

Programación multimedia y dispositivos móviles

Actividad 4.3. Aplicación para la
previsión meteorológica.

Francisco José García Cutillas | 2FPGS_DAM



Índice

Ejercicio 1	3
-------------------	---

Ejercicio 1

Se pide desarrollar una aplicación para la previsión meteorológica. Para ello, utilizaremos la API Rest ofrecida por la AEMET, que ya trabajamos en el apartado anterior.

Adjunto el fichero con el JSON con los diferentes municipios de España, varias imágenes e iconos para utilizar.

El aspecto de la aplicación será parecido al de la imagen siguiente:



Solución (Primer paso para comenzar.)

Para la solución a este caso práctico, partiremos de la solución al caso práctico anterior, donde ya realizábamos la petición a la API de la AEMET y mostrábamos los resultados por la terminal.

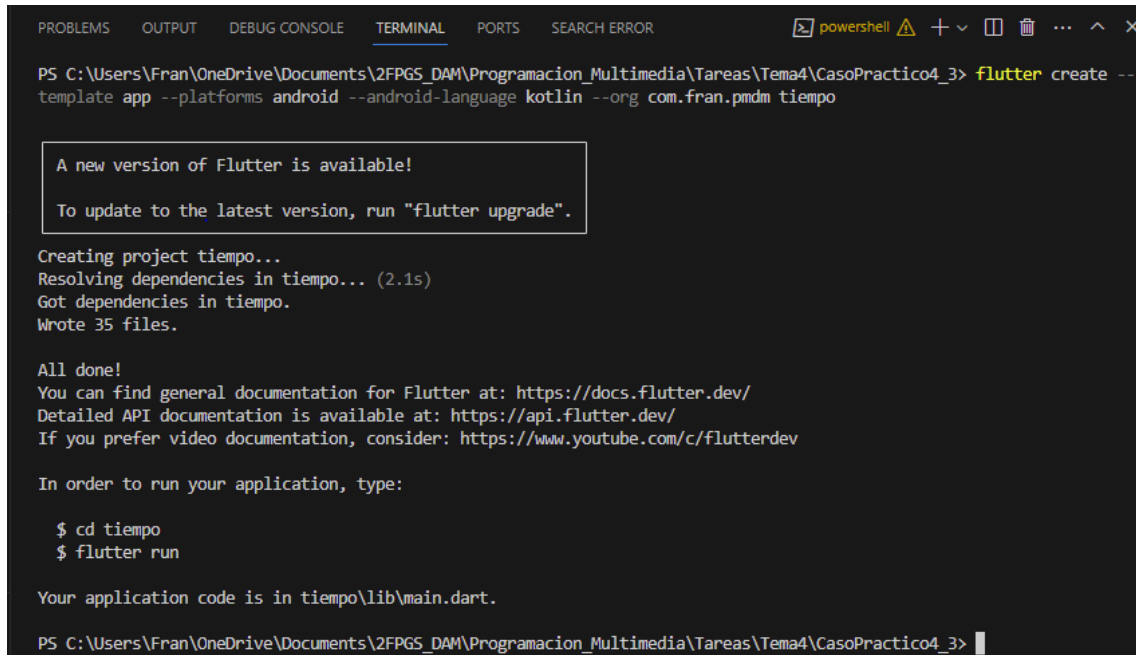
Creación del proyecto y creación de la carpeta de recursos

En primer lugar, creamos el proyecto para Android con:

```
flutter create --template app --platforms android --android-language kotlin --org com.mgh.pmdm mi_tiempo
```

Con el esqueleto de la aplicación, creamos en el directorio raíz del proyecto una carpeta llamada *assets*, que contendrá los recursos de la aplicación: tanto las imágenes como el fichero JSON con las provincias.

