**专业方向综合项目 项目需求分析规约**

**张尧 陈东 姜其升**

**TEam 12**

**MEETING！**

基于微信小程序的协同日程安排应用

**目录**

[1. 引言 2](#_Toc44760796)

[1.1. 背景 2](#_Toc44760797)

[1.2. 用户的特点 2](#_Toc44760798)

[1.3. 参考资料 3](#_Toc44760799)

[1.4. 假定和约束 3](#_Toc44760800)

[2. 功能需求 4](#_Toc44760801)

[2.1. 系统范围 4](#_Toc44760802)

[2.2. 需求分析 4](#_Toc44760803)

[2.2.1. 功能建模 4](#_Toc44760804)

[2.2.2. 数据建模 17](#_Toc44760805)

[2.2.3. 行为建模 24](#_Toc44760806)

[2.2.4. 内容建模 26](#_Toc44760807)

[3. 非功能需求 28](#_Toc44760808)

[3.1. 性能要求 28](#_Toc44760809)

[3.1.1. 响应时间 28](#_Toc44760810)

[3.1.2. 业务量 28](#_Toc44760811)

[3.1.3. 系统容量 28](#_Toc44760812)

[3.1.4. 精度 28](#_Toc44760813)

[3.1.5. 资源使用率 28](#_Toc44760814)

[3.2. 数据管理能力要求 28](#_Toc44760815)

[3.3. 安全及保密性要求 29](#_Toc44760816)

[3.3.1. 权限访问控制 29](#_Toc44760817)

[3.3.2. 日志控制 29](#_Toc44760818)

[3.3.3. 恶意请求防御 29](#_Toc44760819)

[3.4. 兼容性需求 29](#_Toc44760820)

[3.4.1. 系统与其他软件兼容性 29](#_Toc44760821)

[3.4.2. 与操作系统交互 29](#_Toc44760822)

[3.4.3. 可移植性 30](#_Toc44760823)

[3.5. 其他专门要求 30](#_Toc44760824)

[3.5.1. 可靠性需求 30](#_Toc44760825)

[3.5.2. 易用性 30](#_Toc44760826)

[4. 运行环境规定 31](#_Toc44760827)

[4.1. 设备 31](#_Toc44760828)

[4.2. 支持软件 31](#_Toc44760829)

[4.3. 接口 31](#_Toc44760830)

[4.3.1. 用户接口 31](#_Toc44760831)

[4.3.2. 软件接口 32](#_Toc44760832)

[4.3.3. 硬件接口 32](#_Toc44760833)

[4.3.4. 通信接口 32](#_Toc44760834)

[5. 需求跟踪 33](#_Toc44760835)

# 引言

## 背景

当今的大学教育都愈发提倡素质教育，鼓励团队协作。团队协作完成项目可以培养青年团队精神，增进队友之间感情，提高工作效率……因此，越来越多的大学课程要求学生进行团队合作完成项目。有时，甚至一个人会有数十个团队项目需要完成。在每一个团队小组中，小组成员开会讨论必不可少。一个学生如何在众多小组中权衡时间？这是现在还未能完美解决的一个问题。

我们本次将要开发的Meeting系统（以下称本系统）就主要为了解决当今小组开会时的痛点：当一个小组中的所有成员需要敲定一个开会时间时，会进行大量费时费力的讨论，尤其是在小组成员数量多于5人时。在聊天软件中进行开会时间讨论的现状是：每个人主动地提出自己的空闲时间，由其他组员进行比对，只要有一个组员时间有冲突，则又要重复这个过程，效率十分低下。而本系统将从本质出发，提供一个日程管理的功能，使小组成员间能够共享显示每个人的日程安排，并根据所有小组成员的空闲时间自动推荐开会时间，实现快速的开会或其他活动时间确定。

本系统由同济大学软件学院2017级专业方向综合项目第12组成员进行分析、设计开发。小组四人将从软件工程过程架构入手，以软件工程思想讨论、规划、建模、开发、部署，完成本系统构建。

本系统拟使用微信小程序作为前端平台，采用SpringBoot框架作为后端服务架构。在数据库方面，我们拟使用Redis存储用户有关信息，进行鉴权相关操作，使用MySQL进行其他基本用户数据的存储。

## 用户的特点

本系统的使用用户可能包括：

1. 需要进行大量组队项目工作的大学生群体
2. 需要商定集体活动时间的群组成员
3. 需要进行简单日程管理的所有人

综上，使用本系统的用户特征归纳为：

需要一个系统来进行自己短期内（7日）的日程管理。除此之外，在一些场景下，需要进行多人参与的小组活动。这些活动需要确认所有或大部分小组成员的空闲时间信息才能敲定时间，而本身的讨论又十分费时费力。

经过项目开发人员的科学估计，在上线后，本系统预期使用人数约为500人，高峰时期使用人数约1000人，用户并发请求不超过100个。

开发人员都具备良好的职业素养及较高的技术水平，能够胜任开发本系统的工作，并能够在项目规定时间内完成系统开发、测试并撰写详细文档。

## 参考资料

[1]Rogers S.pressman 软件工程：实践者的研究方法[M].北京：机械工业出版社，2011.4：106-151

## 假定和约束

本系统的开发完全基于PC，并要求在50天内完成整个系统的开发，包括讨论、计划、建模、构建、部署等过程，并撰写有关文档。本系统的开发过程将完全由小组成员3人完成，且无经费、无相关支持。开发环境为3台个人计算机，阿里云Ubuntu云服务器一台。

# 功能需求

## 系统范围

本系统将在提供日程管理的功能基础上，提供小组事件的管理。以小组中所有成员的日程信息为基本数据，为组长快速确定小组事件时间提供便捷。

主要的功能有：

1. 个人事件的添加与管理
2. 工作组的创建、退出
3. 通过微信群的邀请自动加入同一个工作组
4. 查看工作组中他人日程
5. 根据工作组成员空闲时间及事件优先级自动推荐公共事件时间
6. 勿扰模式

不同于其他日程管理软件，本系统在个人日程管理的基础上，创新地引入了工作组的概念，每个用户可以作为组长创建工作组并通过分享到微信群，邀请微信群中的小组成员，也可以作为成员加入到多个工作组中。

在工作组中，每个成员可以查看小组其他成员的日程信息，当然为了隐私考虑，可以在每个工作组中设置他人是否可以查看事件的详细描述。

除此之外，我们为每个日程增添了优先级，并辅以不同的颜色区分显示。

本系统最核心也是最具特色的功能是：在工作组中快速创建公共事件。当组长要创建一个公共事件时，他将选择公共事件的持续时间及优先级，系统则将自动地筛选所有小组成员的空闲时间进行时间推荐。如果没有合适的空闲时间，系统也将通过覆盖掉低优先级的事件进行公共事件时间确定。通过这样的每个人主动设置-被动反馈的方式，系统实现了公共事件时间快速设定，为工作增加了效率。

本系统将业务范围集中在一周内的个人日程及小组事件管理上，尽最大可能将功能集中并考虑详尽，主要提供的核心功能为工作组中共享事件的自动推荐和创建、管理，辅以小组成员及权限的管理、个人事件的管理等功能。而将不会涉及到超过一周的事件管理、万年历查看功能等与核心功能无关的功能。

本系统的最终目标是提供一个能够便捷地通过微信小程序和微信群的良好联动生态，提供给用户管理个人短期内日程、便捷创建工作组并快速约定集体事件时间的应用，增加小组合作或活动策划的效率。

## 需求分析

### 功能建模

在需求工程第一阶段的讨论中，我们已经根据需求的讨论与取舍构建出了用例图（如下图所示），并对每一个用例进行了详细分析与说明（见文档1-需求说明书）。

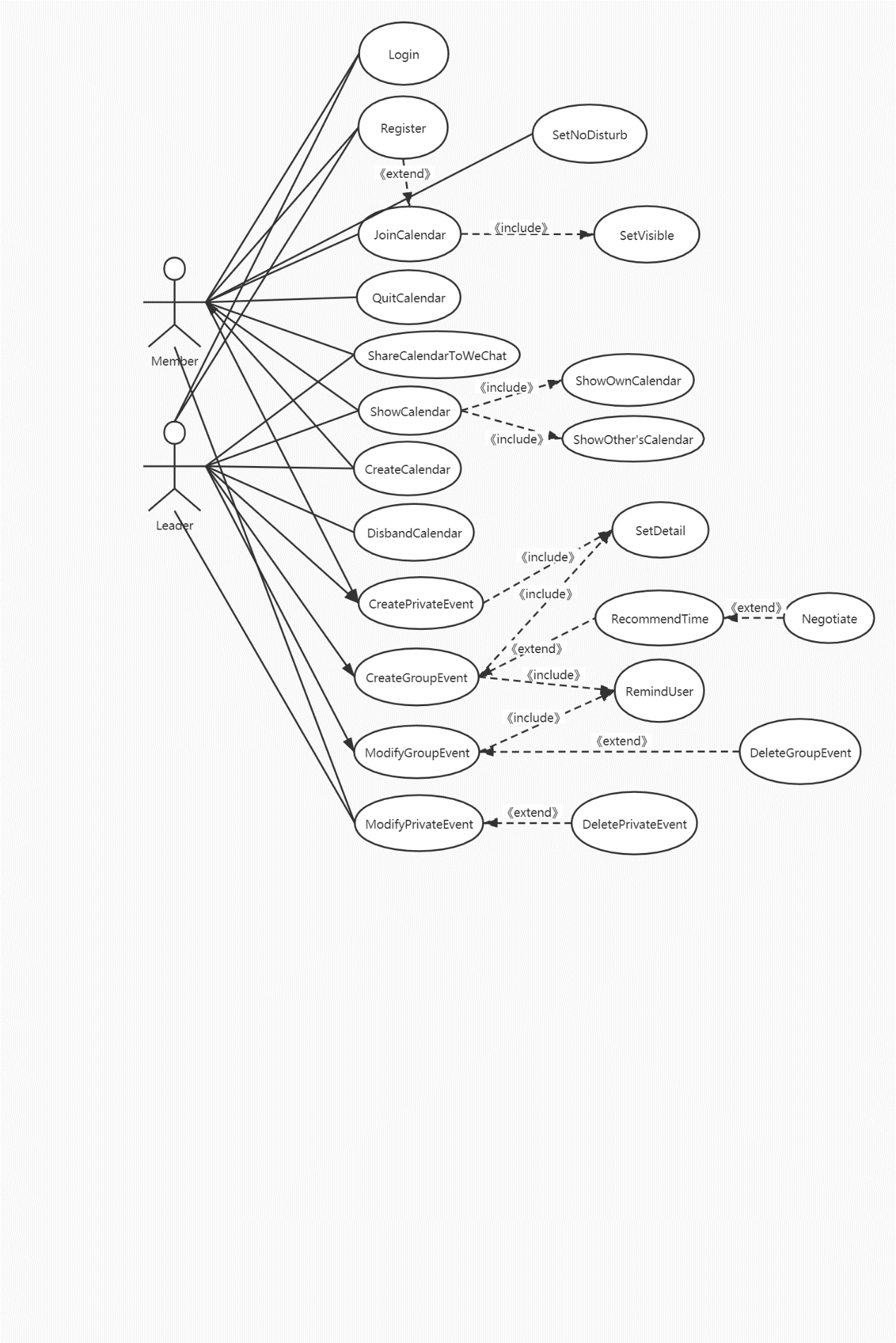


图2.1.系统的用例图

根据每一个详细用例说明，读者已经可以了解每一个用例的详细信息及具体需求。但纯文字和表格的内容还不足以简明扼要地传递信息。为此，我们在需求分析阶段使用UML活动图和时序图来描述每一个重要的用例。

