy vị trí hiện tại của thiết bị

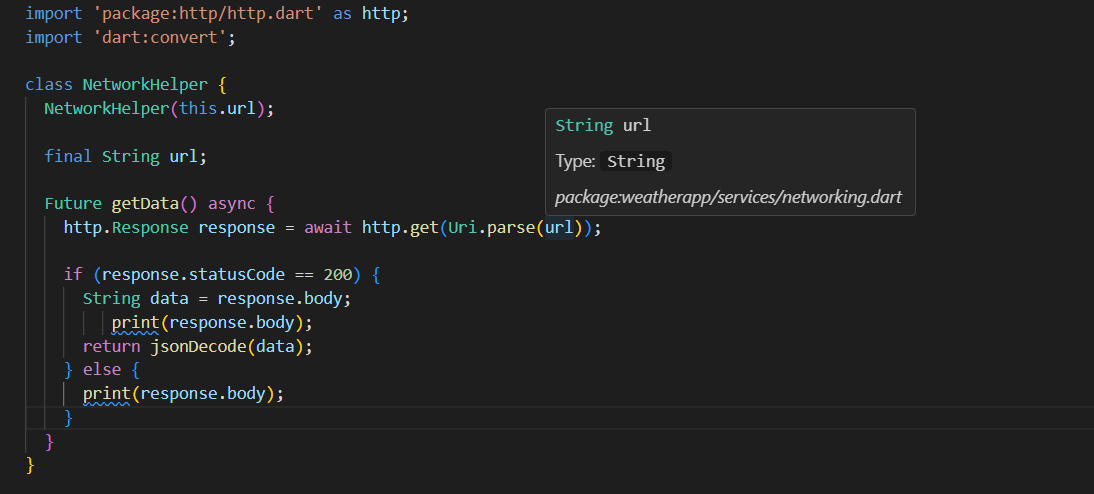
- ‘try-catch’ : block này được sử dụng để bắt và xử lý các exception ( nếu có) trong quá trình lấy vị trí

‘await’ : để đợi kết quả trả về từ ‘Geolocator.getCurrentPosition()’. Phương thức này trả về một đối tượng ‘Position’ chứa thông tin vị trí

‘Geolocator.getCurrentPosition(desiredAccuracy : LocationAccuracy.Low)’ :yêu cầu với độ chính xác ‘low’ (độ chính xác thấp nhất)

Và sau khi có kết quả từ ‘getCurrentPosition’, lấy giá trị ‘ latitude’ và ‘longtitude’ từ ‘Position’ và gán vào các thuộc tính tương ứng của ‘Location’. Nếu có lỗi xảy ra trong quá trình lấy vị trí, thông váo lỗi được in ra màn hình

1. networking.dart

****

Hình 2 : file networking.dart

-import ‘package:http/http.dart’ as http:’ : Import package ‘http’ và đặt tên là ‘http’

-import ‘dart:convert’ : import thư viện ‘dart:convert’ để sử dụng hàm ‘jsonDecode’

- class ‘NetworkHelper’ : lớp ‘netWorkHelper’ có một constructor nhận một tham số ‘url’, đại diện cho URL mà coder muốn gửi yêu cầu HTTP tới

- thuộc tính ‘final String url’ dùng để lưu giữ URL được truyền vào

-Method ‘getData()’ : là một phương thước không đồng bộ (‘async’)

-‘await http.get(Uri.parse (url));’ : Gửi yêu cầu HTTP GET đến URL được cung cấp ‘http.get()’ trả về một ‘Future’ của đối tượng ‘Response’. Vì vậy sử dụng ‘await’ để đợi cho đến khi yêu cầu hoàn tất và có kết quả trả về

-‘http.Response response =…’ : Dùng để lưu trữ kết quả từ yêu cầu vào biến ‘response’

-‘if (response.statusCode ==200) {…} : Kiểm tra nếu mã trạng thái là 200 thì hoàn thành :

‘String data = response.body’ : Lấy nội dung phản hồi ( dữ liệu trả về từ server) và lưu vào biến ‘ data,

‘return jsonDecode(data);’ : Decode dữ liệu JSON từ chuỗi ‘data’ sử dụng ‘jsonDecode()’ và trả về kết quả. Phương thức này trả về một đối tượng Dart có kiểu dữ liệu tương ứng với JSON

- Nếu mã trạng thái không phải 200 (‘else’ brach’ :

