“酒井实践在线”设计文档

1. 引言

> 描述软件设计的概况，由哪些部分组成。

本文档的编写目的是描述 酒井实践在线（THUPracticeOnline）软件的设计信息，以支持后续的开发和测试。

本文档包括以下章节内容：

- **软件设计约束部分：**描述软件设计目标和原则、软件设计的约束和限制、技术选型的具体内容。
- **软件设计部分：**描述用例设计、软件体系结构设计、数据库设计和部署设计的具体内容。

2. 软件设计约束

2.1 软件设计目标和原则

> 描述软件设计欲达到的目标，如实现用户需求，软件系统具有良好的可扩展性等。
>
> 描述为实现软件设计目标，在设计软件过程中应遵循的设计原则。

酒井实践在线（THUPracticeOnline）软件设计的目标是给出软件系统的实现解决方案和蓝图，产生可指导编码实现的设计模型及文档。

酒井实践在线（THUPracticeOnline）软件设计须遵循相关的策略和原则，以指导软件设计人员的行为，并对设计成果提出约束和要求。具体地，这些设计策略和原则描述如下：

- 抽象和逐步求精原则
- 模块化与高内聚度、低耦合度原则
- 信息隐藏原则
- 多视点以及关注点分离原则
- 软件重用原则
- 迭代设计原则
- 可追踪性原则

2.2 软件设计的约束和限制

> 列举和描述软件设计需要考虑的约束和限制
>
> - 操作系统要求
> - 软件依赖
> - 硬件要求

- 硬件要求：
  - 前端网页运行环境要求：
    - 手机CPU型号不低于骁龙630，手机运行内存不少于2G，至少4G的存储空间，建议使用较新的操作系统版本，如 Android 10 及以上或 iOS 13 及以上。
    - 电脑CPU型号不低于 Intel Core i3 或 AMD Ryzen 3 系列，内存最低要求为 4G RAM，推荐配置为 16GB 或更高。
  - 后端服务器运行环境要求：4核CPU，8G内存，500G硬盘，带宽5M以上。
- 软件要求：
  - 前端网页运行环境要求：
    - 手机：Android 4.4以上。
    - 电脑：Windows 10 或更高版本、macOS Catalina（10.15）或更高版本、Ubuntu 20.04 LTS或更高版本均可。
  - 服务器软件的具体要求见以下表格：
    - 后端：

软件名称	软件用途	版本
操作系统Ubuntu	服务器的操作系统	16.04
Docker	用于持续集成和持续部署流程	20.10.17
MySQL	数据库管理系统	15.1
Redis	缓存数据库	7.0.15
Nginx	域名转发，代理服务器	1.10.1
Python	后端编程语言	3.11
Uwsgi	后端并发代理	2.0.29
Mariadb-client	后端数据库工具	11.6.2

    - 前端：

软件名称	软件用途	版本
操作系统Ubuntu	服务器的操作系统	16.04
Docker	用于持续集成和持续部署流程	20.10.17
Nginx	域名转发，代理服务器	1.10.1
Node	前端运行环境	22

2.3 技术选型

> 描述为了开发软件，使用的已有技术和工具，包括：编程语言、前后端框架、数据库方案、部署方案等。

前端开发技术选型

- 编程语言：HTML，CSS，TypeScript
- 前端界面的开发框架：React, Next.js
- 第三方工具库：ant-design
- 集成开发环境（IDE）：VSCode, Vim
- 软件打包工具：pnpm

后端开发技术选型

- 编程语言：Python 3.11
- 数据库：MySQL关系型数据库，Redis缓存数据库
- 基于 Mariadb 的数据库访问
- 接口开发规范：RESTful API标准
- 部署工具：Docker，Nginx

3. 软件设计

3.1 用例设计

> 给出各个用例的设计模型，比如描述用例实现的顺序图等，并提供必要的文字补充说明。

用户注册用例实现的设计方案

“用户注册”功能的实现主要通过SendEmail视图和Register视图，根据用户邮箱合法性发送验证码，并检查用户输入的用户信息，如果通过合法性检查则创建用户数据。详见图 1 所描述的用例实现顺序图。

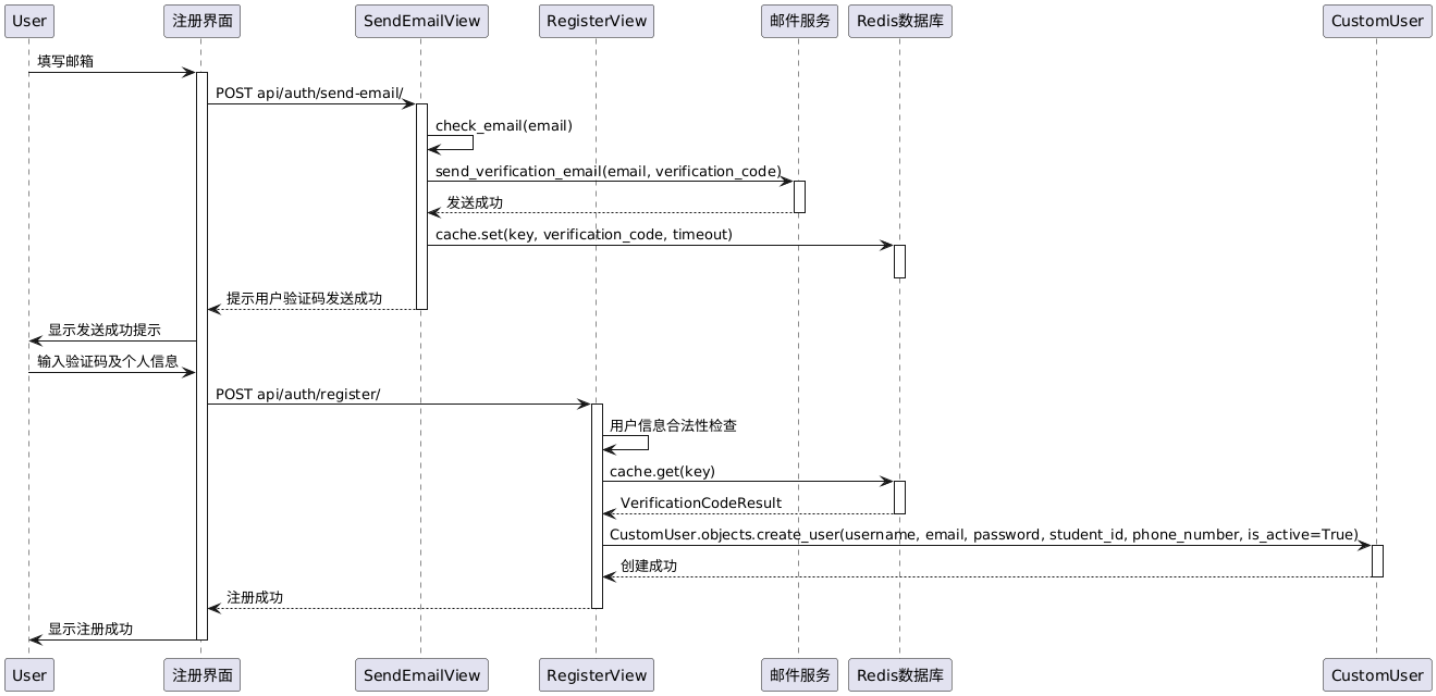


图1. 用户注册用例设计的顺序图

- 用户进入注册界面，输入邮箱并向后端的api/auth/send-email/请求发送验证码。
- SendEmailView收到用户邮箱后进行格式验证，确保为清华邮箱。
- 邮箱格式验证通过后生成并通过邮件服务发送验证码，并在Reddis数据库中临时存储验证码，前端界面提示用户验证码发送成功。
- 用户将完整的个人信息及收到的验证码向后端的api/auth/register/请求注册。
- RegisterView收到用户数据后进行合法性检查，并与Reddis中记录的验证码进行比对，若比对成功则在MySQL数据库中创建用户记录，前端界面提示注册成功，否则前端界面显示具体错误。

用户登录用例实现的设计方案

“用户登录”功能的实现主要通过LoginView视图提供的服务，在接收到用户发送的登录请求后查询数据库中的用户数据，验证用户信息是否正确，判定用户身份是否合法。详见图 2 所描述的用例实现顺序图。

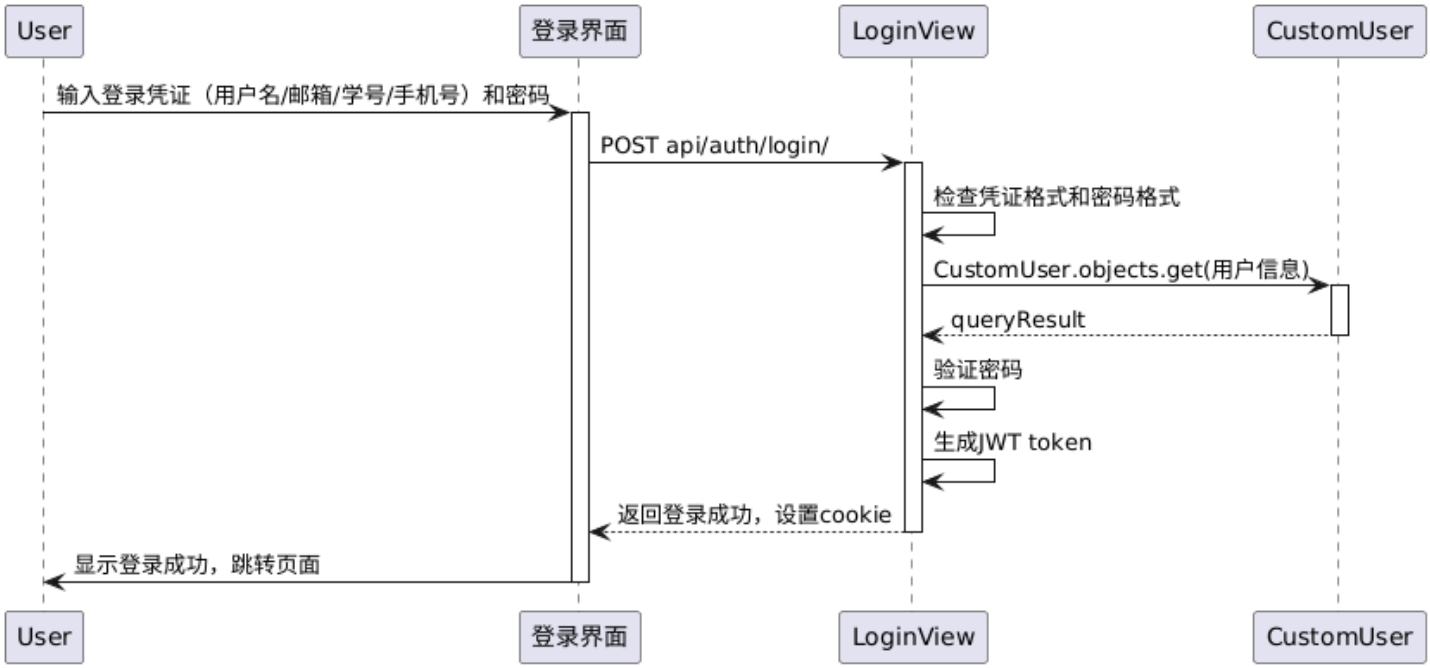


图2. 用户登录用例设计的顺序图

- 用户进入登录界面，输入登陆凭证和密码并向后端的api/auth/login/请求登录。
- LoginView收到信息后进行格式验证，并向CustomUser模型查询用户信息。
- 信息验证通过后生成JWT token并在响应中设置cookie。
- 登陆成功后将提示登陆成功、跳转到主页面并展示用户信息。

用户绑定飞书账号用例实现的设计方案

“用户绑定飞书”功能的实现主要通过FeishuBindView视图和FeishuCallbackView视图提供的服务，将用户重定向至飞书授权页面，完成绑定后再跳转回当前页面，并保存用户飞书相关的信息。详见图 3 所描述的用例实现顺序图。

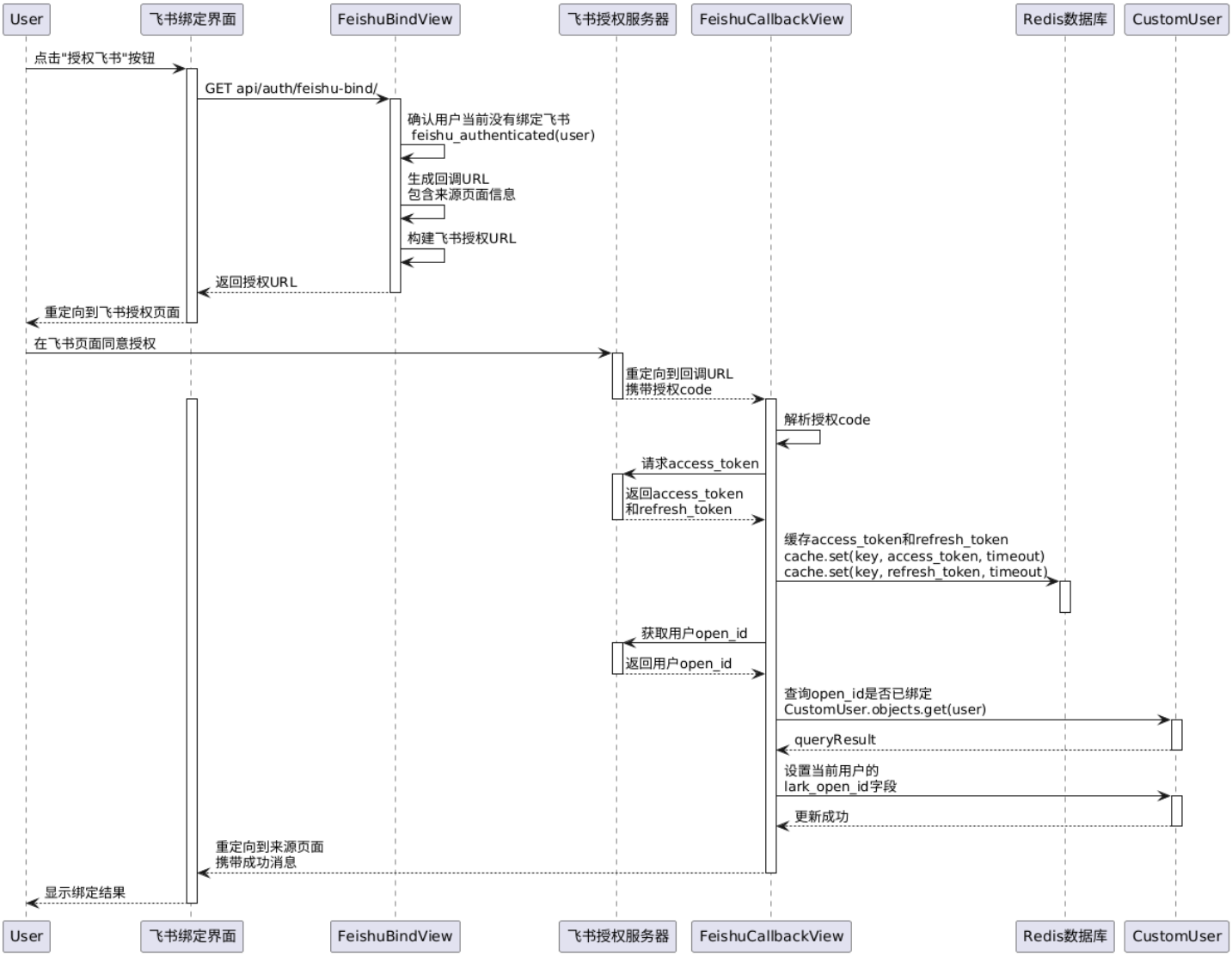


图3. 用户绑定飞书账号用例设计的顺序图

- 用户在支队长手册界面点击授权飞书按钮，向FeishuBindView发送请求。
- FeishuBindView接收到请求后调用feishu_authenticated()函数验证用户是否已经绑定过飞书。
- 若用户未绑定飞书则生成包含回调URL和来源页面URL的飞书授权URL返回至前端。
- 用户被重定向到飞书授权界面进行授权。
- 授权成功后跳转到回调URL即FeishuCallbackView视图。
- FeishuCallbackView视图解析授权码code，并使用该授权码向飞书授权服务器请求获取用户的token。
- 获取用户的access_token和refresh_token并将其存入redis缓存，使用token再次向飞书授权服务器请求获取用户的信息。
- 获取用户的open_id后将其存入数据库中已有的用户数据的lark_open_id字段。
- 成功绑定后将提示用户成功绑定飞书。

超级管理员修改用户权限用例实现的设计方案

“修改用户权限”功能的实现主要是通过ModifyPermissionView视图提供的服务，在验证超级管理员权限后允许用户修改任意用户的权限。详见图 4 所描述的用例实现顺序图。

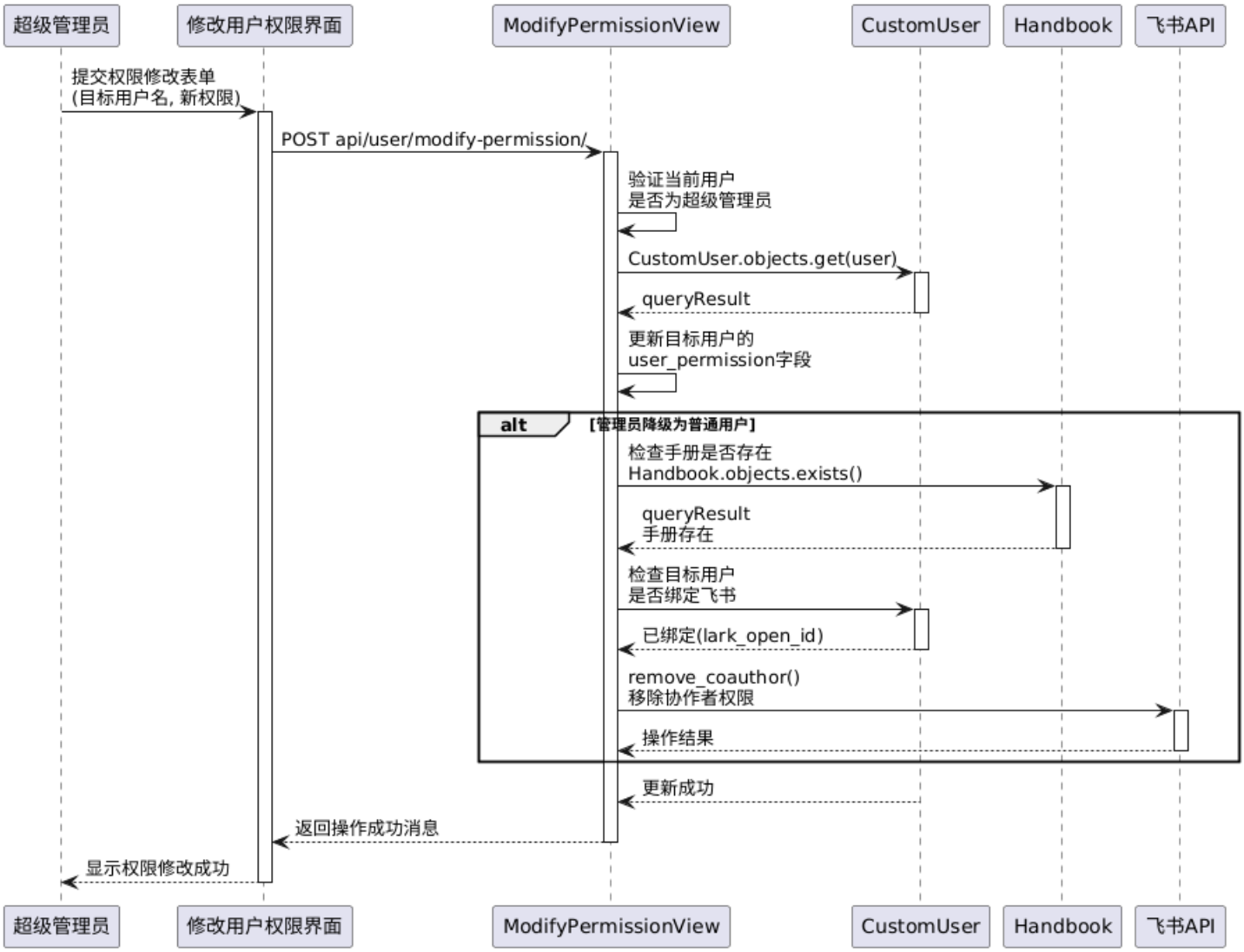


图4. 修改用户权限用例设计的顺序图

- 用户进入用户权限管理界面，选定用户及新的权限后点击修改权限按钮向ModifyPermissionView发送请求。
- ModifyPermissionView接到请求后首先检查用户是否为超级管理员，通过检查后从数据库中找出对应的用户实例，将其修改为对应的权限。
- 如果目标用户从管理员降为普通用户，则到数据库中查询支队长手册是否存在。
- 若支队长手册存在且用户已经绑定飞书账号，则调用remove_coauthor()移出该用户对支队长手册的协作者权限。
- 成功调整权限后向前端返回操作成功消息，用户将看到成功修改权限的提示。

创建支队用例实现的设计方案

“创建支队”功能的实现主要是通过CreateDetachmentView视图提供的服务，可以根据用户传入的支队信息创建支队。详见图 5 所描述的用例实现顺序图。

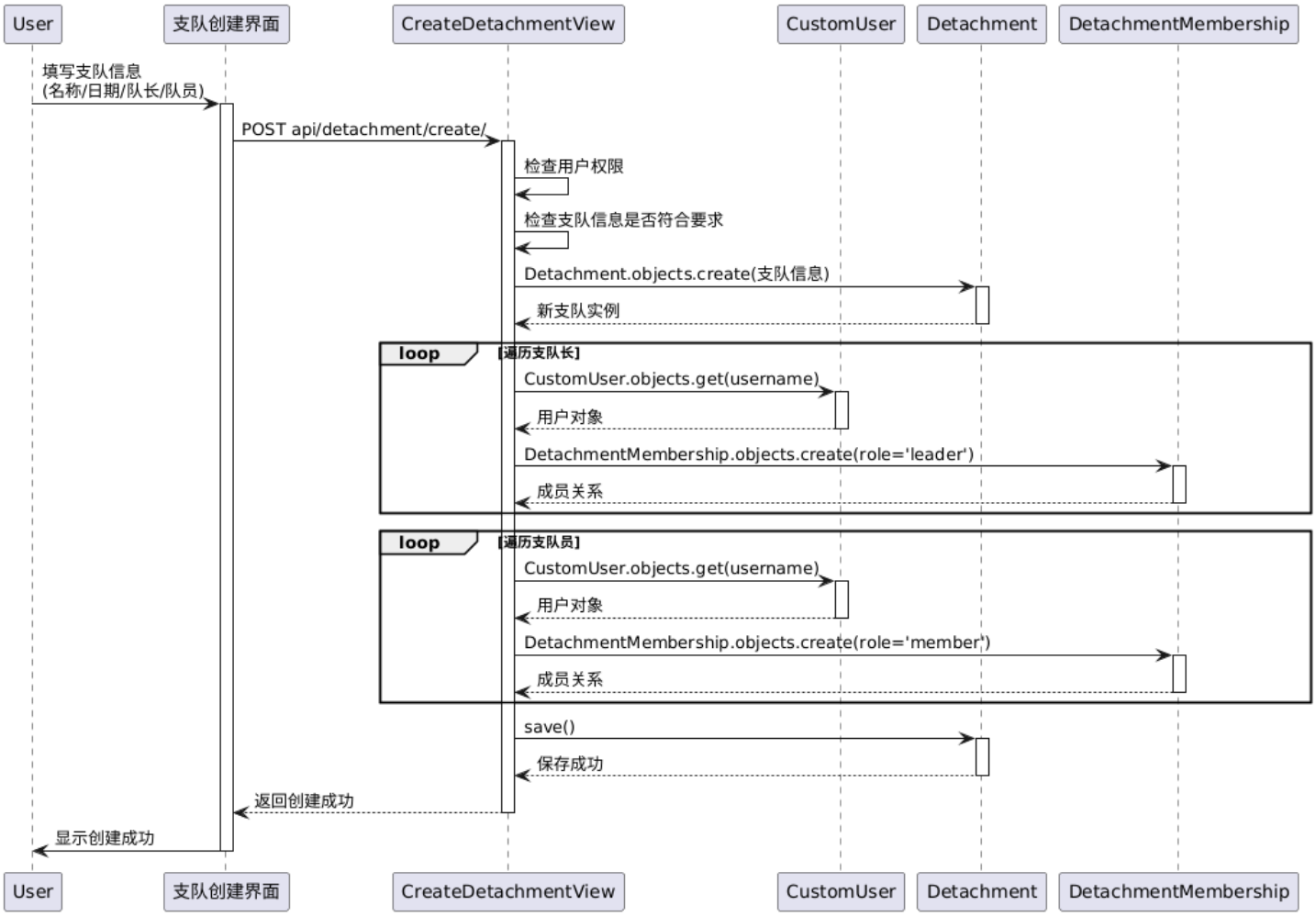


图5. 创建支队用例设计的顺序图

- 用户进入支队创建界面，填写支队信息，点击“提交”按钮后向CreateDetachmentView发送请求。
- CreateDetachmentView收到请求后首先检查用户是否为管理员，接着检查用户输入的支队信息是否合法。
- 通过检查后在数据库中根据信息创建新的Detachment实例。
- 依次遍历支队长和支队员，通过用户名找到用户实例后创建DetachmentMembership实例，记录支队人员。
- 保存实例并向前端返回创建成功，用户将看到成功创建支队提示。