1. **创建个人事件**

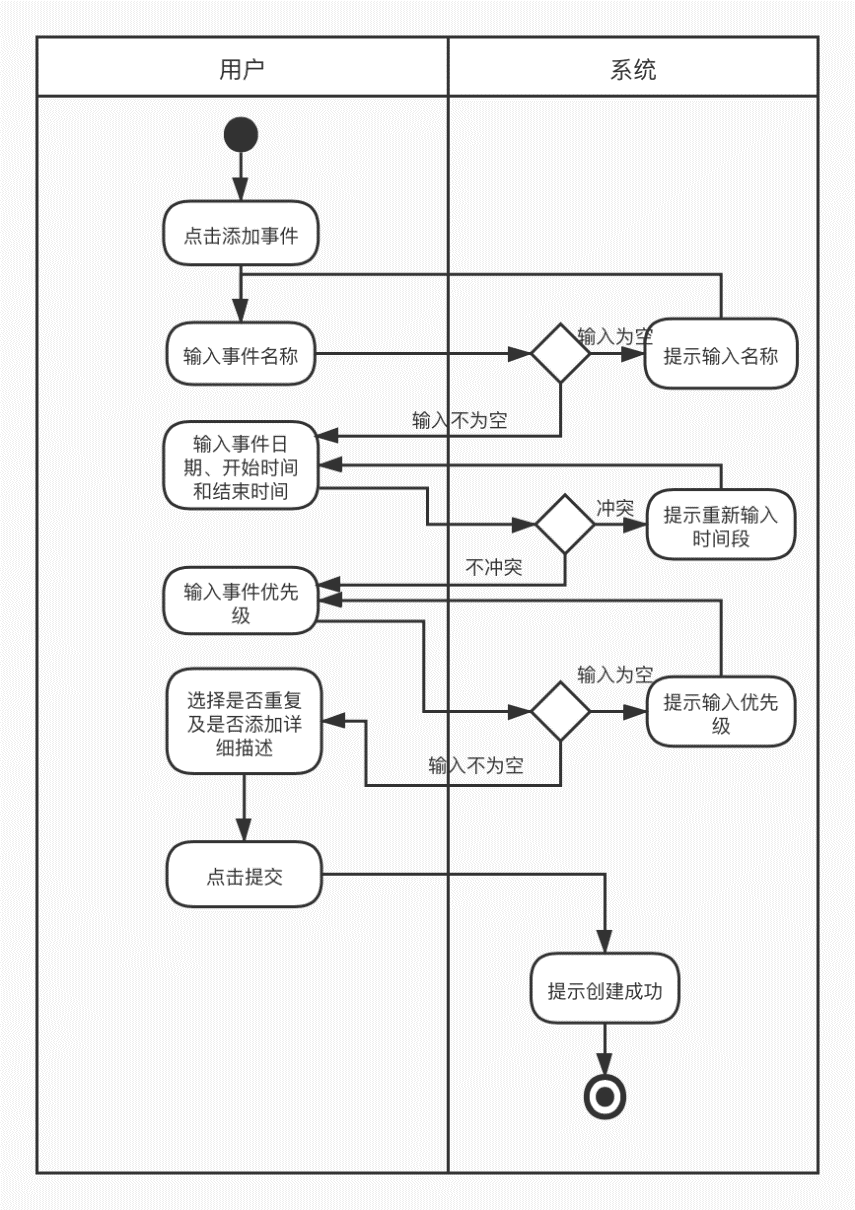


图2.2.创建个人事件用例的泳道图

可以从创建个人事件用例的泳道图清晰地看出该用例的逻辑流程。用户点击添加事件按钮后，需要首先输入事件名称，此时会有系统判断若输入为空则提示重新输入；接下来输入事件的日期、开始时间和结束时间，此时也会有系统判断若输入的时间与已有事件冲突，则提示重新输入时间段；接下来输入事件的优先级，系统也会判断是否为空并提示；最后用户选择是否重复及是否添加详细描述，再点击提交按钮，在网络良好时系统将会提示创建成功，创建个人事件完成。

1. **修改个人事件**

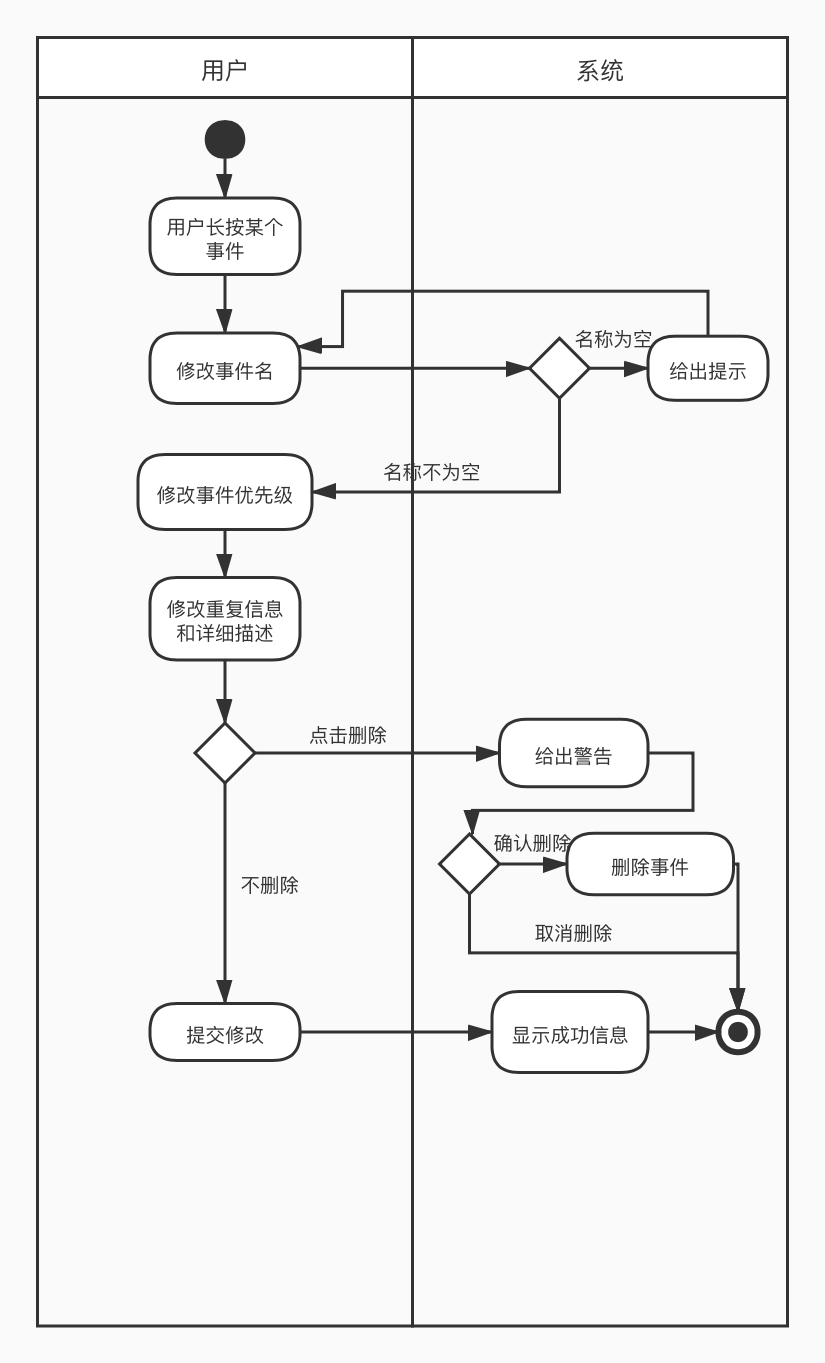


图2.3.修改个人事件用例的泳道图

可以从修改个人事件用例的泳道图清晰地看出该用例的逻辑流程。用户长按事件卡片后，需要首先输入修改后的事件名称，此时会有系统判断若输入为空则提示重新输入；接下来输入修改后的事件优先级，修改后的重复信息和事件详细描述。此时用户可以选择是否删除该事件，若不删除，则点击提交修改按钮，在网络良好时系统将会提示修改成功，修改个人事件完成。若选择删除，则系统会给出警告提示，用户依然选择删除后，事件将会被成功删除，修改个人事件结束。

