旅途印济详细设计规约

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
2025-04-01		1.0	完成初版详细设计	刘文美 高妤婕 邵任飞	2025-04-01	杜庆峰 邵任飞	2025-04-01	
2025-04-02		1.1	补充模块	刘文美 高妤婕 邵任飞	2025-04-02	杜庆峰 邵任飞	2025-04-02	
2025-04-03		1.2	修改	刘文美 高妤婕 邵任飞	2025-04-04	杜庆峰 邵任飞	2025-04-04	

1 引言

1.1 编写目的

本文档用于细化旅途印济系统的详细设计，我们将通过微服务设计、接口设计、类详细设计三个角度规范旅途印济系统开发中的编码方向、不同微服务之间的接口调用以及方法设计等。

1.2 背景与依据

本项目名为旅途印济，旨在开发一款智能旅游软件，帮助用户高效规划行程、管理预算并获取实时天气信息。该软件通过整合多个功能，提供一站式解决方案。用户不仅可以灵活安排旅行计划，还能根据预算、季节和目的地等因素，获得个性化的智能行程推荐。同时，软件将帮助用户实时跟踪开销，并提供实时天气更新和提醒，确保旅行更顺畅、个性化和高效。

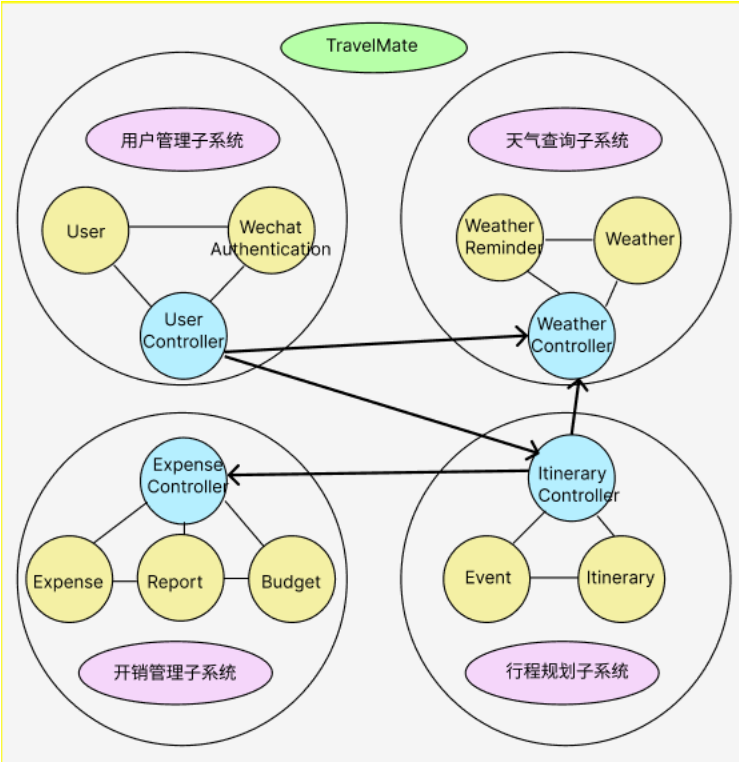
开发者团队为同济大学计算机科学与技术学院的三名本科生高妤婕、刘文美、邵任飞一起完成；项目预期用户是自由行爱好者，软件可以让用户在规划和执行旅行计划时更加高效、智能，确保他们的旅行更加顺利和愉快。

1.3 参考资料

1. 《重构：改善既有代码的设计》

2. 《设计模式：可复用面向对象软件的基础》
3. 《旅途印济-软件需求规约》
4. 《旅途印济-软件需求分析规约》
5. 《旅途印济-软件概要设计规约》

2 软件体系结构



共分为四个子系统，包括用户管理子系统、天气查询子系统、开销管理子系统和行程规划子系统。每个子系统都有控制器（Controller），它是子系统的入口。控制器负责处理请求，并与其他控制器协作，实现跨模块的功能。各子系统职责单一，数据流通过控制器完成，避免模块间的直接依赖，便于维护和扩展。

3 微服务设计

共有四个微服务：用户管理微服务、天气查询微服务、开销管理微服务和行程规划微服务。

3.1 功能

用户管理微服务负责用户身份验证与信息管理，是系统的核心组成部分。该服务支持用户通过小程序授权码登录，系统通过解析授权码获取用户唯一标识openID，并根据需要在数据库中创建新用户。用户可通过微服务查看个人信息，包括昵称、性别等详细数据，并支持修改个人信息。服务还提供小程序授权功能，如地理位置授权。该微服务以高效、安全的方式管理用户数据，为其他模块提供基础支持和数据共享能力。

天气查询微服务为用户提供实时天气信息查询和天气提醒设置功能。用户可以通过该服务获取当前及未来的天气状况，包括温度、风速等详细信息，为行程规划和活动安排提供支持。服务接入高德接口，确保数据准确可靠。同时，用户可设置天气提醒功能。

开销管理微服务负责用户的财务数据处理和分析，主要功能包括开销管理、预算管理和报告生成。用户可以通过该服务新建、删除和修改开销记录，确保开销数据的实时更新和准确性。同时，用户能够管理预算，设置预算上限，进行预算调整，并随时查看预算状态，避免超支。此外，服务还提供报告生成功能，用户可以基于开销和预算数据生成可视化的财务报告。通过这一微服务，用户能够轻松管理日常开销、控制预算，并获取直观的财务报告。

行程规划微服务主要包括行程管理、行程中的事件管理和行程智能推荐功能。用户可以通过该服务方便地创建、删除或修改行程安排，确保行程信息始终准确并及时更新。该服务还允许用户在行程中添加、修改或删除特定事件，如景点游览、餐饮安排等，以满足个性化需求。行程智能推荐功能结合用户的预算、时间等数据，推荐旅行路线和活动，帮助用户节省时间并优化旅行体验。

