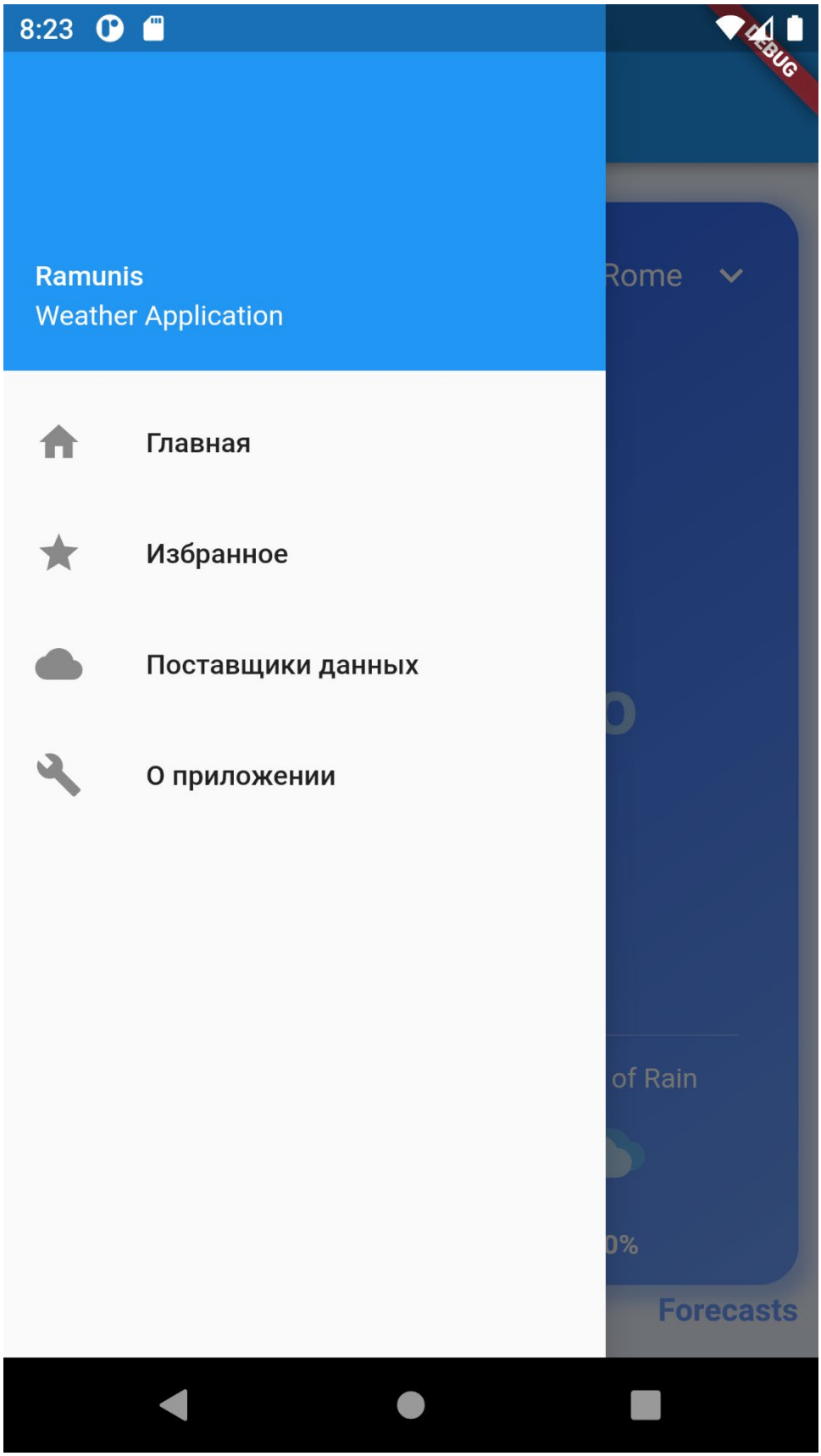
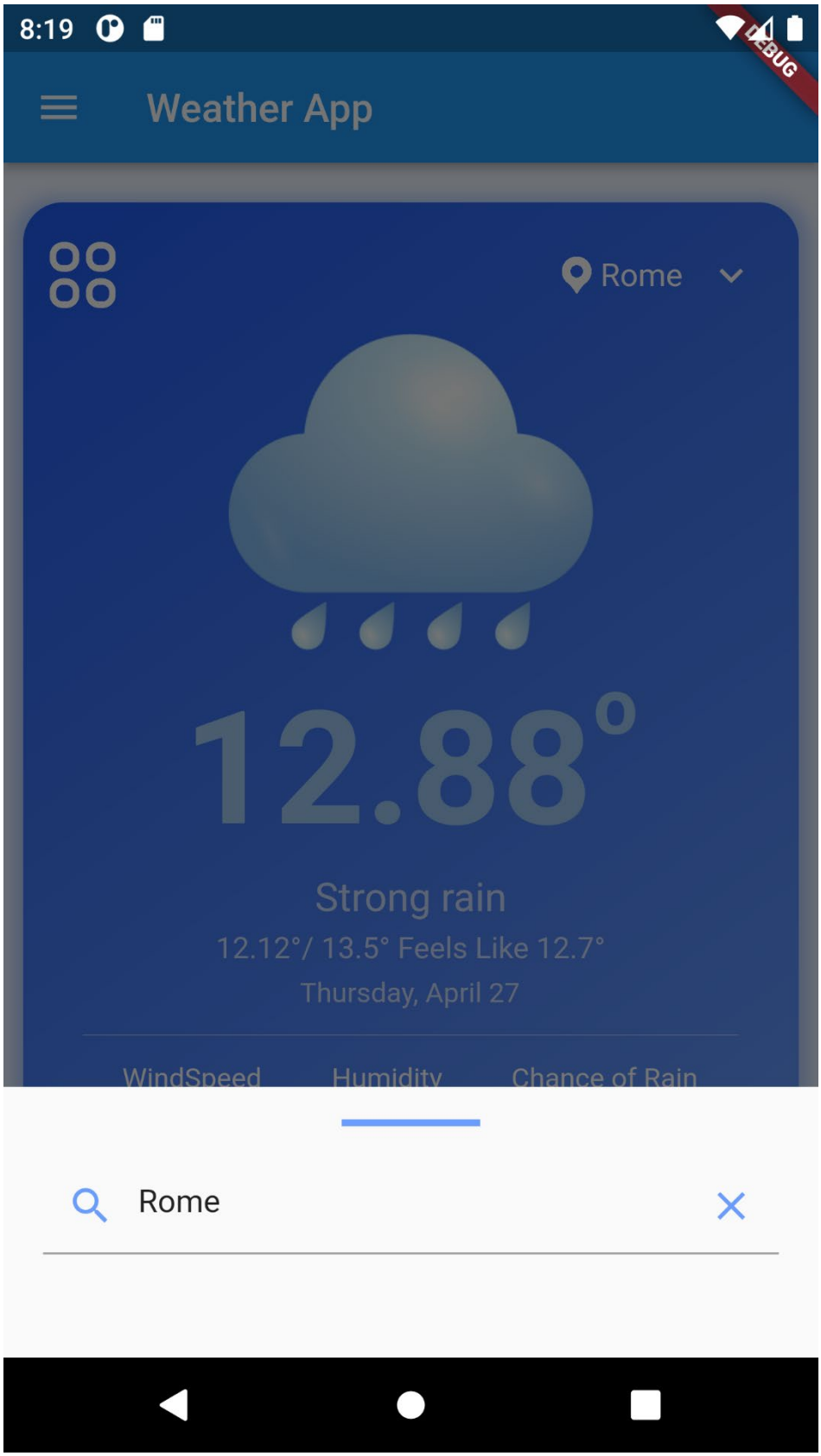
Обзор приложения