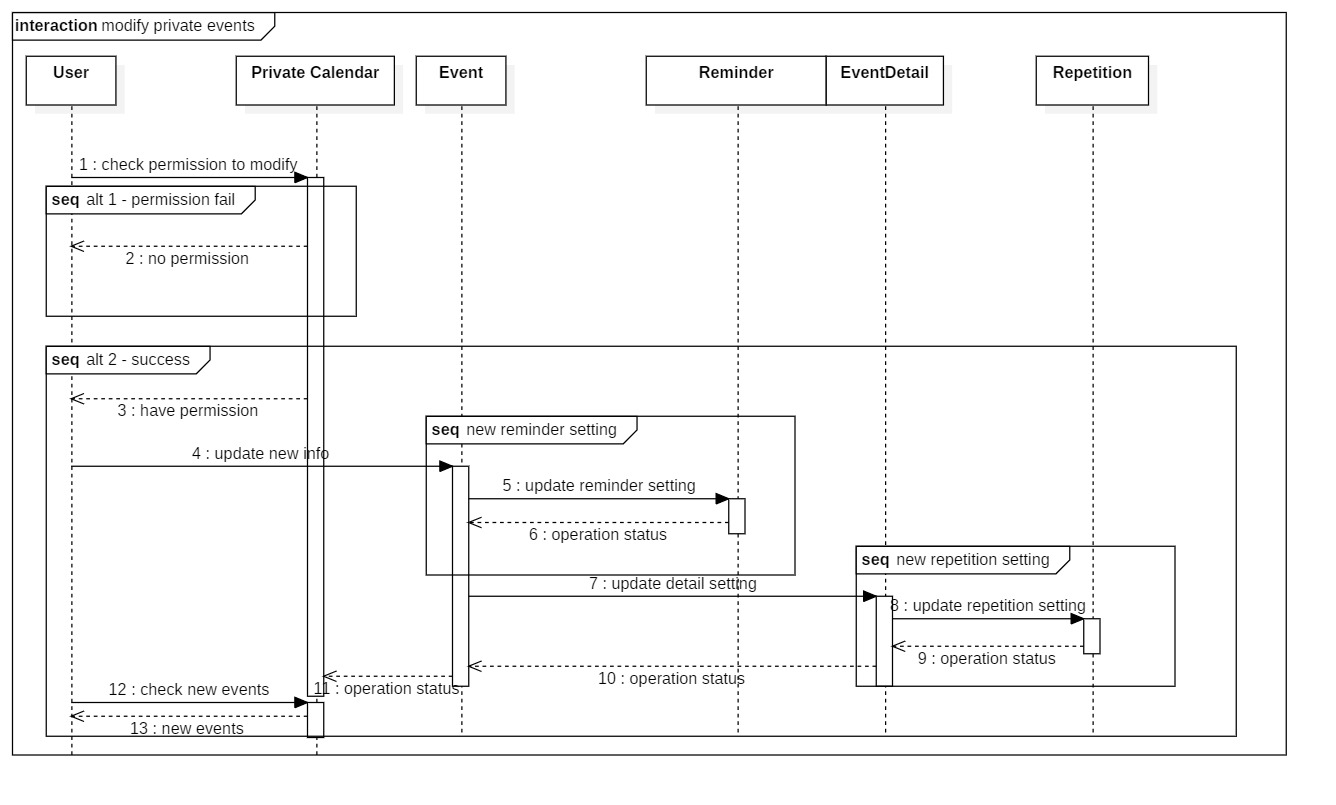


图2.4.修改个人事件用例的时序图

1. 创建团队事件

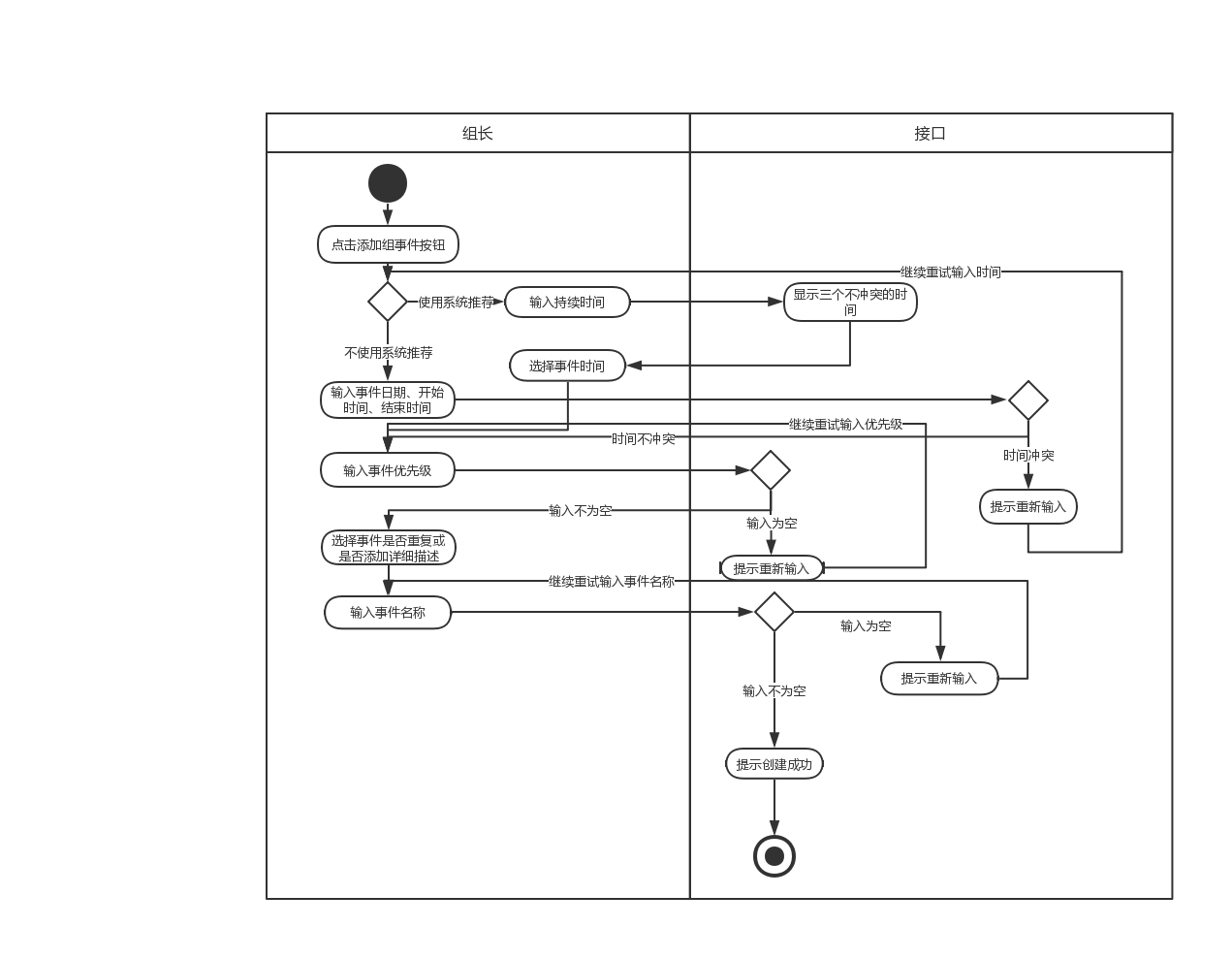


图2.5.创建团队事件用例的泳道图

可以从创建团队事件用例的泳道图清晰地看出该用例的逻辑流程。组长点击添加组事件按钮后，需要首先输入事件的日期、开始时间和结束时间。此时组长可以选择手动输入时间或使用系统推荐时间。若选择手动输入，则在输入后会有系统判断该时间是否与组员的已有事件时间冲突，若有冲突，则提示重新输入，若没有冲突，则可进行下一项输入。而组长选择系统推荐时间后，系统将使用算法计算出三个不会冲突的时间，需要组长选择一个时间填入，进入下一项输入。接下来输入事件优先级，此时会有系统判断若输入为空则提示重新输入；接下来输入事件重复信息和详细描述以及事件名称，系统也会判断事件名称是否为空，若不为空，且网络良好，则显示创建团队事件成功。

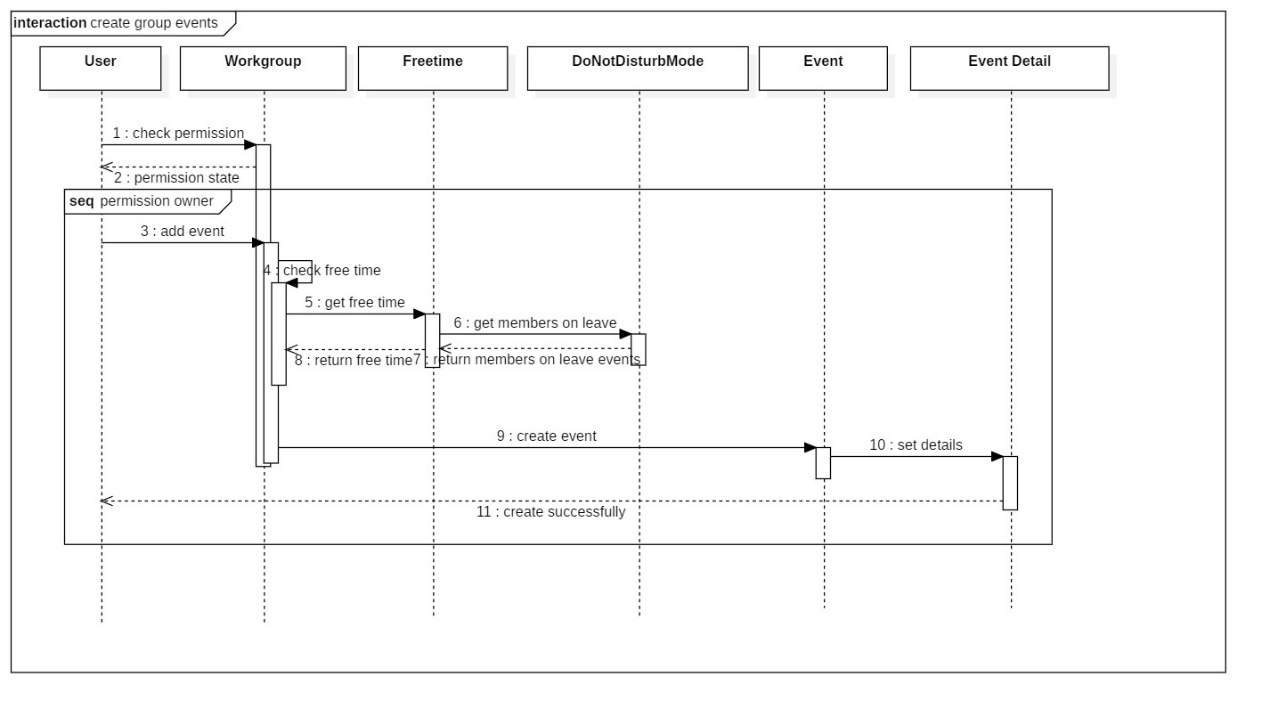


图2.6.创建团队事件用例的时序图

1. 修改团队事件

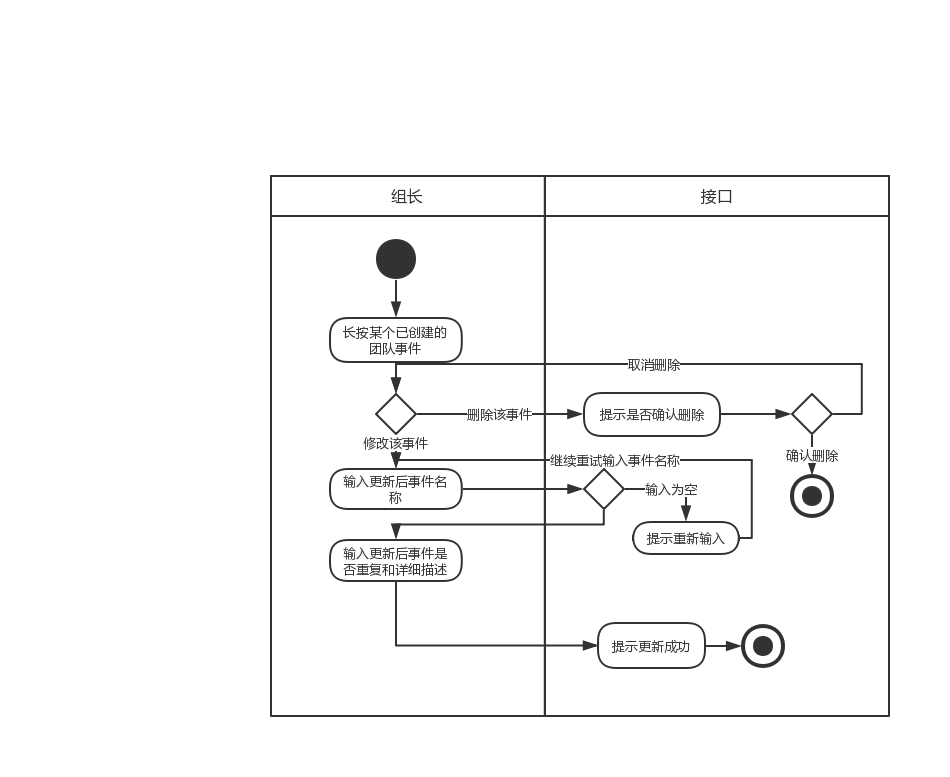


图2.7.修改团队事件用例的泳道图

可以从修改团队事件用例的泳道图清晰地看出该用例的逻辑流程。组长先长按某个已创建的团队事件卡片后，需要首先选择是否删除此事件，若选择删除，则系统再次确认是否删除，若用户确认，则删除事件，修改团队事件结束。若不选择删除，则进行事件信息的修改：首先输入修改后的事件名称，系统判断若为空则提示重新输入；接下来输入修改后的事件重复和详细描述，在网络状态良好的情况下，系统显示修改成功，完成修改团队事件。。