3.2 接口

（1）行程规划微服务向开销管理微服务请求某一事件的开销列表

请求方式：get

请求路径：/expense/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
```

```
            {
                "id": 2,
                "eveID": 1,
                "type": 3,
                "time": "2024-11-21T12:30:00",
                "money": 20.0,
                "name": "buy some tickets"
            },
            {
                "id": 3,
                "eveID": 1,
                "type": 1,
                "time": "2024-11-21T12:50:00",
                "money": 50.0,
                "name": "taking a taxi"
            }
        ]
}
```


（2）行程规划微服务向开销管理微服务请求某一事件的预算列表

请求方式：get

请求路径：/budget/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 2,
            "eveID": 1,
            "type": 1,
            "money": 100.0,
            "name": "Test Budget"
        },
        {
            "id": 3,
            "eveID": 1,
            "type": 2,
            "money": 40.0,
```

```
                "name": "breakfast"
            }
        ]
}
```


（3）行程规划微服务向开销管理微服务请求删除某一事件的所有开销记录

请求方式：delete

请求路径：/expense/deletebyeveid

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


（4）行程规划微服务向开销管理微服务请求删除某一事件的所有预算记录

请求方式：delete

请求路径：/budget/deletebyeveid

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


（5）开销管理微服务向行程规划微服务请求某一用户对应的事件 **id**

请求方式：get

请求路径：/event/getall

参数示例：userID=681295877

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        6,
        7
    ]
}
```

（6）开销管理微服务向行程规划微服务请求某些行程对应的事件 **id**

请求方式：get

请求路径：/event/getbyitiIDs

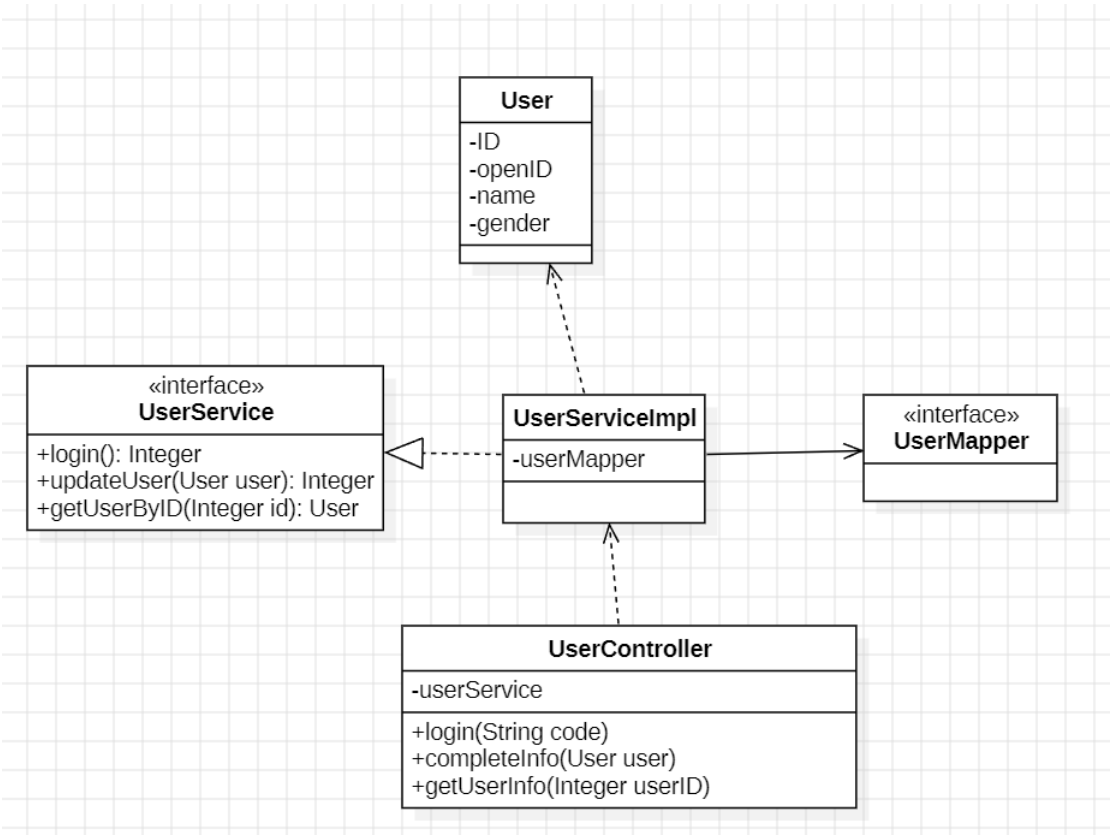
参数示例：itiIDs=1,2

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        6,
        7
    ]
}
```


4 类设计

4.1 用户管理子系统



4.1.1 User 实体类

属性名	类型	说明
ID	Integer	用户的唯一标识，主键，自增
openID	String	用户微信的唯一标识
name	String	用户的昵称
gender	Integer	用户的性别

4.1.2 UserMapper

UserMapper 用于实现对 User 实体对应的数据库表的操作。通过继承 MyBatis-Plus 提供的 BaseMapper 接口，UserMapper 无需手动编写基础的增删改查 SQL 方法，即可完成对 User 表的查询、插入、更新和删除等操作，极大地提高了开发效率。

4.1.3 UserServiceImpl 业务逻辑类

（1）Integer login(String code)

用户登录，主要实现通过微信 code 获取用户的 openID，并根据 openID 判断用户是否已注册，若未注册则插入新用户信息。

（2）Integer updateUser(User user)

根据用户的 ID 更新其信息，可选择性地更新 name 和 gender。

（3）User getUserByID(Integer id)