修改支队信息用例实现的设计方案

“修改支队信息”功能的实现主要是通过ModifyDetachmentView视图提供的服务。管理员或支队长可以提交新的支队信息，在通过检查后更新已有的支队信息。详见图 6 所描述的用例实现顺序图。

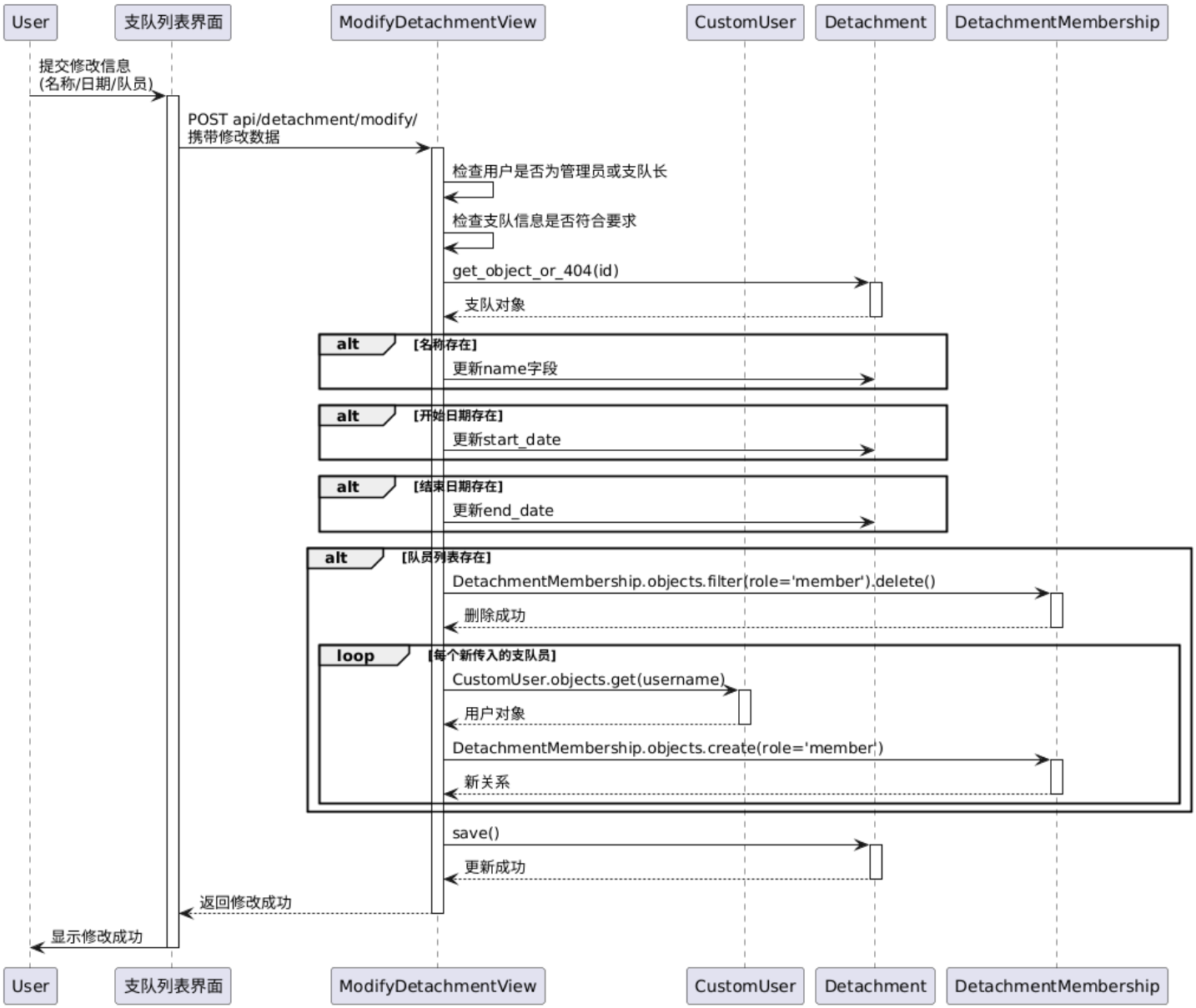


图6. 修改支队信息用例设计的顺序图

- 用户进入支队列表界面点击“修改”按钮填入新的支队信息，点击提交按钮后向ModifyDetachmentView发送请求。
- ModifyDetachmentView收到请求后首先检查用户是否为管理员或支队长，接着检查用户输入的支队信息是否合法。
- 通过get_object_or_404()方法获取对应的Detachment实例，逐个字段对实例进行更新。
- 在更新支队员时首先通过delete()方法删除原先的全部支队员，并加入新的支队员。
- 实例保存后向前端返回修改成功，用户将看到成功修改支队信息提示。

停用支队用例实现的设计方案

“停用支队”功能的实现主要是通过DeactivateDetachmentView视图提供的服务。查询到用户请求停用的支队后将其valid字段设为False。详见图 7 所描述的用例实现顺序图。

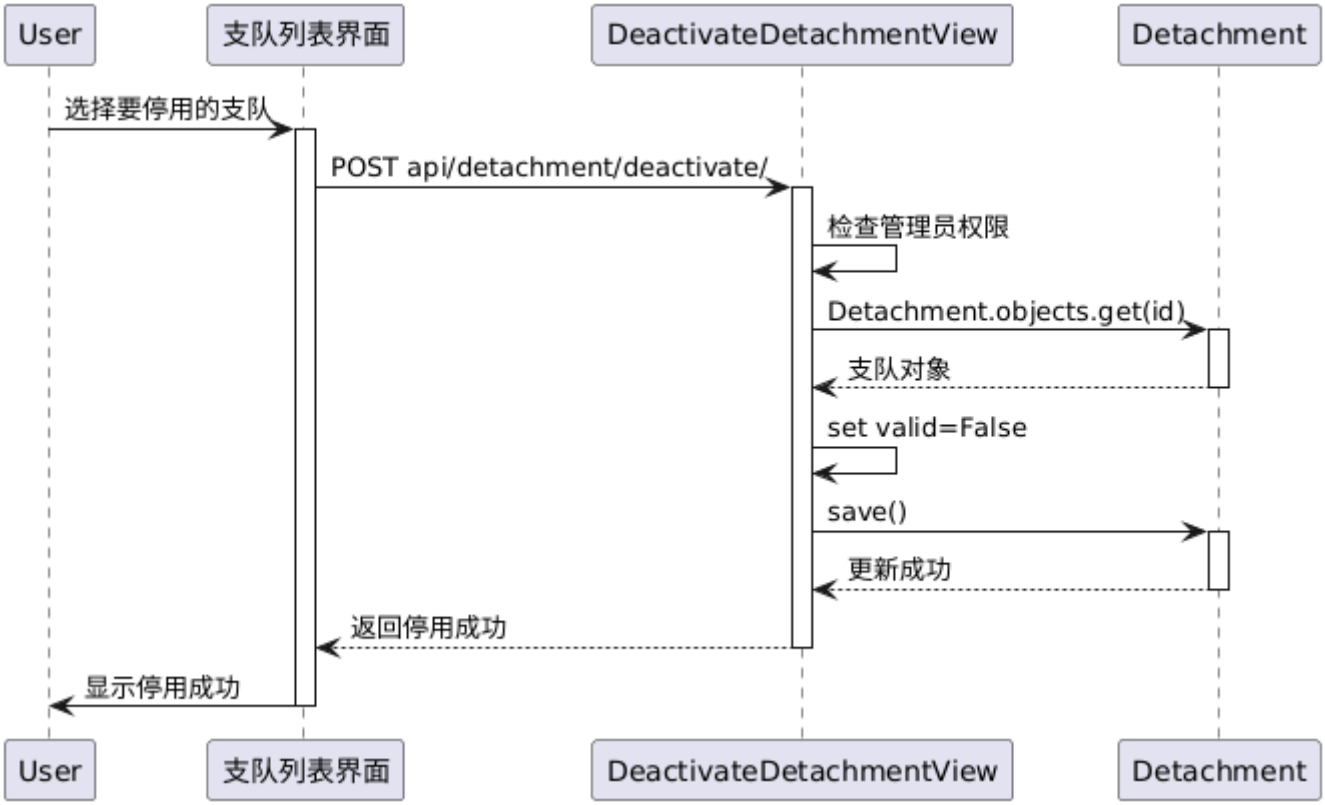


图7. 停用支队用例设计的顺序图

- 用户进入支队列表界面点击“停用”按钮，向DeactivateDetachmentView视图发送请求。
- DeactivateDetachmentView收到请求后首先检查管理员权限。
- 使用Detachment.objects.get()方法从数据库中获取Detachment实例，将valid字段修改为False并保存。
- 向前端返回停用成功，用户将看到停用成功的提示。

删除支队用例实现的设计方案

“删除支队”功能的实现主要是通过DeleteDetachmentView视图提供的服务。接收到用户的请求并确认权限后将数据库中对应的Detachment实例删除。详见图 8 所描述的用例实现顺序图。

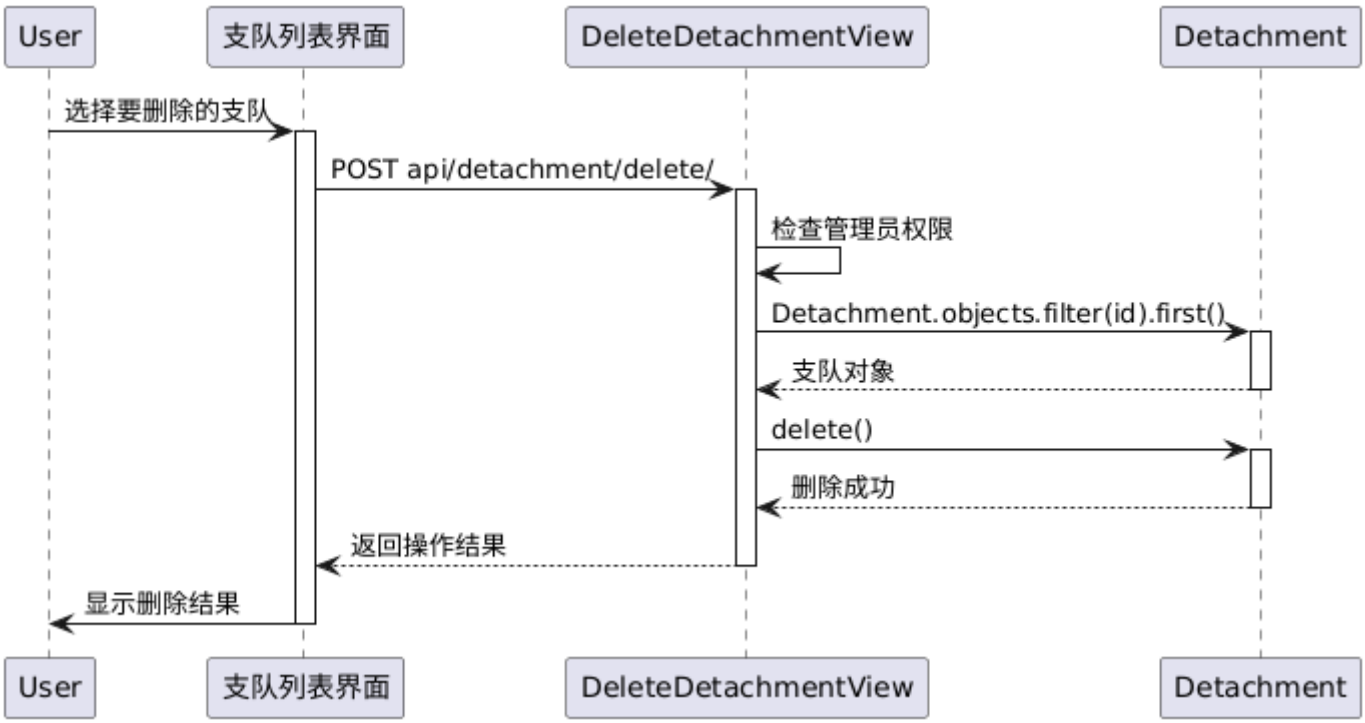


图8. 删除支队用例设计的顺序图

- 用户进入支队列表界面点击“删除”按钮，向DeleteDetachmentView视图发送请求。
- DeleteDetachmentView收到请求后首先检查用户是否为管理员。
- 使用Detachment.objects.filter(id).first()方法从数据库中获取Detachment实例，并通过delete()方法将其删除。
- 向前端返回删除成功，用户将看到删除成功的提示。

发布公告用例实现的设计方案

“发布公告”功能的实现主要是通过SendNoticeView视图提供的服务。接收到请求并确认权限后使用序列化器验证数据并创建Notice实例，并添加UserNotice实例记录用户与通知的对应关系，最后使用邮件系统为收到通知的人发送邮件。详见图 9 所描述的用例实现顺序图。

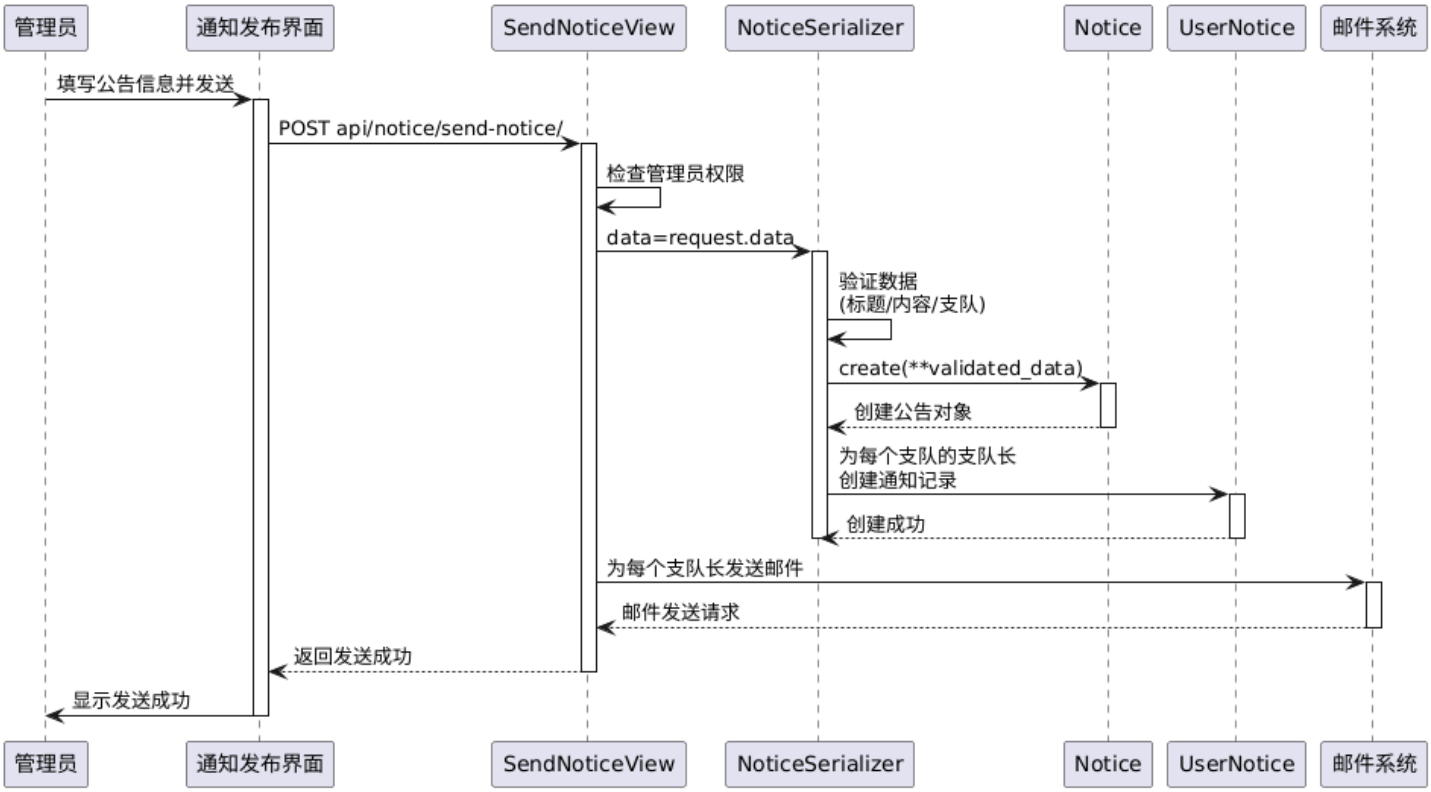


图9. 发布公告用例设计的顺序图

- 用户进入通知发布界面，填入公告的信息后向SendNoticeView发送请求。
- SendNoticeVeiw接收到请求后首先检查用户是否为管理员。
- 将request.data传入NoticeSerializer进行数据格式的验证，验证通过后通过create()方法在数据库中创建Notice实例。
- 为每一个接受通知的用户创建UserNotice实例，用于记录用户的通知确认状态。
- SendNoticeView再为每个接收到统治的用户发送邮件通知。
- 向前端返回发送成功，用户将看到发送成功的提示。

确认公告用例实现的设计方案

“确认公告”功能的实现主要是通过ConfirmView视图提供的服务。接收到请求后将对应的通知设为接收状态。详见图 10 所描述的用例实现顺序图。

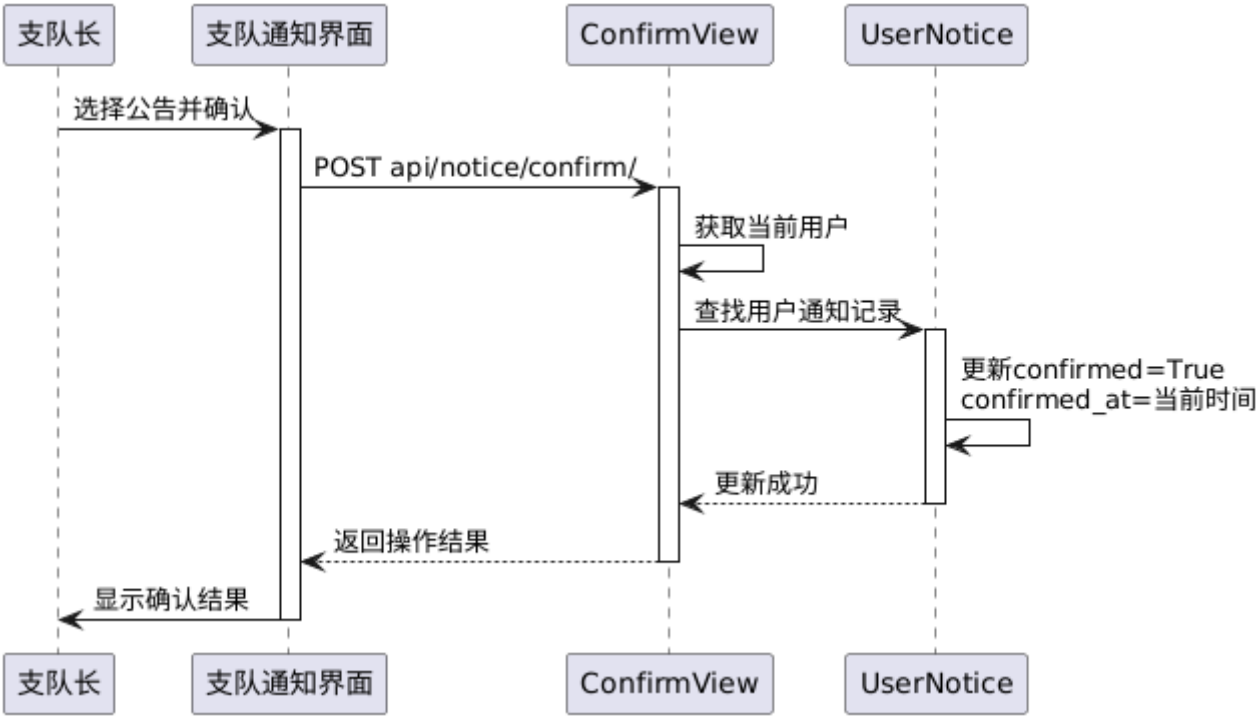


图10. 确认公告用例设计的顺序图

- 用户进入支队通知界面，即可查看所有收到的通知，选择需要确认的通知后向ConfirmView发送请求。
- ConfirmView收到请求后根据发来的request获取当前的用户。
- 根据用户和通知的id到数据库中的UserNotice中查找对应的通知记录，并将其confirmed字段置为True，confirmed_at字段设为当前时间，调用save()方法保存到数据库中。
- 向前端返回成功确认通知，用户将看到通知状态变为已确认。

创建支队长手册用例实现的设计方案

“创建支队长手册”功能的实现主要是通过CreateHandbookView视图提供的服务。收到请求后检查用户的权限及飞书绑定状态，然后调用飞书API进行文档创建并添加协作者权限。详见图 11 所描述的用例实现顺序图。

