RapidWeather
Manual técnico

1. Introducción
RapidWeather surge para resolver la necesidad de acceder a información meteorológica precisa, en tiempo real y con una experiencia de usuario intuitiva. Este software está diseñado para:

- **Resolver**: La falta de aplicaciones que combinen datos detallados (humedad, viento, pronósticos por horas/días) con gestión de ubicaciones favoritas.
- **Importancia**: Ofrece un valor diferenciador al integrar geolocalización automática, persistencia local de datos (SQLite) y una API robusta (WeatherAPI), optimizando el consumo de recursos.
- **Clientes potenciales**: Viajeros, deportistas al aire libre y usuarios que requieren planificación basada en condiciones climáticas.

2. Análisis del problema
2.a Problemática
- **Desafíos técnicos**:
    - Integrar datos en tiempo real desde una API externa con actualizaciones frecuentes.
    - Gestionar permisos de ubicación en diferentes versiones de Android.
    - Optimizar el almacenamiento local para guardar datos de ubicaciones favoritas
- **Necesidades no cubiertas**:
    - Falta de uso offline, cuando se aplica la capa offline la aplicación pierde mucho rendimiento quedando inutilizable para un usuario que quiere las cosas rápido.

2.b Clientes potenciales
- Usuarios finales:
    - Personas que realizan actividades sensibles al clima (senderismo, eventos).
    - Profesionales que trabajan al aire libre (agricultores, fotógrafos).
- Canales: Distribución directa mediante APK y futura publicación en Google Play Store.

2.c Análisis DAFO

Fortalezas

- Integración fluida con WeatherAPI
- Diseño intuitivo
- Persistencia local con SQLite
  - º

Debilidades

- Dependencia de una API externa
- Solo disponible para Android
- Sin autenticación de usuarios

Oportunidades

- Expansión a iOS
- Monetización con anuncios

Amenazas

- Competencia de apps establecidas
- Cambios de políticas de WeatherAPI

2.d Monetización y beneficios

Desarrollaría el modelo freemium:

- **Modelo freemium**:
  - Versión gratuita con anuncios no intrusivos.
  - Versión premium ($2.99/mes): Sin anuncios, pronóstico extendido a 7 días.
- **Beneficios esperados**:
  - 10,000 descargas en el primer año (APK + Play Store).
  - Alianzas con empresas de turismo para integraciones personalizadas.

3. Diseño de la solución

3.a Tecnologías elegidas

Componente	Tecnología	Motivo
Desarrollo móvil	Flutter (Dart)	Multiplataforma y hot reload para desarrollo ágil
API Externa	WeatherAPI	Ofrece datos precisos y capa gratuita con 1M llamadas/mes
Base de datos local	SQLite (sqflite)	Ligera y compatible con operaciones CRUD rápidas
Geolocalización	geolocator	Permite obtener la geolocalización

3.b Arquitectura

La arquitectura elegida es:

Patrón por capas con servicios centralizados

- Capa UI (Vistas y Widgets)
- Servicios (API y BBDD)
- Modelos (DTOs)

El flujo de datos funciona de la siguiente manera:
Ejemplo:

1. Las vistas (principal_screen.dart, buscar_screen.dart) inician peticiones.

```dart
// Método que inicializa el tiempo desde una llamada a la API con la geolocalización
Future<void> _initializeWeatherData() async {
  try {
    Position? lastKnownPosition = await Geolocator.getLastKnownPosition();

    if (lastKnownPosition != null) {
      latitude = lastKnownPosition.latitude;
      longitude = lastKnownPosition.longitude;
    } else {
      Position currentPosition = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.medium,
      );
      latitude = currentPosition.latitude;
      longitude = currentPosition.longitude;
    }

    String latLongString = '$latitude, $longitude';

    setState(() {
      weatherData = ApiService().fetchWeatherForThreeDays(latLongString);
    });
```