根据用户 ID 查询用户信息。

4.1.4 UserController 控制类

提供了三个前后端接口：

（1）**用户登录**

请求方式：GET

请求路径：/user/login

参数示例：code=0e1lM7Ga1enqMI0FE0Ha1T0FGv2lM7Gg

返回：

```
{
    "code": 1,
    "msg": "success",
```

```
    "data": 681295881
}
```


（2）完善用户信息

请求方式：PUT

请求路径：/user/info

参数示例：


```
{
    "id":681295877,
    "name":"hello",
    "gender":1
}
```


返回：


```
{
    "code": 1,
    "msg": "success",
    "data": 681295881
}
```


（3）得到用户信息

请求方式：GET

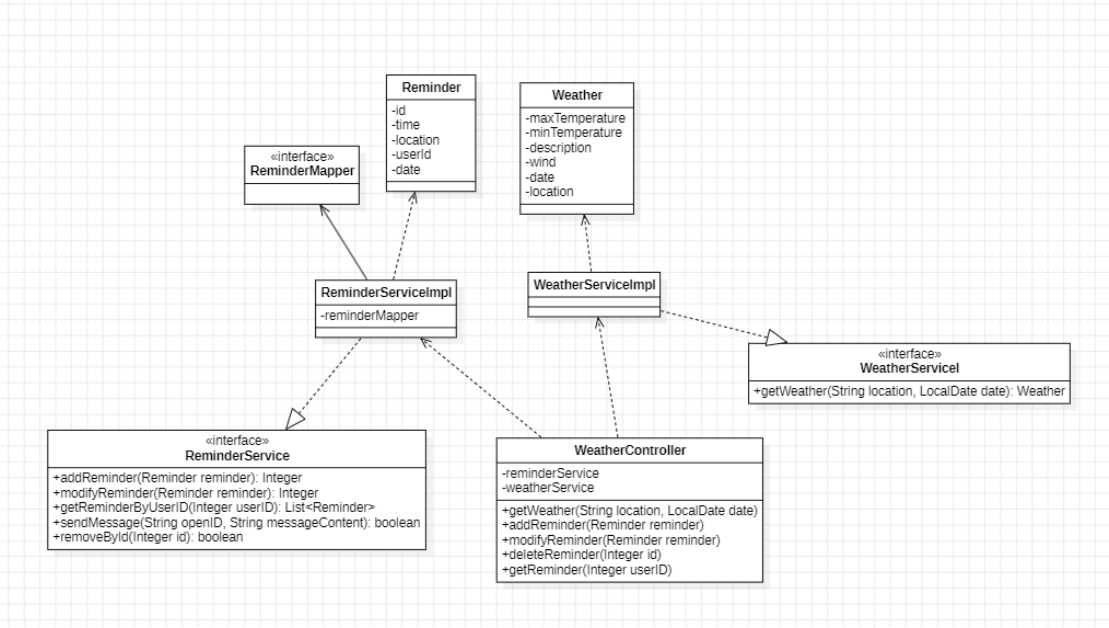
请求路径：/user/info

参数示例：userID=681295878

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "openID": "sadq1dwqeq1aqsgadwd",
        "name": "hahaha",
        "gender": null,
        "id": 681295882
    }
}
```

4.2 天气查询子系统



4.2.1 Reminder 实体类

属性名	类型	说明
id	Integer	主键，自增
time	LocalDateTime	提醒的时间
location	String	提醒的天气对应的地点
userId	Integer	关联的用户 ID
date	LocalDate	提醒的天气对应的日期

4.2.2 Weather 实体类

属性名	类型	说明
maxTemperature	float	最高温度
minTemperature	float	最低温度
description	String	天气描述
wind	String	风速
date	LocalDate	天气日期
location	String	天气地点

4.2.3 UserMapper

ReminderMapper 是一个用于操作 Reminder 实体类对应的数据库表的接口。它通过继承 MyBatis-Plus 提供的 BaseMapper 接口，自动获得对 Reminder 表的基本增删改查（CRUD）功能。这样，开发人员无需手动编写常见的 SQL 方法，可以直接使用 BaseMapper 中的方法来执行查询、插入、更新和删除操作，从而提高开发效率并减少冗余代码。

4.2.4 ReminderServiceImpl 业务逻辑类

（1）Integer addReminder(Reminder reminder)

用于将一个天气提醒对象插入到数据库中。

（2）Integer modifyReminder(Reminder reminder)

用于更新 Reminder 对象在数据库中的信息。

（3）List<Reminder> getReminderByUserID(Integer userID)

根据用户 ID 获取该用户的所有提醒记录。

（4）boolean removeById(Integer id)

通过提醒 ID 删除提醒。

（5）boolean sendMessage(String openID, String messageContent)

给微信用户发送天气提醒

4.2.5 WeatherServiceImpl 业务逻辑类

(1) Weather getWeather(String location, LocalDate date)

调用高德地图天气 API，根据指定地点和日期获取当天的天气信息，解析 API 返回的 JSON 数据，并返回包含天气详情的 Weather 对象。

4.2.6 WeatherController 控制类

提供了五个前后端接口：

（1）**查询天气**

请求方式：GET

请求路径：/weather/get

参数示例：location=保定&date=2024-12-31

返回：

{

```
    "code": 1,
    "msg": "success",
    "data": {
        "maxTemperature": 6.0,
        "minTemperature": -3.0,
        "description": "晴",
        "wind": "1-3",
        "date": "2024-12-31",
        "location": "保定"
    }
}
```


（2）添加提醒

请求方式：POST

请求路径：/reminder/add

参数示例：


```
{
  "time": "2025-01-03T10:00:00",
  "location": "上海",
  "userId": 681295877,
  "date": "2025-01-03"
}
```


返回：