Para comenzar ejecutamos el comando “flutter create --template app --platforms android --android-language kotlin --org com.fran.pmdm tiempo”



```
PS C:\Users\Fran\OneDrive\Documents\2FPGS_DAM\Programacion_Multimedia\Tareas\Tema4\CasoPractico4_3> flutter create --template app --platforms android --android-language kotlin --org com.fran.pmdm tiempo

A new version of Flutter is available!

To update to the latest version, run "flutter upgrade".

Creating project tiempo...
Resolving dependencies in tiempo... (2.1s)
Got dependencies in tiempo.
Wrote 35 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

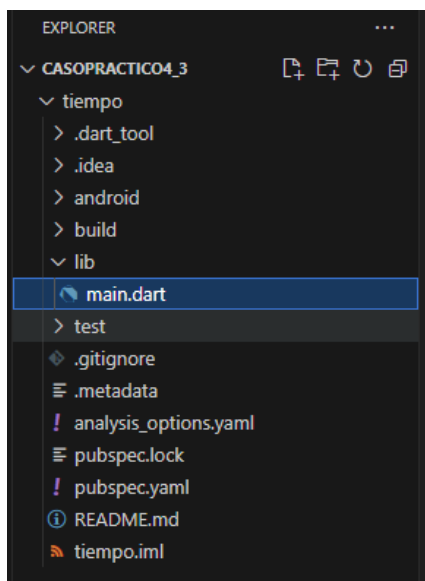
In order to run your application, type:

$ cd tiempo
$ flutter run

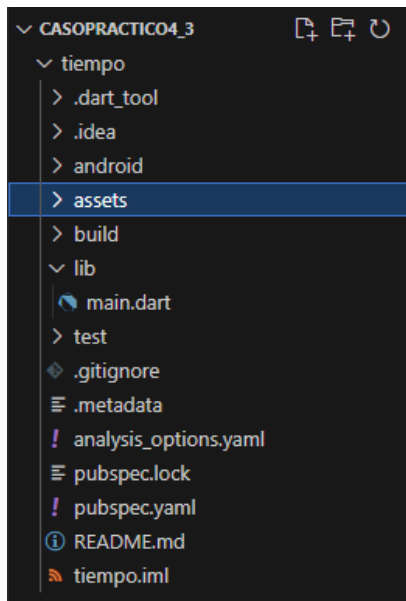
Your application code is in tiempo\lib\main.dart.

PS C:\Users\Fran\OneDrive\Documents\2FPGS_DAM\Programacion_Multimedia\Tareas\Tema4\CasoPractico4_3>
```

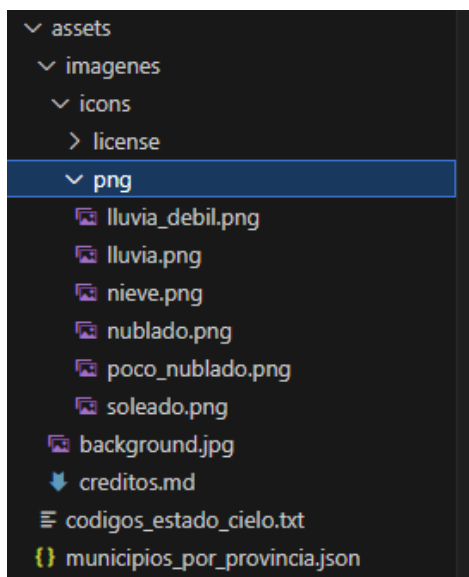
Nos generará nuestro proyecto en flutter.



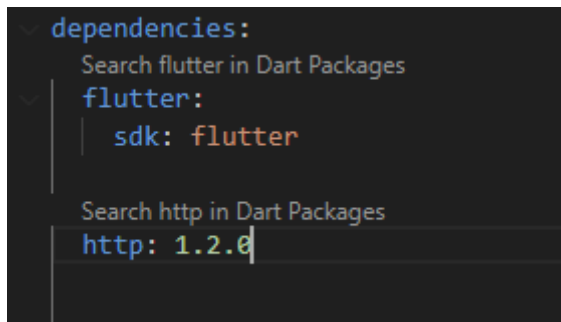
Creamos el directorio “assets” en el raíz del proyecto.



