摘要

本论文从实际应用场景出发，以医学论文投审流程为基础，说明了医学论文投审系统的软件及其安全方案的设计与实现。本论文包含对该投审系统的需求分析、详细设计以及实现情况，主要涉及软件工程方法、React 和 Ant Design 等框架的使用和安全策略的设计与实现等。

本论文还总结了在毕业设计期间所获得的经验以及对软件工程的认识，以及知识技能的学习情况。

关键词：React、Ant Design、安全策略、医学论文审阅

Abstract

This bachelor thesis introduces the design and implementation of Submission & Revision System of Medical Articles on the basis of the use cases and workflow of medical articles' submission and revision. It includes the requirement analysis, detailed design and development of this system, mainly related to methodology of software engineering, implementation of React and Ant Design and security solution design.

This thesis also summarizes the experience in graduation project, idea of software engineering and study situation of knowledge and skills.

Keywords: React, Ant Design, Security Solution, Submission & Revision of Medical Articles

目 录

第一章 毕业设计（顶岗实习）概况

1.1 实习单位概况

本人于 2019 年 1 月到 2019 年 5 月在四川安若泰科技有限公司从事软件开发实习生工作。

成都安若泰科技有限公司是一家软件开发公司，作为软件实习生，其主要任务是完成公司所承接的软件项目开发工作，我的工作职责是参与需求分析与设计，进行 Web 前端系统的开发工作、部分后端开发工作和安全方案的设计与实现。

1.2 实习项目概况

本人所在的项目组从事的是某事业单位下属医学期刊编辑部的投稿与审阅系统的设计与建设工作。

系统的主要使用者为：该医学期刊编辑部的编辑、负责进行同行评议的审稿人、向期刊投稿的投稿人和负责系统基础信息维护的管理员。

该系统作为一体化平台，负责整合一篇文章从投稿开始，到被录用为止的整个期刊投审流程，包括：投稿、编辑初审、评审费缴纳、审稿人分配、同行评议、评议结果反馈、再次修改、再次评审、格式修改和版面费缴纳在内的投审业务流程以及配套信息的管理工作，包括学术分区设置、栏目设置、用户管理等。还有首页相关信息的展示工作，如公告发布、文件下载、最新期刊文章展示与下载、已发表文章检索等，以及必要的安全功能。

1.3 课题背景与国内外情况

在过去，论文的投稿常使用线下或邮件投递，投稿人、编辑和审稿人之间常使用邮件等其他通讯方式进行联系。整个业务流程需要多个分散、不同的部分共同协作完成，对于编辑来说，其可能需要使用即时通讯工具与投稿人联系；单独确认审

稿人缴费情况；通过邮件与匿名审稿人联系，发送稿件和获取审稿结论；与投稿人进行进一步反馈和修改等等，整个投审业务流程分散于不同的软件工具之中，且没有相互关联和统一的信息保存机制，提升用户的使用成本。

此系统的设计目标是改变原有论文投审体系中的人工投稿与编辑线下审查处理的模式，构建一个集中式信息处理平台，将论文投稿、编辑审稿、同行评议、发行等功能统一在一个平台上，实现一站式处理。线上一体化平台的实现，可以把用户从线下复杂的交流和散落在各处的资料中解放出来，集中化处理也方便用户对资料进行管理与分析，降低工作成本。统一化的报表处理也使得整个业务流程更加规范、完整，方便进行统计与分析。本系统将传统论文出版行业与互联网结合，促进了传统行业的发展。

国内外的学术期刊也有一些使用了网络处理平台，但大多仍然需要邮箱等传统通讯方式的参与，网络处理平台更多地是起到辅助的作用，并没有实际统领起业务流程。所以，尽可能地降低用户使用软件的数量，把业务逻辑集中在一起，降低用户的使用成本就显得很有必要。

1.4 毕业设计整体完成情况概述

在毕业设计中，主要完成了以下工作。

1. 医学论文投审系统的需求分析及需求分析说明书的制定；
2. 医学论文投审系统的概要设计与详细设计，以及设计说明书的制定；
3. 医学论文投审系统用户交互界面（前端）系统的开发与测试；
4. 医学论文投审系统部分服务器（后端）系统模块的开发与测试；
5. 医学论文投审系统在安全策略上的设计与实现。

本次毕业设计的难点在于，理清论文编辑部对论文投稿、审阅、缴费与出版等的整个业务逻辑，以及以本系统为核心业务处理环节，灵活运用多种目前使用较为普遍的工具（如微信、支付宝等），在多个关键时间节点及时提醒用户完成业务任务，提高业务处理的效率，在多个环节保障用户的个人信息、正在审阅中的论文内容不会被泄露，保证系统可以抵抗一定的恶意攻击。

第二章 复杂工程问题归纳与实施方案可行性研究

2.1 用户需求初步分析

在医学论文投审系统中，客户的需求如下：

1. 医学论文投审系统必须在公网下可用，面向该医学期刊编辑部内部的编辑、医学编辑部所聘请的进行同行评议的审稿人、提交稿件的投稿人和医学编辑部的系统管理员；

2. 该系统必须能够支撑起该编辑部对期刊审阅的整个业务流程，并具备一定的可扩展性；

3. 提供系统管理员的管理功能（非开放注册用户的添加，如编辑、审稿人账号；所有账户的删除与密码重置；基础信息的设置，如学术分区、每卷期刊的期数、每期的最大文章数、教育背景等；安全功能所需参数的设置；全编辑部的工作数据统计与图表化显示；公告的添加、删除、修改与管理；友情链接的添加、删除与修改；资料文件列表的添加、删除与修改等；首页展示证书图片的添加与删除；首页要展示的编委会成员信息的配置）；

4. 提供投稿人的基础功能（用户的注册、登录、密码重置流程、安全信息的设置；个人信息的修改与管理等）、向编辑部的投稿功能（包括文章元数据的上传、文章文件及附件的上传与管理等）、投稿后的业务流程支撑（如审稿费缴费与确认、审稿结论反馈、修改后的文章上传、重审反馈、格式修改确认、版面费缴费与确认等；该投稿人的所有相关稿件的状态获取以及单一稿件的业务状态记录等）；

5. 提供编辑的基础功能（用户的登录、密码重置流程、安全信息的设置；个人信息的修改与管理等）、投稿人投稿后的业务流程支撑（如投稿文章及相关附件的下载、初审、审稿费金额确定、以汇款方式缴纳审稿费的到款确认、审稿人分配、审稿结论的决定、修改后的再审、再审审稿结论的确定、版面费金额确定、以汇款方式缴纳版面费的到款确认；所有相关稿件当前状态的获取以及单一稿件的业务状态记录等）、该编辑的近期工作量统计与图表化显示等；

6. 提供审稿人的基础功能（用户的登录、密码重置流程、安全信息的设置；个人信息的修改与管理、银行卡信息的管理等）；审稿的业务流程支撑（显示当前所有的审稿任务、获取当前审稿任务的文章元数据和文章附件、上传审稿意见、拒绝

审稿任务）、查看任务量统计等；

7. 对于文章文件，提供在线阅读功能；

8. 提供首页功能：编辑部介绍、编辑部投稿须知、编辑部公告小列表/完整列表、公告内容及附件下载、当期最新期刊的元数据展示与下载、期刊根据关键词检索、期刊按期检索、编辑部编委信息的展示、文件中心的展示与下载等；

9. 有必要的安全保障，保护用户的个人信息与银行卡信息不会被泄露，保护未审阅完成的稿件不会被获得，具备抵抗多种攻击的能力；

10. 对于一些关键的时间节点，如提交审稿意见、缴费等，需要通过微信进行提示，提示用户登录系统进行操作；

11. 由于医学论文包含比较多的图片需要处理，编辑需要能根据图片文件确定可能包含该图片的文章，又由于使用云服务器作为系统搭载的平台，需要进行保密存储与搜索。

在该系统中，我负责所有的模块的前端设计与编码工作，以及需求点 7、9、10、11 的前后端设计与实现，还有上述各功能点在的安全保护策略。

2.2 待解决的工程问题

在本次毕业设计的系统中，对于本人的工作，涉及的工程问题有：

1. 由于系统部署在公网，必须保证其有一定的安全性，对于数据的传输与保存必须满足“机密性-完整性-可用性”的 CIA 三原则，并尽可能减小安全措施的使用对性能造成的损失；

2. 由于绑定了微信，通过微信接口访问系统的用户应获得与已登录用户相同的地位，应尽可能复用代码，减少代码量；

3. 因为对用户角色进行了划分，所以在前端和后端都要进行访问权限的限制，阻止用户去非法访问资源，同时对于分步执行的任务，应保证在未进行前一步的情况下，禁止对下一步页面的直接访问；

4. 在后台管理中，可能会在一个页面内多个控件之间进行数据传递和共享，在一个控件关闭后需要对临时数据进行保存，并且在业务流程结束后清空临时数据，系统需要对当前业务的任务执行状态进行记录并做正确的处理，由于数据流向上比较复杂，需要对控件的规模进行合理的划分，优化代码结构；

5. 由于前端系统需要与后端系统进行异步交互，且业务逻辑中有许多内容需

要得到数据后立即进行处理，所以需要在部分异步交互的过程中为用户渲染载入界面和进行同步化，避免组件运行时出现数据无法正常获取的问题；

6. 在用户登录的模块下，需要处理自动登录的 cookies 和登录状态过期后的续期；

7. 需要对业务逻辑根据用户角色进行划分，在每个角色上设置与自身相关的业务逻辑，并尽可能使用统一的页面风格处理位于不同业务状态的业务。

2.3 复杂工程问题的解决方案和可行性分析

在毕业设计所涉及的系统中，系统用户主要分为以下四种角色——系统管理员、投稿人、审稿人和编辑，按照不同角色为其提供不同的功能，可行性上，由于分角色管理权限在管理系统上使用非常普遍，而且也满足用户对于权限控制的要求。在前端设计上，在登录时将权限角色存储在 localStorage 中，利用路由表对每个角色可以访问的模块进行控制，不符合要求的访问将重定向到 403 错误界面。

该系统将全部使用 HTTPS 协议进行访问，保护传输路径上的数据安全。客户已经决定向 CA 机构购买证书，可以使用此证书执行 HTTPS 保护。

对数据库中的用户密码进行散列化加盐处理，盐值将尽可能地长且随机。可行性上，其属于业内常用方法，且设计过的复杂盐值可以保证用户密码的安全性。

将数据合法性验证和权限判断逻辑交由前后端分别执行，起初是决定将合法性验证交由前端进行，但后来在模拟渗透时发现，即使通过 HTTPS 传输数据包，客户机依然可以通过 BurpSuite 等数据包嗅探工具通过 HTTPS 代理的方式获得 HTTPS 报文内容并进行修改后转发给服务器，这可以绕过前端对数据合法性的校验，对于服务器的安全是十分危险的，所以对于数据的校验将由前端和后端分别进行。

在微信登录上，需要熟悉微信所提供的 API 文档，但在后文中会叙述，由于版本的不同，微信提供的 API 文档有些接口与实际系统返回的参数不同，这在设计和实现时需要特别注意。需要熟悉的 API 接口有：OpenID 的获取、微信消息的发送等。

在跨组件的数据保存上，由该页面的类作为父类，实现该页面各个功能的分组件形成子类，以点击所弹出的作为表单的子窗口 Modal 为例，该 Modal 设置为每一次退出时都会被销毁。在每次点击按钮时，父类将设置 Modal 的可见性为 true，并把自身 state 中所存储的 Modal 的表单值作为参数传给 Modal，Modal 根据 props

中所存储的参数为表单项赋值，在表单没有点击提交就关闭的情况下，Modal 会调用父类传入的用于操作父类 state 的方法，利用当前的表单值更新父类 state，之后再销毁自身，这样 Modal 的表单值就存储在了父类的 state 中，下次打开 Modal 时，父类会把 state 中所存储的临时表单值赋给 Modal，显示上实现了记忆功能。父类还负责此页面所有的异步交互工作，在父类传入表单值和操作 state 的方法的同时，还会传入负责进行通讯的方法，在点击完成后，Modal 会把表单值作为参数执行通讯方法，通讯方法在执行完毕后，会清空自身存储临时表单值的 state，下次再显示 Modal 时，初始值会为空，所以表单值也为空，为一个新表单。

在异步通讯上，若需要组件共享的内容，将使用 dva.js 进行异步交换和数据存储。它是一个基于 redux 的框架，基于命名空间进行管理，异步交换后，可以通过 reducer 来将交换获得的数据存储在该命名空间的 state 中，所有控件均可以将命名空间作为 key 来访问对应命名空间中的 state 内容，可以很方便地实现数据共享。在非交换环境下，使用 fetch 接口来进行异步交换，fetch 基于 Promise，通过 Promise 可以将异步操作同步化，从而实现获得数据之后的立即处理。

对于自动登录，在登录界面会设置是否设置记住密码，若设置，前端系统会将用户名和密码的 SHA-256 散列值存储在 cookies 中，登录页面在加载时，会先检测 cookies 对应的表单项是否存在，若存在则将用户名和密码自动填写到表单框中，并设置标记不进行 SHA-256 散列化。

对于统一的风格，以显示一篇文章的当前状态为例，在所有的业务环节中，均可以显示文章的元数据和提供下载接口。根据业务当前的状态，在下方渲染进度提示和不同的操作表单，统一了在不同操作步骤下的页面风格，均为页面区上方元数据，中部当前进度提示条，下方为操作表单。

2.4 详细需求分析

我们项目组采用了采访、实地调查等手段，与该机构医学编辑部的工作人员进行了交流，实地参观了编辑部的业务处理流程。在初步需求分析的基础上，进一步细化了对用户需求的分析，包括用户对功能点的要求和一些约束的要求，以及实现的优先级。

在该设计项目中，使用系统的人员根据角色被分为五类用户：系统管理员、投稿人、编辑、审稿人和游客，其中游客的权限是前面四种用户的权限子集，不需要

用户进行登录，故不作专门说明。

系统用例图如图 2-4-1 所示。

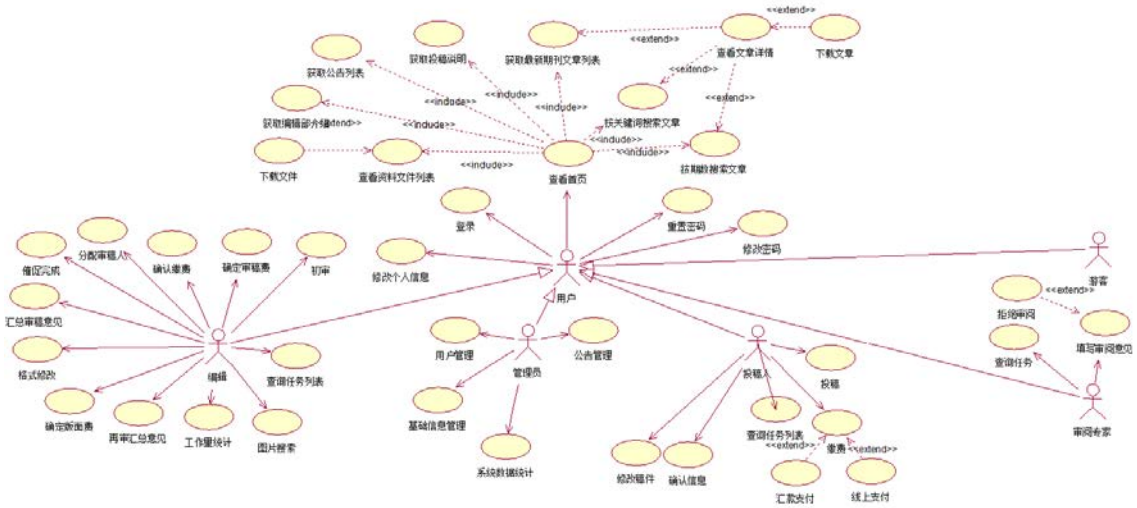

图 2-4-1 系统用例图

对于整个投审流程，其业务顺序如下所述：

1. 投稿人在系统中注册账号，获得投稿权限（只需执行一次）；

2. 投稿人在稿件投稿模块中，填写关于稿件的各项基本信息，提交稿件文件，系统为其生成业务编号，进入系统业务处理流程；

3. 系统为其自动分配编辑，编辑获得该稿件的处理任务，编辑对稿件进行初审，通过初审的进入下一步业务处理环节，未通过的业务自动退出业务流程；

4. 通过初审的稿件，将进入待提交审稿费环节，投稿人可以通过汇款方式进行支付，并在支付完成后填写发票邮寄信息；

5. 待状态验证通过后，系统将为其分配一到两名审稿人，进行同行评议工作，稿件业务状态进入审稿状态，投稿人和审稿人保持双盲；

6. 审稿人登录系统，下载稿件，在系统中填写审稿意见表，确定审稿意见，意见分为通过、不通过和待修改三种，审稿意见为不通过的，其业务推出业务流程；

7. 审稿意见为待修改的，系统将为投稿人反馈修改意见，投稿人将可以向编辑进行一次解释，回答审稿意见中的问题，编辑将会综合回答问题的情况反馈最终修改意见，投稿人在根据审稿意见整改后，上传最新版本的待审阅稿件；

8. 投稿人上传了新的待审阅稿件后，系统将为其再次进行同行评议工作，审稿人再次填写审稿意见表，确定审稿意见，但审稿意见只有通过（格式修改）和

不通过两种，最终裁定为不通过的稿件将退出审稿流程；

9. 最终通过审阅的业务，将进入格式修改环节，投稿人将根据编辑的格式修改意见，完成最后的格式修改工作，并将最终版稿件提交到系统；

10. 编辑在系统中完成确认工作，进入投稿人确认状态；

11. 投稿人确认后，业务进入待缴版面费状态，投稿人可以通过汇款方式进行支付，并在支付完成后填写发票邮寄信息，系统确认缴纳成功后，业务处理流程结束。

业务在整个业务流程中，将经过多种状态，业务的状态机图如图 2-4-2 所示。

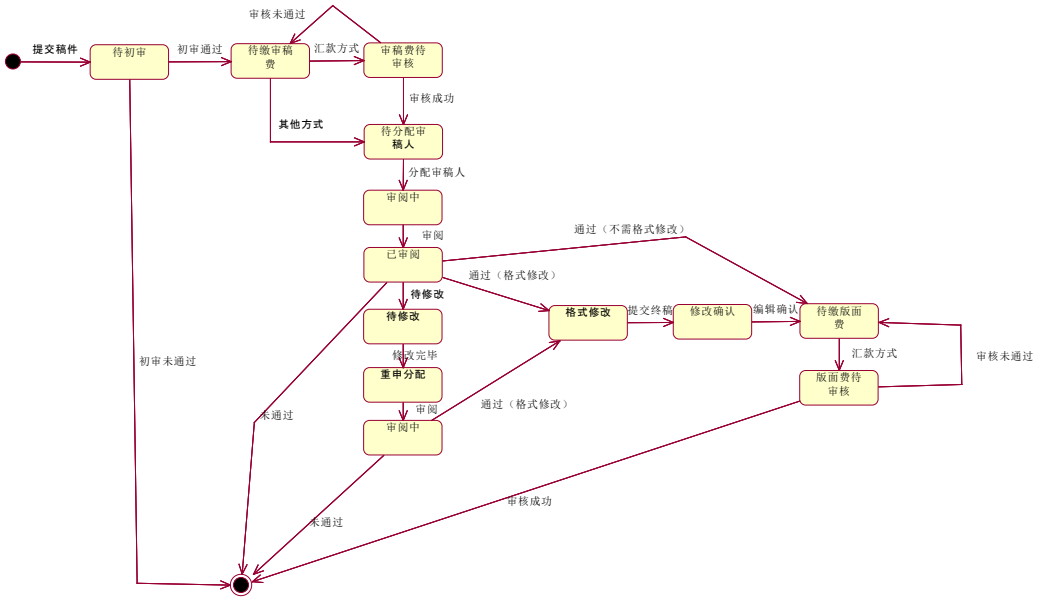

图 2-4-2 业务流程状态机图

对于前端系统，整个系统主要分为两部分。

一是首页部分，首页用于向游客展示编辑部介绍、投稿须知、编辑部公告、最新期刊的文章列表显示、下载期刊的指定文章、编辑部相关证书的展示和编辑部与投稿审阅相关的资料文件的下载等，是面向游客的重要窗口。所有在未登录情况下可以执行的功能均需要在首页部分进行设计。

以下是首页的需求功能点表格，如表 2-4-1 所示。

表 2-4-1 首页的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	期刊检索	PC 端	对符合查找关键字的期刊文章进行查找	分成两类查找。 按关键字查找：关键字类型（标题、学术领域、中文关键词、英文关键词、作者名、中文摘要、英文摘要）、关键字、起止时间 按期数查找：指定期刊期数	符合条件的文章列表：显示文章标题、作者和出版时间	1
2	文章详细信息	PC 端	显示指定文章的详细信息	文章的内部 id	文章标题、作者列表、文章学术领域、发表时间、中文关键词、英文关键词、中文摘要、英文摘要、在线显示、下载链接	1
3	编辑部公告列表	PC 端	显示每期期刊的发布公告以及其他告示列表	无	显示所有公告的列表	1
4	编辑部公告显示详细信息	PC 端	显示指定公告的具体内容	公告的内部 id	公告标题、发布时间、详细内容、图片以及附件下载	1
5	最新期刊列表	PC 端	显示最新期刊的文章列表	无	最新一期期刊的文章列表、作者和发表时间	1
6	期刊介绍	PC 端	显示该期刊预先设置好的期刊介绍	无	预先设置好的期刊介绍	1

7	资料中心	PC 端	显示文章投稿、审阅所需的资料的列表与下载功能	下载功能:资料的id	所有资料的列表，以及对应每一个资料的下载链接	1
8	投稿须知	PC 端	显示给游客的投稿提醒	无	预先设置好的投稿须知	1
9	编委会列表	PC 端	显示给游客的编委会成员信息	无	每一位编委会成员的照片以及相关介绍	1
10	编辑部证书	PC 端	显示编辑部的证书	无	走马灯形式展示编辑部的证书	2
11	联系我们	PC 端	显示编辑部的联系方式	无	编辑部的联系方式	2

其中，本人负责所有需求点前端系统的设计与编码。

二是后台系统部分，此为系统的主要部分，也是业务逻辑主要实现的部分。

后台系统共分为四类角色——系统管理员、编辑、审稿人和投稿人，下面按照四种角色分别进行需求分析。

对公共需求进行分析，上述四种角色均需要以下功能：登录、修改密码、忘记密码、个人信息修改。

登录时用户需要提供用户名、密码和所登录的角色，提交系统进行验证，在验证身份时，需要提供验证码信息做安全检测。登录时应具备记住密码选项，在第一次登录后可以实现记住密码功能。

修改密码时，用户需要保持登录状态，并在修改密码之前对预先设置好的三个安全问题进行验证，只有验证通过后才允许修改密码。

忘记密码时，用户可以使用验证登录问题或进行微信验证，验证通过后可修改口令。

个人信息的修改包括基础个人信息的修改、学术领域的修改（投稿人、审稿人）等，个人信息的修改在登录状态进行提交即可。

对于审稿人，其个人信息会包含用于接收审稿费的银行卡信息，银行卡信息需要在验证安全问题之前保持不可见状态，验证后才可进行修改。

微信催促功能将在当前业务状态即将到达处理截止日期时由系统自动激活，系统将通过微信发送通知消息给用户。对于编辑，其可以通过在每个业务的详情界面点击“提醒”按钮主动发出提醒消息。

功能点如表 2-4-2 所示。

表 2-4-2 公共的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	登录	PC 端	拥有登录权限的用户通过系统入口进行登录	用户名、密码、角色、验证码	登录结果（成功/失败）	1
2	修改密码	PC 端	已经登录的用户对自己的密码进行修改	安全问题答案、旧密码、新密码、新密码重复	修改结果（成功/失败）	1
3	忘记密码	PC 端	未登录且忘记密码的用户重置自己的密码	邮箱、角色、安全问题答案、新密码、新密码重复	重置结果（成功/失败）	1
4	个人信息修改	PC 端	已经登录的用户，修改自己的个人信息	新的个人信息（姓名、联系方式等） 审稿人的银行卡信息：安全问题答案、新的银行卡信息	修改结果（成功/失败）	1
5	微信催促	PC 端	在临近阶段处理截止日期时，系统将自动通过微信发送通知提醒；编辑也可以主动发出提醒	到达规定时间的提醒：无 编辑的提醒：点击“提醒”按钮	发送微信提醒消息	2