```
{
    "code": 1,
    "msg": "success",
    "data": 1
}
```


（3）修改天气提醒

请求方式：PUT

请求路径：/reminder/modify

参数示例：


```
{
  "id":13,
  "time": "2025-01-02T10:00:00"
}
```


返回：


```
{
```

```
    "code": 1,
    "msg": "success",
    "data": 1
}
```


（4）删除天气提醒

请求方式：delete

请求路径：/reminder/delete

参数示例：id=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


（5）获取某用户天气提醒列表

接口说明：天气提醒列表

请求方式：get

请求路径：/reminder/get

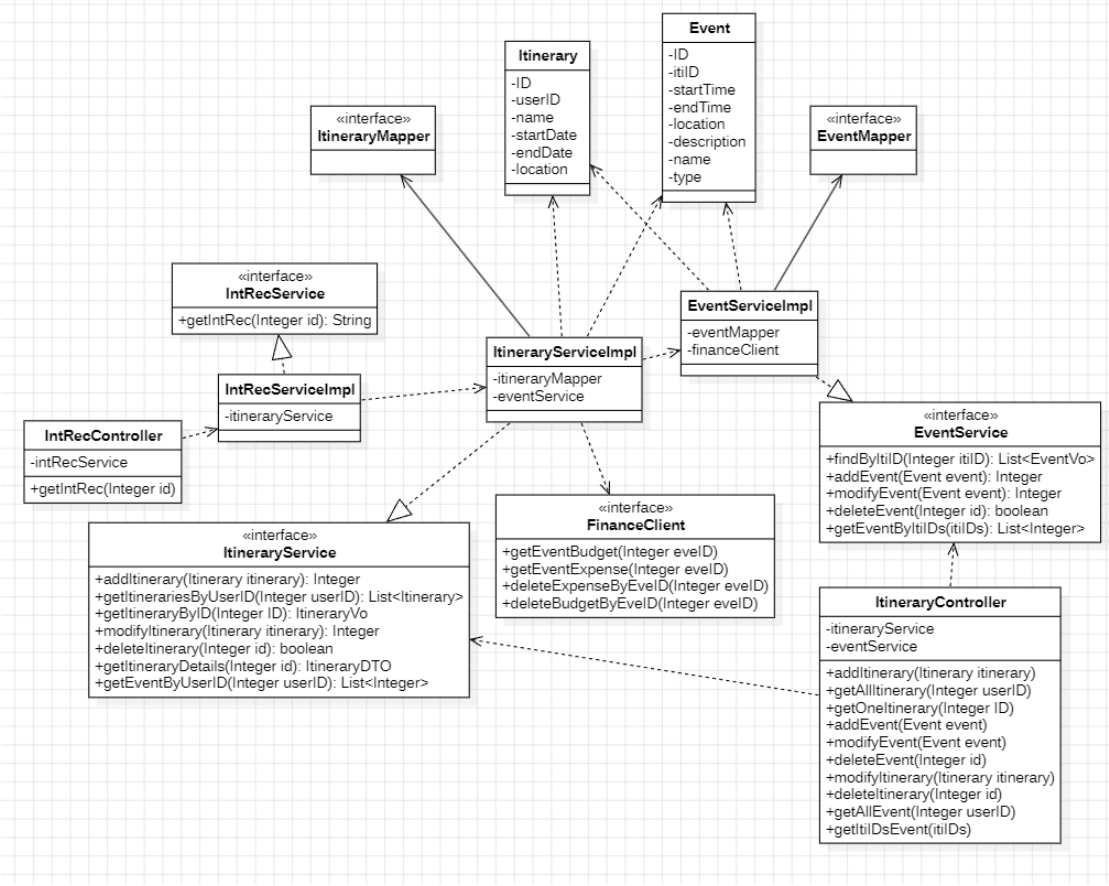
参数示例：userID=681295877

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 14,
            "time": "2025-01-03T10:00:00",
            "location": "上海",
            "userId": 681295877,
            "date": "2025-01-03"
        },
        {
            "id": 15,
            "time": "2025-01-03T10:00:00",
            "location": "济南",
            "userId": 681295877,
            "date": "2025-01-03"
        }
```

]
}

4.3 行程规划子系统



4.3.1 Itinerary 实体类

属性名	类型	说明
ID	Integer	主键，自增
userID	Integer	关联的用户 ID
name	String	行程名称
startDate	LocalDate	行程开始日期
endDate	LocalDate	行程结束日期
location	String	行程地点

4.3.2 Event 实体类

属性名	类型	说明
ID	Integer	主键，自增
itiID	Integer	关联的行程 ID
startTime	LocalDateTime	事件开始时间
endTime	LocalDateTime	事件结束时间
location	String	事件地点
description	String	事件描述
name	String	事件名称
type	Integer	事件类型

4.3.3 ItineraryMapper

ItineraryMapper 是一个用于操作 Itinerary 实体类对应的数据库表的接口。它同样继承了 MyBatis-Plus 提供的 BaseMapper 接口，自动获得对 Itinerary 表的基础增删改查（CRUD）功能，极大简化了数据库交互的实现过程。

4.3.4 EventMapper

EventMapper 是一个用于操作 Event 实体类对应的数据库表的接口。它通过继承 MyBatis-Plus 提供的 BaseMapper 接口，自动获得对 Event 表的基本增删改查（CRUD）功能。开发人员可以直接使用 BaseMapper 中提供的方法进行常规操作，无需手动编写 SQL，从而提高开发效率。

4.3.5 EventServiceImpl 业务逻辑类

（1）Integer addEvent(Event event)

插入一条新的事件记录到数据库中。

（2）List<EventVo> findByItiID(Integer itiID)

