广州大学华软软件学院

South China Institute of Software Engineering,GU

毕业论文(设计)

课题名称 基于Electron跨平台的"学生作品展示"应用的设计和开发

系别 游戏系

专业(方向) 网络与新媒体(新媒体设计与开发)

班级 14网络与新媒体2班

学生姓名 陈秋龙

指导教师 袁冠远

完成日期 2018年5月10日

教务处 制

毕业论文（设计）真实性诚信保证书

本人所撰写的毕业论文（设计）是在老师指导下独立完成，没有弄虚作假，没有抄袭或拷贝他人研究成果。我郑重承诺：文责自负。

签　名：陈秋花

2018年 5月 10日

广州大学华软软件学院

本科毕业设计任务书

设计题目 基于Electron跨平台的

“学生作品展示”应用的设计和开发

系　　别 游戏系

专　　业 网络与新媒体

班　　级 14网络与新媒体2班

学　　号 1440624214

学生姓名 陈秋龙

指导教师 袁冠远

下发时间：　2017 年　10 月 24 日

毕业设计须知

1、认真学习和执行广州大学华软软件学院学生毕业论文（设计）工作管理规程；

2、努力学习、勤于实践、勇于创新，保质保量地完成任务书规定的任务；

3、遵守纪律，保证出勤，因事、因病离岗，应事先向指导教师请假，否则作为缺席处理。凡随机抽查三次不到，总分降低 10 分。累计缺席时间达到全过程 1 / 4 者，取消答辩资格，成绩按不及格处理；

4、独立完成规定的工作任务，不弄虚作假，不抄袭和拷贝别人的工作内容。否则毕业设计成绩按不及格处理；

5、毕业设计必须符合《广州大学华软软件学院普通本科生毕业论文（设计）规范化要求》，否则不能取得参加答辩的资格；

6、实验时，爱护仪器设备，节约材料，严格遵守操作规程及实验室有关制度。

7、妥善保存《广州大学华软软件学院本科毕业设计任务书》。

8、定期打扫卫生，保持良好的学习和工作环境。

9、毕业设计成果、资料按规定要求装订好后交指导教师。凡涉及到国家机密、知识产权、技术专利、商业利益的成果，学生不得擅自带离学校。如需发表，必须在保守国家秘密的前提下，经指导教师推荐和院领导批准。

课题名称	基于 Electron 跨平台的“学生作品展示”应用的设计和开发

完成日期：　　2018 年　4 月　10 日

一、题目来源及原始数据资料：

题目来源：科研

原始数据资料：

1. 近些年来，随着 Javascript 在网页，手机，pc 平台领域的发展，基于 Electron 跨平台技术将会带来高效开发，节约成本，一处编写，多端运行的好处，
2. Node.js 语言的发展，Javascript 也能像传统的后台语言一样操作和管理用户本地的资源。
3. 游戏系学生急需一个作品网站和应用来展示他们的优秀作品。

二、毕业设计要求：

1. 开发技术：Vue.js+Electron
2. 程序有良好的设计风格和代码规范
3. 了解与跨平台相关的软件技术
4. 程序能正确地运行
5. 进行必要的调研和资料搜集、文献阅读；
6. 要有完整的开发文档
7. 有应用的详细说明文档
8. 每周有开发进度报告

项目目标：

1、后台管理员登录注册，对管理员和用户的管理

2、前台登录注册和个人信息的增删改查

3、作品的上传，下载，修改，显示和删除，后台对作品的审核，修改，发布和删除

4、作品排行，搜索作品，作品的点赞和评论，后台对于评论的增删改查

5、批量作品导入和导出，批量人员的导入和导出

6、新增数据分析模块，使用图表展示用户作品的一些分析

7、升级后台评论管理模块，增加的敏感词管理

8、登录注册模块的升级，对接学校系统

9、登录注册模块的升级，对接学校系统

10、升级作品发布富文本编辑器等，进一步优化作品发布

11、前端进行 Electro 打包，正式上线，交付

三、进度安排、应完成的工作量：

1. 2017 年 10 月 23 日前，下达任务书。
2. 2017 年 10 月 30 日前，上交任务书，完成毕业论文开题，开始项目设计。
3. 2017 年 12 月 5 日前，完成项目第一版
4. 2018 年 1 月 20 日前，完成项目第二版，开始撰写毕业论文。
5. 2018 年 2 月 10 日前，提交毕业论文的初稿。
6. 2018 年 3 月 10 日前，提交毕业论文的完成稿。
7. 2018 年 3 月 15 日前，提交毕业论文的打印稿。
8. 2018 年 3 月 24—3 月 30 日，组织毕业论文（设计）答辩及成绩评定。

四、主要参考文献

[1] 张佳伟,张涛,周叶. 基于 Electron 的跨平台客户端技术[J]. 智能计算机与应用,2017,7(03):120-122.

[2] Muhammed Jasim. Building Cross-Platform Desktop Applications with Electron

[3] Node.js 6 紧跟 V8 引擎更新,提高了速度和安全性[J]. 电脑编程技巧与维护,2016,(09):4.

[4] 朱丽英. 基于 Node-Webkit 平台的 JavaScript 工具集研究与实现[D].电子科技大学,2016.

[5] 王姝文. 嵌入式浏览器跨平台服务组件研究与设计[D].电子科技大学,2011.

[6] 戴翔宇. Web 前端工程组件化的分析与改进[D].吉林大学,2016.

[7] 吴森林,张玺. Weex 跨平台开发方案研究与应用[J]. 信息通信,2017,(04):112-113.

[8] 唐成. 基于 Blink 的 WebCL 基础模块的设计和实现[D].中山大学,2015.

[9] 黄雄. 基于 HTML5 的视频音频传输技术的研究与设计[D].广东技术师范学院,2014.

[10] 常闻宇. HTML5 跨平台技术在视频点播系统中的研究与应用[D].东华大学,2013

[11] 王玉平. 视频、音频文件跨平台播放的研究与实现[D].中国地质大学(北京),2016.

[12] 麦冬,陈涛,梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版),2017,(07):58-59.

[13] 王胜,张靖. 基于 Vue.js 高速路政管理系统的设计与实现[J]. 电脑知识与技术,2017,13(21):86-88+101.

[14] 徐岫,朱广华,贾瑶. 基于 VueJs 的 WEB 前端开发研究[J]. 科技风,2017,(14):69.

指导教师（签名）：	系（教研室）主任（签名）：
	罗林

广州大学华软软件学院
本科毕业论文（设计）成绩单

姓名	陈秋龙	学号	1440624214	专业（方向）	网络与新媒体（新媒体设计与开发）
题目	基于 Electron 跨平台的“学生作品展示”应用的设计和开发				
指导教师评语（占论文总成绩的60%）	该生在毕业论文（设计）制作期间，态度积极，和另一位同学组队完成了跨平台应用“学生作品展示”的设计与开发。主要负责前端实现，作品功能较全面，目前已经在游戏系试用。开发中使用了组件化思想及面向对象设计，设计程序结构合理。论文写作中，论述内容有条理，有充分的论据，能够按照学校格式要求编写，详细设计中对流程图、游戏功能结构表述较为清晰。对“学生作品展示”应用前端开发中的常用技术进行了深入研究和总结。建议把部分流程图放到概要设计效果更好，概要设计应有清晰的总体类关系图，另外测试部分缺乏跨平台的测试记录。 成绩：54 签名：[illegible] 2018年 4月 6日				
评阅教师评语（占论文总成绩的20%）	该生利用 js 语言的跨平台特点，开发出一款跨 window, macOs, linux 系统，兼容 PC 和手机的应用。使用这款应用，学生能有效发布作品，促进学生之间交流，也能加强学校老师和学生和外部企业外部人士之间的联系。论文写作中，论述部分有条有理，并有较多论据；格式也较为规范。 成绩：18 签名：[illegible] 2018年 4月 8日				
答辩组评语（占论文总成绩的20%）	在答辩过程中，该生演示了完成的作品，效果比较专业，具备较强的实用性；论文答辩中表达明确，对答辩老师的提问，能给出合理的解释；对程序代码熟悉。论文论述有条理，有充分的论据，格式严格按照要求，希望概要设计能密切结合自身作品展开，并突出作品的创新之处。 成绩：18 答辩组成员：[illegible] 2018年 4月17日				
总成绩	该生总成绩为优(90 分)。 系主任签名：[illegible] 系(章) 2018年 5月 10日				

摘要　本文主要讲解学生作品展示系统的实现过程，从需求分析，项目架构，系统实现，到部署上线。项目开发的流程考虑到了多人协作，团队管理，代码维护及复用性。利用 js 语言的跨平台特点，开发出一款跨 window,macOs,linux 系统，兼容 PC 和手机的应用。使用这款应用，学生能有效发布作品，参加大赛，也能促进学校老师和学生还有外部企业外部人士之间的联系。最后，在此过程中个人总结了一套符合迭代开发要求的开发模式，利用这一套模式，完全能实现只写一套代码，多端运行。

关键词　Electron；Vue.js；Node.js；Webpack；作品展示平台

ABSTRACT This article mainly explains the implementation process of student work display system, from requirements analysis, project architecture, system implementation, to the deployment of the line. The project development took multi-person collaboration,team management,code maintenance and reusability into account. With thecross-platform features of the JS language,an application which can be compatible PC as well as mobile and its system cross Window,macOs and linux were exploit.Not only student can publish works effectively and take participate in competitions,but also it can promote the relation between school and the outside people. Last but not least, I have summarized a set of development models of requires of iterativedevelopment.Using this series of models,it is entirely possible to write only a setof code and run it in multiple ways.