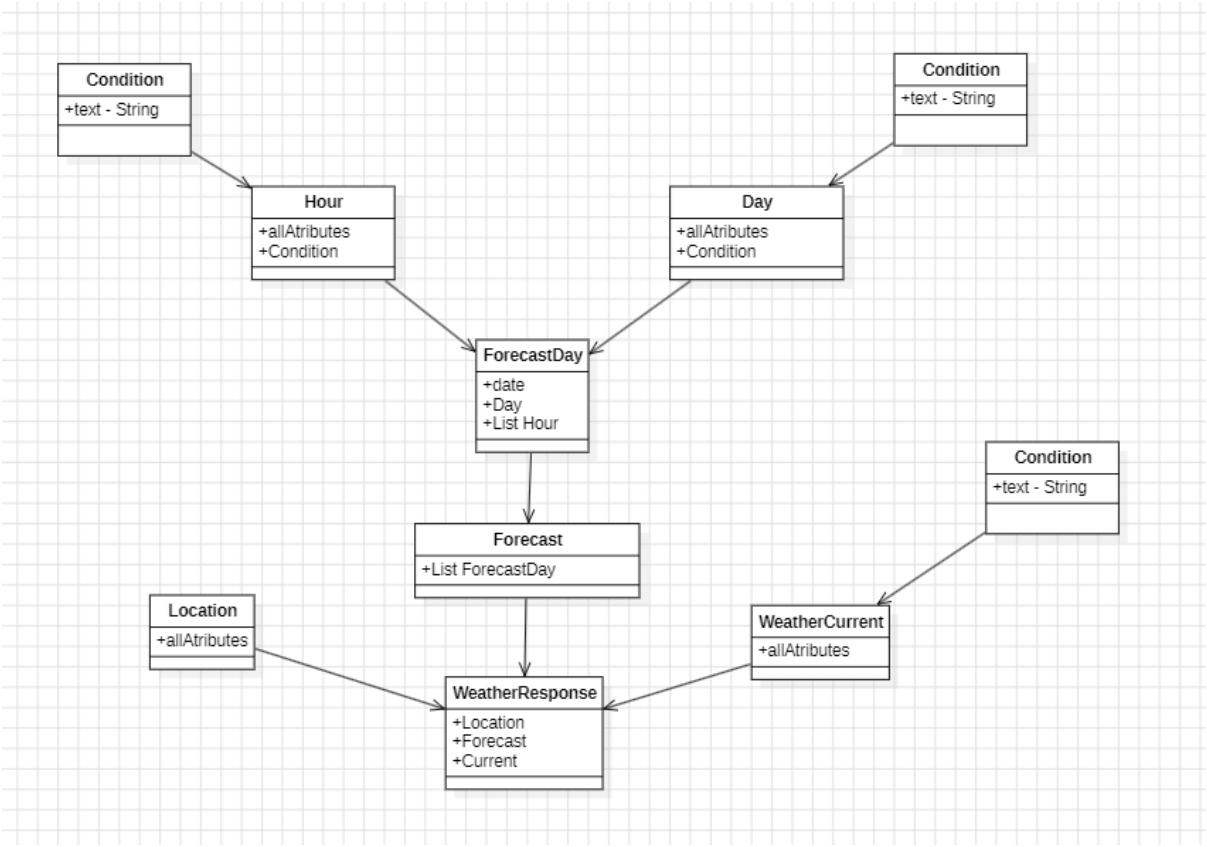
2. Los servicios (api_service.dart, bbdd_service.dart) gestionan lógica.

```dart
/// Método para obtener el pronóstico del clima para los próximos 3 días
///
/// Parámetros:
/// - [localizacion]: Nombre de la ciudad o coordenadas (latitud,longitud)
///
/// Retorna un objeto `WeatherResponse` con el pronóstico extendido de 3 días
Future<WeatherResponse> fetchWeatherForThreeDays(String localizacion) async {      You, hace 6
  final url = Uri.parse(
      '$baseUrl/forecast.json?key=$apiKey&q=$localizacion&days=3&aqi=yes&alerts=yes&lang=es');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final decodedResponse = utf8.decode(response.bodyBytes);
      final Map<String, dynamic> data = json.decode(decodedResponse);

      // Convertir la respuesta JSON en un objeto `WeatherResponse`
      WeatherResponse weatherResponse = WeatherResponse.fromJson(data);

      return weatherResponse;
    } else {
      throw Exception('Failed to load forecast: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Error fetching forecast: $e');
  }
}
}
```

3. Los modelos (weather_response.dart, location.dart) estructuran los datos.

```dart
// Método para transformar el JSON a un objeto WeatherResponse y lo retorna
factory WeatherResponse.fromJson(Map<String, dynamic> json) {      You, hace 6
  return WeatherResponse(
    location: Location.fromJson(json['location'] ?? {}),
    current: Current.fromJson(json['current'] ?? {}),
    forecast: Forecast.fromJson(json['forecast'] ?? {}),
  ); // WeatherResponse
}
```

3.c Diagrama de clases



3.d Diagrama E/R (Entidad-Relación)

Las tablas tienen sus datos cada una y no hay relación entre ellas

4.Documentación de la solución

El enlace al repositorio es este https://github.com/OscarLuque-31/RapidWeather.git
El repositorio es público y se puede descargar el código cuando se quiera.

5. Enlaces de interés

5.1 Documentación técnica

Recurso	Enlace	Descripción
Flutter	Documentación oficial	Guías, widgets y API de Flutter.
WeatherAPI	Documentación de endpoints	Detalles de los parámetros y respuestas de la API.
SQLite en Flutter	Paquete sqflite	Cómo ejecutar consultas CRUD.
Geolocalización	Paquete geolocator	Manejo de permisos y obtención de coordenadas.

5.2 Herramientas de diseño

Recurso	Enlace
Prototipo en Figma	Ver diseño
Paleta de colores	Archivo app_colors.dart

5.3 Comunidad y soporte

Recurso	Enlace
Stack Overflow	Etiqueta Flutter
GitHub Issues	Reportar bugs

5.4 Paquetes útiles para futuras mejoras

Paquete	Enlace	Uso potencial
flutter_bloc	pub.dev	Gestión avanzada de estado.
firebase_core	pub.dev	Sincronizar favoritos entre dispositivos.
flutter_local_notifications	pub.dev	Alertas climáticas.