‘print(response.body)’ : In nội dung phản hồi ra màn hình để debug ( hoặc xử lý lỗi theo nhu cầu )

1. weather.dart

****

**A screen shot of a computer program

Description automatically generated**

Hình 3: file location\_screen.dart

- import các file cần thiết từ package ‘weatherapp’ chứa các service như ‘location.dart’ và ‘networking.dart’

- Khai báo const ‘apiKey’ và ‘openWeatherMapURL’ để sử dụng trong các yêu cầu API. ‘apiKey’ là khóa API của coder từ OpenWeatherMap, ‘openWeatherMapURL’ là ỦL của API để lấy thông tin thời tiêt

- Class ‘WeatherModel’:

+ Phương thức ‘getCityWeather(String cityName)’ :

Nhận vào một tham số ‘cityName’ là tên thành phố

Tạo một đối tượng ‘Network Helper’ và gửi yêu cầu GET tới API của OpenWeatherMap để lấy thông tin thời tiết cho thành phố được chỉ định

Và sau khi nhận được dữ liệu từ API (‘weatherData’), nó sẽ được trả về dưới dạng một ‘Future<dynamic>’ , trong đó ‘dynamic’ là dữ liệu được giải mã từ JSON

+ Phương thức ‘getLocationWeather()’ :

Tạo một đối tượng’Location’ và sử dụng phương thức ‘getCurrentLocation()’ để lấy vị trí hiện tại của thiết bị

Tạo ‘Network’ và gửi yêu cầu GET tới API của OpenWeatherMap, sử dụng vị trí lấy được (‘latitude’ và ‘longtitude’), để lấy thông tin thời tiết hiện tại

Tương tự ‘getCityWeather()’, phương thức này cũng trả về một ‘Future<dynamic>’ chứa dữ liệu thời tiết

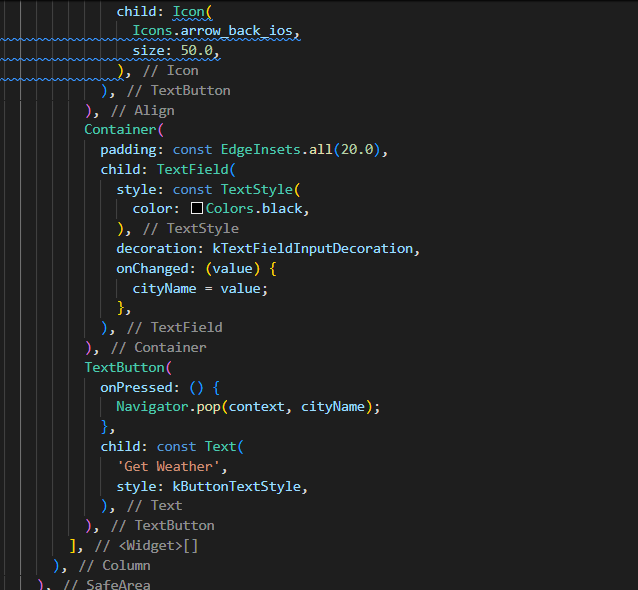
+ Phương thức ‘getMessage (int temp) : Dựa vào nhiệt độ ‘temp’, phương thức này trả về một message phù hợp để mô tả trạng thái thời tiết. (vd “ nếu nhiệt độ cao hơn 25 độ ‘Một ngày thật đẹp để uống trà sữa’

CHƯƠNG 2: SERVICES SOURCE FLUTTER\_TASK\_7

1. screens
2. **city\_screen.dart in**

**A screen shot of a computer program

Description automatically generated**

****

Hình 4: cityscreen

*-*  import 'package:flutter/material.dart';: Import các thành phần của Flutter để xây dựng UI.

- import 'package:flutter\_task\_7/utilities/constants.dart';: Import file chứa các hằng số (constants), như kiểu trang trí cho TextField và TextButton.

- CityScreen là một StatefulWidget, bởi vì nó có trạng thái thay đổi được (state) - cụ thể là cityName có thể thay đổi khi người dùng nhập vào TextField.

- ‘CityScreenState’ là trạng thái của CityScreen, nơi cityName được khai báo.