KEY WORDS Electron; Vue.js; Node.js; Webpack; Workshop Display Platform

目 录

第一章　绪 论

1.1　开发背景

JavaScript 有着最广阔的应用环境，以及数量众多的开发者。这些开发者虽然雄据着最大的应用平台，但野心仍未满足，不断试图继续开疆拓域。于是，一个个原本不属于 Web 开发的领域被生生挤入，Electron 就是我们向桌面端进军的尝试。Electron 是一个开发平台，或者说打包工具。它以普通网页作为软件 UI，辅以计算逻辑和本地系统交互，实现和原生桌面软件接近的使用体验。随着 Web 技术的不断发展，它能够实现的效果也越来越好，功能也越来越丰富。更重要的是，这样的技术方案可以大量复用网页代码，极大节省开发人力。

Electron 不是第一款这样做的产品，但眼下看来应该是最成功的一家。它背靠 GitHub，以 Atom 作为开山之作，已经超过前作 NW.js 成为最热门的 JS 桌面端开发方案。

1.2　研究的意义

随着 Javascript 的不断壮大，定能解决桌面端开发门槛高、人力成本高等各种痛点。Visual Studio Code 的出现证实了这一点，Visual Studio Code 也是使用 Electron 开发的，凭着能做到大项目秒开，万行代码也不会出现卡顿，有完善的插件机制，支持任何一种编程语言等，它已经成为程序员最受欢迎的编辑器。本文主要介绍从开发一个中型的前端项目到华丽变身为桌面应用的过程，以此来展现 Javascript 跨平台的优点，还有 Electron 魔法般的魅力。下文会阐述项目中遇到的各种问题和我的开发经验。

第二章　作品展示平台业务分析

2.1　作品展示平台应用诞生背景

游戏系与数码系老师与学生设计作品展示的平台，对外展示，对内交流，最终实现一个在校内学生作品之间的沟通平台，与外界进行展示的应用。该网站可以让广州大学华软软件学院游戏系及数码系的学生上传各类比赛的作品及自身优秀作品来进行展示，针对游戏系专业比赛及数码系专业比赛的众多作品有一个作品投稿系统来进行分类、保存、展示等功能，能让学生自身专业知识有个展示平台得以展现，让平台用户得以提升自身；平台目标有游客、浏览应用的所有用户（学生、领导、老师、企业等）；作用范围：主要面向游戏系与数码系全体，涉及学院及网站浏览其他用户。

2.2　用户需求分析

1.游客：　没有登录账号浏览网站的用户进入网站观看作品，搜索内容。（想要了解我校数码系及游戏系学生的大赛作品便于自身提升）

2.注册用户：　注册登录账号浏览网站的用户对作品进行点赞评论，上传作品，观看感兴趣的作品。

1)　非校内人员：浏览作品，可以点赞或评论。（想浏览各类型的作品及游戏作品，寻找志同道合的设计师，互相交流）

2)　校内人员：浏览作品，可以点赞或评论，申请邀请码进行上传作品通过管理员审核。（浏览各类大赛作品，提高自身技术及眼界，对欣赏的作品献上一份支持和讨论作品的话语权；上传自身的作品展示给其他人浏览，也可以从其他作品看到自己作品的优点与缺点。老师可以对作品给出中肯建议，给予设计师肯定或改善的方案）

3)　外部企业：浏览作品，可以点赞或评论，联系设计师。（想了解本校游戏系及数码系的作品，并且寻找企业所需要的人才及作品）

3.栏目管理：对作品进行审核，删除与校正。

4.用户管理：对不当言论用户进行评论删除，屏蔽删除注册用户。

5.权限管理：对管理员权限进行审核，给予和移除。

2.3 业务分析

1.管理员及用户上传游戏系及数码系的各类大赛作品或其本身做的优秀作品到网站进行展示。

2.游客想要了解本校游戏系及数码系的各类大赛作品或者浏览某一类型的作品便可以到网站进行搜索及浏览。

3.用户想要了解本校游戏系及数码系的各类大赛作品或者浏览某一类型的作品即登录网站，并且有点赞评论功能进行各方面的交流，也可通过申请上传自己的作品。如图 2-1 所示。

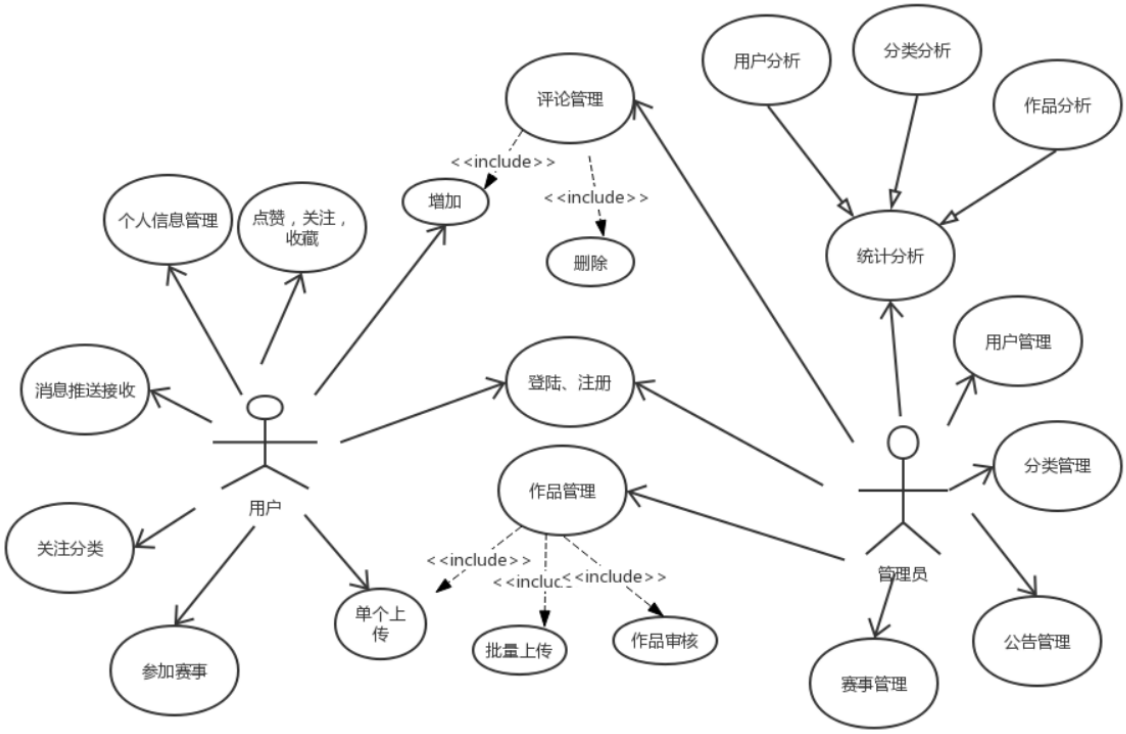

图 2-1 作品展示应用用例图

2.4 作品展示平台应用核心功能介绍

1. 用户注册账号，成功后可上传作品，包含图文介绍、视频等相关资源。

2. 管理员通过后台界面处理用户想要上传作品的通知，并对用户上传的作品进行审核，若无违规则则同意发布到网站进行演示。若有涉嫌违规，则不进行发布。

3. 网站有了相应的作品展示后：

1) 游客可以搜索浏览想要了解的作品及介绍。

2) 已登录用户可以搜索浏览想要了解的作品及介绍，还可以进行点赞、收藏、下载、评论、收到推送等互动功能。在个人界面可以看到个人信息，编辑资料界面进行账号的各项编辑。

第三章　相关技术介绍

3.1　Electron 介绍

3.1.1　什么是 Electron

简单来说，Electron 将“Node + Webkit”连接在一起，UI 使用 Webkit 呈现网页，系统资源操作逻辑使用 Node.js。开发人员可以使用前端语言来开发桌面软件。

3.1.2　Electron 的优势

1) 多平台共享代码

能够在多个平台上使用同一套代码，提高开发效率。虽然不是 100％重用，但 80％也是很好的。尤其对于已经有 Web 产品的开发者来说，甚至可以做到把 Web 产品一键发布成桌面产品。

2) 不需要考虑兼容性

Web 开发常常因为浏览器版本或者厂商，编写了大量“兼容代码”。在 Electron 上开发可以忽略兼容问题，因为它集成了较高版本的 Node.js 和 Chromium，基本上支持所有新的 api。因此，我们可以愉快地使用新技术和新功能。

3) 免费开源

Electron 是完全开源的，无论用 Electron 做任何产品，都不会因为协议的问题而对自己的商业价值造成影响。

3.1.3　Electron 的劣势

1) 内存消耗巨大，难以优化

Electron 要吃掉大量的内存。Chrome 是个内存消耗大户，Electron 是基于 v8 开发的，对于内存消耗这块较大。

2) 体积巨大

因为 Electron 封装了 Node.js + Chromium + V8，导致它的体积非常大，无论应用程序有多小，打包出来都是 50M+，这可能导致分发应用的时候，用户难以接受浪费太多的流量来下载应用。

3) 性能上限低

Web 技术在绘制 UI 的速度很快，但在复杂的 3d 图像渲染上面并不是很好。甚至会造成开发代价远大于传统的开发方式，比如大型游戏，就不适合用 Electron 来做。

3.1.4 Electron 技术小结

使用 Electron 开发产品有它的优势，也有一些短板。我认为如果一个项目有以下特点，就适合用 Electron 开发：

1) 已经是上线的产品，希望尽快铺到桌面端，比如 Slack。

2) Web 产品，比如字典类产品、博客类产品（WordPress、Ghost 等）。

3) 团队擅长 Web 开发，偶遇桌面软件需求。

如果有以下特点，则可能不适合用 Electron 开发：

1) 对网络、性能、存储空间要求严格。

2) 功能单一，受众小的 Web 产品。

3.2 Vue.js 介绍

3.2.1 更少的代码，更强劲的功能

简单来说，Vue.js 是一个高度封装的框架。mvvm 的思想，双向绑定的功能，模块化的管理方式，还有强大的指令系统，最主要的是轻量，效率高，适用于中小型的应用开发。

3.2.2 组件化开发

前端组件化并不是指 js 的模块化，是基于 es6 的模块化系统之上的一种概念。在 Vue.js 里面，每一个 vue 组件单独是一个以后缀名为.vue 的文件，在这个文件内

部写的任何函数，语法，css 跟其他组件既不互相干扰，但是又互相作用，最后类似搭积木的方式，一块一块的组成了这个项目。最好的状态就是，这个项目的部分代码还是可迁移到其他项目的。在团队开发中，即使团队成员水平层次不齐，但是只要 html,css,js 基础很好的话，也能写出比较好的可维护的代码，这就是我选择 Vue 框架的原因。

