天气 APP 文档

小组负责人姓名：许梁超 学号：221900103 院系：智能软件与工程学院

小组成员姓名： 王诗瑶 学号：221900084 院系：智能软件与工程学院

小组成员姓名： 张淳皓 学号：221900068 院系：智能软件与工程学院

小组成员姓名： 周天远 学号：221900448 院系：智能软件与工程学院

目录

一、使用文档

（1） 注册账号

- 功能概述

  用户通过填写唯一用户名与强密码完成注册，为应用内身份做标识，保障数据安全；

  注册账号可定制城市天气信息的推送，实现精准推送。

- 操作步骤

1. 打开应用

   在手机或电脑上找到对应的天气应用程序，点击打开。

2. 进入注册页面

   在首页（登录页面）可以找到“立即注册”按钮，点击该按钮进入注册页面。

3. 填写注册信息

   需要填写用户名、设置密码、确认密码三个信息；

   确保填写信息准确无误，密码设置符合要求，至少 6 位（包含字母和数字）。

4. 点击注册按钮

确认填写的信息无误后，点击“注册”按钮提交注册信息。

5. 注册成功

系统会提示注册成功，并跳转到登录页面。

6. 注册失败

系统会显示失败原因的提示信息，如：用户名已存在，密码不符合条件等。

(2) 登录账号

- 功能概述

确认用户身份，让已注册用户能访问其个性化设置和数据，实现应用使用的连贯性与个性化体验。

- 操作步骤

1. 输入用户名

在登录界面的“请输入用户名”输入框中，准确输入您已注册的用户名。

2. 输入密码

在“请输入密码”输入框中，输入您注册账号时设置的密码；

如果您不确定密码是否正确，可以点击密码输入框右侧的眼睛图标，查看密码输入情况，但请注意保护个人隐私，避免在公共场合使用此功能。

3. 点击登录按钮

确认用户名和密码输入无误后，点击下方蓝色的“登录”按钮。

4. 登录结果反馈：

a. 登录成功

系统会跳转到相应的主页面或个人中心页面，您可以开始使用该账号相关的功能和服务。

b. 登录失败

系统会显示失败原因的提示信息，如：用户名不存在，密码错误等；

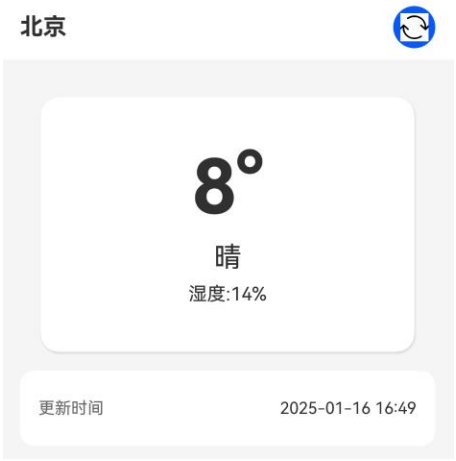
请检查用户名和密码是否正确，然后重新输入并尝试登录。

(3) 默认城市

- 功能概述

提供一个首页简洁的默认城市天气信息的显示和刷新功能。

- 操作步骤



1. 默认城市的显示
   在天气应用界面的顶部，明确显示了“南京”，这就是当前的默认城市。
2. 默认城市天气信息的刷新
   为确保获取到最新的天气，用户可以点击界面右上角的蓝色圆形的刷新图标（带有旋转箭头）；点击后，系统会重新获取默认城市的天气数据，并更新主界面上的显示内容。
3. 默认城市简单天气信息的显示
   在顶部的下方的区域内，显示了当前城市的简要天气信息，包含：温度、天气状况、湿度。
4. 默认城市天气信息的更新时间
   在用户信息下方显示了天气的更新时间，让用户清楚上面天气信息的时效性。

