修订历史:

修改或初始编写日期	SEPG	版本	说明	作者	评审时间	评审参与人员	评审后修改批准日期	确认签字人员
2025-03-20		1.0	完成功能建模	刘文美 高妤婕 邵任飞	2025-03-21	刘文美 高妤婕 邵任飞	2025-03-21	
2025-03-22		1.1	完成数据建模与行为建模	刘文美 高妤婕 邵任飞	2025-03-23	刘文美 高妤婕 邵任飞	2025-03-23	
2025-03-24		1.2	完成文档编写	刘文美 高妤婕 邵任飞	2025-03-25	刘文美 高妤婕 邵任飞	2025-03-25	

1 引言

1.1 背景

本项目将开发一款名为旅途印记（TravelMate）的小程序。该小程序旨在为自由行用户提供一站式旅游解决方案，通过整合行程规划、预算管理和天气查询等功能，帮助用户简化旅行计划、提高效率并提升个性化体验。

本软件的开发始于对现代旅游需求的分析。用户通常需要多个工具来处理行程安排、开销监控等问题，导致信息分散、效率低下。该软件通过整合多个功能，提供一站式解决方案。本项目的开发者为邵任飞、高妤婕和刘文美，目标用户为自由行爱好者、旅游规划者以及希望高效管理行程和预算的旅行者。软件将作为微信小程序运行，数据存储和计算功能主要依托云服务器实现，支持高并发访问和数据安全。

该软件通过多个接口与其他系统和机构交互，形成数据和功能的协同网络，包括：通过天气服务 API 获取实时天气信息和未来天气预报，为用户提供目的地天气更新和提醒；通过 OpenAI 的 ChatGPT API，根据预算、季节和目的地等信息提供个性化的智能行程推荐。

1.2 参考资料

1.2.1 IEEE802.3 标准-2018 年版

本项目用到的软件开发标准为 IEEE802.3 标准-2018 年版，发布单位为 Institute of Electrical and Electronics Engineers（IEEE）。

1.2.2 ChatGPT API 文档

发布日期为 2024 年 11 月，来源为 https://platform.openai.com。

1.3 假定和约束

设备条件：开发团队需配备高性能开发工作站，服务器采用阿里云提供的云计算资源。

交流问题：由于项目团队分布式工作模式，需使用高效的项目管理和协作工具进行日常沟通。

安全问题：系统将采用严格的安全措施，确保用户账户、行程与开销数据的安全，防止未经授权的访问和数据泄露。

界面：系统将设计为直观易用的界面，帮助用户轻松操作并理解各项功能。

1.4 用户的特点

本软件最终用户主要为自由行旅行者，教育水平涵盖高中及以上，具备基本的智能手机操作技能。用户无需具备专业的技术知识，但对旅行工具和应用有一定的使用经验，大多数用户无需专业培训即可熟练使用软件功能。考虑到少量用户对智能手机的操作不熟悉，本软件需要提供简单直观的界面设计，同时支持指导或提示，以满足不同用户的需求。预计用户主要在旅行计划阶段和旅途中使用软件，访问频率高，特别是在行程调整和预算管理环节。

系统的维护人员需要熟悉云计算架构、API 集成、以及前后端技术栈（如 Spring Boot）以确保系统的稳定性与高性能运行。此外，还需具备安全协议和隐私保护相关的技术能力，维护系统的数据安全性。对于维护人员而言，需要提供日常监控和定期维护，以保证系统的正常运行和用户体验的持续优化。

2 功能需求

2.1 系统范围

本项目旨在开发一款智能旅游软件，满足用户在旅行中的多种需求，帮助用户高效规划行程、管理预算并获取实时天气信息。系统的主要目标是通过整合多个独立工具功能，为用户提供一站式的旅游解决方案，克服信息分散和效率低下的问题。

应用目标主要包括：提供全面的行程管理功能，包括灵活的行程规划、修改和调整；实现预算管理功能，跟踪支出，帮助用户控制旅行成本；提供目的地的实时天气更新和提醒，确保用户可以根据天气动态调整行程计划；借助大语言模型和旅游信息接口，为用户生成个性化的智能行程推荐。

本系统通过接口与多个外部服务和系统集成，包括：天气服务 API 用于提供实时天气信息和未来天气预报；OpenAI 大语言模型接口实现智能行程推荐功能；小红书爬取内容与携程 API 提供旅游行程数据、景点推荐、住宿信息等内容；云端数据存储服务确保用户数据的安全性和持久性。

2.2 系统总体流程

用户登录进入系统，开始新行程或管理已有行程。

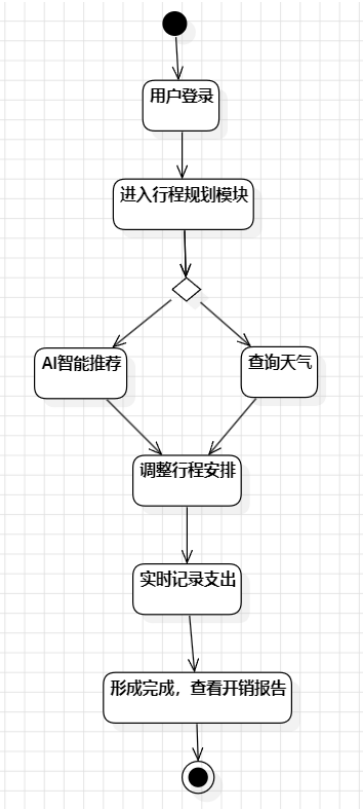
在行程规划模块，用户输入目的地、时间、预算等信息，创建新行程，系统调用大语言模型生成智能推荐行程。

用户进一步规划行程，添加修改删除事件、修改预算等，生成最终行程。

在预算管理模块，用户实时记录支出，系统提供支出分析报告。

系统调用天气 API 提供目的地实时天气及提醒，帮助用户动态调整行程。

用户在旅行结束后查看总结报告，包括行程回顾、预算执行情况等。



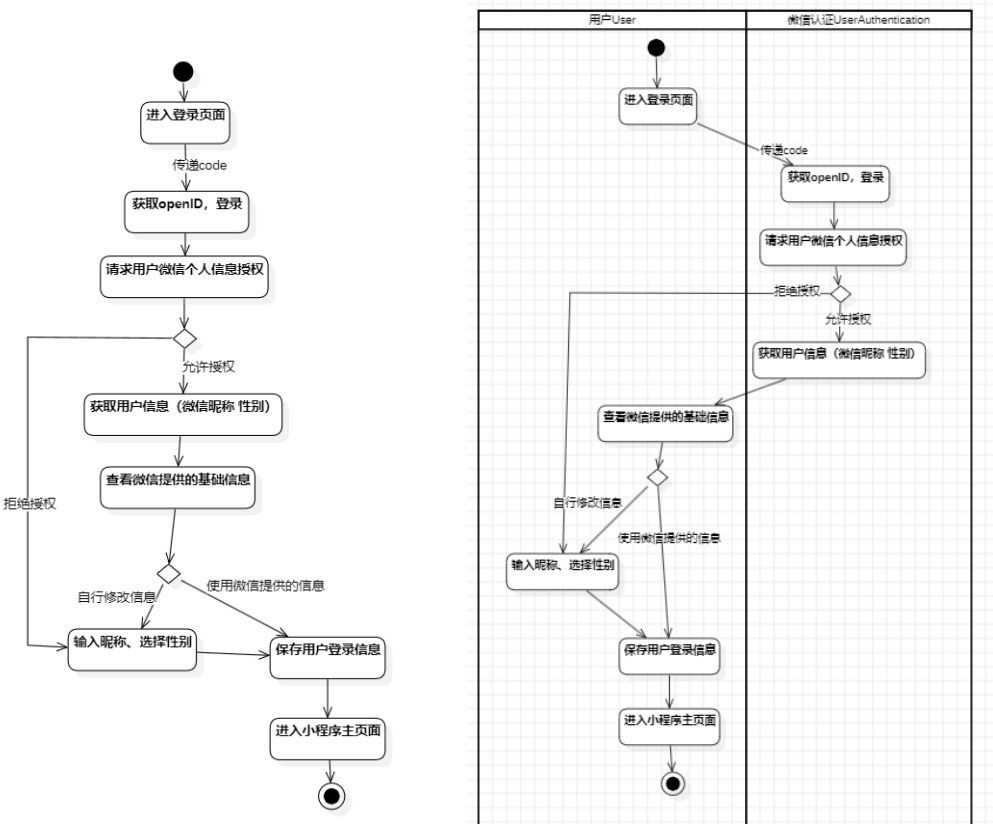
2.3 需求分析建模

2.3.1 功能建模

功能建模为每个用例创建了活动图，并进一步转化为泳道图以进一步抽象。

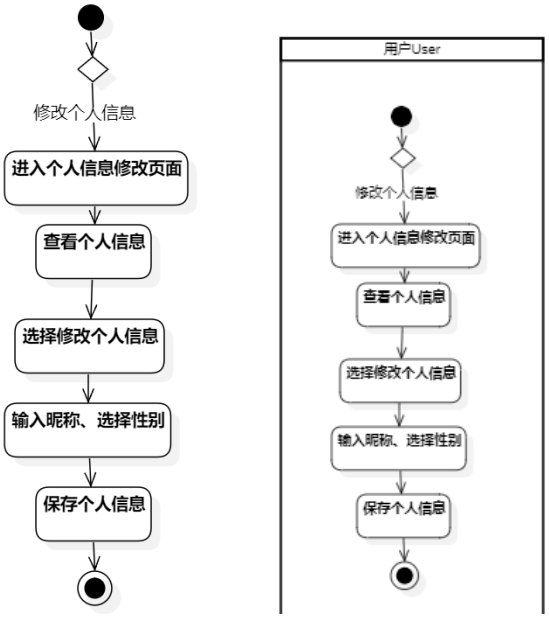
用户管理子系统

- **登录账号**



在登录用例中，用户首先进入登录页面。系统会通过 code，获取 openID，进行登录。然后系统会请求用户微信个人信息授权，如果用户允许授权，系统请求获取用户微信信息，查看微信提供的基础信息，用户可选择直接适用微信提供的信息作为本系统的信息，也可自行输入昵称、选择性别；如果用户拒绝授权，则用户智能自行输入昵称、选择性别。系统保存用户登录信息后，用户即可进入小程序主页面。

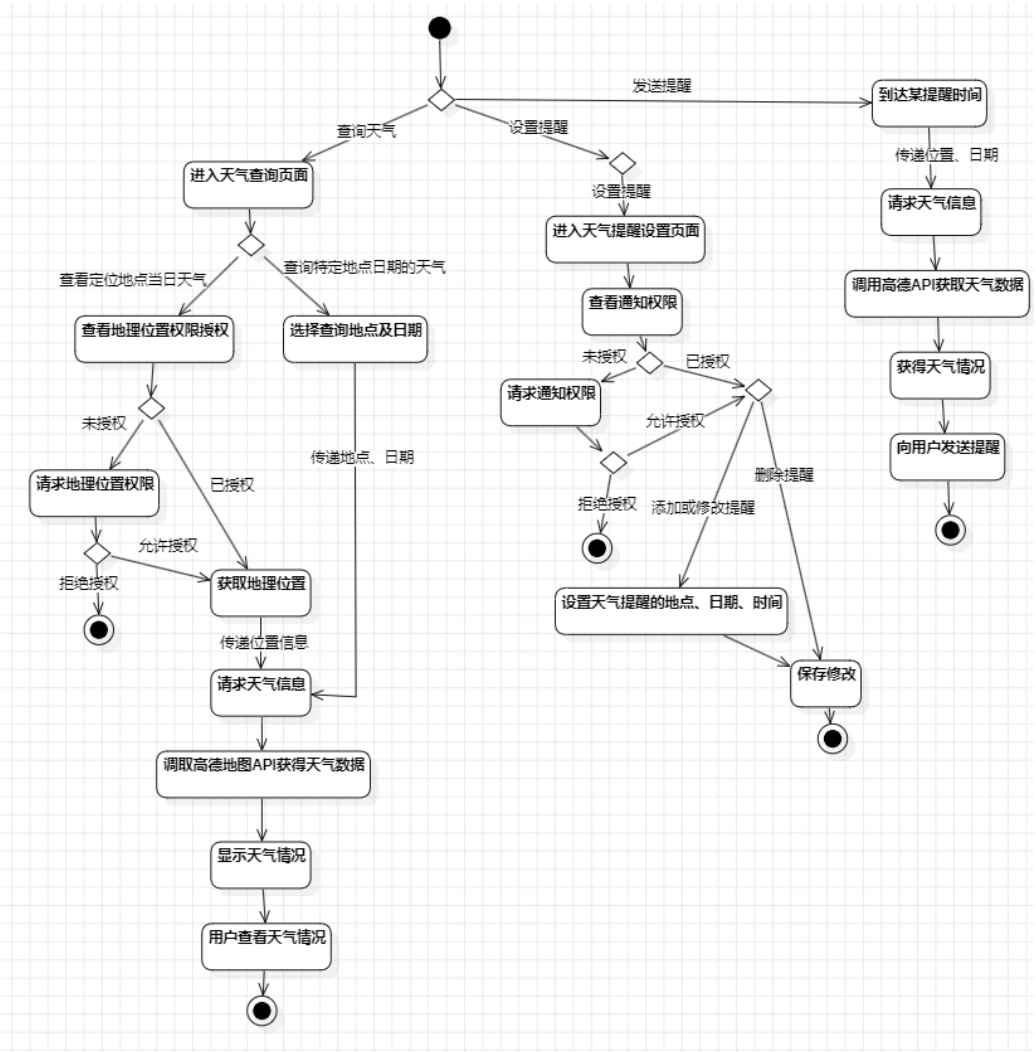
- **个人信息设置**

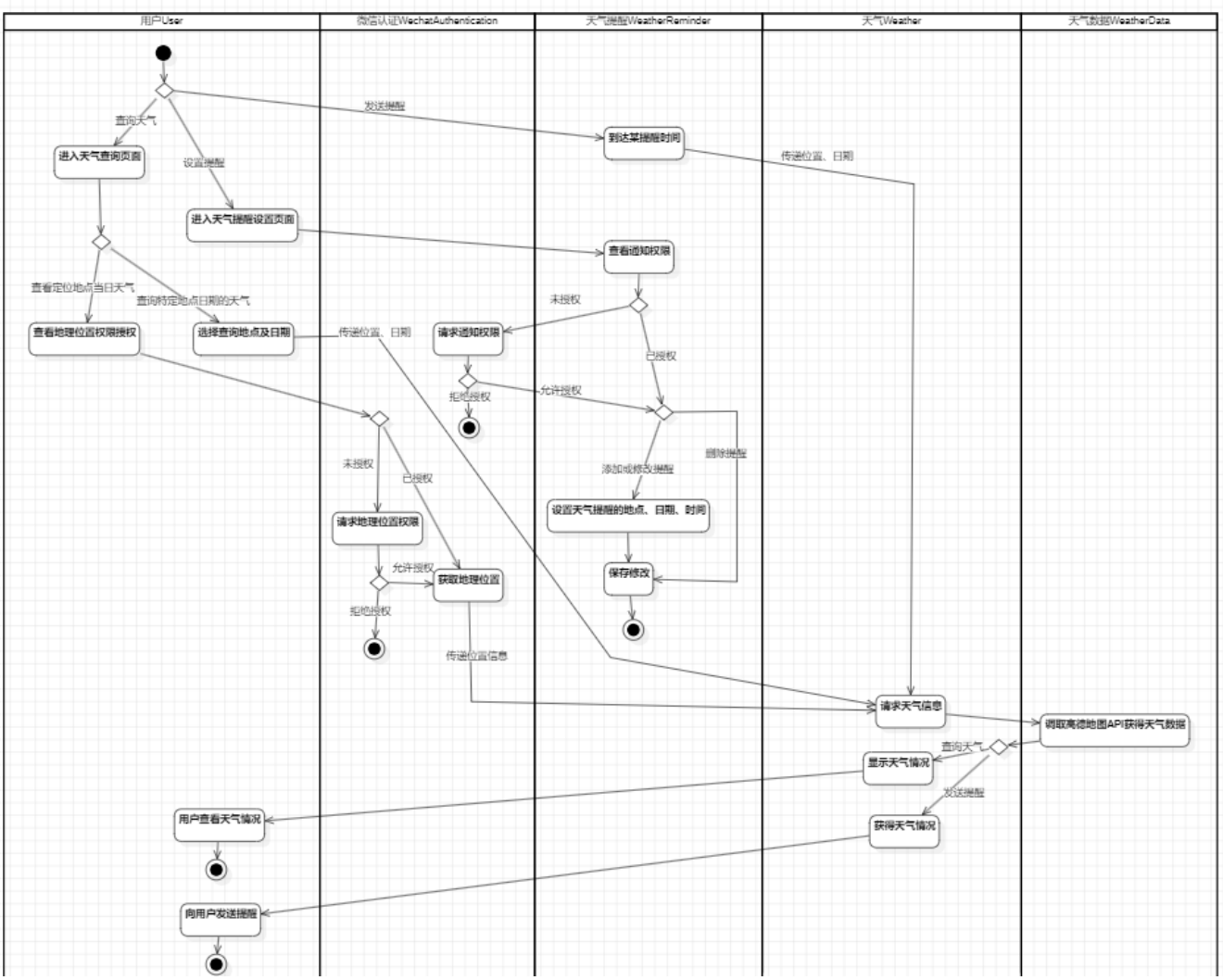


在个人信息设置用例中，流程始于用户选择修改个人信息的操作。系统首先会进入个人信息设置页面，显示当前的个人信息内容。用户可以选择要修改的信息项目，输入新的内容，系统会保存用户的个人信息。整个过程支持多个信息项的批量修改，并在完成后及时展示更新结果。