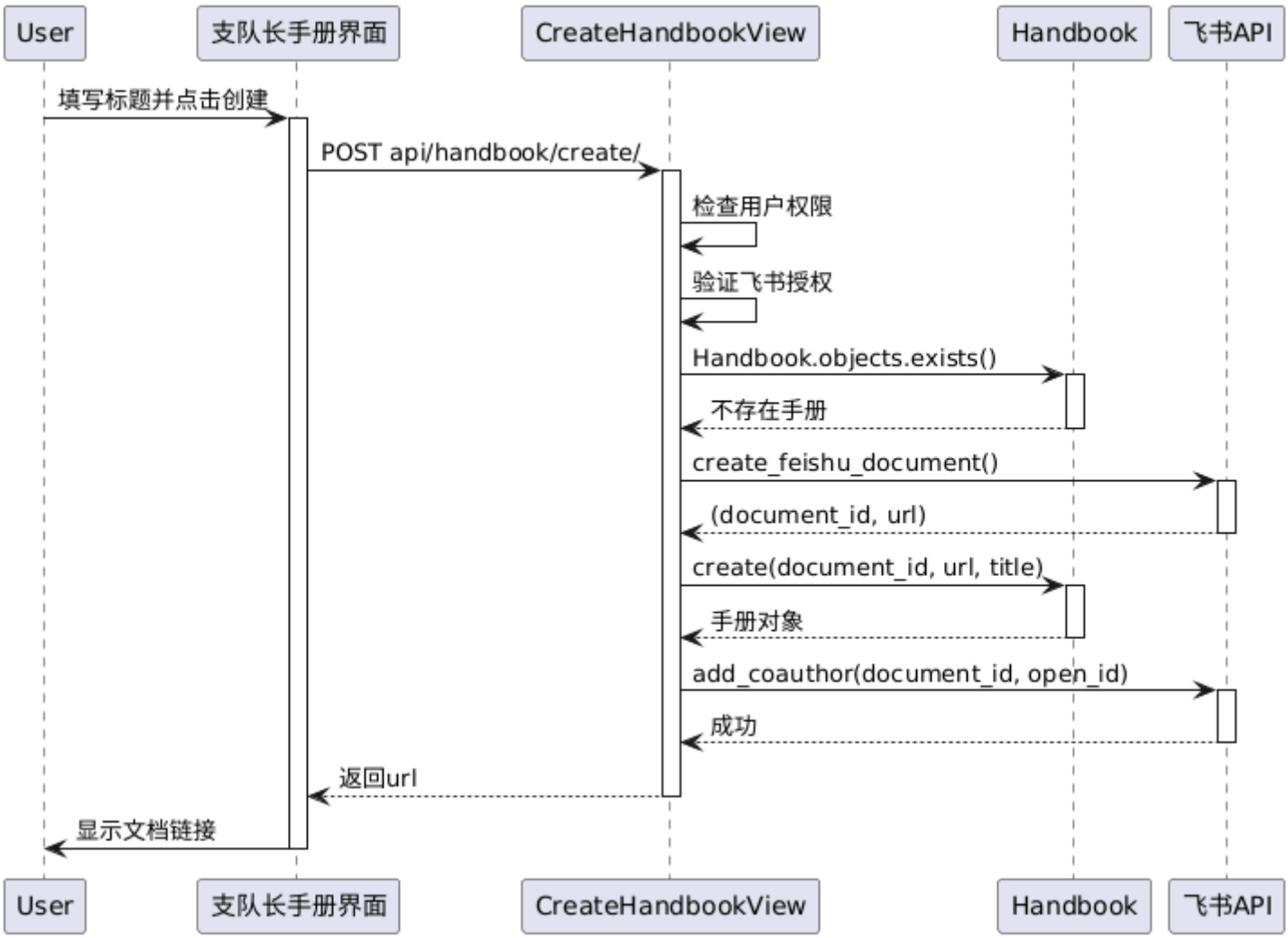


图11. 创建支队长手册用例设计的顺序图

- 用户进入支队长手册界面，点击“创建支队长手册”按钮向CreateHandbookView发送请求。
- CreateHandbookView接收到请求后首先确认用户权限及飞书授权状态。
- 查询数据库中Handbook是否已经存在实例，确保只有一个Handbook实例。
- 通过create_feishu_document()函数调用飞书API创建飞书文档，得到文档的URL及document_id。
- 通过Handbook.objects.create(document_id, url, title)方法创建Handbook实例。
- 通过add_coauthor()函数调用飞书API，将创建文档的用户添加为文档的协作者。
- 向前端返回新创建的飞书文档的URL，用户将在支队长手册界面看到新创建的飞书文档。

修改支队长手册标题用例实现的设计方案

“修改支队长手册标题”功能的实现主要是通过ModifyTitleView视图提供的服务。收到请求后确认用户权限，将对应的Handbook实例的title字段修改为新标题。详见图 12 所描述的用例实现顺序图。

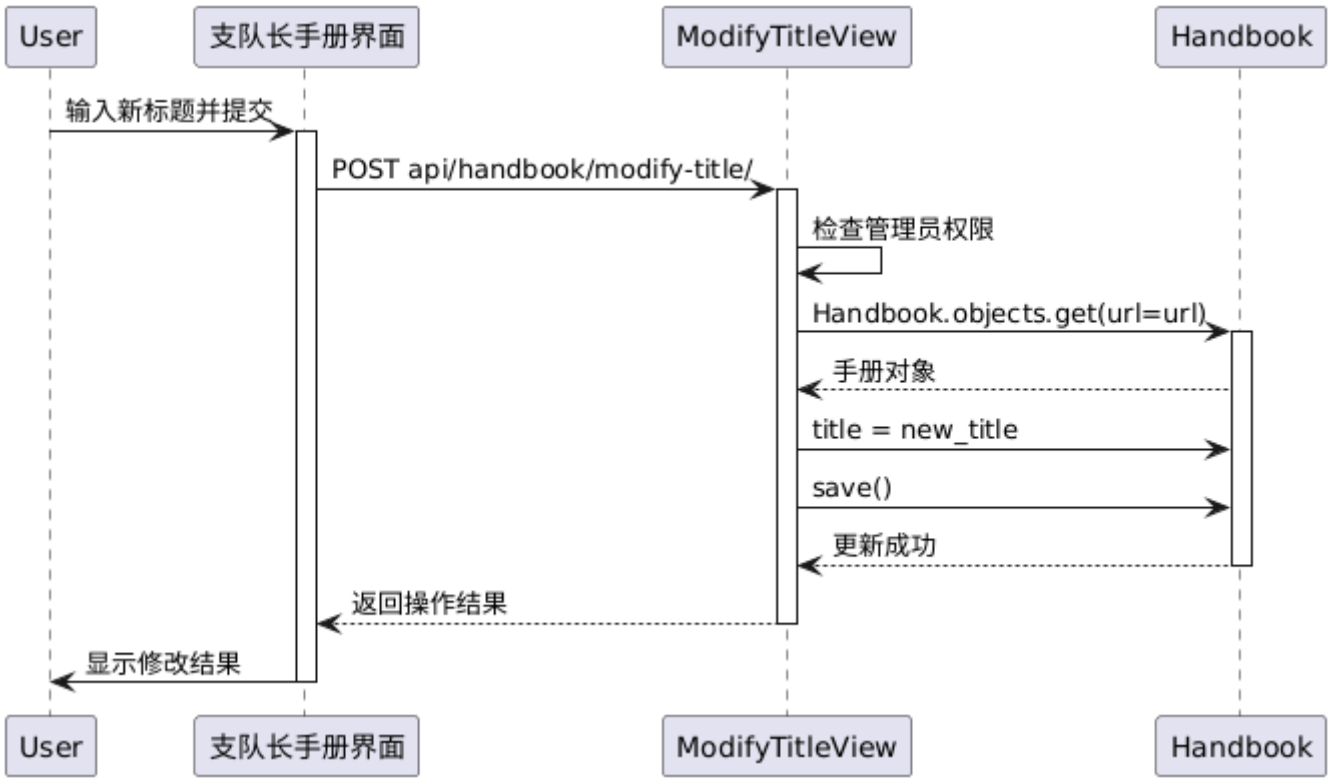


图12. 修改支队长手册标题用例设计的顺序图

- 用户进入支队长手册界面，点击“修改标题”按钮并输入新的标题，向ModifyTitleView发送请求。
- ModifyTitleView收到请求后确认用户的管理员权限。
- 通过Handbook.objects.get(url=utl)从数据库中获取对应的Handbook实例。
- 将Handbook实例的title字段修改为新标题并调用save()方法保存到数据库中。
- 向前端返回成功修改标题，用户将在支队长手册中看到新的标题。

启用编辑用例实现的设计方案

“获取支队长手册编辑权限”功能的实现主要是通过AddCoauthorView视图提供的服务。收到请求后向通过飞书API为用户获取协作者权限。详见图 13 所描述的用例实现顺序图。

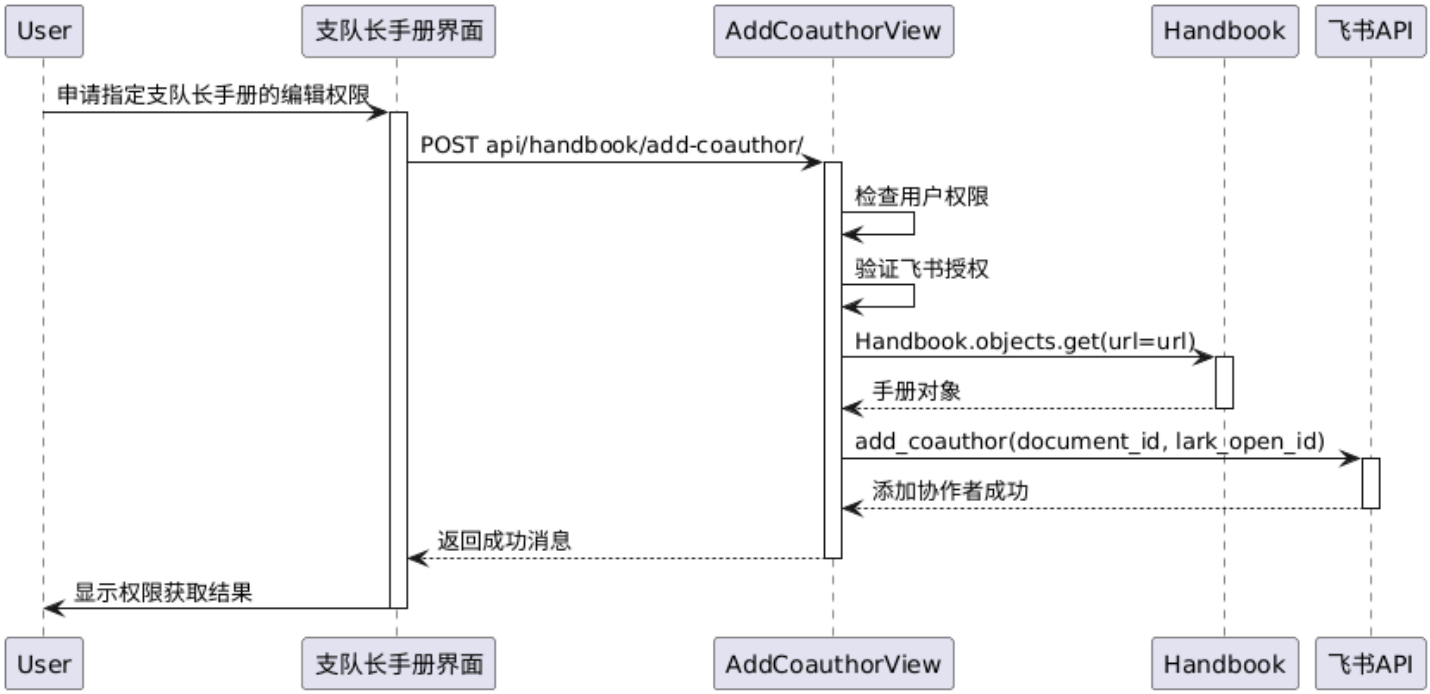


图13. 获取支队长手册编辑权限标题用例设计的顺序图

- 用户进入支队长手册界面，点击“启用编辑”按钮向AddCoauthorView发送请求。
- AddCoauthorView收到请求后检查用户权限及飞书授权状态。
- 通过Handbook.objects.get()方法获取数据库中的Handbook实例，并通过add_coauthor()方法调用飞书API将当前用户设为文档的协作者。
- 向前端返回成功添加协作者，用户再次进入文档时拥有编辑权限。

删除支队长手册用例实现的设计方案

“删除支队长手册”功能的实现主要是通过DeleteHandbookView视图提供的服务。接收到用户请求后同时在数据库中和飞书云空间中删除文档。详见图 14 所描述的用例实现顺序图。

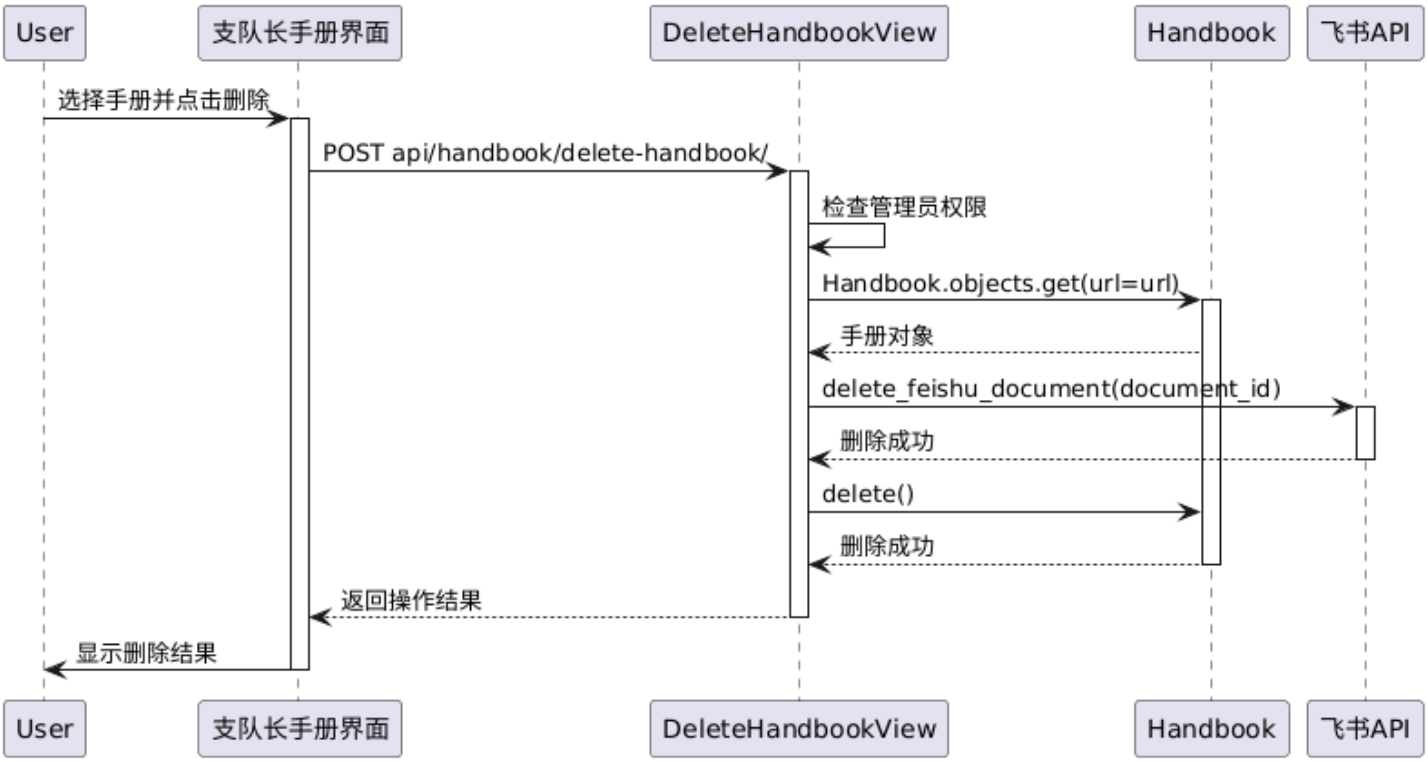


图14. 删除支队长手册用例设计的顺序图

- 用户进入支队长手册界面，点击“删除”按钮向DeleteHandbookView发送请求。
- DeleteHandbookView接收到请求后检查用户权限是否为管理员。
- 通过Handbook.objects.get()方法获取数据库中的Handbook实例，并通过delete_feishu_document()方法调用飞书API将云文档删除。
- 通过delete()方法删除数据库中的Handbook实例。
- 向前端返回成功删除文档，用户界面将不再显示被删除的文档。

创建问卷用例实现的设计方案

“创建问卷”功能的实现主要是通过CreateQuestionaireView视图提供的服务。接收到用户发来的问卷信息后通过序列化器验证数据格式并创建Questionaire实例。详见图 15 所描述的用例实现顺序图。

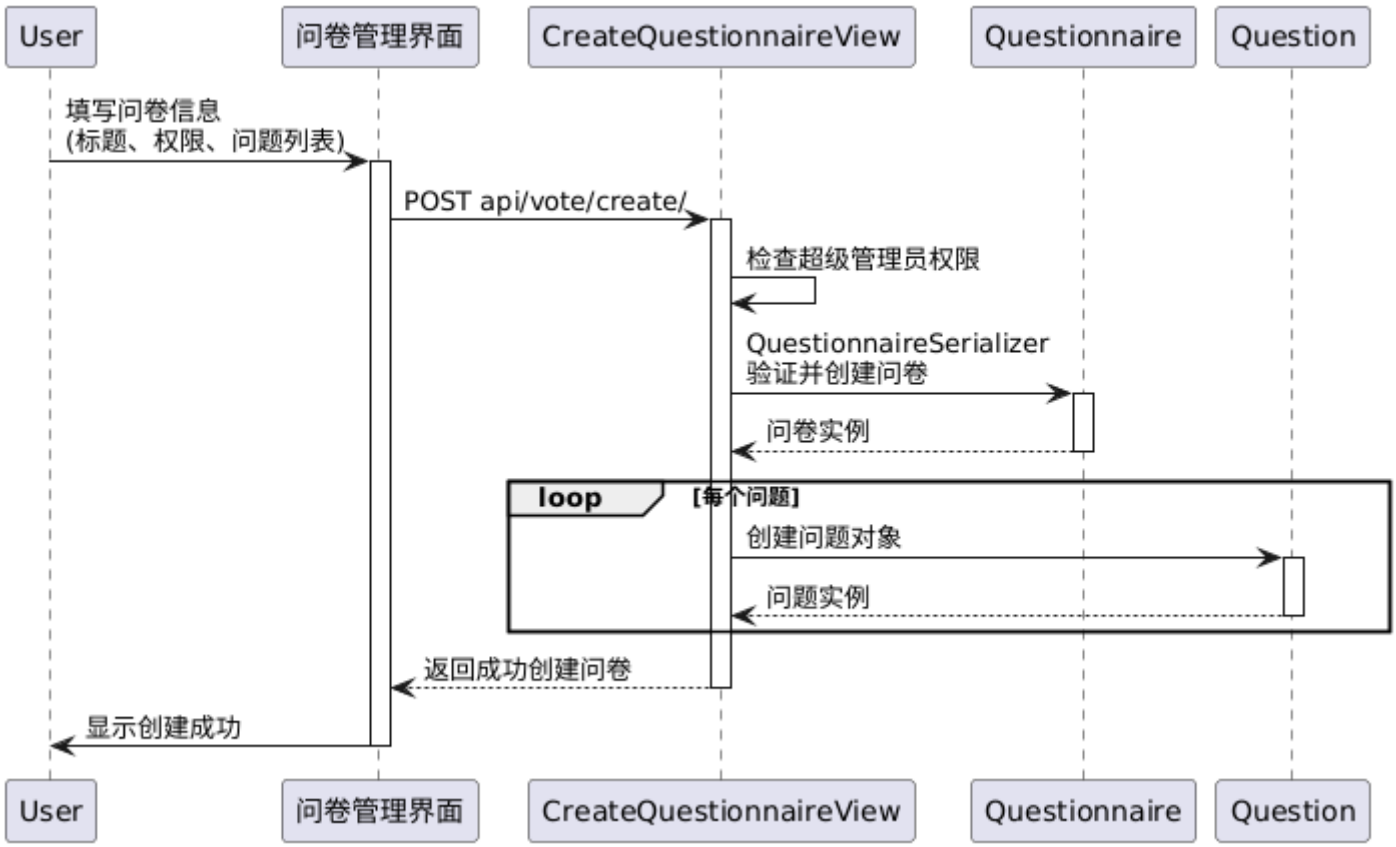


图15. 创建问卷用例设计的顺序图

- 用户进入问卷管理界面，点击“创建新问卷”按钮并填写好问卷内容后向CreateQuestionView视图发送请求。
- CreateQuestionView接收到请求后检查用户的超级管理员权限。
- 使用QuestionaireSerializer验证数据并通过create()方法将其存入数据库。
- 遍历问卷中的每一道问题，为其创建Question实例。
- 向前端返回成功创建问卷，用户将在问卷管理界面中看到新创建的问卷。

发布问卷用例实现的设计方案

“发布问卷”功能的实现主要是通过PublishQuestionaireView视图提供的服务。接收到请求后将对应的Questionaire实例is_published字段设为True。详见图 16 所描述的用例实现顺序图。

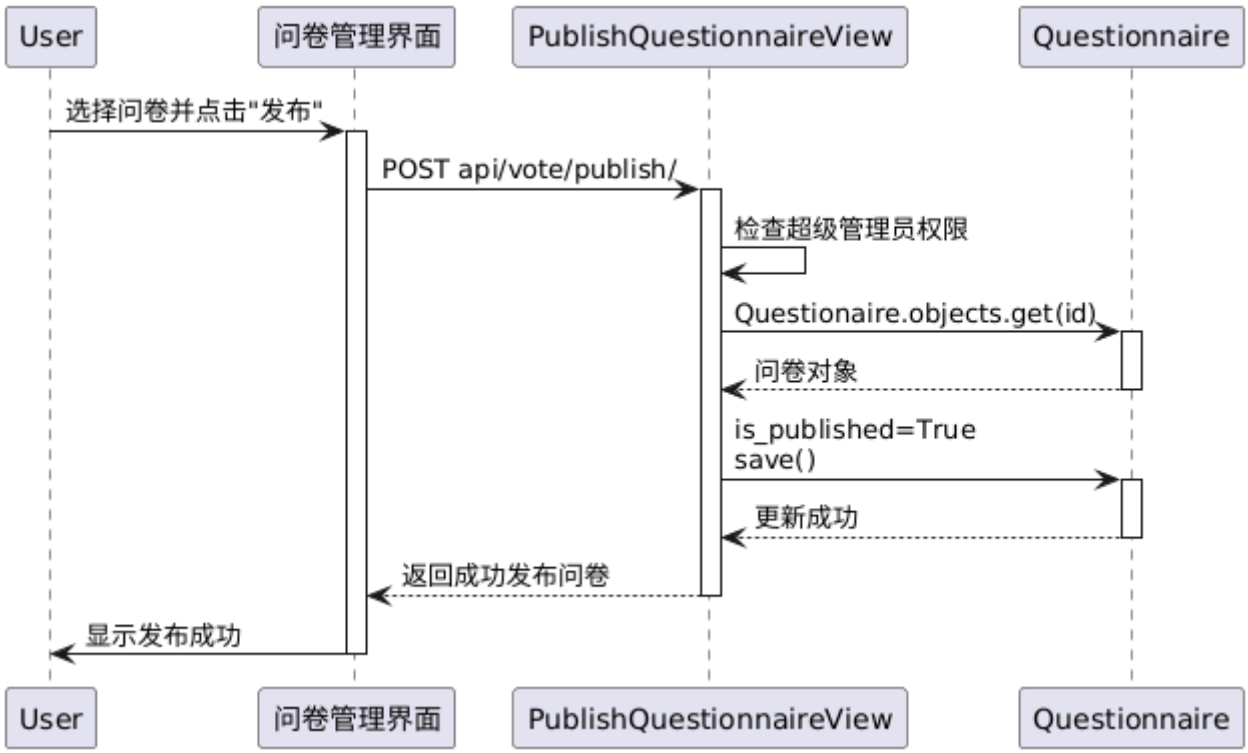


图16. 发布问卷用例设计的顺序图

- 用户进入问卷管理界面，点击“发布问卷”按钮向PublishQuestionaireView视图发送请求。
- PublishQuestionaireView接收到请求后检查用户超级管理员权限。
- 使用get()方法从数据库中取出对应的Questionaire实例，将其is_published字段设为True并通过save()方法保存。
- 向前端返回成功发布问卷，用户将在问卷管理界面看到问卷已经成功发布。