Меню

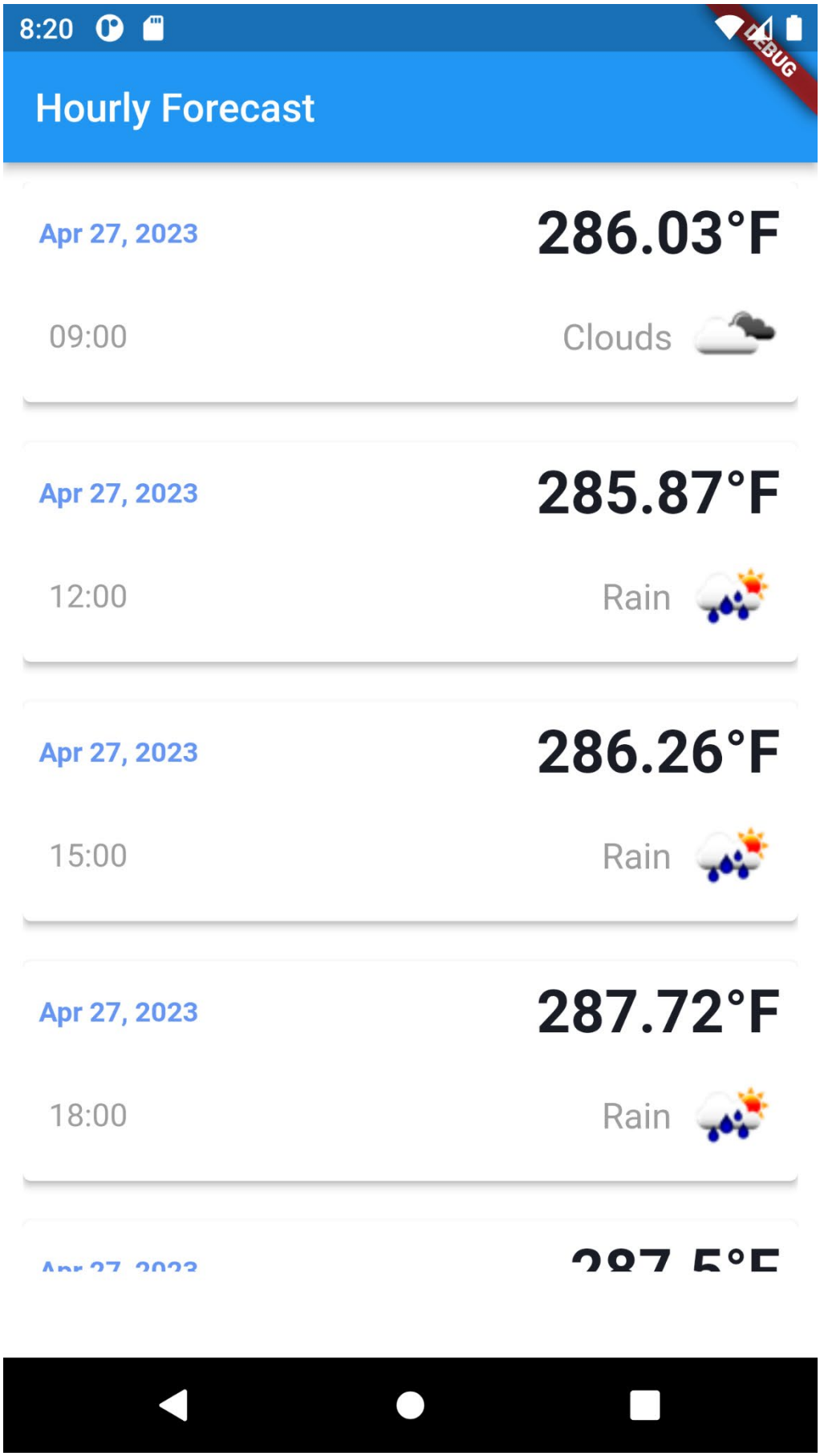


Экран 1) Базовая погода по Геолокации.

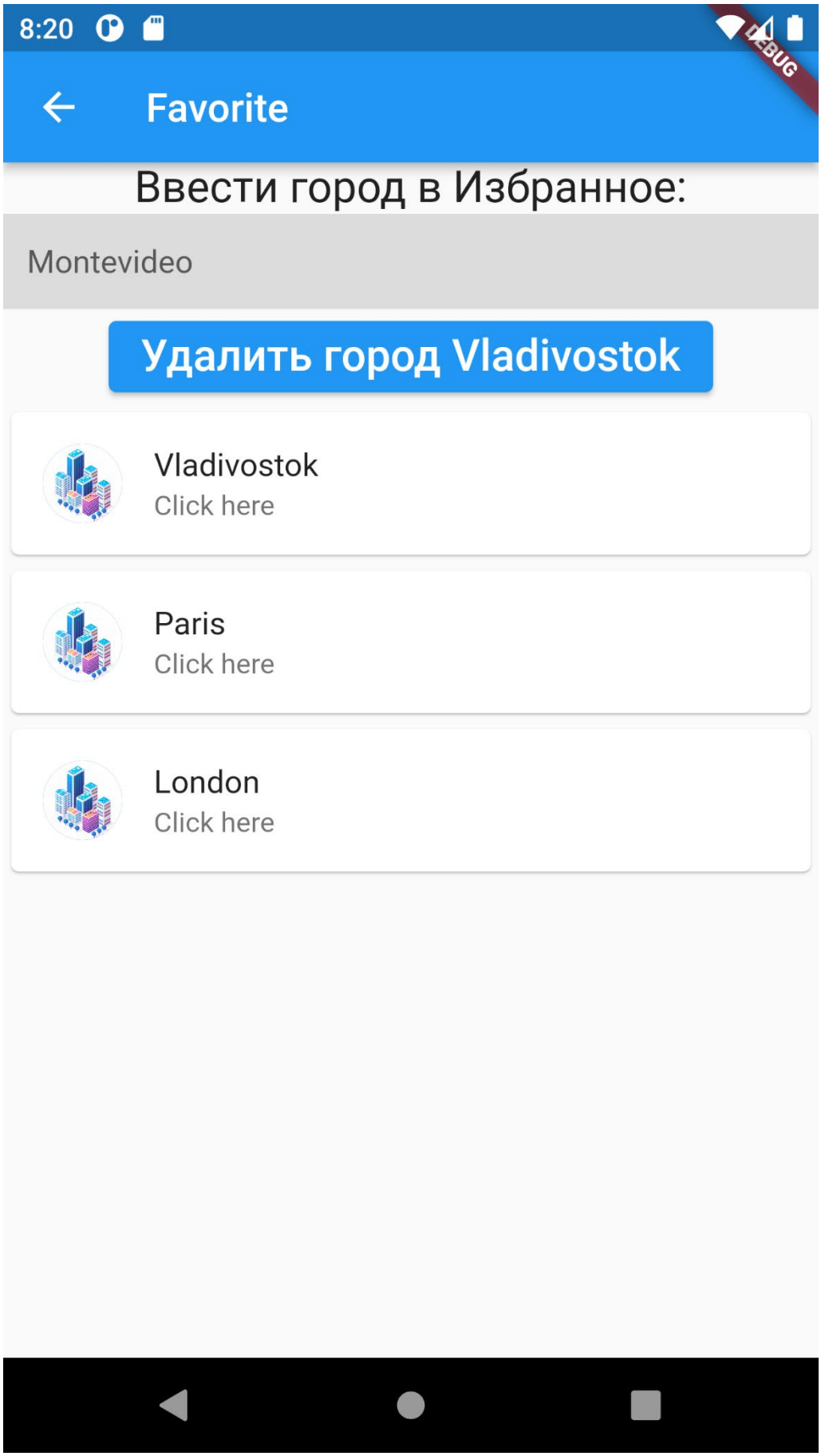
Базовая погода по запрошенному городу



Экран 2) Почасовая погода (переход по Forecast)