天气查询子系统

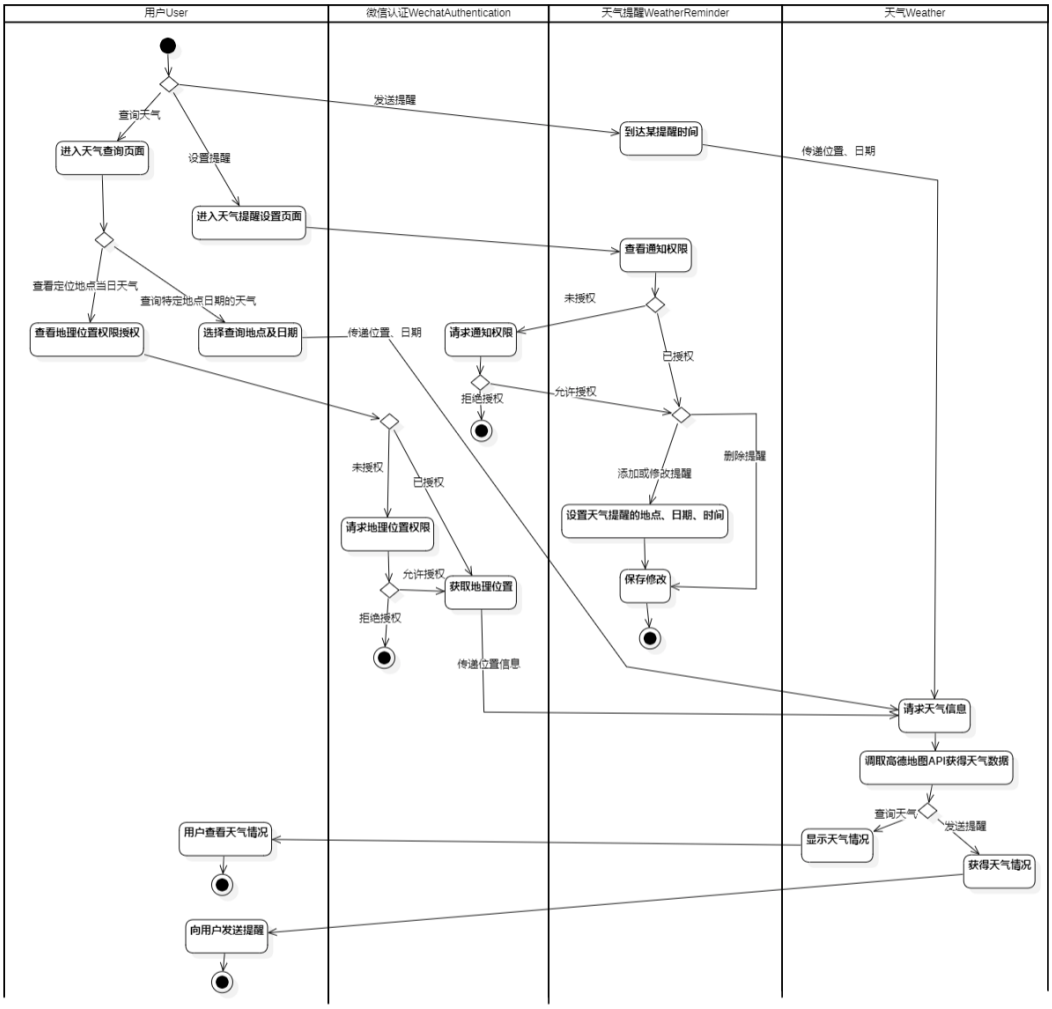
- **查询天气及接受提醒**





用户进入天气查询页面后，可以选择两种查询方式：查询定位地点当日天气和查询指定地点日期的天气。对于查询定位地点当日天气，系统首先判断用户是否授权地理位置查看权限，如果已授权，系统会显示当前位置的天气状况，如果未授权，系统请求地理位置权限，若用户拒绝授权，则无法查看定位地点当日天气，若用户允许授权，系统会显示当前位置的天气状况；对于查询指定地点日期的天气，用户需要输入查询的时间和地点，系统会根据输入获取相应的天气数据并展示。用户还可以设置天气提醒，用户进入天气提醒设置页面后，系统首先判断用户是否授权通知权限，若已授权，可设置提醒，若未授权，系统请求通知权限，若用户拒绝授权，则用户无法设置提醒，若用户允许授权，则可设置提醒。提醒设置后，可对提醒进行修改删除。当到达提醒时间时，系统会自动发送天气提醒通知。

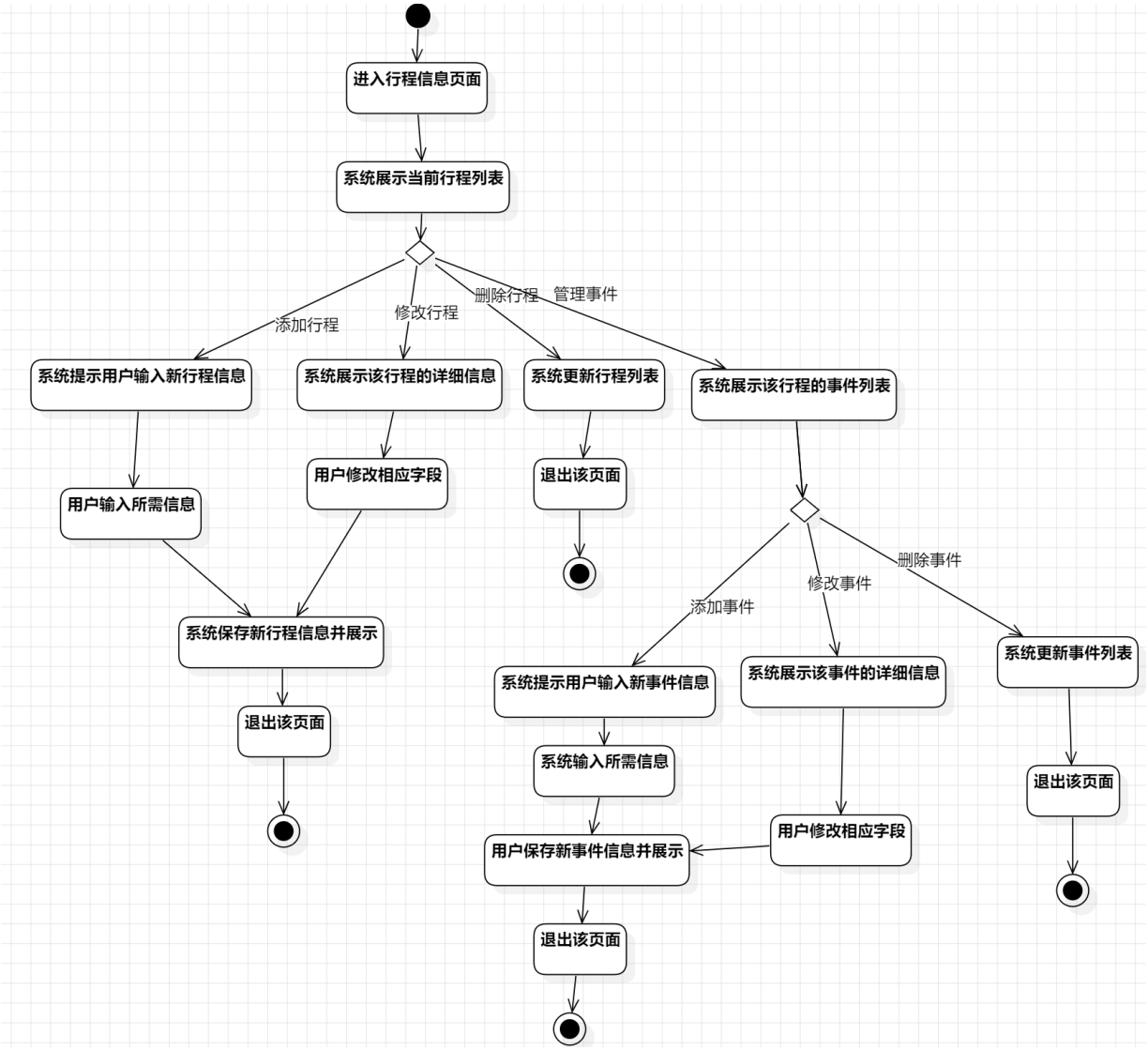
Iteration 2:

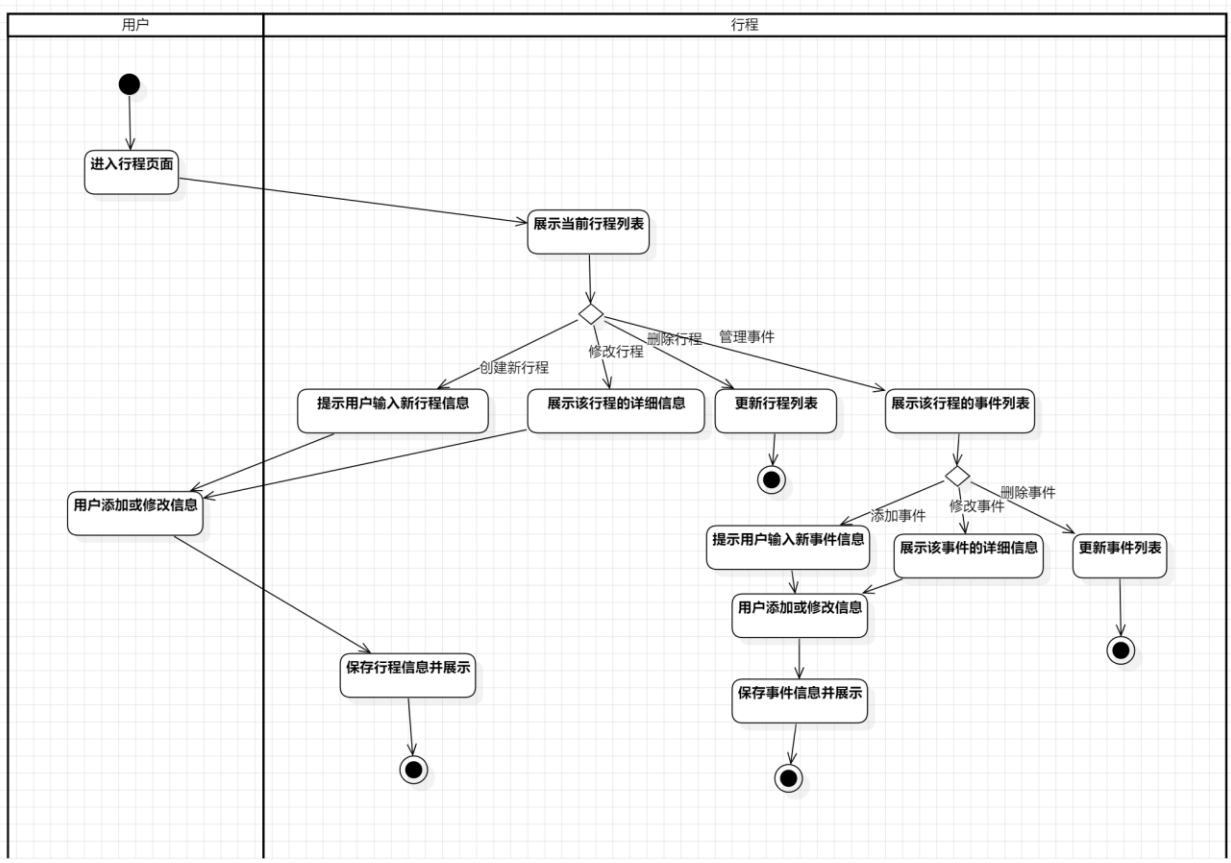


将天气数据类合并入天气类，天气类负责调取高德地图 API 获得天气数据的 action。

行程规划子系统

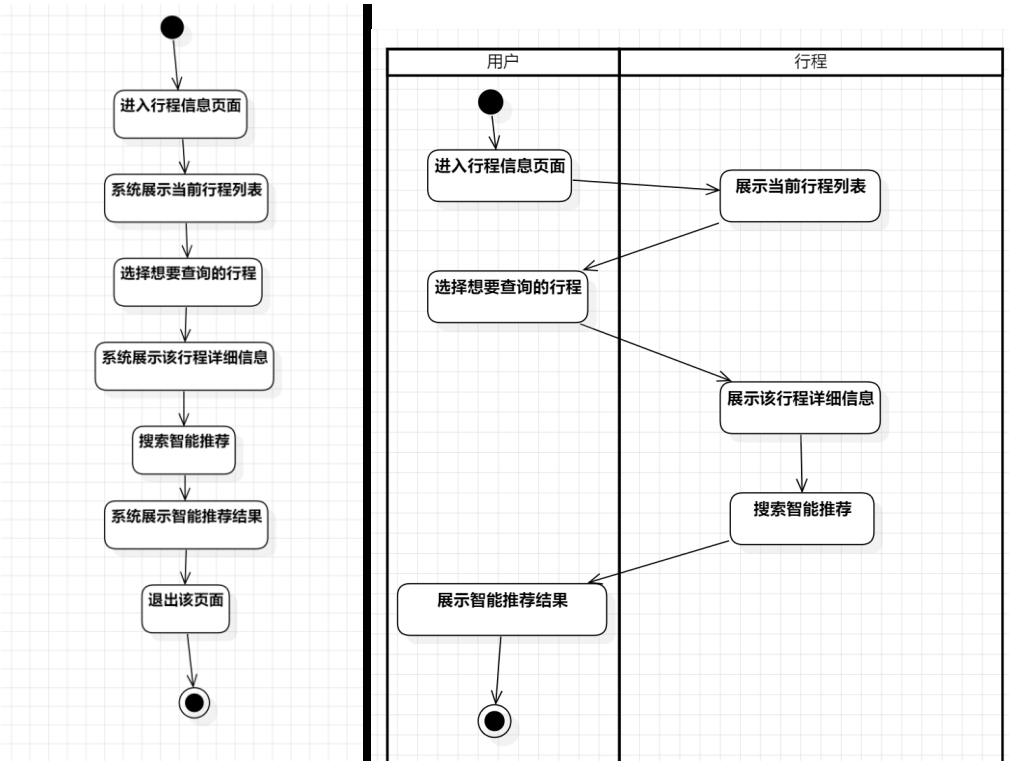
- 行程规划


进入行程信息页面
系统展示当前行程列表
添加行程
修改行程
删除行程
管理事件
系统提示用户输入新行程信息
系统展示该行程的详细信息
系统更新行程列表
系统展示该行程的事件列表
用户输入所需信息
用户修改相应字段
退出该页面
系统保存新行程信息并展示
退出该页面
添加事件
修改事件
删除事件
系统提示用户输入新事件信息
系统展示该事件的详细信息
系统更新事件列表
系统输入所需信息
退出该页面
用户保存新事件信息并展示
用户修改相应字段
退出该页面



在行程规划用例中，用户首先进入行程界面，系统展示行程列表。用户可以进行四种操作：添加新行程、修改现有行程、删除行程或管理行程中的事件。添加新行程时，需要输入行程详细信息，包括起始时间、结束时间等；修改行程时，系统会显示当前行程信息供用户编辑；删除行程直接更新形成列表；管理事件时，系统展示该行程的事件列表，用户可选择添加事件、修改现有事件或删除事件。添加新事件时，需要输入事件详细信息，包括时间、地点等；修改行程时，系统会显示当前行程信息供用户编辑；删除行程直接更新形成列表。

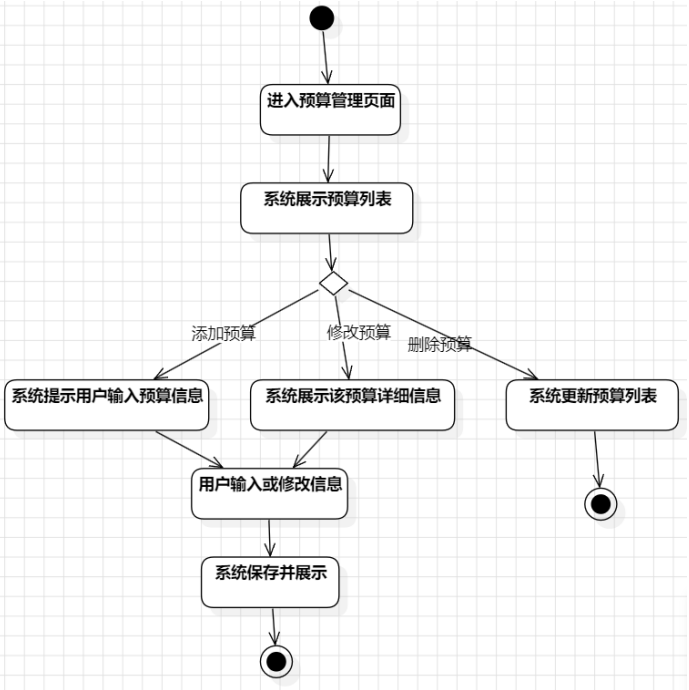
- **行程智能推荐**

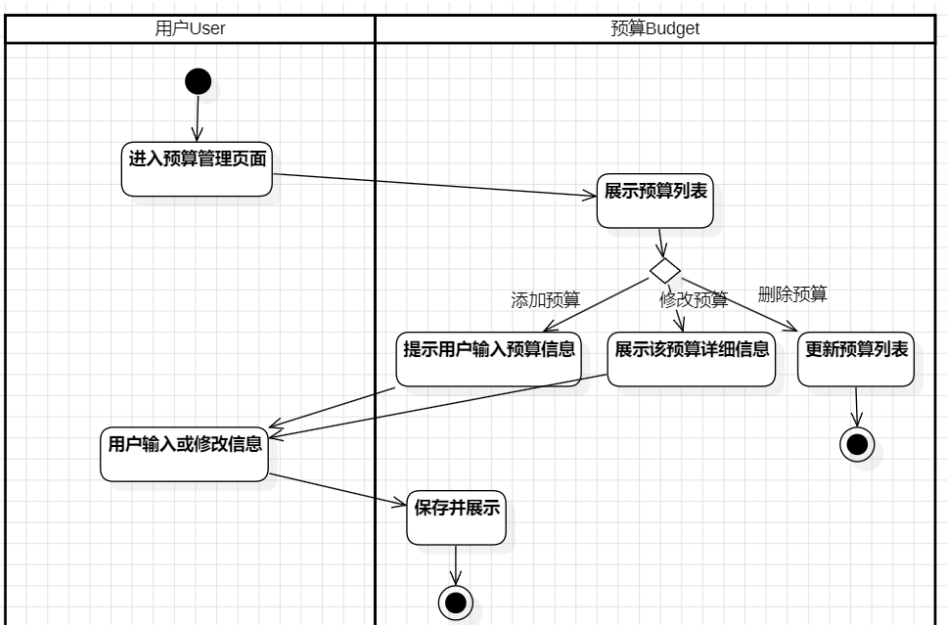


在行程智能推荐用例中，用户进入行程信息页面后，系统首先展示当前的行程列表。用户可以选择想要获取推荐的行程，系统会展示该行程的详细信息。推荐系统会考虑多个因素，如预算、时间等，生成个性化的行程建议。