- Phương thức build:

* Scaffold: Một widget cung cấp cấu trúc cơ bản cho màn hình, bao gồm app bar, body, và các thuộc tính khác.
* Container: Được sử dụng để bao bọc và trang trí cho màn hình.
* BoxDecoration: Trang trí nền của Container bằng một hình ảnh (city\_backgrounds.jpg) và làm cho hình ảnh phủ kín toàn bộ Container.
* SafeArea: Đảm bảo rằng nội dung không bị che khuất bởi các phần tử hệ thống như notch hay thanh trạng thái.
* Column: Sắp xếp các widget con theo chiều dọc.
* Align: Canh chỉnh widget con, trong trường hợp này là TextButton, sang góc trên trái.
* TextButton: Một nút không có nền, chỉ chứa biểu tượng mũi tên để quay lại màn hình trước đó.
* Container: Bao bọc TextField với padding.
* TextField: Widget nhập liệu cho người dùng để nhập tên thành phố.
* style: Đặt màu chữ của TextField.
* decoration: Sử dụng hằng số kTextFieldInputDecoration từ constants.dart để trang trí TextField.
* onChanged: Callback khi nội dung của TextField thay đổi, gán giá trị mới cho cityName.
* TextButton: Một nút khác để lấy thông tin thời tiết khi người dùng nhấn vào.
* onPressed: Callback khi nút được nhấn, trả về cityName và quay lại màn hình trước đó thông qua Navigator.pop.

1. loading\_screen.dart

Hình**A screen shot of a computer program

Description automatically generated** 5: loading\_screen.dart

- Import:

* flutter/material.dart: Cung cấp các thành phần UI cơ bản.
* geolocator/geolocator.dart: Cung cấp các chức năng liên quan đến vị trí.
* location\_screen.dart: Màn hình hiển thị thông tin thời tiết.
* flutter\_spinkit/flutter\_spinkit.dart: Gói cung cấp các hiệu ứng tải (loading spinners).

-initState():

* Được gọi ngay khi widget được tạo. Gọi phương thức getLocationData() để bắt đầu quá trình lấy dữ liệu.

-getLocationData():

* Phương thức không đồng bộ để lấy dữ liệu vị trí và thời tiết.
* LocationPermission permission;: Biến để lưu trữ trạng thái quyền truy cập vị trí.
* permission = await Geolocator.requestPermission();: Yêu cầu quyền truy cập vị trí từ người dùng.
* var weatherData = await WeatherModel().getLocationWeather();: Lấy dữ liệu thời tiết hiện tại từ WeatherModel.
* Navigator.push(context, MaterialPageRoute(builder: (context) {...}));: Điều hướng đến LocationScreen và truyền dữ liệu thời tiết (weatherData) vào.

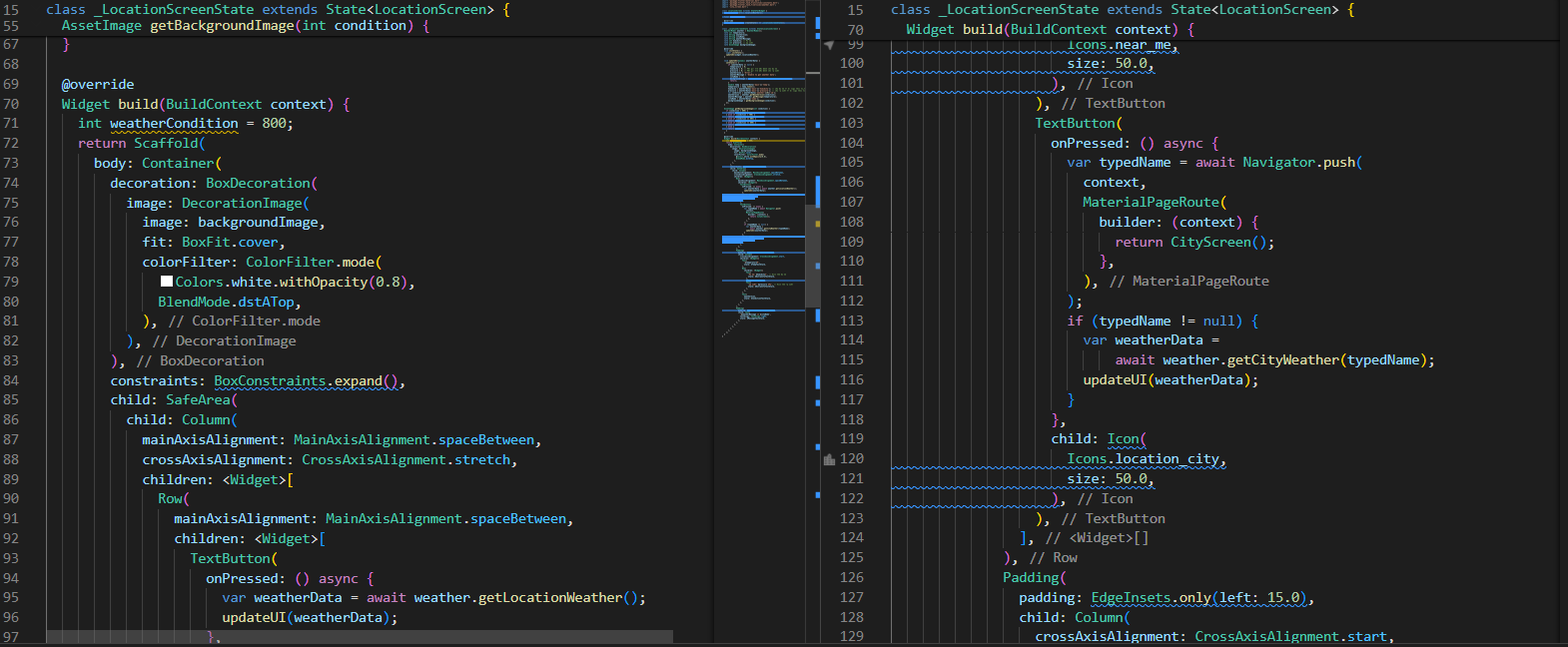
**-**Phương thức ‘build’