3.2.3 Vue.js 技术小结

选择 Vue.js 用在这个项目上，并不是盲目选择的，而是对比了一系列其他的的框架，Vue.js 作为一个后起之秀，他吸收了 React.js 和 Angular.js 的优点，摒弃了他们的缺点，下面我列举一下我在此项目中感受到的优点：

1) Vue.js 在逻辑控制上不需要 DOM 操作，只需要专注业务逻辑开发。
2) 虚拟 Dom 让代码更加高效，Vue 的加载速度是前端最快的框架。
3) 通过简单的指令来使页面结构与逻辑数据的解耦。
4) 单文件组建让项目结构更加合理，更易于维护。
5) 生态圈和社区十分完善，vuex vue-router 等。

3.3 Webpack 介绍

3.3.1 前端最强大的打包工具

简单来说，Webpack 是最适合业务项目的打包工具，它不像打包一个 js 库一样需要考虑太多的代码优化。他对于维护老项目可以兼容 AMD、CMD、CommonJS 的模块方式。我们可以根据 Webpack 提供的 Loader，Plugin 机制，编写自己的 Loader 和 Plugin，很方便。

3.3.2 Webpack 结合 Vue.js 常用插件

编译单文件组件的 loader

```
{
        test: /\.(js|vue)$/,
```

```
            loader: 'eslint-loader',
            enforce: 'pre',
            include: [resolve('src'), resolve('test')],
            options: {
                formatter: require('eslint-friendly-formatter')
            }
        }
```

代码压缩插件

```
new webpack.optimize.UglifyJsPlugin({
        compress: {
            warnings: false,
            drop_debugger: true,
            drop_console: true
        },
        sourceMap: true
})
```

公共代码抽取

```
new webpack.optimize.CommonsChunkPlugin({
        name: 'manifest',
        chunks: ['vendor']
})
```

3.3.3 Webpack 技术小结

当合理运用 Weboack 去编译生产环境，打包开发环境的时候，开发效率和体验都会提高很多，项目上线能做到资源较好的分配，减少 http 请求的次数和防止单个文件过大。在这个项目里，我甚至还把单页面的应用改造成了多页面的应用，按照自己的理解，在一个大的应用里面，包含了两个应用，每一个应用都有各自的入口，每

一个应用可以使用不同的 css 框架，也可以共用一套状态管理。最后基于 Vue 的模板，写了一套专门打包 Electron 应用的 Webpack 配置文件。我总结了一下几点

1）合理区分生产环境和开发环境，达到一种资源自动化管理。

2）热加载(hmr)模式能提高开发的效率。

3）可以对资源打上版本号，有效使用浏览器的缓存策略，平滑，精准的升级。

4）scope hoisting 和 tree shaking 能提高 js 加载速度和去除无用的代码块。

5）懒加载和动态加载实现模块的颗粒化管理，提高用户首屏加载速度。

第四章　概要设计

4.1　应用整体设计

4.1.1　应用整体结构的设计

作品展示应用主要由发布作品、赛事、作品渲染、用户、分类、消息推送、数据分析系统七大子系统组成。其中，用户系统就是记录用户账号、行为属性等作用。作品渲染系统主要是作品的显示以及用户对作品的一系列操作，包括评论、点赞、下载、收藏、审核、编辑。赛事系统主要负责发布赛事、参加赛事的功能。发布作品系统负责作品的添加、批量添加、审核作品。分类系统用于提升用户体验。消息系统用于消息的推送，推送分为用户提醒和系统公告。数据分析系统便于让管理员分析系统数据、导出数据，如图 4-1 所示。

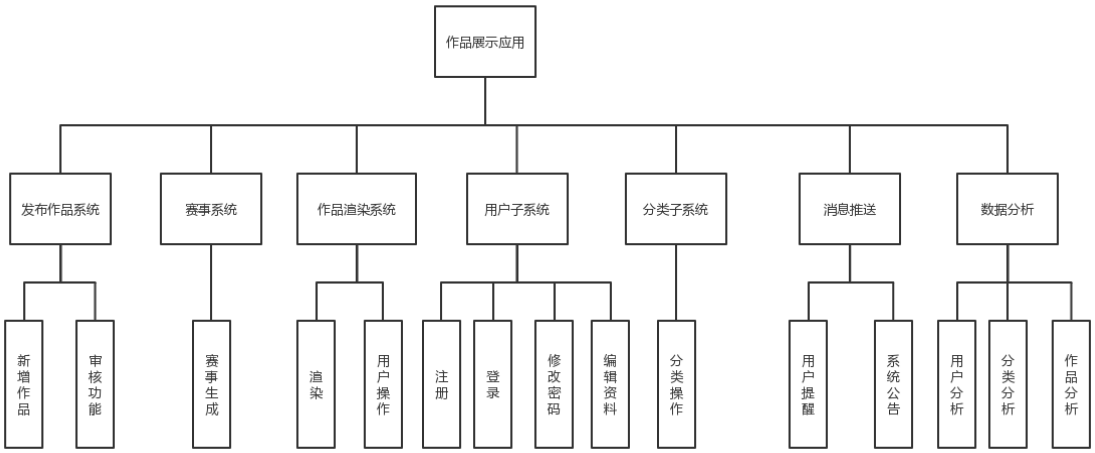

图 4-1 应用整体结构图

4.1.2　应用整体类的设计

EntryManager 类为入口启动类，负责应用的加载功能，控制用户进入的是后台管理页面还是用户展示页面。ProductionManager、ReleaseManager、UserManager、AnalysisManager、ClassifyManager、NotificationManager、GameManager 类是应用的子系统管理类。 ReleaseManager 类进行发布的时候需要依赖 Edit 类和 Upload 类的

功能，包括实现的 VideoUpload 类和 ImgUpload 类。GameManager 类用于生成赛事码并且管理赛事功能。NotificationManager 类负责整个应用所有页面的消息通知。ClassifyManager 类的功能是对作品进行分类，并且对分类进行增删改的功能。ChartUI 类负责图表数据展示那方面的功能，如图 4-2 所示。

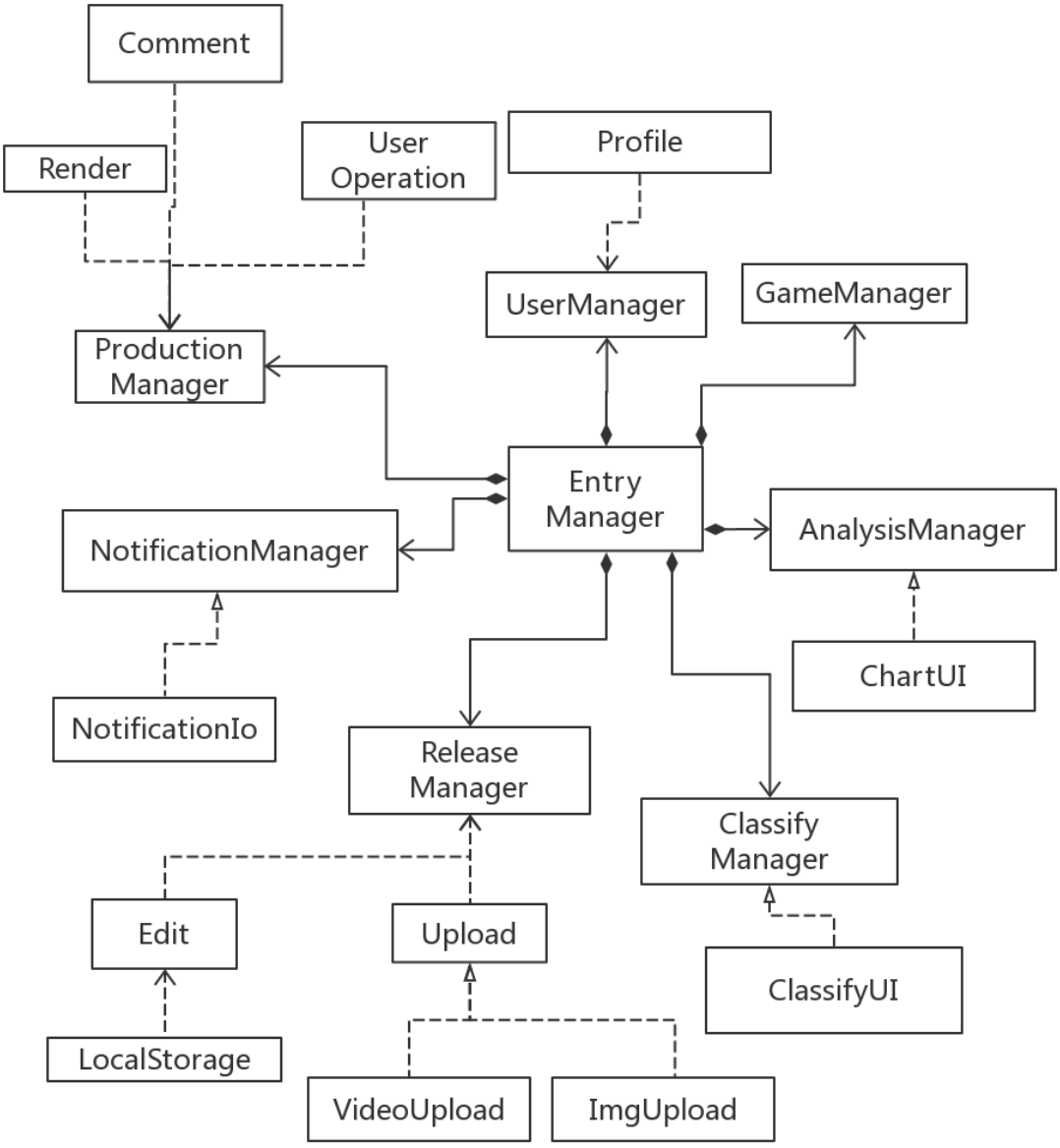

图 4-2 应用整体类图

4.2 关键功能系统设计

4.2.1 发布作品系统设计

发布作品有两种模式可选。分别是个人作品和赛事作品。当用户填写作品的时

候，默认不需要填写赛事码，即个人作品模式，当参赛的时候，填写赛事码，就会进入赛事作品模式。赛事码是管理员后台生成的一个密钥，这个密钥一般通过老师手动传播到参赛学生手里。填写完毕，上传成功后，后台管理员会进行审核作品，通过则显示在首页上面，审核未通过的话需要重新修改再提交，如图 4-3 所示。