开销管理子系统

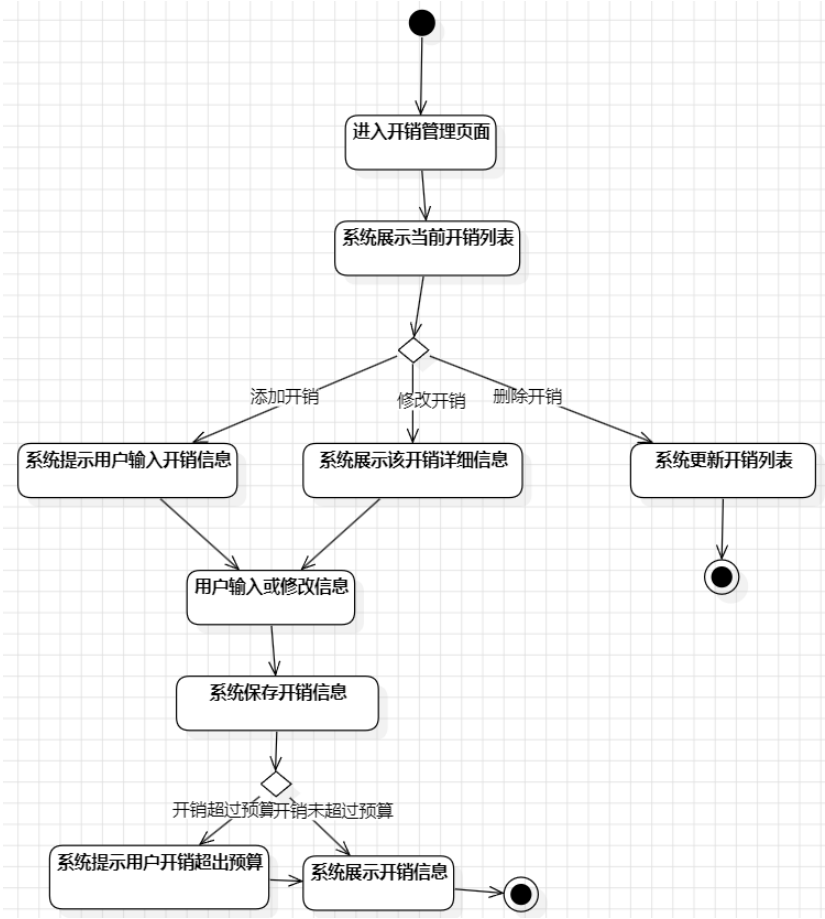
- **预算管理**

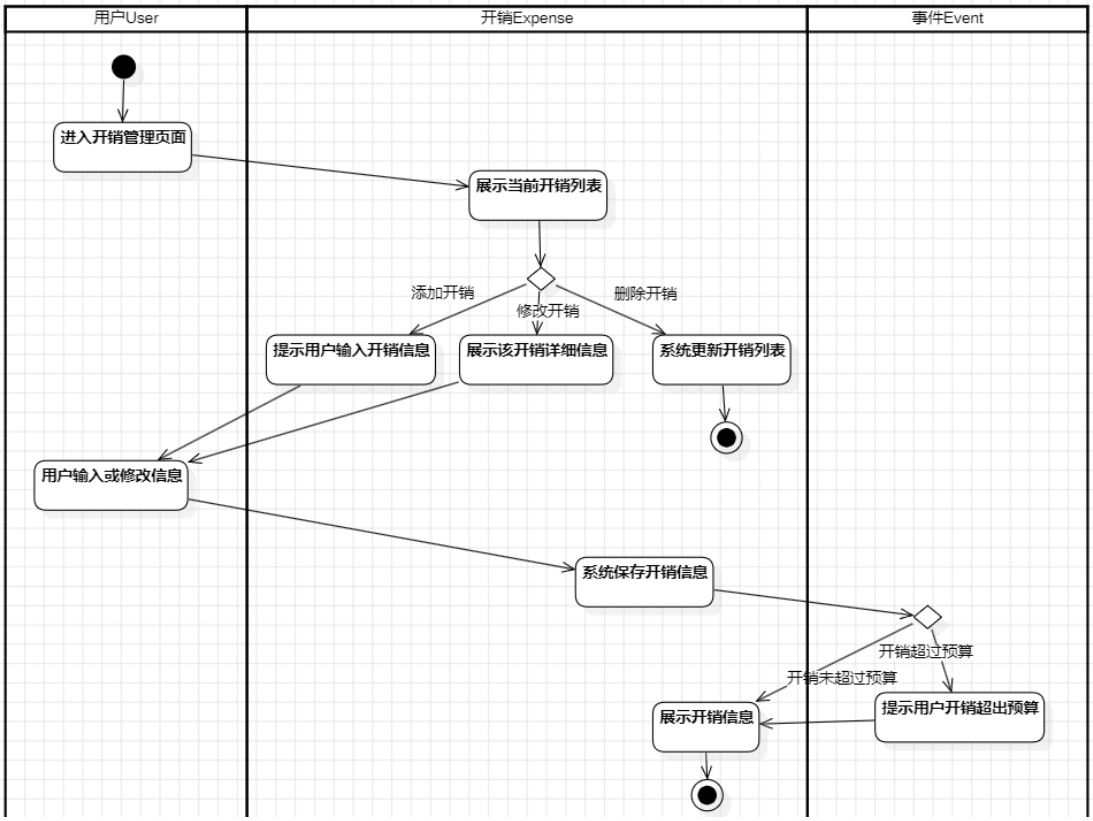




在预算管理用例中，用户进入预算管理界面后，系统展示预算列表，用户可以选择添加新的预算计划、修改现有预算或删除预算。添加预算时，用户需要输入预算类型、金额等信息；修改预算时，系统会显示当前的预算详细信息供用户修改；删除预算时，系统会直接更新预算列表。

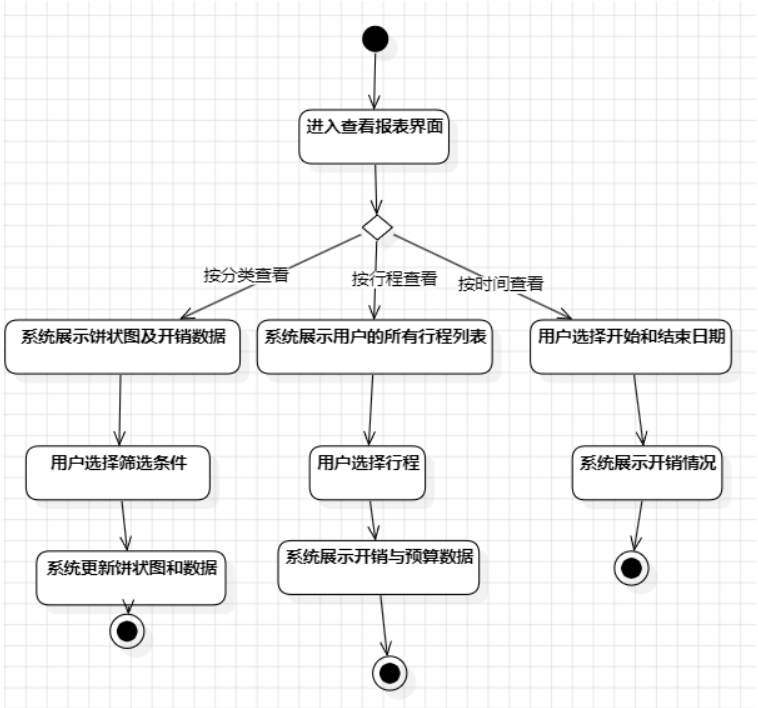
- **开销管理**

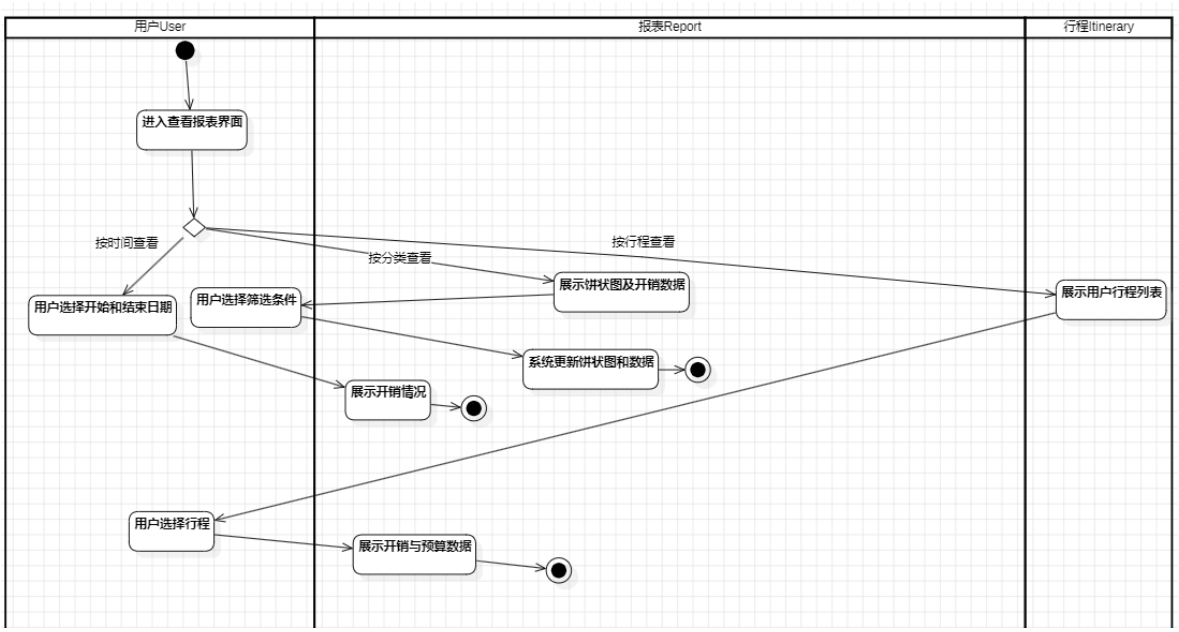




在预算管理用例中，用户进入开销管理界面后，系统展示开销列表，用户可以选择添加新的开销、修改现有开销或删除开销。添加开销时，用户需要输入开销类型、金额等信息；修改开销时，系统会显示当前的开销详细信息供用户修改；删除开销时，系统会直接更新开销列表。用户添加或修改开销后，系统会判断开销是否超过预算，若开销超过预算，则提示用户开销超出预算。

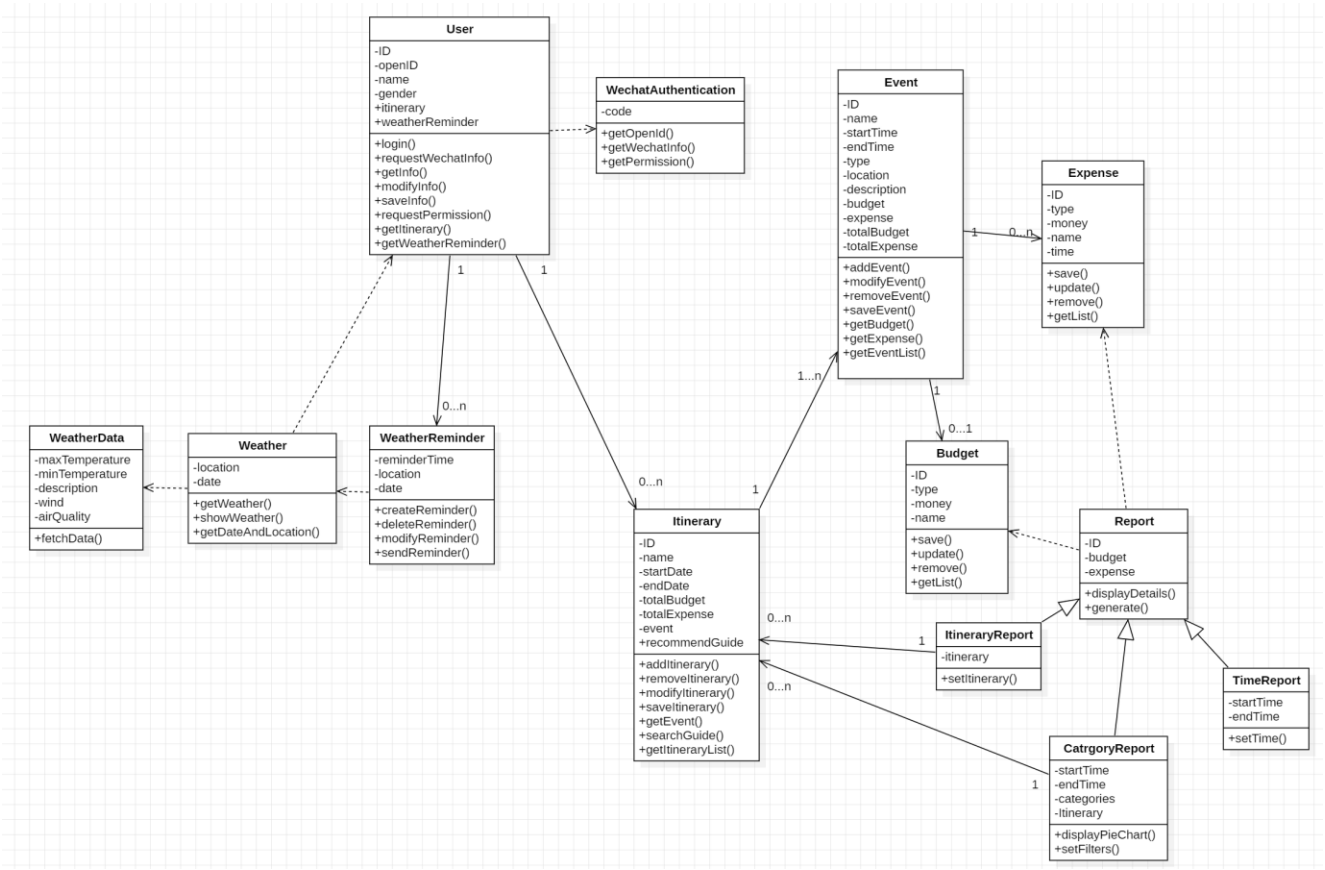
- **查看报表**





在查看报表用例中，用户可以选择不同类型的统计报表：按时间查看报表、按分类查看报表、按行程查看报表。如果按分类查看报表，用户可以选择筛选条件，系统会根据筛选条件，从数据库中提取相关数据进行处理和分析并展示；如果按时间查看报表，用户可以设置时间范围；如果按行程查看报表，系统会展示用户行程列表，用户选择行程后，系统提取数据并展示。