截止收集问卷用例实现的设计方案

“截止收集问卷”功能的实现主要是通过CloseQuestionairView视图提供的服务。接收到请求后将对应的Questionaire实例is_closed字段设为True。详见图 17 所描述的用例实现顺序图。

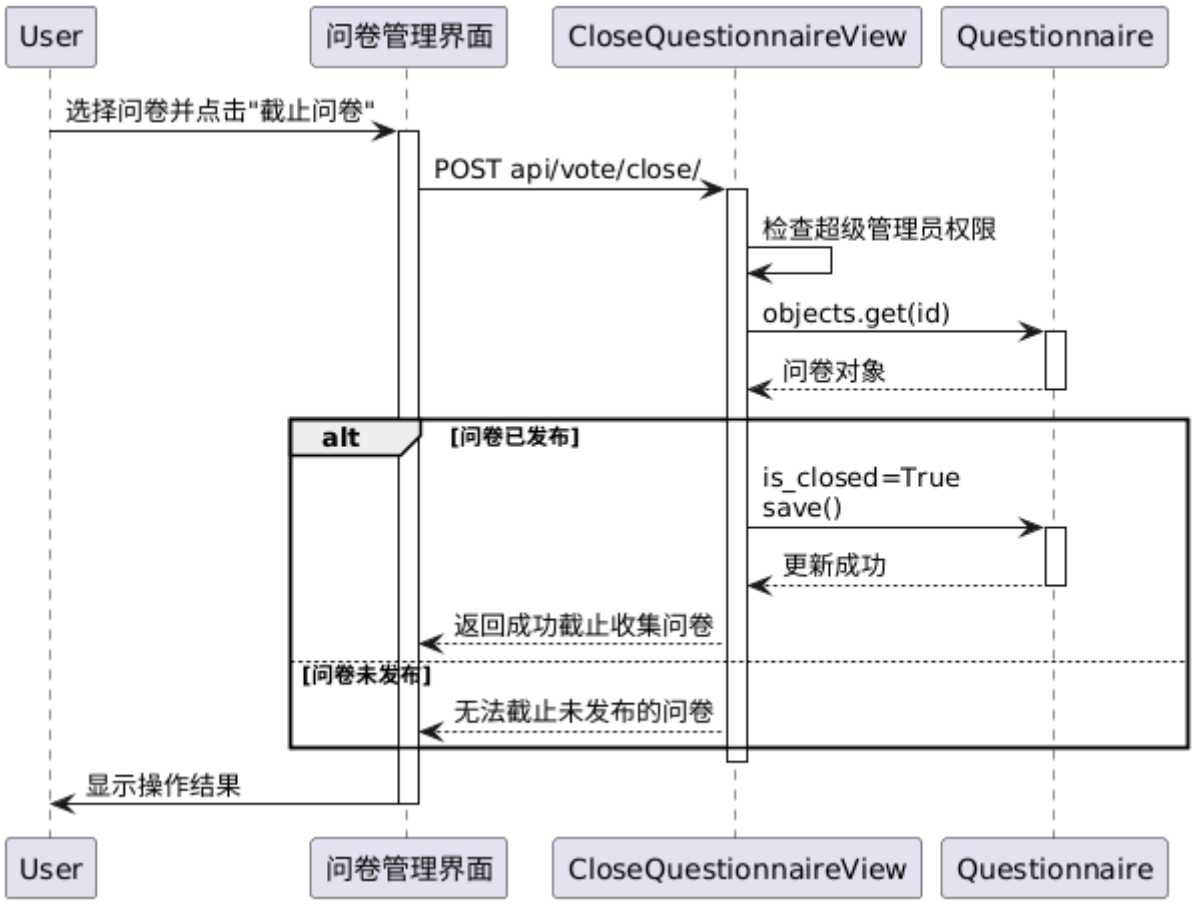


图17. 截止收集问卷用例设计的顺序图

- 用户进入问卷管理界面，点击“截止问卷”按钮向CloseQuestionaireView视图发送请求。
- CloseQuestionaireView接收到请求后检查用户超级管理员权限。
- 使用get()方法从数据库中取出对应的Questionaire实例，若该问卷已发布将其is_closed字段设为True并通过save()方法保存，向前端返回成功截止收集问卷。
- 若该问卷未发布则向前端返回无法截止未发布的问卷。
- 用户将在问卷管理界面看到操作后的结果。

填写问卷用例实现的设计方案

“填写问卷”功能的实现主要是通过SubmitAnswerView视图提供的服务。接收到用户的答案后查询Questionaire实例，并为每个Question实例创建对应的Answer实例。详见图 18 所描述的用例实现顺序图。

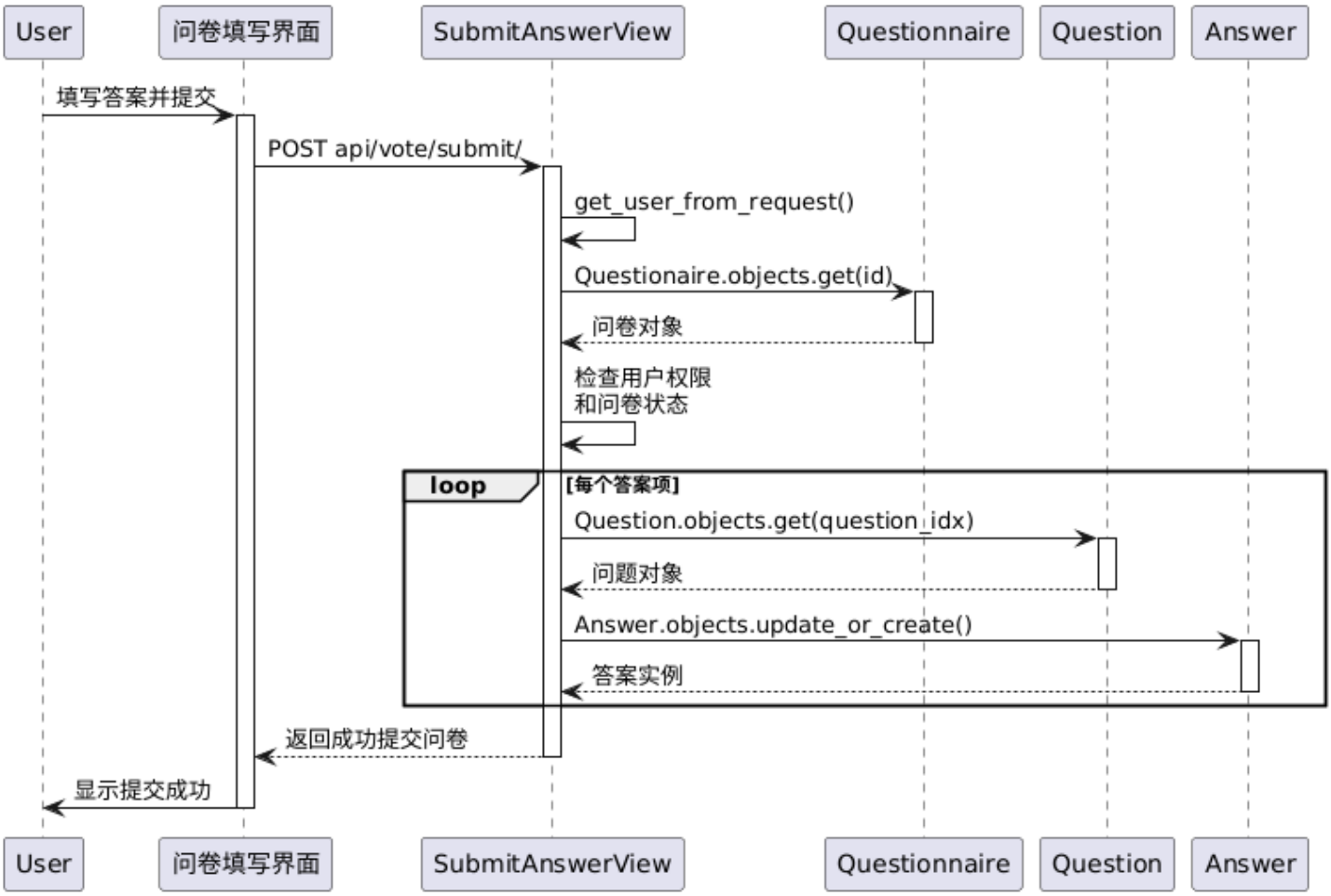


图18. 截止收集问卷用例设计的顺序图

- 用户进入问卷填写界面，点击“填写”按钮填入答案后点击“提交”按钮，向SubmitAnswerView视图发送请求。
- SubmitAnswerView收到请求后通过get_user_from_request()方法获取CustomUser实例。
- 使用get()方法获取Questionaire对象，并检查用户权限是否在问卷权限列表内，以及问卷是否处于发布状态。
- 遍历每个答案项，使用get()方法获取Question实例，并通过update_or_create()方法创建或更新Answer实例。
- 向前端返回成功提交问卷，管理员可以看到新提交的答案记录。

删除问卷用例实现的设计方案

“删除问卷”功能的实现主要是通过DeleteQuestionaireView视图提供的服务。接收到请求后到数据库中删除对应的记录。详见图 19 所描述的用例实现顺序图。

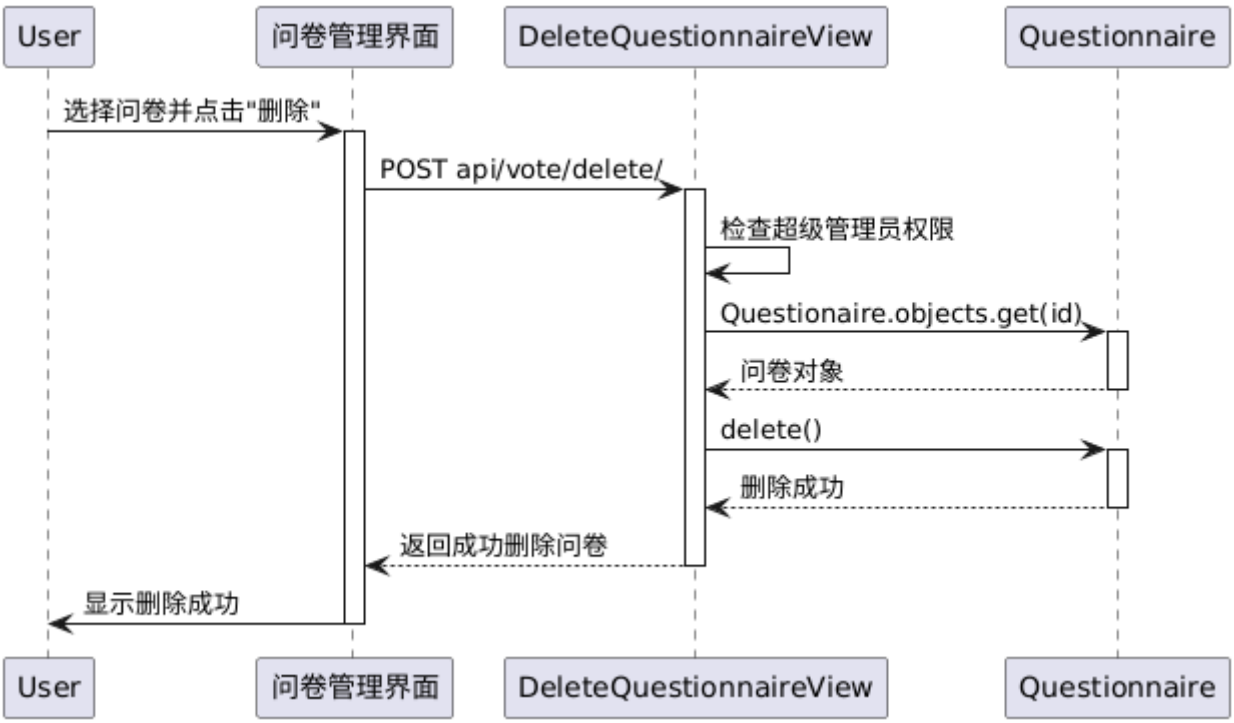


图19. 删除问卷用例设计的顺序图

- 用户进入问卷管理界面，选择问卷并点击“删除”按钮，向DeleteQuestionView视图发送请求。
- DeleteQuestionView收到请求后检查用户是否为超级管理员。
- 使用get()方法获取对应的Questionaire实例，并使用delete()方法删除问卷。
- 向前端返回成功删除问卷，用户将无法再看到删除的问卷。

查看问卷结果用例实现的设计方案

“查看问卷结果”功能的实现主要是通过QuestionaireResultView视图提供的服务。接收到请求后查找问卷实例及对应的答案实例，整理后返回给前端。详见图 20 所描述的用例实现顺序图。

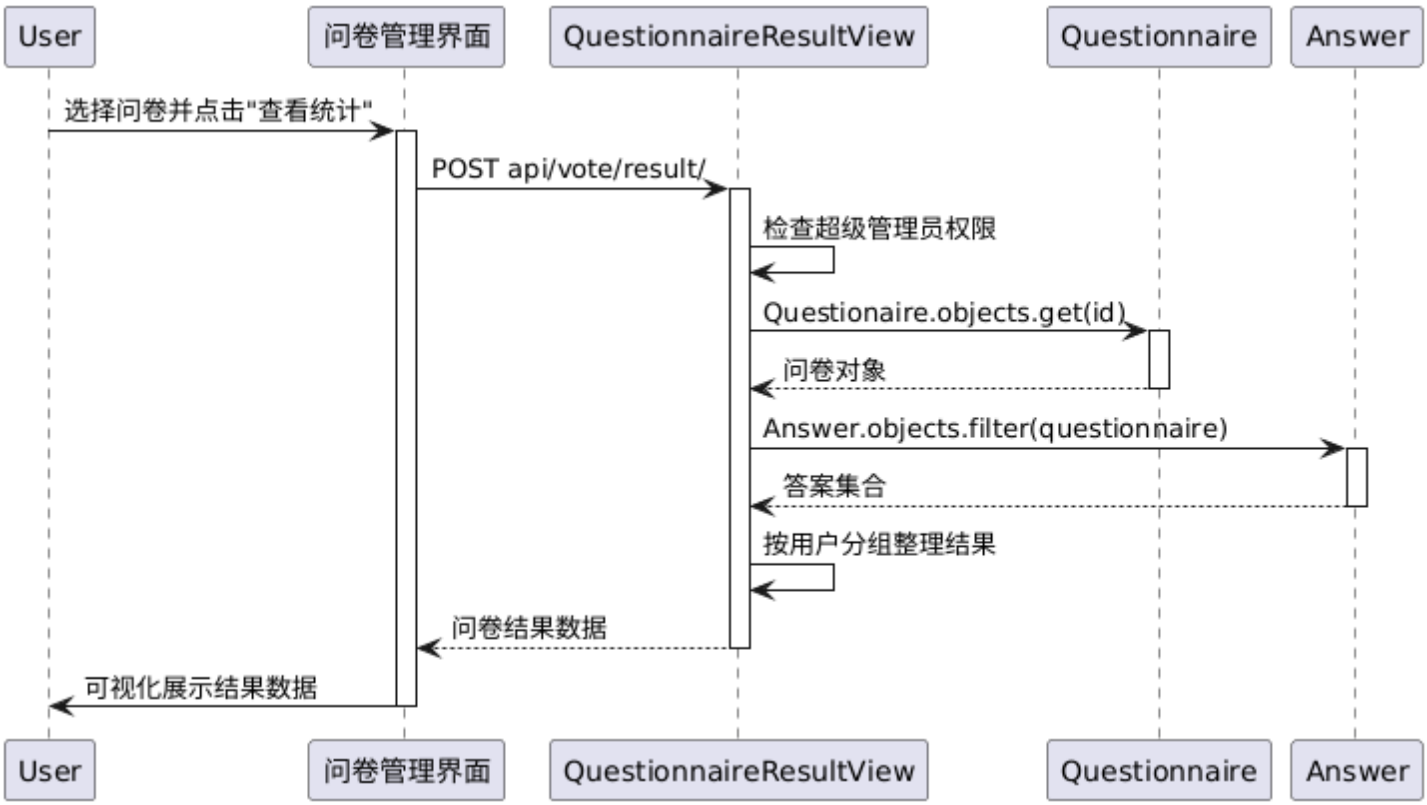


图20. 获取问卷结果用例设计的顺序图

- 用户进入问卷管理界面，点击“查看统计”按钮，向QuestionaireResultView视图发送请求。
- QuestionaireResultView收到请求后检查用户是否为超级管理员。
- 通过get()方法获取对应的Questionaire实例。
- 通过filter()方法获取该Questionaire实例的所有Answer实例。
- 将Answer实例按照用户进行归类整理，得到每个用户提交的答案。
- 将结果返回给前端，用户将实时地看到问卷的结果及可视化展示。

上传公函模板用例实现的设计方案

上传公函模板使用UploadTemplateView视图。管理员可以上传word文件或pdf文件作为公函模板。详见图21所描述的用例实现顺序图。

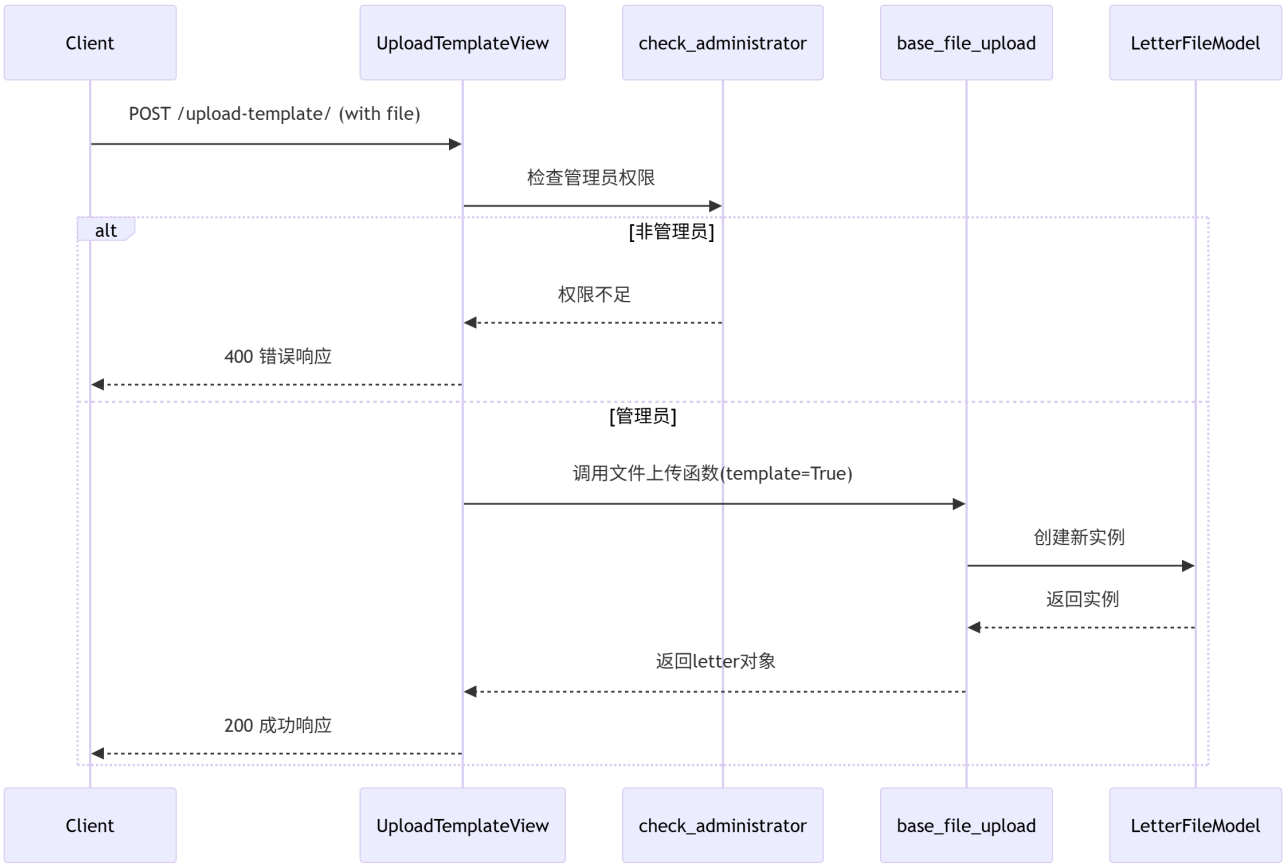


图21.上传公函模板用例设计的顺序图

- 管理员点击“上传模板”，选择要上传的模板文件，向后端UploadTemplateView发送请求。
- UploadTemplateView接受到请求后先检查用户是不是管理员。
- 接着调用基类文件上传函数，指定该文件是模板，创建新的FileLetterModel示例并返回。
- 后端将结果返回前端，用户在前端将看到“文件上传成果”。

上传公函用例实现的设计方案

上传公函示例使用UploadLetterView视图。用户可以上传填写好的公函（word或pdf）供管理员后续处理详见图22所描述的用例实现顺序图。

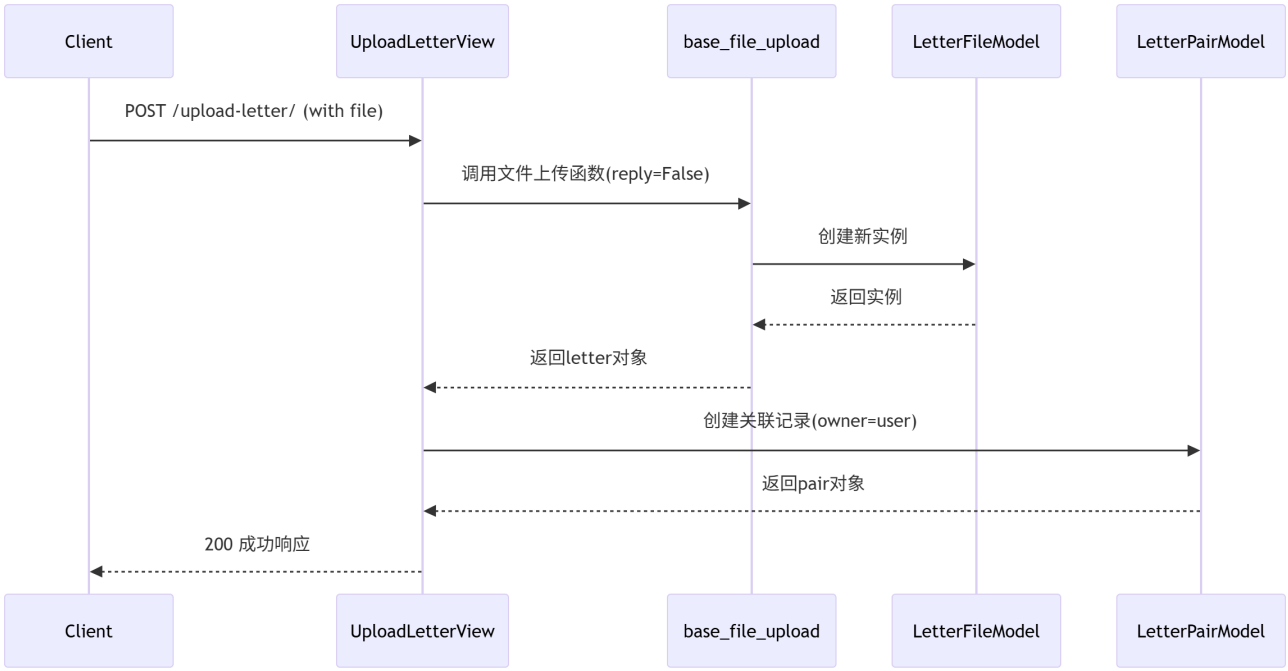


图22.上传公函用例设计的顺序图

- 用户点击“上传公函”，选择要上传的文件，向后端UploadLetterView发送请求。
- UploadLetterView接收到请求后，先调用基类文件上传函数，创建LetterFileModel的实例。
- 接着创建关联记录LetterPairModel的实例，并正常返回。
- 前端接受到后端的正常返回，界面显示“上传公函成功”。