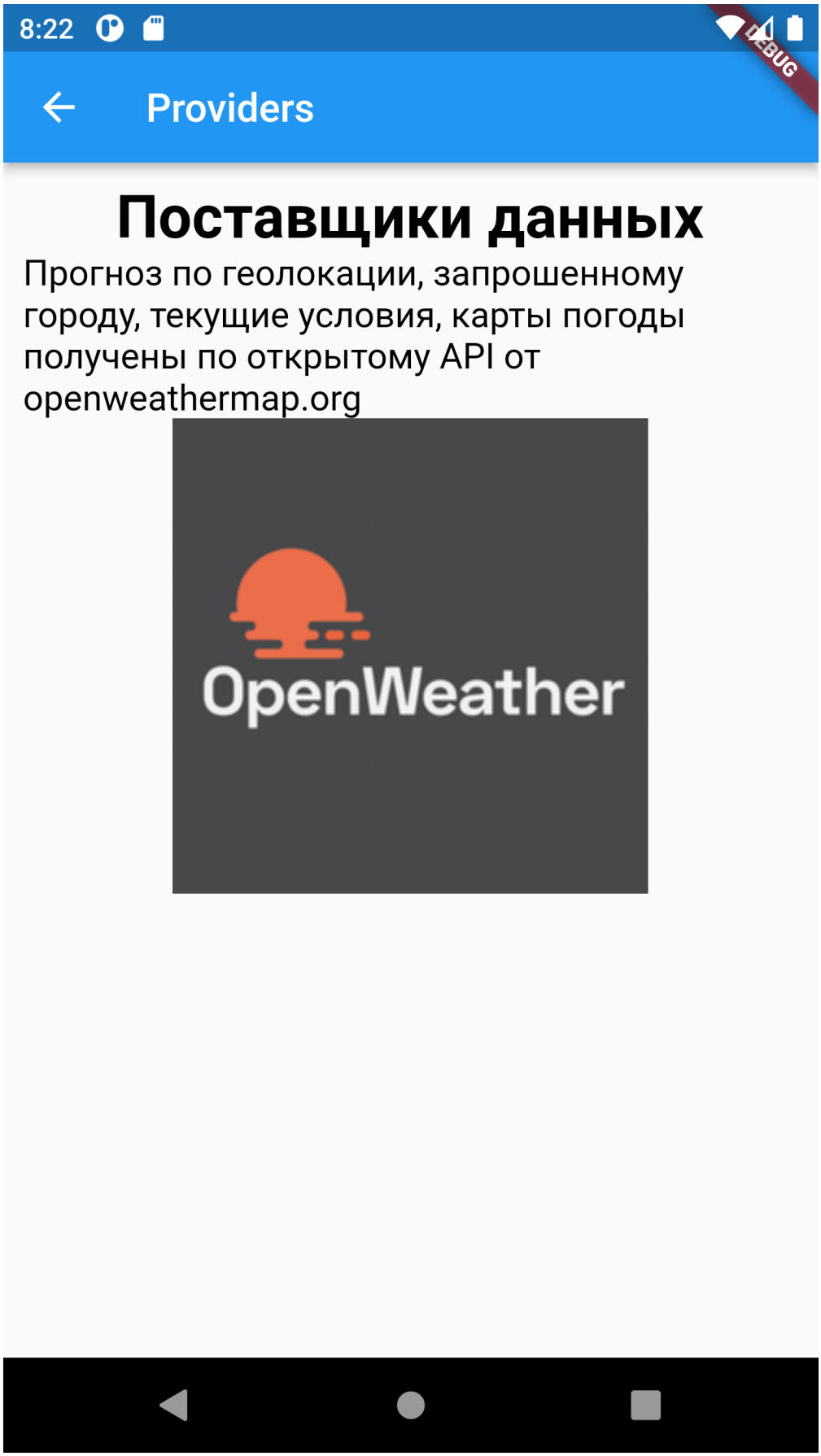


Экран 3) Избранное

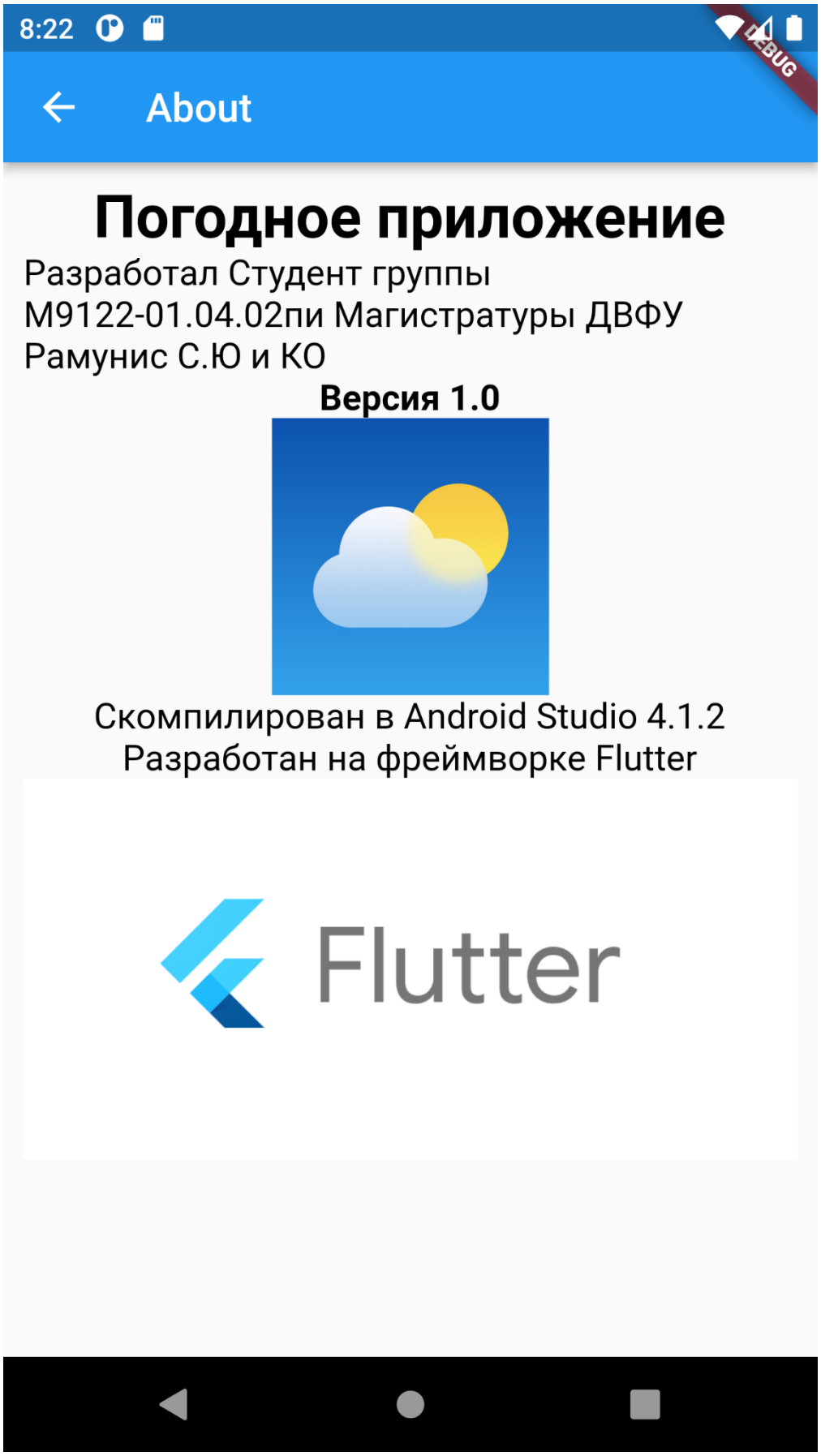


Экран 4) Переход в Избранный город

Экран 5) Поставщики данных

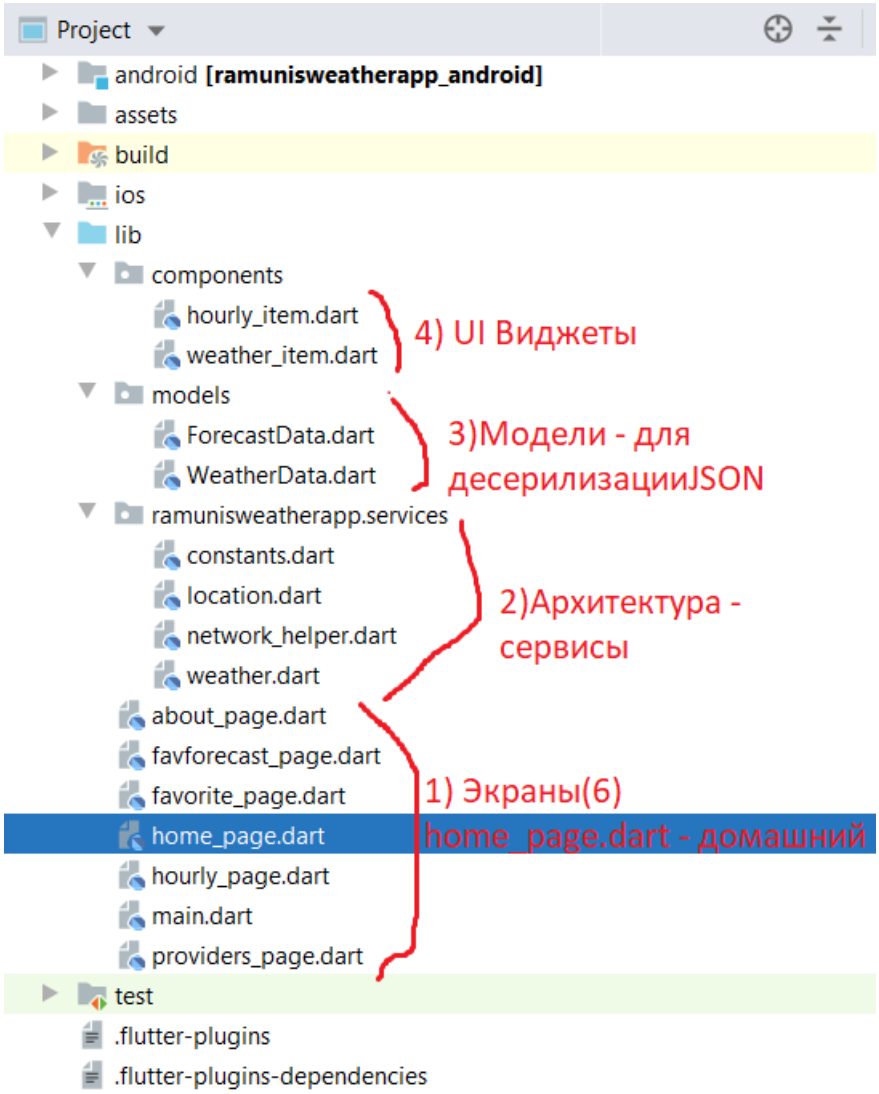


Экран 6) О приложении



Обзор кода

0) Структура проекта и библиотеки



pubspec.yaml

Flutter commands

```yaml
# The following defines the version and
# A version number is three numbers sepa
# followed by an optional build number s
# Both the version and the builder numbe
# build by specifying --build-name and -
# In Android, build-name is used as vers
# Read more about Android versioning at
# In iOS, build-name is used as CFBundle
# Read more about iOS versioning at
# https://developer.apple.com/library/ar
# In Windows, build-name is used as the
# of the product and file versions while
version: 1.0.0+1

environment:
  sdk: '>=2.19.4 <3.0.0'

# Dependencies specify other packages th
# To automatically upgrade your package
# consider running `flutter pub upgrade
# dependencies can be manually updated b
# the latest version available on pub.de
# versions available, run `flutter pub o
dependencies:
  flutter:
    sdk: flutter

  http: ^0.13.5
  geolocator: ^9.0.2
  intl: ^0.17.0
  modal_bottom_sheet: ^2.0.1
```

4) Подвиджеты – код GUI

1.weather_item.dart используется в home_page.dart для отображения WindSpeed, Humidity, Chance of Rain в Базовой погоде

home_page.dart × weather_item.dart ×

'Pub get' has not been run Get dependencies

```dart
import 'package:flutter/material.dart';