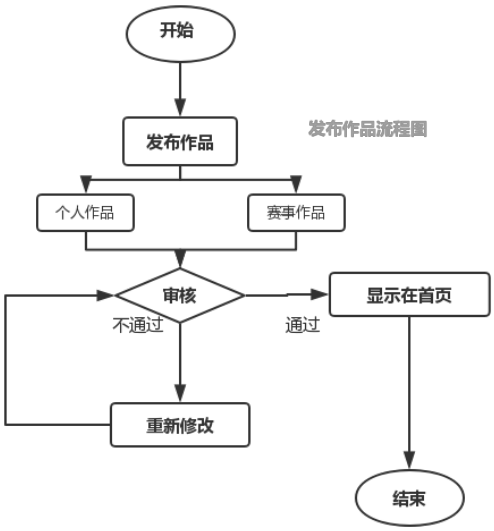

图 4-3 发布作品流程图

ReleaseManager 类依赖 Edit 类和 Upload，Edit 类主要是编辑功能，Upload 用于上传功能，LocalStorage 类用于记录功能。每一个 Edit 类实例都必须依赖 LocalStorage 类，因为编辑器要做记住用户操作的功能。UEditor 是对 Edit 的一种实现，后期可以根据不同的需求来实现不同的 Edit 的接口，通过在 UEditor 类里面判断是否有填写 game_token 来区分上传的是比赛作品学生还是个人作品。Selector 是选择器类，用于填写或者选择输入标签，名字等，因为上传需求也是多种多样的，我这里主要基于 Upload 类实现了目前所需要的图片上传和基于 Socket 的视频上传，如图 4-4 所示。

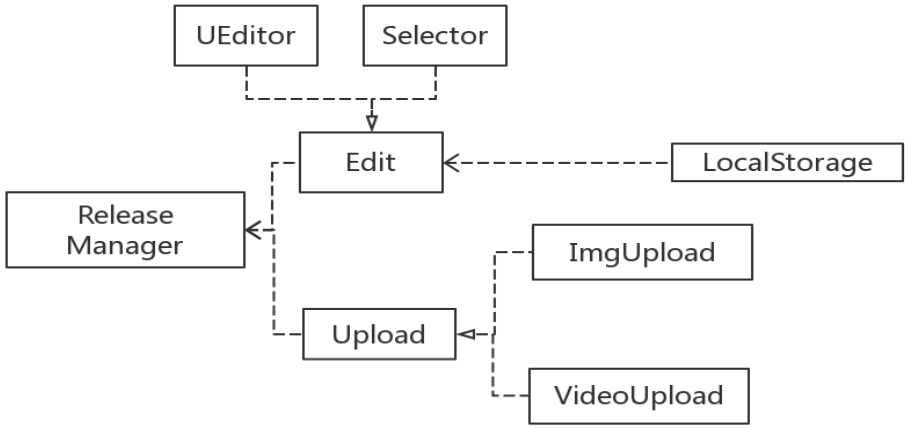

图 4-4 发布作品类关系图

4.2.2 作品渲染系统设计

作品渲染系统依赖 Render 类、Comment 类和 UserOperation 类。Render 类主要负责作品的展示，例如查看作品文字内容、视频、查看高清图片。Comment 类负责这个作品的所有评论以及管理评论用户的信息，包含头像、名字、内容，还有过滤功能。UserOperation 类负责点赞、收藏、审核、编辑的功能，AdminOperation 类和 GeneralOperation 类是 UserOperation 类的实现，主要用于区分用户操作。当用户进入页面时，会判断是否登陆，用户未登陆时 GeneralOperation 类里面的方法并不会完全暴露出来，如图 4-5、4-6 所示。

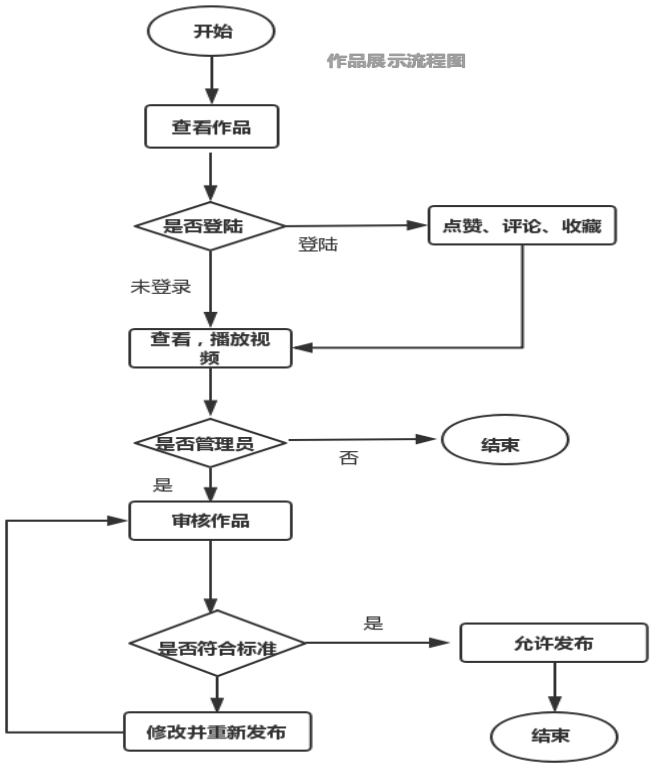

图 4-5 作品渲染子系统流程图

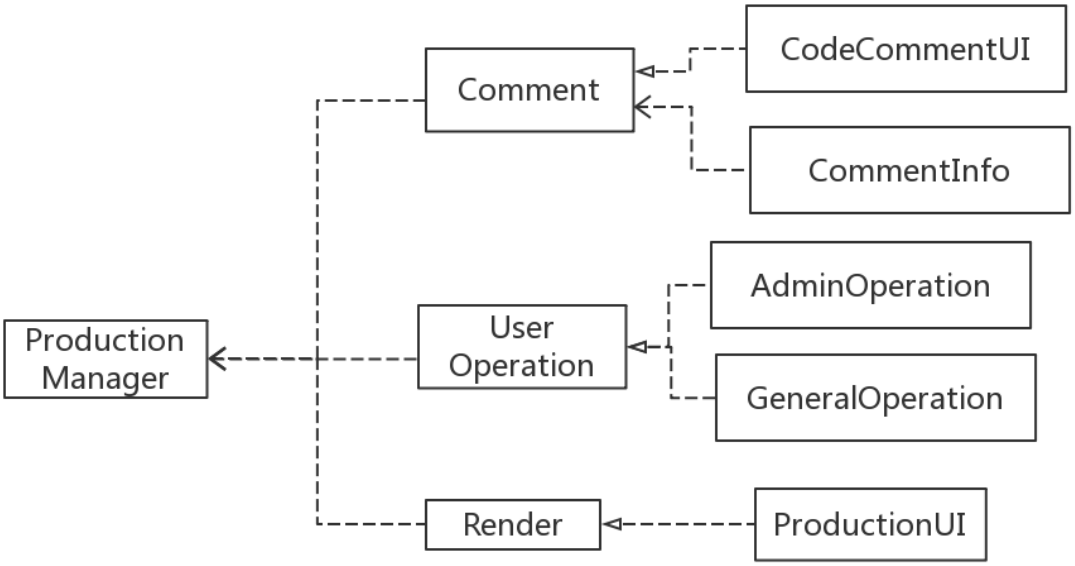

图 4-6 作品渲染子系统类关系图

4.3 多页面管理设计

前面有说到,项目是使用 vue-cli 生成的 Webpack 模板，但是脚手架内置的 Webpack 模板只支持单页面应用,作品展示平台分为前端展示页面和后台管理页面，所以是一个大应用内嵌有两个小应用，所以需要对模板进行改造。

核心思想就是基于于 Webpack 的多入口配置，一个入口就是单页面，多个入口就是多页面，再结合 html-webpack-plugin 生成不同的 html。如图 4-7 所示。

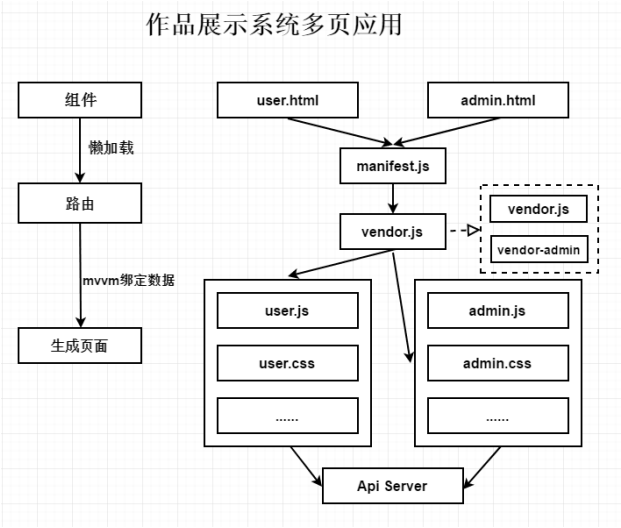

图 4-7 Webpack 多页面架构图

4.4 权限管理设计

前后端都可以做权限控制、控制目的和控制手段都不一样，如果仅从功能上来讲，只在后端做控制就足够了，但在实际项目中，前端权限控制也是非常重要的。所以，项目一开始就必须考虑和搭建的权限校验功能，主要体现为三点：

1. 提高被越权的门槛，防止被篡改的可能。

2. 过滤一些无效的请求，有利于减轻服务器请求的压力，也能提高一定的安全性。

3. 用户体验更好，有权限看的东西才看得到

4.4.1 权限控制核心思想

总的来说，所有的请求发起都触发自前端路由和控件，但是最根本原因还是 http 请求会导致越权，所以我们可以从这三方面入手，对不同的阶段进行不同的操作，我给他们命名为页面级，控件级，请求级的权限控制