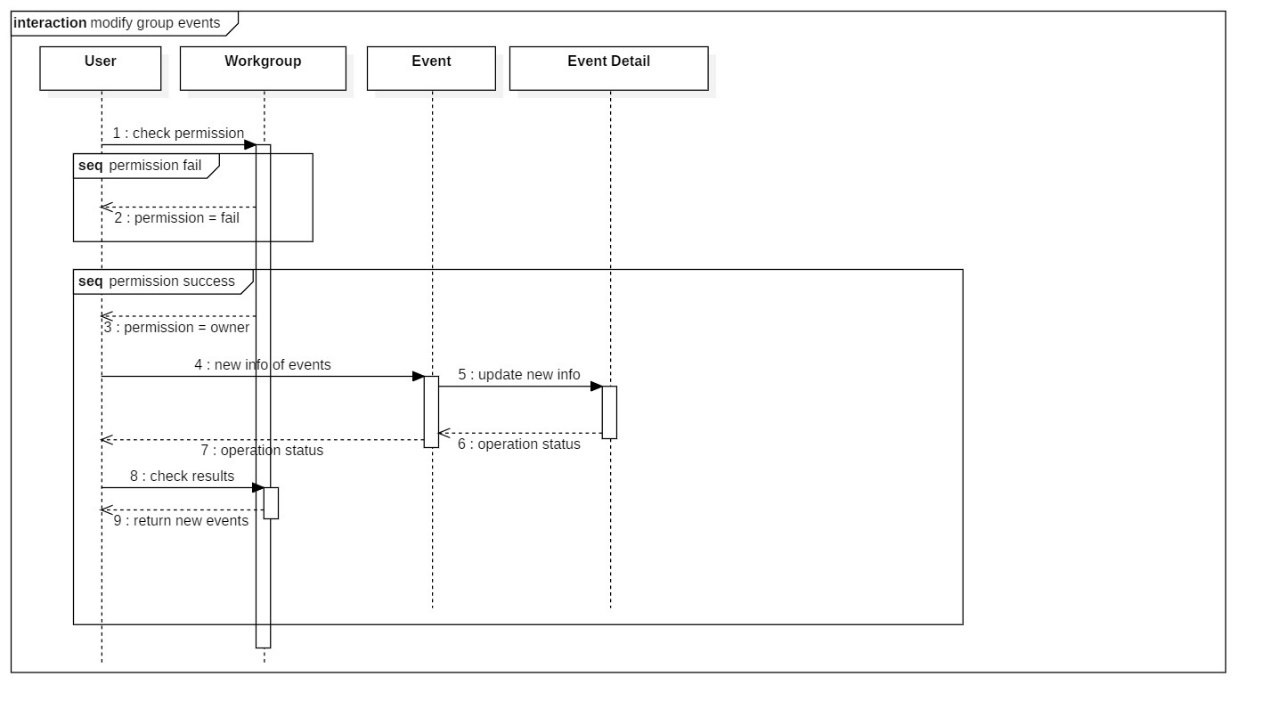


图2.7.修改团队事件用例的时序图

1. 创建团队日历

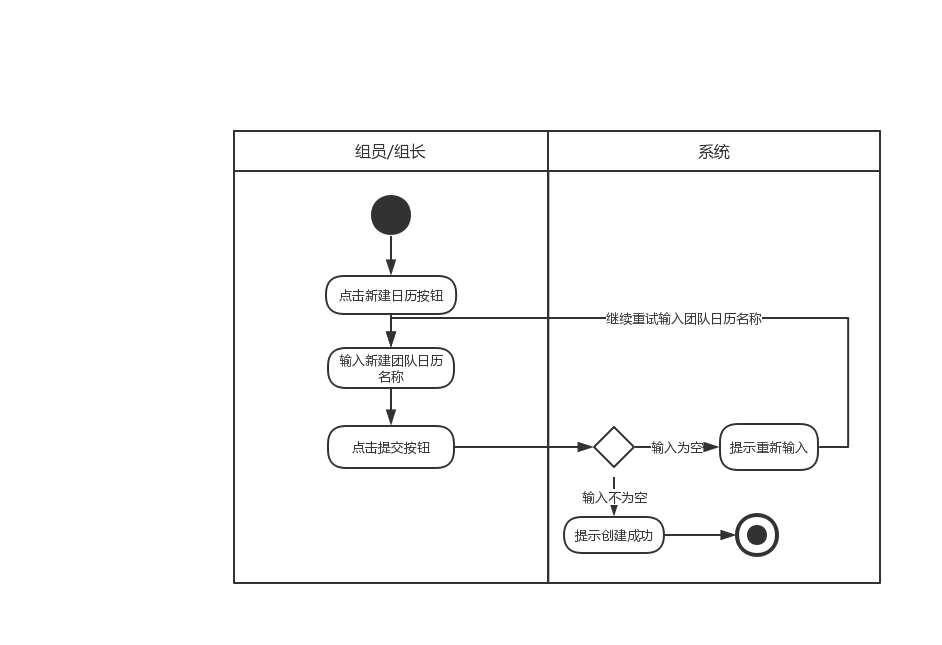


图2.8.创建团队日历用例的泳道图

可以从创建团队日历用例的泳道图清晰地看出该用例的逻辑流程。一个用户（组长或组员）点击新建日历按钮后，需要首先输入新建的团队日历的名称，并点击提交按钮。系统则会检测输入的团队日历名称是否为空，若为空则提示重新输入。若不为空，则提示创建成功，完成团队日历的创建操作。

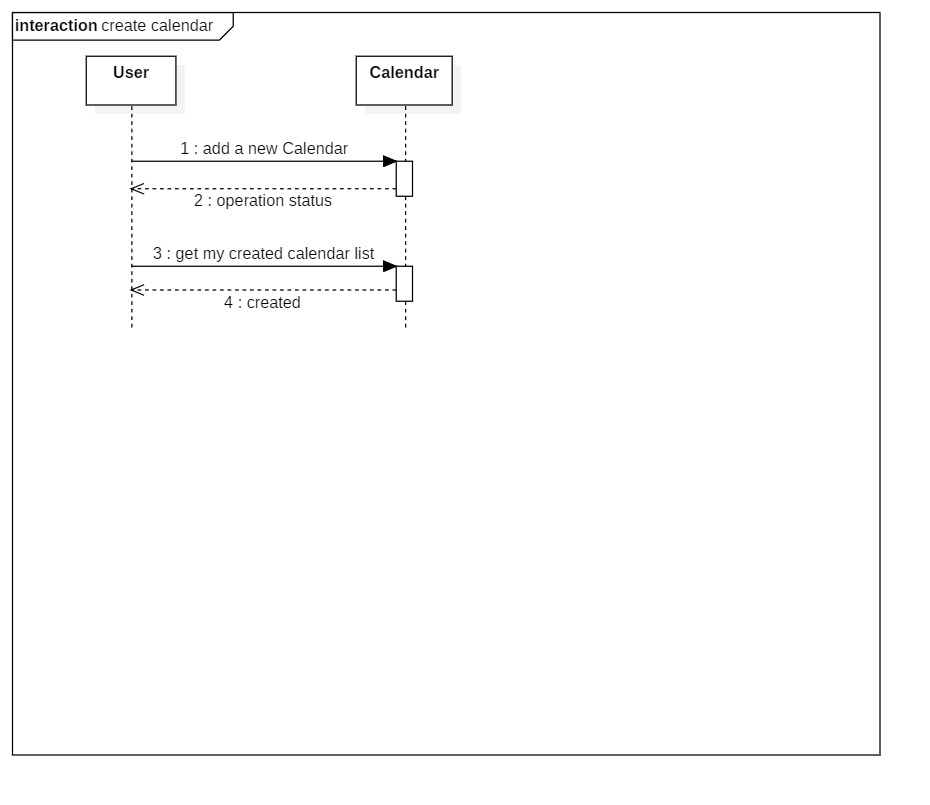


图2.9.创建团队日历用例的时序图

1. 解散团队日历

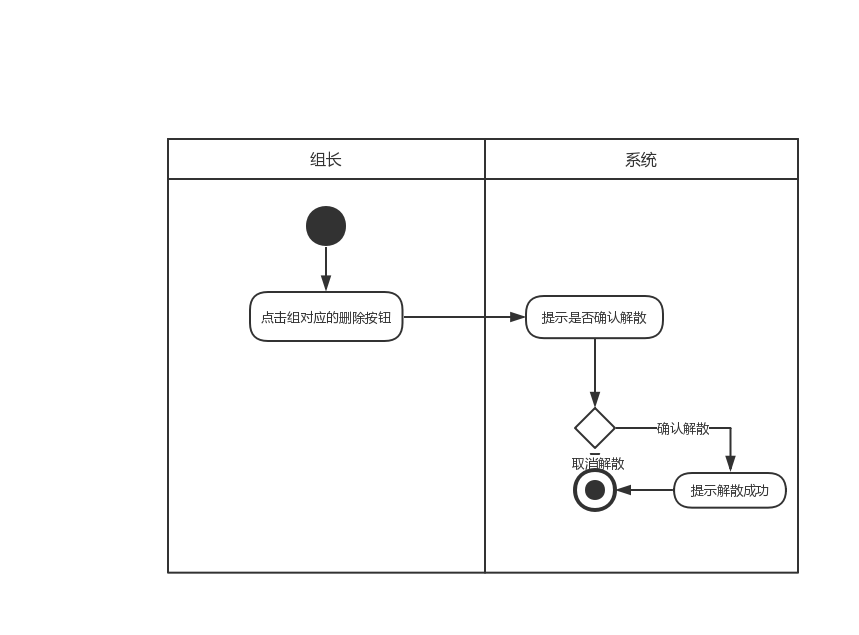


图2.10.解散团队日历用例的泳道图

可以从解散团队日历用例的泳道图清晰地看出该用例的逻辑流程。组长点击页面中当前组对应的传输按钮后，系统将会提示组长是否确认解散该团队。若用户确认解散且网络良好，则显示解散成功，完成解散团队日历操作。若用户取消解散，则退出该用例。

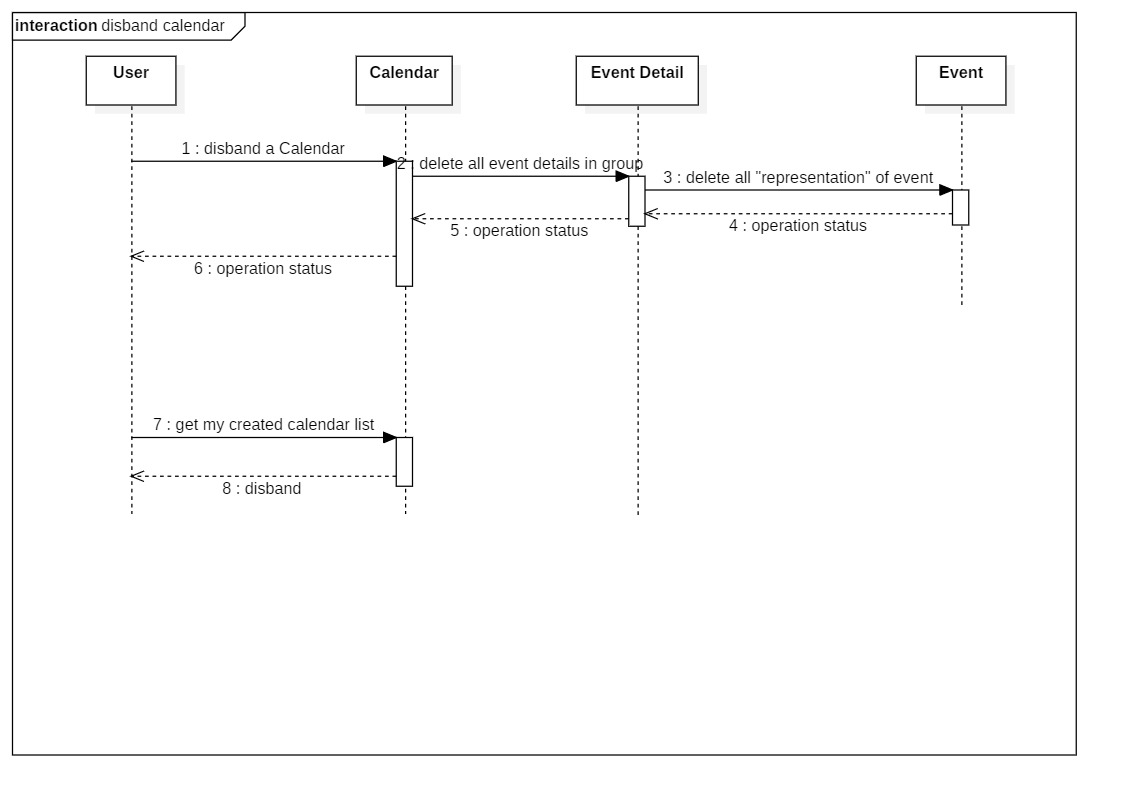


图2.11.解散团队日历用例的时序图

1. 分享团队日历

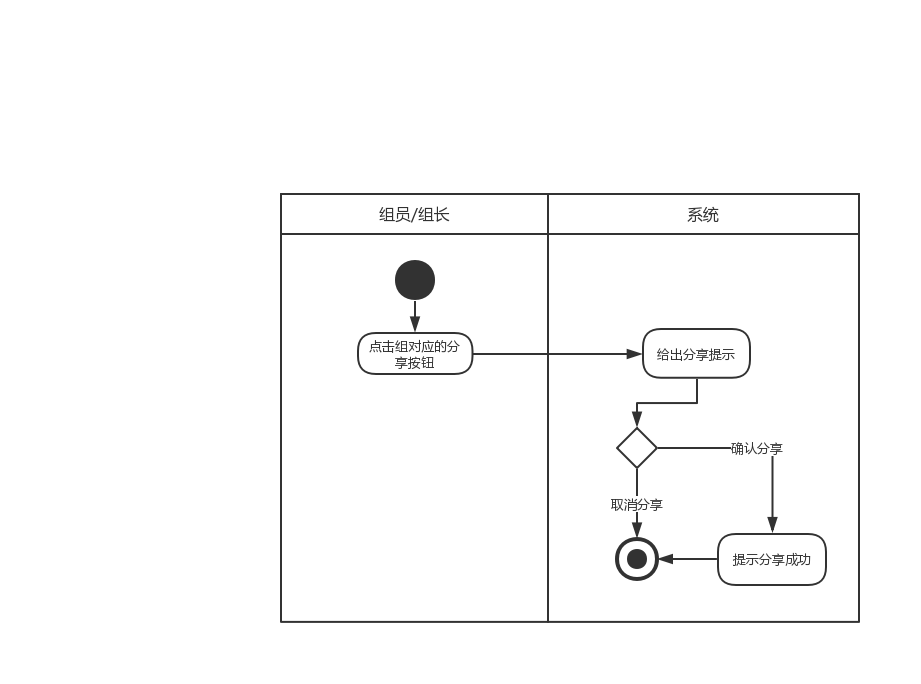


图2.12.分享团队日历用例的泳道图

可以从分享团队日历用例的泳道图清晰地看出该用例的逻辑流程。用户点击当前组对应的分享按钮后，系统会提示用户是否确认分享并选择分享对象，若用户选择确认分享，则提示分享成功，并完成分享团队日历操作。若取消分享，则退出用例。