2.3.2 数据建模



清晰的职责划分：每个类都专注于特定的功能，例如用户、天气、行程等。

高内聚性：类内方法与属性紧密相关，便于维护和扩展。

低耦合性：类之间通过关联、依赖等关系连接，保持了适度的耦合，便于模块独立开发和测试。

面向对象原则：便于扩展功能。

- **User 类**

属性

ID: 唯一标识用户的属性。

openID: 微信账户对应的 openID。

name: 用户的昵称。

gender: 用户性别。

itinerary: 与用户相关的行程信息。

weatherReminder: 用户设定的天气提醒。

方法

login(): 用户登录功能。

requestWechatInfo(): 请求微信相关信息。

getInfo(): 获取用户基本信息。

modifyInfo(): 修改用户信息。

saveInfo(): 保存用户信息。

requestPermission(): 请求权限，比如访问位置。

getItinerary(): 获取用户的行程信息。

getWeatherReminder(): 获取天气提醒信息。

关系

与 Itinerary 类和 WeatherReminder 类有关联，表示一个用户可以有多个行程和天气提醒。

通过 WechatAuthentication 类间接实现身份验证、权限设置和得到微信相关信息。

- **WechatAuthentication 类**

属性

code: 认证过程中与微信账户一一对应。

方法

getOpenId(): 获取用户的微信唯一标识。

getWechatInfo(): 获取微信账号的详细信息。

getPermission(): 请求用户授权。

关系

与 User 类通过依赖关系关联，提供微信认证支持。

- **Weather 类**

属性

location: 天气对应的地点。

date: 天气信息的日期。

方法

getWeather(): 获取天气信息。

showWeather(): 显示天气信息。

getDateAndLocation(): 获取日期和地点。

关系

与 WeatherData 类通过关联关系连接，表示天气信息是由具体的数据组成。

与 WeatherReminder 类关联，为天气提醒提供数据支持。

- **WeatherData 类**

属性

maxTemperature: 最高温度。

minTemperature: 最低温度。

description: 天气描述。

wind: 风速或风力。

airQuality: 空气质量。

方法

fetchData(): 获取天气数据。

关系

提供详细天气数据，与 Weather 类关联。

- **WeatherReminder 类**

属性

reminderTime: 提醒的时间。

location: 提醒对应的地点。

date: 提醒日期。

方法

createReminder(): 创建新的天气提醒。

deleteReminder(): 删除已有的提醒。

modifyReminder(): 修改提醒内容。

sendReminder(): 向用户发送天气提醒。

关系

与 User 类关联，一个用户可以创建多个天气提醒。

与 Weather 类连接，获取天气信息进行提醒。

- **Itinerary 类**

属性

ID: 唯一标识行程的编号。

name: 行程的名称。

startDate: 行程的开始日期。

endDate: 行程的结束日期。

totalBudget: 行程的预算总额。

totalExpense: 行程的总开销。

event: 行程包含的事件。

方法

addItinerary(): 新增行程。

removeItinerary(): 删除行程。

modifyItinerary(): 修改行程。

saveItinerary(): 保存行程信息。

searchGuide(): 行程智能规划。

getItineraryList(): 获取行程列表。

关系

与 User 类关联，一个用户可以拥有多个行程。

与 Event 类关联，一个行程包含多个事件。

与 Budget 和 Expense 类相关联，为行程提供预算和开销管理支持。

- **Event 类**

属性

ID: 事件的唯一标识。

name: 事件的名称。

startTime: 事件的开始时间。

endTime: 事件的结束时间。

type: 事件的类型。

location: 事件的地点。

description: 事件的描述。

budget: 事件的预算。

expense: 事件的支出。

totalBudget: 事件的预算总额。

totalExpense: 事件的支出总额。

方法

addEvent(): 添加事件。

modifyEvent(): 修改事件信息。

removeEvent(): 删除事件。

saveEvent(): 保存事件信息。

getBudget(): 获取事件预算。

getExpense(): 获取事件支出。

getEventList(): 获取事件列表。

关系

关联到 Itinerary 类，一个行程包含多个事件。

与 Budget 和 Expense 类关联，管理预算和支出。

- **Budget 类**

属性

ID: 预算的唯一标识。

type: 预算类型。

money: 预算金额。

name: 预算名称。

方法

save(): 保存预算信息。

update(): 更新预算内容。

remove(): 删除预算。

getList(): 获取预算列表。

关系

与 Event 类关联，为事件提供预算支持。

- **Expense 类**

属性

ID: 开销的唯一标识。

type: 开销类型。

money: 开销金额。

name: 开销名称。

time: 开销时间。

方法

save(): 保存开销信息。

update(): 更新开销记录。

remove(): 删除开销记录。

getList(): 获取开销列表。

关系

与 Event 相关联，记录事件具体支出情况。

- **Report 类**

属性

ID: 报告的唯一标识。

budget: 报告的预算信息。

expense: 报告的支出信息。

方法

displayDetails(): 显示报告的详细内容。

generate(): 生成报告。

关系

依赖于开销和预算。

包含三个具体的子类：ItineraryReport、TimeReport 和 CategoryReport。

- **ItineraryReport 类**

属性

itinerary: 报告对应的行程。

方法

setItinerary(): 设置与报告相关的行程。

关系

是 Report 类的子类，专注于行程相关的报告。

与行程关联，一个行程报告可以对应多个行程。

- **TimeReport 类**

属性

startTime: 报告的起始时间。

endTime: 报告的结束时间。

方法

setTime(): 设置时间范围。

关系

是 Report 类的子类，专注于特定时间范围内的支出和预算分析。

- **CategoryReport 类**

属性

startTime: 报告的起始时间。

endTime: 报告的结束时间。

categories: 报告的支出类别。

itinerary: 与报告相关的行程。

方法

displayPieChart(): 显示支出类别的饼图。

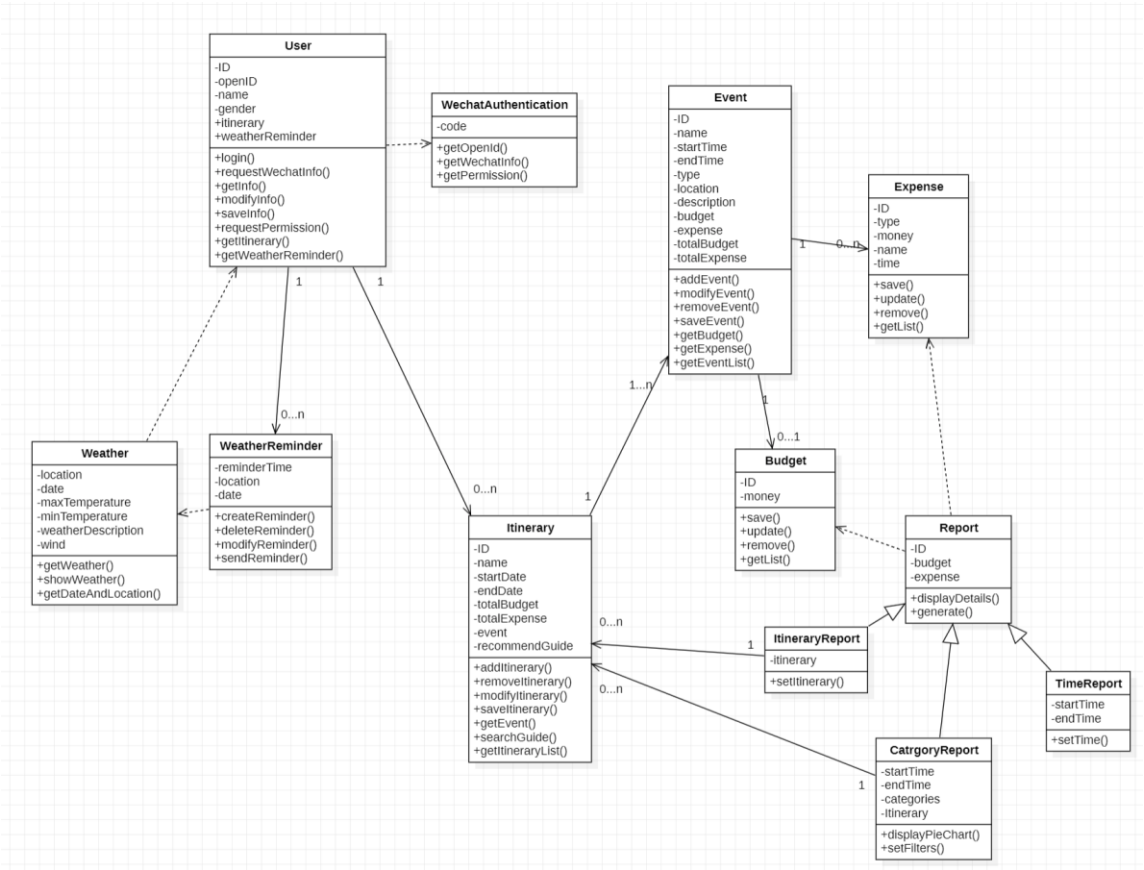
setFilters(): 设置报告过滤条件。

关系：

是 Report 类的子类，侧重于按类别分析支出。

与行程关联，一个类型报告可以对应多个行程。

Iteration 2：



- **Weather 类**

属性

location: 天气对应的地点。

date: 天气信息的日期。

maxTemperature: 最高温度。

minTemperature: 最低温度。

description: 天气描述。

wind: 风速或风力。

方法

getWeather(): 获取天气信息。

showWeather(): 显示天气信息。

getDateAndLocation(): 获取日期和地点。

关系

与 WeatherReminder 类关联，为天气提醒提供数据支持。

Note：将天气数据类合并入天气类，通过 getWeather 方法直接调用高德地图 API 获取天气详细信息。

- **Budget 类**

属性

ID: 预算的唯一标识。

money: 预算金额。

方法

save(): 保存预算信息。

update(): 更新预算内容。

remove(): 删除预算。

getList(): 获取预算列表。

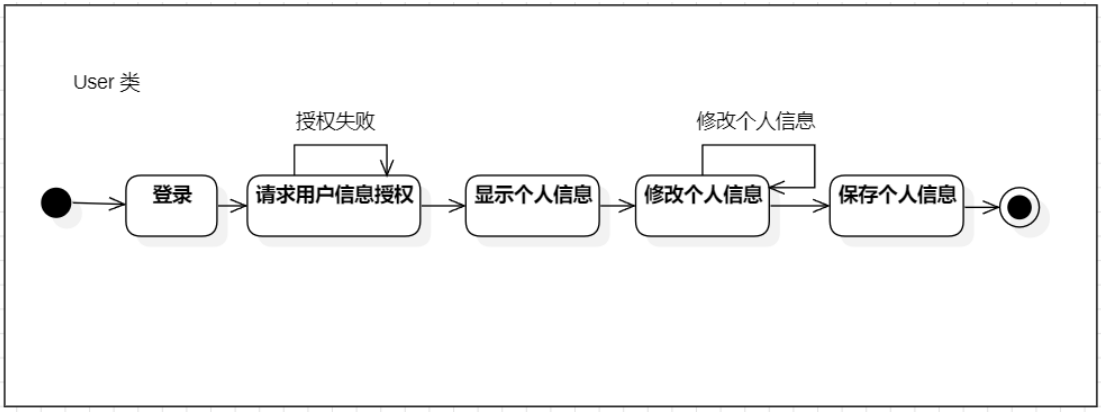
关系

与 Event 类关联，为事件提供预算支持。

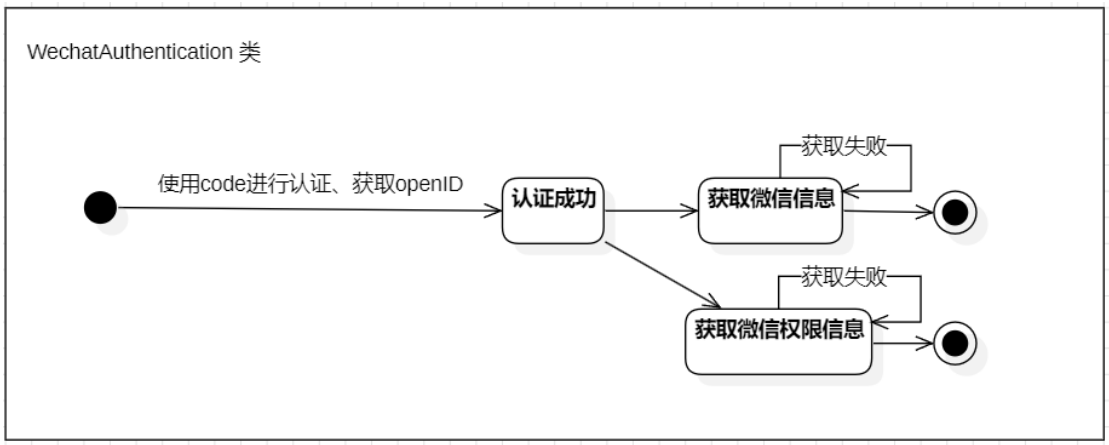
Note：删除了预算类的类型和名称，因为一个事件只有 0/1 个预算，所以它不需要类型和名称来做区分。

2.3.3 行为建模

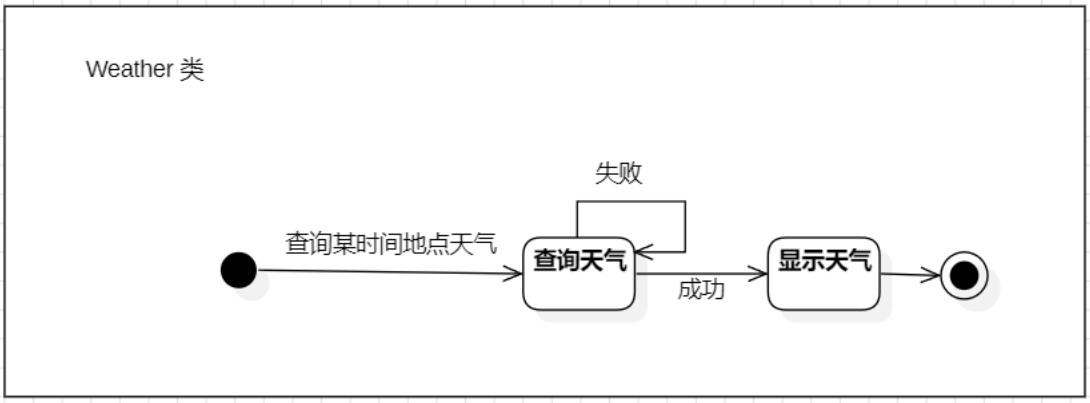
状态图



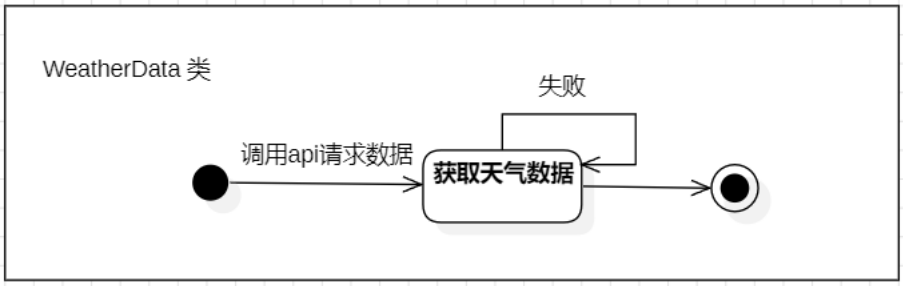
初始状态为用户未登录。调用 login 函数后到达登录状态，调用 requestPermission 函数后到达请求用户信息授权状态，调用 getInfo 函数后到达显示个人信息状态，调用 modifyInfo 函数后到达修改个人信息状态，调用 saveInfo 函数后到达保存个人信息状态。



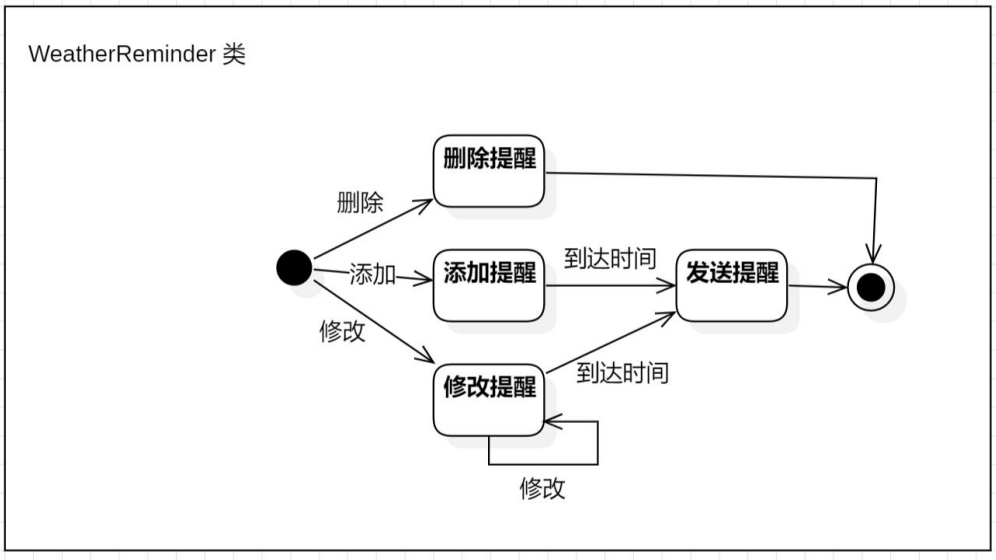
初始状态为等待验证，调用 getOpenId 函数后，到达认证成功状态，此时，若调用 getWechatInfo 函数，则到达获取微信信息状态；若调用 getPermission 函数，则到达获取微信权限信息状态。



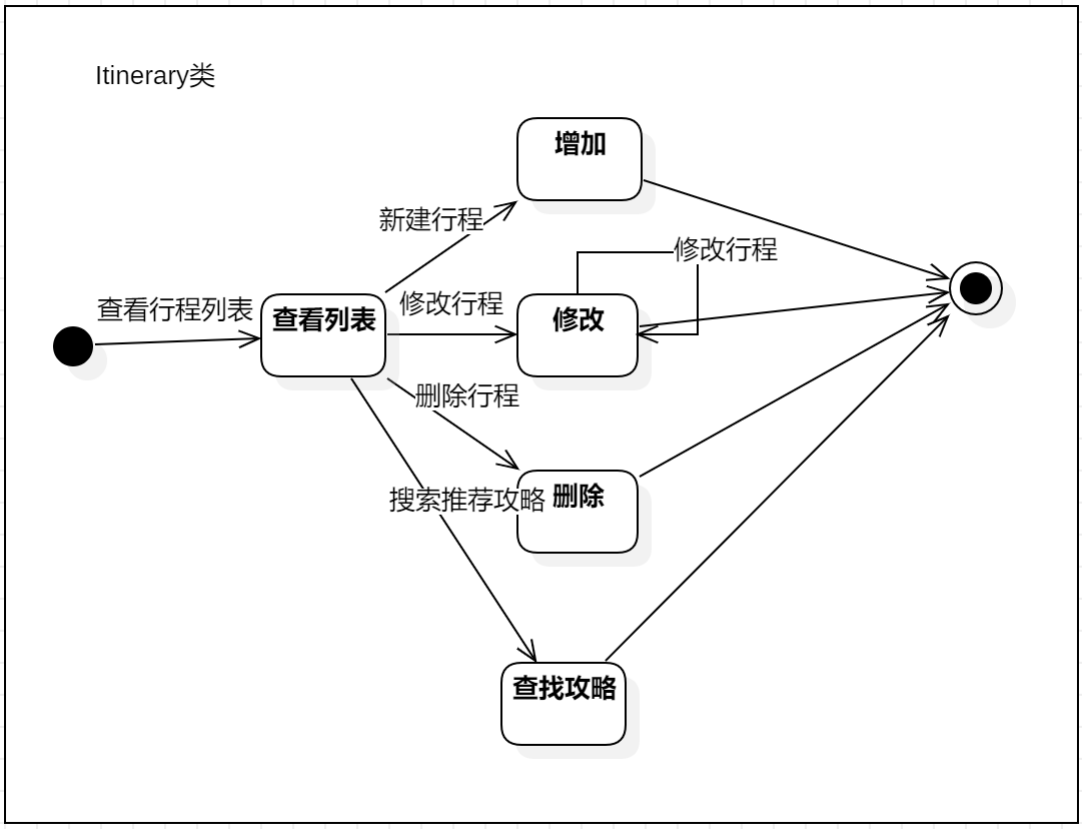
初始状态为等待查询状态，调用 getWeather 函数后，到达查询天气状态，调用 showWeather 函数后，到达显示天气状态。