1.从页面级来说，用户登陆后，异步生成用户所需要的路由表。

2.从控件级来说，用户只能看到自己有权操作的控件，例如自己的信息是显示修改按钮的，别人只能看到你的信息。

3.从请求级来说，路由和按钮都可能没有配置权限，都可以在 http 请求之前的钩子拦截掉越权操作，但这种校验需要维护路由表与请求地址的对应关系，会显得繁琐和冗余，甚至是资源浪费，但是他的安全性是最高的。

4.4.2 权限控制的具体方法

1.对于页面级的权限控制，前后端需要协商，然后给每个不同权限的用户指定不同的页面路径表 (当然这一份表也可以由后端返回回来)，当用户登录之后，通过后台返回的权限等级，动态根据用户的权限算出其对应有权限的路由，再通过 vue-router 的 addRoutes 方法实现动态添加权限路由及菜单。

2.对于控件级的权限控制，其实就是利用 v-if 来判断当前查看的用户和登陆的用户是否是同一个人，是的话就显示出来。

3.对于请求级的权限控制，我们可以根据维护的路由表和地址来判断是否有权限，无权限则拦截掉当次的请求。如图 4-8 所示。

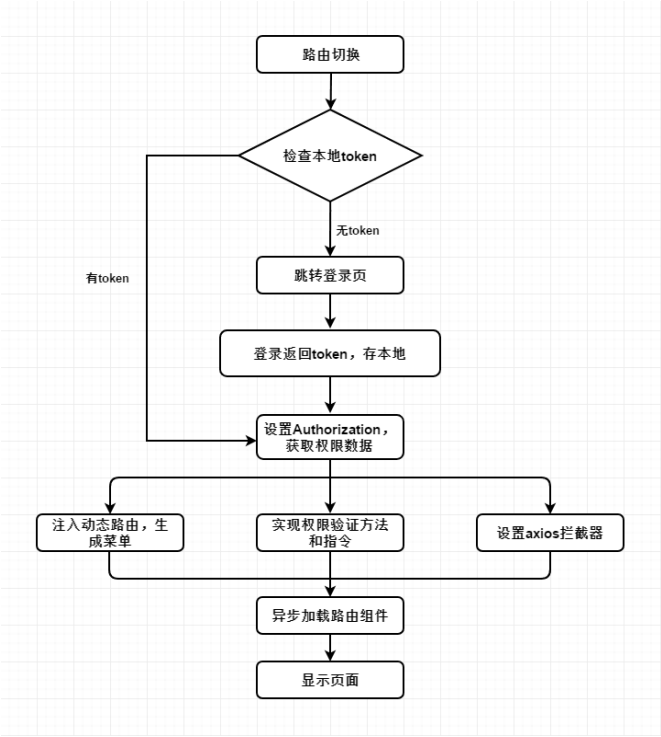

图 4-8 Webpack 多页面架构图

4.5 组件化设计

如果说模块化解决的是如何在封装和复用的基础上管理好各个逻辑代码块的依赖关系，那组件化要解决的是如何封装和复用一个用户界面元素之间的依赖关系，作品展示网站共计组件接近 90 个，后期还会不断拓展，具体划分为，基础组件，业务组件，布局组件。基础组件相当于项目的基石，通常是灵活度最高的组件，他可以跨项目去使用。例如常用的 buttom,slider,dialog，所以本文选择了通过 vue.js 的事件监听机制来进行数据的传递和交互，其次是业务组件，业务组件的灵活度没那么高，通常是一些业务处理的代码，不一定完全依赖 vue.js 的事件机制来进行数据的传递，我在一些组件上选择 vuex 来进行数据传递，例如 header 组件里的的名字和头像。布局组件通常分为两种，一种是类似栅格化布局的组件，另一种是自己根据自己需求写的适用于自己项目的布局组件，他就像一个容器，你只需要把基础组件和业务组件放在此容器内，页面就能呈现出来。如图 4-9 所示。

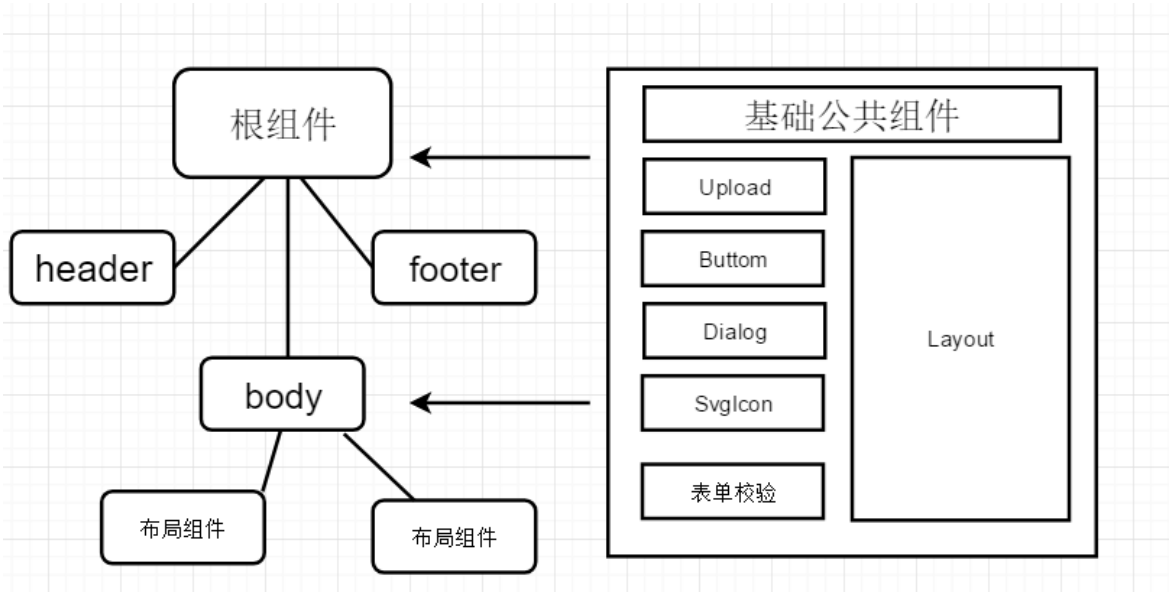

图 4-9 组件抽象树

组件化的方式有以下优点

1. 能够根据需求变换，快速的做出调整，

2. 能够快速准确的定位错误代码进行修改，

3. 团队分配更加合理，不同的技术人员写不同的组件，最后拼凑在一起，就像乐高一样，而且不需要担心人员的流动。

4.6 http 请求设计

我们经常遇到过，每一个组件都写 http 请求，每一个页面都去处理服务器返回的错误提示，甚至难以区分当前 http 请求时针对线上环境还是开发测试环境，一个改动，牵连全部。结合作品展示平台的业务，我封装了一个 request.js 做统一的处理，这个 js 的功能很强大，可配置线上和开发环境的 api 地址，可以自动处理 Token 的权限，可以统一拦截服务器的错误提示。如图 4-10 所示。

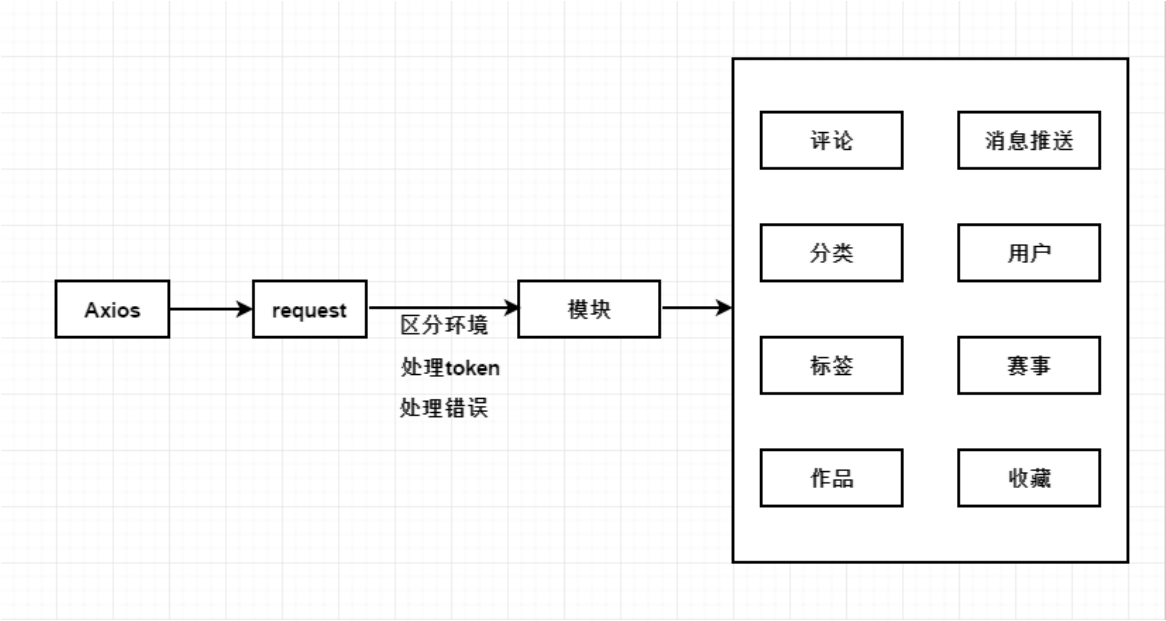

图 4-10 请求 service 层

4.6.1 封装 axios，统一处理所有请求

Axios 是一个可以用于浏览器和 node.js 的基于 Promise 的 HTTP 请求库，面向对象的实例方式，而且它十分轻量。

```
const service = axios.create({
  baseURL, // api 的 base_url
  timeout: 15000
})
export default service
```

在这里我们只需要每次改变 baseURL 来区分线上的地址和开发的地址就可以了。然后把所有的 api 根据不同的模块，放到不同的 js 文件里，引用这个 request.js 来分发不同的请求

```
import request from '@/utils/request'
export function canFollow () {
  return request({
    url: '/kind/canFollow',
    method: 'get'
```

```
    })
}
```