上传公函回复用例实现的设计方案

上传公函回复使用UploadCompletedLetterView视图实现。管理员可以选择特定的公函上传处理好的公函回复。详见图23所描述的用例实现顺序图。

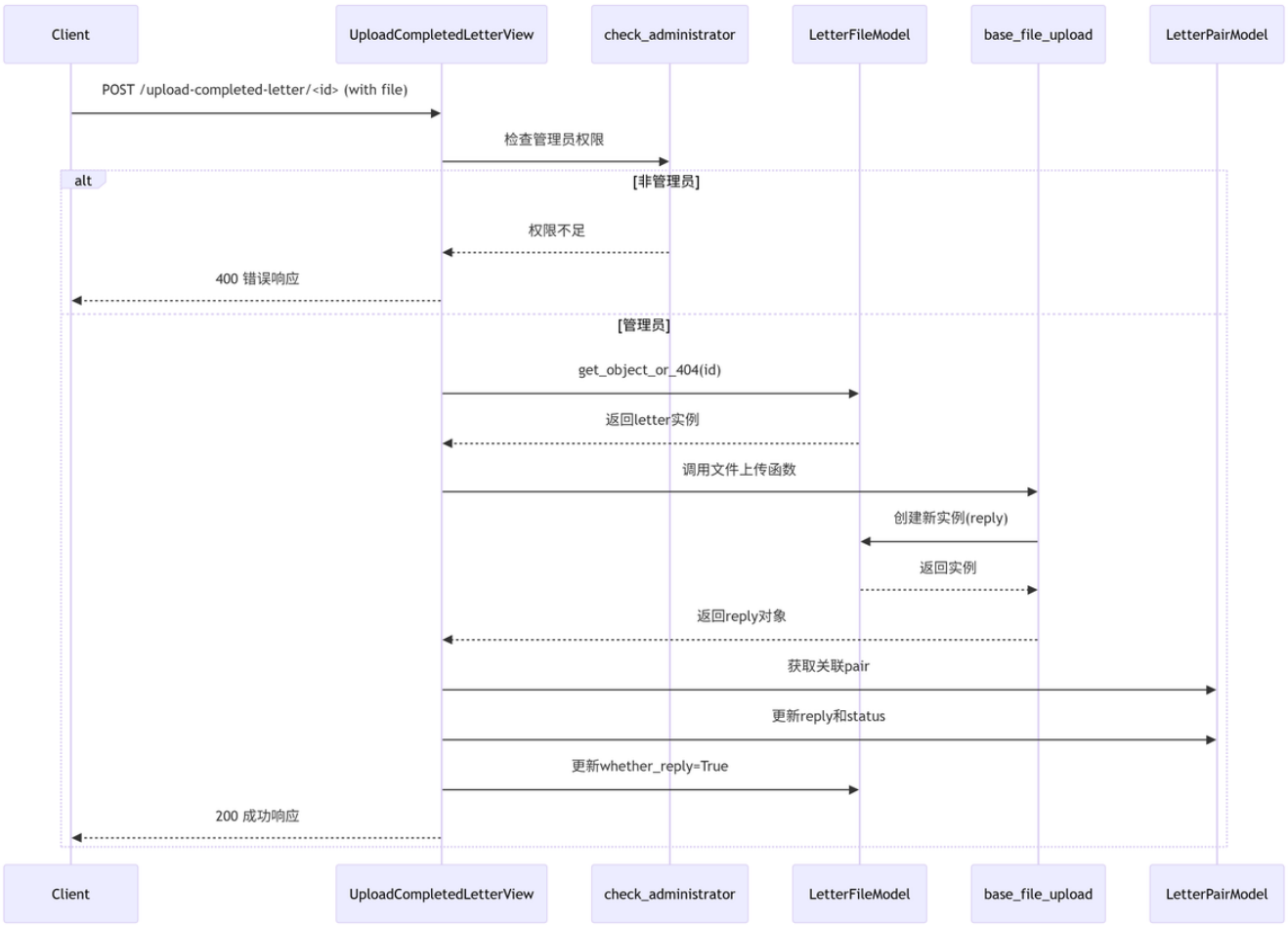


图23.上传公函回复用例设计的顺序图

- 前端选择公函，上传回复文件，向后端UploadCompletedLetterView视图发送请求。
- UploadCompletedLetterView接收请求后，先校验用户的权限是不是管理员。
- 如果是，再使用get_object_or_404尝试获取用户填写的公函。
- 如果公函存在，那么调用文件上传函数，将新上传的公函回复创建为LetterFileModel的实例并标记为reply。
- 根据公函获取LetterPair，更新reply文件为公函回复与status，正常返回。
- 前端接收到后端的返回，显示“上传成功”。

下载公函用例实现的设计方案

下载公函模板、公函、公函回复都使用DownloadView来实现。用户通过主键来选择文件并下载。详见图24所描述的用例实现顺序图。

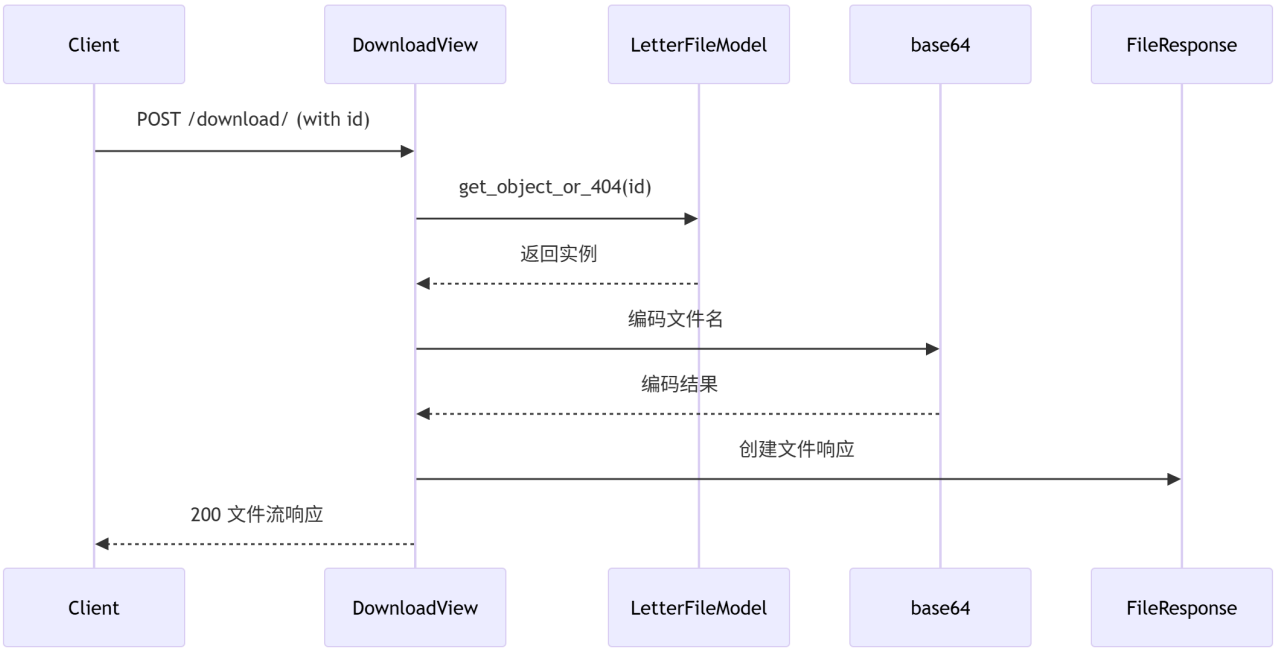


图24.下载公函用例设计的顺序图

- 前端选择文件，向后端DownloadView发送请求。
- DownloadView使用get_object_or_404来尝试获得LetterFileModel的实例。
- 如果能正常获得，则使用utf-8 base64 编码文件名，再创建文件响应。
- 响应传回前端，正常下载文件。

查询公函模板用例实现的设计方案

查询公函模板使用GetTemplateView视图实现。用户可以获得全部模板的信息。详见图25所描述的用例实现顺序图。

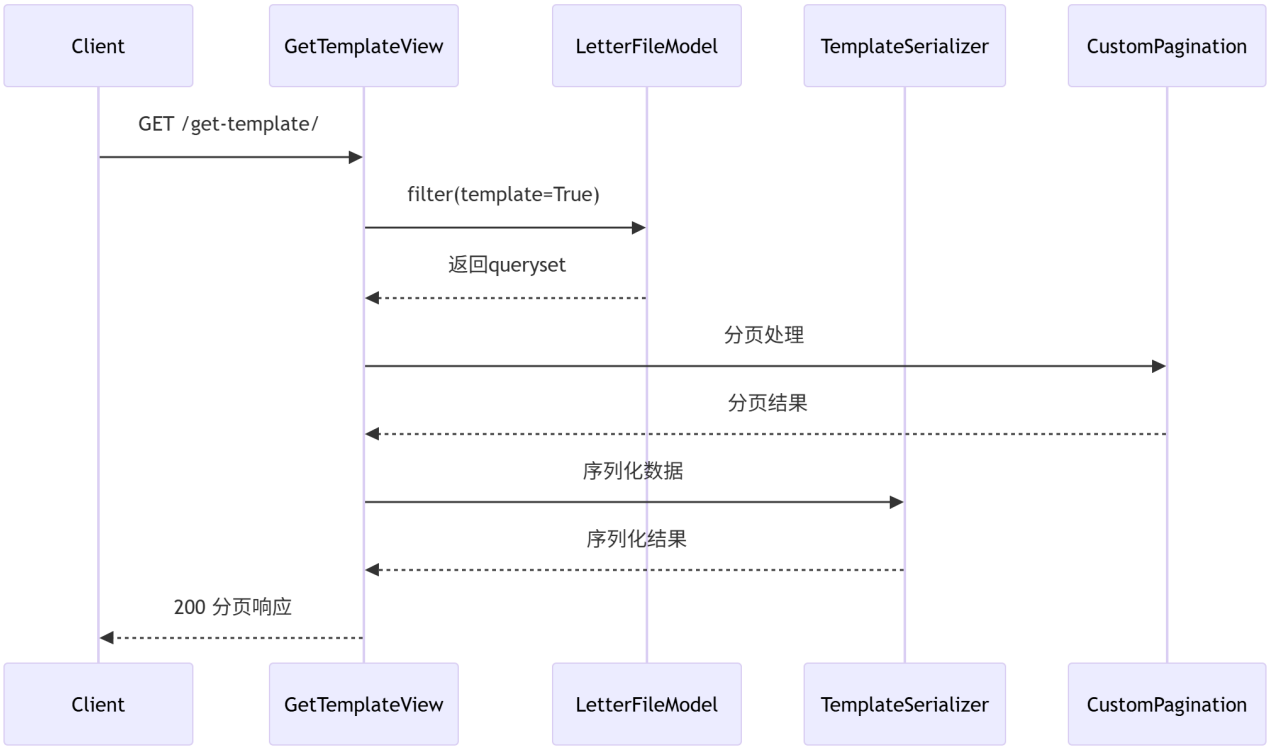


图25.查询公函模板用例设计的顺序图

- 用户在前端点击“公函”模板，向后端GetTemplateView视图发送请求。
- 后端使用filter函数创建queryset。
- 使用CustomPagination作为分页处理器。
- 使用TemplateSerializer作为序列化器，返回结果。
- 前端接收结果，展示结果。

查询公函状态用例实现的设计方案

查询公函状态使用GetStatusView视图实现。用户可以查询自己上传的公函的状态以及是否有回复。详见图26所描述的用例实现顺序图。

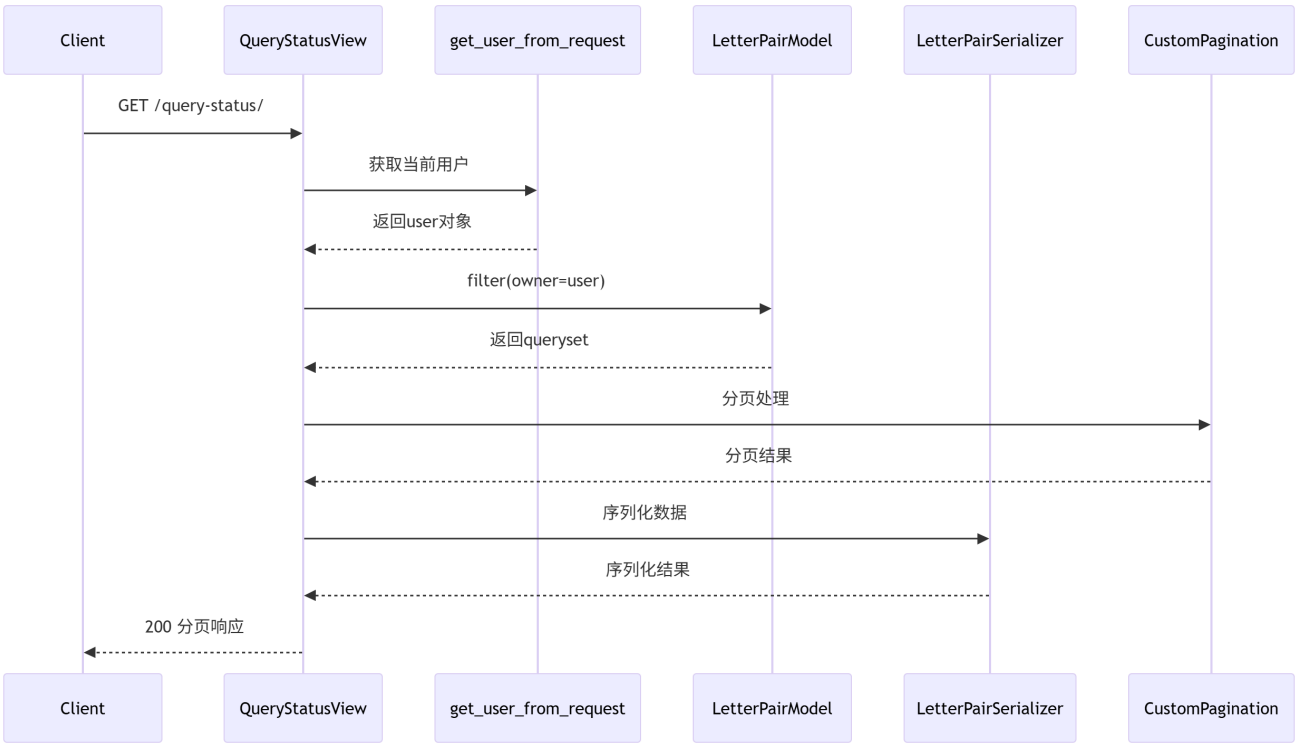


图26.查询公函状态用例设计的顺序图

- 前端用户点击“公函填写”，向后端GetStatusView发送请求。
- GetStatusView收到请求后，先使用get_user_from_request校验用户登录状态。
- 如果用户登录状态正常，则获取用户上传的所有LetterPair
- 使用CustomPagination作为分页处理器。
- 使用TemplateSerializer作为序列化器，返回结果。
- 前端接收结果，展示结果。

删除公函模板用例实现的设计方案

删除公函模板使用DeleteTemplateView视图实现。管理员选择公函模板删除。详见图27所描述的用例实现顺序图。

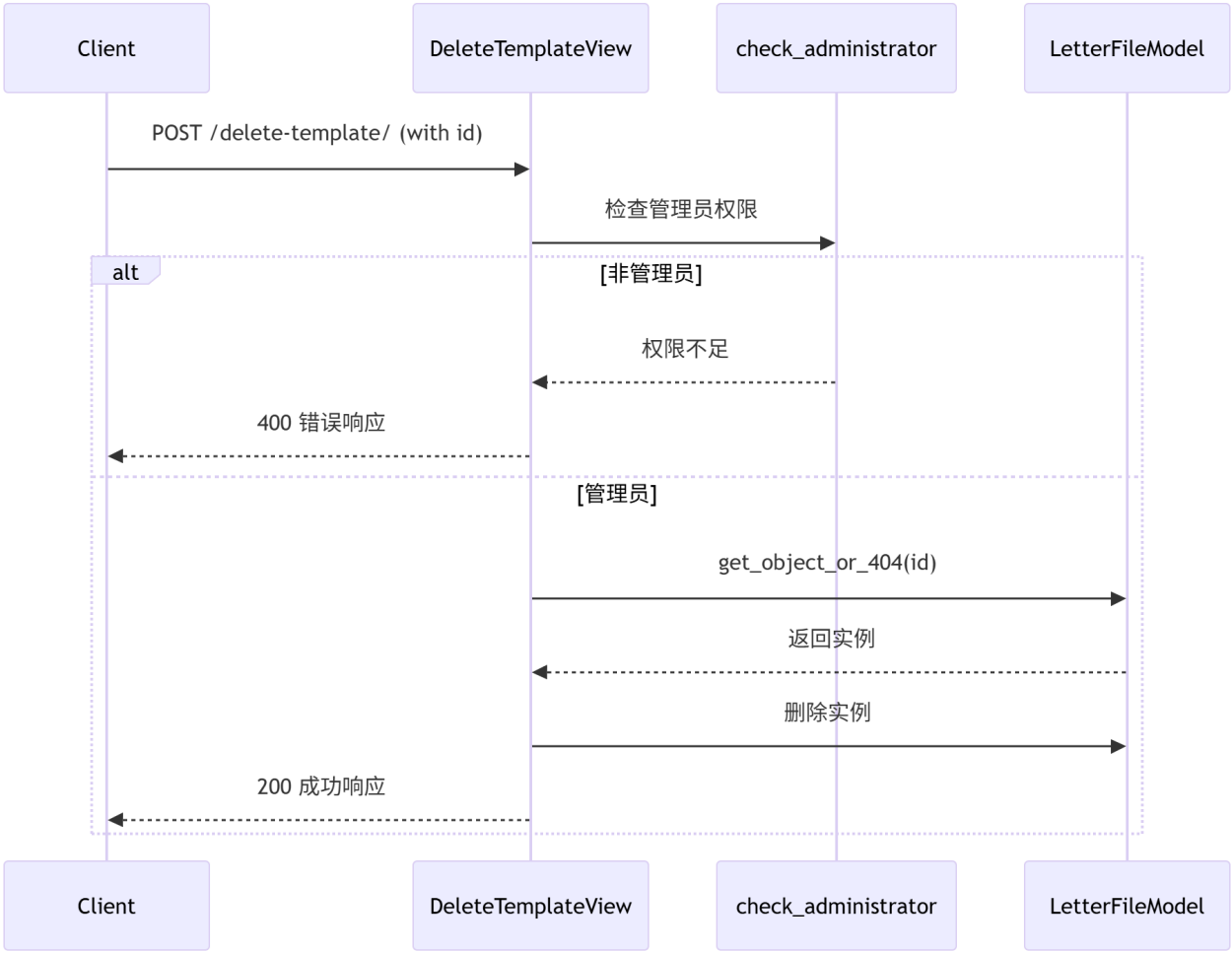


图27.删除公函模板用例设计的顺序图

- 用户在前端选择要删除的公函模板，前端向后端DeleteTemplateView视图发送主键。
- DeleteTemplateView接收后，先校验用户是不是管理员。
- 若是，使用get_object_or_404尝试获取指定的模板。
- 若存在，则删除并正常返回。
- 前端接收返回，显示“删除成功”。

实践日报初始化用例实现的设计方案

实践日报初始化使用InitLogView视图实现。支队队员可以填写本支队的出行日期、支队人数、调研主题、实践地点等信息。详见图28所描述的用例实现顺序图。

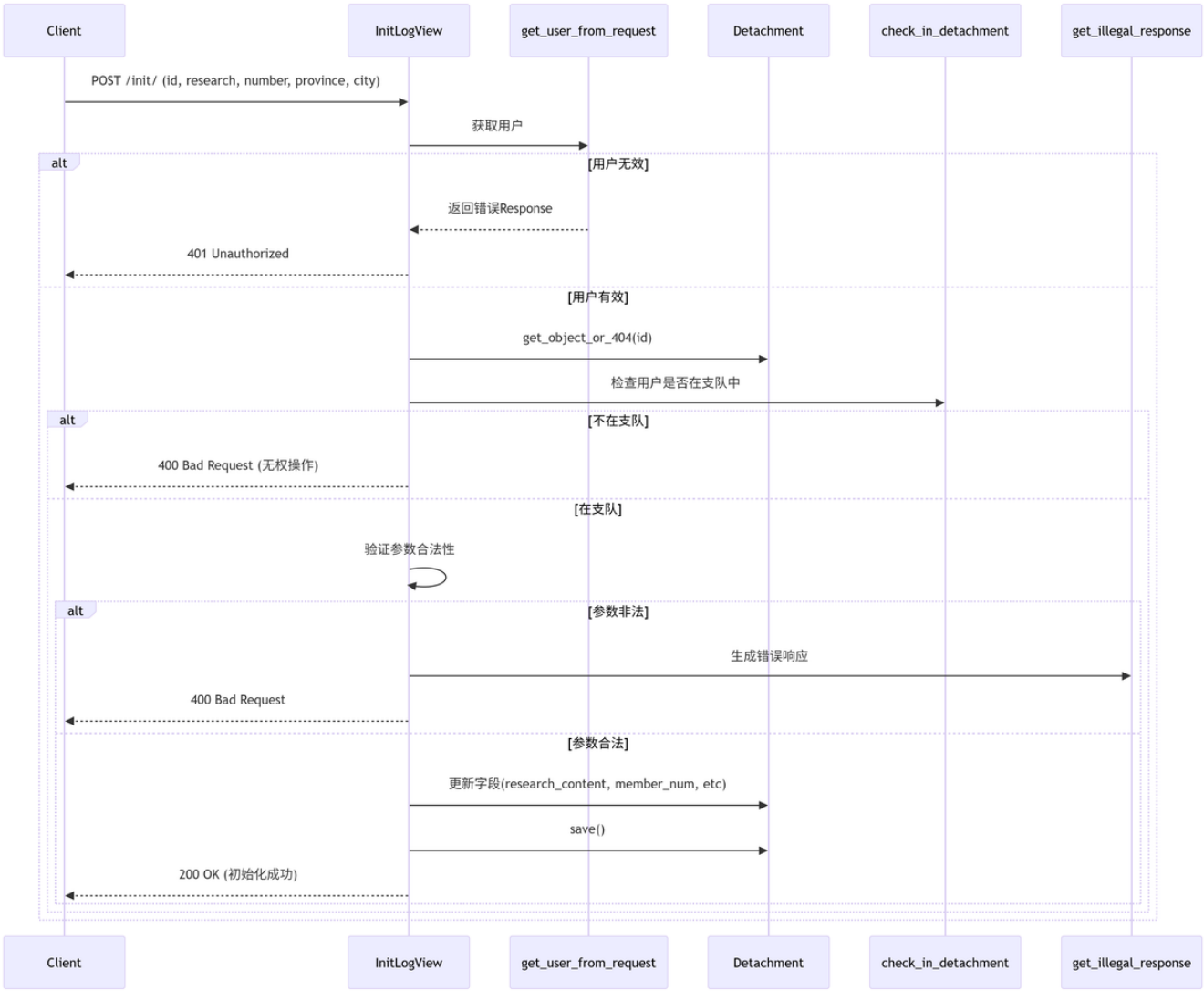


图28.实践日报初始化用例设计的顺序图

- 用户在前端填写好数据，向后端InitLogView视图发送请求。
- InitLogView校验用户的登录状态，并尝试获取用户信息。
- 若用户登录状态正常，则使用get_object_or_404来尝试获取支队信息。
- 若成功获取支队，则校验用户在不在支队中。
- 若用户身份符合要求，则校验传入数据是否合法。
- 若数据合法，则修改支队信息并保存，向前端正常返回。
- 前端接受后端响应，显示“初始化成功”。