1. 加入团队日历

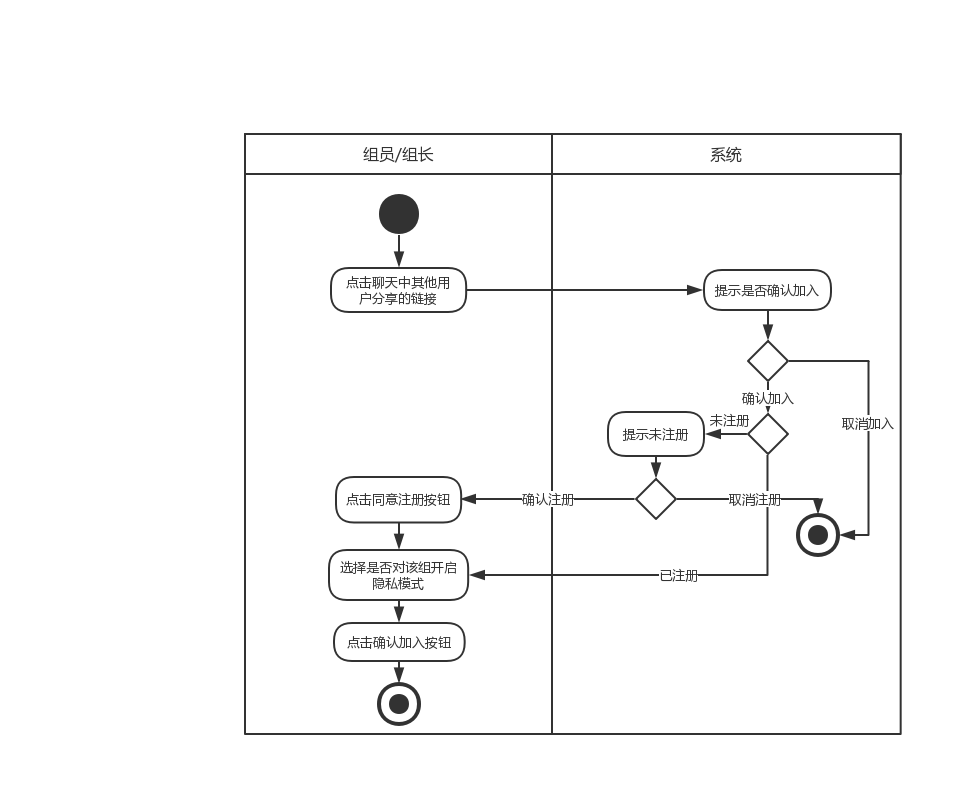


图2.13.加入团队日历用例的泳道图

可以从加入团队日历用例的泳道图清晰地看出该用例的逻辑流程。用户在微信聊天中点击其他用户分享的链接后，系统会提示用户是否确认加入团队日历，若用户取消加入，则退出该用例。若用户确认加入，则判断用户是否已经注册，若未注册，则系统提示用户注册，需要用户点击同意注册按钮即可，下一步已注册用户将选择是否对该组开启隐私模式，选择完成后点击确认加入按钮，则加入团队日历成功，完成加入团队日历操作。

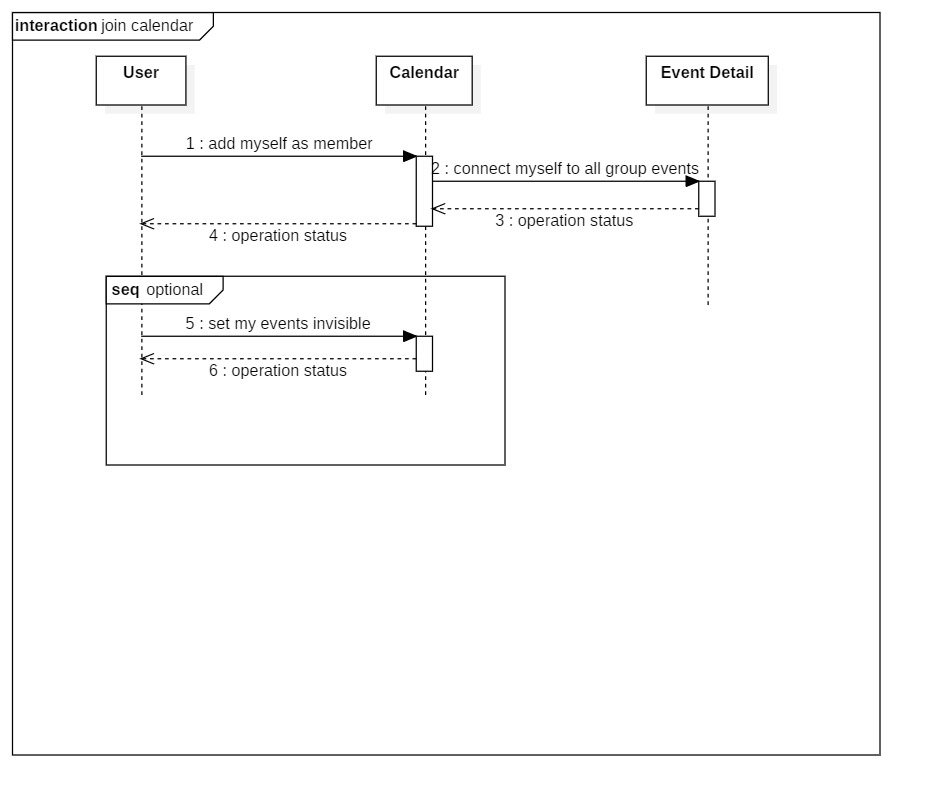


图2.13.加入团队日历用例的时序图

1. 退出团队日历

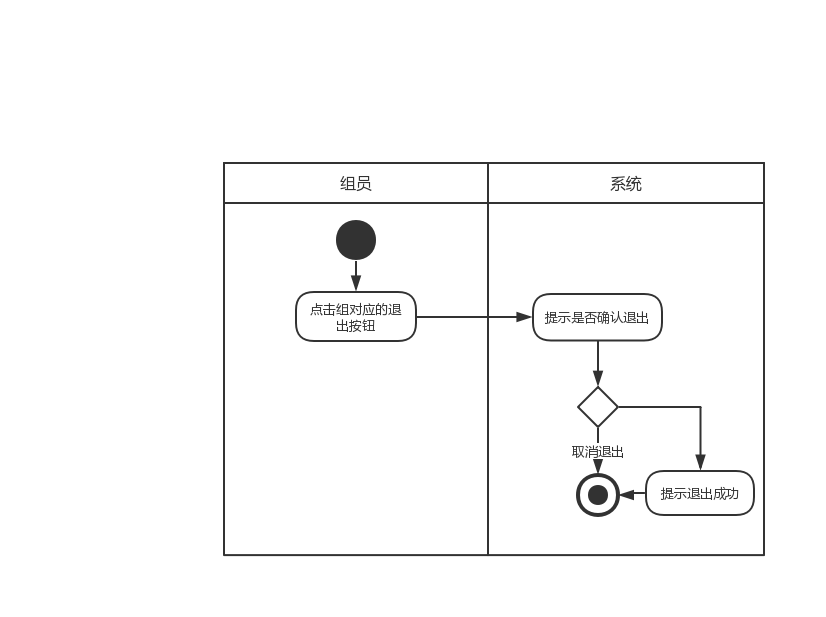


图2.14.退出团队日历用例的泳道图

可以从退出团队日历用例的泳道图清晰地看出该用例的逻辑流程。组员点击当前组对应的退出按钮后，系统将会提示用户是否确认退出，若用户确认退出，则系统提示退出成功，并完成退出团队日历用例。若用户取消退出，则退出用例。

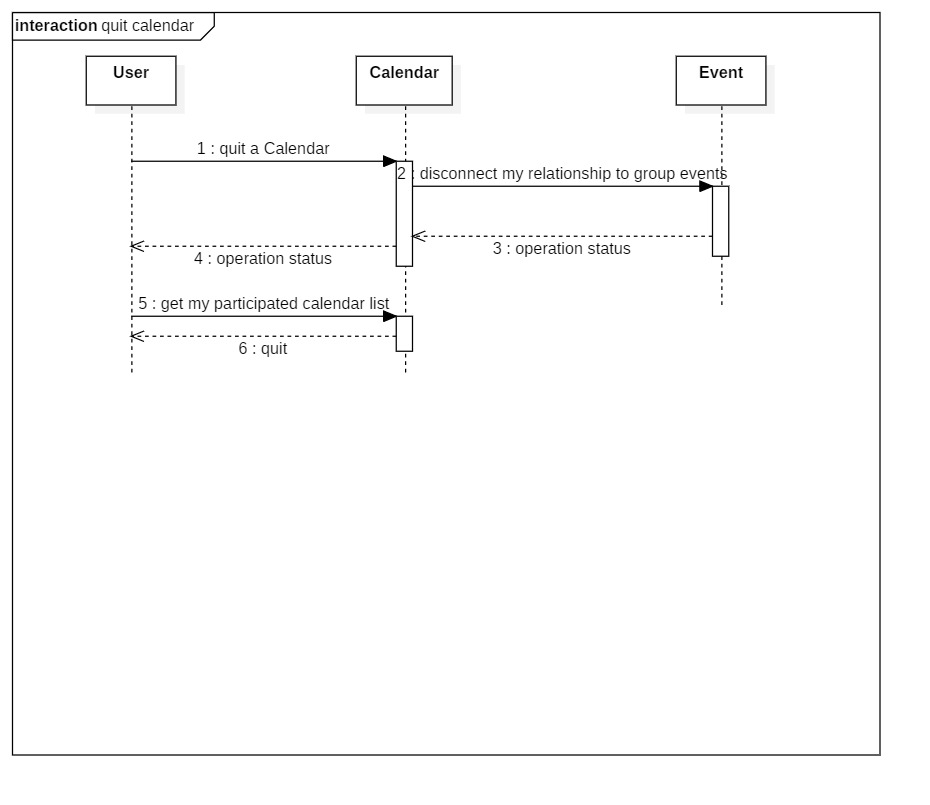


图2.15.退出团队日历用例的时序图

1. 注册

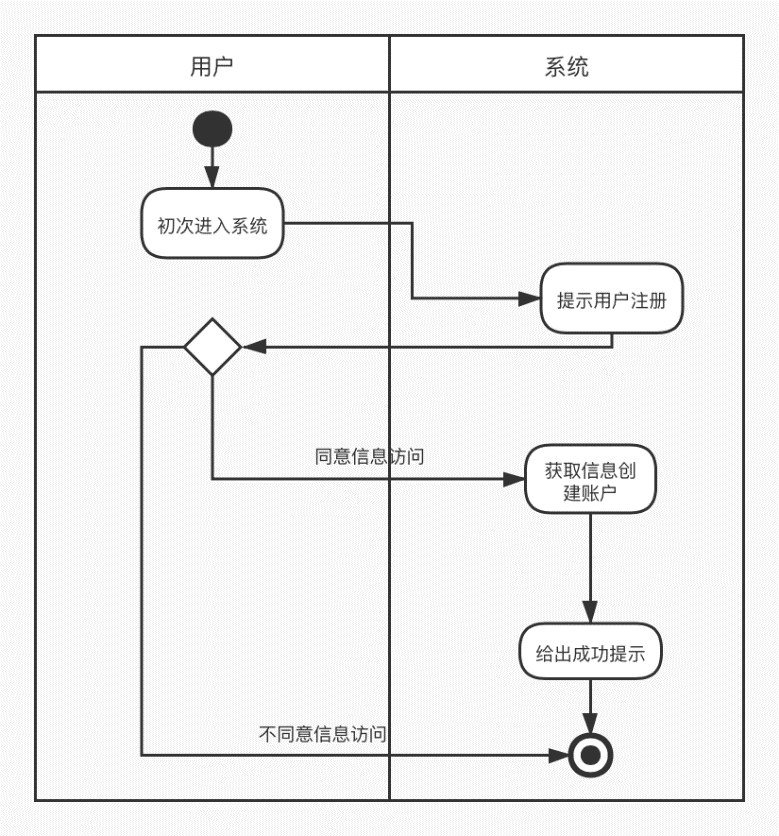


图2.16.注册用例的泳道图

可以从注册用例的泳道图清晰地看出该用例的逻辑流程。用户初次进入系统时，系统将提示用户进行注册，若用户不同意系统访问用户信息，则退出注册用例。若用户同意系统访问用户信息，则系统获取用户信息并创建账户，给出创建成功的提示，完成注册用例。