其中，本人负责所有需求点前端系统的设计与编码，以及各需求点的安全策略设计和需求点 5 的后端实现。

对于系统管理员，主要需要实现的功能有：

1. 首页公告管理：管理员可以对首页显示的公告列表进行管理，包括增加公告、删除公告、修改公告、上传附件和图片。

2. 基础数据管理：管理员需要可以配置包括学术领域表、教育水平表、地点表、专业表在内的多种变量表，并可以配置安全模块的基础参数。

3. 账号管理：可以查看其它账号的个人信息，对遗忘的密码进行重置，删除无效的账号，添加新的编辑和审稿人账号等。

4. 数据统计：可以对系统的数据，例如访问次数、稿件学术领域分布、稿件数量变化数据，进行统计和可视化显示

系统管理员的需求功能点如表 2-4-3 所示。

图 2-4-3 系统管理员的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	显示公告列表	PC 端	管理员显示当前所有的公告列表	无	公告列表，包含标题等信息	1
2	新增公告	PC 端	管理员新增一条公告	公告标题、公告富文本、附件	添加结果（成功/失败）	1
3	删除公告	PC 端	管理员删除公告	公告的内部 id	删除结果（成功/失败）	1
4	修改公告	PC 端	管理员打开已经发布的公告，修改其内容	新的公告的标题、公告富文本、附件	修改结果（成功/失败）	1
5	基础信息增加	PC 端	管理员对基础信息的变量表添加信息	基础信息类别、值	添加结果（成功/失败）	1
6	基础信息修改	PC 端	管理员对基础信息的环境表量表修改信息	基础信息类别、新值	修改结果（成功/失败）	1

7	基础信息删除	PC 端	管理员删除基础信息的环境表量表中的值	基础信息类别、该基础信息对应的 id	删除结果（成功/失败）	1
8	修改个人信息	PC 端	管理员可以修改其他用户的个人信息	用户内部 id，新的个人信息	在打开时先返回原有的个人信息，修改后返回修改结果（成功/失败）	1
9	重置密码	PC 端	管理员可以重置其他用户的安全信息	用户内部 id，新的密码	返回修改结果（成功/失败）	1
10	添加新账号	PC 端	管理员可以根据信息，添加非公开注册的账号（编辑、审稿人），添加账号应支持批量导入	用户信息	返回添加结果（成功/失败）	1
11	删除无用账号	PC 端	管理员可以删除不再使用的账号，删除应可以执行批量删除	用户内部 id	返回删除结果（成功/失败）	1
12	数据统计	PC 端	管理员可以对系统的访问情况、稿件的领域分布、编辑部总体工作量变化等进行统计和可视化显示	点击相应的数据统计按钮	显示数据统计表格或图表可视化信息	2

其中，本人负责所有需求点前端系统的设计与编码和批量添加账号功能的后端实现，以及各需求点的安全策略设计。

对于编辑，他应该通过系统完成分配的任务。每个编辑可以查询当前分配给

自己的业务并进行操作。对于编辑，其主要工作是在业务流程可以进行的操作如下所述。

1. 业务流程功能：

（1）初审：根据要求，对稿件进行初步审查，审查是否符合初审要求，对于符合要求的稿件，予以通过；对于不符合的，予以退回。

（2）确定审稿费缴费金额：对于通过初审的稿件，编辑将需要通过系统确定需要缴纳的审稿费金额和确定需要参与同行评议的审稿人数量（1~2 名）。

（3）确认审稿费缴纳情况：对于使用线下汇款方式进行缴费的业务，编辑需要在系统中确认审稿费是否已经成功缴纳。对于所有缴费方式，均需确认发票邮寄信息。

（4）分配审稿人：系统将自动根据业务所确定的自身学术领域，结合审稿人预先设定的学术领域，生成符合要求的审稿人列表，编辑将根据之前确定的审稿人数量，为其选择适当的 1~2 名匿名审稿人。

（5）作出审稿结论：审稿人反馈同行评议结论后，编辑需结合该 1~2 名匿名审稿人的意见，在系统中提交对这篇文章的审稿结论以及意见，审稿结论分为“通过（不需进行格式修改）”、“通过（需要进行格式修改）”、“待修改”和“不通过”。其中，结论为“待修改”和“不通过”的，编辑需要提交审稿意见供投稿人参考。

（6）审稿意见复议：审稿结论为“待修改”的，投稿人有一次机会对审稿意见中的问题在系统中进行回复，编辑可以在系统中查看投稿人的回复，根据回复情况，对当次审稿意见进行修改并重新提交，但是审稿结论不可变动。

（7）再审前置确认：投稿人上传修改后的文章后，编辑需要对修改后的文章进行初步确认，确认后系统将把文章发给第一次审稿的审稿人进行再审工作。

（8）作出最终审稿意见：在审稿人反馈再审审稿结论后，编辑需结合该 1~2 名匿名审稿人的再审意见，在系统中提交对这篇文章的最终审稿结论，审稿结论分为“通过”和“不通过”。

（9）格式修改：当第一次审稿结论为“通过（需要进行格式修改）”和再审结论为“通过”时，编辑需要在系统中提交符合编辑部期刊格式规范的文件。

（10）确定版面费缴费金额：当作者确认了格式修改后的文章后，编辑需要在系统中提交此次版面费的应缴金额。

（11）确认版面费缴纳情况：对于使用线下汇款方式进行缴费的业务，编辑需要在系统中确认审稿费是否已经成功缴纳。对于所有缴费方式，均需确认发票邮寄信息。

2. 系统功能：

包括工作量统计、获取目前需要处理的业务列表、获取该名编辑涉及到的所有业务的当前状态、从图片寻找对应的文章业务编号。

编辑的需求功能点如表 2-4-4 所示。

图 2-4-4 编辑的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	所有业务列表	PC 端	显示所有业务列表，可以进行根据状态筛选	无	显示该编辑的所有业务（默认显示待处理的业务）	1
2	初审	PC 端	编辑可以查看该稿件，并给出初审结论	业务 id、初审结论	初审结果	1
3	确定审稿费缴费金额	PC 端	初审通过后，编辑需要提交确定的审稿费金额	业务 id、审稿费金额	提交结果	1
4	确认审稿费缴纳情况	PC 端	投稿人使用汇款方式缴费后，编辑可在系统内确认缴费情况和发票信息	业务 id、确认选项	确认结果	1
5	分配审稿人	PC 端	编辑可以为业务分配审稿人，进行下一步工作	业务 id，审稿人 id	分配结果	1
6	作出审稿结论	PC 端	编辑可以根据审稿人的意见，作出最终的审稿结论	业务 id，审稿结论	审稿结果	1

7	审稿意见复议	PC 端	编辑可以根据投稿人对审稿意见的回复，重新确定一次审稿意见	业务 id、新的审稿意见	审稿结果	1
8	再审前置确认	PC 端	投稿人上传了修改后的稿件文件后，编辑需要对修改后的文件进行确认	业务 id、再审前置确认结论	再审前置确认结果	1
9	作出最终审稿意见	PC 端	编辑可以根据审稿人的意见，得出最终的审稿结论	业务 id，审稿结论	审稿结果	1
10	格式修改	PC 端	编辑需要根据编辑部格式要求，对初审状态为“通过（需要进行格式修改）”和再审状态为“通过”的文章进行格式修改，并上传文件	业务 id、格式修改后的文件	提交结果	1
11	确定版面费缴费金额	PC 端	在格式修改结果被投稿人确认后，编辑需要在系统中提交投稿人要缴纳的版面费金额	业务 id、版面费金额	提交结果	1

12	确认审稿费缴纳情况	PC 端	投稿人使用汇款方式缴费后，编辑可在系统内确认缴费情况和发票信息	业务 id，确认结果	确认结果	1
13	工作量统计	PC 端	编辑可以查看自己在一定时间区间内的工作量统计和变化图表，以及接手的稿件所在领域的比例	时间区间	统计数据和/或图表	2
14	由图片反查所在业务 ID	PC 端	编辑可以通过一张图片反查其所在的业务 ID 或存在相似图片的业务 ID	待查图片文件	搜索结果	2

其中，本人负责所有需求点前端系统的设计与编码，以及各需求点的安全策略设计和需求点 14 的设计与实现，还有需求点 1 设计的改进。

对于审稿人，虽然属于编辑部以外的人员，但是其仍将参与核心的投审业务流程，审稿人主要需要获得其任务列表，并填写审稿意见。特别地，审稿人可以拒绝审稿任务要求。审稿人也可以获得自己的工作量统计数据。审稿人的需求功能点如表 2-4-5 所示。

图 2-4-5 审稿人的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	接受/拒绝审稿任务	PC 端	审稿人在面对审稿任务时，可以选择接受或者拒绝	业务 id、接受/拒绝	处理结果	1

2	查看文章	PC 端	对于接受的任务，审稿人可以下载该稿件（保持双盲）	业务 id	文件	1
3	填写审稿意见	PC 端	对于接受的任务，审稿人需要填写审稿意见供编辑处理	业务 id、审稿结论、审稿意见	处理结果	1
4	所有业务列表	PC 端	显示所有业务列表，可以进行根据状态筛选	无	显示该编辑的所有业务（默认显示待处理的业务）	1
5	工作量统计	PC 端	审稿人可以查看自己在一定时间区间内的工作量统计和变化图表	时间区间	审稿记录	2

其中，本人负责所有需求点前端系统的设计与编码，以及各需求点的安全策略设计和需求点 4 设计的改进。

对于投稿人，他可以通过首页的入口注册新账号，注册账号并填写必要的信息后，可以参与投审环节。

1. 业务流程功能：

（1）投稿：已经注册账号的投稿人，可以在系统中提交文章元数据（题目、作者、学术领域、摘要、关键词等）、基金证明文件（如有）和去除作者信息之后的文章文件及其附件，该篇文章会成为一个业务并进入系统的业务处理部分。

（2）缴纳审稿费：在业务通过初审后，编辑会确定要缴纳的审稿费金额，投稿人可以通过汇款的方式缴纳指定的金额，并将汇款凭据和发票信息提交到系统中。

（3）提交复议：当第一次审稿意见为“待修改”时，投稿人可提交复议，按照初次审稿意见对意见进行回复，把回复在系统中提交。

（4）提交初次修改文件：在第一次审稿结论正式反馈后，若状态为“待修改”，则投稿人可以将根据审稿意见修改后的文件在系统中提交。

（5）确认格式修改文件：当第一次审稿后的状态为“通过（需要进行格式修

改)”或第二次审稿通过后，编辑会对文件按照编辑部格式要求进行格式调整，投稿人需要在系统中下载编辑格式修改后的文件，并在系统中确认修改，进入下一流程。

（6）缴纳版面费：在确认格式修改文件后，编辑会确定要缴纳的版面费金额，投稿人可以通过汇款的方式缴纳指定的金额，并将汇款凭据和发票信息提交到系统中。

2. 系统功能：

投稿人可以查看现在待处理业务的列表以及所有投稿过的业务的列表，以及自己投稿分区的分布以及处于通过、审查中以及待通过的业务的比例示意图。

投稿人的需求功能点如表 2-4-6 所示。

图 2-4-6 投稿人的需求功能点

序号	功能名	功能配置	功能简述	输入	输出	必要性
1	注册	PC 端	投稿人需要通过首页的注册接口获得参与业务流程的权限	邮箱、密码、姓名、姓名(拼音)、性别、地址、邮政编码、工作单位（中文)、工作单位(英文)、专业、教育信息、职位、办公室电话、联系电话、地点、研究方向、学术分区 1、学术分区 2、学术分区 3、个人介绍以及三个安全问题及其答案	注册结果（成功/失败）	1
2	投稿	PC 端	投稿人在注册后，可以提交自己的稿件供编辑审核	投稿须知确认选项、版权转让协议书确认选项、稿件标题、分类、关键词、摘要、通讯作者、基金信息元数据、稿件文件、基金批文扫描件	提交结果	1

3	缴纳审稿费	PC 端	可以选择线下汇款方式支付审稿费，并在支付完成之后填写发票信息，汇款需要提供汇款单扫描件	共用：发票抬头、纳税人识别号、发票邮寄地址和发票邮寄收件人 线下汇款：汇款单扫描件	缴费结果和/或信息提交结果	1
4	提交复议	PC 端	在初次审稿结束后，若状态为“待修改”，可根据审稿意见在系统中提交答复给编辑	审稿意见答复	提交结果	1
5	提交初次修改文件	PC 端	在初次审稿意见最终决定后，若状态为“待修改”，投稿人可根据最终初次审稿意见对文章进行重新修改并上传到系统	修改后的文件	处理结果	1
6	确认格式修改文件	PC 端	编辑完成格式修改工作后，投稿人需在系统中确认修改的文件	确认格式修改结果	处理结果	1

7	缴纳版面费	PC 端	可以选择线下汇款方式支付版面费，并在支付完成之后填写发票信息，汇款需要提供汇款单扫描件	共用：发票抬头、纳税人识别号、发票邮寄地址和发票邮寄收件人 线下汇款：汇款单扫描件	缴费结果和/或信息提交结果	1
8	所有业务列表	PC 端	显示所有业务列表，可以进行根据状态筛选	无	显示该投稿人的所有业务（默认显示待处理的业务）	1

其中，本人负责所有需求点前端系统的设计与编码，以及各需求点的安全策略设计和需求点 8 设计的改进。

除了上述的通用需求和各角色额外的功能需求之外，系统还有额外的安全需求。额外的安全需求如下。

1. 由于系统可能部署在云服务平台上，为了避免被攻击或云平台盗取尚处在业务流程中未出版的文章文件，需要对文章进行加密，并保证拥有获取权限人员的可用性。

2. 由于审稿人的银行卡信息存储在数据库中，需要保证审稿人的银行卡信息只可被其他用户阅读，不可被篡改。

3. 由于投稿人账号属于公开注册账号，需要对批量注册作出一定的限制。

4. 由于图片也属于未出版文章文件的一部分，由于第 1 条的原因，图片也需要加密存储并能够正常实现搜索功能。

5. 在上述需求功能点中，以保护用户个人信息不被非法获取或篡改为目标，对需求功能点执行一定程度上的安全保护。

第三章 针对复杂工程问题的方案设计与实现

3.1 针对复杂工程问题的方案设计

根据客户的要求，此系统按照 Web 项目开发，使用 B/S 架构进行设计。该系统在设计与实现上前、后端分离，后端采用 Java 语言，使用 Spring Boot 作为后端服务器系统框架，使用 Apache MyBatis 作为数据库驱动接口，使用 MySQL 作为数据库管理系统。

对于前端系统，react.js 已经是业内普遍所使用的 JavaScript 开发框架，其组件生态比较完善。Ant Design 作为基于 react.js 的组件集，其组件样式美观，也易于定制，非常适合前端设计。而其衍生框架 Ant Design Pro 也提供了非常易于控制的组件集和布局。

前端使用 umi.js 进行路由控制，动态路由可以根据 URL 动态渲染页面，嵌套路由可以很方便地使用布局页面，将具体的功能页面作为布局页面的子页面进行渲染，通过布局页面控制系统的大致结构，并使样式统一化，还可以使用 umi.js 对根据用户的权限进行路由控制，限制用户的非法访问。

在数据驱动上，使用 dva.js 作为前端的数据驱动接口，dva.js 是基于 react-redux 的数据驱动框架，其 state 位于 react 顶层，可以供各组件共享数据。其使用命名空间对该 state 进行划分，并使用命名空间作为 key，获取该命名空间 state 中所存储的信息。组件可以通过 connect 函数把 state 中的信息映射到组件的 props 中，组件通过 props 获取 dva.js 内的 state 信息。

在 dva.js 中，使用 effects 执行异步通信，即前端系统可以通过 effects 中的函数，访问后端系统异步获取结果，并使用 reducers 更新该命名空间的 state，实现数据更新。

3.1.1 首页部分的方案设计

3.1.1.1 期刊检索的方案设计

期刊检索是首页的重要功能，供所有用户可以检索已发布的文章，前端系统将

根据数据结果渲染符合查找要求的文章列表，并提供每篇文章元数据的显示和下载。

根据需求分析功能点，期刊检索分为两种检索模式——按关键字搜索和按期数搜索。在设计上，为方便用户使用，应把搜索输入框和搜索结果放在一个页面下显示；两种搜索方式也应尽量放在一个页面中，方便用户进行操作。

若用户通过关键字搜索，用户将可以通过标题、学术领域、中文关键词、英文关键词、作者名、中文摘要和英文摘要等七种搜索方式对已发表的文章进行检索，在检索时，需要设置搜索的时间范围，提交到后端系统进行处理，后端系统取得搜索结果的第一页返回给前端系统。

若用户通过期数搜索，用户将使用级联下拉菜单选择年份，前端系统需要在用户选择年份时请求后端系统，获取当前年份的期数信息并在级联选择器中进行渲染，这样可以避免不必要的数据加载，提高速度。获取到年份和期数后，提交到后端系统进行处理，后端系统去的搜索结果的第一页返回给前端系统。

前端渲染后，对于每一个搜索结果项，需要渲染链接指向详情页，文章详情页将显示文章的元数据和下载链接。将由前端对搜索结果进行分页，前端在获得数据总页数后，渲染分页器总数量，在分页切换时重新发送搜索请求和页码，取得该页的结果信息。

在此部分需要注意，由于两种搜索方式共享一个搜索结果页面容器，在搜索框提交搜索结果时，需要记录当前所使用的搜索方式和所使用的关键词，以使得分页器在页码变换时可以向后端提交正确的搜索请求。

按关键字搜索方式的时序图如图 3-1-1-1-1 所示。

按期数搜索方式的时序图如图 3-1-1-1-2 所示。

3.1.1.2 文章详细信息的方案设计

文章详细信息页面是为首页的使用用户（即所有用户）提供已出版/完成业务流程的文章的详细信息，包括文章的作者、文章领域、发表时间、中文关键词、英文关键词、中文摘要、英文摘要和下载接口。文章详细信息页面会通过 URL 传入文章 id，页面通过 id 向后端请求详细信息。

由于文章领域是基础信息，在数据库中以数值 id 存储，需要从后端数据库中取得数值 id 所对应文章领域的文字内容。所以在页面加载时，需要先提前取得后端学术领域的对应表，在获得文章数据之后对 id 进行翻译。由于是基础信息并存放在 dva.js 中，可使用时间戳验证是否被修改过，若未修改则不传送完整信息。

文章详细信息的时序图如图 3-1-1-2-1 所示。

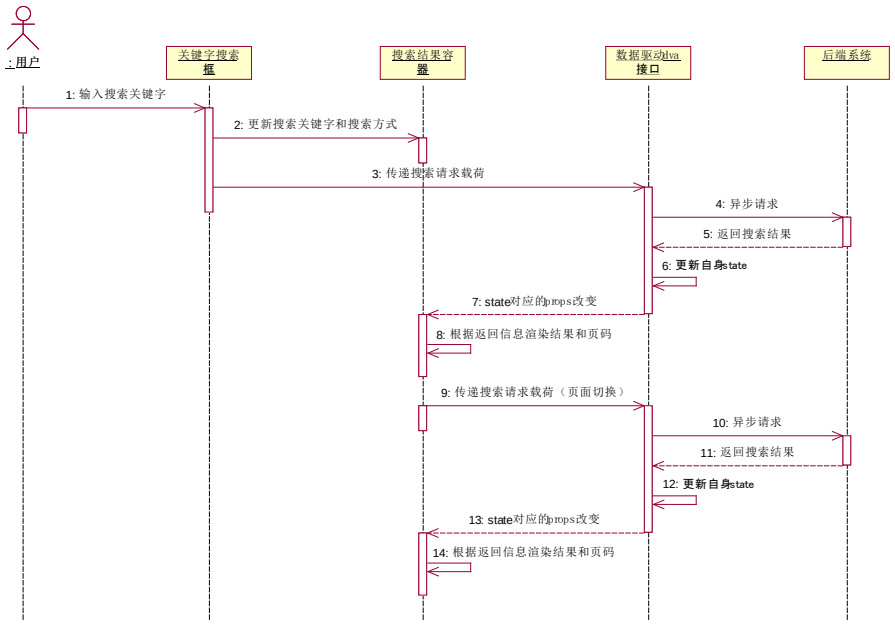

图 3-1-1-1-1 按关键字搜索方式时序图

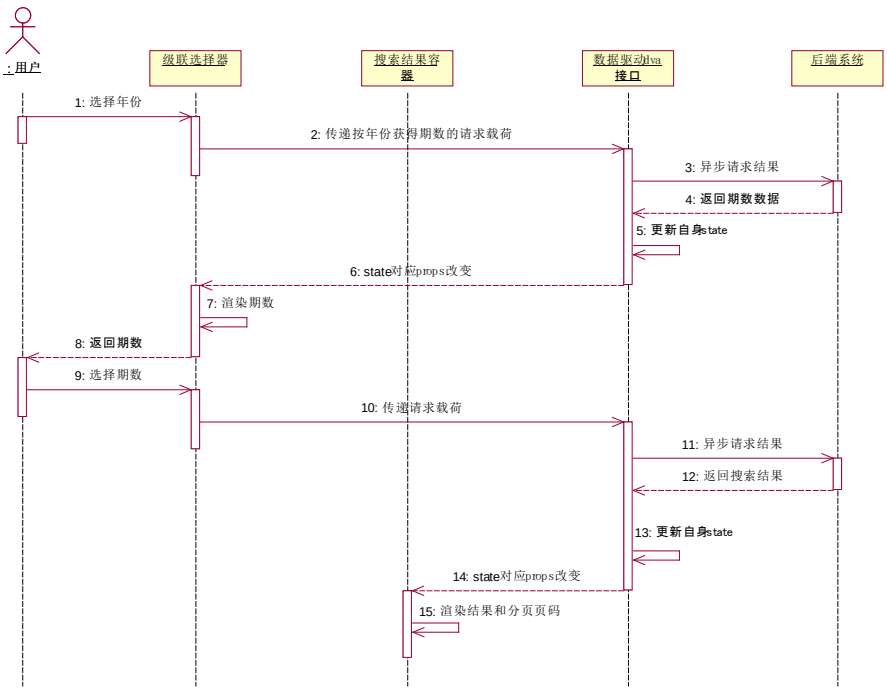

图 3-1-1-1-2 按期数搜索方式时序图

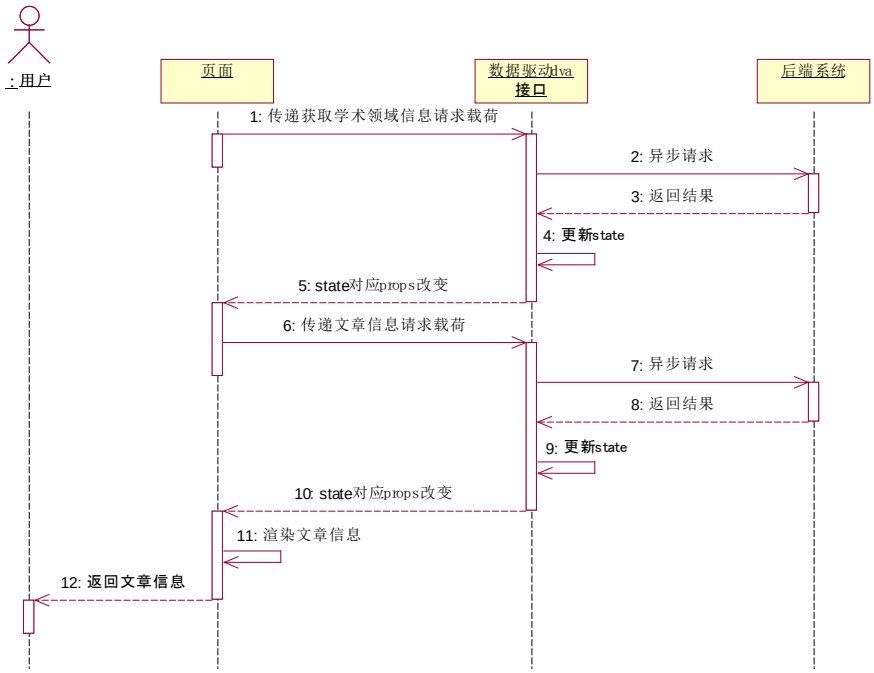

图 3-1-1-2-1 文章详细信息时序图

3.1.1.3 编辑部公告列表与编辑部公告列表详细信息的方案设计

编辑部公告列表分为两部分，一是显示在首页页面的公告列表，该公告列表只显示最新的前五条公告；二是显示全部的公告列表，该页面通过首页页面的公告列表“显示更多”入口进入，负责显示所有的公告列表。