（4） 修改城市

- 功能概述
  根据用户需求对默认城市进行修改并存储相关偏好，实现对应城市天气的查询
- 操作步骤

1. 找到设置页面
   在 APP 下方 Tabs 中，点击设置按钮，即可进入对应的设置页面

2. 点击城市管理

在设置页面，找到最上面的城市管理条目，点击该条目

3. 选择想要查询天气的城市

输入想要查询的城市或者选择下方已经有的历史搜索记录进行对应天气的查询

(5) 查看天气详情

- 功能概述

用户有访问未来天气信息或者是具体的天气变化等信息的需求时，使用这一功能可以实现中央气象台网站的访问，查询对应天气情况。

- 操作步骤

1. 找到天气详情页面

在 Tabs 中找到天气详情按钮，点击进入当前城市的天气详情页面

2. 显示对应天气详情

在天气详情页，显示默认城市（或用户修改的城市）的详细天气信息

二、技术文档

（1） API 来源

1. 天气数据供应商：高德地图开放平台

   本应用所使用的 API 来自高德地图开放平台，其提供了丰富的地理信息和天气相关服务

2. API 接入方法

   1) 获取城市编码接口：https://restapi.amap.com/v3/config/district

      功能：该接口用于根据城市名称获取对应的城市编码

      请求参数

      keywords：经过 encodeURIComponent 编码的城市名称，用于指定要查询的城市

      subdistrict：设置为 0，表示不查询下级行政区

      key：即 API_KEY，用于身份验证，确保只有合法的开发者能够使用该 API

   2) 获取实时天气接口：https://restapi.amap.com/v3/weather/weatherInfo

      功能：该接口用于获取指定城市编码对应的实时天气数据

      请求参数

      city：城市编码，通过 getCityId 方法获取，用于指定要查询天气的城市

      key：同样为 API_KEY，用于身份验证

（2） API 使用，数据解析

1. 获取城市编码的 API 使用

   1) 构建 URL

      用 encodeURIComponent 对 cityName 编码，再构建 URL

      https://restapi.amap.com/v3/config/district?keywords=${encodedCity}&subdistrict=0&key=${API_KEY},

      其中 keywords 是编码后的城市名，subdistrict=0 不查下级行政区，key 为认证密钥

   2) 发送请求

      用 http.createHttp()创建 HttpRequest 对象，通过 httpRequest.request 发 GET 请求

   3) 处理响应

      请求完成后，用 await 获取 HttpResponse 对象，打印状态码。若状态码为 OK，将响应转字符串，用于后续解析

2. 获取实时天气的 API 使用

   1) 构建 URL

      getNowWeather 方法中，构建 URL

      ${BASE_URL}/weatherInfo?city=${cityCode}&key=${API_KEY},

      BASE_URL 为天气 API 基础地址，city 是城市编码，key 为密钥

   2) 发送与处理

      与获取城市编码类似，发 GET 请求、设置相同请求头。获响应后打印状态码和数据，若成功则将响应转字符串用于解析

3. 数据解析
   1) 获取城市编码
      请求成功后，用 JSON.parse 将响应字符串解析为 DistrictResponse 对象
      判断 status 为 1 且 districts 有数据，提取首个城市的 adcode 返回
   2) 获取实时天气
      请求成功后，解析响应为 GaoDeWeatherResponse 对象
      判断 status 为 1 且 lives 有数据，从中提取温度、天气状况、湿度信息，赋值给 WeatherData

(3) 数据库的实现
1. 数据库初始化
   1) 数据库创建
      使用 SQL 语句定义数据库的表结构，包括存储天气数据各个字段及它们的类型
   2) 数据库初始化
      通过 initDatabase 方法进行实际的数据库创建操作
      包括设置数据库配置（名称和安全级别），获取 RdbStore 对象，以及创建 weather_data 表
2. 数据操作方法
   1) 保存天气数据
      创建 ValuesBucket 对象 value，将 WeatherData 对象中的数据以及城市名称和当前时间填入
      用 insert 方法将 ValuesBucket 中的数据插入 weather_data 表中
   2) 获取最新天气数据
      getLatestWeatherData 方法用于获取指定城市的最新天气数据
   3) 获取天气历史记录
      getWeatherHistory 方法用于获取指定城市的历史记录，通过传入 limit 参数获取对应数量
   4) 清楚天气历史数据
      clearWeatherHistory 方法用于清除指定城市的所有天气历史数据