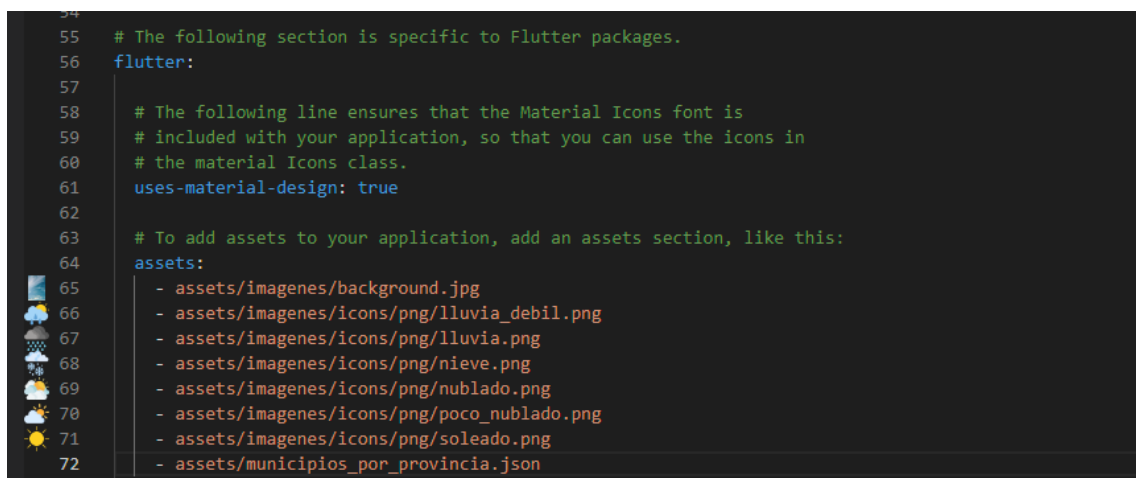
Ahora introducimos los recursos de imágenes, el json de los códigos de municipios y el txt de los códigos del estado del cielo dentro del directorio assets.



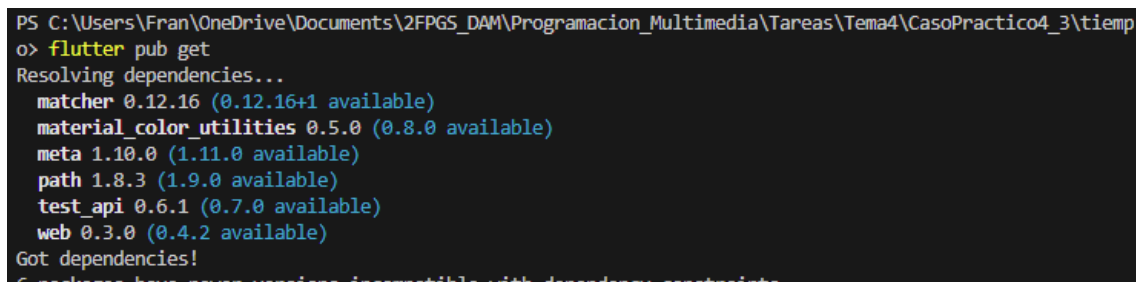
Nos vamos al fichero “pubspec.yaml” y añadimos la librería http.



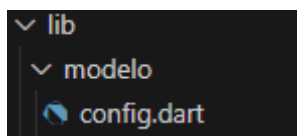
Si bajamos dentro del fichero, tenemos la sección flutter/assets. Añadimos los recursos que va a utilizar la aplicación.



Ahora para que surjan efecto los cambios, vamos a ejecutar el comando “flutter pub get”.



Creamos un directorio dentro de “lib” que le hemos puesto de nombre “modelo”. Dentro de él vamos a crear “config.dart” que va a ser la clase que contenga los datos de la apiKey y el código de la localidad que vamos a ver el tiempo.



[illegible]

Después creamos, dentro del directorio “modelo” también, la clase prevision.dart.

- ▼ modelo
 - config.dart
 - prevision.dart

```

tiempo > lib > modelo > prevision.dart > Prevision > obtenerImagen
Click here to ask Blackbox to help you code faster
//Clase encargada de obtener las imágenes dependiendo del valor que nos devuelva el servidor
class Prevision {
  String municipio = "";
  String codMunicipio = "";
  int precipitacion = 0;
  int tMax = 0;
  int tMin = 0;
  String estado = "";
  String valorEstado = "";
  String periodo = "";

  /*Constructor de la clase. Recibe nombre del municipio de tipo String y
  | su código de tipo int*/
  Prevision(this.codMunicipio, this.municipio);

  //Método getter para la imagen
  String get imagen {
    return obtenerImagen(valorEstado);
  }

  /*Método para obtener la imagen dependiendo del código que nos
  | devuelva el servidor*/
  String obtenerImagen(valor) {
    Set<String> nieve = <String>{
      "33",
      "33n",
      "34",
      "34n",
      "35",
      "35n",
      "36",
      "36n",
      "71",
      "71n",
      "72n",
      "73",
      "73n"
    };
    Set<String> lluviaDebil = <String>{
      "23",
      "23n",
      "24",
      "24n",
      "43",
      "43n",
      "44",
      "44n",
      "45",
      "45n",
      "46n",
      "61",
      "61n",
      "62",
      "62n",
      "63",
      "63n"
    };
  }
}