两个页面均通过后端数据接口获取公告列表，并为每一条列表项渲染显示详情的链接与发布时间。

显示公告详情的页面通过页面的 URL 获取公告 id 作为参数，通过后端数据接口获得公告详情后对公告富文本进行渲染，并设置附件下载接口。

显示公告列表的时序图如图 3-1-1-3-1 所示。

显示公告详情的时序图如图 3-1-1-3-2 所示。

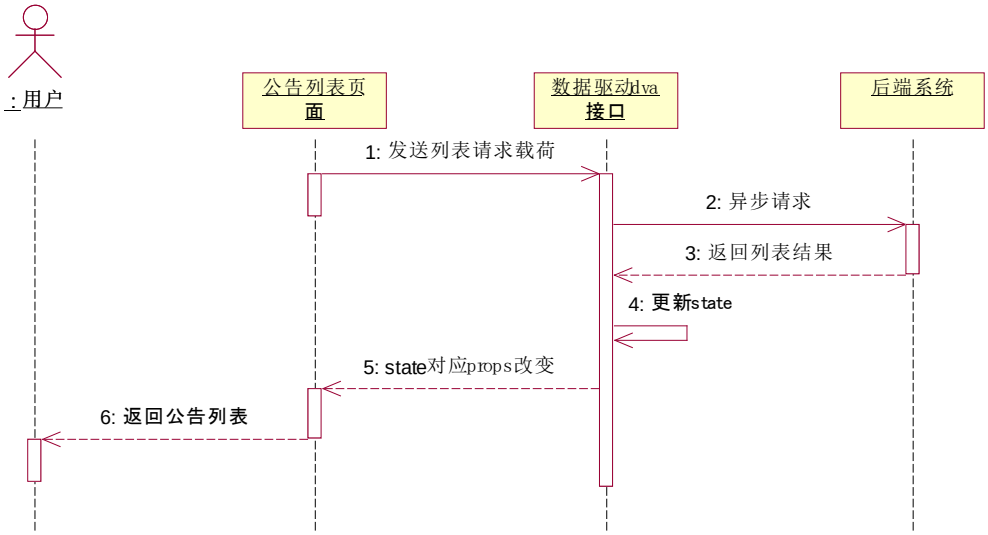

图 3-1-1-3-1 获取公告列表时序图

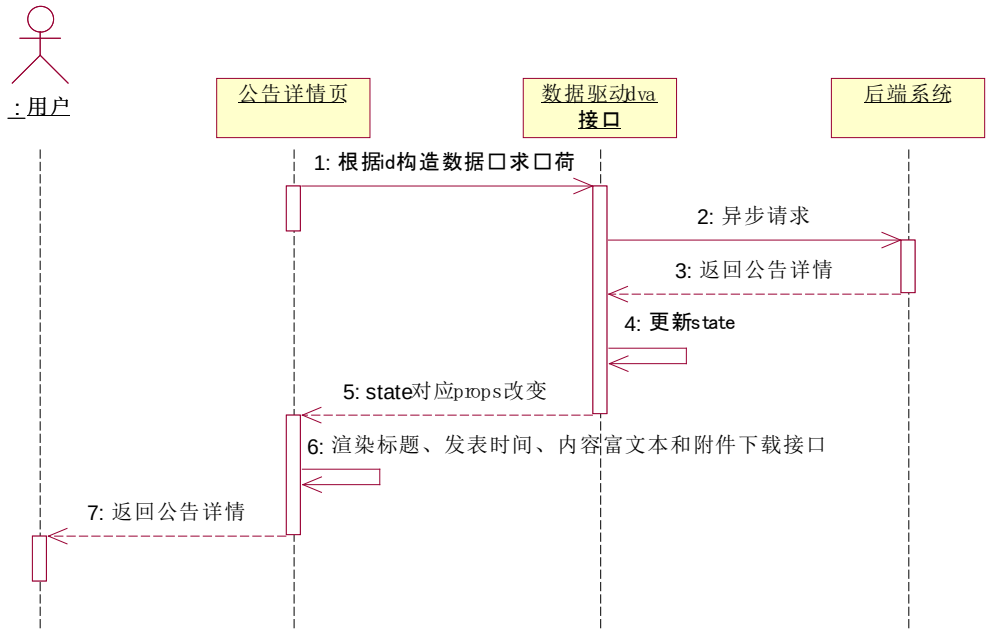

图 3-1-1-3-2 公告详情时序图

3.1.1.4 最新期刊列表的方案设计

最新期刊列表是显示在首页页面上的一个组件，负责显示当前最新一期期刊的

文章列表，其设计除后端请求接口不同、只需传递页码参数外，列表的渲染设计可复用 3.1.1.1 节期刊检索中的文章列表渲染方式，页面在加载时利用 dva.js 向后端请求最新期刊的文章列表第一页，并进行渲染，分页器设置亦可采用 3.1.1.1 节的分页方式，在分页器当前页码改变时，向后端接口请求新页面的数据并进行渲染。

3.1.1.5 期刊介绍与投稿须知的方案设计

期刊介绍和投稿须知是在首页页面显示的两个组件，主要作展示用，用以在首页向所有使用者展示期刊编辑部的介绍文本和投稿须知文本，大小分别约为首页页面的六分之一左右。

由于期刊介绍文本和投稿须知文本可能会随着时间而需要修改，所以期刊介绍文本和投稿须知文本会存放在数据库中，使用后端接口从后端获取数据后在前端进行渲染，并在后台系统中为管理员提供设置入口。

期刊介绍和投稿须知的时序图如图 3-1-1-5-1。

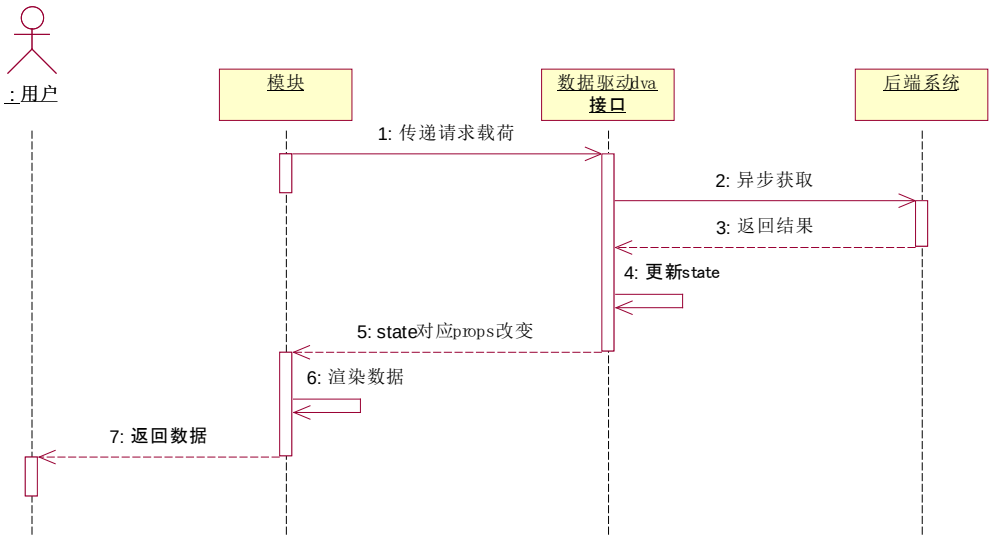

图 3-1-1-5-1 期刊介绍和投稿须知时序图

3.1.1.6 资料中心的方案设计

资料中心用于在首页模块显示单独页面，用以下载业务流程所需的资料。其设计可复用 3.1.1.3 节获取公告列表的设计，采用获取资料中心列表的后端接口，获取资料中心列表，并把列表项映射为下载接口，用户可点击列表项启动下载接口下载资料文件。

3.1.1.7 友情链接的方案设计

友情链接是显示在首页页面的一个分区，用于显示管理员设置的友情链接列表。该部分可以复用 3.1.1.5 节的设计，在首页页面加载时，从后端系统拉取友情链接名称和 URL，根据返回的 JSON 串对名称和 URL 进行渲染，生成友情链接在首页位置。

3.1.1.8 编辑部证书的方案设计

编辑部证书是在首页页面显示的一个走马灯展示区域，用于交替展示获得的证书图片。图片列表和名称由管理员在后台设置并上传，前端在打开首页时获取图片列表，以此设置走马灯组件的图片列表进行展示。

3.1.2 后台部分通用模块的方案设计

3.1.2.1 登录部分的方案设计

在登录部分中，本人主要负责两个部分——前端系统对登录数据的传输和后端验证码系统的设计。

对于前端系统的数据传输，主要是在登录环节，用户需填写自己的用户名、密码。因为一个账号有可能对应多个角色，所以还需要选择角色。以及用户填写的验证码，将密码在本地做 SHA-256 散列化后将所有数据提交到后端系统后进行验证。验证后，后端会返回关于用户的 ID、用户名、姓名和角色等账户信息，前端需要将账户信息更新到 dva.js 的 state 中，将账户信息和权限信息写入到 localStorage 便于后续功能对当前已登录身份的获取。

此外，前端需要具备记住密码功能。在用户勾选记住密码选项后，前端系统应把用户名、密码的 SHA-256 散列值和要登录的用户角色记录到 cookies 中。在打开登录页面时，前端系统应先检查 cookies 是否有对应表项，若有则应读取 cookies 的信息后则将信息填入表单框中，并设置好标记，使提交时不会再进行散列化。

在登出时，前端应清除所有 localStorage 的内容，清除 cookies 中对应键值对，退出登录。

由于 dva.js 本身也是一个 JavaScript 框架，所以 cookies 的处理、localStorage 的处理和页面跳转可以交给 dva.js 对应的登录命名空间进行控制。

在 cookies 不存在的情况下，登录阶段前端部分的时序图如图 3-1-2-1-1 所示。

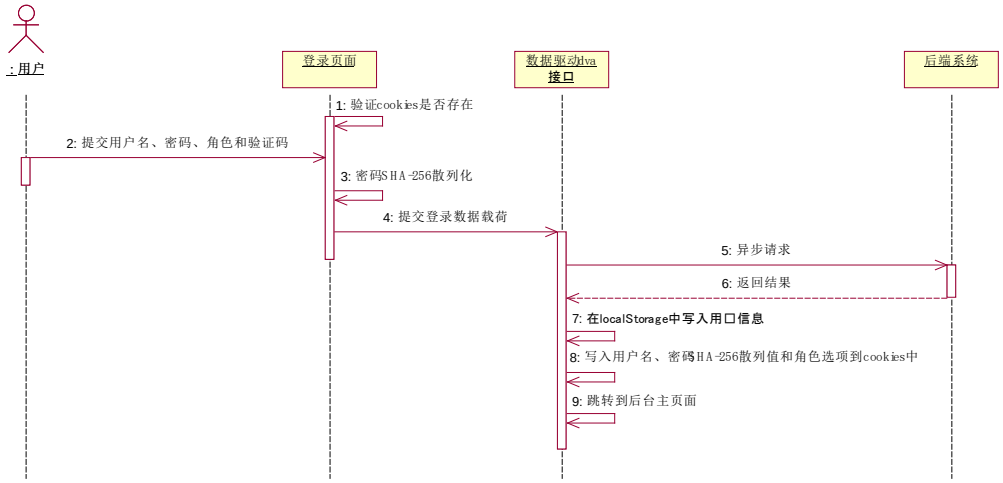

图 3-1-2-1-1 登录阶段前端部分时序图

登出阶段前端部分的时序图如图 3-1-2-1-2 所示。

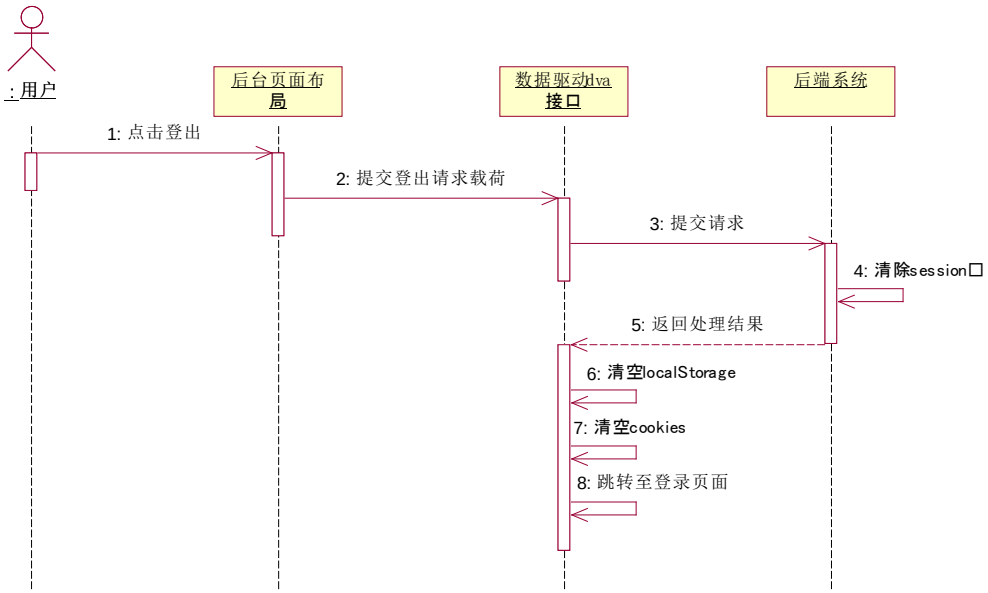

图 3-1-2-1-2 登出阶段前端部分时序图

在每次向后端提交报文时，若后端返回的报文提示后端 session 已过期，则前端需要提示登录状态过期，清空 localStorage 所存储的信息（包括权限信息）并跳转到登录页面。

对于验证码系统，由于攻击者可以通过直接向后端接口发送数据包的方式进行请求，前端设置验证码起不到验证的作用，所以验证码需要设置在后端。后端随机

生成一个 6 位的包含字母、数字的验证码序列，将验证码序列存储在 session 中，把验证码序列生成图片传递给前端进行渲染。后端也需要提供接口，使得用户可以刷新验证码。需要注意的是，验证码字符集中应避免出现“I、L、G”的大小写形式，因为可能会出现混淆。此外，验证码还需要有一定的反识别能力。

验证码部分前后端的时序图如图 3-1-2-1-3 所示。

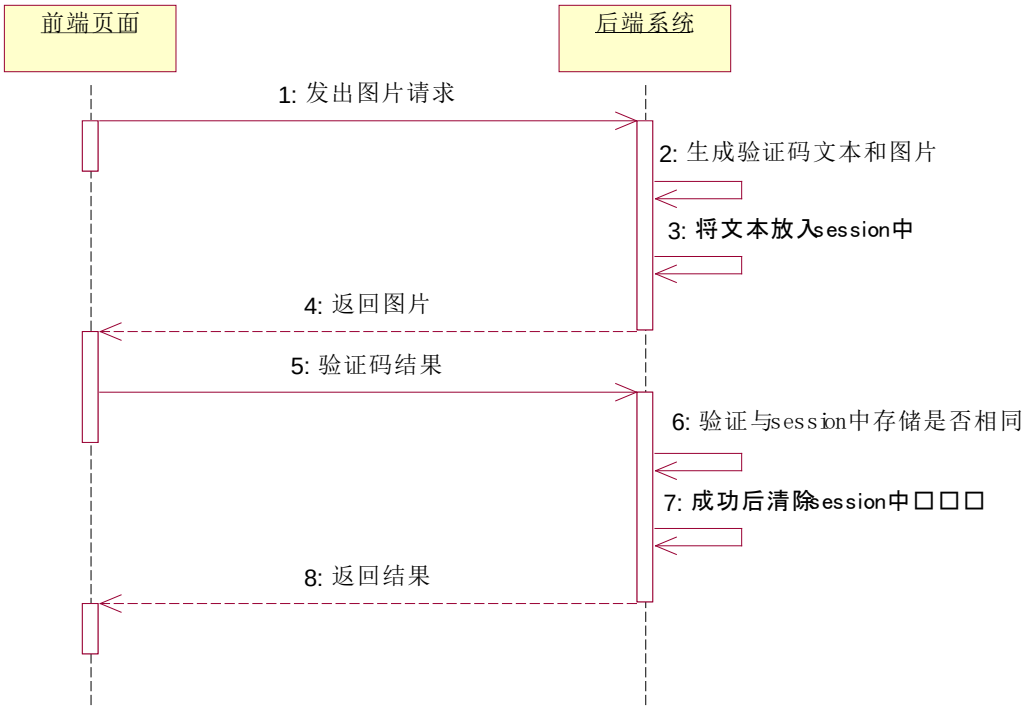

图 3-1-2-1-3 验证码部分前后端交互时序图

3.1.2.2 修改密码的方案设计

修改密码是登录之后的安全选项，用户可以验证安全问题后对用户密码进行修改。

在修改密码时，需要提交原密码、新密码和新密码的再次输入，若原密码不正确或新密码与再次输入不相同，则修改密码不予通过。其中，原密码不正确的判断由后端进行，新密码和新密码的再次输入不相同的判断由前端进行。

在输入新密码的过程中，前端需要根据密码强度要求，为用户实时渲染密码强度指示，对于密码强度为“弱”的不予提交。

在提交前，需要对旧密码和新密码均进行 SHA-256 散列化，散列化后再提交

给后端系统。时序图如图 3-1-2-2-1 所示。

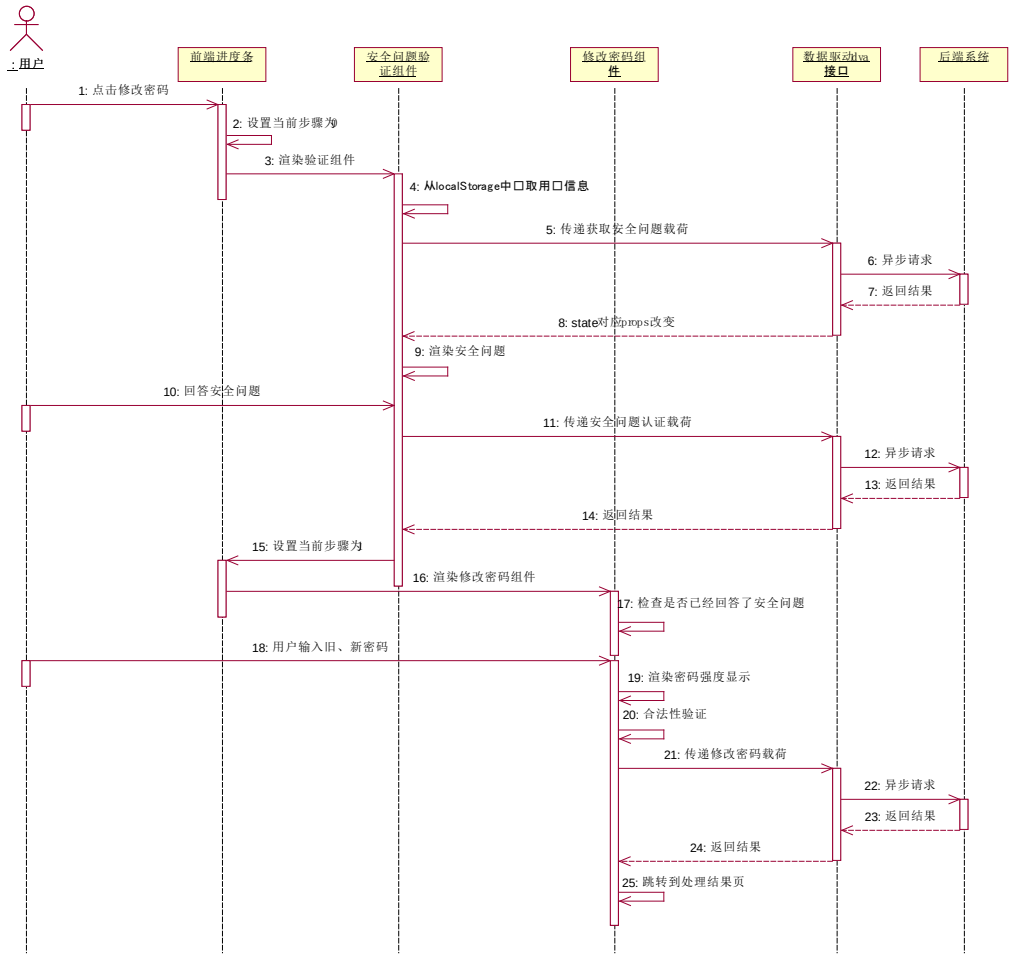

图 3-1-2-2-1 修改密码时序图

3.1.2.3 忘记密码的方案设计

忘记密码是在未登录前使用的功能，未登录的用户在忘记密码时，可以选择两种验证方式——通过安全问题修改密码和通过微信认证修改密码（已绑定微信的用户）。找回密码必须输入注册时提供的 token，不然只能联系管理员找回，token 的作用见 3.1.7.1 节。

对于通过安全问题修改密码的用户，前端系统将先渲染信息填写页面，填写需要修改的用户名和角色，确认该角色的用户真实存在后，渲染验证安全问题页面，正确回答安全问题后渲染密码重置页面，若新密码与再次输入不相同，则修改密码不予通过。新密码和新密码的再次输入不相同的判断由前端进行。

在输入新密码的过程中，前端需要根据密码强度要求，为用户实时渲染密码强度指示，对于密码强度为“弱”的不予提交。

对于此功能点，可以复用 3.1.2.2 节修改密码部分的安全问题验证和新密码输入部分的设计。

对于通过微信验证重置密码的用户，其在填写完用户名和角色信息，选择微信重置后，其可跳转到微信页面扫码验证，验证通过后，系统将检查数据库，确定重置操作是否合法，若合法则进入密码重置页面。时序图如图 3-1-2-3-1 所示。

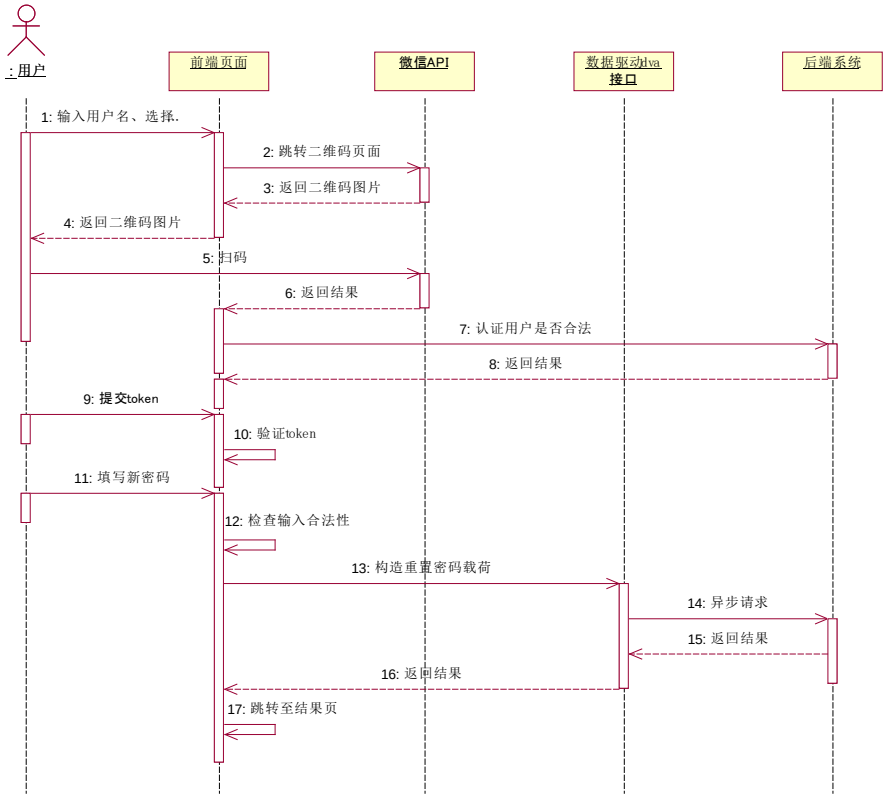

图 3-1-2-3-1 微信重置密码时序图

3.1.2.4 个人信息修改的方案设计

个人信息修改页面是登录后所显示的页面，根据权限的不同，显示的内容也不同。

对于投稿人，可以维护基础信息、地址信息、教育信息和学术信息。

对于编辑，可以维护基础信息。

对于审稿人，可以维护基础信息、地址信息、学术信息和银行卡信息。

在设计上，将每一个信息的修改单独形成一个页面，对于多角色共用的页面可根据权限对显示的表格内容进行调整。

个人信息修改页面作为主页面，根据权限渲染主页面的菜单，对菜单选项进行映射，映射到上述单独形成的页面上，上述形成的单独页面作为主页面的 children 进行渲染，路由使用 umi.js 嵌套路由进行控制。