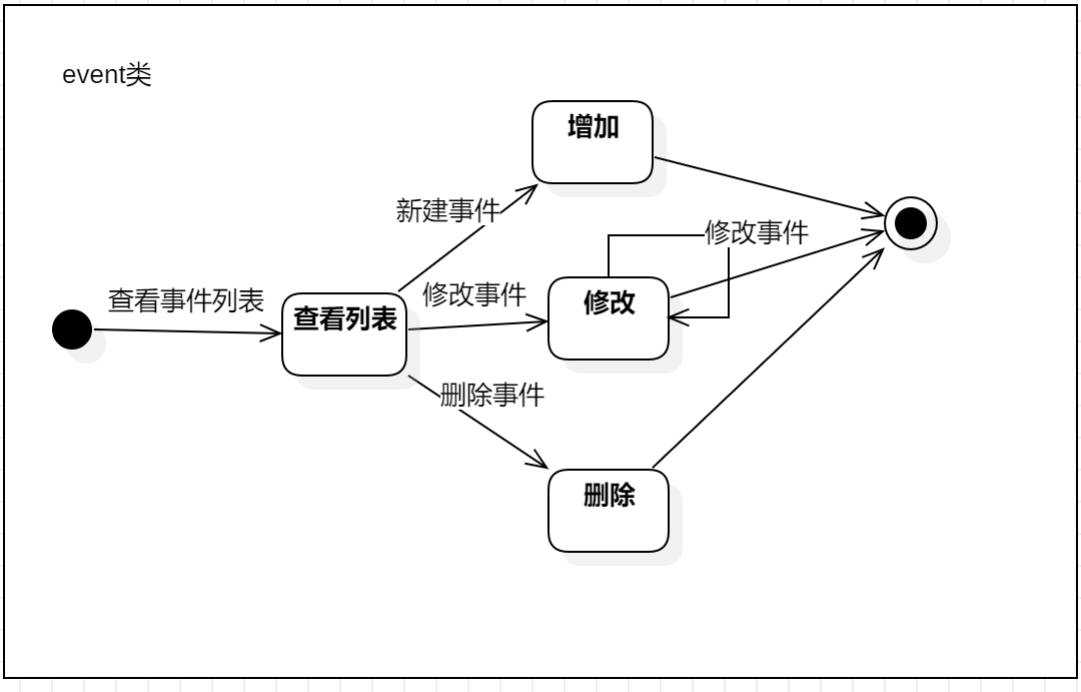
初始状态为等待请求状态，调用 fetchData 函数后，到达获取天气数据状态。



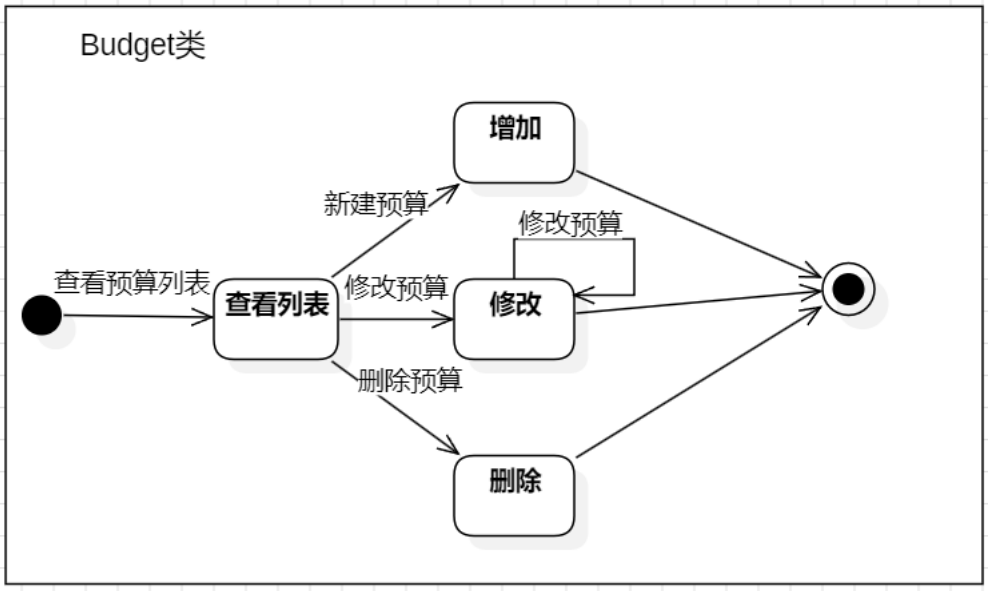
初始状态为等待操作状态，若调用 createReminder 函数，则到达添加提醒状态；若调用 modifyReminder 函数，到达修改提醒状态；若调用 deleteReminder 函数，到达删除提醒状态。在添加提醒状态和修改提醒状态，到达时间后，调用 sendReminder 函数，到达发送提醒状态。



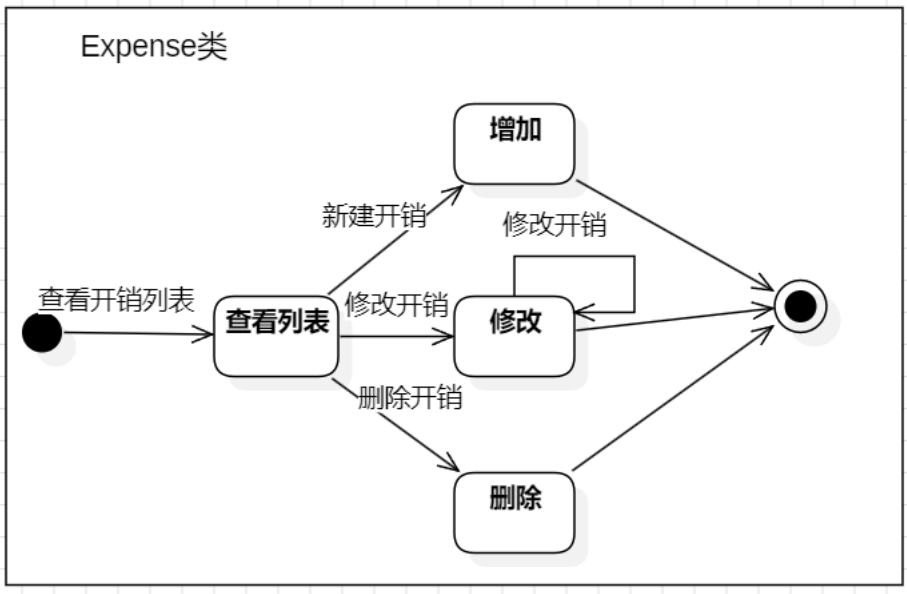
初始状态为等待操作状态，调用 getItineraryList 函数，则到达查看行程列表状态，在此状态下，若调用 addItinerary 函数，则到达添加行程状态；若调用 modifyItineraryr 函数，到达修改行程状态；若调用 removeItinerary 函数，到达删除行程状态；若调用 SearchGuide 函数，则到达查找攻略状态。



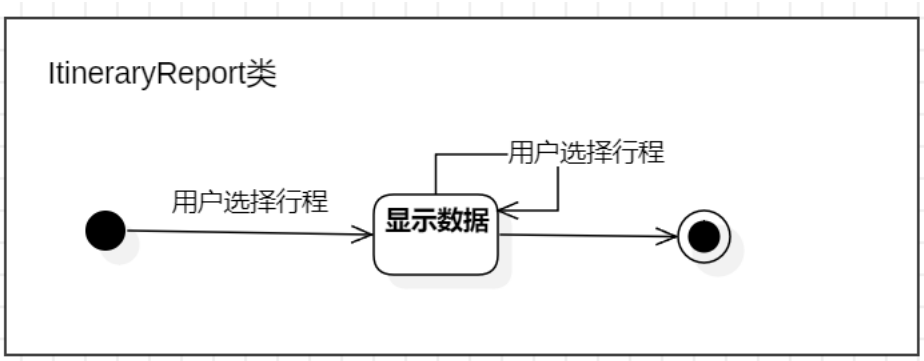
初始状态为等待操作状态，调用 getEventList 函数，则到达查看事件列表状态，在此状态下，若调用 addEvent 函数，则到达添加事件状态；若调用 modifyEvent 函数，到达修改事件状态；若调用 removeEvent 函数，到达删除事件状态。



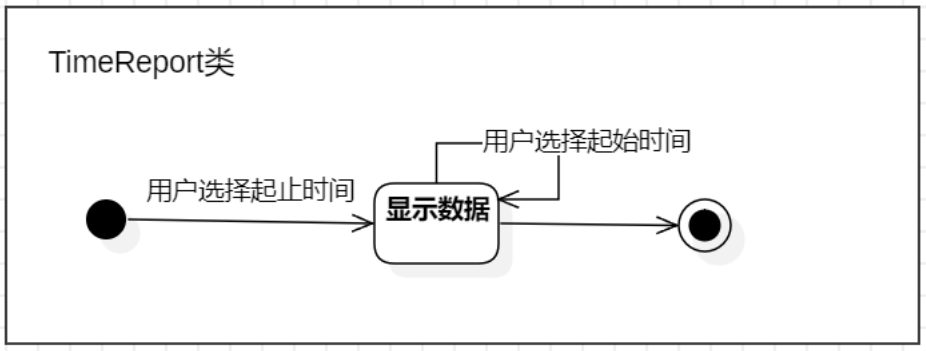
初始状态为等待操作状态，调用 getEventList 函数，则到达查看事件列表状态，在此状态下，若调用 addEvent 函数，则到达添加事件状态；若调用 modifyEvent 函数，到达修改事件状态；若调用 removeEvent 函数，到达删除事件状态。



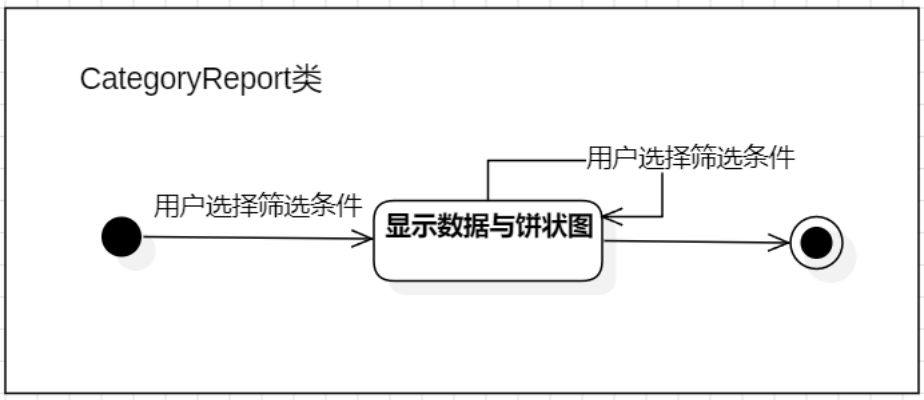
初始状态为等待操作状态，调用 getList 函数，则到达查看支出列表状态，在此状态下，若调用 save 函数，则到达添加支出状态；若调用 update 函数，到达修改支出状态；若调用 remove 函数，到达删除支出状态。



初始状态为等待操作状态，调用 setItinerary 函数，则到达显示行程数据状态。



初始状态为等待操作状态，调用 setTime 函数，则到达显示时段数据状态。



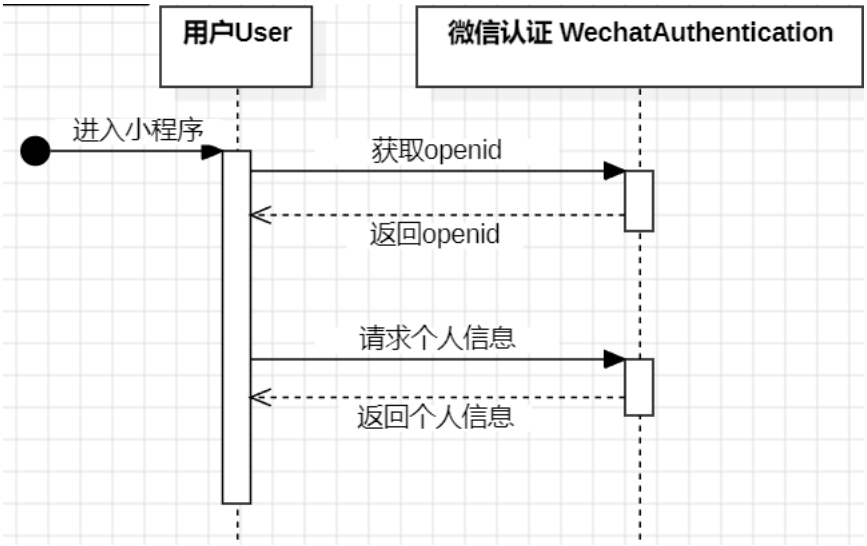
初始状态为等待操作状态，调用 setFilters 函数，则到达显示数据与饼状图状态。

Iteration 2：

删除 WeatherData 类的状态图。

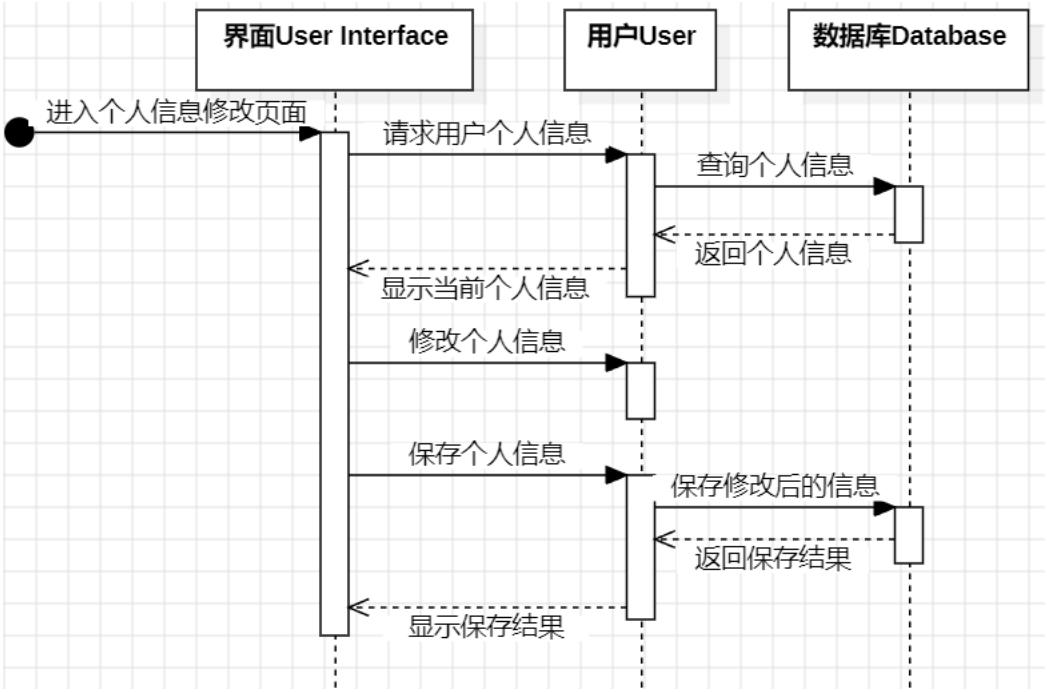
时序图

- 登录账号



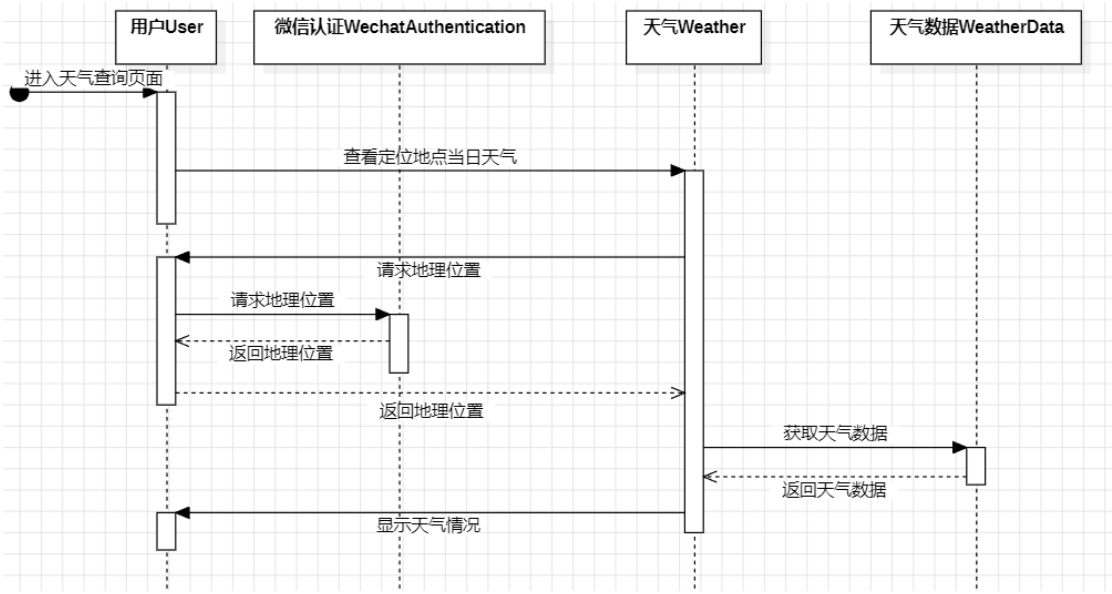
用户进入小程序后，首先向微信认证服务请求获取唯一标识 openid，随后基于 openid 请求用户的个人信息。微信认证服务依次返回对应的数据，完成用户身份认证及信息获取。