(4) UI 布局实现：导航和页面的响应式布局
1. 整体布局
   1) 整体：使用 Column 组件作为根组件，构建了一个垂直方向的布局结构
   2) 包含：在这个 Column 中包含了多个子组件，形成了应用的整体页面布局
   3) 分区：整个布局分为三个主要部分：顶部导航栏、Tabs 组件和页面内容。
2. 导航栏布局
   1) 作用：展示 APP 中最重要的内容——当前城市信息
   2) 布局：使用 Row 组件实现水平布局，包含：
      一个显示当前城市的 Text 组件、一个 Blank 占位组件、一个用于刷新的 Button 组件
3. Tabs 组件
   1) 作用：实现标签页切换功能，将页面划分为多个不同的部分
   2) 布局：包括当前天气、天气详情和设置三个页面
4. 响应式布局
   1) 当前天气页面：使用 Column 组件构建垂直布局，根据 isLoading 状态显示不同的内容
      当 isLoading 为 true 时，显示 LoadingProgress 组件，提示用户正在加载数据;
      当 isLoading 为 false 时，显示 WeatherCard 组件和更新时间信息。
   2) 天气详情页面
      用 Stack 组件包裹 Web 组件，使用 src 属性设置网页的源地址，通过 controller 进行控制
   3) 设置页面
      使用 Column 包含 List，List 中包含多个 ListItem，以列表形式展示设置选项

(5) 突出体现我们的离线缓存与数据同步

1. 存储机制：偏好设置存储（Preferences）
   定义存储的名称为 PREFERENCES_NAME，其值为 'weather_preferences'，作为存储数据的容器
2. 存储和同步方法
   1) 存储和获取当前城市
      a. 存储：saveCity(city: string): Promise<void>
         将用户当前选择的城市存储到偏好设置中，以便以后能够记住用户选择的城市
      b. 获取：getCurrentCity(): Promise<string>
         从偏好设置中获取之前存储的用户当前城市信息
   2) 存储和获取更新时间
      a. 存储：saveUpdateTime(time: string): Promise<void>
         存储最近一次天气数据更新的时间，方便后续检查数据的时效性
      b. 获取：getLastUpdateTime(): Promise<string>
         从存储偏好中获取最近一次天气数据更新的时间
   3) 存储和获取天气数据
      a. 存储：saveWeatherData(weatherData: WeatherData): Promise<void>
         将 WeatherData 类型的天气数据存储到偏好设置中，需要转化为 JSON 字符串存储
      b. 获取：getCachedWeatherData(): Promise<WeatherData | null>
         从偏好设置中获取存储的天气数据，并将其 JSON 字符串解析为 WeatherData 对象
   4) 存储和获取城市列表
      a. 存储：saveCityList(cityList: string[]): Promise<void>
         存储用户关注的城市列表，以数组形式存储，需要将数组序列化为 JSON 字符串
      b. 获取：getCityList(): Promise<string[]>
         从存储偏好中获取存储的城市列表，并将存储的 JSON 字符串解析为字符串数组

(6) 多线程优化的实现

1. 模块依赖：@ohos.taskpool
   提供了创建和管理并发任务的功能，使得 getUrl 函数能够在独立的线程中执行，避免了主线程的阻塞，提高了程序的并发处理能力和响应速度。
2. URLTask 类
   1) 功能：提供了一个静态方法 getUrlForWebView，用于创建并执行 getUrl 任务，并对任务执行结果进行处理
   2) 方法：getUrlForWebView
      参数：currentCity：表示当前城市的名称，类型为字符串，例如 "上海"
      CityID：表示城市的唯一标识，类型为字符串，例如 "310000"
      CurBp：表示屏幕断点类型，类型为字符串，例如 "sm" 或其他
   3) 实现细节
      创建 taskpool.Task 对象，将 getUrl 函数和所需参数传递给它
      使用 await taskpool.execute(task) 执行任务，并将结果存储在 res 变量中
      若执行成功，将结果作为 URL 输出；若出现异常，打印错误信息并返回 -1
3. 使用方法
   1) 首先，确保已经正确导入 UrlTask 类和相关依赖
   2) 调用 UrlTask.getUrlForWebView(currentCity, CityID, CurBp)方法，将所需参数传递进去
   3) 等待 getUrlForWebView 方法返回生成的 URL
   4) 使用返回的 URL 执行后续操作，如使用 this.controller.loadUrl(this.url)加载相应的网页内容