时序图如图 3-1-2-4-1 所示。

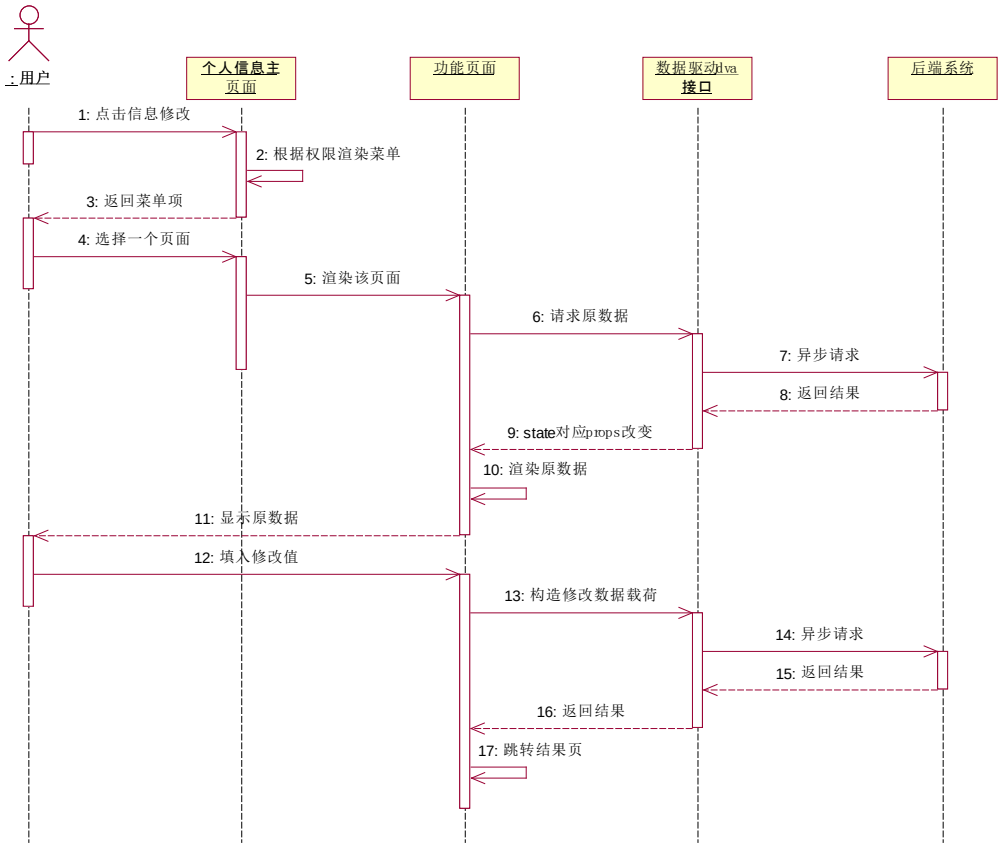

图 3-1-2-4-1 个人信息修改时序图

3.1.2.5 微信催促方案设计

微信催促功能是后台部分业务功能之一，主要供编辑和系统使用，用于在业务进度出现明显停滞时的提醒。仅限当前业务状态距离该状态结束时间 10 天内使用，可以通过点击提醒按钮的方式提醒当前业务状态处理人尽快处理。在距离结束时间 5 天时，系统会自动发送提醒消息，该消息会通过微信发送给待处理用户所绑定的微信号上。

该功能在设计上，分为上述两种发送方式，其发送消息的功能设计相同，均为利用微信平台 API 发送模板消息，发送消息内容和截止日期，以提醒尽快完成任务。前置步骤分为两类，一种是主动点击提醒，一种是系统自动提醒。系统自动提醒需要设置计划任务，在每天规定的时间检查任务列表，对即将超期的进行提醒。

主动点击的超期提醒时序图如图 3-1-2-5-1 所示。

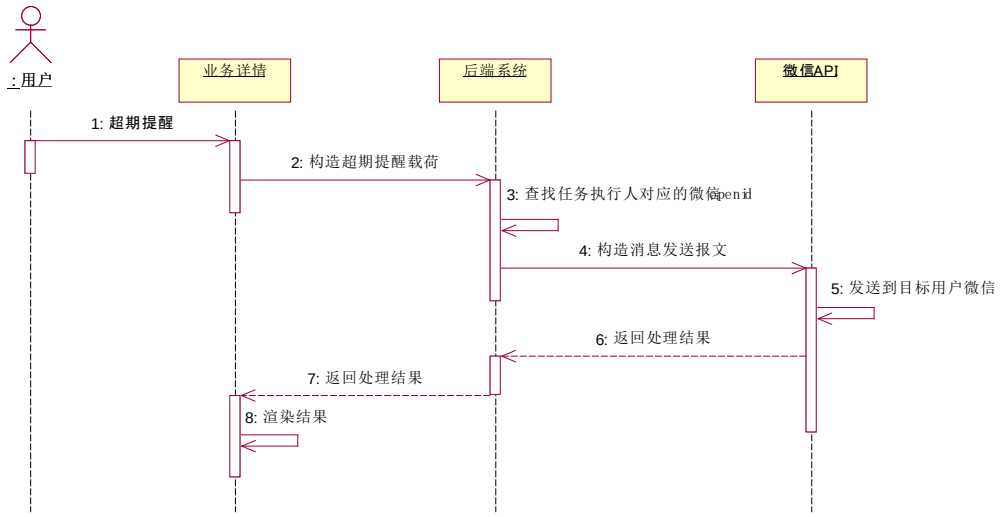

图 3-1-2-5-1 主动点击的超期提醒时序图

系统自动任务的超期提醒时序图如图 3-1-2-5-2 所示。

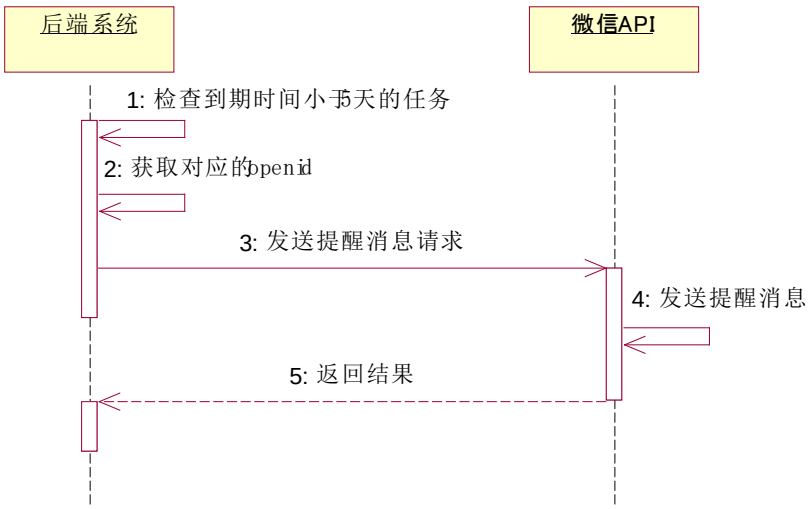

图 3-1-2-5-2 自动任务的超期提醒时序图

3.1.3 后台部分系统管理员的方案设计

3.1.3.1 公告管理的方案设计

公告管理分为公告的查找、添加、修改与删除。

公告由于用户需求需要能提供图片插入和格式的控制，所以公告正文的添加、修改与渲染需要使用富文本格式进行。由于后端需要通过每一篇文章在数据库中记录的图片名称来清除已经无用的图片，所以在富文本编辑器使用后端上传接口上传图片时，需要记录已经上传图片的文件名，记录在最后提交操作时上传的报文中。

公告还可以添加附件，由于附件的格式不能明确确定，所以需要通过表单格式向服务器提交公告的标题、富文本字符串和公告附件。

由于采用富文本格式，存在被 XSS 攻击的可能，所以在提交给服务器报文前，需要对富文本内容进行过滤。

管理员在维护公告时，需要为其渲染所有的公告项，并为每一个表项渲染修改入口，使得系统管理员可以通过点击公告名的方式进入修改页面。在进入修改页面时，系统需要自动填写原有的公告信息到组件中。修改公告时，需要提供删除附件或重新上传附件的选项。公告列表提供按标题搜索功能，可以输入关键词搜索公告并在列表中进行渲染。

渲染公告列表的时序图如图 3-1-3-1-1 所示。

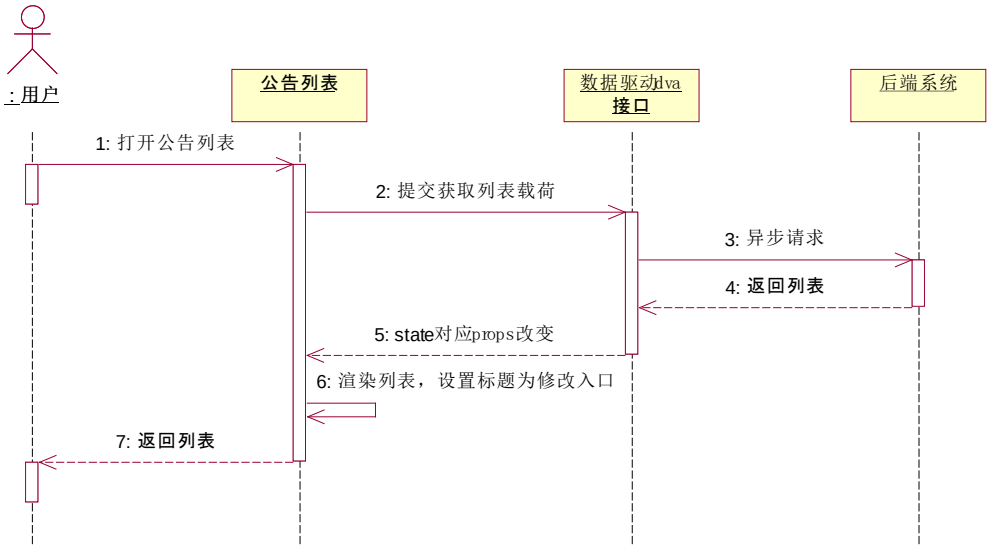

图 3-1-3-1-1 渲染公告列表时序图

新增公告的时序图如图 3-1-3-1-2 所示。

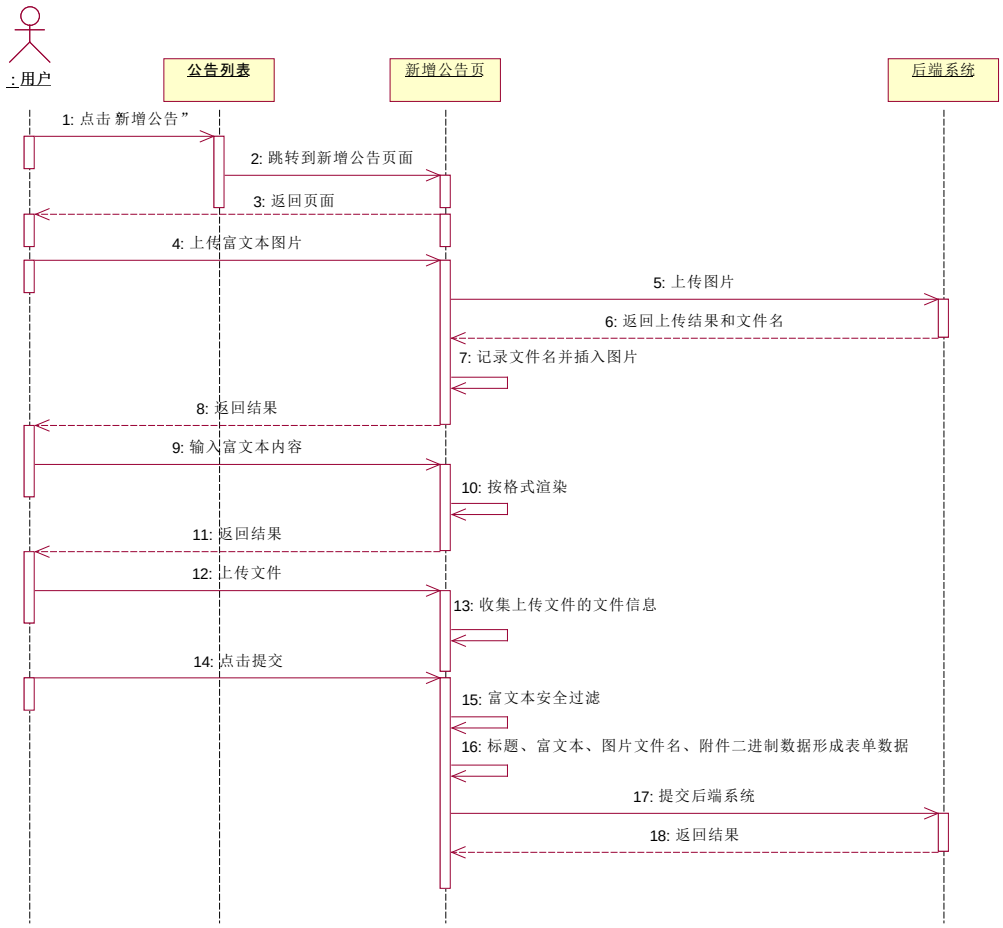

图 3-1-3-1-2 新增公告时序图

修改公告时序图如图 3-1-3-1-3 所示。

在删除公告中，公告列表渲染时会为对应表项渲染操作选项，当点击一条公告旁边的删除时，会激活隐藏的 Modal 进行确认，确认是否删除，若确认删除，将会向后端提交删除报文，并刷新公告列表。

在查找公告时，管理员可以在列表上方的表单中输入要查找的关键词，点击搜索后，前端将会根据关键词向后端请求列表数据并进行渲染，并把关键词存储在 state 中以进行后续的分页查找。表单还提供清空按钮，当管理员点击清空后，页面将清除原来的搜索关键词并重新做完整搜索。

删除公告时序图如图 3-1-3-1-4 所示。

搜索公告时序图如图 3-1-3-1-5 所示。

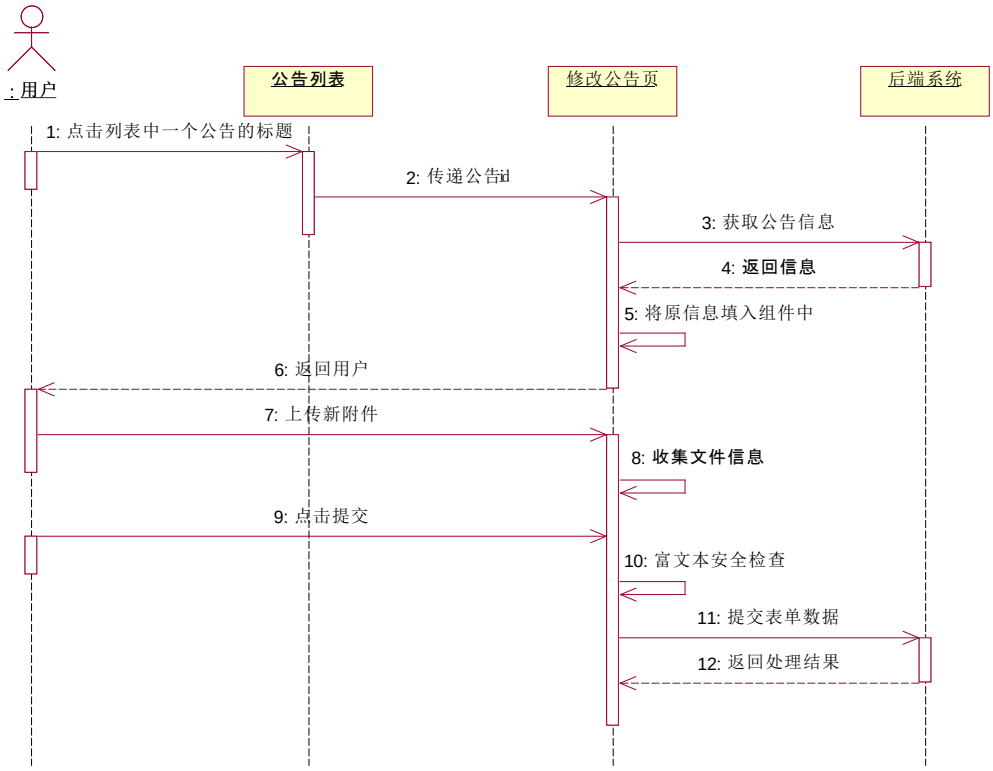

图 3-1-3-1-3 修改公告时序图

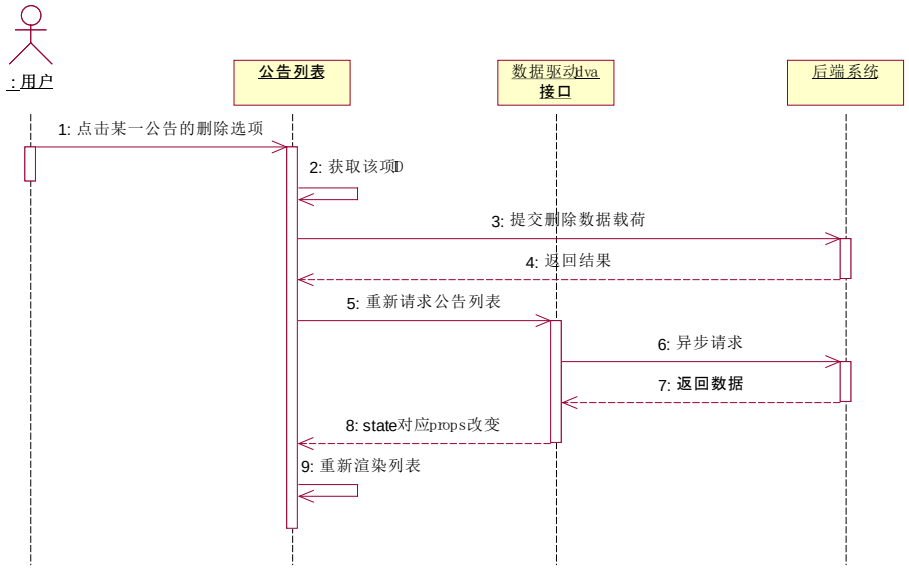

图 3-1-3-1-4 删除公告时序图

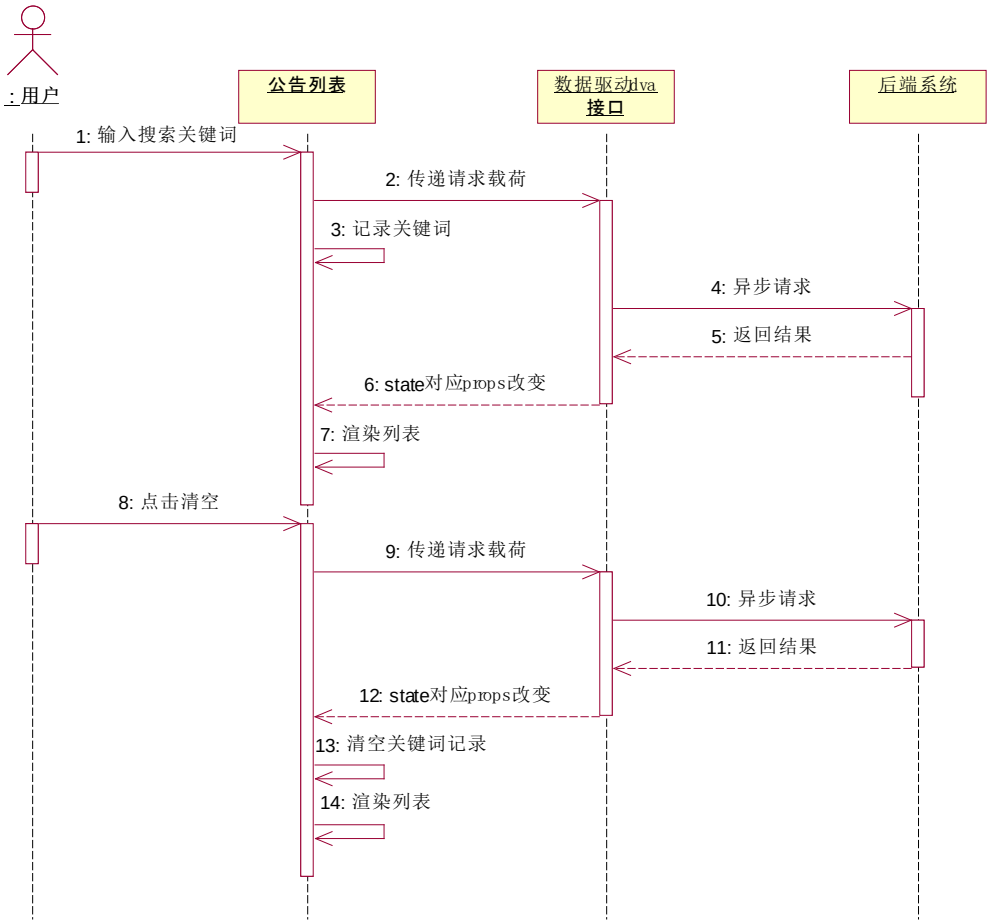

图 3-1-3-1-5 搜索公告时序图

3.1.3.2 用户管理的方案设计

管理员的用户管理共分为三类用户的管理——投稿人账号的管理、编辑账号的管理和审稿人账号的管理。

管理员拥有三种角色账号的修改、删除和重置密码权限，以及编辑和审稿人账号的添加权限。

对于三种角色的修改，管理员有权修改账号的除安全信息、银行卡信息（审稿人）之外的其他个人信息。在前端系统设计上，管理员在已经渲染的用户列表上，点击用户名，可以跳转到修改网页进行修改，其中，原用户的账号信息已经自动填入页面组件中。

对于三种角色的删除，后端系统应提供删除单个用户的接口，前端系统应同时

实现单个用户的删除和批量删除。

对于三种角色的重置密码，后端系统应提供重置单个用户的接口，前端系统应同时实现单个用户的密码重置和批量密码重置。

对于角色的添加，系统应支持逐个添加和批量添加两种方式。

逐个添加时，管理员可以点击列表的增加用户，激活 Modal，Modal 中使用分步表单展示，在未提交之前，Modal 支持在关闭后再打开仍可以保存临时数据。即在主页面的 state 中保存 Modal 的临时数据，Modal 设置在关闭时和分步表单点击下一步时调用主页面传递的函数将 Modal 的表单数据保存在主页面 state 中，在提交后清空主页面 state 中所存储的临时数据。

批量添加时，前端系统向后端系统传递包含表项的 csv 文件，后端系统解析 csv 文件后逐个调用数据接口添加用户。

添加角色时，后端系统会设置特殊标记。前端系统在获得数据后，使其在第一次登录时，必须设置安全问题才可继续业务活动。

为方便用户，按用户名或姓名的查询功能，查询设计可复用 3.1.3.1 节的搜索功能设计。

以审稿人为例。

管理员点击“审稿人管理”页面后，系统将自动拉取未按各种关键词过滤的第一页用户信息，并设置好分页器。

管理员点击页面上的“添加审稿人”，系统设置 Modal 的可见性为 true，并用自身 state 中所存储的数据初始化新增审稿人的表单。步骤共分为四步——基本信息、密码、学术领域和银行信息，管理员可以逐步添加，Modal 的页脚也根据步骤数值进行渲染，在第零步时只渲染“下一步”和“取消”，中间步骤渲染“取消”、“上一步”和“下一步”，最后步骤渲染“取消”和“完成”。当管理员点击“下一步”时，Modal 会调用主页面通过 props 传递的函数，用自身表单数据更新主页面 state，从而使得 Modal 在退出后重新进入时，仍有之前尚未完成的表单数据和进度。在提交后，Modal 会调用 props 传递进来的函数，利用主页面的接口向后端提交，并把自身 state 和主页面的 state 中存放的表单数据清空，以保证下次新增用户时，管理员可以从头开始输入用户数据。在通讯成功后，主页面将重新通过 dva.js 数据接口拉取最新的用户名单。