- 个人信息设置

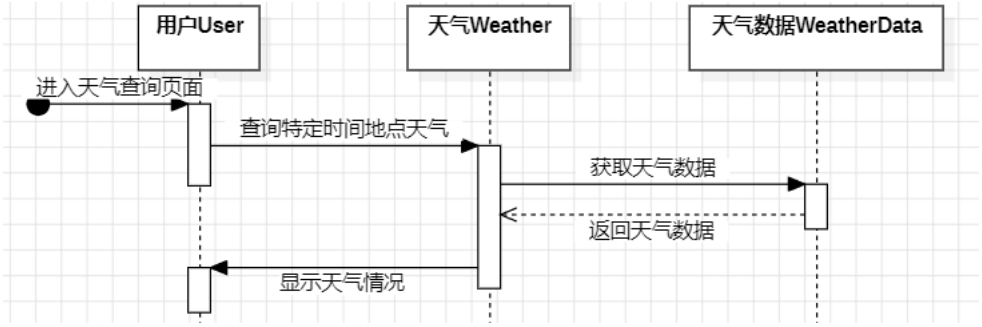


用户进入修改页面后，系统向数据库请求并查询当前的个人信息，返回后在页面显示。用户修改个人信息后，提交保存请求，系统将修改后的信息保存到数据库，数据库返回保存结果，最后页面显示保存的操作结果。

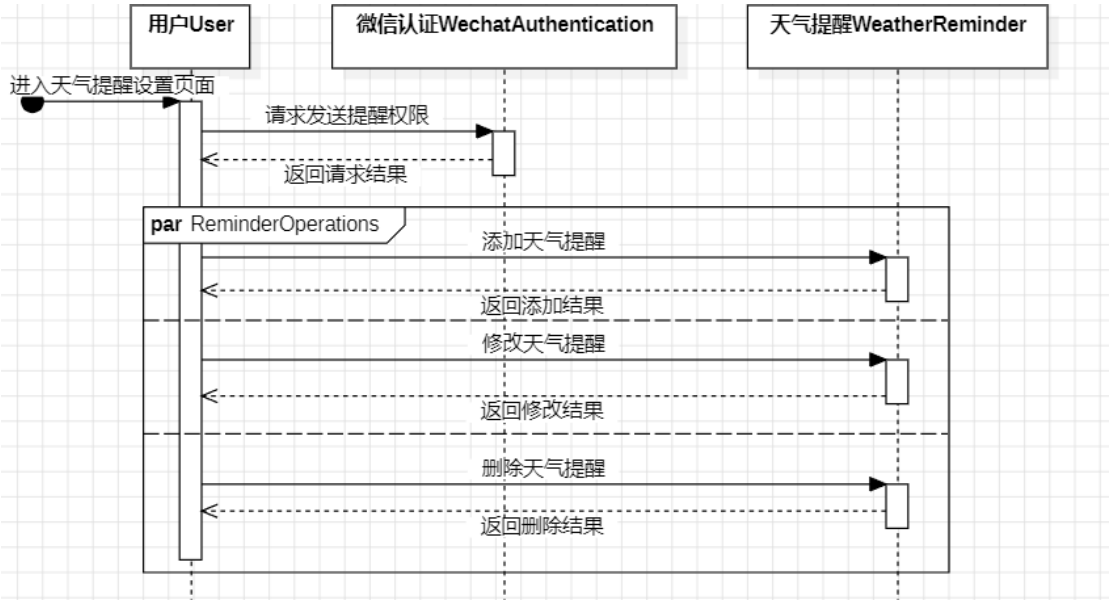
- 查询天气及接受提醒



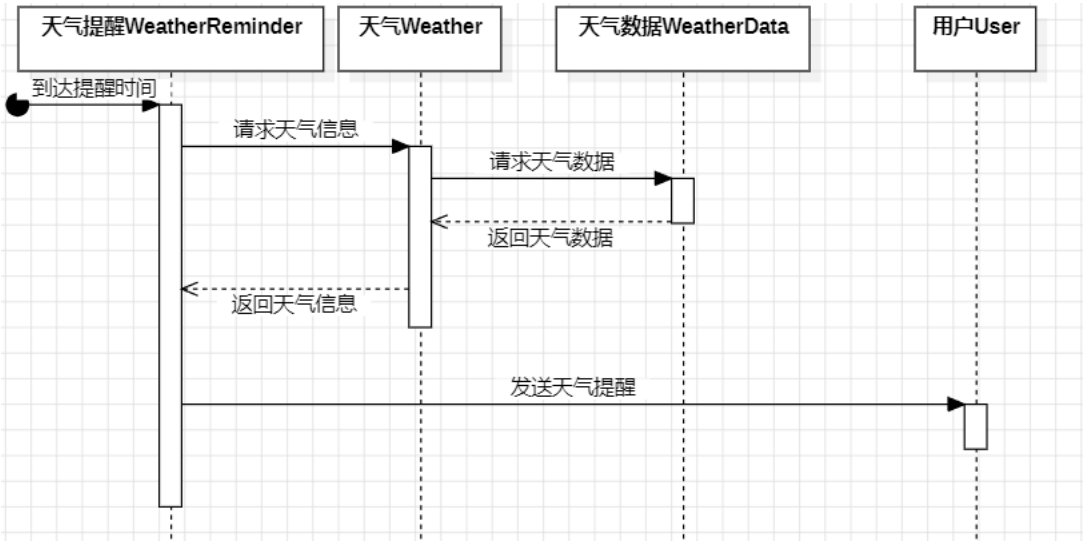
用户进入天气查询页面后，向天气类发送请求查看定位地点的当日天气，天气类请求地理位置，系统通过微信认证请求地理位置，获取用户当前的位置并返回。随后，系统根据地理位置向天气服务请求当日的天气数据，天气服务从天气数据源获取天气信息并返回给系统，最后将天气情况显示给用户。



用户进入天气查询页面后，向天气类发送请求查看特定地点的当日天气，系统根据选择的特定位置向天气服务请求当日的天气数据，天气服务从天气数据源获取天气信息并返回给系统，最后将天气情况显示给用户。

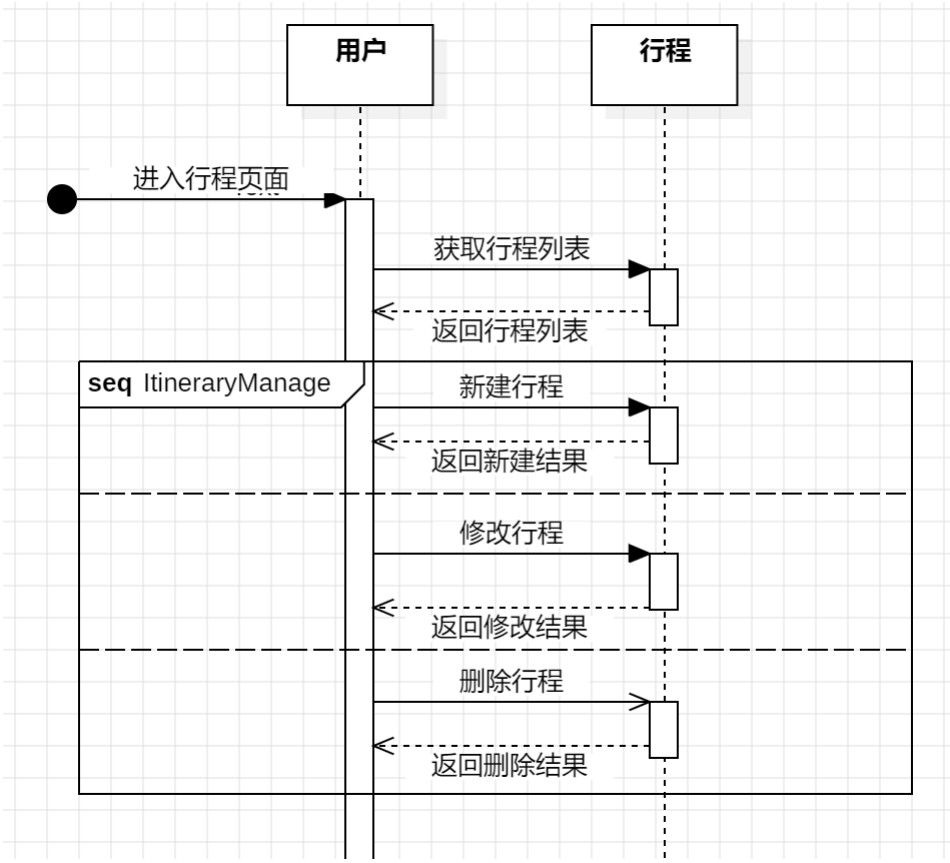


当用户进入天气提醒设置页面后，首先需要请求发送提醒权限并获得返回结果。之后在 ReminderOperations 部分，用户可以进行三个主要操作：添加天气提醒、修改天气提醒和删除天气提醒。每个操作执行后系统向天气提醒发送请求，返回相应的操作结果。



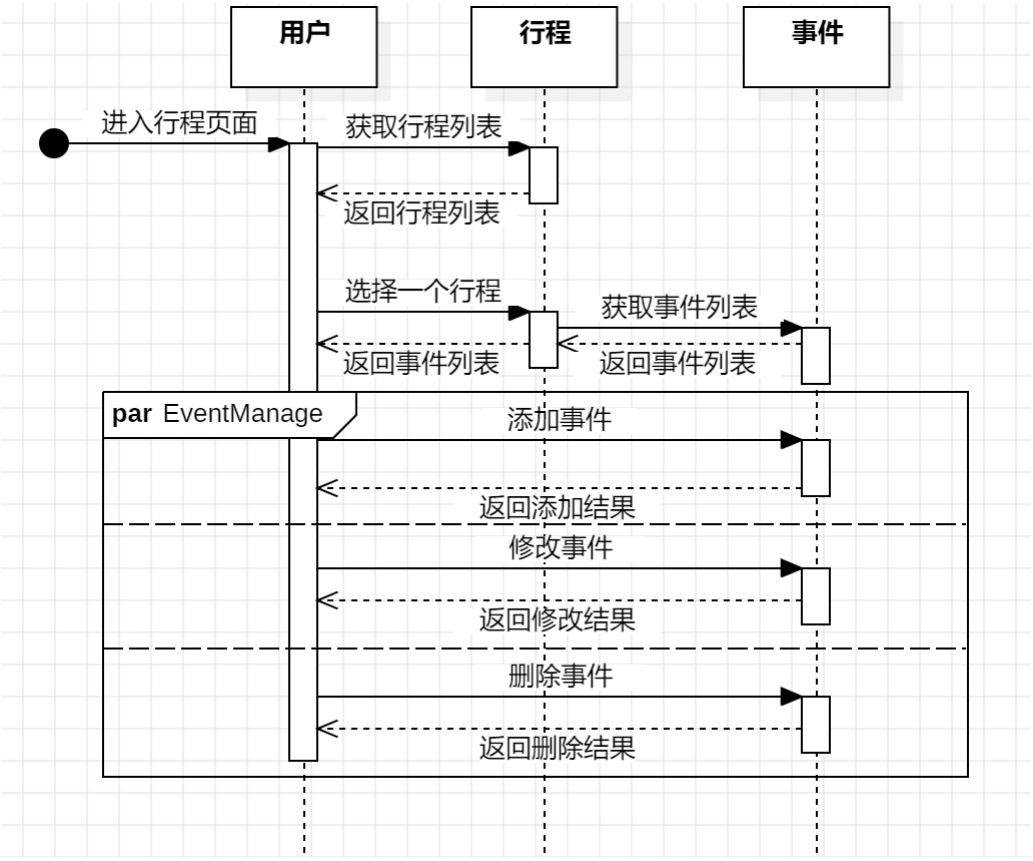
当到达预设的提醒时间时，天气提醒系统首先向天气系统请求天气信息，天气系统随后向天气数据源请求天气数据并获得返回。在天气系统得到天气数据后，将天气信息返回给天气提醒系统。最后，天气提醒系统将处理好的天气提醒信息发送给用户。

- 行程规划



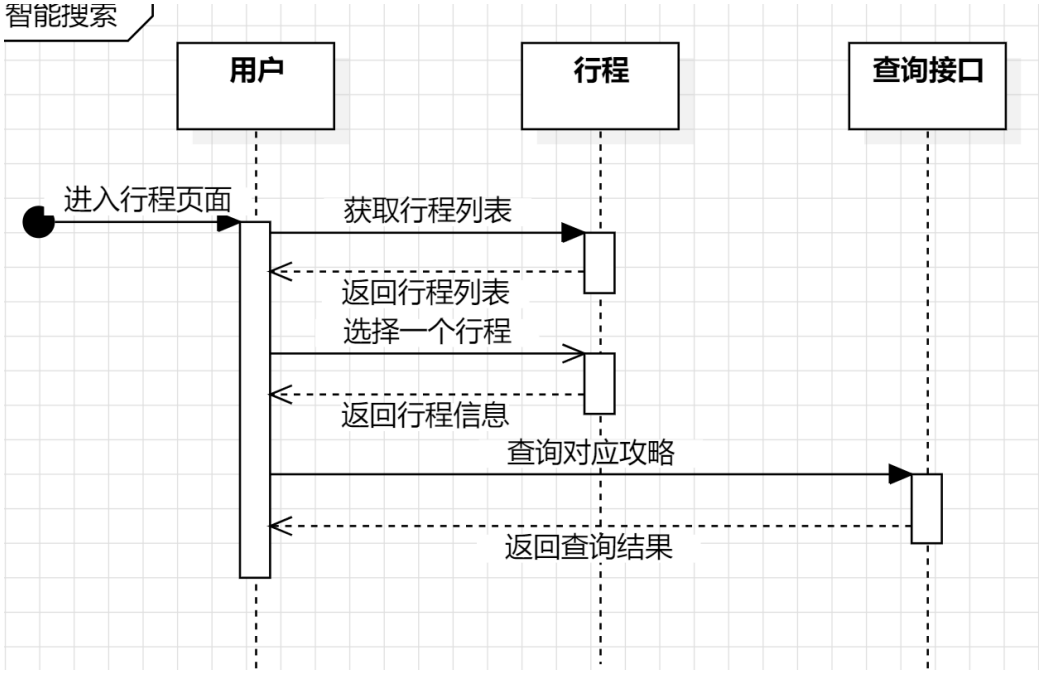
当用户进入行程页面后，首先获取现有的行程列表。在 ItineraryManage 流程块中，用户可以进行三个主要操作：新建行程、修改行程和删除行程。每个操

作都会对行程类发送相应请求，执行后系统都会返回相应的操作结果。



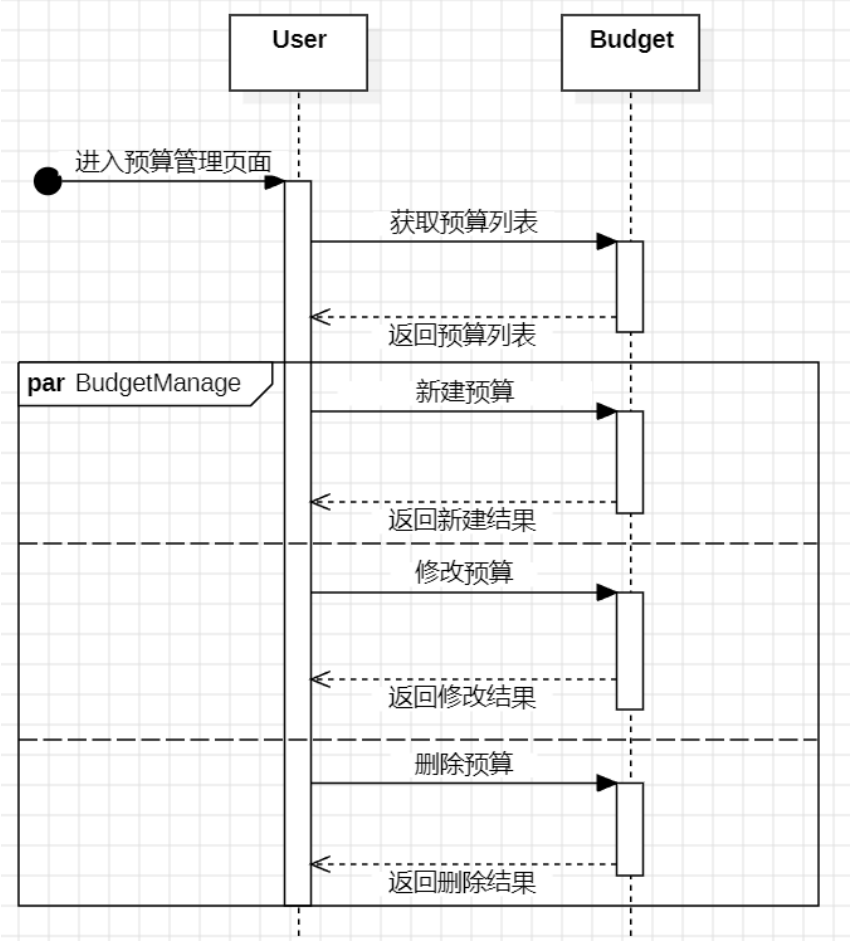
当用户进入行程页面后，首先获取现有的行程列表。用户选择一个行程后可以返回该行程对应的事件列表。在 EventManage 流程块中，用户可以进行三个主要操作：新建事件、修改事件和删除事件。每个操作都会对事件类发送相应请求，执行后系统都会返回相应的操作结果。