据行程 ID (itiID) 查询该行程下的所有事件，并将事件信息和其预算、支出数据封装为 EventVo 对象列表返回。

（3）Integer modifyEvent(Event event)

根据事件 ID 更新事件记录的部分或全部字段。

（4）boolean deleteEvent(Integer id)

删除指定事件 ID 的事件记录，并清理相关的预算和支出数据。

（5）List<Integer> getEventByItiIDs(List<Integer> itiIDs)

根据一组行程 ID 查询对应的事件 ID 列表。

4.3.6 ItineraryServiceImpl 业务逻辑类

（1）Integer addItinerary(Itinerary itinerary)

插入一条新的行程记录到数据库中。

（2）List<Itinerary> getItinerariesByUserID(Integer userID)

根据用户 ID (userID) 查询该用户的所有行程记录。

（3）ItineraryVo getItineraryByID(Integer ID)

根据行程 ID 查询行程详细信息及其所有关联的事件。

（4）Integer modifyItinerary(Itinerary itinerary)

根据行程 ID 更新行程记录的部分或全部字段。

（5）boolean deleteItinerary(Integer id)

删除指定行程及其所有关联的事件及事件中的开销和预算。

（6）ItineraryDTO getItineraryDetails(Integer id)

根据行程 ID 查询行程的基本详细信息。

（7）List<Integer> getEventByUserID(Integer userID)

根据用户 ID 查询其所有行程中关联的事件 ID 列表。

4.3.7 IntRecServiceImpl 业务逻辑类

（1）String getIntRec(Integer id)

根据指定行程的基本信息（如目的地、日期范围等），调用 AI 服务生成一份旅行推荐计划。

4.3.8 IntRecController 业务逻辑类

提供了一个前后端接口：

（1）行程智能推荐

请求方式：get

请求路径：/itinerary/getintrec

参数示例：id=1

返回：

```
{
    "code": 1,
    "msg": "success",
    "data": "#  旅游攻略\n\n## **Day 1:  哈尔滨....."
}
```


4.3.9 ItineraryController 业务逻辑类

提供了八个前后端接口：

（1）**新建行程**

请求方式：POST

请求路径：/itinerary/add

参数示例：


```
{
  "userID": 681295877,
  "name": "带我回北京",
  "startDate": "2024-02-21",
  "endDate": "2024-03-21",
  "location": "北京"
}
```


返回：


```
{
    "code": 1,
    "msg": "success",
    "data": 3
}
```


（2）**得到某用户的所有行程**

请求方式：GET

请求路径：/itinerary/getall

参数示例：userID=681295878

返回：


```
{
    "code": 1,
"msg": "success",
"data": [
        {
            "userID": 681295877,
```

```
                "name": "Trip to Paris",
                "startDate": "2024-11-21",
                "endDate": "2024-11-30",
                "location": "Paris",
                "id": 1
            },
            {
                "userID": 681295877,
                "name": "带我回北京",
                "startDate": "2024-02-21",
                "endDate": "2024-03-21",
                "location": "北京",
                "id": 3
            }
]
}
```


（3）得到某一个行程的信息

请求方式：GET

请求路径：/itinerary/getone

参数示例：ID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "userID": 681295881,
        "name": "巴黎三日游",
        "startDate": "2024-12-11",
        "endDate": "2024-12-13",
        "location": "巴黎",
        "events": [
            {
                "itiID": 1,
                "startTime": "2024-12-11T09:00:00",
                "endTime": "2024-12-11T18:00:00",
                "location": "巴黎教堂",
                "description": "Annual Conference",
                "name": "参观巴黎教堂",
```

```
                "type": 2,
                "expenses": [],
                "budgets": [
                    {
                        "id": 1,
                        "eveID": 1,
                        "money": 40.0
                    }
                ],
                "id": 1
            },
            {
                "itiID": 1,
                "startTime": "2024-12-11T19:00:00",
                "endTime": "2024-12-11T23:00:00",
                "location": "Little Red Door",
                "description": "放松一下，体验世界第五的酒吧",
                "name": "drink something",
                "type": 3,
                "expenses": [],
                "budgets": [],
                "id": 2
            }
        ],
        "id": 1
    }
}
```


（4）新建事件

请求方式：POST

请求路径：/event/add

参数示例：


```
{
   "itiID": 1,
   "startTime": "2024-11-22T09:00:00",
   "endTime": "2024-12-22T18:00:00",
   "location": "巴黎教堂",
   "description": "Annual Conference",
   "name": "参观巴黎教堂",
```

```
   "type": 2
}
```


返回：


```
{
      "code": 1,
      "msg": "success",
      "data": 3
}
```


（5）删除事件

请求方式：delete

请求路径：/event/delete

参数示例：id=1

返回：


```
{
      "code": 1,
      "msg": "success",
      "data": true
}
```


（6）修改事件

请求方式：PUT

请求路径：/event/modify

参数示例：


```
{
   "id": 4,
   "startTime": "2029-12-12T19:00:00",
   "endTime": "2029-12-12T23:00:00"
}
```


返回：