对于批量导入功能，系统管理员可以上传一个包含有用户信息的 Excel 文件，后端系统经过读取后，可以将文件中的用户信息导入到数据库中。

对于单独添加用户，时序图如图 3-1-3-2-1 所示。

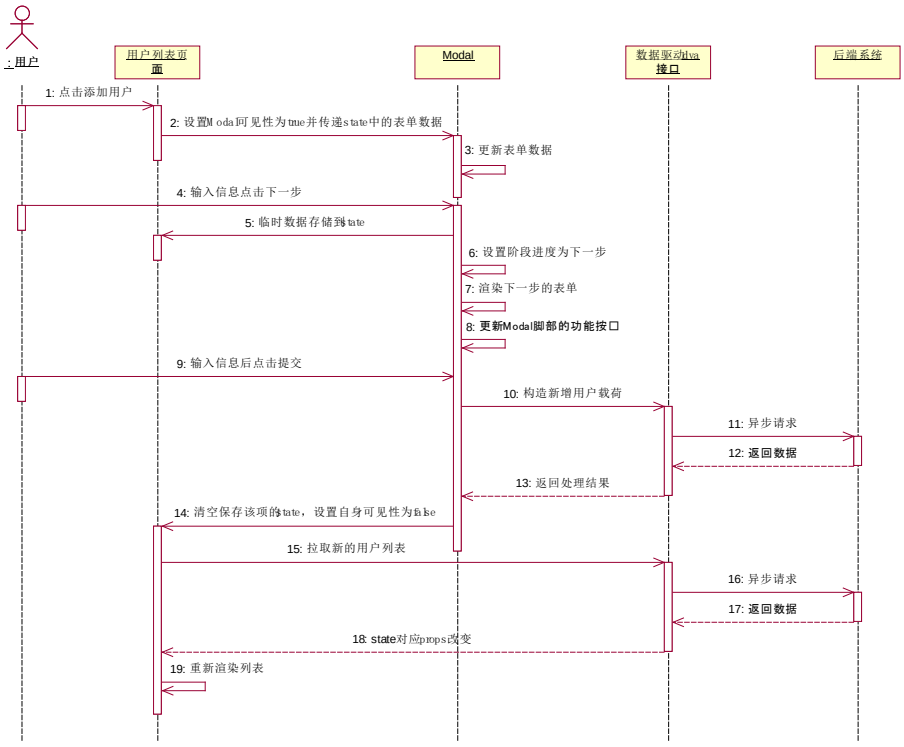

图 3-1-3-2-1 单独添加用户的时序图

对于批量导入用户，时序图如图 3-1-3-2-2 所示。

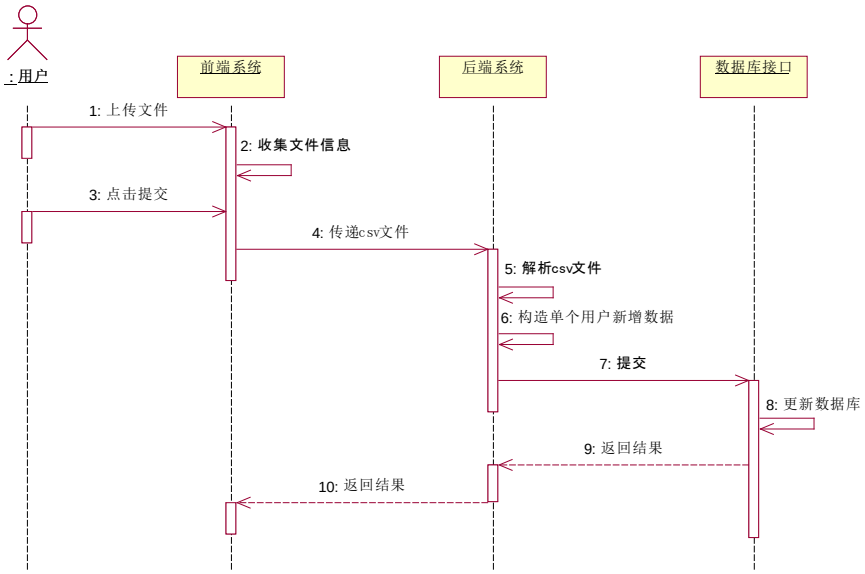

图 3-1-3-2-2 批量添加用户的时序图

当点击用户名时，将会跳转到修改用户信息页面，页面将根据用户 id 和角色从后端获取用户信息，填充到表单组件中，管理员可在修改后提交并更新用户信息。

在添加审稿人时，需要选择学术分区和其他基础信息，基础信息是由管理员设置，并存储在数据库中。在渲染页面时，需要先调用接口从后端获取到列表并存储在 dva.js 命名空间中供系统各组件共享，同样使用时间戳验证是否被修改过，若未修改则证明目前 dva.js 中的信息仍为最新，不需要传送完整的基础信息表格。

3.1.3.3 基础信息维护的方案设计

管理员在后台可以维护学术领域、证书列表、编委会名单、分区、文件中心和期刊信息等六项基础信息。

对于学术领域、证书列表和编委会名单，其前端系统运行模式均为：用户点击相应的菜单项，渲染列表页面，提供添加、更新、多选、删除、批量删除选项。在渲染列表页面时，前端系统从后端拉取数据列表，设置分页器，渲染表格列为 ID 与名称，把 ID 设置成修改接口，点击 ID 时会触发修改逻辑。

添加新基础信息可复用 3.1.3.2 节添加时的设计，主页面存储添加 Modal 的临时信息，点击添加时渲染 Modal 执行添加操作。

修改信息时，不跳转到新页面进行修改，在主页面内弹出修改 Modal 修改。在点击每一项的修改选项时，主页面将把该项的 record 信息和向后端通讯的函数接口作为 props 参数传入 Modal 中，并设置 Modal 的可见性为 true。Modal 在渲染时根据传入的 props 设置内部表单初值。管理员点击提交后，将调用传入的数据接口对后端进行通讯，更新完成后 dva.js 将更新自身，dva.js 更新后会触发引用其的列表更新。

修改信息的时序图如图 3-1-3-3-1 所示。

对于期刊信息修改来说，根据业务逻辑，期刊将按年分为若干卷，每年为一卷，每卷中要设置期数、显示和修改每一期最大文章数量、设置默认审稿费、版面费等。系统管理员需要可以添加新卷、对已存在的卷数的上述信息进行修改。

添加部分与 3.1.3.2 节相似，在修改方面有所不同。修改上，在点击列表内每一卷的年份后，前端系统需要获取该年份的详细信息并跳转到详情页面，将获取的数据载入到页面中。

详情页面上方需要按刊数逐期显示每一期的最大文章数量，并在点击文章数量时，弹出修改 Modal，将最大文章数量作为参数传入 Modal，使 Modal 在渲染时可以把数据自动填入表单供修改。

详情界面下方需要提供表单来修改卷基础信息，并且在跳转时应该自动填充原数据在表单中。

修改该期基础信息时序图如图 3-1-3-3-2 所示。

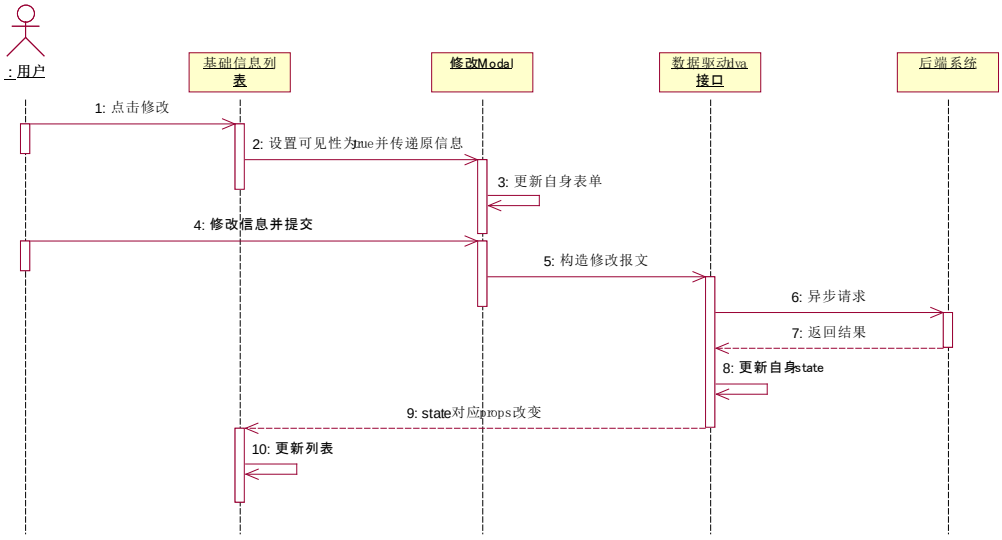

图 3-1-3-3-1 修改三类基础信息时序图

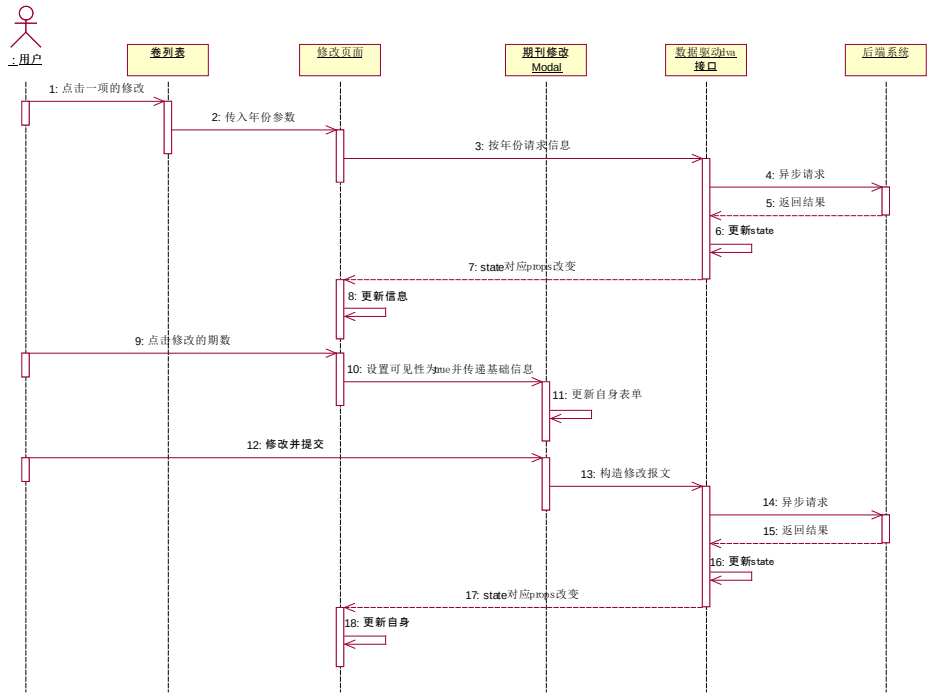

图 3-1-3-3-2 修改期刊基础信息时序图

对于文件中心，其列表与本节之前的列表渲染类似，在新增、删除和修改上，由于文件的特殊性，将使用表单进行前后端通讯。而后端系统需要的文件元数据，将由前端系统组织成一个 JSON 字符串，放在表单的一项中传输。

3.1.3.4 数据统计的方案设计

在数据统计中，页面载入时前端系统将向后端系统请求当前时间到之前六个月的逐月数据，整理为可视化插件所需的坐标格式，对于变化图，包含 x 和 y 的 map，x 和 y 分别是一个数组，两者按顺序组成 x-y 坐标。对于其他图，按照数据格式要求进行组织。将此数据组交由可视化插件进行可视化处理。

部分数据统计（如工作量）支持用户自选时间区间进行统计并可视化处理。

数据统计的时序图如图 3-1-3-4-1 所示。

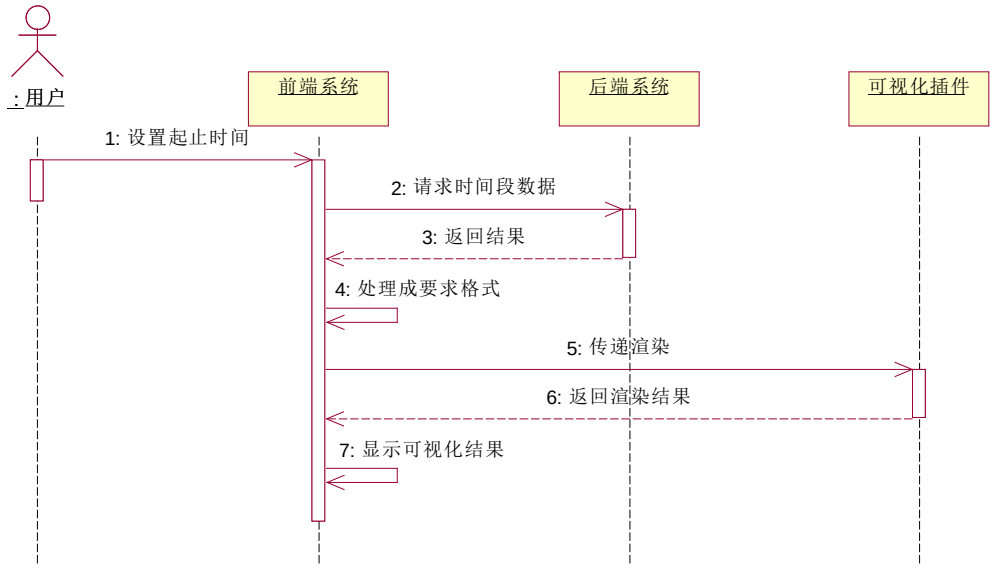

图 3-1-3-4-1 数据可视化时序图

3.1.4 后台部分编辑的方案设计

3.1.4.1 业务列表和业务操作表单的方案设计

编辑是整个业务流程的主体，编辑承担了整个业务流程的大多数工作并起到了业务的枢纽作用。

对于一篇文章，在系统中存储为一个业务，而任务负责在业务的一个阶段中标

记工作状态和内容。一个业务在一个阶段对应一个或多个任务，因为要记录整个业务状态历史，所以其在整个业务流程中会与一条任务链绑定。在一个阶段中，业务的处理主要是任务的处理，即任务是主体。

但对于用户来说，其进行的业务处理是对文章的处理，而后端处理针对的是业务所对应的当前任务，在设计上需要进行协调，把实际上的任务处理转化为文章处理展示给用户。

以任务显示中的审稿部分（审稿人数为 2 人）为例。

在此阶段，由于一篇文章对应两个审稿人，其当前任务链分叉为两条，分别对应两个审稿人的审稿任务。但是编辑在查看其所有负责的业务列表时，需要以业务为单位进行显示。若以任务为单位进行显示，该文章会在列表中显示为两条，且由于进行后端分页，有可能出现该文章的第一个任务在一页，第二个任务在下一页的情况，十分不便于使用。所以，在前端为用户的渲染上，需要以文章为标准进行设计。

因此，在设计上，业务将不止存储文章信息，也需要有一个字段用以存储当前状态码。后端系统在收到前端传递的状态码过滤条件后，从文章库进行搜索和分页，并根据文章 id 在任务库中查找每篇文章当前对应的任务并一并返回给前端进行渲染。

前端系统在收到后端系统发送的数据后，按照文章数组渲染表单，列将包括文章名和当前状态，指针可以通过悬停在状态上获取当前状态的详细信息，该详细信息前端将从后端传递来的当前对应的任务数组中获得。

任务列表提供按状态过滤功能，包括待完成任务、初审中、确定审稿费中、待缴审稿费、待确认审稿费中、待分配审稿人中、审稿中、确定审稿意见中、复核中、修改中、再审确认中、审稿中、确定审稿意见中、格式修改中、待格式确认、确定版面费中、待缴版面费、待确认版面费、被拒绝等状态，其中，待完成任务将筛选出所有需要当前角色现在处理的文章。

所有在列表中的文章标题在渲染时均会设置跳转接口，跳转到文章详情页面。对于待完成任务的文章，还会渲染供编辑处理该文章当前状态的业务表单。

在文章详情页面中，上部显示文章的元数据，中部根据业务进度渲染进度条，下部显示业务表单。

由于在获取列表时，后端已经传递了包括任务信息在内的充足数据，所以在新页面再次请求会重复请求。所以，使用 dva.js 的命名空间进行信息交换。在触发列表的修改接口后，接口会把当前行对应的文章信息（article）、任务信息（task）和

缴费信息（invoice，如有）提交到 dva 命名空间中存储，然后跳转到详情页面，详情页面根据 dva.js 中的数据进行渲染。以待完成任务为例，时序图如图 3-1-4-1-1 所示。

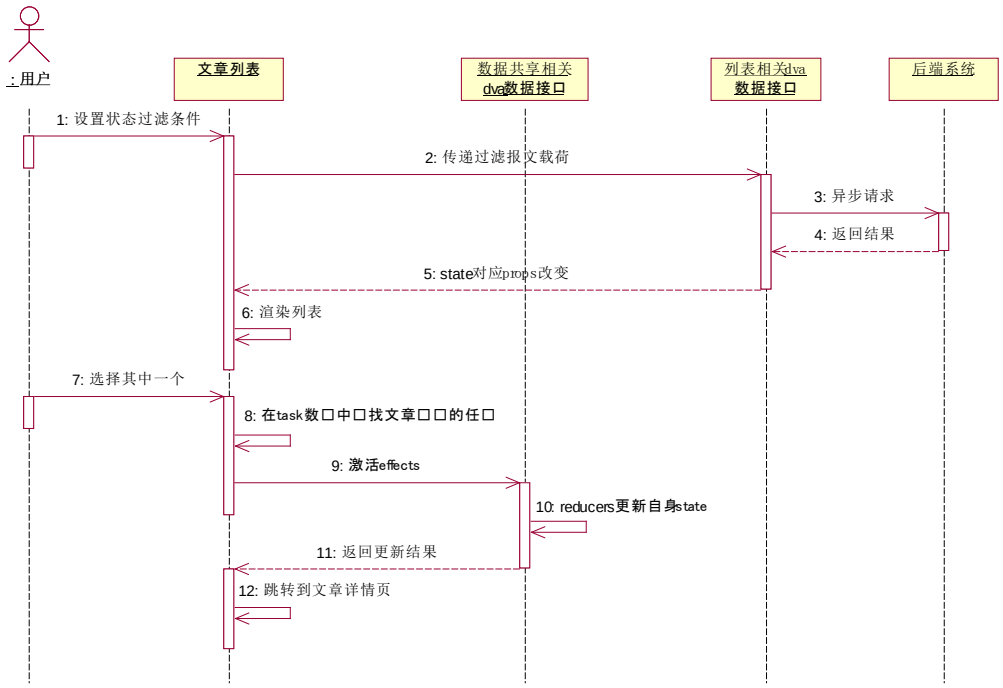

图 3-1-4-1-1 显示任务列表时序图

在跳转到详情页面后，页面会先从 dva.js 的 state 所映射的 props 中拉取列表所存储的数据。拉取到 article、task 和 invoice 信息后，根据信息渲染文章元数据和倒计时组件，根据状态代码渲染进度条。并根据不同的进度渲染不同的表单。

初审阶段渲染两个按钮：同意和拒绝。

确定审稿费渲染表单内容为：审稿费金额、审稿人数量。

确认审稿费表单内容为：投稿人所提交的审稿费汇款信息（如有）和发票信息的展示。

分配审稿人表单内容为：根据确定审稿费阶段所设置的审稿人数量，显示若干下拉菜单，供编辑选择学术领域合适的审稿人。

汇总审稿意见的表单内容为：审稿人的审稿意见显示、编辑的总审稿决定选择下拉菜单、编辑的总审稿意见。

复议表单内容为：投稿人的复议请求内容展示、编辑的总审稿意见。

再审确认表单内容为：同意和拒绝按钮。

再审意见汇总的表单内容为：审稿人的审稿意见显示、编辑的总审稿决定选择下拉菜单、编辑的总审稿意见。

格式修改的表单内容为：经过格式修改后的文件、提交按钮。

确定版面费渲染表单内容为：版面费金额。

确认版面费表单内容为：投稿人所提交的版面费汇款信息（如有）和发票信息的展示。

业务逻辑处理的时序图如图 3-1-4-1-2 所示。

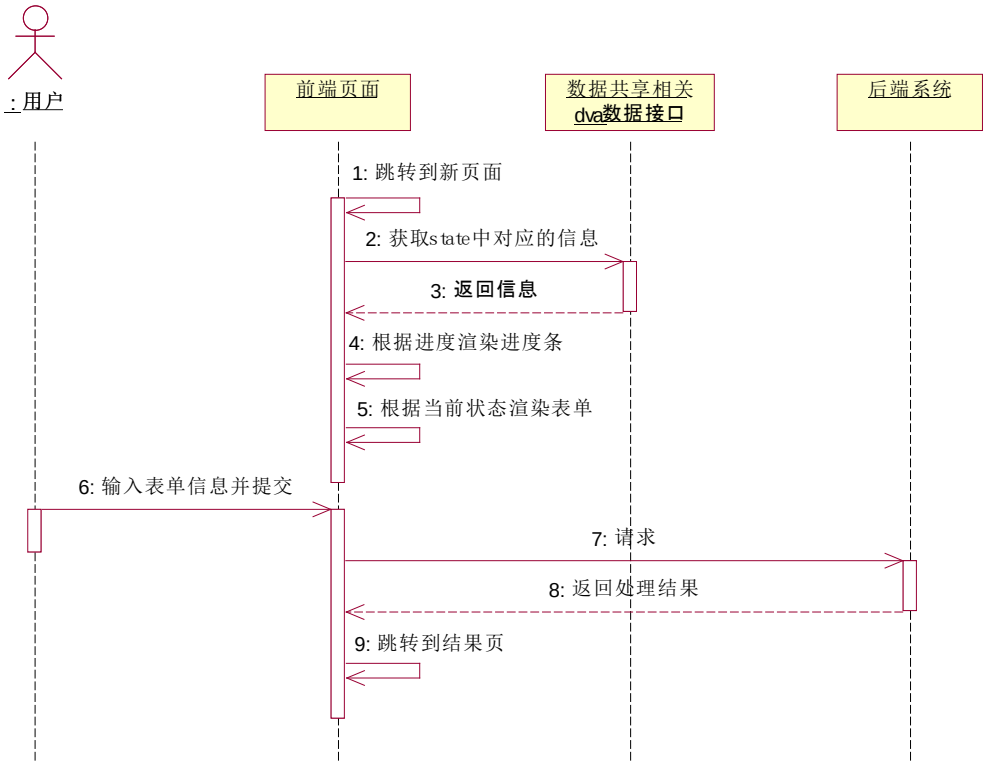

图 3-1-4-1-2 业务逻辑处理时序图

以分配审稿人为例，介绍详细表单的处理流程。

在分配审稿人前，后端系统提供了接口获取所有审稿人的姓名和学术领域信息，在表单渲染时，前端系统需要从后端系统获取所有审稿人的信息，并根据文章的学术领域对审稿人菜单进行筛选，筛选后再进行显示供编辑选择。对于被拒绝的任务，也会返回拒绝的审稿人的 ID，从而在下拉菜单中过滤。分配审稿人阶段的时序图如图 3-1-4-1-3 所示。

在非编辑操作阶段，虽然编辑不能提交表单操作，但是可以显示当前正在参与

流程的用户，并可以对在提醒时间内的用户进行提醒。

在审稿中时，编辑可以看到审稿人当前的审稿状态，包括审稿中、已审稿等状态。

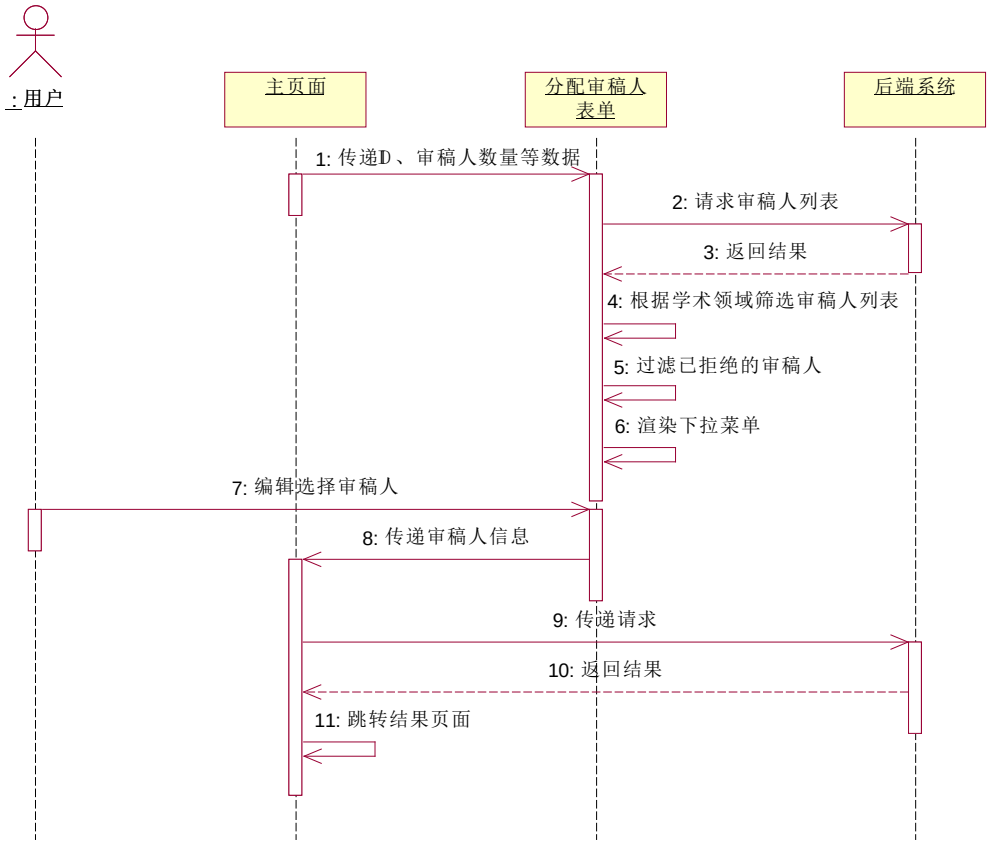

图 3-1-4-1-3 分配审稿人时序图

3.1.4.2 任务量统计的方案设计

由于在向后端请求待完成任务时，是一次性传递所有待完成任务和文章信息，所以在请求时可以通过使用 article 数组长度来获取当前未完成的任务量，显示在用户登录后的控制台首页上。