```

```

58     Set<String> lluvia = <String>{
59         "25",
60         "25n",
61         "26",
62         "26n",
63         "51",
64         "51n",
65         "52",
66         "52n",
67         "53",
68         "53n",
69         "54",
70         "45n"
71     };
72     Set<String> pocoNublado = <String>{"13", "13n", "14", "14n"};
73     Set<String> nublado = <String>{"15", "16n", "16", "17n", "17"};
74     Set<String> soleado = <String>{"11", "11n", "12", "12n"};
75
76     if (nieve.contains(valor)) return "nieve.png";
77     if (lluviaDebil.contains(valor)) return "lluvia_debil.png";
78     if (lluvia.contains(valor)) return "lluvia.png";
79     if (pocoNublado.contains(valor)) return "poco_nublado.png";
80     if (nublado.contains(valor)) return "nublado.png";
81     if (soleado.contains(valor)) return "soleado.png";
82
83     return "soleado.png";
84 }
85
86 //Sobreescribimos el método toString de la clase
87 @Override
88 String toString() {
89     return (""Municipio: $municipio ($codMunicipio)
90         Probabilidad de precipitaciones: $precipitacion
91         Temperatura mínima: $tMin
92         Temperatura máxima: $tMax
93         Estado del cielo: $estado ($valorEstado)
94         Rango horario: $periodo
95         "");
96 }
97 }
98

```


Después, volvemos al raíz del directorio “lib” y creamos la clase “obtener_prevision.dart”. Desde aquí es desde donde se van a obtener los datos de la AEMET.

```

tiempo > lib > obtener_prevision.dart > getMunicipio
Click here to ask Blackbox to help you code faster
1 import 'package:flutter/services.dart';
2 import 'package:http/http.dart' as http;
3 import 'package:tiempo/modelo/config.dart';
4 import 'dart:convert';
5
6 import 'package:tiempo/modelo/prevision.dart';
7
8 //Método para obtener el nombre del municipio guardado en el fichero de configuración
9 Future<String> getMunicipio() async {
10   final datosJson =
11     await rootBundle.loadString('assets/municipios_por_provincia.json');
12   var municipios = json.decode(datosJson);
13   for (var provincia in municipios["municipios"]) {
14     for (var municipio in provincia["municipios"]) {
15       if (municipio["codigo"] == Config.codigo) {
16         return (municipio["nombre"]);
17       }
18     }
19   }
20   return "";
21 }
22
23 /*Método que se va a encargar de hacer la petición al servidor de la
24 AEMET y nos va a devolver el tiempo del municipio*/
25 Future<Prevision> obtenerPrevision() async {
26   String codigo = Config.codigo;
27   String apiKey = Config.apiKey;
28
29   //Obtenemos el municipio y creamos un objeto de tipo Prevision
30   String? municipio = await getMunicipio();
31   Prevision prevision = Prevision(codigo, municipio);
32
33   //Url que nos proporciona la AEMET para realizar la consulta
34   String url =
35     'https://opendata.aemet.es/opendata/api/prediccion/especifica/municipio/diaria/$codigo?api_key=$apiKey';
36
37   //Realizamos la consulta al servidor
38   var respuestaServidor = await http.get(Uri.parse(url));
39
40   //Si el servidor nos responde con el código 200 significa que la respuesta ha tenido éxito
41   if (respuestaServidor.statusCode == 200) {
42     //Decodificamos el json que nos devuelve
43     var cuerpoJSON = jsonDecode(respuestaServidor.body) as Map;
44
45     //Realizamos otra petición para que nos devuelva los datos
46     var respuestaDatos = await http.get(Uri.parse(cuerpoJSON["datos"]));
47
48     //Si la respuesta es 200. Todo OK
49     if (respuestaDatos.statusCode == 200) {
50       //Obtenemos la información del tiempo en forma de Lista
51       var infoTiempo = jsonDecode(respuestaDatos.body) as List;
52       prevision.precipitacion = infoTiempo[0]["prediccion"]["dia"][0]
53         ["probPrecipitacion"][0]["value"];
54       prevision.tMax =
55         infoTiempo[0]["prediccion"]["dia"][0]["temperatura"]["maxima"];
56       prevision.tMin =
57         infoTiempo[0]["prediccion"]["dia"][0]["temperatura"]["minima"];
58
59       /*Estado del cielo.
60        Se hace con un while porque está dividido en 7 franjas horarias y no siempre
61        hay datos. Por ello recorreremos con el bucle hasta encontrar el primer dato*/
62       int i = 0;
63       while (prevision.estado == "" && i < 7) {
64         prevision.estado = infoTiempo[0]["prediccion"]["dia"][0]["estadoCielo"]
65           [i]["descripcion"];
66         prevision.valorEstado =
67           infoTiempo[0]["prediccion"]["dia"][0]["estadoCielo"][i]["value"];
68         prevision.periodo =
69           infoTiempo[0]["prediccion"]["dia"][0]["estadoCielo"][i]["periodo"];
70         i++;
71       }
72
73       return prevision;
74     }
75   }
76
77   return Prevision(Config.codigo, municipio);
78 }
79