4.6.2 开发环境下跨域问题

众所周知，浏览器的同源策略导致请求非同源的服务器都会有跨域的限制，一种方式是后端设置 CORS，但是这种方式是需要后端的支持，还有一种方式是根据服务器反向代理到指定的服务器，其实跟 Webpack 配合的 weback-dev-server 就是一个微型的开发服务器，Weback-dev-server 提供了 proxy 的配置，在这里我选择了使用 Webapck-dev-server 代理的方式，我们只需要把这个配置独立抽取出来成一个配置文件。

```
const proxyTable = {
  '/api': {
    target: '需要访问的真实地址',
    host: '当前请求地址',
    changeOrigin: true,
    pathRewrite: {
      '/api': ''
    }
  }
}
```

4.6.3 统一处理 Token 和拦截服务器错误

Axios 自带请求和相应的拦截器，因为每一个请求都是需要携带 Token 去后端鉴权的，不可能每一个请求都手动设置请求头，思想就是，每一次请求都从本地拿到 Token,每一次服务器返回的 Token 都存到本地。拦截错误就更简单了，错误处理一律在响应的勾子上做出处理，跟后台协商之后，状态是”fail”的就是错误的请求，错误的信息一律携带在 data 参数里面统一处理 Token。

第五章　详细设计

5.1　发布作品子系统

5.1.1　富文本编辑器

分析：发布作品的页面是一个可以图片上传、视频上传、文件上传功能的富文本组件，简单的文本编辑发布功能,采用 socket 方式传输文本，富文本编辑器是用了 svg+ueditor 开发，选择 ueditor 并不是因为他很好，而是因为基于 vue 的富文本编辑器真的没有一款好用的，要么是轻量但是功能不完善，要么是很丑也不好用，综合考虑我选择了 ueditor，因为 ueditor 编辑器的应用性很广，文档完善，可配置性强，但其也有两个缺点，一个是需要依赖 jquery,另一个是样式很丑。针对这两个缺点，通过对 ueditor 的上层封装，把外观样式用了 svg 的技术代替了它原有的面貌，对于需要引用 jquery 来说，也不算是太难以接受的事情，因为页面是按需加载的，jquery 压缩之后还是可以接受的，他有以下两个实现难点。

1）编辑器数据未完成保存功能

原理是用 localStorage 来存储未填完的数据，一打开这个界面首先会判断是否有一个存储的 storageKey 字段，有的话就去读取上一次未写完的数据继续填写，没有的话就创建字段，然后把编辑器未完成的数据保存，伪代码如下。。

```
autoSaveBody () {
        if (this.isAutoSaved && this.editorData.body) {
          let storage = {}
          Object.assign(storage, this.editorData)
          const pms = JSON.stringify(storage)
          this.isAutoSaved = false
          setTimeout(() => {
            localStorage.setItem(this.storageKey, pms)
            this.isAutoSaved = true
          }, 500)
```

```
        }
    }
```

2) 初始化判断状态

必须要考虑一个情况，就是判断当前是第一次写还是发布之后进行编辑修改，所以外部使用的时候，你只需要操作 innerValue 这个属性，这个属性的值你可以通过后台来获取(后台获取的就是修改状态)，然后编辑器就会呈现什么样的数据,内部的实现方式就是加了一个 init 函数，伪代码如下。。

```
init: function () {
        // 外部有默认值
        if (this.value.body) {
          this.editor.setContent(this.value.body)
        } else {
          // 有本地缓存
          const storage = localStorage.getItem(this.storageKey)
          if (storage) {
            const storageJson = JSON.parse(storage)
            Object.assign(this.editorData, storageJson)
            if (this.editorData.body) {
              this.editor.setContent(this.editorData.body)
            }
          } else {
            // 没有本地缓存
            Object.assign(this.editorData, this.$options.data().editorData)
          }
        }
        this.autoSaveInterval = setInterval(() => {
          this.autoSaveBody()
        }, 5000)
    }
```

5.1.2 Socket.io 流式上传文件

分析：WebSocket 是 HTML5 提供的一种浏览器与服务器间进行全双工通讯的网络技术。

优点： 长连接，双向、低延迟和易于处理错误，没有跨域问题。

缺点：HTML5 的新规范，只有部分浏览器支持。

因为视频和大文件上传用 WebSocket 流处理更好，支持断点续传，上传失败也可以按照上一次失败的进度继续上传，节约带宽，伪代码如下。

```
this.$socket.on('moreData', data => { // {place 第几段, percebntage 百分比}
        let place = data.place * this.uploadSize
        let newFile = this.selectFile.slice(place, place + Math.min(this.uploadSize, this.selectFile.size - place))
        this.fileReader.readAsBinaryString(newFile)
        this.percentage = Math.round(data.percentage)
    })
    this.$socket.on('done', data => {
        this.$refs.video.src = `/api/pro/player?name=${data.name}`
        this.percentage = 100
})
if (this.selectFile) {
        let self = this
        this.fileReader = new FileReader()
        this.fileReader.onload = function (event) {
            self.$socket.emit('upload', {name: self.name, data: event.target.result})
        }
        this.$socket.emit('start', {name: self.name, fileSize: this.selectFile.size})
    } else {
        alert('请选择上传文件')
    }
```

5.1.3 Socket.io 断点续传

分析：断点续传的功能，主要操作逻辑在后端，当前端上传文件的时候出现网络问题导致上传失败，后端会记录当前上传的 chunk 的位置，当下次再次上传的时候，会在上次的断点处继续上传，而不会导致重复上传。后端的处理逻辑主要是用文件的名字和用户的 id 来匹配缓存中的文件，如果对应上了，那就继续上传，伪代码如下。

```
let files = {}
module.exports = {
  async upload ({type, name, data, year, month}, socket) {
    name = `${verifyToken(socket.handshake.query.token, config.authSecret)._id}_${createHash(name)}` // 设置 nam 为用户_id 和文件名字确保唯一性
    files[name].dowloaded += data.length
    files[name].data += data
    await pfs.write(files[name].handle, files[name].data, null, 'Binary')
    log.debug(`${type}已经快要上传完，数据写入 temp 后，将数据流向 upload 后 emit done`)
    let {proDir, proVideoDir} = (year && month) ? mkdirPro(year, month) : mkdirPro() // 视频地址
    let filePath = `${type === 'mp4' ? proVideoDir : proDir}/${Date.now()}.${type}` // 最后保存的地址，固定为 mp4 或 zip
    let readStream = fs.createReadStream(`${config.upload_tmp}/${name}`)
    readStream.on('end', async () => {
      // 上传完毕
      // 关闭文件句柄
      // 删除缓存文件
      // 清除文件暂存信息
      // 返回前加密地址
    })
```

}

5.2 赛事子系统

分析：在赛事系统中，赛事类由名称 name，图片 cover_pic，开始 begin_time 和结束日期 end_time，赛事的描述 description 组成。release 和 delete 方法用于发布赛事和删除赛事，generate 方法可以生成赛事码。前端通过调用接口 POST /game/add 成功就可以创建赛事了，如图 5-1 所示，赛事功能详细接口见表 5-2。

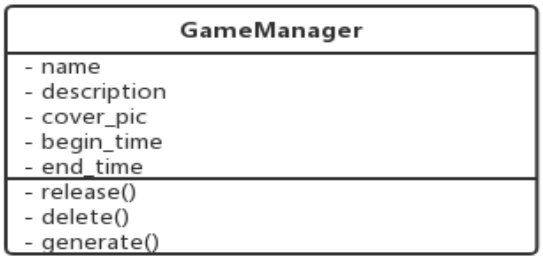

图 5-1 赛事类图

5-2 赛事功能接口表

功能:	url:	method:	params:
根据赛事 id 查找赛事	/game/get/	get	gameId
根据赛事 id 生成一个赛事码	/game/gameToken/	get	gameId
根据赛事 id 获取赛事作品	/game/pros/	get	gameId
首页赛事推荐	/game/hot/	get	无

5.3 作品渲染子系统

5.3.1 内容渲染

分析：Render 类的主要负责的是渲染作品上传系统里面富文本里面的内容，然后加上一个视频播放器。文本内容的渲染是 getProduct 方法来实现的，前端通过调用接口 GET /pro/get/:proId?token=xxx，渲染的核心是通过指令 v-html 直接渲染后台返回的数据，返回的数据里面，图片数量是未知的，所以特地封装了一个 zoomImg 的方法可以来查看高清图片，图片放大查看的功能的核心就是事件委托机制，给父级绑定一个事件，而不是给所有图片绑定事件。最后通过调用 playVideo 方法来控制播放器调用接口 GET /pro/player/:name，这里的视频播放器是使用了第三方 video.js，支持全屏，快进，倍速的操作，如图 5-3 所示。

图 5-3 内容渲染类图

5.3.2 点赞、收藏

分析：点赞和收藏功能看起来其实很像，但是实现是不同的，因为收藏的功能比点赞多很多，点赞是依赖作品表的，直接在 pro 表里定义一个 stars 属性，存储用户的 id,前端通过接口 patch /pro/star/:proId，携带参数作品的 id，需要注意的是每次点赞成功之后再去后台重新获取最新的点赞数，而收藏是在 collec 表里，然后分别定义 names 和 pros，然后通过外键的方式关联起来，前端通过接口 POST /collect/addPro/:collectId，并且携带参数收藏夹 id 和作品 id，成功之后会自动隐藏收藏夹，点赞、收藏功能详细接口见表 5-4。

5-4 点赞、收藏功能接口表

功能:	url:	method:	params:
根据用户 id 获取收藏夹	/collect/userCollect/	get	userId
新增收藏夹	/collect/add/	post	name cover_pic description
给收藏夹新增一个作品	/collect/addPro/	patch	collectId proId
取消/关注指定的收藏夹	/collect/follow/	patch	collectId
给收藏夹取消一个作品	/collect/delPro/	patch	collectId proId
根据用户 id 获取用户点赞过的作品	/pro/starPros/	get	userId
给作品点赞	/pro/star/	patch	proId

5.3.3 评论

分析：Comment 类就四个属性 pro_id、user_id、content、create_time，评论的时候，首先把 pro_id 和 user_id 发送给后端，然后后端在两个表之间建立关联之后再把 content 存到数据库，下次查询的时候直接把评论渲染在页面上，前端通过调取接口 POST /comment/add/:proId，并且携带上参数作品 id 和评论内容，如图 5-5 所示，评论功能详细接口表见 5-6。