剩余的工作量统计是对所有与该用户相关的任务进行统计，由于数据量比较大，所以交由后端进行。

工作量统计的形式是显示在控制台首页的可视化图表，计算以任务（task）的完成量为标准，渲染近一年的工作量变化情况。由于后端接口是提供起止时间计算工作量的接口，需要由前端读取当前时间，获取一年前的时间，逐月提供起止时间

获取工作量，形成数据表格并渲染。

此外，还有使用饼状图渲染当前编辑在近一年内所处理的文章所属各学术领域的比例，查看论文通过情况等。设计可复用 3.1.3.4 节的系统设计。

3.1.5 后台部分审稿人的方案设计

3.1.5.1 业务列表与业务操作表单的方案设计

审稿人所参与的业务流程只有审稿环节，所以需要设计的内容相对其他角色较少。

审稿人列表和操作表单设计与 3.1.4.1 节的设计相似，由于使用双盲审核，审稿人只能在系统中获得待处理的文章列表，且在文章详情中不会显示作者名。

审稿人点击审稿任务后，前端系统会从后端拉取待处理的 article 和 task 数组，并按照 article 进行渲染，审稿人点击文章名即可进入详情界面。

详情界面会渲染除作者信息之外的其他元数据，但不提供下载。只有审稿人接受此审稿任务后，才会允许进行下载。

审稿人在接受审稿任务后，表单将会允许审稿人选择审稿结论和填写审稿意见。

审稿人在整个审稿流程中将不会得知文章作者的信息，在审稿结束后记录将无法查询，但可以通过统计信息入口查询工作量情况。

审稿人审稿过程的时序图如图 3-1-5-1-1 所示。

在审稿流程中，审稿人可以直接拒绝稿件的审阅任务。拒绝后，该稿件在系统中的状态将会回滚到待分配审稿人状态，若有多个审稿人，编辑可以在系统中看到该稿件处于待分配审稿人状态，但其中有的审稿人处于待接受或已接受状态，有审稿人处于拒绝状态。只要审稿人中有一人拒绝且编辑还未重新分配审稿人，该篇文章的状态就会是待分配审稿人状态。

3.1.5.2 工作量统计的方案设计

虽然审稿人在审稿后无法获得再从列表中查到此任务，但该任务仍在数据库中被记录。通过工作量查询接口，审稿人可以查到其在一定时间段内的审稿量情况、拒审比例情况以及审稿稿件所属领域情况。

工作量统计的方案设计可以复用 3.1.3.4 节的一部分系统设计，用以对查询到的数据进行可视化处理。

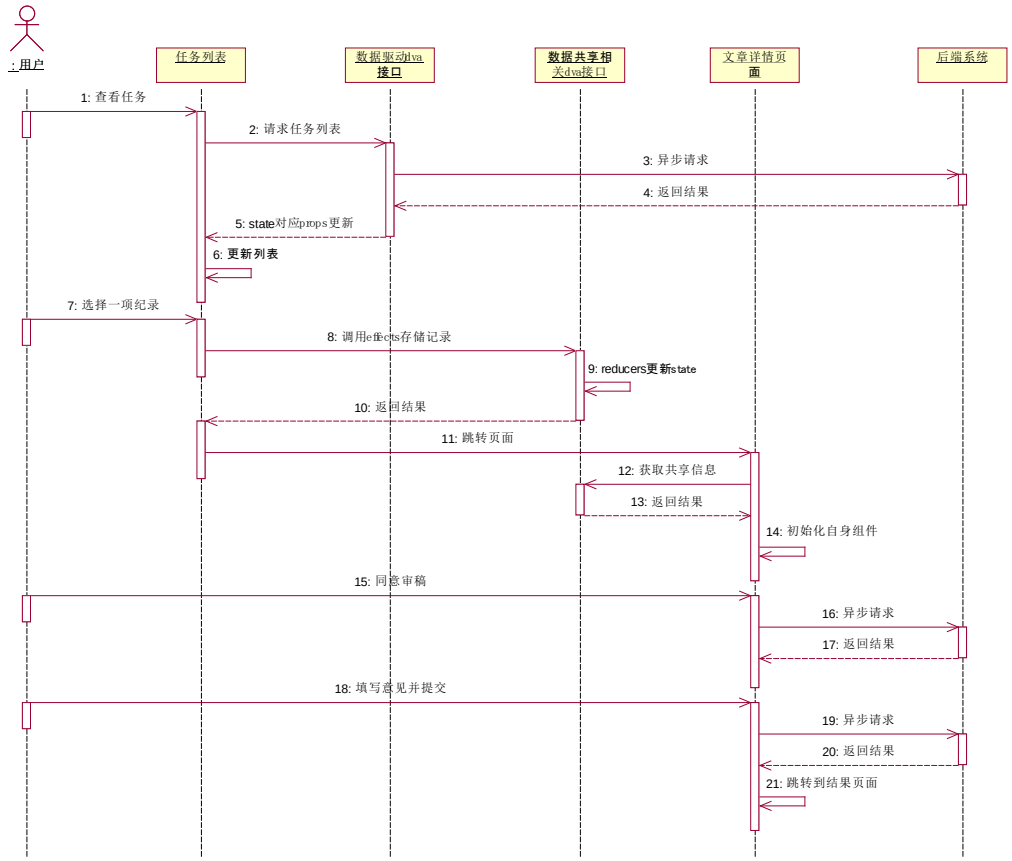

图 3-1-5-1-1 审稿流程前端时序图

3.1.6 后台部分投稿人的方案设计

3.1.6.1 注册的方案设计

在拥有账号的四类角色中，只有投稿人可以进行公开注册。公开注册入口在登录界面处。希望拥有账号的投稿人可以在注册页面注册自己的投稿人账号。

账号注册界面使用分布表单的方式进行，分为主页面、基础信息页面、地址信息、学术信息、安全问题和结果页。分页面通过作为主页面的 children 进行渲染。分页面设置好路由表后，主页面的分布进度条可通过 URL 来判断当前进度。

其中各分表单涉及到的基础信息可通过 3.1.1.2 节中获取学术领域的方式获得，并渲染到下拉菜单中。当用户选定一项后，在于后端进行交互时需要翻译回 ID 进行传递。

用户注册时，将每一个分步表单的临时信息更新到 dva.js 的命名空间中，在最后提交时，页面需要检查 dva.js 中的完整性，若完整则进行提交。

填写邮箱时，需要先检查邮箱是否存在，若不存在则进行邮箱验证，后端将会发送包含验证码的邮件，由用户输入验证码进行验证。

注册部分的时序图如图 3-1-6-1-1 所示。

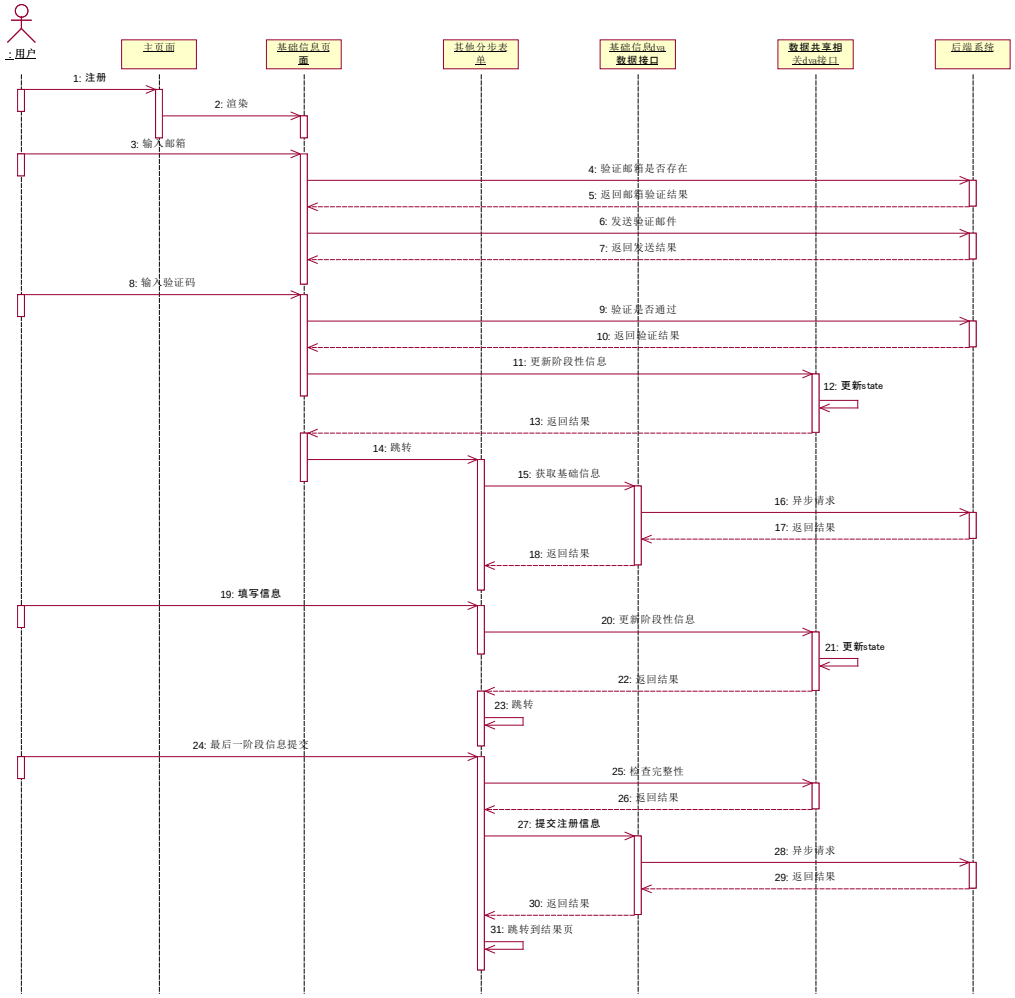

图 3-1-6-1-1 用户注册时序图

3.1.6.2 添加新稿件的方案设计

投稿人在注册账号后，可以向系统提交新稿件的请求。

对于前端系统，添加新稿件时，投稿页面应分为如下几部分——稿件基础信息（标题、学术分区、基金信息等）、稿件摘要部分（包含中文和英文摘要）、稿件关

键字部分（包含中文和英文关键字）、作者信息部分（该部分应可以提供作者数量的动态添加、删除）和文件的上传部分。

下面介绍读者信息部分的动态添加删除过程。

由于一篇文章的作者可能不只是上传稿件的用户，且可能有数量不可预测的作者，所以需要能够动态添加、删除作者。

添加作者需要按第一作者、第二作者、通讯作者等顺序依次添加作者，由于需要动态变化，所以把该部分表单写成单独的一个组件。组件中 state 开放一个区域存储数据。所有的数据操作围绕 state 进行，数据中每一个数据项还应有一个唯一的 key 供检索。

在表单中，需要注意，若当前可编辑标记不为 true，则只显示文本；若为 true，则显示表单文本框，并渲染保存按钮。

作者动态添加删除的时序图如图 3-1-6-2-1 所示。

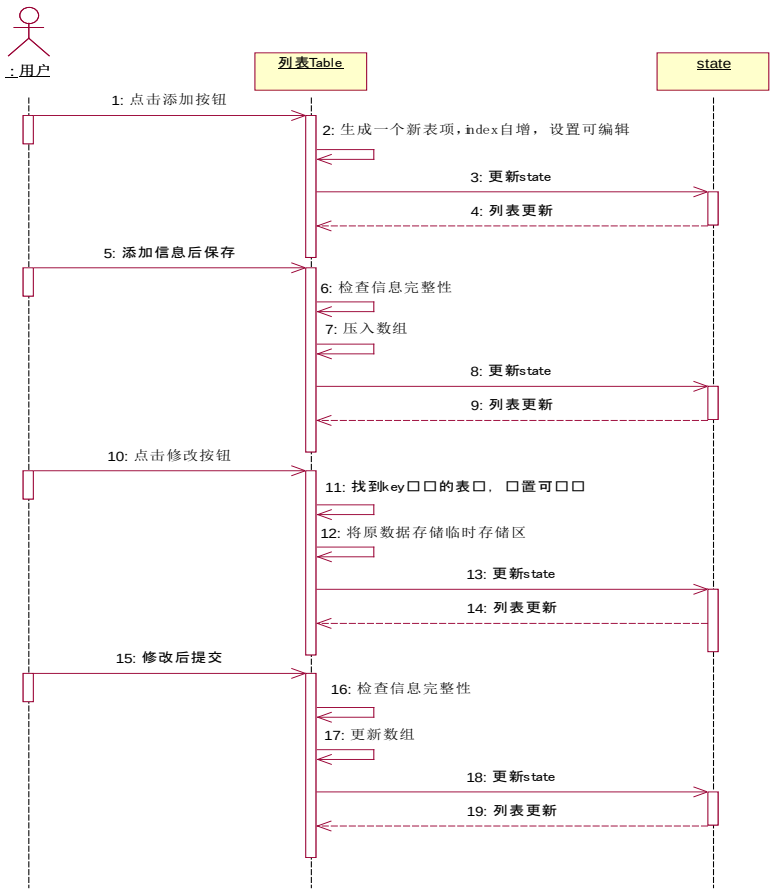

图 3-1-6-2-1 动态更新作者信息

上传文件后，前端系统将把文章元数据整合为一个 JSON 字符串，与上传的两个文件共同作为 FormData 表单项提交给后端处理。

3.1.6.3 业务列表与业务操作表单的方案设计

对于投稿人来说，其主要参与以下业务流程，其表单项大致如下。

审稿费缴费：均需填写是否需要发票和上传汇款单图片（以汇款方式缴费），若需要发票，则需填写发票抬头、纳税人识别号、发票收件人地址和发票邮寄地址。对于通过在线支付方式进行的，则先进行在线支付，确认支付完成后，再输入发票信息。

审稿复议：填写审稿问题答复。

修改提交：修改后的文件。

格式修改确认：确认按钮。

版面费缴费：同“审稿费缴费”。

其表单提交环节设计可复用 3.1.4.1 节编辑业务操作表单和列表的设计。对于投稿人，其可以查询所有已提交文章当前的业务状态（在审稿中时不允许知道审稿人信息），和目前需要处理的业务状态。

3.1.6.4 工作量统计的安全设计

对于投稿人的统计工作，主要是统计其投稿数量的变化折线图、通过与拒稿比例饼状图、待处理任务数量统计等，主要在后台系统控制台首页展示相关数据，用以进行可视化展示。

设计上可复用 3.1.3.4 节的统计设计。

3.1.7 系统安全的方案设计

本系统的安全策略基于以下假设进行设计。

1. 由于采用了 HTTPS 连接，其传输信道是安全的，不会被窃听的。
2. 用户不会主动泄露自己的加密素材和口令。
3. 管理员不会监守自盗。
4. 未在系统中注册的用户绝对不可信。
5. 注册用户存在主动攻击系统的可能。
6. 负责存储的云服务器是不可信的。

3.1.7.1 文件加密的方案设计

由于未出版的，在审阅流程中的文章需要保密，为了防止因服务器被攻击造成的数据泄露风险，需要对文章文件执行安全保护措施。

对于每个进入业务流程，暂未出版的文章，存储在服务器的文件将进行 AES 加密，其密钥随机生成。加密完成后，该密钥将分别由投稿人口令、编辑口令和管理员口令的 MD5 散列值扩展的密钥再次进行加密进行存储。

对于审稿人，每个审稿人还将会生成一个以口令扩展出的 RSA 公-私钥对，在分配审稿人后，编辑将会使用他的口令解密出文件密钥，并用该审稿人的公钥加密，在审稿人确定接受任务后，会使用其私钥解密文件的加密密钥，此后加密与其他角色相同。

在注册时会生成一个 token，转换为 base64 编码后由用户保存。token 借鉴了 Java Web Token 的设计思想，payload 部分存储了该用户的账号信息和所有的加密素材，并使用服务器公钥加密存储。在重置密码时必须提供 token，验证后根据 token 中的加密素材与新密码，对与该用户相关文件的加密进行更新。

3.1.7.2 银行卡号安全的方案设计

在业务逻辑中，审稿人提供了银行卡号用于支付审稿费。但系统有可能受到恶意攻击或篡改，导致后续转账工作可能转到了恶意账户上，或者服务器被拖库导致银行卡信息泄露，所以系统需要保证自己系统中记录的银行卡号是准确、未被篡改的，同时也要兼顾速度。

因此，在存储卡号时，对卡号加盐后使用 HMAC 进行散列化处理，存储到数据库中。然后把卡号使用服务器密钥进行 AES 加密，加密后存储在数据库中。

在每次需要显示卡号时，会对卡号解密后先验证 HMAC 散列值，只有散列值正确才会认为此卡号是正确卡号。

在每次修改时，审稿人需要验证安全问题才可以进行修改。在后端验证安全问题正确后，将返回前端一个 token，token 的 payload 部分包含用户名和 token 过期时间（过期时间设置在十分钟），signature 部分后端包含对 payload 部分的签名。修改卡号时前端必须随卡号修改报文上传该 token，后端验证无误后才可以进行修改。

3.1.7.3 服务器与前端交互安全的方案设计

在服务器与前端交互上，由于 HTTP 已经几近淘汰，所以系统将全部使用 HTTPS 连接，保护传输路径上的数据安全。客户已经准备向 CA 机构申请商业证书，所以可以使用此证书进行安全保护。

将数据合法性验证和权限判断逻辑交由前后端分别执行，起初是决定将合法性验证交由前端进行，但后来在模拟渗透时发现，即使通过 HTTPS 传输数据包，客户机依然可以通过 BurpSuite 等数据包嗅探软件通过 HTTPS 代理的方式获得 HTTPS 报文内容并进行修改后转发给服务器，这对于服务器的安全是十分危险的，所以为了保证用户可用性和服务器数据的安全性，决定将此判定交由前后端分别执行。同时在涉及用户登录、修改密码等操作中，对报文 JSON 串进行应用层加密，使用 RSA 公钥加密算法，在应用层加密用户报文，以防止数据包嗅探软件进行修改。

3.1.7.4 图片搜索的安全设计

由于医学论文涉及到的图片比较多，对于一张图片，编辑可能需要寻找它在哪篇文章里出现过。而且，由于图片复用情况时而出现，所以至少在编辑部自身的论文库中，客户希望系统能够检查是否有相似度比较高的图片。

所以在此系统的第一个版本中，设计了该模块，并在后续迭代中进行改进。

下面是第一个版本图片安全搜索的方案设计。

图片及其他文件的存储不同于负责处理业务逻辑的服务器，是存储在云服务器上。云服务器在运行时有可能被恶意人员攻击并获取数据，也有可能因为自身利益的考量或受外部压力的影响，云服务提供商可能会违背商业道德，试图获取用户的私密信息。所以，需要在上传服务器之前即做好加密，但是对于存储的云服务器来说，加密的数据如何实现客户要求的搜索功能非常重要。

在此模块的设计中，后端获取到文章文件后，先使用 Apache PDFBox 或 Apache POI 组件自动抓取文章中的图片。对于每个图片，提取其方向梯度直方图（HOG）矩阵，并处理为一个 1152 维的向量，作为该图片的特征向量。

在加密处理环节，需要的密钥组为两个 1160*1160 的可逆矩阵 M1，M2 的转置，以及一个 1160 维的 0，1 数组 s，以及下标 1154-1160 中随机生成的随机数数组 w。

在提交给搜索服务器之前，生成的 1152 维特征向量 p 或查询向量 q 需要扩展为 1160 维，来提高安全性。

扩展规则为：p 从 1153 维到 1160 维，如果 Si=1，则令 pi 为 wi；否则令其等于一个新随机数。向量 q 则与其相反，这样对于 p 和 q，其扩展维度是已知随机数

还是新随机数是完全相反的。

这样对于一个 p，完全可以确定对应的 q 在 S_i 为 0 的位置值一定是 w_i。可以在 p 的最后一个为 0 的位置设置（假设位置为 t）为从 1154 维开始到该维所有 S_i=0 的位置的值 p_i*w_i 的相反数除以 w_t，这样可以使得求点积后扩展维不会影响点积值。

对于 q，则对所有 S_i=1 的位置做类似的处理，这样可以使得所有扩展维在点积后为 0。

对于第 1153 维，p 设定为-0.5|p|；对于 q，1-1152 维全部乘以一个正整数 r，1153 维设定为 r。用于更改语义和提高安全性，但是其不会影响最终的运算结果。[1]

之后，需要对 p 和 q 进行分割处理。对于 p，若 S_i=1，则将 p_i 分割为 p_{ia} 和 p_{ib}，其中 $p_i=p_{ia}+p_{ib}$；若 S_i=0，则 $p_i=p_{ia}=p_{ib}$。对于 q，则相反处理。以二维向量为例，若 $p_a=(x_1, x_{2a})$，$p_b=(x_1, x_{2b})$，$q_a=(y_1, y_{2a})$，$q_b=(y_1, y_{2b})$。可以证明，$p \cdot q=p_a \cdot q_a+p_b \cdot q_b$。[1]

进行加密处理：$p'_{ia}=M_1^T p_{ia}$，$p'_{ib}=M_2^T p_{ib}$；$q'_{ia}=M_1^{-1} q_{ia}$，$q'_{ib}=M_2^{-1} q_{ib}$。

对于距离的比较，“p1 比 p2 查询距离向量 q 更近”可以等价于式（3-1-7-4-1）[1]。

$$(p'_{1a}-p'_{2a})*q'_a+(p'_{1b}-p'_{2b})*q'_b>0 \qquad \text{公式（3-1-7-4-1）}$$

利用此作为比较大小的条件后，配合排序即可找到相对距离比较小的几张图片，通过其对应的文章 ID 即可查到对应文章。

3.2 复杂工程问题的具体实现

3.2.1 后端验证码模块的具体实现

由于前端设置验证码对于恶意使用者的拦截强度不大，容易被通过直接发送报文的方式破解，所以使用后端设置验证码的方式进行验证。

Kaptcha 是 Google 开发的一种后端验证码生成插件，其支持自定义字符集和设置噪声（干扰），支持传递 jpg 字节流到前端，方便进行渲染。验证码的文字版本也可以记录在 session 中，方便进行验证。

在使用之前，需要根据文档编写生成的基础设置，包括字体颜色、图片宽度、

图片高度、字体大小、所用的字符集和验证码边框等，如代码 3-2-1-1 所示。

代码 3-2-1-1 设置 Kaptcha

```
@Component
public class KaptchaConfig {
    @Bean
    public DefaultKaptcha getDefaultKaptcha() {
        com.google.code.kaptcha.impl.DefaultKaptcha defaultKaptcha = new
com.google.code.kaptcha.impl.DefaultKaptcha();
        Properties properties = new Properties();
        properties.setProperty("kaptcha.textproducer.font.color",
"red");
        properties.setProperty("kaptcha.textproducer.font.size", "25");
        properties.setProperty("kaptcha.textproducer.char.length",
"6");
        properties.setProperty("kaptcha.image.width", "110");
        properties.setProperty("kaptcha.image.height", "40");
        Config config = new Config(properties);
        defaultKaptcha.setConfig(config);
        return defaultKaptcha;
    }
}
```

在设置好后，即可在后端开放验证码获取接口，将验证码文本放入 session 中，发送经过 base64 编码的图片文件，前端随用户名和密码一起提交。生成验证码的核心代码如代码 3-2-1-2 所示。

代码 3-2-1-2 生成验证码

```
String kaptchaText = kaptchaProducer.createText();
httpSession.setAttribute(“kaptchaText”, kaptchaText);
BufferedImage bi = kaptchaProducer.createImage(kaptchaText);
ByteArrayOutputStream        byteArrayOutputStream        =        new
ByteArrayOutputStream();
ImageIO.write(bi, "jpeg", byteArrayOutputStream);
String       base64Image       =       ""       +
Base64.encodeBase64String(byteArrayOutputStream.toByteArray());
```