- 行程智能推荐



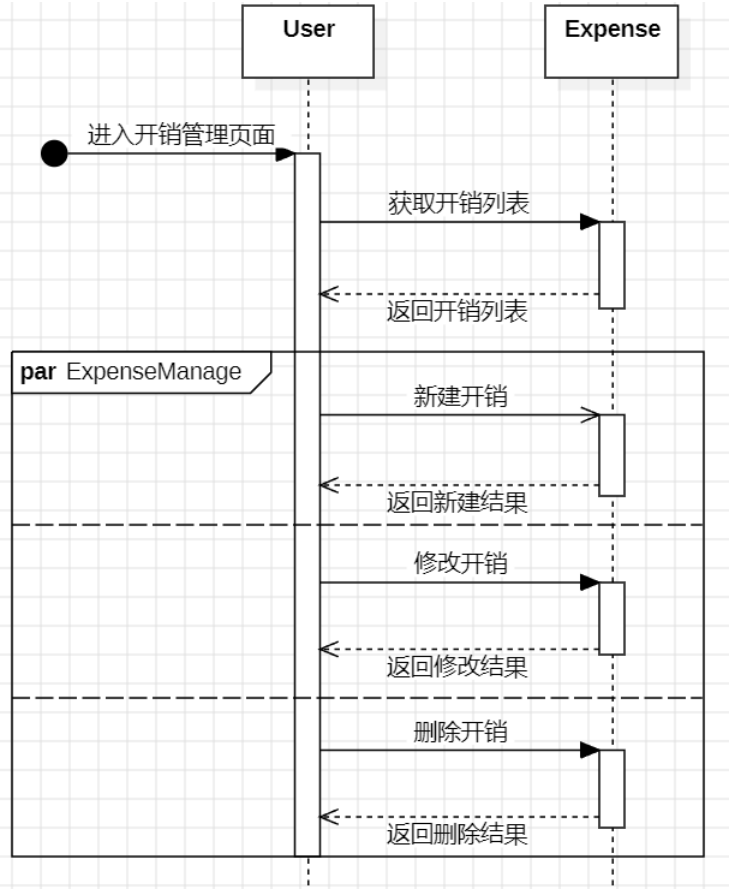
当用户进入行程页面后，首先获取现有的行程列表。用户选择一个行程后，形成类返回该行程的具体信息。用户查询对应的攻略，将得到查询接口得到的查询结果。

- 预算管理



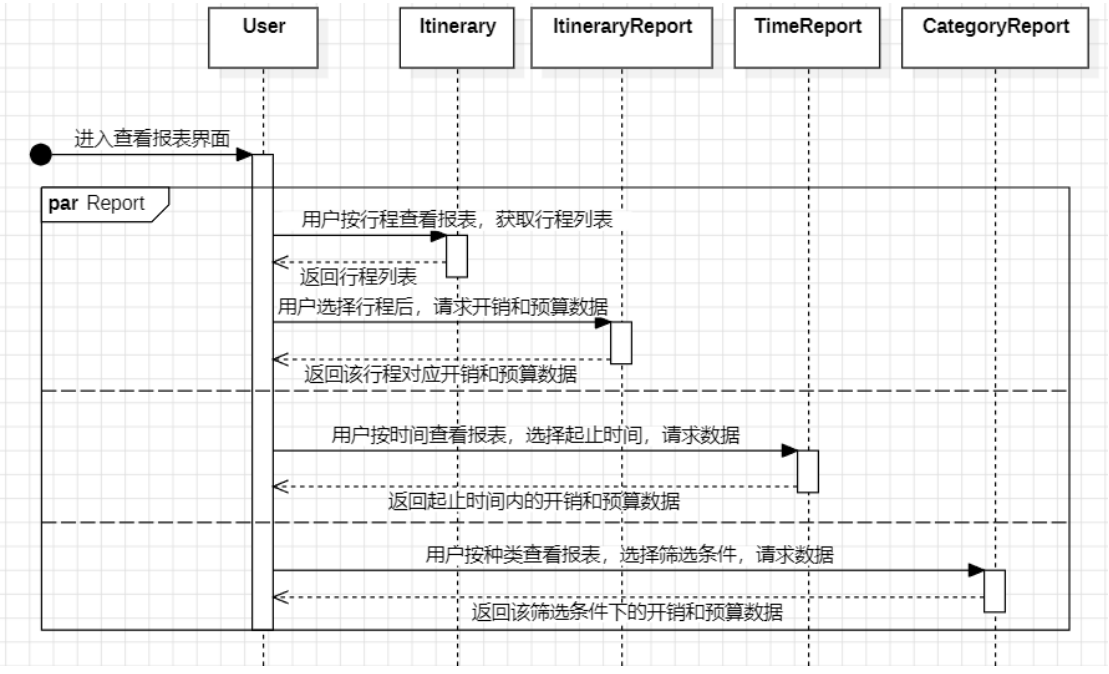
当用户进入预算管理页面后，首先获取现有的预算列表。在 BudgetManage 流程块中，用户可以进行三个主要操作：新建预算、修改预算和删除预算。每个操作都会对预算类发送相应请求，执行后系统都会返回相应的操作结果。

- 开销管理



当用户进入开销管理页面后，首先获取现有的开销列表。在 ExpenseManage 流程块中，用户可以进行三个主要操作：新建开销、修改开销和删除开销。每个操作都会对开销类发送相应请求，执行后系统都会返回相应的操作结果。

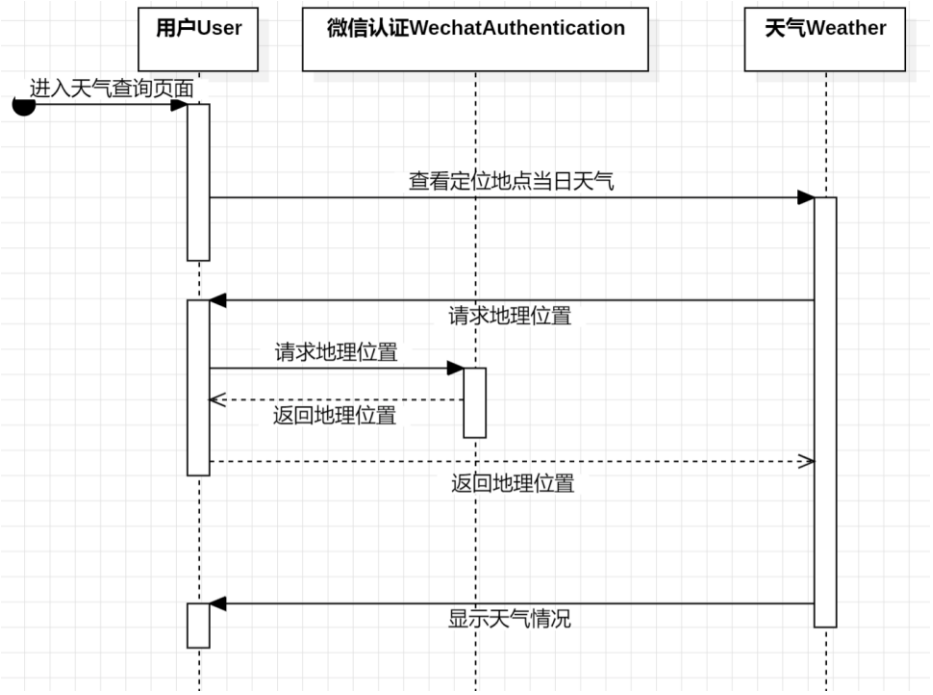
- 查看报表



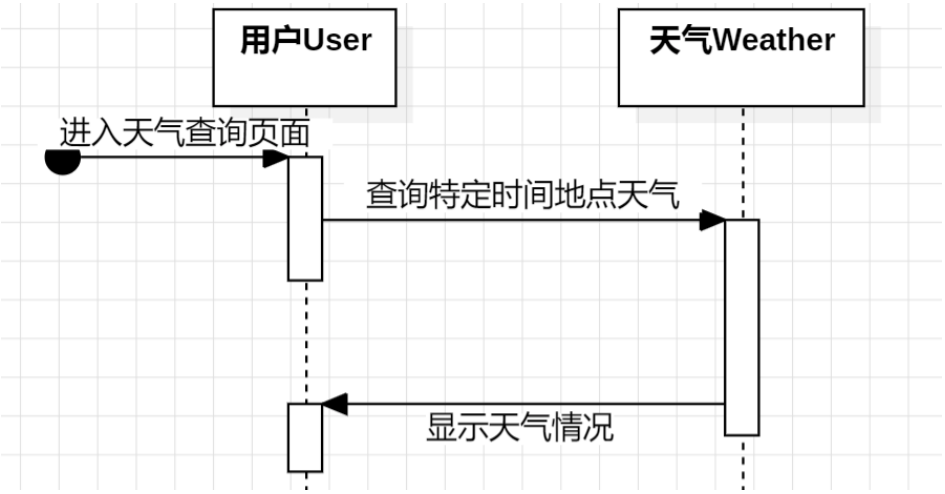
当用户进入查看报表页面后，在 Report 流程块中，用户可以进行三个主要操作：按行程查看报表、按时间查看报表和按种类查看报表。每个操作都会对不同的报表类发送相应请求，执行后都会返回相应的数据。

Iteration 2：

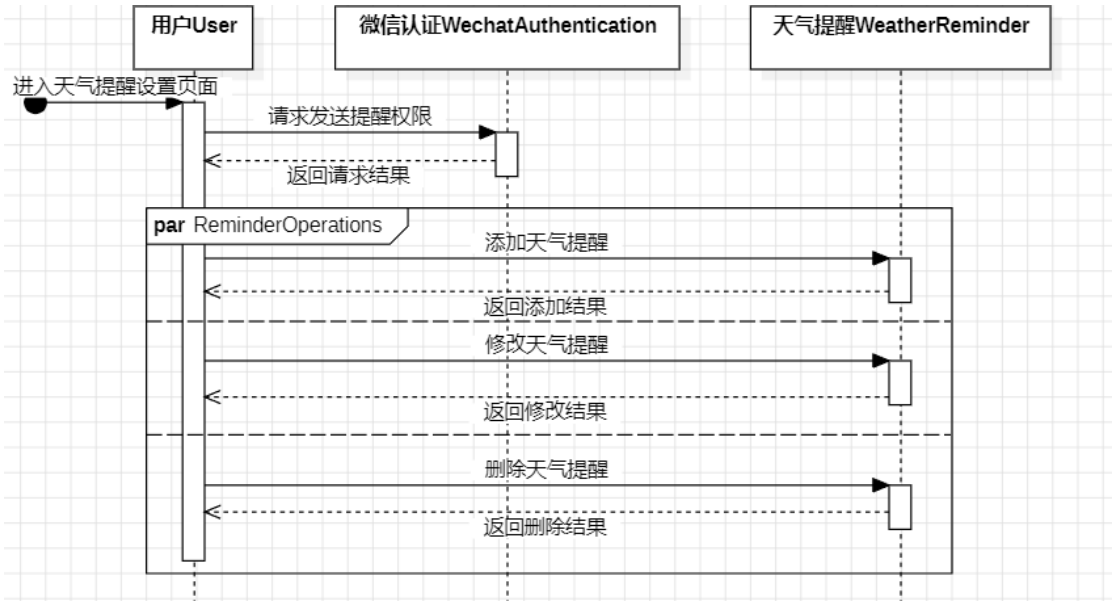
- 查询天气及接受提醒



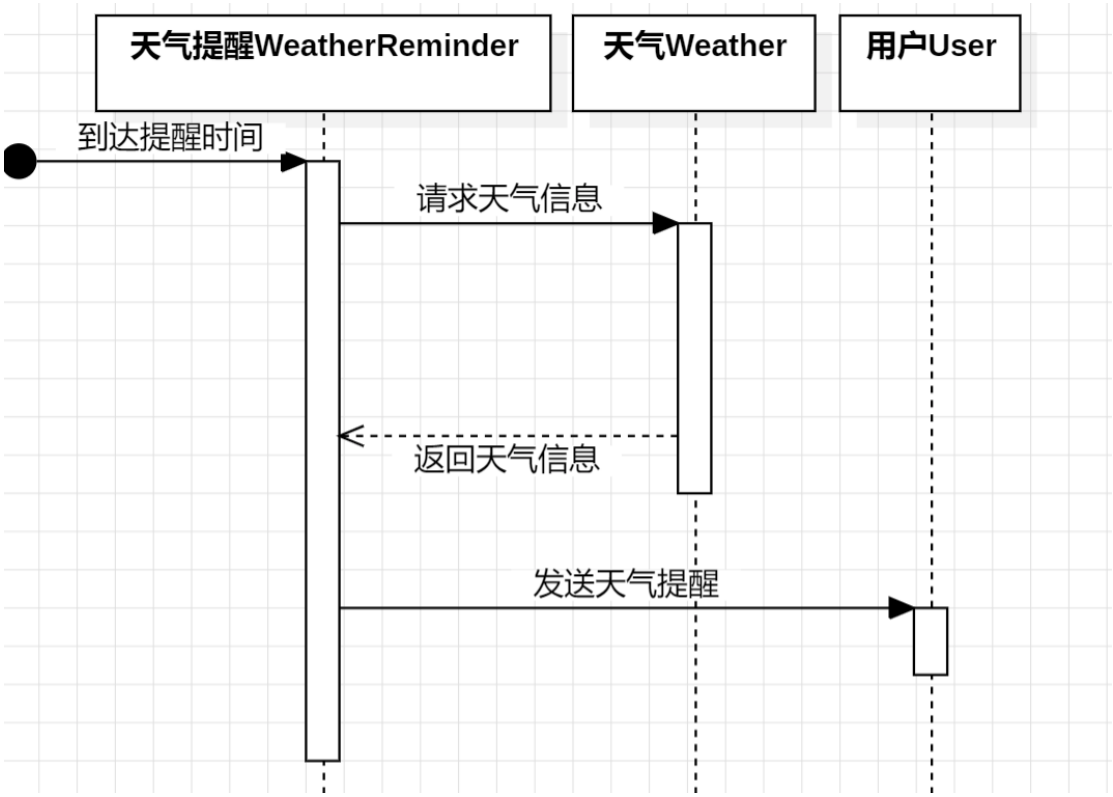
用户进入天气查询页面后，向天气类发送请求查看定位地点的当日天气，天气类请求地理位置，系统通过微信认证请求地理位置，获取用户当前的位置并返回。随后，天气类根据地理位置调用 API 请求当日的天气信息，最后将得到的天气情况显示给用户。



用户进入天气查询页面后，向天气类发送请求查看特定地点的当日天气，系统根据选择的特定位置和日期调用 API 请求天气信息，最后将得到的天气情况显示给用户。



当用户进入天气提醒设置页面后，首先需要请求发送提醒权限并获得返回结果。之后用户可以进行三个主要操作：添加天气提醒、修改天气提醒和删除天气提醒。每个操作执行后系统向天气提醒发送请求，返回相应的操作结果。



当到达预设的提醒时间时，天气提醒系统首先向天气系统请求天气信息，天气系统返回调用高德地图 API 得到的天气信息给天气提醒系统。最后，天气提醒系统将处理好的天气提醒信息发送给用户。

3 非功能需求

3.1 性能要求

3.1.1 精度

输入数据精度要求如下：

考虑到日常花销的商品价格，用户输入的预算金额需要精确到小数点后两位。例如预算金额为“50.00 元”

制定行程时选择的出发日期和到达日期，用户输入时应保证精确到天，且避免出现跨越日期的错误。时间字段（如交通工具的时间、景点的开放时间等）应当精确到分钟，避免误差导致行程冲突。

输出数据精度要求如下：

预算总额、各项支出的估算与输入相对应，应精确到小数点后两位。单项支出如餐饮、交通等也应精确到小数点后两位。

如果输出了每个景点的开始和结束时间，时间精度应保持在分钟级别。特别是涉及多个目的地和交通工具时，时间精度可以影响行程的合理性。

天气预报外部 API 获取并在前端输出展示的数据（例如温度、湿度、气压等信息），考虑到日常使用的需求，保留到整数即可。

3.1.2 时间特性要求

响应时间要求如下：

系统在用户登录或注册时，应该在 2 秒以内给出响应。确保用户能够快速进入系统，避免等待过长导致的流失。

用户查询目的地或景点信息时，系统应在 3 秒以内返回相关信息。这些数据可能涉及到外部 API 调用，因此需要保证网络延迟不会对用户体验产生负面影响。

用户在生成行程或进行预算计算时，系统应在 5 秒以内返回结果。对于涉及较多数据的计算，系统应优化性能，确保快速响应。

天气数据查询中，由于这些数据可能需要通过外部 API 获取，系统应保证查询响应时间在 5 秒以内。如果延迟过长，可以考虑添加数据缓存机制，以减少外部 API 调用次数。

其余功能最长响应时间不应超过 5 秒。

更新处理时间要求如下：

用户在修改行程时，系统应在 3 秒以内完成更新并返回操作结果，确保用户修改后的行程能够及时保存。

当用户在预算管理模块修改某项支出或预算金额时，系统应在 2 秒以内完成更新并展示新的预算数据。

用户记录新的费用支出时，系统应在 2 秒以内更新并显示当前的总费用。

其余更新时间最长不应长于 3 秒。

数据转换和界面更新传送时间要求如下：

当用户提交新的行程数据或预算数据时，系统应能够在 3 秒以内完成数据转换，并将计算结果反馈给用户。对于涉及复杂计算的功能（如预算分配、时间调度等），应进行性能优化，确保在短时间内完成数据处理。

系统界面在更新时，特别是涉及数据展示的部分，如预算统计、天气信息、行程安排等，更新时间应保持在 1 秒以内，确保页面响应流畅，用户可以实时看到信息更新。界面更新时间过长会导致卡顿和不流畅的体验，影响用户使用。

如果用户查看行程中的图像或景点照片，图像加载时间应控制在 2 秒以内。如果图像较大或用户网络环境较差，应该提供加载进度提示。

其余更新时间最长不应长于 3 秒。

数据同步和传输时间要求如下：

当系统从外部 API 获取信息时，数据同步的时间应控制在 3 秒以内。如果涉及到较为复杂的数据（如城市天气、交通状况等），系统应该提前缓存常用数据，减少频繁的请求次数，从而减少同步时间。

对于外部 API 的请求，网络延迟应尽可能低。理想情况下，请求的往返时间不应超过 2 秒，即使在网络状况不佳时，系统也应提供合理的加载提示或缓存数据，避免用户感到延迟过长。