填写实践日报用例实现的设计方案

填写实践日报使用WriteLogView视图实现。本支队用户可以填写本支队的实践日报。日报以日期作为区分，如果该日期的日报不存在则创建；如果存在则修改。详见图29所描述的用例实现顺序图。

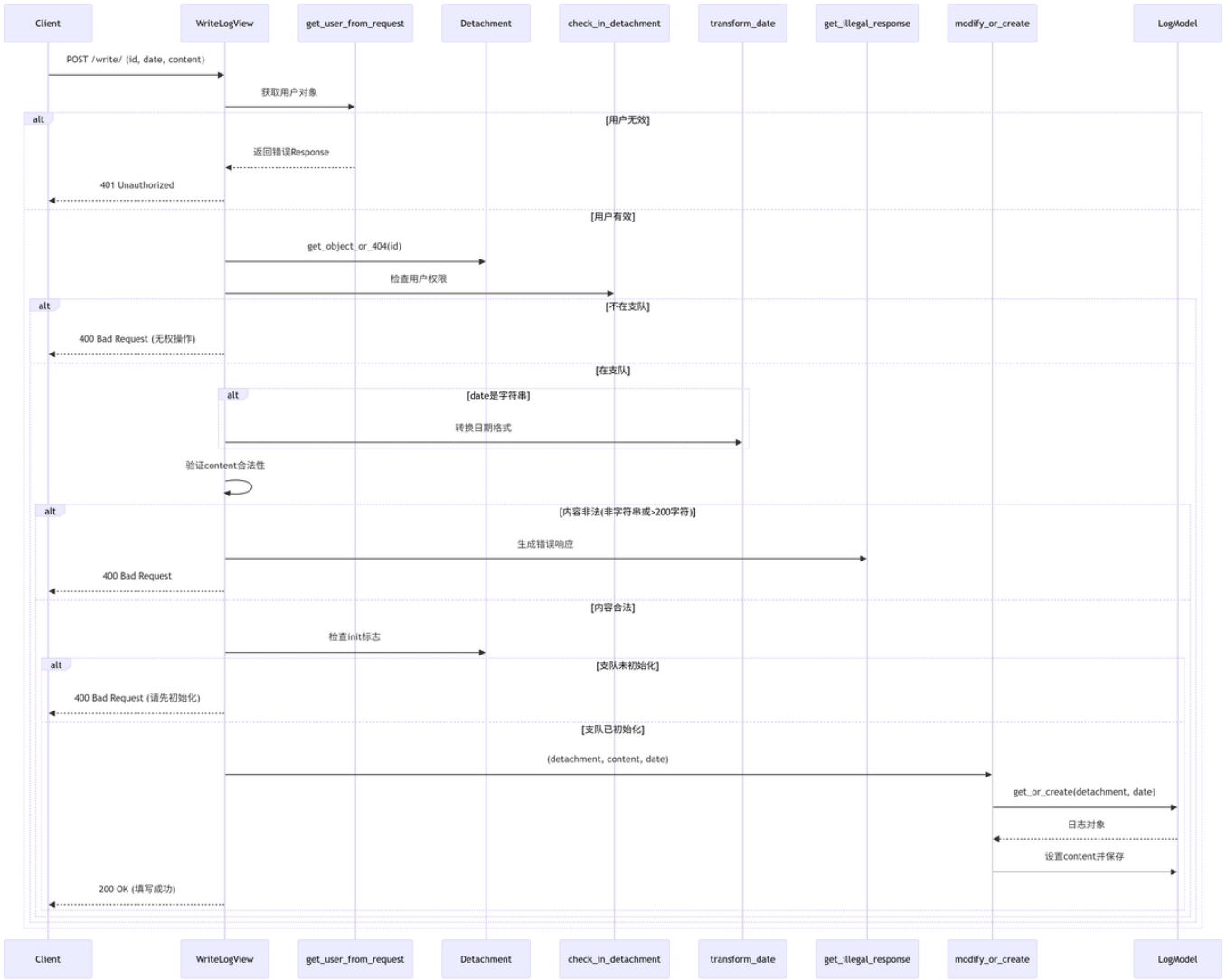


图29.填写实践日报用例设计的顺序图

- 用户在前端选择支队，填写好数据，向后端WriteLogView发送请求。
- WriteLogView接受请求，校验用户登录状态并尝试获取用户。
- 若用户登录状态正常，则使用get_object_or_404来尝试获取支队信息。
- 若成功获取支队，则校验用户在不在支队中。
- 若用户身份符合要求，则校验传入数据是否合法。
- 若数据合法，则检验日报有没有初始化。
- 若合法，则创建或修改日报，并向前端返回。
- 前端接收后端信息，并显示“填写成功”。

查询实践日报用例实现的设计方案

查询实践日报使用QueryLogView视图实现。用户可以看到所在所有支队的所有日报，管理员可以看到所有的日报。详见图30所描述的用例实现顺序图。

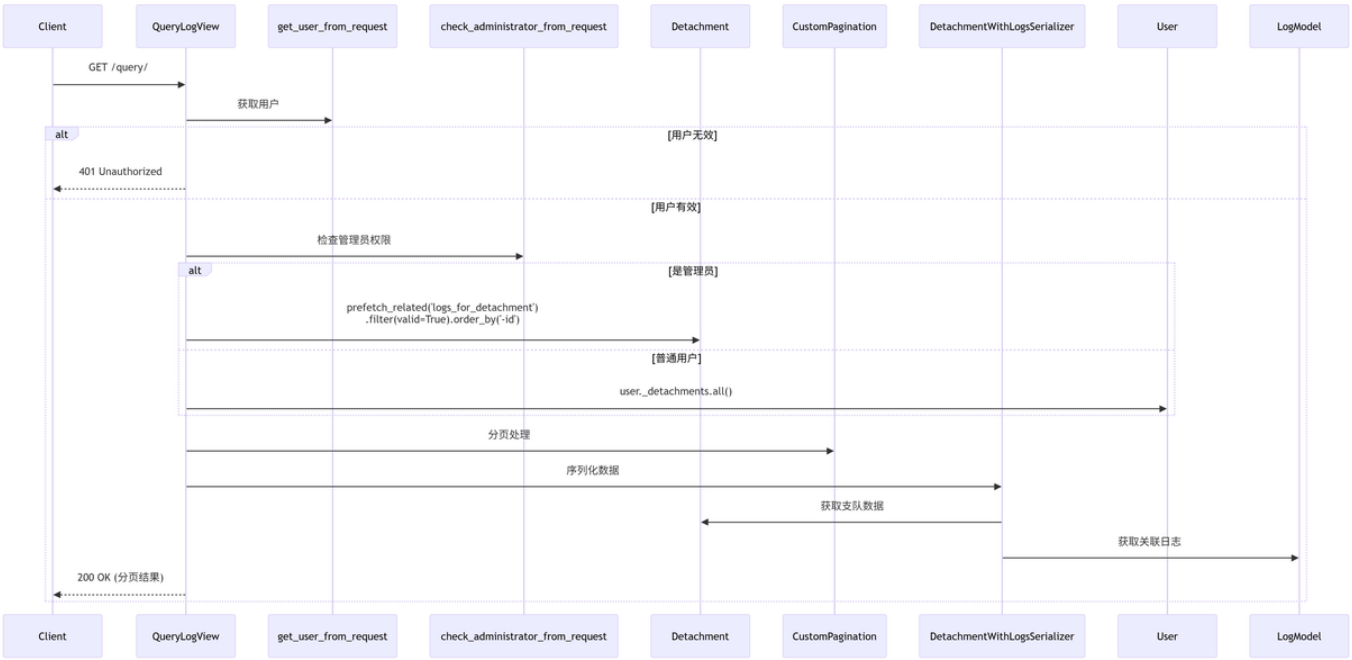


图30.查询实践日报用例设计的顺序图

- 用户在前端点击“实践日报”，向后端QueryLogView视图发送请求。
- QueryLogView视图接受到前端请求，校验用户登录状态并尝试获取用户。
- 若用户是管理员，则queryset是所有的支队。
- 若用户是普通用户，则queryset是所在的所有支队。
- 使用CustomPagination作为分页器，使用DetachmentWithLogSerializer作为序列化器。
- 序列化器会自动请求和支队关联的所有的日志，并向前端返回。
- 前端接收返回并展示。

上传外联清单用例实现的设计方案

上传外联清单使用UploadConnectionListView实现。管理员可以上传特定格式的Excel文件来更新外联清单。详见图31所描述的用例实现顺序图。

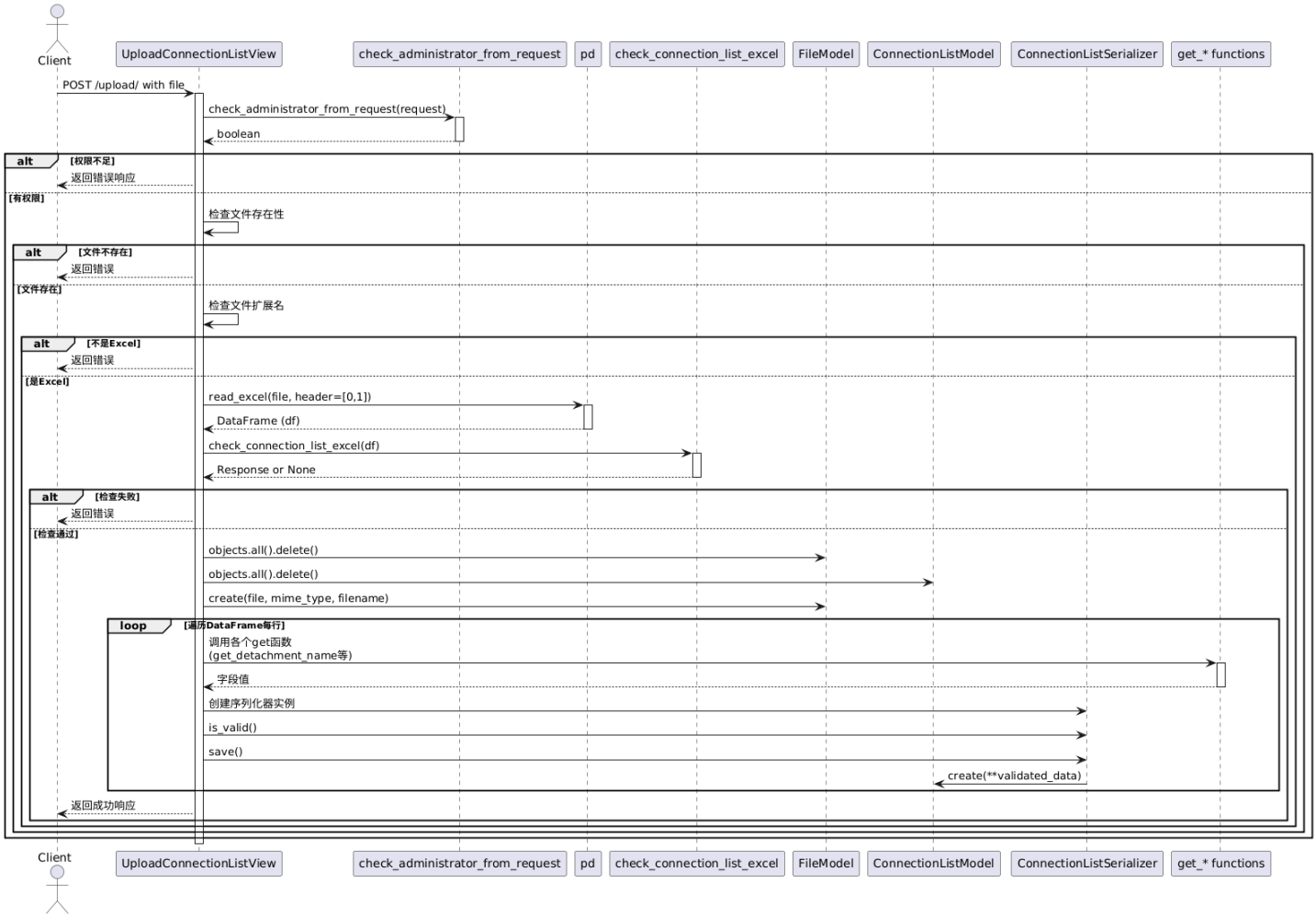


图31.上传外联清单用例设计的顺序图

- 前端点击“上传外联清单”，选择文件，向后端UploadConnectionListView视图发送请求。
- UploadConnectionListView视图接收请求，先校验用户登录状态与管理员权限。
- 若用户是管理员权限，则校验传入文件的格式是否正确。
- 若正确，则遍历文件所有条目，创建ConnectionListModel的实例。
- 创建完成，后端正常返回。
- 前端接收后端返回，显示“上传成功”。

下载外联清单用例实现的设计方案

下载外联前端使用DownloadConnectionListView实现。用户可以点击“下载外联清单”来下载。详见图32所描述的用例实现顺序图。

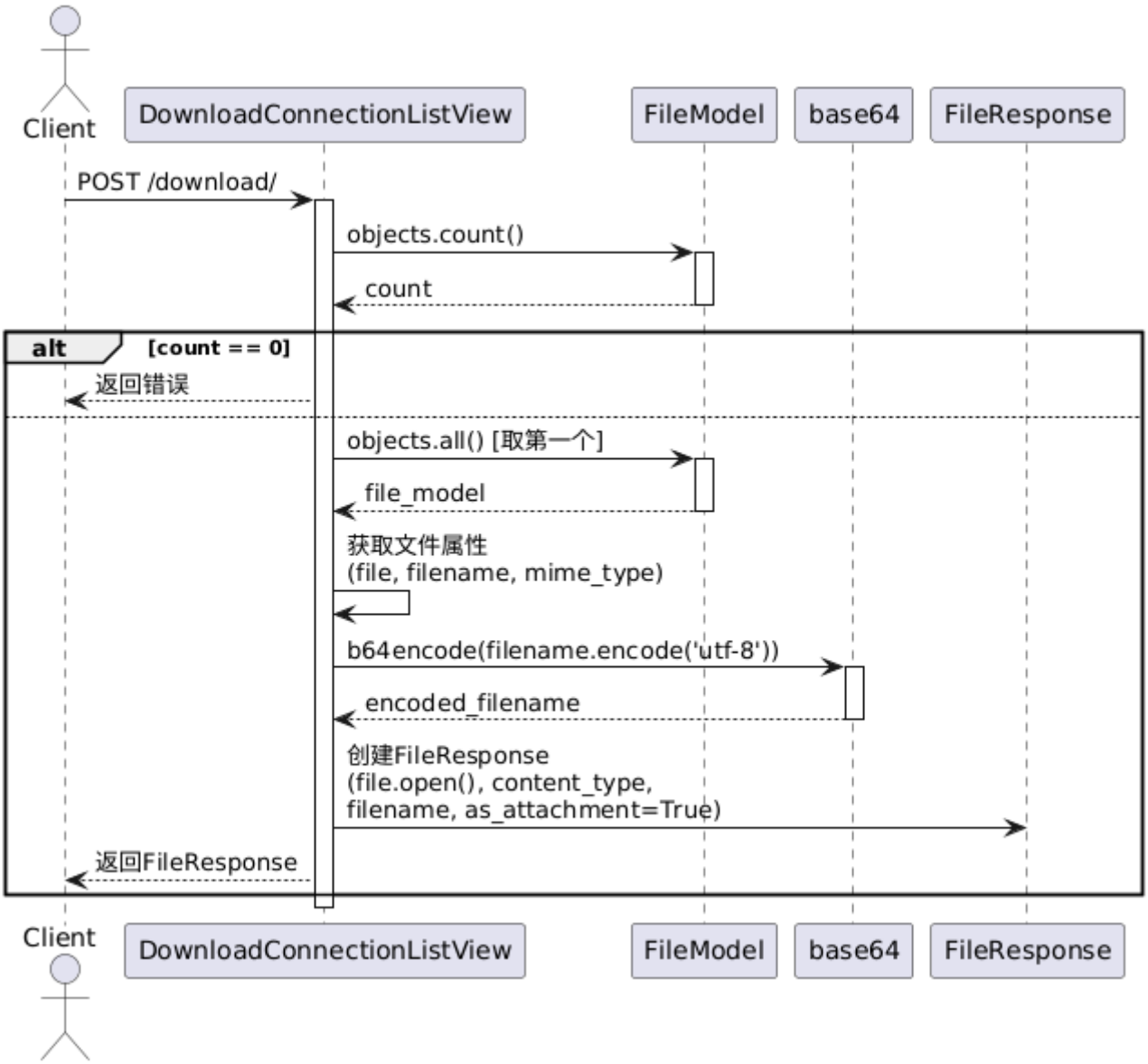


图32.下载外联清单用例设计的顺序图

- 前端点击“下载外联清单”，向后端DownloadConnectionListView视图发送请求。
- DownloadConnectionListView视图先检查有没有外联清单文件。
- 如果有，则获取文件属性，并对文件名采取utf-8 base64编码。
- 编码完成，创建文件响应并返回。
- 前端接收返回，开始下载文件。

查询外联清单用例实现的设计方案

查询外联清单使用QueryConnectionListView视图实现。用户可以查看所有的外联清单。详见图33所描述的用例实现顺序图。

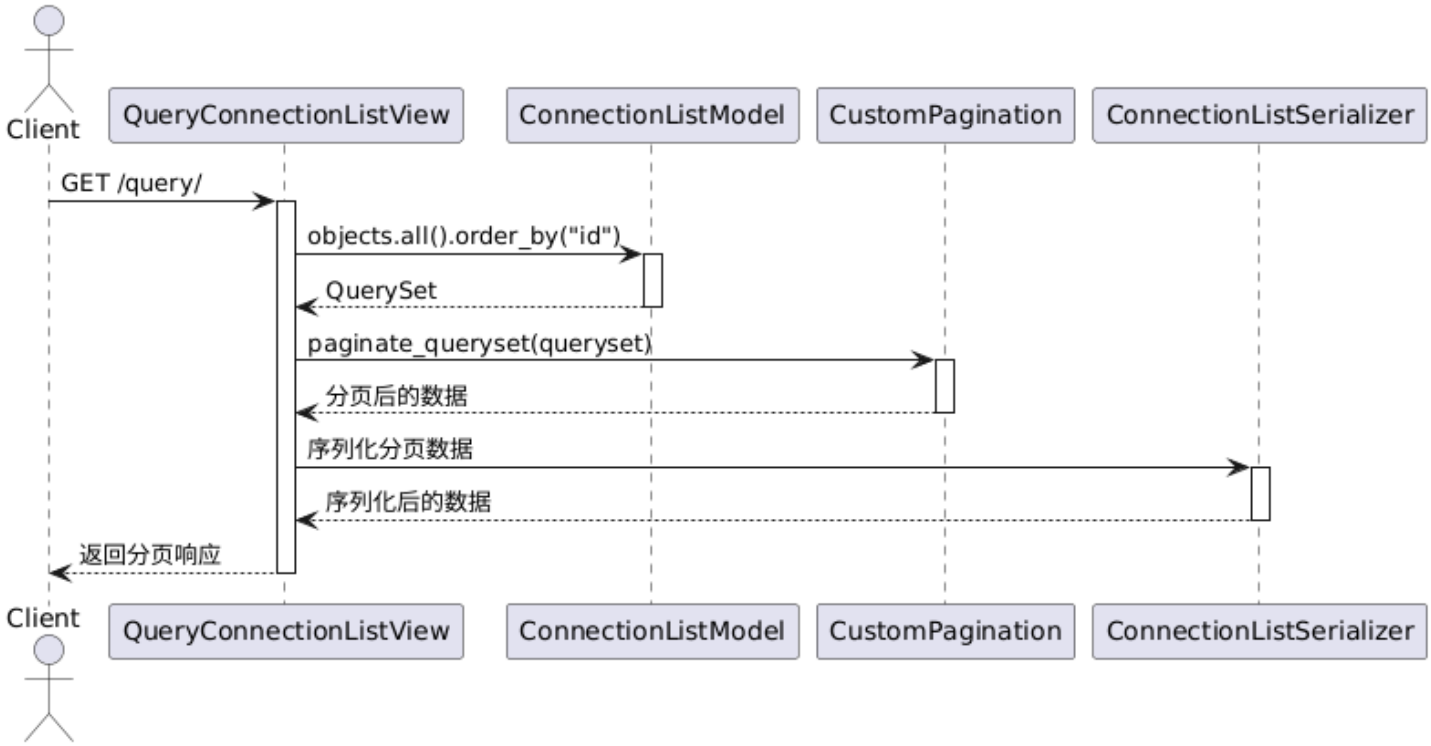


图33.查询外联清单用例设计的顺序图

- 用户在前端点击“外联清单”，向后端QueryConnectionListView视图发送请求。
- QueryConnectionListView视图接收请求，获取所有的ConnectionListModel的实例。
- 使用CustomPagination作为分页处理器。
- 使用ConnectionListSerializer作为序列化器，返回结果。
- 前端接收结果，展示结果。

清空外联清单用例实现的设计方案

清空外联清单使用ClearConnectionListView视图实现。管理员可以点击“清空外联清单”来清空。详见图34所描述的用例实现顺序图。

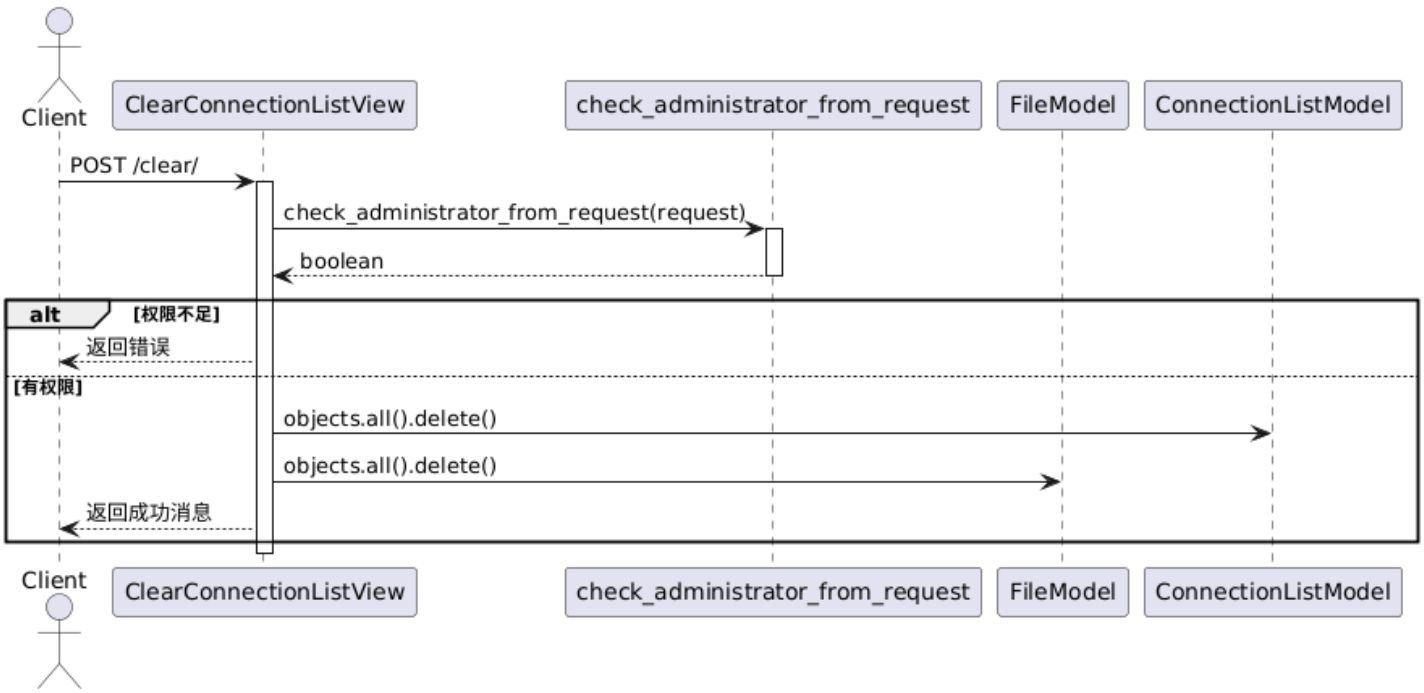


图34.查询外联清单用例设计的顺序图

- 用户在前端点击“清空外联清单”，向后端ClearConnectionListView视图发送请求。
- ClearConnectionListView视图接收请求，先校验用户的登录状态与管理员权限。
- 若通过，则清空ConnectionListModel与FileModel的所有实例，并正常返回。
- 前端接收后端返回，显示“清空成功”。