后端还需设置好更新验证码接口，在用户点击时为其重新生成一个验证码，并将文本存入 session 中。

在登录时，登录接口可以从 session 中取出正确的验证码文本，与用户填写的

文本进行比对，若相同则代表验证码输入正确。

3.2.2 前端登录模块的具体实现

在前端登录中，可分为用户名/密码登录和微信登录两种，用户名/密码登录可以为所有用户完成登录过程，微信登录只为已经绑定过微信的用户进行登录过程。

对于用户名/密码登录，前端页面组件主要负责收集用户信息，由于 dva.js 本身也是 JS 组件，所以可以把登录的数据装载和跳转逻辑放在 dva.js 的中进行。

在 render 渲染部分，使用 Ant Design 的 Login 对应的 UserName 和 Password 作为输入，两者分别对应 Input 组件的明文和 Password 类型，还需要设置一个下拉菜单供用户选择要登录的角色。以 Password 部分为例，render 中的核心代码如代码 3-2-2-1 所示。在提交到后端之前，前端会调用 CtyptoJS 的插件计算密码的 SHA-256 散列值放入报文载荷中。

代码 3-2-2-1 渲染组件的核心代码

```
<Login.Password
    name="password"
    placeholder="请输入密码"
    rules={[
    {
        required: true,
        message: "密码不能为空",
    },]}
    onPressEnter={e => {
        e.preventDefault();
        this.loginForm.validateFields(this.handleSubmit);
    }}
/>
```

对于验证码，在获得 base64 编码后，可直接放入 img 标签的 src 属性中渲染。

在提交时，可调用 dva.js 提供的 dispatch 函数，触发 login 命名空间的业务逻辑，开始登录流程，如代码 3-2-2-2，其中，uploadJson 为发送给 dva.js 的报文载荷。

在 dva.js 中，reducers 负责更新 state，其返回值即为对 state 的更新，但其不支持异步交互。带有“副作用”的操作，如异步操作，将由 effects 中的方法执行，effects 方法在交互得到报文后，处理后交由 reducers 方法对 state 进行更新。

代码 3-2-2-2 提交给 dva.js

```
if(!err){
      this.props.dispatch({
        type: `login/login`,
        payload: uploadJson,
      });
}
```

登录的 effects 核心代码如代码 3-2-2-3 所示在收到报文后，dva.js 先将报文提交后端进行异步处理，若登录成功，则将用户登录信息记入 cookies 中，并把返回的登录信息记入 localStorage。

代码 3-2-2-3 dva.js 的 effects 处理核心代码

```
*login({ payload }, { call, put }) {
    const response = yield call(serviceUserLogin, payload);
    yield put({
      type: 'changeLoginStatus',
      payload: response,
    });
    // 登录成功
    if (response.result === 1) {
        if(payload.autoLogin){
        // 若自动登录，则记住 cookie
        cookie.save('username',response.username);
        cookie.save('password',response.password);
        cookie.save('role',response.role);
        }
    localStorage.setItem('username', response.username);
    localStorage.setItem('id',response.id);
    localStorage.setItem('name',response.name);
    localStorage.setItem('role',response.role);
    localStorage.setItem('keys',response.keys);
    reloadAuthorized();
    yield put(routerRedux.replace('/dashboard'));
    }
},
```

而 reducers 对应的代码如代码 3-2-2-4 所示，在设置好本命名空间 state 之后，根据角色 ID 翻译为可读的权限信息，并通过接口写入 localStorage，供路由管理插

件 umi.js 进行权限管理。

在 umi.js 中，路由表的部分内容如代码 3-2-2-5 所示，umi.js 根据写入 localStorage 的权限信息进行控制，页面的权限信息由 authority 段控制，对于权限不合法的用户，路由插件会自动跳转到 403 页面。

代码 3-2-2-4 dva.js 的 reducers 处理核心代码

```
changeLoginStatus(state, { payload }) {
    let currentAuthority="";
    if(payload.role == "0"){
      currentAuthority="admin";
    }
    else if(payload.role == "1"){
      currentAuthority="author";
    }
    else if(payload.role == "3"){
      currentAuthority="editor";
    }
    else if(payload.role == "4"){
      currentAuthority="referee";
    }
    else if(payload.role == -1){
      currentAuthority="";
    }
    setAuthority(currentAuthority);
    return {
      ...state,
      status: payload.result,
      name: payload.name,
    };
}
```

对于登出流程，在触发关于登出的 logout 之后，其会向后端发送登出报文，并在返回成功后调用 reducers 清除 state 和权限信息，并清空 localStorage 所存储的数据和 cookies（若有）。

代码 3-2-2-5 路由表的部分代码

```
path: '/',
    component: '../layouts/BasicLayout',
    Routes: ['src/pages/Authorized'],
    routes: [
      // dashboard
      { path: '/dashboard', redirect: '/dashboard/board', authority:
['admin', 'author', 'editor', 'referee'] },
```

对于已经绑定微信的用户，通过点击微信登录接口，前端将跳转到微信的地址上，在跳转时，在参数内设置回调地址。由于系统采用前后端分离的架构，所以跳转将由前端进行，回调地址也应是前端地址。

在调用回调地址后，前端通过对 URL 的处理，使用 split 函数以“?”为分隔符分割出参数，后以“&”分隔符分割出 code 参数和 STATE 参数，传送给后端接口。

后端编写了 httpGet 函数用以从微信接口获取 JSON 对象返回给 Controller 接口处理，如代码 3-2-2-6 所示。在从微信接口获得字节流后，将把其构造成 JSON 对象返回处理。

后端在获得参数后，执行以下步骤获得用户的 OpenID（OpenID 对于一个公众号唯一，用以标识用户）。

1. 获取 AccessToken 和 OpenID：在获得 code 后，后端需要从微信 API 获取 AccessToken 以获得用户数据。AccessToken 的有效期很短，填入公众号的 appid、密钥和所获得的 code 后，微信 API 将会返回 AccessToken 和 OpenID。根据文档，用户详细信息可使用 AccessToken 获得，但是在登录阶段只需要 OpenID 即可。

2. 获得 OpenID 之后，即可调用数据库接口在数据表中查找 OpenID 对应的用户，若没有找到，则返回不存在；若找到，则直接调用后续的登录流程接口，将用户数据写入 session 并返回给前端处理，核心代码如代码 3-2-2-6 所示。

代码 3-2-2-5 httpGet 请求核心代码

```
/**
     * 进行 HTTP 调用
     *
     * @param urlString
     * @return jsonObject
     * @throws Exception
*/
private JSONObject httpGet(String urlString) throws Exception {
    URL url = new URL(urlString);
    StringBuffer buffer = new StringBuffer();
    HttpsURLConnection                    connection                    =
(HttpsURLConnection)url.openConnection();
    InputStream inputStream = connection.getInputStream();
    InputStreamReader          inputStreamReader          =          new
InputStreamReader(inputStream, "UTF-8");
    BufferedReader            bufferedReader            =            new
BufferedReader(inputStreamReader);
    String str = null;
    while ((str = bufferedReader.readLine()) != null) {
        buffer.append(str);
    }
    bufferedReader.close();
    inputStreamReader.close();
    inputStream.close();
    JSONObject jsonObject = null;
    jsonObject = (JSONObject) JSONValue.parse(buffer.toString());
    return jsonObject;
}
```

代码 3-2-2-6 微信登录流程核心代码

```
//获取微信的 openId
urlUser = String.format(urlOpenID, code.toString());
jsonObject = httpGet(urlUser);
openId = jsonObject.getAsString("openid");
int index = userService.verifyWeixinUserPwd(openId);
…（执行登录流程）
```

3.2.3 前端投稿人注册模块的具体实现

投稿人注册模块采用分步表单方式，有密码强度检测功能，临时信息存储在之前在 dva.js 设置好的数据共享区域中，每段表单通过 effects 写入 dva.js 命名空间，然后进行跳转，dva.js 的 state 数据通过 connect 装饰器映射到组件的 props 中进行读取。在最后一步的表单提交中，会检查共享区域中的所有表项是否已经填写完整，只有完整才会提交。

在主页面，渲染了进度条和显示表单信息的位置，其中，表单将会加载到主页面的 children 位置，render 渲染核心代码如代码 3-2-3-1 所示。

在路由表设置时，将子页面设置在主页面的 routes 中，在装载时主页面即可通过 props 中的 children 项获取子页面信息。路由表设置部分如代码 3-2-3-2 所示。对于子页面，也为其设置了单独的路由，并把主页面的路由自动跳转到第 0 步的子页面的路由上，在渲染时，umi.js 会把主页面当作布局页，把子页面渲染在父页面中。

代码 3-2-3-1 render 的部分核心代码

```
<Card bordered={false} className={style.Card}>
    <Fragment>
        <Steps current={this.getCurrentStep()} className={style.steps}>
            <Steps.Step title="基础信息" />
            <Steps.Step title="地址信息" />
            <Steps.Step title="教育信息" />
            <Steps.Step title="学术范围" />
            <Steps.Step title="安全问题" />
            <Steps.Step title="成功" />
        </Steps>
        { this.props.children }
    </Fragment>
</Card>
```

对于进度条，当前所进行到的阶段可以通过处理 URL 获得，通过 props 中的 location 项的 pathname 项获取路由路径，使用 split 根据“/”分离字符串，由于在路由设置中每个子页面只有最后一段地址不同，所以其分离出的最后一项即为该子页面与同功能的其他子页面相区别的路由名。该部分的核心代码如代码 3-2-3-3 所示。

代码 3-2-3-2 路由表部分代码

```
// 注册
{
        path: '/user/register',
        name: '注册',
        component: './User/Register',
        routes: [
        {
            path: '/user/register',
            redirect: '/user/register/info',
        },{
            path: '/user/register/info',
            name: '基础信息',
            component: './User/Register/Step1',
        },
```

代码 3-2-3-3 获得当前步骤处理代码

```
getCurrentStep() {
    const { location } = this.props;
    const { pathname } = location;
    const pathList = pathname.split('/');
    switch (pathList[pathList.length - 1]) {
        case 'info':  // 输入基础信息（用户名、密码、etc）
            return 0;
        case 'address': // 输入地址信息
            return 1;
        case 'education': // 输入教育信息
            return 2;
        case 'academic': // 输入学术范围信息（区间 etc）
            return 3;
        case 'safequestions':  // 设置安全问题
            return 4;
        case 'result': // 处理结果
            return 5;
        default:
            return 0;
}
```

对于子页面，以设置密码部分为例。

在点击密码输入框时，需要能够弹出显示密码强度的气泡窗口，在此部分使用 Ant Design 组件库的 Popover 组件，根据输入密码的文本，在组件中渲染强度指示条。

设置好密码强度的指示文本和强度进度条组件所需的强度指示映射，如代码 3-2-3-4 所示。

在组件类中，通过正则表达式对输入密码文本的强度检查。根据预先设置的密码强度标准对密码强度进行检查并返回密码强度信息，通过代码 3-2-3-4 中设置的映射信息进行映射。如代码 3-2-3-5 所示，在强度设计上，密码长度小于 10 的为不合法，大于等于 10 的为中等，大于等于 15 且密码中含有大写字母的为高强度。

代码 3-2-3-4 强度指示文本和进度条设置映射

```
// 映射密码强度为显示的强度信息
const mapPwStrengthToStatus = {
    good: (
        <div className={style.success}>
            强度：强
        </div>
    ),
    ok: (
        <div className={style.warning}>
            强度：中
        </div>
    ),
    poor: (
        <div className={style.error}>
            强度：弱
        </div>
    )
};
// 映射密码强度为显示的进度条
const mapPwStrengthToProgress = {
    good: 'success',
    ok: 'normal',
    poor: 'exception',
};
```

在检验函数检测密码强度后，渲染强度进度条的函数将会根据上述信息进行渲染，如代码 3-2-3-6 所示。

在 Popover 中，通过 content 属性设置需要在气泡框中显示的进度条和提示文本，其表单输入框也通过自定义验证器，使用同样的密码安全验证策略来保证非法密码不会成功提交，若文本框中仍是非法代码，在提交时会提示错误信息。

最终的样式如图 3-2-3-1 所示。

代码 3-2-3-5 密码强度检测函数

```
// 检测密码的强度等级
getPasswordStrengthLevel = () => {
    const value = this.props.form.getFieldValue('password');
    const patternGood = /^\w*[A-Z]+\w*/; // 判断是否有大写字母
    const patternOK = /^\w+/ // 判断是否是合法字符（大写字母、小写字母和数字）
    if(value && value.length >15 && patternGood.test(value)){
        return 'good';
    }
    if(value && value.length >=10 && patternOK.test(value)){
        return 'ok';
    }
    return 'poor';
};
```

代码 3-2-3-6 密码强度进度条渲染

```
renderPasswordStrength = () =>{
    const { form } = this.props;
    const value = form.getFieldValue('password');
    const passwordLevel=this.getPasswordStrengthLevel();
    return value && value.length ? (
      <div className={style[`progress-${passwordLevel}`]}>
        <Progress
          status={ mapPwStrengthToProgress[passwordLevel] }
          className={ style.progress }
          strokeWidth={ 6 }
          percent={ this.getProgressPercent(value) }
          showInfo={false}
        />
      </div>
    ): null;
};
```

Popover 组件在 render 函数中的设置如代码 3-2-3-7 所示，其自定义验证器设置为 checkPassword 函数，该自定义验证器会在表单该 Input 内容发生变化时，调用验证器对表单内容进行检查，如果发现不合法，自定义验证器可以使用 callback 回调函数返回错误信息。

代码 3-2-3-7 Popover 渲染

```
<Popover
    getPopupContainer={node => node.parentNode}
    content={
        <div style={{ padding: '4px 0' }}>
            { mapPwStrengthToStatus[this.getPasswordStrengthLevel()] }
            { this.renderPasswordStrength() }
            <div style={{ marginTop: 10 }}>
                请输入至少 10 个字符的密码，包括字母和数字，建议包含大写字母。
            </div>
        </div>
    }
    overlayStyle={{ width: 240 }}
    placement="right"
    visible={this.state.visible}
    >
        {getFieldDecorator('password', {
            rules: [
            {
                validator: this.checkPassword,
            },],})(
        <Input
            type="password"
            placeholder="请输入密码"
        />
        )}
</Popover>
```

在点击“下一步”后，该页页面会调用 form 的 validateFields 函数验证该表单的数据是否符合各自的验证器要求，若符合将会调用 props 中的 dispatch 函数将数据放入 dva.js 中。

在最后一页（安全问题页面）的提交过程中，前端系统会先根据后端的要求，将安全问题的问题和答案进行拼接，以“;”分隔。之后检查 props 中 stateContent

（stateContent 是 dva.js 中存储共享数据的命名空间的 state 在最后一页组件的 props 中映射的名字）内部的项是否已经全部填写（除学术领域可只填写一个之外）。若检查无误，则调用 dva.js 的 effects 对信息进行提交，提交到后端返回结果正确后，可使用 umi 的 router 函数跳转到登录结果页面，否则通过 message 组件提示错误。

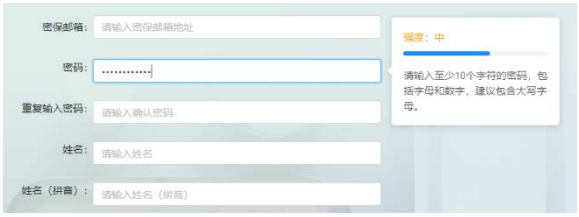

图 3-2-3-1 密码强度气泡

3.2.4 前端搜索模块的具体实现

在首页界面需要为用户提供搜索功能，包括按关键字搜索和按期刊数搜索。

在实现上，两种搜索表单共享一个搜索结果列表，但表单将分为两个组件进行编写，再加之由于使用后端分页，在分页器发生变化时必须要向后端重新请求搜索结果，所以在搜索表单提交时，主页面必须记录下当前所提交的搜索关键字，而且与后端的交互接口将只由主页面进行，两个表单使用主页面通过 props 传递的函数向后端提交结果。

对于主页面，通过设置以下变量表来保存当前搜索关键字，如代码 3-2-4-1 所示。

代码 3-2-4-1 主页面记录搜索关键字

```
let startYearByKeyword = 0;
let endYearByKeyword = 0;
let keywordByKeyword = "";
let typeByKeyword = 0;
let nowMode=0;  // 当前列表搜索是按关键词(1)还是按期刊搜索(2)
let magNoByNo=[]; // 按期刊期数查找的期数
```

以按关键字搜索为例，搜索过程被分为两个函数，handleSubmitByKeyword 负责初始化，即传递给关键字搜索表单的函数，用于初始化主页面记录和启动第一页的搜索，handleSearchByKeyword(page)负责使用已初始化的主页面记录，并根据页面参数进行搜索。

初始化过程核心代码如代码 3-2-4-2 所示。

代码 3-2-4-2 handleSubmitByKeyword 核心代码

```
handleSubmitByKeyword = (startYearByKeywordParam, endYearByKeywordParam, keywordByKeywordParam, typeByKeywordParam) =>{
    startYearByKeyword = startYearByKeywordParam;
    endYearByKeyword = endYearByKeywordParam;
    keywordByKeyword = keywordByKeywordParam;
    typeByKeyword = typeByKeywordParam;
    nowMode=1;
    this.handleSearchByKeyword(1);
};
```

表单组件通过该段代码将初始化数据存放到主页面中，之后调用 handleSearchByKeyword，正式开始搜索。搜索过程的核心代码如代码 3-2-4-3 所示。

代码 3-2-4-2 handleSearchByKeyword(page)核心代码

```
// 处理关键词搜索过程
handleSearchByKeyword(page){
    this.props.dispatch({
    type: `${namespaceSearch}/queryByKeyword`,
    payload: {
        startyear: startYearByKeyword,
        endyear: endYearByKeyword,
        keyword: keywordByKeyword,
        type: typeByKeyword,
        page: page
    },
    }).then(()=>{
        if(this.props.resultJson.result !== 1){
        this.setState({ resultList: [] });
        }
        else{
            this.setState({ resultList: this.props.resultJson.articlelist});
        }
    });
}
```

将搜索结果列表的 dataSource 设置为 state 中的 resultList，这样当上述搜索代码导致 resultList 变化时，可以重新渲染搜索结果列表。

在构造 list 时，需要设置分页器的 onChange 事件的响应函数，当用户切换页面时，需要根据预先存储的数据重新请求后端，请求指定的页面。当 nowMode 为 1 时，代表当前是按关键字搜索，所以调用 handleSearchByKeyword 请求指定的页面；当 nowMode 为 2 时，代表当前是按期数搜索，所以调用 handleSearchByNo 请求指定的页面。

对于按关键字搜索的表单，父组件会将之前存储当前搜索关键字的函数传入，搜索关键字类型通过 select 下拉菜单控制，select 下拉菜单的 option 对应的 value 一并提交后端进行搜索。在渲染表单时，需要通过存放公共参数表的 publicVaribles 和存放期刊基础信息的 magStandard 命名空间取得下拉菜单选项，并在下拉菜单中渲染。

在点击提交按钮后，表单组件将会通过预先传入的操作主页面存储关键字的函数对关键字进行转储，并执行搜索代码，渲染搜索结果。

对于按期数搜索的表单，使用 Cascader 级联选择器进行选择。为了提高加载速度和降低通讯负荷，Cascader 级联选择器会按“年份-期数”的方式进行选择，级联选择器先选择年份，之后级联选择器会与后端通讯，获得年份对应的期数列表，再渲染期数列表，最终选定后，点击“提交”按钮提交。

点击年份后，级联选择器会运行其 loadData 所指定的函数，如代码 3-2-4-3 所示，该部分代码会先获得所选择的年份项，向后端通讯获得年份对应的期数列表，把期数列表渲染为该 option 的子列表，通过 loading 属性的设置来提醒用户正在加载。最终懒加载样式如图 3-2-4-1 所示。

图 3-2-4-1 懒加载样式

代码 3-2-4-3 Cascader 懒加载函数的实现

```
// 按“期刊编号”搜索时，级联选择的动态加载
loadCascaderData= async (selectedOptions) => {
const targetOption=selectedOptions[selectedOptions.length-1];
const childrenOptions=[];
targetOption.loading=true;
await fetch("/search/getcontent/year=".concat(targetOption.value))
    .then(response => response.json())
    .then((data) => {
    if(data.result != 1){
        return;
    }
    const list = data;
    for (let i = 0; i < list.ddl.length; i++) {
        childrenOptions.push({
        label: "第".concat(list.ddl[i]).concat("期"),
        value: list.ddl[i],
        });
    }
});
targetOption.children=childrenOptions;
targetOption.loading=false;
this.setState({
    cascaderOptions: [... this.state.cascaderOptions],
});
}
```

3.2.5 前端用户管理模块的具体实现

在用户管理模块，管理员可以获得用户列表，并进行添加、修改、删除等操作。在实现上，围绕 StandardTable 组件进行。

StandardTable 组件 Ant Design 对于 Table 组件的扩展，提供了多项选择等功能。

对于列的渲染，该组件根据预先设置好的 Columns 数组对数据集进行渲染，Columns 规定的 Table 的列，在 Columns 数组中，可以通过 dataIndex 确定数据集中对应的属性，也可以通过在 render 设置箭头函数来决定在列表中以什么样的形式显示该数据。

以审稿人的管理为例，Columns 的设置如代码 3-2-5-1 所示。在 Columns 中，通过 render 属性可以很灵活地对该行数据的显示格式进行控制。

在批量处理的流程中，可以通过 StandardTable 的 selectedRows 和 onSelectRow 两个属性对多选操作进行处理，两者配合可以把被选中的 record 记录集记录到 state 中。在点击批量操作时，可以把记录集传入处理函数。以批量删除为例，点击批量删除按钮后，将会先激活 Modal，确认是否删除。在激活 Modal 时传入 record 记录集，对 Modal 的 onOk 属性进行设置，若用户点击 onOk，则可遍历整个 record 记录集，逐个向后端提交请求，对于发生错误的通过 message 进行提示。执行完毕后将重新向后端拉取用户列表。

对多选操作，可以对多选操作按钮进行动态设置。可利用&&运算符，使只有 state 中当 selectedRows 长度大于 0 时才会显示多选操作按钮。如代码 3-2-5-2 所示。

对于添加用户，其分步表单与注册环节类似。在实现上需要注意的是表单数据的临时存储和分步表单的页脚设置。

在激活 Modal 时，设置 values 属性，将主页面 state 中的 stepFormValues 传入 Modal。Modal 根据 stepFormValues 中的数据和进度，渲染该进度的表单和进度条。对于在最后一步前，每一个下一步按钮的 onClick 事件都要检查当前表单的数据是否符合校验器要求，若符合则通过主页面传入的接口将数据保存到主页面的 state 中。在最后一步中，需要检查 Modal 的 state 中所存储的数据是否已经填写完整，若已经填写完整，则利用主页面传入的接口向后端提交，并重新拉取列表。

用户管理页面的样式如图 3-2-5-1 所示。

图 3-2-5-1 用户管理页面样式

代码 3-2-5-1 Columns 设置

```
columns = [
    {
      title: '用户名',
      dataIndex: 'username',
      render:      username      =>      <a      onClick={()      =>
this.getRefereeItem(username)}>{username}</a>,
    },{
      title: '姓名',
      dataIndex: 'name',
    },{
      title: '性别',
      dataIndex: 'gender',
      render: gender => gender == 1 ? "男" : "女" ,
    },{
      title: '操作',
      render: (text, record) => (
        <Fragment>
          <a onClick={() => this.handleResetPasswordOneLine(record)}>重
置密码</a>
          <Divider type="vertical" />
          <a    onClick={()    =>    this.handleDeleteRecordOneLine(true,
record)}>删除</a>
        </Fragment>
      ),
    }
];
```

代码 3-2-5-2 批量操作按钮的动态渲染

```
{this.state.selectedRows.length > 0 && (
    <span>
        <Button onClick={this.handleDeleteRecordLines}>批量删除</Button>
        <Button   onClick={this.handleResetPasswordLines}> 批 量 重 置
    </Button>
    </span>
)}
```

3.2.6 前端动态表单的具体实现

在投稿人添加新文章时，对于作者可能需要填写不固定数量个作者，所以在前端实现动态的添加、删除和修改很有必要。

在实现上，参考 Ant Design Pro 框架的高级表单进行编码。[2]

如 3.2.5 节中所述，Table 的每一行 record 都可以通过 render 对显示的模式进行控制，故可以在数据集中，为每一行添加一个属性 editable，在 render 中，若该 record 属性为 editable，则以 Input 组件进行渲染，否则只渲染文本。

对于每一条数据，有一个唯一的 key 与其匹配。在组件中有一个 key 用于标记下一个可用 key 的数值，当添加新条目时，将该条目的键设置好，并使 key 加 1。将新条目加入到数据集中即可渲染。