class WeatherItem extends StatelessWidget {
  final String text;
  final String value;
  final String unit;
  final String imageUrl;

  const WeatherItem({
  super.key, required this.text, required this.value, required this.unit, required this.imageUrl,
  });

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text(text,style: const TextStyle(
          color: Colors.white,
          fontSize: 14,
        ),), // TextStyle, Text
        Container(
          padding: const EdgeInsets.all( 10),
          height: 60,
          width: 60,
          decoration: BoxDecoration(
```

2. hourly_item.dart используется в hourly_page.dart как элемент списка ListView для отображения по часовой погоды

hourly_item.dart × hourly_page.dart ×

'Pub get' has not been run

```dart
import 'dart:ui';

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:ramunisweatherapp/ramunisweatherapp/services/constants.dart';
import 'package:ramunisweatherapp/models/WeatherData.dart';

class WeatherItem extends StatelessWidget {
  final WeatherData weather;

  const WeatherItem(
      {super.key, required this.weather});

  @override
  Widget build(BuildContext context) {
    final Constants _constants = Constants();
    return Card(
      elevation: 3.0,
      margin: const EdgeInsets.only(bottom: 20),
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            Row(
```

3) Модели – нужны для дессериализации JSON массива, для отображения почасовой погоды в hourly_page.dart

Здесь используется ООП