```
{
      "code": 1,
      "msg": "success",
      "data": 4
}
```


（7）修改行程

请求方式：PUT

请求路径：/itinerary/modify

参数示例：

{
"id": 3,
"name": "北京欢迎你"
}

返回：

{
"code": 1,
"msg": "success",
"data": 3
}

（8）删除行程

请求方式：delete

请求路径：/itinerary/delete

参数示例：id=1

返回：

{
"code": 1,
"msg": "success",
"data": true
}

提供了两个为开销管理子系统服务的接口：

（1）开销管理子系统向行程规划子系统请求某一用户对应的事件 id

请求方式：get

请求路径：/event/getall

参数示例：userID=681295877

返回：

{
"code": 1,
"msg": "success",
"data": [
6,
7
]
}

（2）开销管理子系统向行程规划子系统请求某些行程对应的事件 id

请求方式：get

请求路径：/event/getbyitiIDs

参数示例：itiIDs=1,2

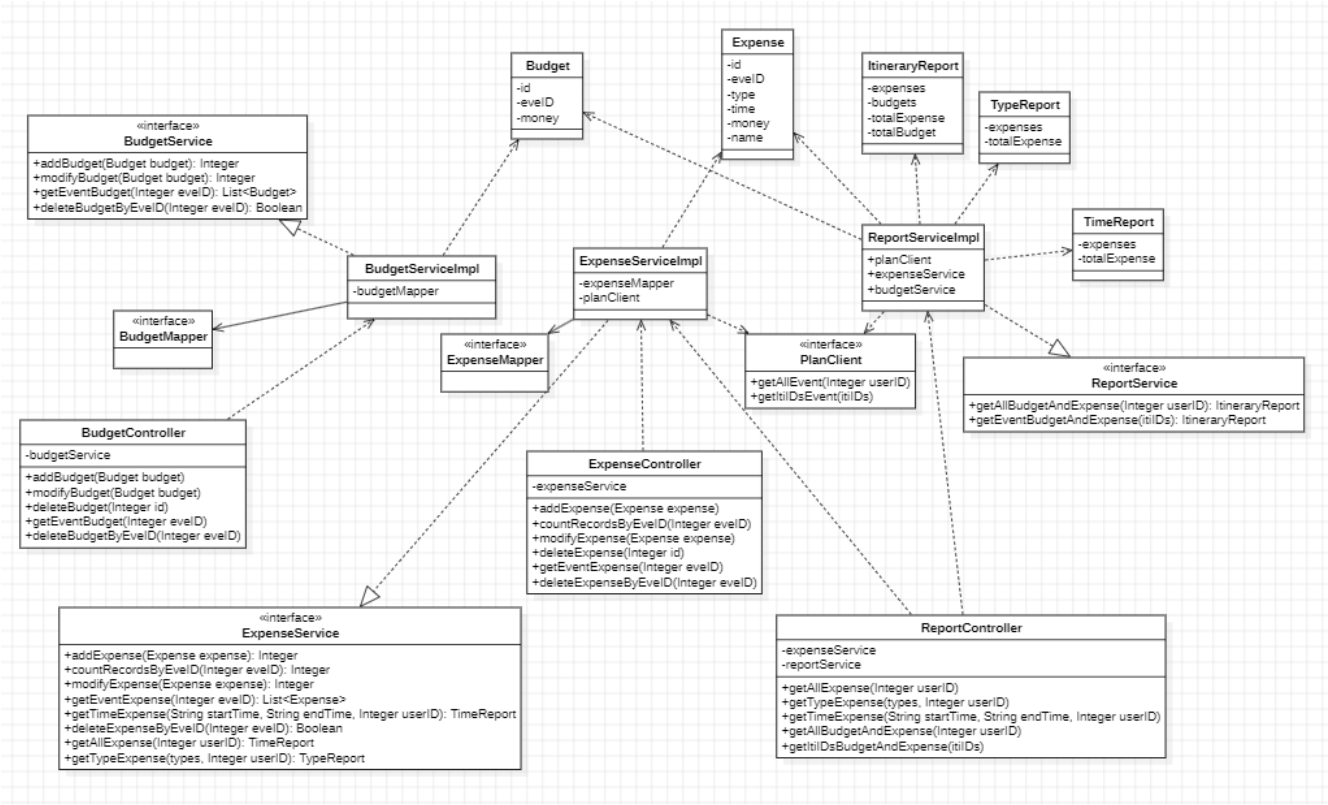
返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        6,
        7
    ]
}
```


4.3.10 FinanceClient

FinanceClient 是一个 Feign 客户端接口，用于与 tm-finance 微服务进行通信。通过该类，当前服务可以轻松调用 tm-finance 微服务中与预算和支出相关的 API，无需手动编写 HTTP 请求代码。

4.4 开销管理子系统



4.4.1 Budget 实体类

属性名	类型	说明
id	Integer	主键，自增
eveID	Integer	关联的事件 ID
money	Float	预算金额

4.4.2 Expense 实体类

属性名	类型	说明
id	Integer	主键，自增
eveID	Integer	关联的事件 ID
type	Integer	支出类型
time	LocalDateTime	支出时间
money	Float	支出金额
name	String	支出名称

4.4.3 ItineraryReport 实体类

属性名	类型	说明
expenses	List<Expense>	行程的所有支出记录
budgets	List<Budget>	行程的所有预算记录
totalExpense	Integer	行程总支出
totalBudget	Integer	行程总预算

4.4.4 TimeReport 实体类

属性名	类型	说明
expenses	List<Expense>	指定时间范围的支出记录
totalExpense	Integer	指定时间范围的总支出金额

4.4.5 TypeReport 实体类

属性名	类型	说明
expenses	List<Expense>	指定类型的支出记录
totalExpense	Integer	指定类型的总支出金额

4.4.6 BudgetMapper

BudgetMapper 是一个用于操作 Budget 实体类对应的数据库表的接口。它继承了 MyBatis-Plus 提供的 BaseMapper 接口，自动获得对 Budget 表的基础增删改查（CRUD）功能，从而简化了数据库交互的实现过程。

4.4.7 ExpenseMapper

ExpenseMapper 是一个用于操作 Expense 实体类对应的数据库表的接口。它继承了 MyBatis-Plus 提供的 BaseMapper 接口，自动获得对 Expense 表的基础增删改查（CRUD）功能。

4.4.8 BudgetServiceImpl 业务逻辑类

（1）Integer addBudget(Budget budget)

向数据库中添加一条预算记录。

（2）Integer modifyBudget(Budget budget)