在保存时，需要先根据校验器验证输入是否合法，若合法则更改 editable 属性，并存入数据。

当修改数据时，需要先将原有数据存入临时变量中，需要注意的是，由于有可能会有多条数据同时处于编辑状态，所以临时变量需要是一个数组，并且以 key 为键。

动态表单的样式图如图 3-2-6-1 所示。

图 3-2-6-1 动态表单的样式图

3.2.7 业务操作表单和微信催促的具体实现

对于每一个参与业务流程的用户，其在渲染任务列表时需要能够通过业务状态对任务列表中的任务进行过滤。由于登录系统的用户主要目的是完成业务流程，所以在渲染列表时默认渲染当前待完成的任务。

在任务列表中，使用 Ant Design 的 Card 组件对页面进行分区，页面将分为上、下两部分，上部渲染搜索表单，下部渲染搜索列表。

搜索表单即用 Select 组件渲染一个下拉菜单，下拉菜单的选项为业务状态列表，每一个下拉菜单选项都对应了一个 value 值，根据 value 值前端可以确定选择了哪一个表项，并翻译成状态码提交给后端。

对于下部的搜索结果列表，与 3.2.5 节的实现方式类似，使用 StandardTable 进行渲染。为了方便显示，可以在 Columns 中，对于当前任务状态的渲染进行设置。

在 Columns 的状态部分，如代码 3-2-7-1 所示。在渲染时，将当前这条数据记录（record，即后端发送来的文章信息）传递给预先设置好的函数 getStatus，getStatus 将根据 record 的记录，返回一个设置好的 Badge（徽标）组件，Badge 用一个带有颜色的点和文本进行展示，所以可以用来以颜色和状态文本向用户显示当前状态，方便识别，Badge 的渲染代码如代码 3-2-7-2 所示，Badge 示例图如图 3-2-7-3 所示。

代码 3-2-7-1 当前任务状态的 Columns 设置

```
{
    title: '当前状态',
    render: (text, record) => {this.getStatus(record)},
},
```

代码 3-2-7-2 渲染 Badge

```
if(record.role == 4){
    return <Badge color='pink' text="审稿中" />
}
```

图 3-2-7-1 Badge 示例图

在 render 中，若用户点击一篇文章的标题时，将会触发函数，将该条记录相关的 article、invoice 和 task 记录放入公共共享空间中，并跳转到详情（TaskDetail）页面，该页面会从公共共享空间中获得数据并渲染。

在后端返回的 invoice 和 task 数组中，每一项都有一个 id_article 项，该项会记录 invoice 和 task 所对应的 article 的 ID。通过从 record 中读取出 article 的 ID，再进行遍历从而获得对应的 task 和 invoice，之后将信息通过 dispatch 放入 dva.js 的公共共享控件中，如代码 3-2-7-3 所示，在遍历结束后，将 article、invoice 和 task

放入 task 中，之后放入共享空间中，在放入成功后，回调函数会将页面跳转到详情页面。

代码 3-2-7-3 传递共享数据

```
const payload={
      article: record,
      task: task,
      invoice: invoice,
};
this.props.dispatch({
    type: `${namespacePublicStorage}/storePublic`,
    payload: payload,
    callback: ()=>{
        router.push(`/author/taskdetail`);
    }
});
```

在详情页面，将 dva.js 中所设置的公共共享空间的 state 内容通过 connect 装饰器映射到详情页主页面的 props 中。页面使用 PageHeaderWrapper 和 Card 将页面分为上、中、下三部分。上部显示文章基础信息，使用 Ant Design 的 DescriptionList 来显示详细信息。中部使用 Steps 组件渲染进度条，下部根据进度的不同渲染不同的表单。

由于每个业务阶段所使用的表单不同，所以将每个页面的表单单独写成一个类，放在主页面文件中。在从 dva.js 中获取到表单信息后，先把 article 中的状态码映射为 Steps 组件中的步骤值，供 Steps 渲染。并根据当前的步骤找到该步骤所对应的表单进行渲染。对于每一个表单类使用 Form.create() 装饰器对 Form 表单进行初始化，使其可以使用校验器主动校验并向后端提交数据。

对于未在当前用户需要处理的状态的文章，在用户点击进入详情页面后，只会显示文章基础信息和进度条，不会显示表单。但由于编辑在业务逻辑中处于枢纽位置，所以编辑在点击进入详情页面后，对于不是其当前需要处理的文章，在表单的位置会显示当前正在处理该业务的用户，并显示该用户需要处理的截止时间。对于距离截止时间在 10 天以内的，编辑可以点击按钮进行提醒。

在点击提醒后，前端会向后端发送报文，报文中包含该 task 的 id。后端在收到报文后，会根据 task 的 id 查找当前正在执行该任务的用户 id 和任务进度，进入数据库用户表查找用户对应的 OpenID，以进行消息的发送工作。

对于消息的发送，需要预先在微信后台设置好要发送的模板消息，模板消息中包含需要发送的文本模板，在实际发送时，只需要填写模板中需要填写的项，如用户名、需要执行的业务等。

发送信息需要获取 AccessToken，不同于 3.2.2 节的 AccessToken，该 token 用于公众号执行业务操作。通过 GET 方法提交 APPID 和 APPSECRET 向 API 请求，请求得到后有效 7200 秒，需要在后端设置定时刷新。代码如 3-2-7-4 所示。在后端类中建立 accessToken 结构，其中包含 date 和 AccessToken 数据，每一次获取 date 后便更新 date，当发现当前时间距离 date 大于 7200 秒后，执行更新操作。

在获取 AccessToken 后，通过 POST 方式向微信 API 发送消息发送请求，POST 报文如代码 3-2-7-5 所示。touser 项为之前在数据库中获得的 OpenID，template_id 为预先在微信后台设置好的模板 ID，data 为模板要填写的内容。data 中的每一个项的名称为其在模板中的占位符，value 为要填写的内容，color 为填写的内容所要显示的颜色代码。发送报文后，后端向前端返回报文进行提示。

代码 3-2-7-4 更新 AccessToken

```
nowDate = new Date();
if (accessToken.getDate() == null || (nowDate.getTime() -
accessToken.getDate().getTime()) / 1000 >= 7200) {
    accessToken.setDate(nowDate);
    jsonObject = httpGet(urlToken);
    if (jsonObject.getAsString("access_token") == null) {
        response.sendRedirect("/library");
        return;
    }
    accessToken.setAccessToken(jsonObject.getAsString("access_token"));
}
```

代码 3-2-7-5 发送模板消息的 POST 报文

```
{
    "touser":"XXXXXXXXX",
    "template_id":"YYYYYYYYYY",
    "data":{
        "name": {
            "value":`${username}`,
            "color":"#173176"
        },
        "note":{
            "value":"您的业务即将到达处理截止日期，请及时处理",
            "color":"#173176"
        }
    }
}
```

3.2.8 前端公告管理模块的具体实现

在设计中，公告需要使用富文本的方式进行渲染，在实现上，使用 wangeditor 富文本编辑器进行实现。

在页面加载时，需要在 render 函数中使用 div 标签先占位，使用 ref 标记好此 div 方便 react 进行后续处理。在初始化时，将 this.refs.editorElem 作为参数传入富文本编辑器的的构造器进行挂载（editorElem 是该 div 标签的 ref 标记名）。

由于富文本编辑器需要上传数量不等的文件，需要设置富文本编辑器的上传接口，使之上传文件时可以通过后端接口进行上传。

配置代码如代码 3-2-8-1 所示，通过 customConfig.uploadImgServer 设置后端的上传接口地址，并设置最大同时上传数量。之后可使用箭头函数来配置 customConfig.customUploadImg，即自定义上传动作，insert 函数是富文本编辑器提供的，用于将图片插入到公告中。

需要注意的是，由于后端需要定时清理未被使用的图片，所以在上传时需要提供公告内的图片文件名，在从后端返回结果后需要将文件名放入 pic 数组中待后续上传时提交。

代码 3-2-8-1 上传图片接口的设置

```
editor.customConfig.uploadImgServer="/admin/uploadpic";
editor.customConfig.uploadImgMaxLength=1; // 最多同时上传一张图片
editor.customConfig.customUploadImg = (files,insert) => {
    if (files[0]) {
        const payload = new FormData();
        payload.append("file",files[0]);
        fetch('/admin/uploadpic', {
            method: 'POST',
            header: {
                'Content-Type': 'multipart/form-data'
            },
            body: payload
        }).then((res) => {
            return res.json()
        }).then((res) => {
            const result = res.result;
            if (result) {
                insert("/static/pic/".concat(res.url));
                this.pic.push(res.url);
            } else {
                message.error("上传错误");
            }
        })
    } else {
        message.info('请选择要上传的图片');
    }
};
```

在修改公告时，可以从后端获得数据后，使用富文本编辑器的 txt.html(content)来将接收到的富文本填入富文本编辑器中。在提交时，可以使用 txt.html()来获得富文本编辑器中的内容。

需要注意的是，在提交时，由于是富文本可能会有受到恶意攻击的危险。所以需要对危险脚本进行过滤，在前端中，使用 xss.js 工具对文本进行过滤。通过 xss(content)函数过滤的文本才能进行提交。在后端，使用 Jsoup 进行过滤。Jsoup 的 Whitelist 可以初始化白名单，初始化完毕后，使用 Jsoup 的 clean 将 html 代码和白名单作为参数传入，获得的结果即为过滤后的代码。

3.2.9 文件的上传与下载的具体实现

因为后端接口大部分文件都是随着表单一起上传的，所以 Ant Design 的 Upload 组件不太符合使用场景，因此，文件的上传主要基于 HTML 的 input 标签进行。

在上传位置，render 函数所使用的 HTML 标签如代码 3-2-9-1 所示，为了显示上的美观和风格统一，将 input 组件设置为隐藏。上传按钮的 onClick 事件所对应的响应函数，实质是调用 input 的 click 函数。在选定文件后，触发 input 的 onFileChange 事件，从 event.target 中的 files 数组获取选择的文件，将其二进制数据和文件名放入 state 中，待提交时一并上传。

代码 3-2-9-1 上传文件设计

```
<Button onClick={this.uploadFile}>选择附件</Button> <br />
{ this.state.filename }
<input type="file" id="uploadFile" name="file"
onChange={this.onFileChange} style={{display:'none'}} ref="uploadBox"
/>
```

上传时，将文件和同时需要上传其他数据，以 FormData 的形式进行上传，传递给后端处理。

下载上，由于直接跳转到后端界面下载会弹出新页面且前端控制路由会导致直接跳转会跳转到错误的接口上，所以使用 a 标签进行下载，下载部分的核心代码如代码 3-2-9-2 所示（fetch 部分略）。主要注意的是，由于 Firefox 浏览器对 a 标签的 download 功能支持有问题，需要进行兼容处理。

代码 3-2-9-2 下载文件

```
(response)=>{
    response.blob().then(blob => {
        const downloadLink=document.createElement('a');
        document.body.appendChild(downloadLink); // 兼容 Firefox
        downloadLink.style.display='none';
        const url=window.URL.createObjectURL(blob);
        downloadLink.href=url;
        downloadLink.download=response.headers.get('Content-
Disposition').split(";")[1].split("filename=")[1];
        downloadLink.click();
        document.body.removeChild(downloadLink);
    })
}
```

3.2.10 CSV 文件读取的具体实现

由于逐个录入用户信息对于管理员来说可能比较麻烦，所以需要进行设计，使可以通过上传 csv 文件的方式进行批量导入。

为了节省空间和便于录入，决定选用 csv 文件（逗号分隔值文件）作为上传的表格文件。使用 JavaCSV 库作为 csv 文件的读取工具。在 Maven 库中添加 net.sourceforge.javacsv 依赖后，建立添加用户的接口以完成注册工作。

以添加编辑为例，核心代码如代码 3-2-10-1 所示，在前端上传了 csv 文件后，将其存储在临时位置后，建立 CsvReader 对象，该对象可以每次读取一行数据，即一个记录。然后按照列名读取该条记录对应的字段，整理好后交由 Service 类调用 MyBatis 写入。

代码 3-2-10-1 批量写入代码

```
try {
    CsvReader csvReader = new CsvReader(filePath);
    csvReader.readHeaders();
    while(csvReader.readRecord()){
        String username=csvReader.get(“username”);
        String password=userService.getHash(csvReader.get(“password”));
        String name=csvReader.get(“name”);
        String gender=csvReader.get(“gender”);
        userService.insertNewUser(username, password, name, gender);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

之后会删除临时 csv 文件，批量建立用户过程结束。

3.2.11 银行卡安全的具体实现

由于审稿人修改银行卡信息需要先回答安全问题，在回答安全问题之后才可以修改，为了尽量避免记录状态，在回答安全问题之后，后端会返回银行卡信息和一个 token，前端可以凭此 token 进行修改银行卡信息。

在 token 的设计上，共分为两部分——payload（数据载荷）和 signature（签名）两部分。payload 中包含了 username（用户名）和 expireTime（过期时间）两个字

段，signature 包含了 payload 通过服务器私钥进行的 RSA 签名。

在安全问题验证完毕后，后端会构造一个 JSON Object，用于存储 username 和 expireTime，expireTime 一般为 5 分钟，之后将这个 JSON Object 转为 String 字符串并进行 base64 编码，编码后使用服务器密钥对 base64 编码进行 HMAC-SHA256 签名，将签名与 payload 使用“.”进行拼接，与银行卡信息一并返回给前端。

前端在提交修改信息时，连同此 token 一并提交。提交后后端将先会对 token 进行验证，验证后将 base64 编码转为 JSON。验证 token 是否过期以及用户名是否与 session 中的一致，若一致则执行修改过程。

在修改过程中，银行卡信息修改后，对银行卡信息加盐后使用服务器密钥计算 HMAC-SHA256 散列值并存储在数据库中用于标识正确性。在后端计算的过程中，使用 javax.crypto.Mac 类库进行 HMAC 散列值计算工作。

3.2.12 图像搜索的具体实现

对于已经上传的文件，需要先提取出图片。以 PDF 文件为例，使用 Apache PDFBox 对文件中的图片进行提取。

对于传入的 File 文件，先使用 PDDocument.load(file)进行读取赋值给 PDDocument 类型的 document，使用 document.getDocumentCatalog().getAllPages()获取每一个页面，使用迭代器进行遍历。对于每一个页面（page），可使用 page.getResource()获取资源（resource），对资源使用 resource.getImages()获取图片的 map，再进行遍历获得图片。

图片在获取后，使用 OpenCV 库对每张图片求方向梯度直方图（HOG）。

由于加密与搜索算法使用了矩阵，为方便进行运算，可使用 Apache Math3 库进行线性代数运算。

在系统部署时，系统会使用随机数生成两个可逆矩阵 M1 和 M2，以及 M1、M2 的逆矩阵和 M1、M2 的转置的逆矩阵。还要随机生成向量 S 和向量 w，用于扩展维度。

图片加密向量 p 和待搜索的图片向量 q 可采用类似的方法进行处理，只是在扩展维度上有所不同，对向量 S 进行扫描，p 的第 i 维是当 S 的第 i 维为 1 时为 w 的第 i 维，否则为一个随机数；q 是当 S 的第 i 维为 0 时做类似处理。

我们把计算两个向量 p 哪个距离 q 更近的方法称作运算符 A，该运算符相当于

比较符，当 p_1 比 p_2 距离 q 更近时，即 $A(p_1, p_2)>0$ 时，称作 $p_1<p_2$。该运算符的实现代码如代码 3-2-12-1 所示。

将此运算符配合排序算法或其他其它算法相结合，可以查找出比较接近的图片特征向量及其对应的文章 ID。

代码 3-2-12-1 比较运算符

```
private boolean isnear(int a,int b)  //判断哪一个向量更接近查询向量 true: a<=b  false: a>b
{
    double t,t1,t2;
    int i;
    t1=t2=0.0;
    if(a==b)
        return true;
    for(i=0;i<dp;i++)
    {
        t1+=(pa[a][i]-pa[b][i])*qa[i];
        t2+=(pb[a][i]-pb[a][i])*qb[i];
    }
    t=t1+t2;
    if(t>=0)
        return true;
    else
        return false;
}
```

第四章 知识技能学习情况

在本次毕业设计项目中，本人在前端和后端部分都做了一些工作，包括系统全部前端系统代码的编写以及部分后端系统代码的编写。

在后端系统上，我们采用了业内比较常用的 Java-Spring Boot-MyBatis 框架，以 Spring Boot 作为服务器驱动框架，以 MyBatis 作为数据库服务器接口来进行编码。由于之前就参与过其他一些后端项目的设计与编写，所以对此部分比较熟悉，遇到的困难相对来说比较小。

对于前端的编写则是第一次，由于之前并未接触过 React 和其他相关前端框架，在短时间内学习并编写出可以使用的系统是一个比较大的挑战。在编写的过程中也遇到了不少的问题，但通过查阅文档和资料也逐一解决了问题。

React 是由 Facebook 开发的，目前使用比较广泛的前端框架，其使用 JSX 语法、以声明式编写 UI，非常高效和灵活，且便于调试。它将页面上的元素组件化，这些组件可以相互传递数据，简化了 UI 逻辑。本次毕业设计在 React 的基础上使用了 Ant Design 组件库，Ant Design 作为基于 react.js 的组件集，其组件样式美观，也易于定制，在此基础上，其衍生框架 Ant Design Pro 对 Ant Design 组件库进行了进一步封装，以可以很方便地实现更复杂的逻辑。

dva.js 是一个基于 react-redux 的数据驱动框架，其 state 位于 react 顶层，可以供各组件共享数据。其使用命名空间对该 state 进行划分，并使用命名空间作为 key，获取该命名空间 state 中所存储的信息。组件可以通过 connect 函数把 state 中的信息映射到组件的 props 中，组件通过 props 获取 dva.js 内的 state 信息。

在 dva.js 中，使用 effects 执行异步通信，即前端系统可以通过 effects 中的函数，访问后端系统异步获取结果，并使用 reducers 更新该命名空间的 state，实现数据更新。

在此次开发的过程中，对于前端系统如何进行设计和编码学到了很多，在之前后端知识积累的基础上，对前端知识进行了学习和应用，对我的工程思想训练颇多。由于前后端的知识都有所掌握，所以在编写项目上可以更加得心应手。在业余时间，也利用 React 开发了一些应用的前端系统，感受到了便利。

第五章 工程交流协作情况

在项目执行的过程中，涉及到一些与编辑部专业知识和工作流程相关的问题，我们在进行完善的需求分析的基础上，与编辑部方面进行了合作，在实地调研了编辑部的工作流程，对于编辑部的专业知识进行了了解。在调研的过程中，进一步明确了业务流程，对我们的编码十分有帮助。

在前后端的协作上，前后端在进行完整各自部分自主测试的基础上，通过编制符合规范的接口文档来更方便地进行对接与合作。前后端组有每周两次的碰头会，双方对不协调之处进行协商和调整。在文档中提供了前端需要访问的 URL 以及要提交的 JSON 内容，最后有对应的返回值和样例。

我也参与了前期的需求分析和设计环节，在其中与前辈、同事一起与客户面谈，编制需求文档，学到了很多除了编码之外的其他工作方法，为以后更加良好的进行软件工程设计打下了基础，提高了工程素养。

第六章 工程计划管控与执行情况

本次毕业设计过程中，我按时完成了公司交给我的各项任务，并编制了合格的文档供参考备案。

公司交办的任务有：

1. 医学论文投审系统：

（1）参与需求分析；

（2）系统前端系统和部分后端系统的设计工作；

（3）首页部分设计与编写；

（4）后台部分通用模块的设计与编写；

（5）后台部分业务流程处理的设计与编写；

（6）前、后端微信通知和用户管理（微信部分）的设计与编写；

（7）图像搜索的探索，初步设计与实现；

（8）系统中各部分相关安全方案的设计与实现。

我们根据公司要求，每周一向组长提交工作周报，叙述了上一周的工作成果和计划执行情况，以及本周的任务要求，保质保量完成每一周的工作任务。

第七章 职业素养与工程伦理

通过此次毕业设计，我对软件工程技术的应用有了更深的理解。应用了在学校内学到的软件工程方法。通过公司提供的需求分析文档，根据软件工程方法对软件进行建模，所有成员根据规范编制了符合要求的接口文档，提高了人与人之间合作的执行效率。

相比较个人之前的独立软件编程，软件工程更适合大型软件系统、多人协作的设计。大型软件系统个人的能力是不够的，容易让人感觉力不从心，而合作需要有一套完整、高效的规范去遵守，提高效率，而软件工程方法正是提供了这种方法。

通过软件工程方法，显著地提高了我们软件的编写效率，最终能够按期向公司提供我们编写完成的软件产品。

在这次毕业设计中，我们也注意了可扩展性的提高，变量设计、命名都方便维护，注意在编写的过程中对代码进行注释。主要采用接口方式进行调用和通信，方便后续的人员对整个系统模块间的业务逻辑进行维护。

软件工程对人类软件技术的发展有着突破性的作用，软件工程提供了一套人与人之间协作编程的方法论，使我们告别了个人编写软件的时代，在个人编写软件的过程中，代码自己认识即可，丧失了可维护性，同时代码体例也不会很大。软件工程方法使得协作效率提高，代码更加方便更改。软件代码质量提高。而软件质量的提高，对社会日常生活提供了更多的便利，使社会能继续高速前进。

第八章 对软件工程实践以及软件工程领域发展的认识

这次毕业设计我们应用了软件工程方法，用一套共用、高效的规则来编写程序，效率、速度和代码可分析性都比之前的个人编写程序有了很大的提升。相信，如果业内继续广泛应用并改进现有的软件工程方法，将会使软件的质量更上一个台阶。

在本次毕业设计中，我们在每一个阶段都编制了完整且规范的文档，在整个软件开发过程中，文档写作部分占据了相当一部分时间。这相比较于以前我们对软件开发的认识有着很大的不同。但正是因为这些文档，才使得各组的协作十分紧密且稳定，减少了出错的概率，提高了业务处理的效率。

随着软件质量提高，它会促进使用这些软件的社会的发展，提高发展速度。同时，软件也会随着社会需要的变化，依照软件工程方法进行更快、更好的转变，使之更加适应社会的发展，进一步提高社会发展的水平。促进环境与社会的可持续发展。

第九章 结束语

在本次毕业设计中，本人按照公司时间要求和质量要求，保质保量地完成了公司的任务要求，并解决了其中发现的问题和 bug。

本次毕业设计课题还有进一步发展的空间，包括对用户使用便利度上的提升，如在支付环节上，除原有的汇款支付外，可以增加在线支付功能；对沟通方式进一步扩展，例如增加站内聊天功能，方便编辑与投稿人、编辑与审稿人的沟通和记录；对安全功能的改进，对图像安全搜索提升速度，提高识别率等。

我在这次实习中学到了很多，包括软件框架的使用方法，一些 UML 设计工具和数据库设计工具的使用方法，在以前熟悉后端系统编写的基础上，学习到了前端系统如何进行设计和编写。还提高了自己的包括图形设计、文档协作等软件工程能力，整个毕业设计的过程中我受益良多。

参考文献

[1] W. K. Wong, David W. Cheung, Ben Kao, et al. Secure kNN Computation on Encrypted Databases[C]. SIGMOD’09, Providence, Rhode Island, USA, June 29-July 2, 2009, 139-152

[2] Ant-Design: AdvancedForm[EB/OL].
https://github.com/ant-design/ant-design-pro/blob/master/src/pages/Forms/AdvancedForm.js

[3] Ant-Design: Ant Design Document[EB/OL].
https://ant.design/docs/react/introduce

[4] Ant-Design: Ant Design Pro Document[EB/OL]
https://pro.ant.design/docs/getting-started

[5] Project dva.js: dva.js Document[EB/OL]
https://dvajs.com/guide/

[6] Project umi.js: umi.js Document[EB/OL]
https://umijs.org/guide/

致谢

在本次毕业设计的过程中，电子科技大学信息与软件工程学院钱伟中老师和四川安若泰科技有限公司田野老师在选题、设计、实现和论文写作上都向我给予了莫大的帮助，耐心地对我在毕业设计中遇到的问题和困难进行指导和帮助。二位老师渊博的学识、严谨的态度深深地影响了我，在此由衷地感谢二位老师对我的毕业设计的支持和帮助。

此外，感谢我的家人和朋友在我大学本科学习期间的支持与鼓励。还要感谢那些从陌生到熟悉，又从熟悉到陌生的人，他们使我意识到了自身的不足，逐渐地走向成熟，也让我深深地体会到了“花开堪折直须折，莫待无花空折枝”所隐藏的深意，可能没有机会当面表达谢意，故在此表示深深的感谢。