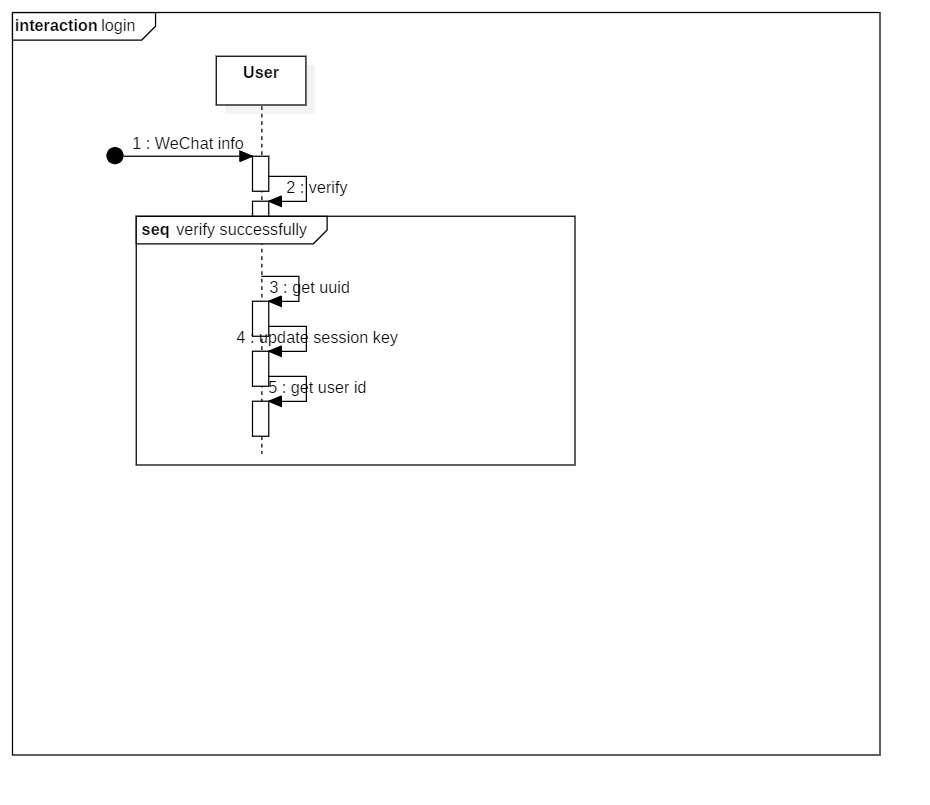


图2.17.注册用例的时序图

### 数据建模

#### 分析类图

我们通过基于类方法的需求建模，完成了系统的数据建模部分。我们首先根据需求模型开发的使用场景，结合用例的详细描述，用一段文字对系统进行“处理说明，再对其进行语法解析，得到了系统的潜在类。再基于需求模型合法类的特征，我们对潜在类进行筛选、描述属性、定义操作、创建关联和依赖，最后得到了本系统的分析类图（如图2.12所示）：

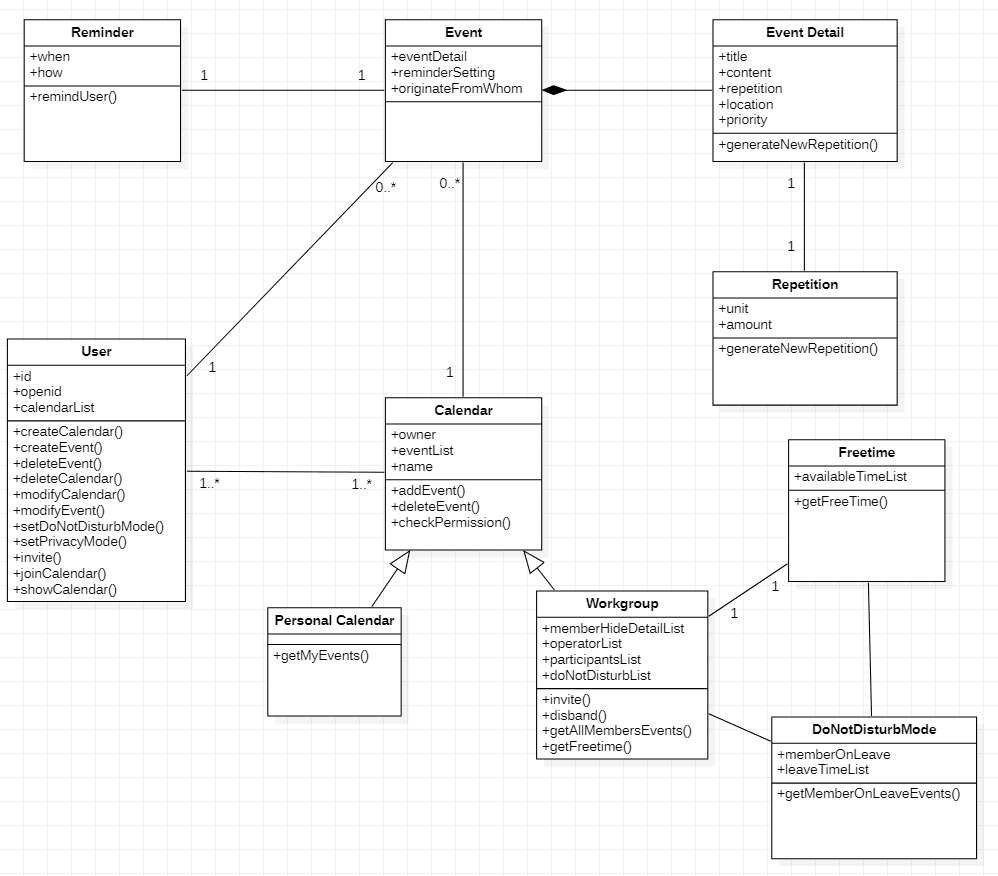


图2.12.系统的分析类图

如上图所示，我们主要识别出了10个分析类。接下来，将对每一个分析类内的属性和操作及与其他类的关联依赖关系做简要描述。

1. **Reminder类**

主要定义了给用户的消息提醒信息。

包括的属性有：

when 时间

how 提醒方式

包含的方法有

remindUser() 用于主动提醒用户

Reminder类与Event类一一对应关联。

1. **Event类**

主要定义用户创建的事件信息。

包括的属性有：

eventDetaill 关联的事件详细信息

reminderSeting 创建的提醒器

originateFromWhom 标识创建者信息

没有包含的方法

Event类与Reminder类一一对应关联。

EventDetail类依赖于Event类。

Event类与User类是多对一的关系。

Event类与Calendar类是多对一的关系

1. **EventDetail类**

主要定义了事件的详细信息。

包括的属性有：

title 事件名称

content 事件内容

repetition 事件创建的重复信息

location 事件具体时间

priority 事件的优先级

包含的方法有

generateNewRepetition() 用于创建一个新的重复信息

EventDetail类依赖于Event类。

EventDetail类与Repetition类一一对应关联。

1. **Repetition类**

主要定义了事件的重复信息。

包括的属性有：

unit 事件名称

amount 事件内容

包含的方法有

generateNewRepetition() 用于创建一个新的重复信息

Repetition类与EventDetail类一一对应关联。

1. **User类**

主要定义了用户的信息。

包括的属性有：

id 用户唯一标识符

openid 系统生成的用户编号

calendarList 用户参与的日历列表

包含的方法有

createCalendar() 用于创建一个新的日历

createEvent() 用于创建一个新的事件

deleteEvent () 用于删除一个事件

deleteCalendar() 用于删除一个日历

modifyCalendar() 用于修改一个日历

modifyEvent() 用于修改一个事件

setDoNotDisturbMode() 用于设置勿扰模式

setPrivacyMode() 用于设置私密模式

invite() 用于邀请他人进入团队日历

joinCalendar() 用于进入他人邀请进入的日历

showCalendar() 用于显示日历

User类与Event类是一对多的关系。

User类与Calendar类是多对多的关系。

1. **Calendar类**

主要定义了日历的信息。

包括的属性有：

owner 拥有者

eventList 包含的事件列表

name 日历名称

包含的方法有

addEvent() 用于创建一个新的事件

deleteEvent() 用于删除一个事件

checkPermission() 用于判断用户权限

Calendar类与Event类是一对多的关系。

Calendar类与User类是多对多的关系。

Calendar类派生出了PersonCalendar类和WorkGroup类。

1. **PersonCalendar类**

主要定义了个人的详细信息。

无包括的属性

包含的方法有

getMyEvents() 用于获取个人事件信息

PersonCalendar类继承于Calendar类。

1. **Workgroup类**

主要定义了团队日历的详细信息。

包括的属性有：

memberHideDetailList 设置为私密模式的用户列表

operatorList 操作列表

participantsLIst 参与团队日历的组员列表

doNotDisturbList 设置为勿扰模式的用户列表

包含的方法有

invite() 用于邀请新成员

disband() 用于解散组日历

getAllMembersEvents() 用于获取所有成员的事件

getFreeTime() 用于确定所有成员的空闲时间

Workgroup类继承于Calendar类。

Workgroup类与Freetime类一一对应关联。

Workgroup类与DoNotDisturbMode类关联。

1. **FreeTime类**

主要定义了空闲时间的详细信息。

包括的属性有：

availableTimeList 空闲时间列表

包含的方法有

getFreeTime() 用于获取所有成员你的空闲时间

FreeTime类与Workgroup类一一对应关联。

FreeTime类与DoNotDisturbMode类关联。

1. **DoNotDisturbMode类**

主要定义了勿扰模式的信息。

包括的属性有：

memberOnLeave 勿扰模式用户列表

leaveTimeList 开启勿扰模式的时间列表

包含的方法有

getMemberOnLeaveEvents() 用于获取开启勿扰模式的用户列表

DoNotDisturbMode类与Workgroup类关联。

DoNotDisturbMode类与FreeTime类关联。

#### CRC模型

在进行基于类的方法建模时，为了识别和组织系统或产品需求的相关类，我们还采用了类-职责-协作者建模（CRC建模）的方法。CRC模型索引卡如下所示：

|  |  |
| --- | --- |
| 类：User |  |
| 职责： | **协作者：** |
| id |  |
| openid |  |
| 创建/删除/修改日历 | Calendar |
| 创建/删除/修改事件 | Event |
| 设置勿扰模式 | Workgroup |
| 共享日历 | Calendar |
| 加入日历 | Workgroup |
| 查看日历 | Calendar |
| 设置隐私模式 | Workgroup |

|  |  |
| --- | --- |
| 类：Calendar |  |
| 职责： | **协作者：** |
| 日历所有者 |  |
| 事件列表 |  |
| 日历名称 |  |
| 加入/删除事件 | Event |
| 检查用户操作日历权限 |  |
| 展示日历事件列表 | Event |

|  |  |
| --- | --- |
| 类：Personalcalendar |  |
| 职责： | **协作者：** |
| 获取个人日历事件 |  |