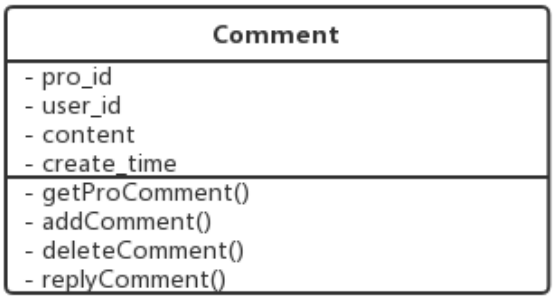

图 5-5 评论类图

5-6 评论功能接口表

功能:	url:	method:	params:
根据作品 id 获取作品评论	/comment/proComment/	get	proId
新增评论	/comment/add/	post	content proId
分页获取评论	/comment/all/	get	condition pageNum pageSize
根据评论 id 删除评论	/comment/delete/	delete	commentId

5.3.4 审核

分析：当用户发表作品之后，作品表默认的 status 为 0 表示待审核，管理员才有权限操作这个 status 的值，管理员通过审核可以修改作品 status 为 1 未通过需要修改或者修改作品 status 为 2 审核成功，前段通过调取接口 PATCH /pro/audit/:proId，如果是通过，只需要携带 status 和作品的 id，如果未通过则需另上传一个 fail_reason 参数。

5.4 用户子系统

分析：用户子系统包括：注册、登录、修改密码、编辑资料、用户类五个部分，注册类主要由账户、密码、邮箱组成。个人编辑类由头像、系别、专业、名字、性别组成，注册、登陆、修改密码、编辑资料最终组成用户类，它们也能单独运行，如图 5-7 所示，下面详细介绍每个类的用法。

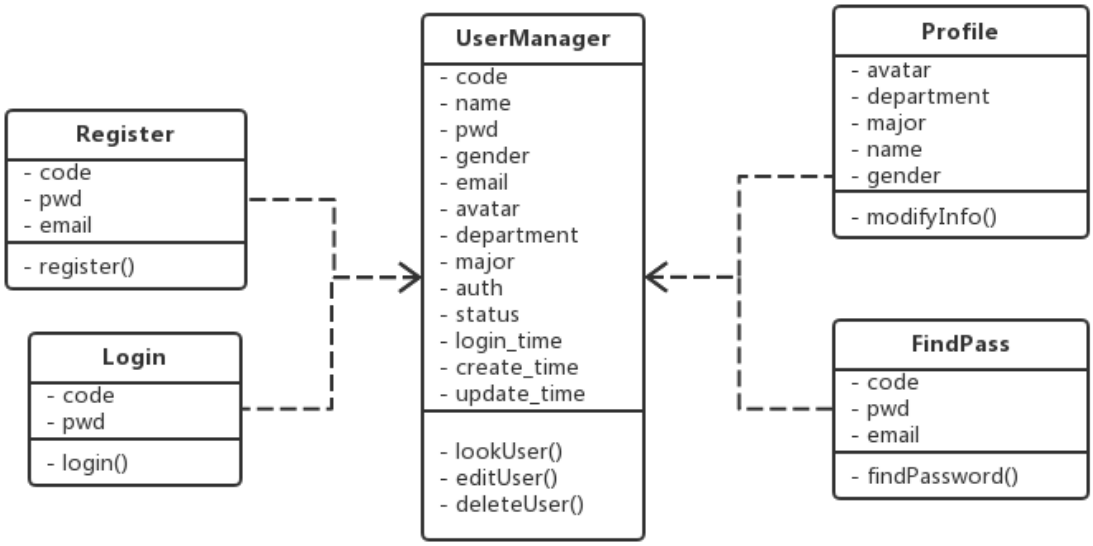

图 5-7 用户子系统关系类图

5.4.1 注册

分析：用户填写账号和密码，还有邮箱，账号是 4 位的工号或者 10 位的学号，学号必须唯一，并且需要写上常用的邮箱，以备后续修改密码时使用，前端和后端都需要校验输入格式是否正确，前端校验完毕后，发送请求 POST /user/register，如果后端也校验成功则注册成功，前后端任意一端校验失败，前端都会提示，注册失败。管理员也可以进入后台页面的用户管理，下载制定的模板，根据下载模板文件填入每一个用户的信息，通过接口 POST /user/batchRegister 把模板文件上传到服务器，这一过程就实现了批量导入用户的功能了。

5.4.2 修改密码

分析：用户假如需要修改密码，可以根据注册时候填写的邮箱进行找回，通过接口 POST /user/send 邮箱会收到一封修改密码的提示邮件，根据这封邮件指示的操作来进行密码的修改。

5.4.3 个人主页展示

分析：用户个人主页页面分为三个子路由，通过给 vue-router 传递不同的参数来分辨不同的页面，收藏夹页面参数是 collectList，作品页面参数是 userPros，点赞页面参数是 star，当进入这个路由的时候，再获取到参数，然后根据不同的参数进行不同的 http 请求，最后显示不同的页面。

5.5 分类管理子系统

分析：分类管理类主要负责用户关注的分类，属性有标题、图标、父节点分类、还有粉丝。管理员有创建、删除、修改分类的权限。例如管理员可以通过接口 POST /kind/add 创建分类，PATCH /kind/update/:kindID 修改分类，DELETE /kind/delete/:kindId 删除分类，如图 5-8 所示。

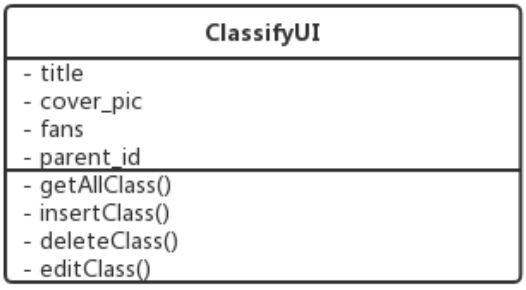

图 5-8 分类管理类图

分类树是基于 element-ui 的 el-tree 组件，有三个功能，新增，删除，和修改，这

三个按钮必须使用 JSX 语法来写，在 vue 里面使用 JSX 语法，尤其事件的时候是需要考虑 this 指向问题的，可以根据 bind 函数或者 es6 的箭头函数来配合，否则按钮不生效，分类树后边的操作新增和修改是根据一个属性 dialogType 来控制的，dialogType 为 1 的时候是添加，dialogType 为 2 的时候是修改，当添加子树的时候，需要记录当前节点的父节点 id，后端根据这个 id 的位置来插入数据库内，所以我在点击事件触发的时候，就记录了所需要的 id，如图 5-9 所示。

图 5-9 分类树

5.6 消息推送子系统

分析：消息推送类主要负责系统消息推送和用户消息推送，主要有类型、推送用户的 id、作品 id、推送的消息内容、状态等。通过 type 属性来区分作品被点赞了，作品被评论了，作品被收藏了，作品审核通过了等不同类型的消息。消息推送的技术是使用 WebSocket 来做的，通过监听 message 事件接收到后台传过来的数据，这里的 WebSocket 也使用了 token 去验证，前端使用了 HTML5 的 Notification 类，Notification 类可以控制弹框的样式，以及显示的时间等，Notification 类的 permission 有三个属性，granted 表示用户允许通知，denied 表示用户拒绝接受，default 表示用

户目前考虑中。配合 Electron，能弹类似系统样式的提示框，体验更好，如图 5-10 所示，具体代码实现如下。

```
new Notification('作品展示应用', {
    body: '有新的消息推送了，快去查看吧',
    tag: 'notific', // 代表通知的一个识别标签，相同 tag 时只会打开同一个通知窗口。
    icon: 'favicon.ico', // 要在通知中显示的图标的 URL。
    requireInteraction: true // 通知保持有效不自动关闭，默认为 false。
})。
```

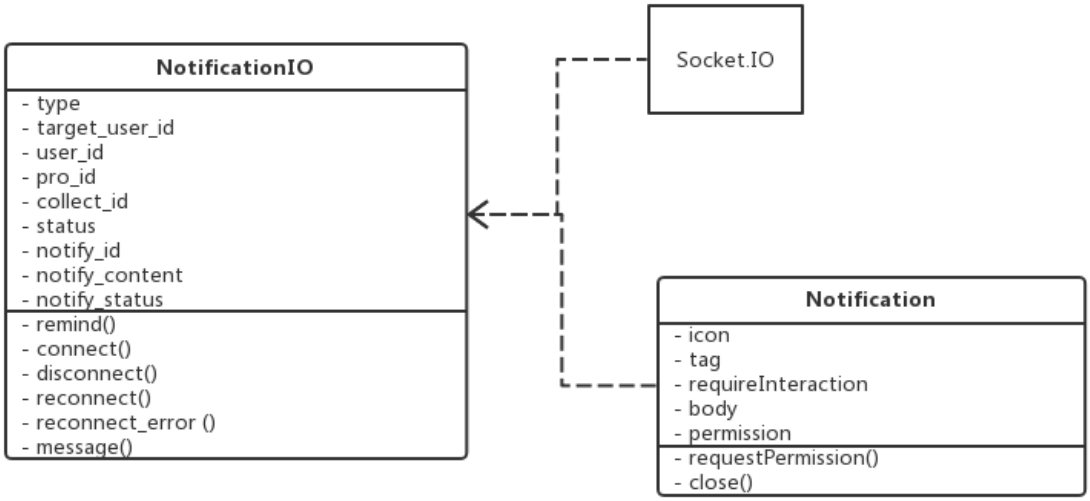

图 5-10 消息推送类关系图

5.7 数据分析子系统

分析：数据分析系统支持分类分析，用户分析，作品分析，以后还可以增加其他的条件分析。为了方便管理员可以准确的获取到有用的数据，我借助了百度开源的 Echarts 把后台数据库的数据可视化在了页面上，不过 Echarts 库太大，我采用了按需加载的方式大大减小了库的体积。ChartUI 的 myChart 属性就是一个 Echarts 实例，所以我通过调用 userAnalysis 方法，去后台请求数据然后结合 myChart 私有的方法来操作。例如，根据 magicType 为 line 或者 bar，数据就会自动按照柱状图，折线图，