```

Finalmente vamos a crear la clase principal “main.dart”, en la misma ruta que la anterior, la cual será la que cree la interfaz gráfica y llame a las diferentes clases y sus métodos para mostrar la información del tiempo por pantalla.

```

tiempo > lib > main.dart > ...
Click here to ask Blackbox to help you code faster
1 import 'package:flutter/material.dart';
2 import 'package:tiempo/modelo/prevision.dart';
3 import 'package:tiempo/obtener_prevision.dart';
4
Run | Debug | Profile
5 void main() {
6   runApp(const MyApp());
7 }
8
//Clase principal
9
10 class MyApp extends StatelessWidget {
11   const MyApp({super.key});
12
13   //Widget principal de la aplicación
14   @override
15   Widget build(BuildContext context) {
16     return MaterialApp(
17       debugShowCheckedModeBanner: false,
18
19       //Tema de la aplicación
20       theme: ThemeData(
21         //Color del tema
22         colorScheme:
23           ColorScheme.fromSwatch().copyWith(primary: Colors.blueGrey),
24         //Fuente
25         fontFamily: 'Roboto',
26         //Tema para el texto
27         textTheme: const TextTheme(
28           headlineSmall:
29             TextStyle(fontWeight: FontWeight.w300, color: Colors.white),
30           headlineMedium:
31             TextStyle(fontWeight: FontWeight.w400, color: Colors.white),
32           headlineLarge:
33             TextStyle(fontWeight: FontWeight.w300, color: Colors.white),
34         ), // TextTheme
35       ), // ThemeData
36       //Título de la aplicación
37       title: 'Aplicación del tiempo',
38       home: const AplicacionTiempo(title: 'Aplicación del Tiempo'),
39     ); // MaterialApp
40   }
41 }
42
//Clase que gestiona la aplicación
43
44 class AplicacionTiempo extends StatelessWidget {
45   // Constructor de la clase
46   // El atributo key se usa para identificarlo en el árbol de componentes
47   // El atributo title será el título de la barra superior
48   const AplicacionTiempo({Key? key, required this.title}) : super(key: key);
49   final String title;
50
51   //Constructor del widget de la aplicación
52   @override
53   Widget build(BuildContext context) {
54     return Scaffold(
55       appBar: AppBar(
56         title: Text(title),
57       ), // AppBar