|  |  |
| --- | --- |
| 类：Workgroup |  |
| 职责： | **协作者：** |
| 隐私模式用户列表 |  |
| 组成员列表 |  |
| 勿扰模式用户列表 |  |
| 生成分享所需日历信息 |  |
| 获取组成员事件 | PersonalCalendar |
| 获得小组成员共有空闲时间 | Freetime, DoNotDisturbMode |
| 添加组成员 |  |
| 去掉组成员 |  |
| 解散工作组 |  |
| 获取设置勿扰用户列表 |  |

|  |  |
| --- | --- |
| 类：Event |  |
| 职责： | **协作者：** |
| 获取事件提醒方式 | Reminder |
| 设置事件提醒 | Reminder |
| 设置事件详细信息 | EventDetail |
| 获取事件详细信息 | EventDetail |
| 设置事件重复 | Repetition |
| 获取事件重复 | Repetition |

|  |  |
| --- | --- |
| 类：EventDetail |  |
| 职责： | **协作者：** |
| 标题 |  |
| 事件开始及结束时间 |  |
| 重复 |  |
| 优先级 |  |
| 事件内容 |  |
| 生成新的重复事件详细信息 | Repetition |
| 获取事件标题、内容、时间、重复、优先级等详细信息 |  |
| 设定事件标题、内容、时间、重复、优先级等详细信息 |  |

|  |  |
| --- | --- |
| 类：Repetition |  |
| 职责： | **协作者：** |
| 提醒时间 |  |
| 生成新的事件重复信息 |  |
| 重复时间间隔 |  |

|  |  |
| --- | --- |
| 类：Freetime |  |
| 职责： | **协作者：** |
| 空闲时间列表 |  |
| 获取空闲时间段列表 |  |
| 计算空闲时间 |  |

|  |  |
| --- | --- |
| 类：reminder |  |
| 职责： | **协作者：** |
| 用户事件提醒 |  |
| 提醒时间 |  |
|  |  |

|  |  |
| --- | --- |
| 类：DoNotDisturbMode |  |
| 职责： | **协作者：** |
| 设置勿扰的用户 |  |
| 获取设置勿扰用户事件 | PersonalCalendar |

之后，将协作类的名称记录在CRC模型索引卡上，仅靠在协作的职责旁边。我们将有协作关系的CRC模型索引卡片紧靠，按协作者的索引关系进行排列，就得到了如图2.13所示的CRC排列图，这将对我们进行需求分析建模提供极大的便利。

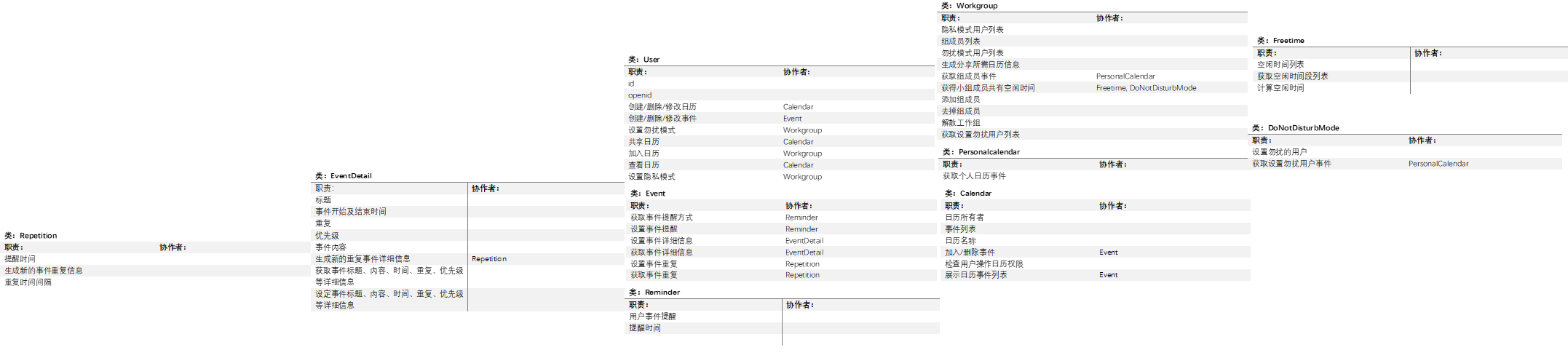


图2.13.系统的CRC排列图

#### 分析包

为了将分析模型中的各种元素（如用例、分析类）以一种方式分类，我们其进行分组打包。根据需求，我们将系统分成Characters、Event、Calendar三个分析包。如图2.14所示。

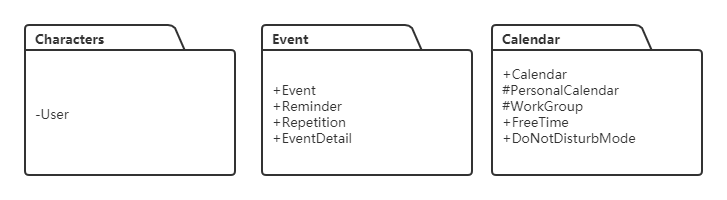


图2.14.系统的分析包图

### 行为建模

经过对需求的严谨分析和对实际相关系统及平台情况的考量，系统在运行期间没有系统级别的状态变化。因此没有系统级别的状态图构建。但在系统运行，执行其功能时，某些类有状态的变化。因此，在行为和模式的需求建模中，我们对一些业务类构建了状态图。

1. **Calendar类**

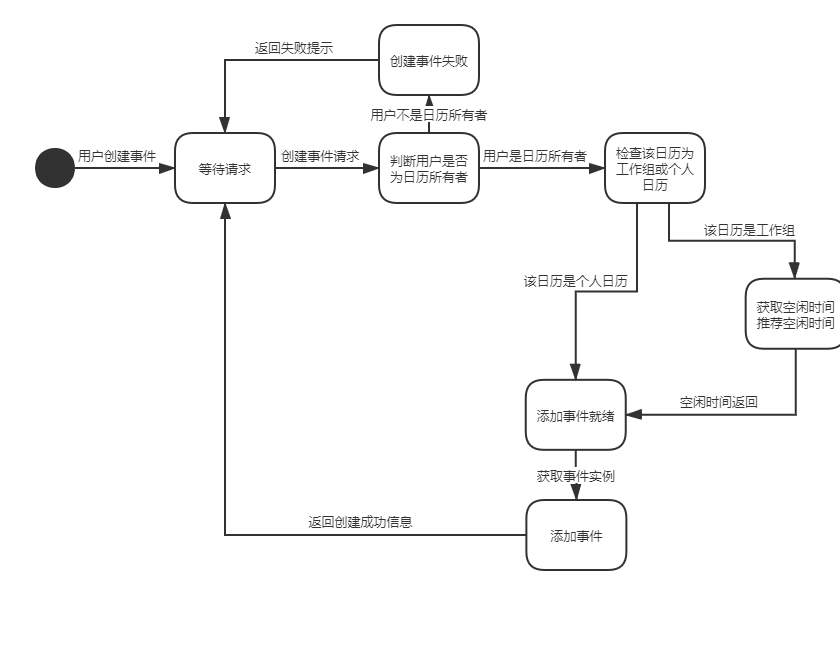


图2.15.Calendar类的状态图

如图2.15所示，Calendar类初始状态会处于等待创建事件的请求的状态。当有用户创建事件信息传入后，它将进入判断用户是否为日历所有者的状态。并根据判断结果进行状态的变更。若用户不是日历所有者，它将处于创建事件失败状态，并返回失败提示，回到初态。若用户是日历所有者，则Calendar类将进入检查该日历为工作组或个人日历状态。若该日历是工作组，则进入获取及推荐空闲时间状态，并返回计算出的空闲时间。若该日历是个人日历，则直接进入添加事件就绪状态。获取事件实例后变为添加事件状态。添加成功后返回创建成功信息，并返回等待请求的初态。

1. **Event类**

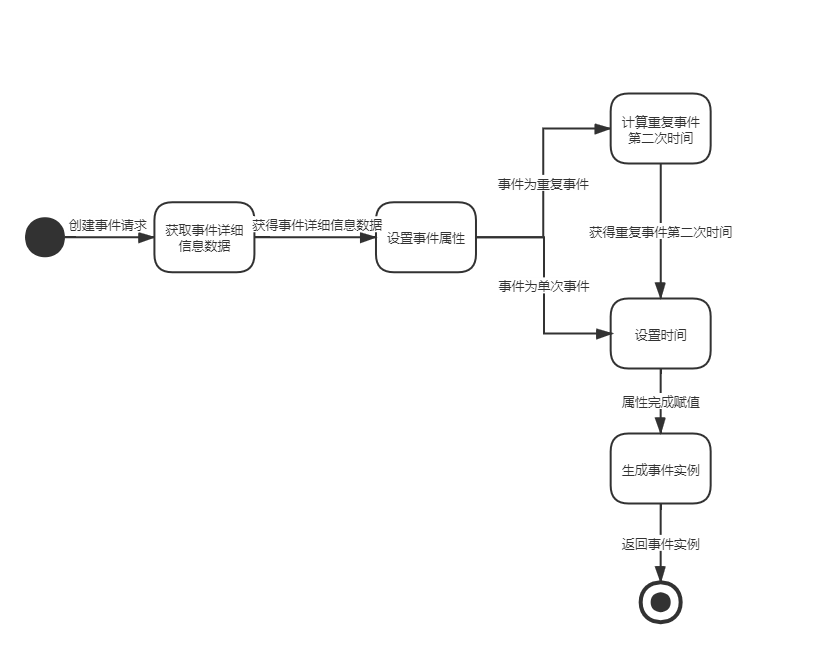


图2.16.Event类的状态图

如图2.16所示，Event类由系统发出创建事件请求而被初始化为获取事件详细信息数据状态，在接收信息获得事件详细信息数据后，它变为设置事件属性状态。若事件为重复事件，则Event类会计算重复事件的第二次时间并完成设置。若事件为单次事件，则直接进入设置时间状态。在属性完成赋值后进入生成事件实例状态。最终返回生成的事件实例。

1. **Freetime类**

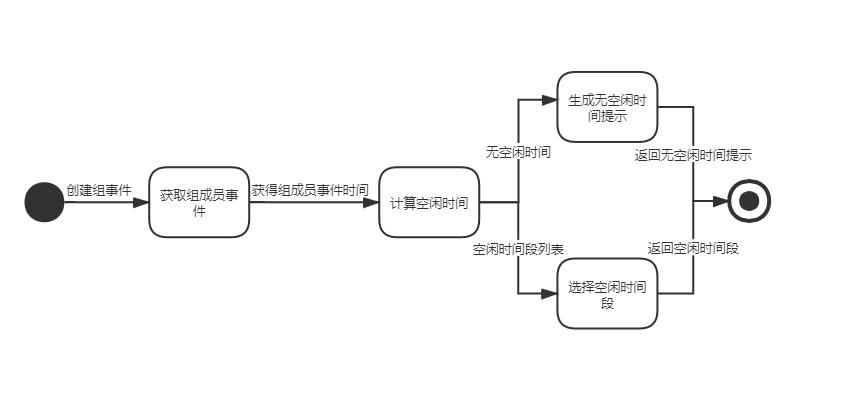


图2.17.Freetime类的状态图

如图2.17所示，Freetime类由创建组时间的请求而初始化为获取组成员事件状态，在获取了组成员事件的时间后，它将计算组成员的空闲时间。将计算出的空闲时间段列表生成出来并提供选择，选择好后返回空闲时间段，结束运行状态。若计算出无空闲时间，则返回无空闲时间的提示，结束运行状态。