创建推送审核用例实现的设计方案

创建推送审核使用SendApprovalView视图实现。用户填写推送的链接和自己的姓名来创建审核。详见图35所描述的用例实现顺序图。

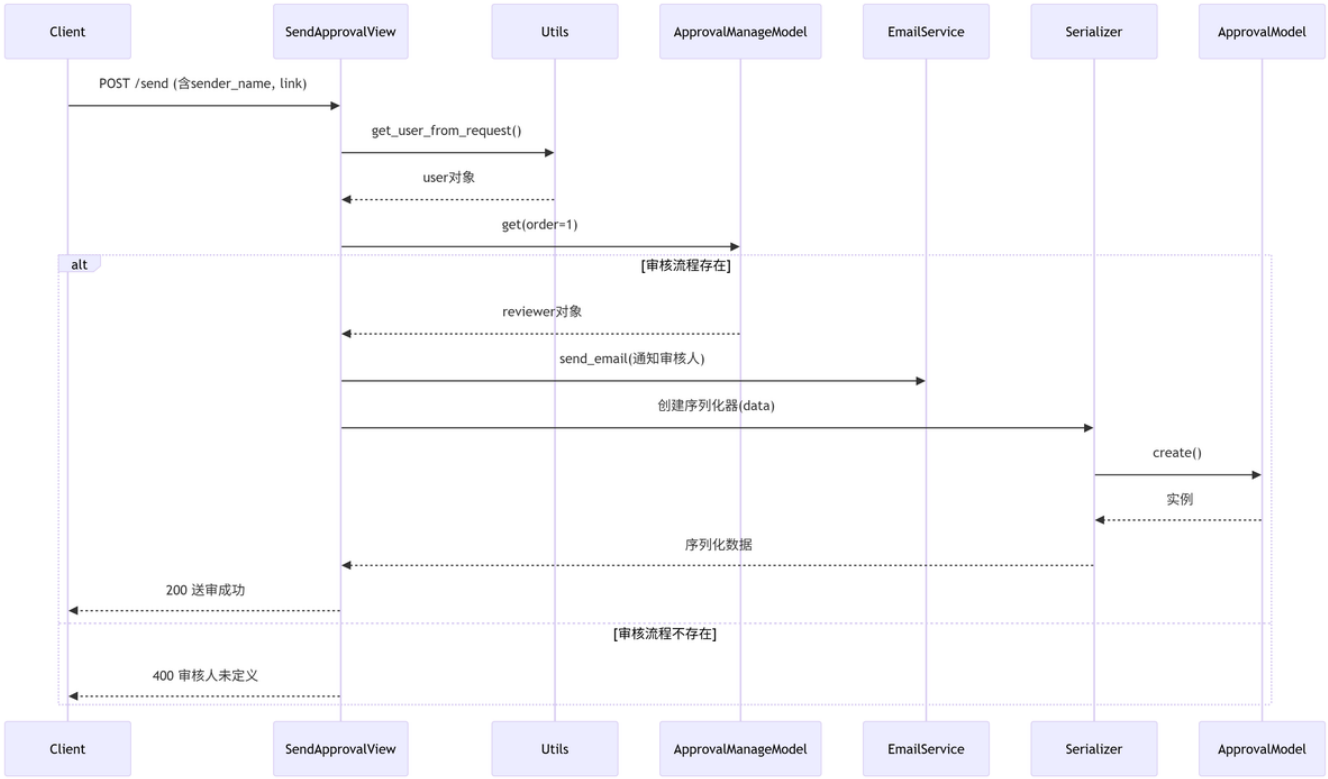


图35.创建推送审核用例设计的顺序图

- 用户在前端点击“推送送审”，填写好链接和姓名，向后端SendApprovalView视图发送请求。
- 后端SendApprovalView视图接收请求，先校验用户的登录状态。
- 若状态正常，则尝试获取第一个审核人。
- 如果审核人存在，那么向他发送邮件提醒审核。
- 使用序列化器创建ApprovalModel实例，校验数据格式是否合法。
- 若合法且成功创建，后端正常返回。
- 前端接收后端返回，显示“送审成功”。

通过审核用例实现的设计方案

通过审核使用PassDownApprovalView视图实现。审核人点击“通过”按钮即可通过，传递给下一个审核人。详见图36所描述的用例实现顺序图。

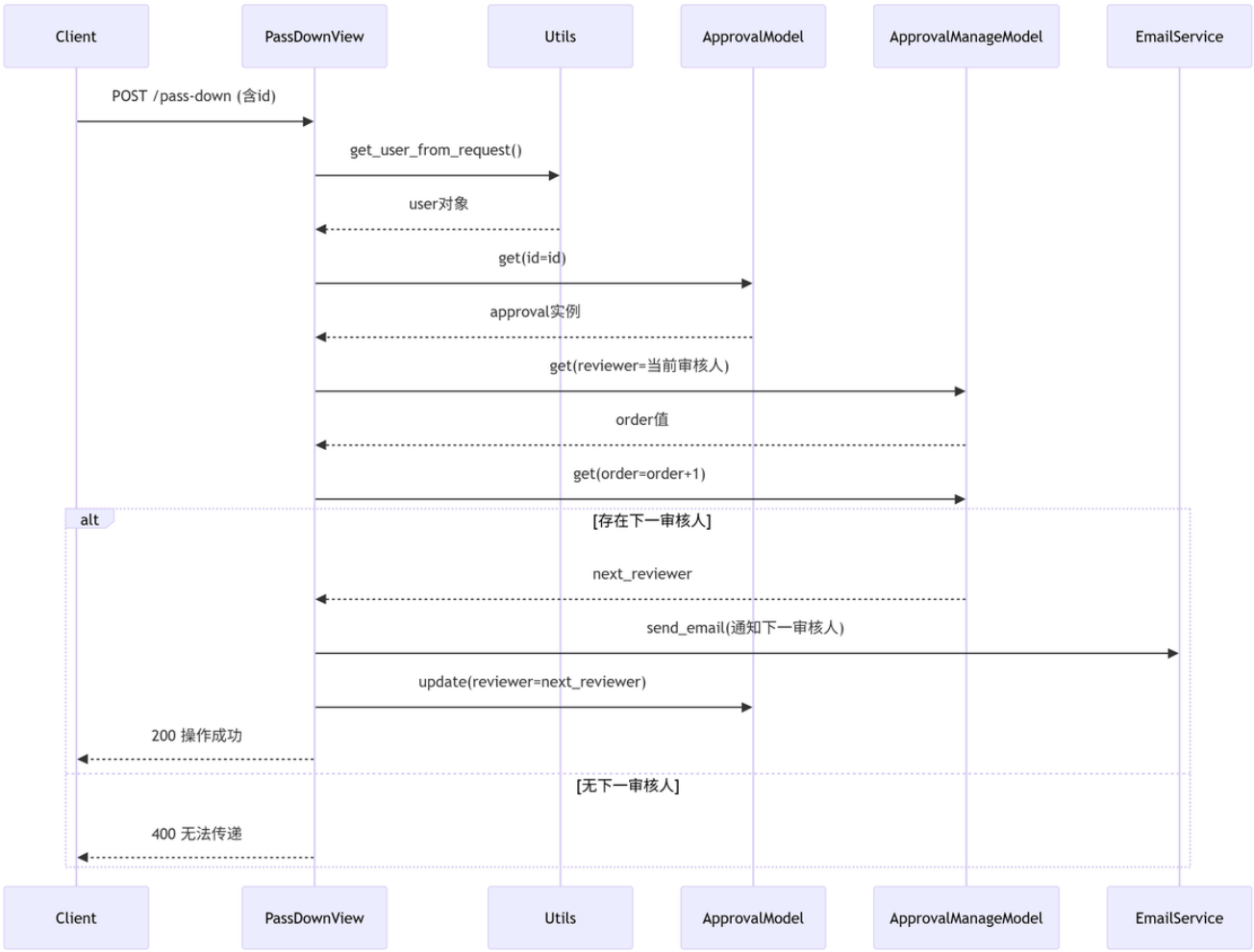


图36.通过推送审核用例设计的顺序图

- 前端用户点击特定推送的“通过”按钮，向后端PassDownApprovalView视图发送请求。
- 后端PassDownApprovalView视图校验用户的登录状态。
- 若状态正常，则使用主键获取ApprovalModel的实例。
- 若成功获取，则检查审核实例的当前审核人是不是当前用户。
- 若是，则获取当前审核人是第几个审核人。
- 若存在下一个审核人，则发送邮件提醒下一位，并修改审核实例的审核人信息。
- 后端正常返回，前端显示“操作成功”。

拒绝推送用例实现的设计方案

拒绝推送使用RejectApprovalView视图实现。审核人点击拒绝并填写理由，可以拒绝推送。详见图37所描述的用例实现顺序图。

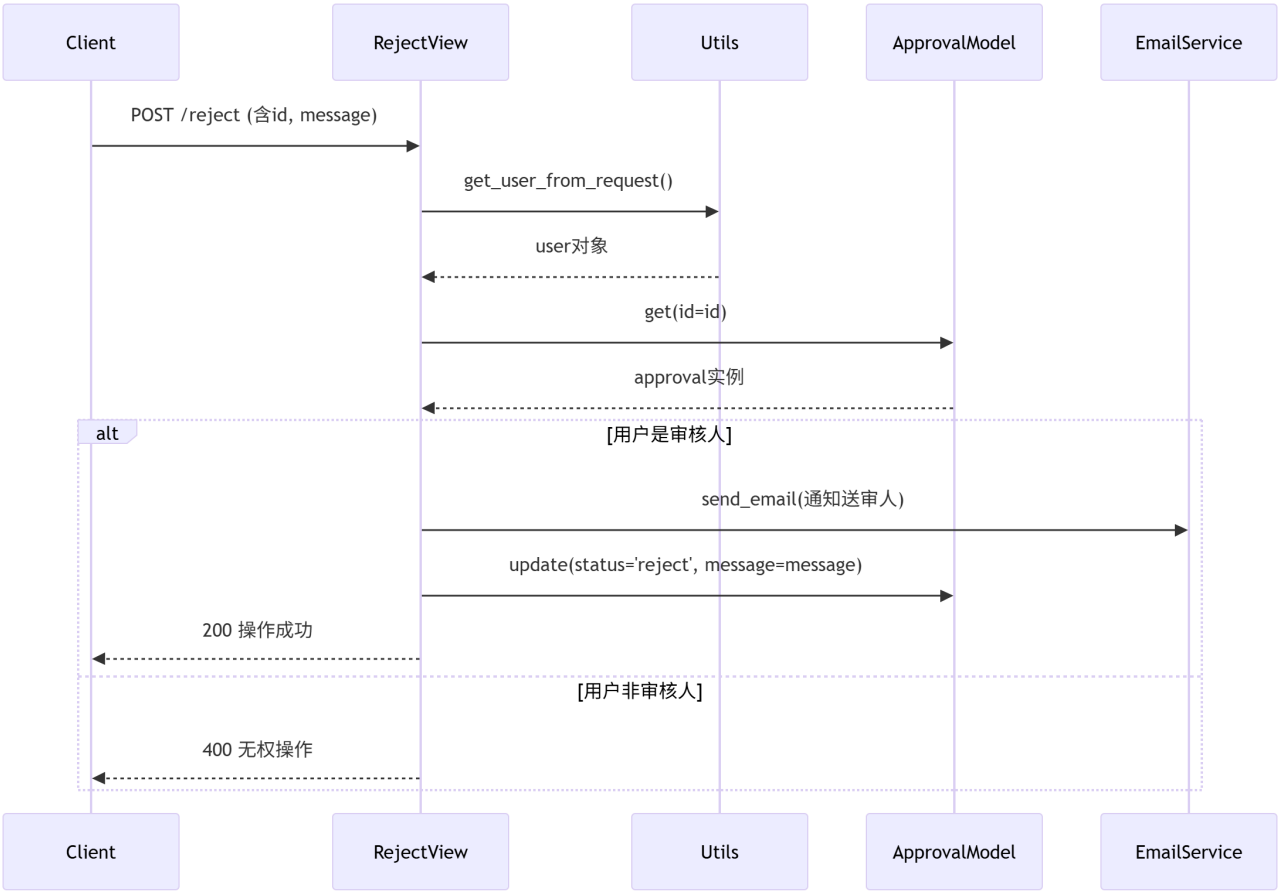


图37.拒绝推送审核用例设计的顺序图

- 前端用户点击特定推送的“拒绝”按钮，填写理由，向后端RejectApprovalView视图发送请求。
- 后端RejectApprovalView视图校验用户的登录状态。
- 若状态正常，则使用主键获取ApprovalModel的实例。
- 若成功获取，则检查审核实例的当前审核人是不是当前用户。
- 若是，则发送邮件提醒送审人不通过，并更新审核实例的状态。
- 后端正常返回，前端显示“操作成功”。

允许发表用例实现的设计方案

允许发表推送使用ApproveApprovalView视图实现。最终审核人点击“允许发表”按钮可以允许推送发表。详见图38所描述的用例实现顺序图。

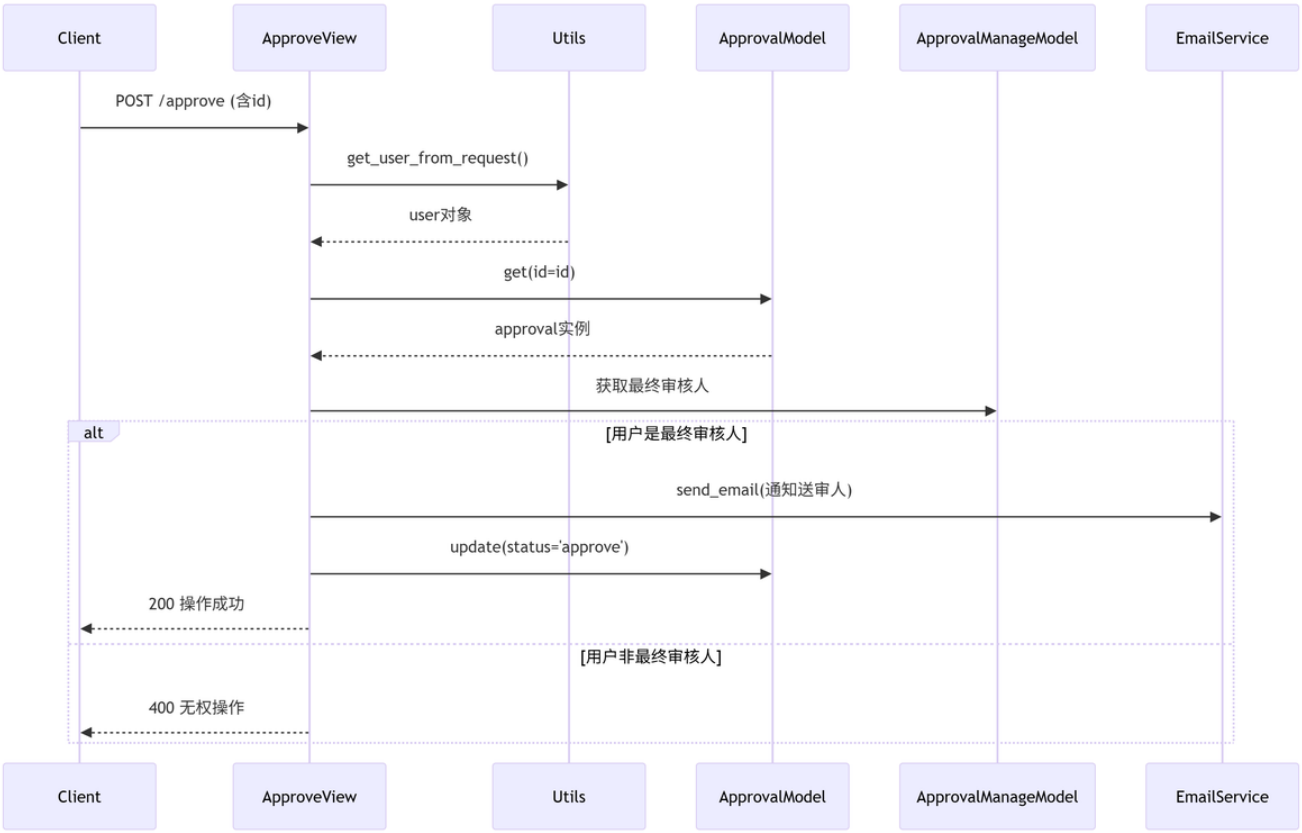


图37.允许发表推送用例设计的顺序图

- 前端用户点击特定推送的“允许发表”按钮，向后端ApproveApprovalView视图发送请求。
- 后端ApproveApprovalView视图校验用户的登录状态。
- 若状态正常，则使用主键获取ApprovalModel的实例。
- 若成功获取，则检查审核实例的最终审核人是不是当前用户。
- 若是，则发送邮件提醒送审人允许发表，并更新审核实例的状态。
- 后端正常返回，前端显示“操作成功”。

管理审核流程用例实现的设计方案

管理审核流程使用ManageApprovalView视图实现。超级管理员可以依序选择用户作为审核人。详见图38所描述的用例实现顺序图。

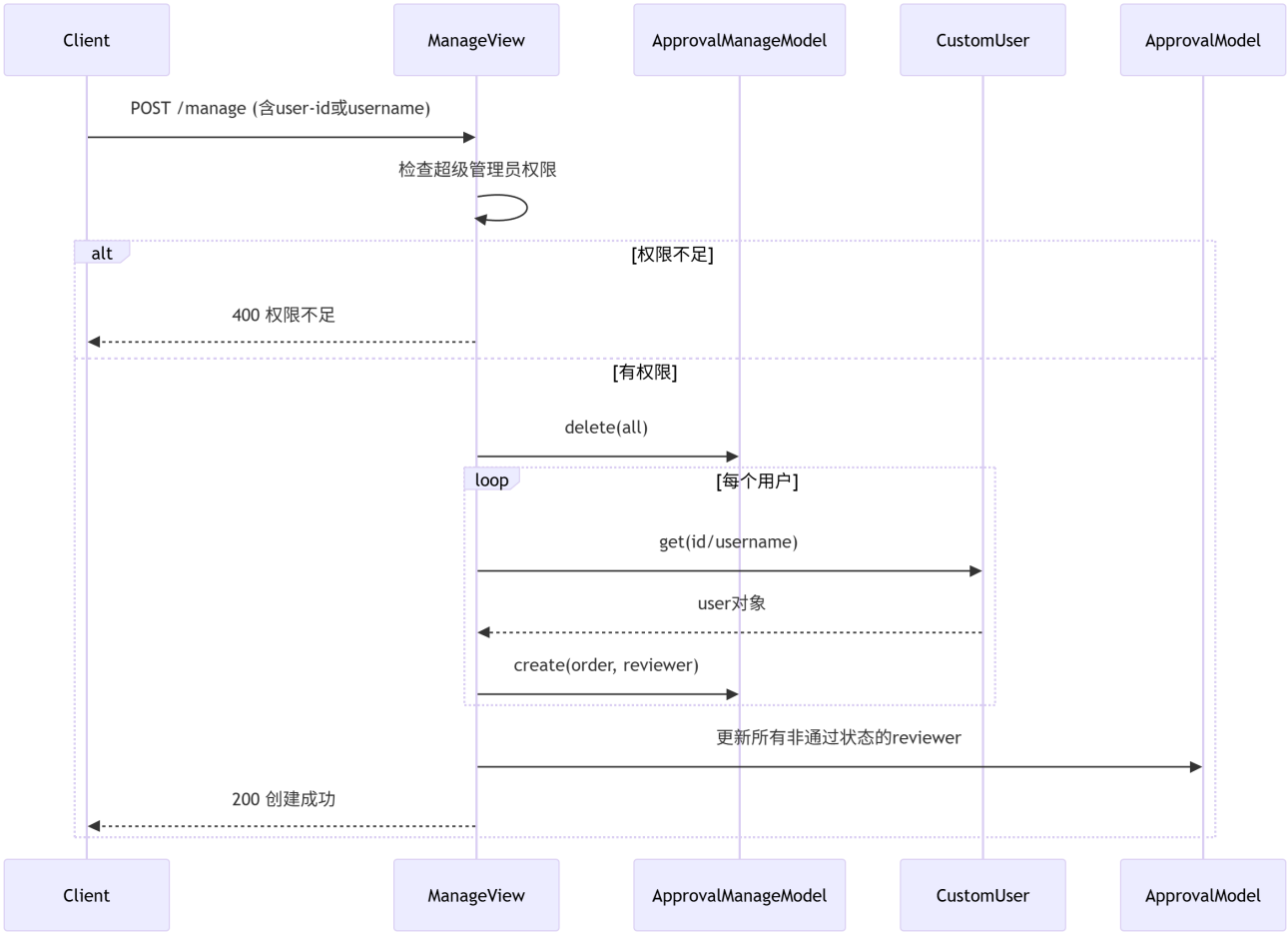


图38.管理审核流程用例设计的顺序图

- 前端点击“推送审核流程管理”吗，选择用户，向后端ManageApprovalView视图发送请求。
- ManageApprovalView视图接收请求，先校验用户的登录状态与超级管理员权限。
- 若校验通过，则先删除当下的使用审核人，再按需创建审核人实例。
- 更新所有非通过状态的推送审核的reviewer。
- 后端正常返回，前端显示“操作成功”。