根据预算的 ID 更新预算记录中的特定字段（如 money）。

（3）List<Budget> getEventBudget(Integer eveID)

根据 eveID 查询关联的所有预算记录。

（4）Boolean deleteBudgetByEveID(Integer eveID)

删除与指定 eveID 相关的所有预算记录

4.4.9 ExpenseServiceImpl 业务逻辑类

(1) Integer addExpense(Expense expense)

添加一条新的消费记录到数据库。

(2) Integer countRecordsByEveID(Integer eveID)

查询指定 eveID 对应的消费记录总数。

(3) Integer modifyExpense(Expense expense)

根据消费记录的 ID 更新其部分字段（如类型、时间、金额、名称）。

(4) List<Expense> getEventExpense(Integer eveID)

查询指定 eveID 关联的所有消费记录。

(5) TimeReport getTimeExpense(String startTime, String endTime, Integer userID)

根据时间范围和用户 ID 获取消费记录，并计算总金额。

(6) Boolean deleteExpenseByEveID(Integer eveID)

删除与指定 eveID 相关的所有消费记录。

(7) TimeReport getAllExpense(Integer userID)

查询指定用户 ID 关联的所有消费记录，并计算总金额。

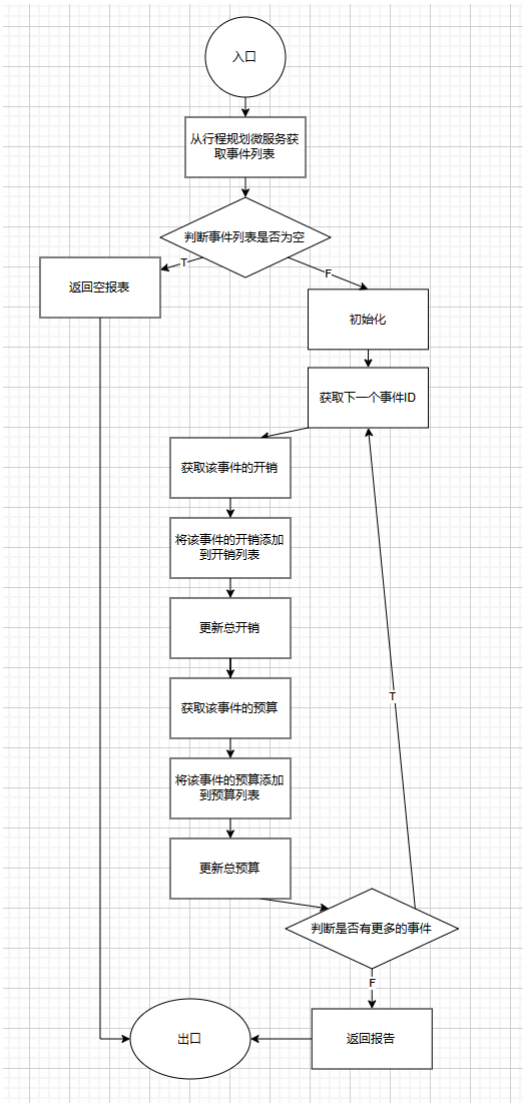
(8) TypeReport getTypeExpense(List<Integer>types, Integer userID)

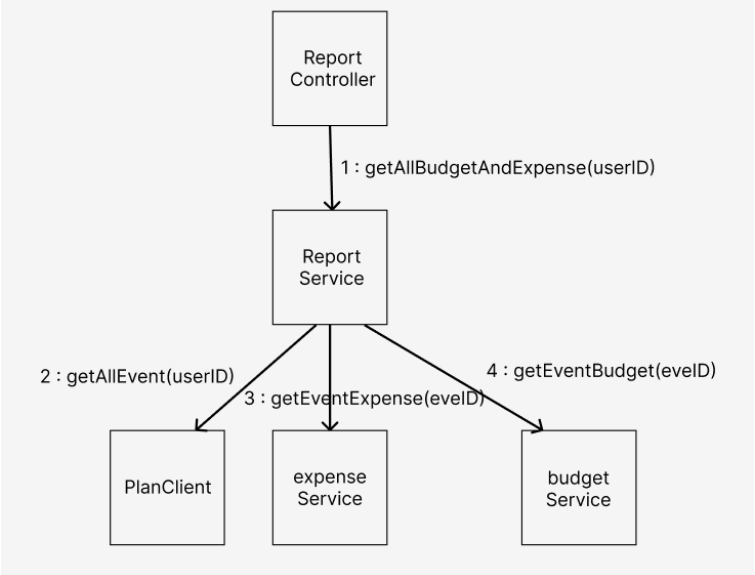
根据消费类型列表和用户 ID 查询消费记录，并计算总金额。

4.4.10 ReportServiceImpl 业务逻辑类

(1) ItineraryReport getAllBudgetAndExpense(Integer userID)

根据用户 userID 获取其关联的所有事件的预算和费用信息，计算总预算和总费用，并返回相关详细数据。





(2) ItineraryReport getEventBudgetAndExpense(List<Integer> itiIDs)