* Scaffold: Một widget cung cấp cấu trúc cơ bản cho màn hình.
* Center: Canh giữa nội dung bên trong.
* SpinKitDoubleBounce: Hiệu ứng tải từ flutter\_spinkit, hiển thị một hiệu ứng double bounce để cho người dùng biết rằng dữ liệu đang được tải.

1. location\_screen.dart

**A computer screen shot of a computer program

Description automatically generated**

****

**A screen shot of a computer program

Description automatically generated**

Hình 6: location\_screen.dart

-Khởi tạo và cập nhật dữ liệu thời tiết:

* initState(): Gọi updateUI() để cập nhật thông tin thời tiết khi màn hình khởi tạo.
* updateUI(dynamic weatherData): Cập nhật các thông tin thời tiết như nhiệt độ, độ ẩm, áp suất, biểu tượng thời tiết, thông điệp thời tiết, và hình nền dựa trên dữ liệu thời tiết nhận được.

-Lấy và hiển thị dữ liệu thời tiết:

* Gọi WeatherModel().getLocationWeather() để lấy dữ liệu thời tiết hiện tại theo vị trí.
* Cập nhật UI với dữ liệu thời tiết nhận được.
* Cho phép người dùng nhập tên thành phố khác để lấy thông tin thời tiết cho thành phố đó.

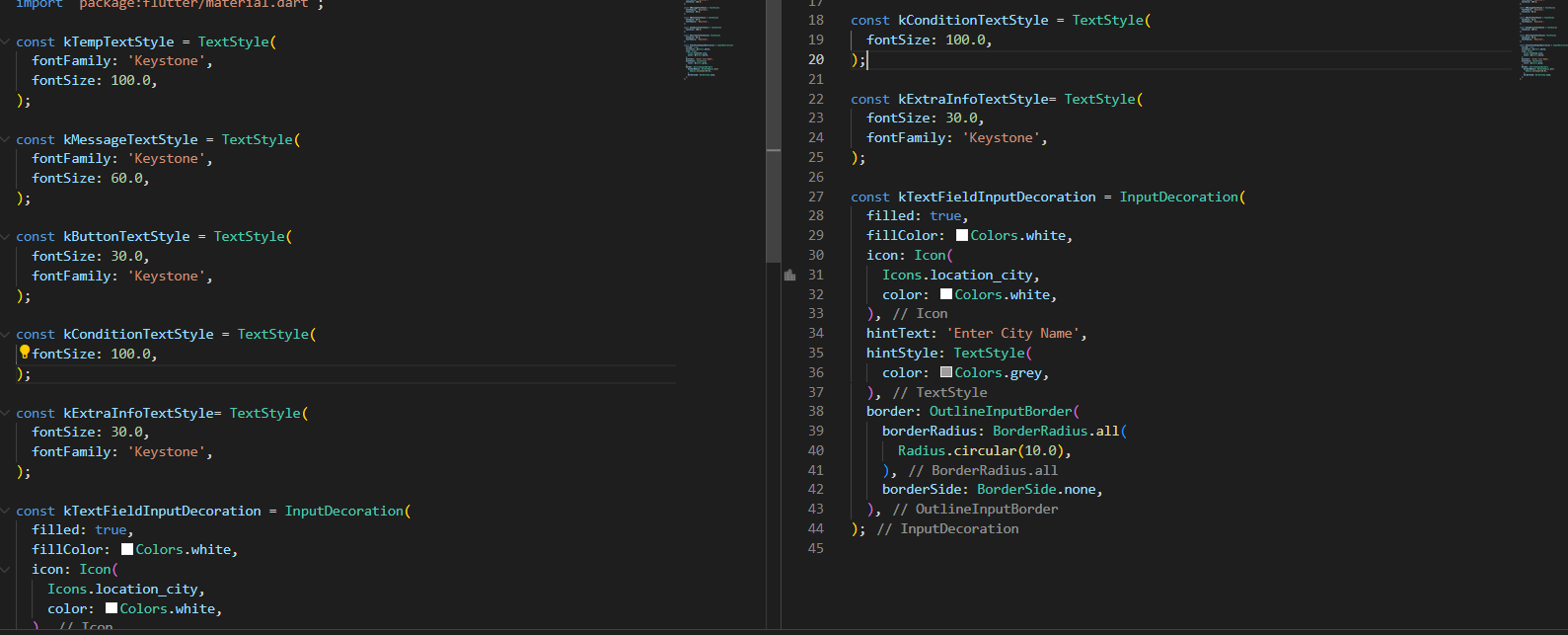
-Hiển thị UI:

* Hiển thị nhiệt độ, độ ẩm, áp suất, biểu tượng thời tiết, và thông điệp thời tiết.
* Hình nền thay đổi dựa trên điều kiện thời tiết.

-Nút tương tác:

* Nút để cập nhật thời tiết theo vị trí hiện tại.
* Nút để nhập tên thành phố mới và lấy thông tin thời tiết cho thành phố đó.

1. ultilities

* **constants.dart**

Hình 7: constanst.dart của file ultilities

1. **main**

**A screen shot of a computer program

Description automatically generated**

Hình 8: main của project task 7

- import 'package:flutter/material.dart'; : Import các thành phần cần thiết từ Flutter.

- import 'package:flutter\_task\_7/screens/loading\_screen.dart' : Import màn hình tải dữ liệu.

- Hàm main khởi chạy ứng dụng bằng cách gọi runApp(MyApp());

- MyApp kế thừa StatelessWidget và định nghĩa giao diện của ứng dụng.

- Trong phương thức build, trả về một MaterialApp với theme tối và màn hình chính là LoadingScreen.

CHƯƠNG 3: TỔNG HỢP

1. Cách sử dụng ‘geolocator’ trong Flutter để lấy vị trí hiện tại của thiết bị và lưu vào đối tượng ‘Location’ thông qua phương thức ‘getCurrentLocation()’.
2. ‘NetworkHelper’ là một lớp dùng để thực hiện yêu cầu HTTP GET tới một URL cụ thể và trả về dữ liệu JSON từ phản hồi . Sử dụng package ‘http’ để thực hiện yêu cầu mạng và ‘ dart:convert’ để xử lý dữ liệu JSON . class có thể được sử dụng để tạo các yêu cầu mạng và xử lý dữ liệu trả về trong ứng dụng Flutter
3. cách sử dụng service để lấy dữ liệu thời tiết từ OpenWeatherMap API và xử lý dữ liệu để hiển thị thông tin thời tiết và biểu tượng thời tiết tương ứng
4. nhập tên thành phố trong ứng dụng Flutter nơi người dùng có thể nhập tên thành phố và yêu cầu thông tin thời tiết cho thành phố đó
5. Xây dựng một màn hình nhập tên thành phố trong ứng dụng Flutter. Người dùng có thể nhập tên thành phố vào TextField, sau đó nhấn nut ‘Get Weather’ để gửi tên thành phố về màn hình trước đó, nơi có thể được sử dụng để lấy và hiển thị thông tin thời tiết.
6. Cách tạo màn hình tải dữ liệu trong Flutter, nơi ứng dụng sẽ lấy dữ liệu vị trí và thời tiết hiện tại, sau đó điều hướng đến màn hình hiển thị thông tin thời tiêt
7. Tạo giao diện Theme tối và khởi động bằng màn hình ‘LoadingScreen’, nơi dữ liệu vị trí và thời tiết sẽ được tải và hiển thị