查询审核状态用例实现的设计方案

查询审核状态使用QueryStatusView视图实现。用户可以查看自己所有推送的详细状态。详见图39所描述的用例实现顺序图。

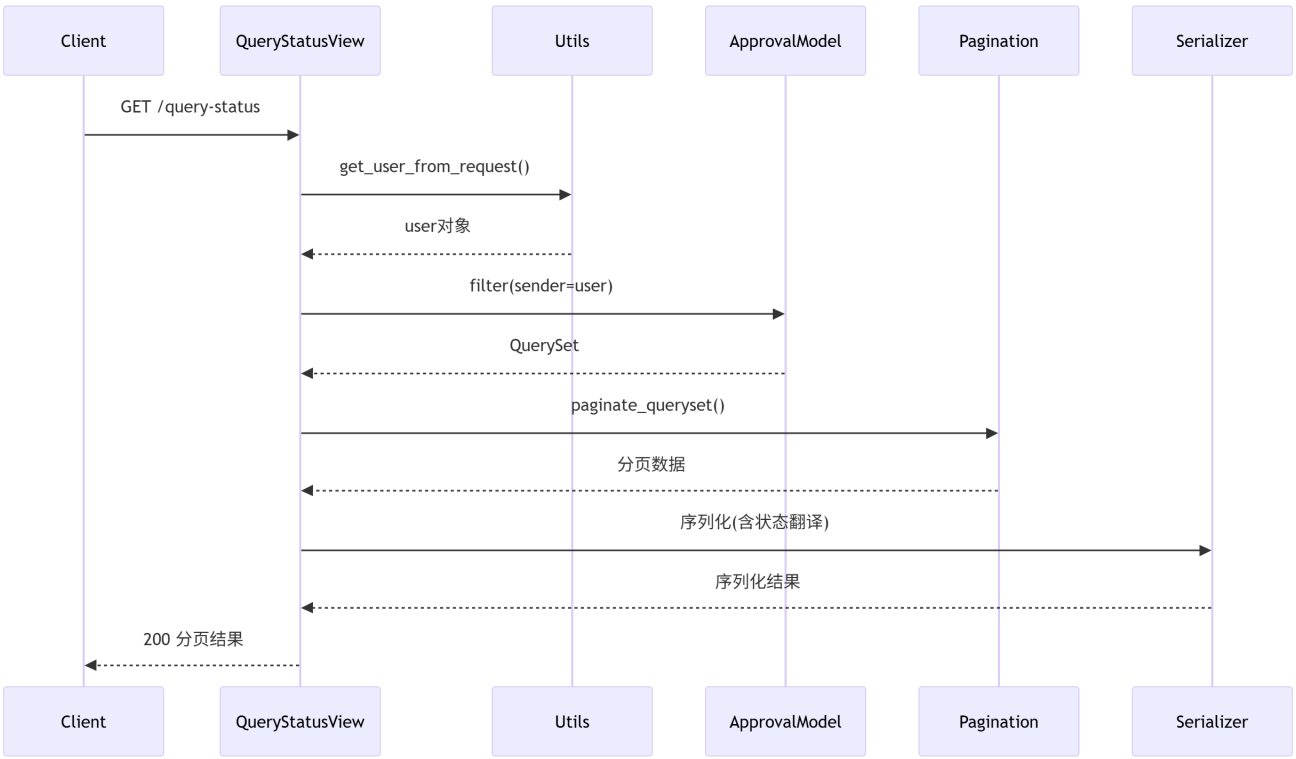


图39.查询审核状态用例设计的顺序图

- 用户在前端点击“推送申请”，向后端QueryStatusView视图发送请求。
- 后端QueryStatusView视图校验用户的登录状态。
- 若登录状态正常，则获取用户的所有推送审核。
- 使用CustomPagination作为分页处理器。
- 使用ApprovalStatusSerializer作为序列化器，返回结果。
- 前端接收结果，展示结果。

修改推送用例实现的设计方案

修改推送链接使用ModifyApprovalView视图实现。用户点击自己的某个审核的“修改”按钮，填写新的链接来达成修改效果。详见图40所描述的用例实现顺序图。

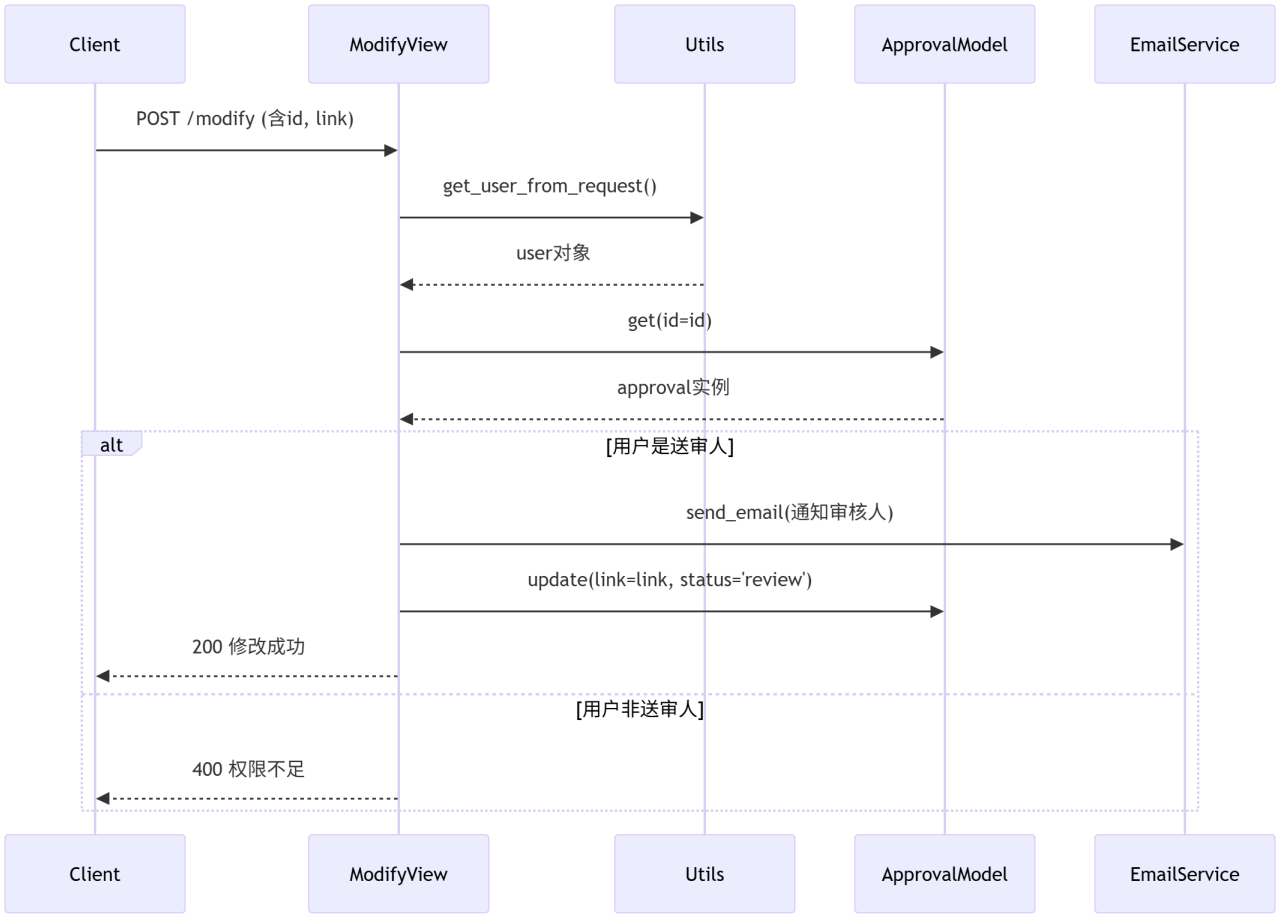


图40.修改推送用例设计的顺序图

- 前端用户点击特定推送的“修改”按钮，填写链接，向后端ModifyApprovalView视图发送请求。
- 后端ModifyApprovalView视图校验用户的登录状态。
- 若状态正常，则使用主键获取ApprovalModel的实例。
- 若成功获取，则检查审核实例的送审人是不是当前用户。
- 若是，则更新推送链接，并发邮件提醒审核人。
- 后端正常返回，前端显示“操作成功”。

3.2 软件体系结构设计

> 按照数据流或者逻辑流分层描述软件系统的体系结构设计，并提供必要的文字补充说明。

图2为 酒井实践在线（THUPracticeOnline）软件的层次化软件体系结构模型，共分为三层：用户界面层、业务处理层、基础服务层。

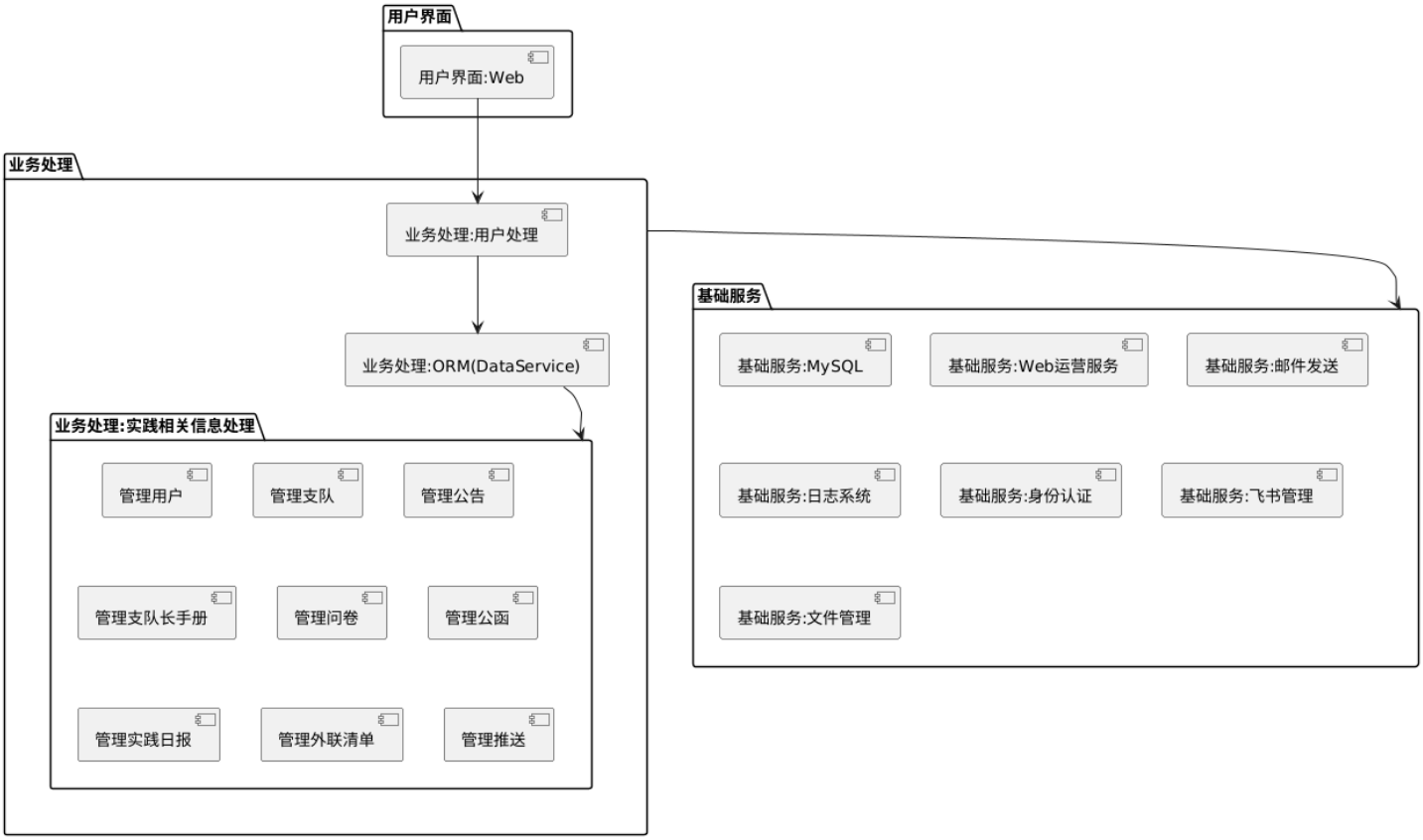


图2. 酒井实践在线（THUPracticeOnline）的体系结构模型

- **用户界面层**

  用户界面以 Web 的形式展示，提供用户与系统交互的界面，具体包括：展示支队信息、公告、用户信息等；支持用户进行管理支队长手册、管理问卷、用户登录与注册等操作；处理用户输入并将其传递给业务处理层。

- **业务处理层**

  负责处理用户通过用户界面层提交的请求，执行具体的业务逻辑，如用户处理、ORM（对象关系映射）和数据服务，以及实践相关信息处理，包括管理用户、管理支队、管理公告等。此外，业务处理层还与基础服务层进行交互

- **基础服务层**

  提供系统运行所需的各种基础服务，如MySQL数据库服务、Web运营服务、邮件发送服务、日志系统、身份认证服务、飞书管理服务以及文件管理服务。

3.3 数据库设计

> 给出软件系统中数据库表结构的设计，描述清楚每个数据字段的类型和意义。

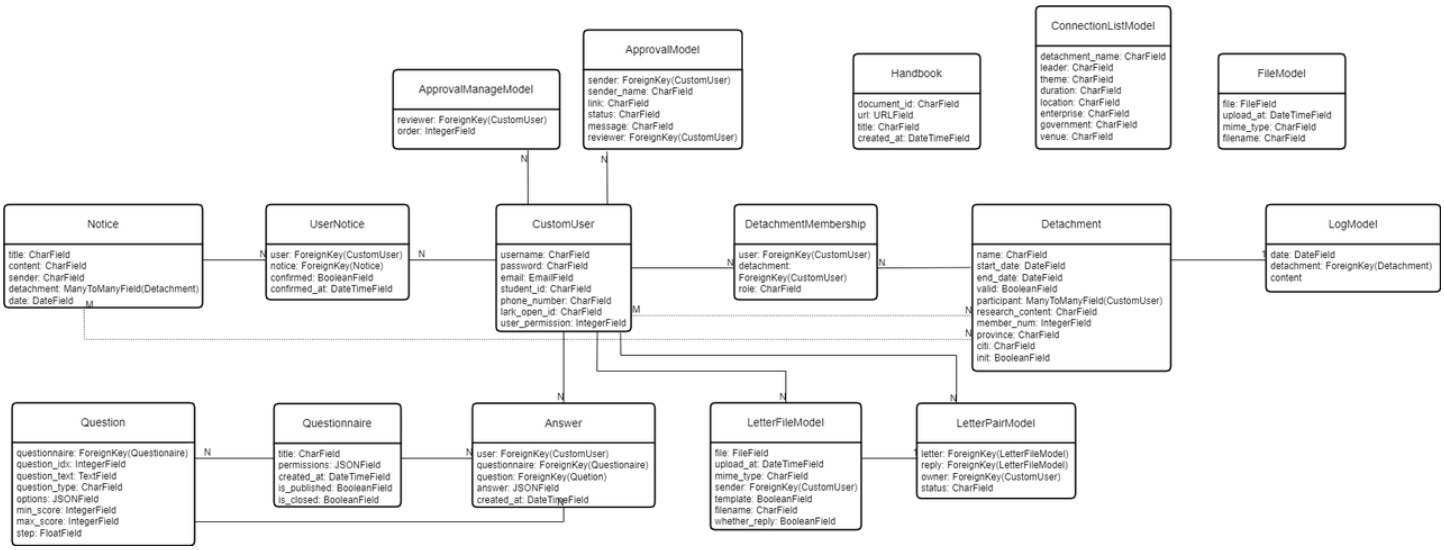


图 x. 酒井实践在线 软件数据库设计

本项目的数据库设计包括多个表，每个表都有其特定的用途和字段。以下是对每个表的详细描述：

CustomUser: 存储用户信息

- username：用户名，类型：CharField
- password：密码，类型：CharField
- email：邮箱，类型：EmailField
- student_id：学生号，类型：CharField
- phone_number：手机号，类型：CharField
- lark_open_id：飞书open id，类型：CharField
- user_permission：用户权限，类型：IntegerField

Detachment: 存储支队信息

- name：实践名称，类型：CharField
- start_date：实践开始日期，类型：DateField
- end_date：实践结束日期，类型：DateField
- valid：实践是否停用，类型：BooleanField
- participant：实践队员，通过DetachmentMembership中间表多对多关联到CustomUser
- research_content：支队的主要实践内容，类型：CharField
- member_num：支队人数，类型：IntegerField
- province：实践具体地市/县，类型：CharField
- city：实践省份/国家，类型：CharField
- init：是否初始化了实践日报，类型：BooleanField

DetachmentMembership: 存储支队成员关系及角色

- user：关联的用户，外键到CustomUser
- detachment：关联的支队，外键到Detachment
- role：角色（支队长/支队员），类型：CharField

Handbook: 存储支队长手册信息

- document_id：文档ID，类型：CharField
- url：文档链接，类型：URLField
- title：标题，类型：CharField
- created_at：创建时间，类型：DateTimeField

LogModel: 存储实践日志

- date：日期，类型：DateField
- detachment：关联的支队，外键到Detachment
- content：日志内容，类型：CharField

Notice: 存储通知信息

- title：通知标题，类型：CharField
- content：通知内容，类型：CharField
- sender：发送方，类型：CharField
- detachment：多对多关联到Detachment
- date：发布日期，类型：DateField

UserNotice: 存储用户对通知的确认状态

- user：关联的用户，外键到CustomUser
- notice：关联的通知，外键到Notice
- confirmed：是否确认，类型：BooleanField
- confirmed_at：确认时间，类型：DateTimeField

Questionnaire: 存储问卷信息

- title：问卷标题，类型：CharField
- permissions：可填写问卷的权限列表，类型：JSONField
- created_at：创建时间，类型：DateTimeField
- is_published：是否发布，类型：BooleanField
- is_closed：是否停止收集，类型：BooleanField

Question: 存储问卷中的问题

- questionnaire：关联的问卷，外键到Questionnaire

- question_idx：题号，类型：IntegerField
- question_text：题目描述，类型：TextField
- question_type：题目类型（单选/多选/填空/打分），类型：CharField
- options：题目选项，类型：JSONField
- min_score：最低分数（仅打分题），类型：IntegerField
- max_score：最高分数（仅打分题），类型：IntegerField
- step：分数步长（仅打分题），类型：FloatField

Answer: 存储用户提交的答案

- user：关联的用户，外键到CustomUser
- questionnaire：关联的问卷，外键到Questionnaire
- question：关联的问题，外键到Question
- answer：答案内容，类型：JSONField
- created_at：提交时间，类型：DateTimeField

ApprovalManageModel: 存储审核流程管理信息

- reviewer：审核人，外键到CustomUser
- order：审核顺序，类型：IntegerField

ApprovalModel: 存储推送审核信息

- sender：送审人，外键到CustomUser
- sender_name：送审人姓名，类型：CharField
- link：秀米链接，类型：CharField
- status：审核状态（审核中/拒绝/通过），类型：CharField
- message：审核评语，类型：CharField
- reviewer：当前审核人，外键到CustomUser

ConnectionListModel: 存储外联清单信息

- detachment_name：支队名称，类型：CharField
- leader：支队长姓名，类型：CharField
- theme：实践主题，类型：CharField
- duration：实践时长，类型：CharField
- location：实践地点，类型：CharField
- enterprise：参访企业，类型：CharField
- government：对接政府机构，类型：CharField

- venue：实践场馆，类型：CharField

FileModel: 存储上传的文件

- file：文件，类型：FileField
- upload_at：上传时间，类型：DateTimeField
- mime_type：文件MIME类型，类型：CharField
- filename：文件名，类型：CharField

LetterFileModel: 存储公函文件

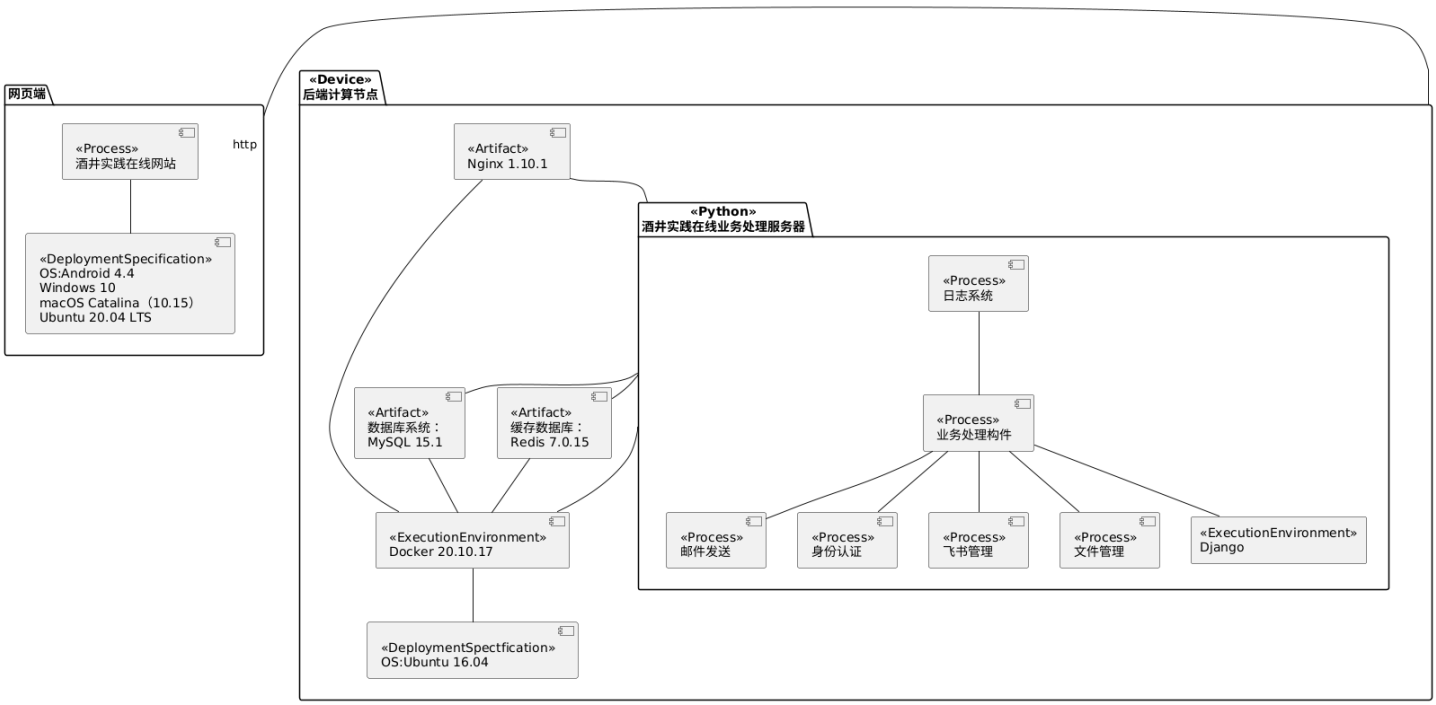
- file：公函文件，类型：FileField
- upload_at：上传时间，类型：DateTimeField
- mime_type：文件MIME类型，类型：CharField
- sender：发送人，外键到CustomUser
- template：是否为模板，类型：BooleanField
- filename：文件名，类型：CharField
- whether_reply：是否有回复，类型：BooleanField

LetterPairModel: 管理公函与回复的关系

- letter：待审批公函，外键到LetterFileModel
- reply：回复公函，外键到LetterFileModel
- owner：所属用户，外键到CustomUser
- status：审批状态（盖章中/已盖章），类型：CharField

3.4 部署设计

> 描述清楚如何部署整个软件。给出各个组件的部署方式，说明如何连接各个组件，如何写入初始数据。



网页端

网页端代表手机或电脑浏览器网页，酒井实践在线运行在Android 4.4以上的Android操作系统上或Windows 10 以上、macOS Catalina（10.15）以上或 Ubuntu 20.04 LTS 以上，使用HTTP协议与后端计算节点进行通信，发送操作请求。

后端计算节点

后端计算节点运行 Ubuntu 操作系统，该操作系统提供了一个稳定且安全的环境，支持 Docker容器技术 。Docker容器技术提供了各软件的运行环境。

- Nginx

  在 Docker 环境中托管，为酒井实践在线应用程序的请求的代理服务器，处理用户的请求，与酒井实践在线处理服务器进行通信。

- 酒井实践在线业务处理服务器

  在 Docker 环境中托管，项目代码部署在此。托管了多个业务构件：业务处理构件、日志系统、邮件发送、身份认证、飞书管理、文件管理。其中酒井实践在线业务处理构件通过 shell 脚本直接调用除日志系统外的构件进行通信。

- 数据库系统

  在 Docker 环境中托管，使用 MySQL 关系型数据库对数据进行持久化存储。MySQL用于存储结构化数据，如用户信息、支队信息和通知详情等。

- 缓存数据库

  在 Docker 环境中托管，使用 Redis 软件对数据进行缓存，在本项目中使用 Redis 实现验证码存储与验证功能，也能对其他数据进行缓存。