### 内容建模

在以上的功能、数据、行为建模之外，项目需求工程中还依照WebAPP的需求建模规范，进行了前端界面的内容建模。我们主要使用Datatree的方式进行内容建模。

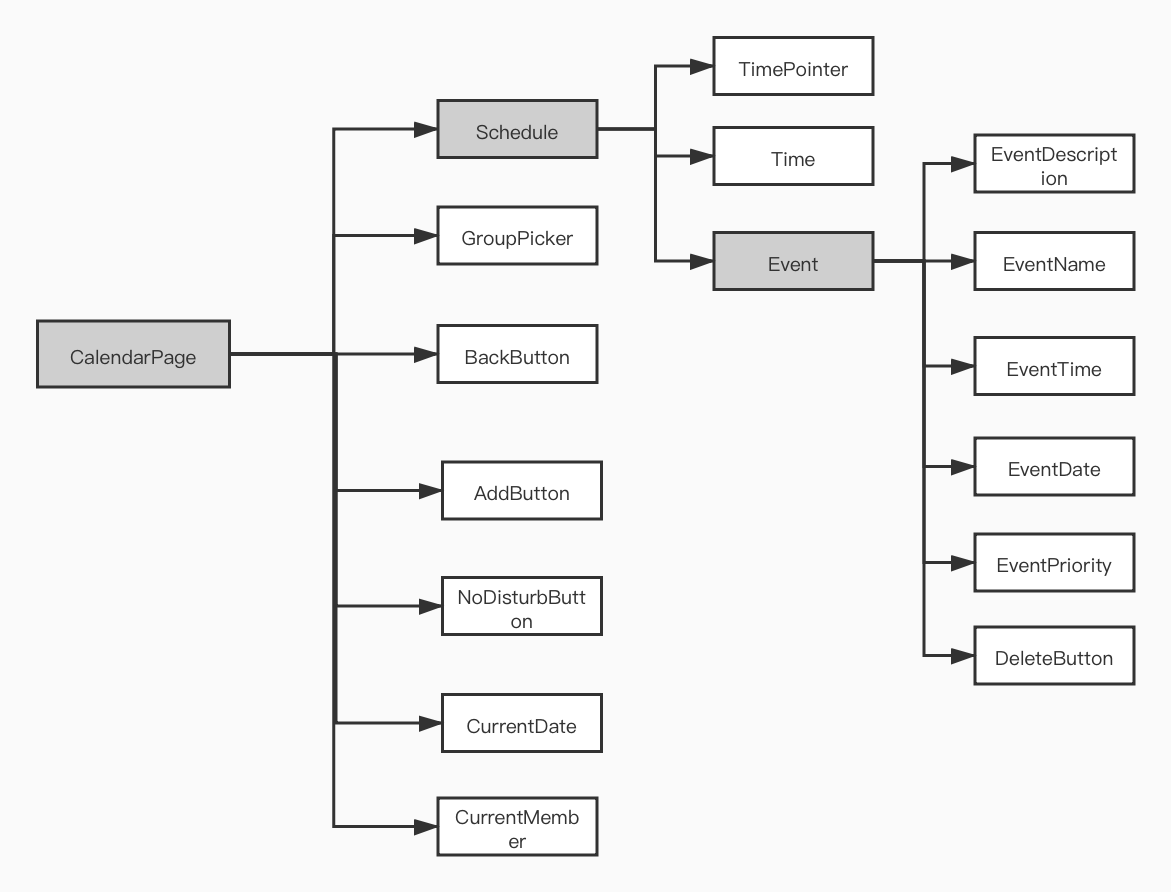


图2.18.Calendar界面的Datatree

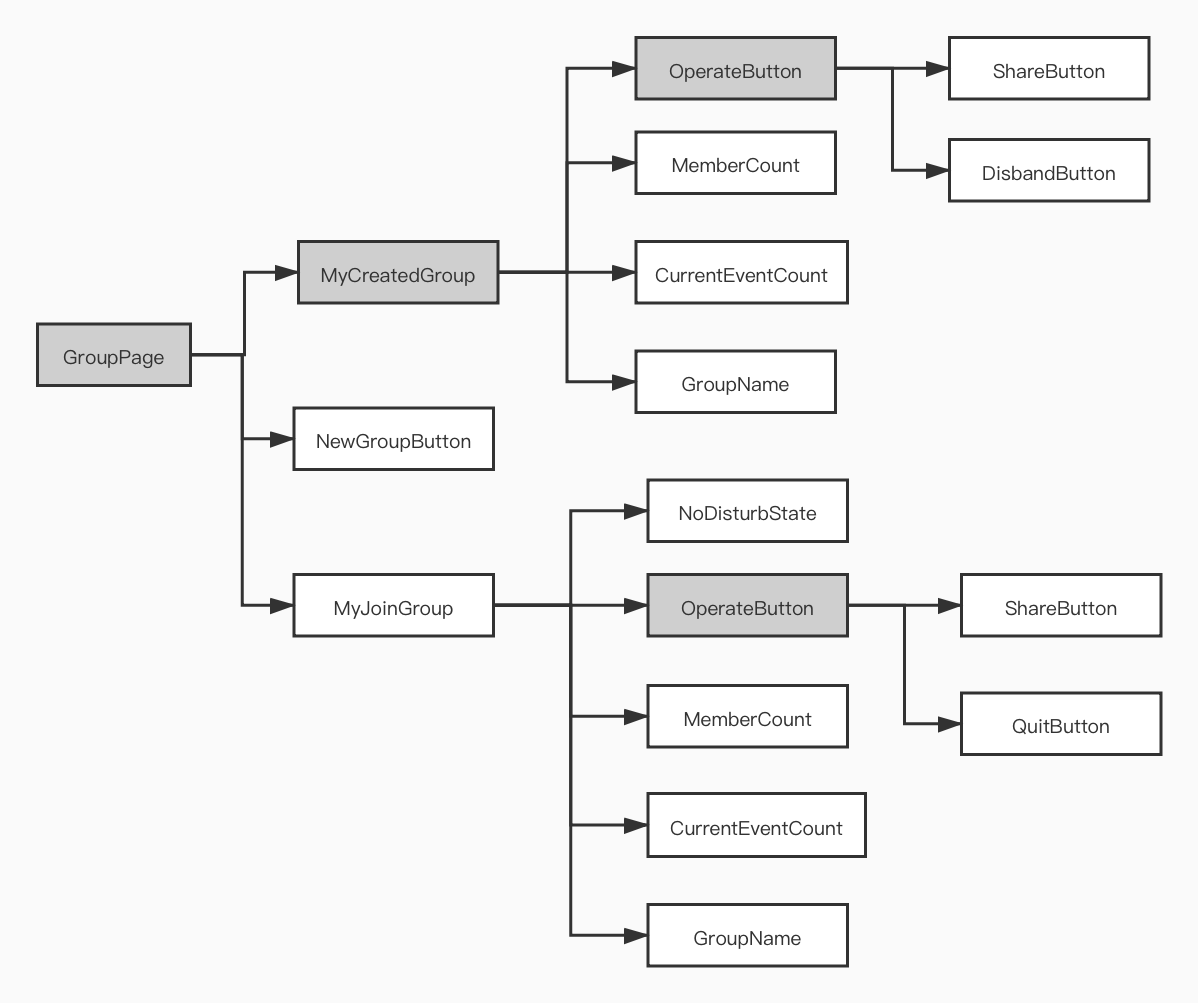


图2.19.GroupPage界面的Datatree

# 非功能需求

## 性能要求

### 响应时间

在99％的情况下，在网络畅通时，一般时段响应时间不超过1.5秒，高峰时段不超过3秒。

每次操作从点击到第一个界面显示出来所需要的时间不得超过1秒。

在推荐配置环境下：登录响应时间在0.5秒内，打开信息条目响应时间1秒内。

### 业务量

估计用户数为500人，每天登录用户数为200左右。

系统可以同时满足100个用户请求，并为1000个并发用户提供浏览功能。

### 系统容量

支持3000个用户，支持GB级数据。

数据库表行数不超过5万行，数据库最大容量不超过30GB，磁盘空间至少需要40G以上。

### 精度

在计算推荐公共事件时间时，推荐时间段误差<0.5小时。

### 资源使用率

对移动设备 CPU占用率<=50%。

内存占用率<=50%。

## 数据管理能力要求

本系统后端环境需要运行在Windows 8及以上、MacOS 10.7及以上或Ubuntu 14.04及以上并装有Java Development Kits 1.8的操作系统之上。

本系统前端环境需要运行在WeChat 7.0.0以上，基于Mobile Chrome 67渲染。

在系统的服务端与MySQL数据库的连接和操作上，我们主要使用MyBatis框架。它支持定制化 SQL、存储过程以及高级映射。相较于JDBC，MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

## 安全及保密性要求

### 权限访问控制

用户在经过登录操作进行身份认证后，只能访问其权限范围内的数据，只能进行其权限范围内的操作。

不同的用户具有不同的身份和权限，需要在用户身份真实可信的前提下，提供可信的授权管理服务，保护数据不被非法/越权访问和篡改，要确保数据的机密性和完整性。

### 日志控制

提供运行日志管理及安全审计功能，可追踪系统的历史使用情况。

### 恶意请求防御

能经受来自互联网的一般性恶意攻击。如病毒（包括木马）攻击、口令猜测攻击、黑客入侵等。

## 兼容性需求

### 系统与其他软件兼容性

后端系统应支持MacOS，Windows，Ubuntu操作系统。

前端系统应兼容前后多个版本的微信软件。

### 与操作系统交互

最多只有5%的系统实现需要具体到特定的操作系统。

### 可移植性

移植系统时无需配置大量环境，

## 其他专门要求

### 可靠性需求

**输入可靠性**

系统对输入字符有提示，数据有检查，防止数据异常造成的系统失效。

**系统健壮性**

系统健壮性强，应该能处理系统运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、硬件设备失败等，系统应该能正确的处理，恰当的回避。

**运行可靠性**

系统7x24小时运行，全年持续运行故障停运时间累计不能超过10小时。

系统缺陷率每1,000小时最多发生1次故障。

因软件系统的失效而造成不能完成业务的概率要小于5%。

### 易用性

**易理解性**

系统界面的图标及文字提示不应有歧义，需便于用户理解。

**易学习性**

绝大多数用户应该能在5分钟内根据系统所给的提示，了解系统的基本操作。

**易操作性**

系统界面组件的排布、按钮的设计及文字大小等应该符合界面友好性，满足HCI的设计规范。在进行页面布局时，除考虑美观外，也应考虑不同用户的操作习惯，用户进行的常用操作按钮放在易操作的位置。

# 运行环境规定

## 设备

**本系统后端运行测试硬件环境为：**

处理器 Inter i5-7250

内存 DDR4 8GB

存储空间 256GB SSD

输入设备 鼠标（触控板），键盘

输出设备 显示器

**本系统前端运行硬件条件为：**

移动设备 手机

处理器 麒麟870/高通骁龙636以上

内存 3GB

存储空间 32GB

输入设备 触控输入、陀螺仪

输出设备 显示器

## 支持软件

系统运行环境为：

前端：WeChat Mini Program平台

服务层：Ubuntu 16.04

Java 1.8

数据库：Ubuntu 16.04

MySQL 5.6.40

## 接口

### 用户接口

用户可使用智能移动设备通过在微信小程序平台搜索“Meeting”进入本系统的用户界面，通过在设备上的点击、滑动、长按、倾斜与系统进行交互。

### 软件接口

本系统后端环境需要运行在Windows 8及以上、MacOS 10.7及以上或Ubuntu 14.04及以上并装有Java Development Kits 1.8的操作系统之上。

本系统前端环境需要运行在WeChat 7.0.0以上，基于Mobile Chrome 67渲染。

在系统的服务端与MySQL数据库的连接和操作上，我们主要使用MyBatis框架。它支持定制化 SQL、存储过程以及高级映射。相较于JDBC，MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

### 硬件接口

访问本系统前端微信小程序，需要能够运行微信小程序的智能移动设备（如手机、平板电脑等），且微信具有访问设备陀螺仪的权限。

本系统后端运行需要计算机硬件要求内存大小至少500M，硬盘64G以上。

### 通信接口

前端与后端进行的数据传输采用HTTPS协议，它是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比HTTP协议更加安全。

# 需求跟踪

本项目开发时，我们采用Scrum敏捷开发框架。在框架活动中，我们将整个开发周期分成三个预定的时间段，分为三个sprint阶段。在项目开始的讨论阶段，我们创建了一个待定项backlog列表，记录所有的有商业价值的项目需求或特性的优先级列表，并按照优先级进行排序，给每一个sprint分配故事并进行故事点估算。在随后的项目开发中，我们持续更新图表，在每个sprint中，我们会每天估算自己完成的工作量，并计算剩余工作量，绘制相应燃尽图。