ForecastData.dart × WeatherData.dart ×

'Pub get' has not been run Get dependencies Upgrade dependenc

```dart
class WeatherData {
  final DateTime date;
  final String name;
  final double temp;
  final String main;
  final String icon;

  WeatherData({required this.date, required this.name, required this.temp, required this.main, required this.icon})

  factory WeatherData.fromJson(Map<String, dynamic> json) {
    return WeatherData(
      date: new DateTime.fromMillisecondsSinceEpoch(json['dt'] * 1000, isUtc: false),
      name: json['name'],
      temp: json['main']['temp'].toDouble(),
      main: json['weather'][0]['main'],
      icon: json['weather'][0]['icon'],
    );  // WeatherData
  }
}
```

ForecastData.dart × WeatherData.dart ×

'Pub get' has not been run Get

```dart
import 'package:ramunisweatherapp/models/WeatherData.dart';

class ForecastData {
  final List list;

  ForecastData({required this.list});

  factory ForecastData.fromJson(Map<String, dynamic> json) {
    List list = [];

    for (dynamic e in json['list']) {
      WeatherData w = new WeatherData(
          date: new DateTime.fromMillisecondsSinceEpoch(e['dt'] * 1000, isUtc: false),
          name: json['city']['name'],
          temp: e['main']['temp'].toDouble(),
          main: e['weather'][0]['main'],
          icon: e['weather'][0]['icon']);  // WeatherData
      list.add(w);
    }

    return ForecastData(
      list: list,
    );
  }
}
```

2)Архитектура – здесь реализован функционал приложения.