```

```

58 |         body: const InfoTiempo(),
59 |       ); // Scaffold
60 |     }
61 |   }
62 |
63 |   //Widget para generar un estado a través de la información recibida de _InfoTiempo
64 |   class InfoTiempo extends StatefulWidget {
65 |     const InfoTiempo({Key? key}) : super(key: key);
66 |
67 |     @override
68 |     _InfoTiempo createState() => _InfoTiempo();
69 |   }
70 |
71 |   //Estado determinado por un objeto de tipo Prevision
72 |   class _InfoTiempo extends State<InfoTiempo> {
73 |     late Future<Prevision> _value;
74 |
75 |     //Estado inicial
76 |     @override
77 |     initState() {
78 |       super.initState();
79 |
80 |       //Obtenemos la previsión
81 |       _value = obtenerPrevision();
82 |     }
83 |
84 |     // Constructor del Widget que nos muestra la imagen de fondo
85 |     @override
86 |     Widget build(BuildContext context) {
87 |       return Container(
88 |         constraints: const BoxConstraints.expand(),
89 |         decoration: const BoxDecoration(
90 |           image: DecorationImage(
91 |             image: AssetImage("assets/imagenes/background.jpg"),
92 |             fit: BoxFit.cover), // DecorationImage // BoxDecoration
93 |
94 |         //Detector de gestos
95 |         child: GestureDetector(
96 |           // Detección de la velocidad en vertical
97 |           onPanEnd: (DragEndDetails downDetails) {
98 |             //Si hacemos un deslizado en la pantalla hacia abajo, actualizamos previsión
99 |             if (downDetails.velocity.pixelsPerSecond.dy > 0) {
100 |               setState() {
101 |                 _value = obtenerPrevision();
102 |               });
103 |             }
104 |           },
105 |
106 |           //Padding de los widgets
107 |           child: Padding(
108 |             padding: const EdgeInsets.all(24.0),
109 |             child: Center(
110 |               child: FutureBuilder(
111 |                 future: _value,
112 |
113 |                 /*Con el builder establecemos la manera de formar el widget cuando
114 |                 obtenemos la respuesta del Future*/
115 |                 builder: (BuildContext context, AsyncSnapshot snapshot) {
116 |                   //Si no tenemos respuesta, realizamos una animación de espera
117 |                   if (snapshot.connectionState == ConnectionState.waiting) {
118 |                     return const Center(
119 |                       child: SizedBox(
120 |                         height: 150.0,
121 |                         width: 150.0,
122 |                         child: CircularProgressIndicator(
123 |                           color: Colors.white,
124 |                         ), // CircularProgressIndicator
125 |                       ), // SizedBox
126 |                     ); // Center
127 |                   } else if (snapshot.connectionState == ConnectionState.done) {
128 |                     //Comprobación de error
129 |                     if (snapshot.hasError) {
130 |                       return const Text('Error');
131 |                     }
132 |                     //Si no hay error, generamos el widget con la previsión
133 |                   } else if (snapshot.hasData) {
134 |                     return MyInfoWeather(prevision: snapshot.data);
135 |                   } else {
136 |                     // Si no recibimos datos
137 |                     return const Text('Sin datos');
138 |                   }
139 |                 } else {
140 |                   // Mostramos cualquier otro estado
141 |                   return Text('Estado: ${snapshot.connectionState}');
142 |                 }
143 |               ),
144 |             ), // FutureBuilder // Center // Padding
145 |           ), // GestureDetector
146 |         ); // Container
147 |       }
148 |     }
149 |

```

```

150 // Widget para el contenido central con la previsión
151 class MyInfoWeather extends StatelessWidget {
152   final Prevision prevision;
153   const MyInfoWeather({Key? key, required this.prevision}) : super(key: key);
154
155   //Widget que nos mostrará la información en un layout vertical
156   @override
157   Widget build(BuildContext context) {
158     return Column(
159       children: <Widget>[]
160       //Imagen de la previsión
161       Image(
162         image: AssetImage("assets/imagenes/icons/png/${prevision.imagen}"),
163         height: 220.0), // Image
164       Center(
165         //Nombre del municipio
166         child: Text(
167           prevision.municipio,
168           style: Theme.of(context).textTheme.displaySmall,
169         ), // Text
170       ), // Center
171       Text(
172         //Previsión
173         "${prevision.estado}",
174         style: Theme.of(context).textTheme.headlineSmall,
175       ), // Text
176       Text(
177         //Probabilidad de precipitaciones
178         "\n Probabilidad de lluvia ${prevision.precipitacion}%",
179         style: Theme.of(context).textTheme.headlineSmall,
180       ), // Text
181       Padding(
182         //Temperaturas
183         padding: const EdgeInsets.all(24.0),
184         child: Row(
185           children: [
186             Expanded(
187               child: Center(
188                 child: Text(
189                   "${prevision.tMin}°",
190                   style: Theme.of(context).textTheme.displayMedium,
191                 ), // Text
192               ), // Center
193             ), // Expanded
194             Expanded(
195               child: Center(
196                 child: Text(
197                   "${prevision.tMax}°",
198                   style: Theme.of(context).textTheme.displayMedium,
199                 ), // Text
200               ), // Center
201             ), // Expanded
202           ],
203         ), // ROW
204       ), // Padding
205     ], // <Widget>[]
206   ); // Column
207 }
208 }

```