根据一组行程 ID (itiIDs) 获取其关联事件的预算和费用信息，计算总预算和总费用，并返回相关详细数据。

4.4.11 BudgetController 控制类

提供了四个前后端接口：

（1）**新建预算**

请求方式：POST

请求路径：/budget/add

参数示例：

```
{
  "eveID": 1,
  "money": 100.0,
}
```

返回：

```
{
    "code": 1,
    "msg": "success",
    "data": 1
}
```

（2）**修改预算**

请求方式：PUT

请求路径：/budget/modify

参数示例：

```
{
    "id":1,
    "money": 50
}
```

返回：

```
{
    "code": 1,
    "msg": "success",
    "data": 1
}
```

（3）**删除预算**

请求方式：delete

请求路径：/budget/delete

参数示例：id=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


（4）得到某一事件的预算列表

请求方式：get

请求路径：/budget/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 2,
            "eveID": 1,
            "money": 100.0
        }
    ]
}
```


提供了两个为行程规划子系统服务的接口：

（1）为行程规划子系统提供某一事件的预算列表

请求方式：get

请求路径：/budget/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 2,
            "eveID": 1,
            "type": 1,
            "money": 100.0,
```

```
            "name": "Test Budget"
        },
        {
            "id": 3,
            "eveID": 1,
            "type": 2,
            "money": 40.0,
            "name": "breakfast"
        }
    ]
}
```


（2）为行程规划子系统提供删除某一事件的所有预算记录

请求方式：delete

请求路径：/budget/deletebyeveid

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


4.4.12 ExpenseController 控制类

提供了四个前后端接口：

(1) **新建开销**

请求方式：POST

请求路径：/expense/add

参数示例：


```
{
    "eveID": 1,
    "type": 3,
    "time": "2024-11-21T12:30:00",
    "money": 20.2,
    "name": "buy some tickets"
}
```


返回：


```
{
```

```
    "code": 1,
    "msg": "success",
    "data": 2
}
```


(2) 修改开销

请求方式：PUT

请求路径：/expense/modify

参数示例：


```
{
    "id":3,
    "money": 50
}
```


返回：


```
{
    "code": 1,
    "msg": "success",
    "data": 3
}
```


(3) 删除开销

请求方式：delete

请求路径：/expense/delete

参数示例：id=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


(4) 得到某一事件的开销列表

请求方式：get

请求路径：/expense/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
```

```
            {
                "id": 2,
                "eveID": 1,
                "type": 3,
                "time": "2024-11-21T12:30:00",
                "money": 20.0,
                "name": "buy some tickets"
            },
            {
                "id": 3,
                "eveID": 1,
                "type": 1,
                "time": "2024-11-21T12:50:00",
                "money": 50.0,
                "name": "taking a taxi"
            }
        ]
}
```


提供了两个为行程规划子系统服务的接口：

(1) **为行程规划子系统提供某一事件的开销列表**

请求方式：get

请求路径：/expense/event

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": [
        {
            "id": 2,
            "eveID": 1,
            "type": 3,
            "time": "2024-11-21T12:30:00",
            "money": 20.0,
            "name": "buy some tickets"
        },
        {
            "id": 3,
            "eveID": 1,
```

```
                "type": 1,
                "time": "2024-11-21T12:50:00",
                "money": 50.0,
                "name": "taking a taxi"
            }
        ]
    }
```


（2）为行程规划子系统提供删除某一事件的所有开销记录

请求方式：delete

请求路径：/expense/deletebyeveid

参数示例：eveID=1

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": true
}
```


4.4.13 ReportController 控制类

提供了四个前后端接口：

（1）得到某一时间段内的开销列表

请求方式：get

请求路径：/report/time

参数示例：

startTime=2024-11-23T00:00:00&endTime=2024-11-29T23:59:59&userID=681295877

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "totalExpense": 9063，
        "expenses": [
            {
                "id": 8,
                "eveID": 6,
                "type": 2,
```

```
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 9,
                "eveID": 8,
                "type": 2,
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 10,
                "eveID": 8,
                "type": 2,
                "time": "2024-11-25T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            }
        ]
    }
}
```


(2) 得到用户的全部开销列表

请求方式：get

请求路径：/report/getall

参数示例：userID=681295877

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "totalExpense": 12084,
        "expenses": [
            {
                "id": 6,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
```

```
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 7,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 8,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 10,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-25T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            }
        ]
    }
}
```


（3）得到某几种类型的开销列表

请求方式：get

请求路径：/report/type

参数示例：userID=681295877&types=1,2,3

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
```

```
        "totalExpense": 12084,
        "expenses": [
            {
                "id": 6,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 7,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 8,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 10,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-25T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            }
        ]
    }
}
```


（4）展示全部行程预算和开销数据

请求方式：get

请求路径：/report/budgetandexpense

参数示例：userID=681295877

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "totalExpense": 12084,
        "totalBudget": 6000,
        "expenses": [
            {
                "id": 6,
                "eveID": 6,
                "type": 3,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 7,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 8,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 10,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-25T12:50:00",
```

```
                "money": 3021.0,
                "name": "buy food"
            }
        ],
        "budgets": [
            {
                "id": 5,
                "eveID": 6,
                "money": 2000.0
            },
            {
                "id": 6,
                "eveID": 7,
                "money": 4000.0
            }
        ]
    }
}
```


（5）得到部分行程预算和开销数据

请求方式：get

请求路径：/report/iti

参数示例：itiIDs=1,2

返回：


```
{
    "code": 1,
    "msg": "success",
    "data": {
        "totalExpense": 12084,
        "totalBudget": 6000,
        "expenses": [
            {
                "id": 6,
                "eveID": 6,
                "type": 3,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
```

```
                "id": 7,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-21T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 8,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-23T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            },
            {
                "id": 10,
                "eveID": 6,
                "type": 2,
                "time": "2024-11-25T12:50:00",
                "money": 3021.0,
                "name": "buy food"
            }
        ],
        "budgets": [
            {
                "id": 5,
                "eveID": 6,
                "money": 2000.0
            },
            {
                "id": 6,
                "eveID": 7,
                "money": 4000.0
            }
        ]
    }
}
```

4.4.14 PlanClient

PlanClient 是一个 Feign 客户端接口，用于与 tm-plan 微服务进行通信。通过该类，当前服务可以方便地调用 tm-plan 微服务中与事件管理相关的 API，无需手动编写 HTTP 请求代码。