后台处理时间要求如下：

在用户查询旅行推荐（如基于预算、季节、目的地等的个性化推荐）时，后台算法应在 5 秒以内完成数据处理并返回推荐结果。推荐算法的优化应确保快速响应，避免冗长的等待。

对于系统批量更新的数据（如定期更新天气、景点开放时间等），系统应保证后台更新的时间不超过 5 分钟，避免数据滞后影响用户决策。

系统容错与恢复时间要求：

当系统出现错误或异常时，恢复时间应不超过 10 秒。系统应具备合适的错误提示机制，并能快速重新加载或恢复操作。

在发生网络错误时，系统应该提供适当的重试机制，并在 5 秒内重试，确保数据能够顺利传输。

3.1.3 输入输出要求

输入数据类型如下：

(1) 文本输入（如目的地、用户姓名）

媒体格式：文本（字符串）

范围：目的地和姓名通常为 1-100 个字符的字符串；姓名不应含有特殊字符。

精度：输入文本无需特别精度要求，但应验证用户输入是否符合合法字符范围（如避免输入非法字符）。

示例：输入目的地为“巴黎”，输入姓名为“张三”。

(2) 日期/时间输入（如旅行日期、事件时间）

媒体格式：日期（yyyy-MM-dd），时间（HH:mm）

范围：日期范围应为用户的有效旅行时间，通常在未来 3 年内；时间应为 24 小时制。

精度：日期和时间需要准确到天和时分。

示例：旅行日期“2024-12-30”，事件时间“12:30”。

(3) 图片输入（如旅行图片）

媒体格式：图片（JPG、PNG、GIF）

范围：图片文件的大小一般不应超过 10MB，避免过大的文件占用过多存储空间。图片应为旅行相关内容的照片，如景点、旅行活动、风景等，用户可以上传一张或多张图片。图片的尺寸应符合合理显示要求，建议宽度不超过 3000px，高度不超过 3000px。

精度：图片内容无精度要求，但应确保图片内容清晰且能正常加载。可以对图片进行简单的大小和格式校验（如是否为有效的 JPG、PNG 格式）。

示例：用户上传一张旅游景点“埃菲尔铁塔”的照片，图片格式为“JPEG”，文件大小为 4MB，图片清晰可见。

(4) 预算/费用输入

媒体格式：数值（浮动点数）

范围：最小 0 元，最大金额取决于预算上限（如 10000 元）。

精度：精确到小数点后两位。

示例：输入预算金额为“5000.00 元”。

输出数据类型如下：

(5) 天气数据（如温度、湿度、天气状况）

媒体格式：JSON 格式

范围：温度值范围通常为-50℃到+50℃；湿度范围为 0%到 100%；天气状况为多个预设字符串（如“晴”“雨”）。

精度：温度需要精确到 1℃，湿度精确到 1%，天气状况为字符描述。

示例：温度为“22℃”，湿度为“75%”，天气为“晴”。

(6) 景点信息（如景点名称、开放时间）

媒体格式：JSON 或 XML 格式

范围：景点名称为非空字符串；开放时间为具体时间段（如“08:00-17:00”）。

精度：时间格式精确到小时和分钟。

示例：景点“巴黎埃菲尔铁塔”，开放时间“08:00-22:00”。

(7) 系统报告输出

1) 正常结果输出（如行程、预算计算结果）

媒体格式：文本报告（如 PDF、Word）、图形报告（如 SVG、PNG）

格式：PDF 文档或 HTML 格式

数值范围：行程详情中涉及的时间、预算金额等都应该在合法范围内（如时间为实际有效的旅行日期，预算金额为大于 0 的数字）。

精度：预算金额精度为小数点后两位，日期和时间精度到天和时分秒。

示例：输出行程报告，包含“巴黎 2024-12-30 08:00”，预算总额为“5000.00 元”。

2) 状态报告（如系统状态、数据同步状态）

媒体格式：文本或 JSON

格式：纯文本报告或 JSON 格式（例如{"status": "OK", "last_sync_time": "2024-12-29 14:00"}）

范围：状态报告通常包括“成功”、“失败”等状态信息。

精度：不涉及精度要求，主要关注数据完整性和时效性。

示例：系统状态报告：“系统正常运行”或“最后同步时间：2024-12-29 14:00”。

3) 异常报告（如系统错误、API 调用失败）

媒体格式：文本（可选 JSON 格式）

格式：纯文本报告或 JSON

范围：错误代码和错误描述必须明确，错误代码范围应符合预设的错误代码表。

精度：错误信息应尽可能详细，提供错误的具体信息（如“请求超时”或“数据格式错误”）。

示例：“系统错误：数据库连接失败”或“API 调用错误：请求超时”。

(8) 图形或显示报告输出

媒体格式：图像格式（如 PNG、JPEG），可嵌入 SVG 图形格式

范围：图像应该能够清晰显示所有关键数据，分辨率应足够高，尤其在生成地图或预算图表时。

精度：图表和地图的精度取决于数据的精度，如预算图表的条形图中每个条形应精确表示预算金额。

示例：生成预算分配图，显示不同项目的预算比例（如餐饮占比 30%，住宿占比 40%）。

3.2 数据管理能力要求

本系统需要管理多个数据类型，包括用户数据、行程数据、预算和费用数据以及外部获取的天气和景点信息。

随着用户量和使用频率的增加，数据存储需求将逐步上升。初期预计用户数量为 10,000 至 50,000 人，随着软件的推广，三年内可能增长到 200,000 至 500,000 人。每个用户的基本信息、行程安排、预算和费用等数据将存储在系统中，预计每个用户的数据大约为 1KB 至 50KB 不等，具体取决于数据类型和行程的复杂度。

以用户数据为例，每个用户的信息（如姓名、联系方式、行程历史等）占用的存储空间约为 1KB。随着用户数量的增加，用户数据总量预计在三年内增长至 500MB。

行程数据会随着每个用户创建多个行程而增加，每个行程包含多个事件。预计每个用户在未来三年内平均会有 10 个行程，每个行程包含 5 个事件，总存储需求可能增长到 50GB。

预算和费用记录也是系统需要管理的一个重要数据类别，预计每个用户的预算数据和费用记录总量大约为 30KB，整体存储需求在三年后可能达到 15GB。

此外，系统还会定期从外部 API 获取天气、景点等数据，每年的存储需求预计在 30MB 左右。

数据类型	初期存储需求	3 年后存储需求
用户数据	200MB	500MB
行程数据	10GB	50GB
预算和费用数据	6GB	15GB
外部数据	3MB	30MB
总计	16.2GB	65.5GB

综合来看，随着数据量的增加，系统的总存储需求将从初期的 16GB 增长至三年后的约 65GB。这一增长主要源自用户数量的增加以及行程、预算等数据的积累。

为了应对数据存储需求的增加，系统需具备良好的扩展性，支持分布式数据库存储（如 MySQL 或 PostgreSQL），并结合缓存技术（如 Redis）提高数据的读取效率。

同时，系统应设计合理的数据备份和恢复策略，确保数据安全，尤其在数据量增长时，备份频率和存储方式需要进行优化。通过数据压缩和归档技术，可以

有效节省存储空间并提高系统性能。因此，系统需要具备强大的数据存储和管理能力，以支持不断增长的数据量并确保长期稳定运行。

3.3 安全及保密性要求

本系统必须具备完善的安全机制，以确保用户数据和机密信息的保护，防止未经授权的访问、篡改和丢失。针对可能出现的各种安全威胁，本系统将从故障处理、数据恢复、安全防护等多个方面进行设计与优化。

故障处理能力和恢复要求

系统必须具备快速的故障检测与处理能力。应对系统故障、硬件故障、网络中断等问题时，系统应能够提供及时的告警机制，确保问题能够在最短时间内被发现并得到解决。为了最大程度减少用户影响，系统将采用高可用性架构，支持故障自动切换。若发生严重故障导致服务中断，系统必须能在规定的时间内（如5分钟内）恢复正常服务，并保证数据的一致性。

数据恢复能力要求

系统将定期进行数据备份，包括用户数据、行程数据、预算记录、费用数据等重要信息。备份将采用异地备份和云存储相结合的方式，以防止由于单一故障源造成数据丢失。对于意外丢失或损坏的数据，系统需支持灵活的数据恢复机制，确保能够在最短时间内恢复数据至最近的有效备份版本，最大程度避免数据丢失对用户的影响。

数据安全性要求

为了防止机密数据的非法访问、篡改或丢失，系统将采用多种安全技术保护数据安全，尤其是用户的行程信息，钱款相关信息等私人信息。首先，用户数据在传输过程中将使用加密协议（如TLS/SSL）进行加密，防止数据在传输过程中被窃取或篡改。其次，系统中的敏感数据（如用户密码、个人信息、钱款数目等）将采用加密存储，确保即便数据存储介质遭到攻击，数据本身也不会被泄露。用户密码将通过哈希算法（如bcrypt）进行存储，而不会以明文形式存储在数据库中。

访问控制和身份验证

为了防止未经授权的用户访问系统，系统将实行严格的身份验证机制。用户需通过用户名和密码登录，并支持基于角色的访问控制（RBAC），确保不同用户角色仅能访问与其权限范围相匹配的数据与功能。对于敏感操作（如修改账户信息、删除数据等），系统将采用二次验证措施（如短信验证、邮箱验证等）提高安全性。

防止恶意攻击

系统还需具备防御常见恶意攻击（如 SQL 注入、跨站脚本攻击 XSS、跨站请求伪造 CSRF 等）的能力。针对 SQL 注入攻击，系统将使用参数化查询和 ORM 框架，以避免恶意 SQL 代码注入。对于 XSS 和 CSRF 攻击，系统将采取输入数据过滤与验证、设置适当的 CORS 策略、使用 CSRF Token 等安全措施，防止恶意脚本执行和非法请求伪造。

安全审计与监控

本系统需要具备完善的安全审计机制。所有用户操作、系统异常事件等都会被记录并生成日志，以便后续进行安全审计和分析。日志中应详细记录用户操作内容、时间、来源 IP 等信息，防止潜在的安全漏洞或滥用行为被忽视。我们的日志文件将采用加密存储，并限制访问权限，以防止被篡改或泄露。系统还应支持安全事件的实时监控，并通过告警机制及时通知管理员以进行处置。

3.4 灵活性要求

操作方式的变化：随着用户需求的变化，小程序允许对操作方式进行灵活调整。例如，软件应支持用户自定义界面布局、快捷键等操作方式的配置，以适应不同用户群体的使用习惯。

同其他软件接口的变化：系统设计为模块化结构，支持灵活的接口管理。若有新功能或服务的加入，系统能够无缝集成新接口或第三方平台，且不会影响现有功能的正常运行。

精度和有效时限的变化：小程序可以允许根据不同应用场景调整相应对象的精度和有效时限要求。精度设置灵活，以支持不同业务场景下的需求，同时能够对实时性要求进行动态调整，确保能够快速响应并处理不同的数据量和任务类型。

3.5 其他专门要求

用户期望软件能够提供直观、易于操作的界面，特别是在复杂功能模块之间的切换和操作流程上，系统界面简洁明了，避免过多复杂的配置或设置。软件界面应符合用户习惯，并且支持个性化定制，用户能够根据个人需求调整界面布局和功能设置。

系统应设计为易于维护和扩展的架构，采用模块化、松耦合的设计方式，确保不同模块之间的依赖关系最小化。代码应具备良好的文档化，采用标准的编码

规范，以便后期开发人员能够快速理解和维护。日志记录应完整且清晰，以便于故障排查和系统监控。

系统应具备良好的扩展能力，以便未来根据业务需求添加新功能或模块。开发时要预留足够的接口和扩展点，确保后期不会因新需求的出现而导致大规模的重构。系统的插件式设计有助于简化新增功能的集成。

系统的用户界面设计应简洁明了，符合现代用户的审美和操作习惯。同时，系统的代码应具备良好的结构，使用清晰、易理解的变量和函数命名，注重代码注释，以便开发人员快速理解和二次开发。

系统应具有较高的可靠性，确保在正常操作和极端情况下都能稳定运行。具体来说，系统要处理好各种潜在的异常情况，能够在出现问题时提供有效的容错机制，并保证数据的完整性。系统应支持自动化监控，及时发现异常并发出警报。

系统应能有效应对各种异常情况，如网络中断、数据错误、操作失误等。每个异常场景都应有明确的错误提示，并提供有效的解决方案或操作指南，确保用户在遇到问题时能快速得到帮助或恢复正常工作。

4 运行环境规定

4.1 设备

a. 处理器型号及内存容量

为了保证小程序能够在大部分现代智能手机上顺畅运行，要求设备具备以下处理器和内存条件：

处理器：支持 ARM 架构的多核处理器，推荐使用至少为四核的处理器，主频不低于 1.8GHz。常见的处理器品牌如高通 Snapdragon 系列、华为 Kirin 系列、苹果 A 系列等。

内存容量：建议至少具备 4GB RAM 以上的内存。对于复杂功能（如天气预报、路线规划、旅游推荐等）和多个其他应用并行运行时，较大的内存可以提供更好的性能和响应速度。

b. 外存容量、联机或脱机、媒体及其存储格式

外存容量：要求设备拥有至少 64GB 的内存存储，支持足够存储用户的个人数据、旅行日程、照片等。应用本身的安装包和数据缓存应不超过 200MB，确保大部分设备能够顺利安装。

联机或脱机存储：应用的数据存储方式应支持在线和离线两种模式。在线模式下，用户的数据实时同步至云端；离线模式下，用户可以使用大部分功能（如查看已下载的旅行计划和日程）即使没有网络连接。离线存储格式应支持常见的压缩格式（如 JSON、SQLite 数据库等）以减小存储占用。

媒体存储：应用将支持用户上传照片，存储为标准的图像格式（如 JPEG、PNG），并在设备中进行本地缓存。

c. 输入及输出设备的型号和数量，联机或脱机

输入设备：主要通过触摸屏输入，包括手势操作、点击、滑动等。大部分智能手机都具备触摸屏，支持多点触控。

输出设备：主要是智能手机的显示屏，要求屏幕分辨率不低于 720p（1280x720），以保证地图、行程、图片等信息的清晰展示。智能手机还应具备音响或耳机插孔，以支持语音提示和播放功能。

联机或脱机操作：所有输入和输出设备均支持联机和脱机操作模式，在网络连接状态下实时更新数据，在离线状态下依然能够正常使用部分功能。

d. 数据通信设备的型号和数量

手机应支持 Wi-Fi 802.11 a/b/g/n/ac 标准，以保证在连接到 Wi-Fi 网络时，能够高速稳定地下载或上传数据。

要求设备支持 4G 或 5G 网络，以保证在没有 Wi-Fi 网络的情况下，仍能够顺畅使用数据传输功能。

用于连接外部设备（如蓝牙耳机、智能手表等）进行数据交换或音频播放，推荐支持蓝牙 4.0 及以上版本。

e. 功能键及其他专用硬件

功能键：智能手机的基本物理功能键（如电源键、音量键）能够支持应用的基本操作，虽然小程序不直接依赖物理按键，但可以通过这些按键调整音量或退出应用等。

对于定位相关功能，需要设备配备高精度的 GPS 模块，以确保用户的地理位置能够准确获取并用于行程规划、路线推荐等功能。此外，若应用包含增强现实（AR）功能，则需设备支持 AR 核心和相关硬件（如加速度传感器、陀螺仪等）。

4.2 支持软件

网络和硬件设备平台

支持互联网接入，通过稳定的网络连接进行数据传输和 API 接口调用。小程序通常通过 HTTP/HTTPS 协议与服务器进行通信，因此需要稳定的网络支持。

考虑到小程序是运行在微信平台上的，因此主要支持智能手机、平板等移动设备。主要设备平台包括：安卓操作系统版本 4.4 及以上的手机或平板。iOS 操作系统版本 9.0 及以上的 iPhone 和 iPad 设备。

操作系统平台

Android 4.4 及以上版本。此版本的设备通常能够支持微信小程序的运行。iOS 9.0 及以上版本，用于支持 iPhone 和 iPad 等设备的正常运行。对于不同版本的操作系统，可能需要进行适配优化，确保小程序在不同版本的设备上都能稳定运行。

数据库系统平台

MySQL：适用于存储和管理关系型数据，如用户信息、旅行行程、预算等。

MongoDB：适用于处理非关系型数据或大规模的、结构化的数据存储。

Redis：用于缓存处理和临时数据存储，如用户登录状态、热门目的地等。

编译和构建工具

微信开发者工具：用于开发和调试微信小程序。支持通过 JS、WXML、WXSS 等文件的编译、构建和调试，确保小程序代码的正确性。

Node.js：用于搭建后端开发环境，处理 API 请求，连接数据库，进行服务器端的业务逻辑处理。

npm/yarn：用于管理前端和后端的 JavaScript 依赖包，构建前端和后端所需的资源和工具。

测试支持软件

前端单元测试使用 Jest，确保前端代码的功能正确性。Postman 用于测试后端 API，验证接口的正确性和性能。使用 JUnit 单元测试框架，测试 Java 程序的各个功能模块

其他支持软件

使用 Git 用于版本控制，方便开发团队协作管理代码，追踪代码历史。

4.3 接口

该小程序将与多个外部系统和服务进行接口对接，以实现不同功能。首先，系统将通过天气接口获取实时天气信息，用于为用户提供旅行相关的天气预测，帮助用户做出更好的旅行计划。该接口使用标准的 HTTP 协议进行数据交换，数据格式为 JSON，能够返回当前天气、未来几天的天气预报等信息。

另外小程序还将与小红书平台接口进行对接，通过行程信息查找对应的旅行规划数据。基于小红书等平台的数据和用户输入的旅行目的地、预算等信息，系统将集成大语言模型智能搜索接口，提供个性化的旅行建议、景点推荐以及其他旅行相关的智能回答。大语言模型接口通常支持文本格式的请求和响应，通信协议为 RESTful API，通过 JSON 格式传递数据。这些接口的集成确保了小程序能够提供丰富的功能，如实时天气更新、智能推荐和语言交互，提升用户体验并增强系统的实用性。