1.Constants.dart – реализует графические эффекты для погодных виджетов.

constants.dart

'Pub get' has not been run

```dart
import 'package:flutter/material.dart';

class Constants {
  final primaryColor = const Color(0xff6b9dfc);
  final secondaryColor = const Color(0xffa1c6fd);
  final tertiaryColor = const Color(0xff205cf1);
  final blackColor = const Color(0xff1a1d26);

  final greyColor = const Color(0xffd9dadb);

  final Shader shader = const LinearGradient(
    colors: <Color>[Color(0xffABCFF2), Color(0xff9AC6F3)],
  ).createShader(const Rect.fromLTWH(0.0, 0.0, 200.0, 70.0));

  final linearGradientBlue =  const LinearGradient(
      begin: Alignment.bottomRight,
      end: Alignment.topLeft,
      colors: [Color(0xff6b9dfc), Color(0xff205cf1)],
      stops: [0.0,1.0]
  );  // LinearGradient
  final linearGradientPurple =  const LinearGradient(
      begin: Alignment.bottomRight,
      end: Alignment.topLeft,
      colors: [Color(0xff51087E), Color(0xff6C0BA9)],
      stops: [0.0,1.0]
  );  // LinearGradient
}
```

2.network_helper.dart – описывает HTTP запрос и возвращает JSON.Decode данные.

network_helper.dart

'Pub get' has not been run

```dart
import 'package:http/http.dart' as http;
import 'dart:convert';

class NetworkHelper {
  NetworkHelper(this.url);

  final String url;
  /*метод для упрощения http запросов*/
  Future getData() async {
    var response = await http.get(Uri.parse(url));

    if (response.statusCode == 200) {
      String data = response.body;

      return jsonDecode(data);
    } else {
      print(response.statusCode);
    }
  }
}
```

3.location.dart – возвращает текущую геолокацию пользователя.

location.dart

'Pub get' has not been run

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

class Location {
  late double latitude;
  late double longitude;
/*Метод для получения геолокации*/
  Future<void> getCurrentLocation() async {
    try {
      Position position = await Geolocator
          .getCurrentPosition(desiredAccuracy: LocationAccuracy.low);

      latitude = position.latitude;
      longitude = position.longitude;
    } catch (e) {
      print(e);
    }
  }
}
```

4.weather.dart – содержит все сетевые запросы и исполняет их асинхронно. Здесь используется многопоточность.

weather.dart

'Pub get' has not been run Get dependencies

```dart
import 'location.dart';
import 'network_helper.dart';
const apiKey = 'a0fff5baeda20538ce0c29c3b57c2209';//'79ab925ad5beea3f1600bb1279bca6a8'; //ключ ap
/*Ссылки для работы с API*/
const apiUrl = 'http://api.openweathermap.org/data/2.5/weather';
const apiUrlHourly = 'http://api.openweathermap.org/data/2.5/forecast';

class Weather {
  /*Запрос для получения данных относительно названия города*/
  static Future<dynamic> getCityWeather(String cityName) async {
    NetworkHelper networkHelper = NetworkHelper(
        '$apiUrl?q=$cityName&appid=$apiKey&units=metric'
    );
    var weatherData = await networkHelper.getData();
    return weatherData;
  }
  /*Запрос для получения данных относительно геолокации*/
  static Future<dynamic> getLocationWeather() async {
    Location location = Location();
    await location.getCurrentLocation();
    print(location.latitude);
    print(location.longitude);

    NetworkHelper networkHelper = NetworkHelper(
        '$apiUrl?lat=${location.latitude}&lon=${location.longitude}&appid=$apiKey&units=metric'
    );
    var weatherData = await networkHelper.getData();
    return weatherData;
  }
```

1)Экраны – только основные.

1.Home_page.dart – домашний экран с погодой

Переменные для хранения информации

home_page.dart × hourly_page.dart ×

'Pub get' has not been run

```
17
18  class HomePage extends StatefulWidget {
19    static final String id = 'TestHomePage';
20
21    @override
22    _HomePageState createState() => _HomePageState();
23  }
24
25  class _HomePageState extends State<HomePage> {
26     final TextEditingController _cityController = TextEditingController();
27     final Constants _constants = Constants();
28     late String _cityName;
29     late String _windSpeed;
30     late String _humidity;
31     late String _cloud;
32     late String _feelslike;
33     var _weatherCode;
34     late String _temp;
35     late String _tempmin;
36     late String _tempmax;
37     bool _flag=false;
38     var bulkData;
39
40     String currentDate = '';
41
```

Вызов Погоды по геолокации в Инициализации

```
42     @override
43     void initState() {
44       super.initState();
45       _getWeatherDataByLocation();
46       DateTime now = DateTime.now();
47       var newDate = DateFormat('MMMMEEEEd').format(now);
48       currentDate = newDate;
49     }
```

Функции для получения погоды и записи её в переменные

```
51  void _getWeatherDataByLocation() async {
52    var data = await Weather.getLocationWeather();
53    _cityName = data['name'];
54    _temp = data['main']['temp'].toString();
55    _tempmin = data['main']['temp_min'].toString();
56    _tempmax = data['main']['temp_max'].toString();
57    _windSpeed = data['wind']['speed'].toString();
58    _humidity = data['main']['humidity'].toString();
59    _cloud = data['clouds']['all'].toString();
60    _feelslike = data['main']['feels_like'].toString();
61    _weatherCode = data['weather'][0]['id'];
62    setState(() {
63      _flag = true;
64    });
65  }
66  void _getWeatherData(String city) async {
67    var data = await Weather.getCityWeather(city);
68    _temp = data['main']['temp'].toString();
69    _tempmin = data['main']['temp_min'].toString();
70    _tempmax = data['main']['temp_max'].toString();
71    _windSpeed = data['wind']['speed'].toString();
72    _humidity = data['main']['humidity'].toString();
73    _cloud = data['clouds']['all'].toString();
74    _feelslike = data['main']['feels_like'].toString();
75    _weatherCode = data['weather'][0]['id'];
76    setState(() {
77      _flag = true;
78    });
79  }
```

Всё остальное код GUI

```
242          Container(
243            padding: const EdgeInsets.symmetric(horizontal: 40),
244            child: Row(
245              mainAxisAlignment: MainAxisAlignment.spaceBetween,
246              children: [
247              WeatherItem(
248                text: "WindSpeed",
249                value: _windSpeed,
250                unit: 'km/h',
251                imageUrl: 'assets/windspeed.png',
252              ),  // WeatherItem
253              WeatherItem(
254                text: "Humidity",
255                value: _humidity,
256                unit: '%',
257                imageUrl: 'assets/humidity.png',
258              ),  // WeatherItem
259              WeatherItem(
260                text: "Chance of Rain",
261                value: _cloud,
262                unit: '%',
263                imageUrl: 'assets/cloud.png',
264              ),  // WeatherItem
265              ],
266            ),  // Row
267          ),  // Container
```

2.fav_forecast.page – переход в избранный город

Всё тоже самое, кроме

home_page.dart × hourly_page.dart × favforecast_page.dart ×

'Pub get' has not been run

```
14  class FavforecastPage extends StatefulWidget {
15    static final String id = 'TestFavforecastPage';
16    final fav;
17
18    const FavforecastPage({Key? key, this.fav}) : super(key: key);
19
20    @override
21    _FavforecastPageState createState() => _FavforecastPageState();
22  }
23
```

В переменной fav – храниться город переданный из экрана Избранных городов.

```
41    @override
42    void initState() {
43      super.initState();
44      print(widget.fav);
45      _getWeatherData(widget.fav);
46      _cityName = widget.fav;
47      DateTime now = DateTime.now();
48      var newDate = DateFormat('MMMMEEEEd').format(now);
49      currentDate = newDate;
50    }
```

3.hourly_page.dart – почасовая погода

hourly_page.dart

'Pub get' has not been run

```
1  import 'dart:convert';
2  import 'package:flutter/material.dart';
3  import 'package:http/http.dart' as http;
4
5  import 'components/hourly_item.dart';
6  import 'package:ramunisweatherapp/models/ForecastData.dart';
7
8  class MyApp extends StatefulWidget {
9    final fav;
10
11   const MyApp({Key? key, this.fav}) : super(key: key);
12
13   @override
14   State<StatefulWidget> createState() {
15     return new MyAppState();
16   }
17 }
18
19 class MyAppState extends State<MyApp> {
20   bool isLoading = false;
21   late ForecastData forecastData;
22
23   @override
24   void initState() {
25     super.initState();
26
27     loadWeather(widget.fav);
28   }
29
```

Для почасовой погоды в коде GUI используется ListView

```
30  @override
31  Widget build(BuildContext context) {
32    if (isLoading == true) {
33      return Scaffold(...);  // Scaffold
45    } else
46    return MaterialApp(
47      title: 'Flutter Weather App',
48      theme: ThemeData(
49        primarySwatch: Colors.blue,
50      ),  // ThemeData
51      home: Scaffold(
52          backgroundColor: Colors.greenAccent,
53          appBar: AppBar(
54            title: Text('Hourly Forecast'),
55          ),  // AppBar
56          body: Container(
57              padding: EdgeInsets.all(10),
58              color: Colors.white,
59              child:Column(
60                children: <Widget>[
61                  Container(
62                    height: 550.0,
63                    child: forecastData != null ? ListView.builder(
64                        itemCount: forecastData.list.length,
65                        scrollDirection: Axis.vertical,
66                        itemBuilder: (context, index) => WeatherItem(weather: forecastData.list.elementAt(index))
67                    ) : Container(),  // ListView.builder
68                  ),  // Container
69                ],  // <Widget>[]
70              )  // Column
71          )  // Container
72      ),  // Scaffold
73    );  // MaterialApp
74  }
```

Десериализация JSON массива через Модель

```
76    loadWeather(String city) async {
77      setState(() {
78        isLoading = true;
79      });
80
81      final forecastResponse = await http.get(Uri.parse(
82          'https://api.openweathermap.org/data/2.5/forecast?q=${city
83              .toString()}&APPID=a0fff5baeda20538ce0c29c3b57c2209'));
84
85      if (forecastResponse.statusCode == 200) {
86        return setState(() {
87          forecastData = new ForecastData.fromJson(jsonDecode(forecastResponse.body));
88          isLoading = false;
89        });
90      }
91
92      setState(() {
93        isLoading = false;
94      });
95    }
96  }
97
```