数据视图的方式呈现在页面上，并且数据分析系统里的每一种分析方式都支持日期条件搜索等，如图 5-11 所示。

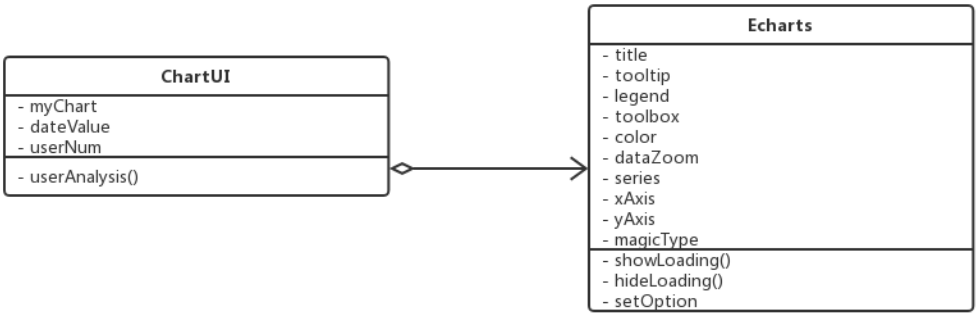

图 5-11 数据分析类关系图

第六章　打包发布

6.1　Web 打包发布

其实 Web 打包很简单，全都得益于一开始 Webpack 就已经划分了生产环境和开发环境，只需要执行 npm run build 就能在项目目录下生成一个 dist 的文件夹，打包出来的文件都有一个独立的 hash 码，只需要把这个文件夹的内容放到服务器上面，这就是前后端分离的好处。假如资源文件要放 cdn 上面，我们只需要进入到 config/index 在 build 里面配置 assetsPublicPath,路径改成 cdn 的地址就可以了。当然我们可以使用 npm run build －report 来查看和分析我们组件的大小。见图 6-1

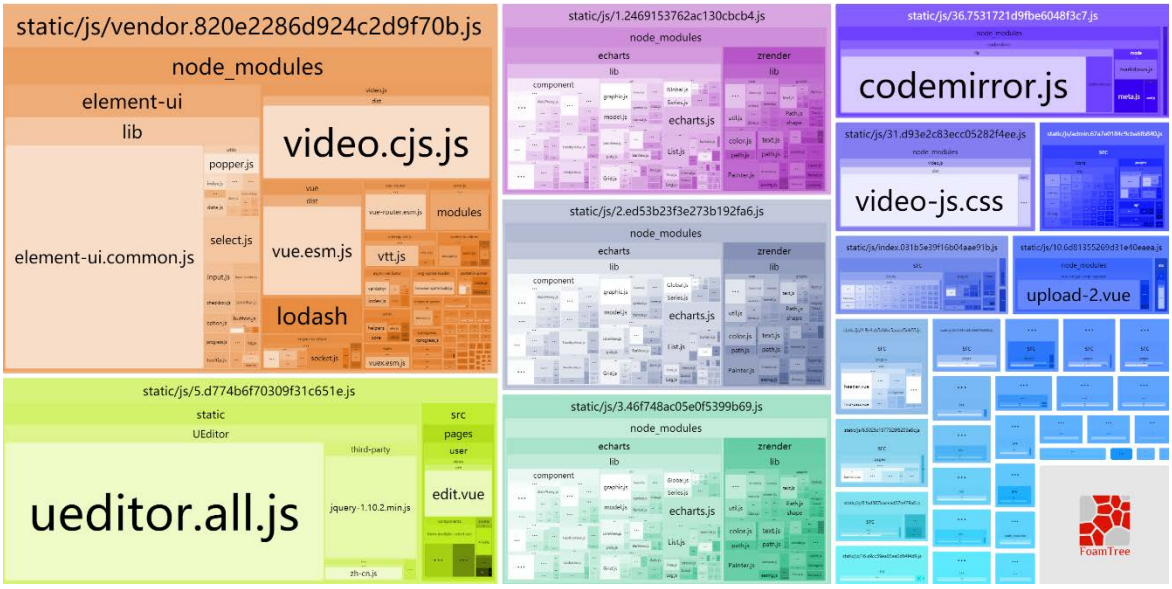

图 6-1 打包优化分析图

顺便说一下，这种打包方式采取了一定的措施来优化前端性能，也是业内比较通用的一种生产环节的打包方式。

1. 合理使用浏览器的本地缓存 - 减少客户端对服务器的请求
2. 根据内容更改生成哈希代码 - 精确的缓存控制。
3. 文件根据内容更改生成唯一的哈希代码 - 不会造成资源浪费。
4. 上线不会造成覆盖旧版本源码，一旦产生 bug 可以回滚 - 安全有效的升级。

6.2 桌面端打包发布

对于 Web 应用来说，我们只需要把链接传播出去就行了。但是使用 Electron 开发的是桌面软件，所以我们不可能指望用户像我们一样安装 Node.js、安装 Electron 包、执行命令打开软件。我们必须把它打包成适合传播的单个文件，然后再想办法去推广分发它。

实际上，我们可以把 Electron 看成一个特殊的浏览器，生成的软件包 = 浏览器 + 全部 Web 文件 + 全部 Node.js 文件。因为开发者的电脑配置各有不同，并且使用 Electron 的多半不是浏览器开发者（要用 C/C++），所以既不具备本地编译浏览器的能力，时间效率上也不允许。

于是 Electron-packager 的实际做法是下载所有目标环境下预制好的 Electron 软件包，然后把里面的业务逻辑部分，也即 Web 文件和 Node.js 文件替换成我们写的代码（即 <sourcedir> 里的代码）。新的软件包会保存在 <appname> 目录 。
对于当前项目，是基于 Web 版本加壳子的思想。只需要 npm run electron 就能在根目录下面生成一个 electron-dist 的目录，我们把这个目录放到 Electron-packager 预制的软件包里面就可以了，我们还可以用 Electron-packager 给软件打上版本号，软件信息，给软件源码进行加密。

第七章　应用测试与运行

7.1　测试环境

测试平台为 64 位 Window10 系统

7.2　测试主要内容

7.2.1　测试内容

1. 页面测试，测试各个页面是否能正常打开，正常跳转，能否正常获取数据。
2. 上传大文件是否有问题
3. 是否能根据不同权限进入不同的页面
4. 图片地址加载是否有问题，能否根据不同环境匹配不同地址
5. 能匹配不同的设备尺寸

7.2.2　结果

7-1 测试结果表

测试	测试结果
上传大文件和视频	正常
图片地址是否正常	正常
不同的设备能否完整看到信息	部分页面不支持
页面跳转，角色转换	正常
能否正常播放视频，下载作品	正常
能否注册	正常
能否登陆	正常
能否修改密码，用户能否收到邮件	正常
能否收藏，点赞，评论	正常

管理员是否能审核作品	正常
管理员能否创建赛事	正常
管理员能否推送消息	正常
用户能否正常收到消息推送	正常
用户能否参加赛事	正常
用户能否发布作品	正常
管理员审核作品	正常
管理员能否创建修改删除分类	正常
用户能否关注分类	正常
用户修改个人信息	正常
用户修改收藏夹信息	正常

7.3 应用功能展示

7.3.1 应用首界面

当用户第一次进入应用时，页面分为导航，导航有搜索功能和菜单按钮和头像显示，导航下边为两栏布局，左边是作品列表有自适应效果，右边是赛事推荐和热门标签栏，如图 7-1 所示。

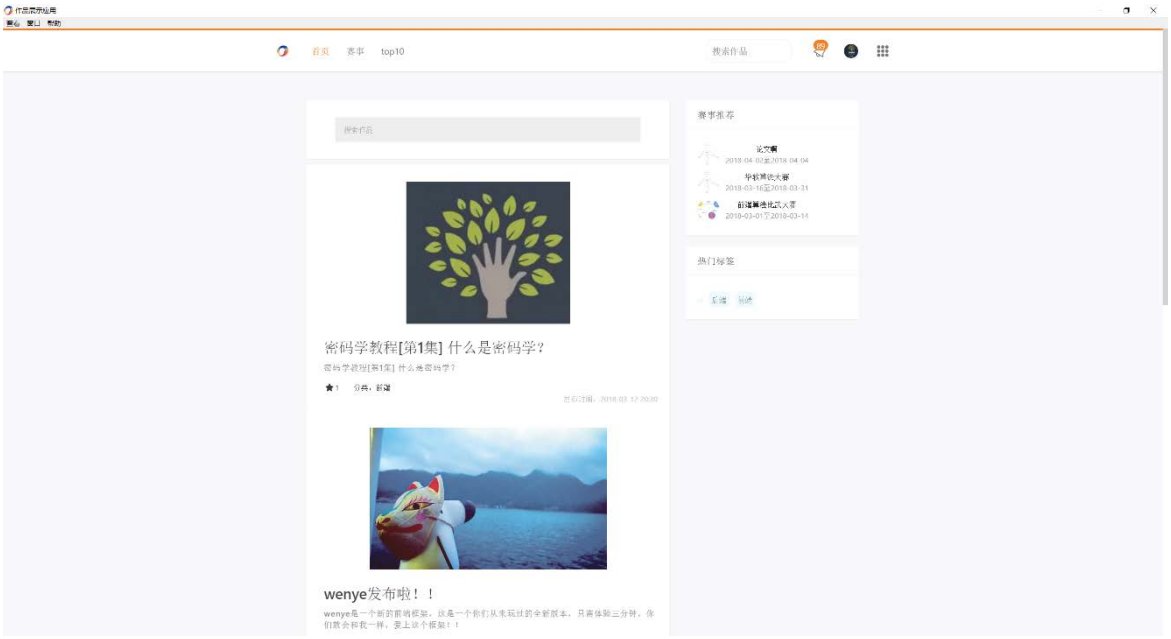

图 7-1 应用首界面

管理员页面就比较单调，左右布局，左边是菜单栏，右边是展示和操作区，如图 7-2 所示。

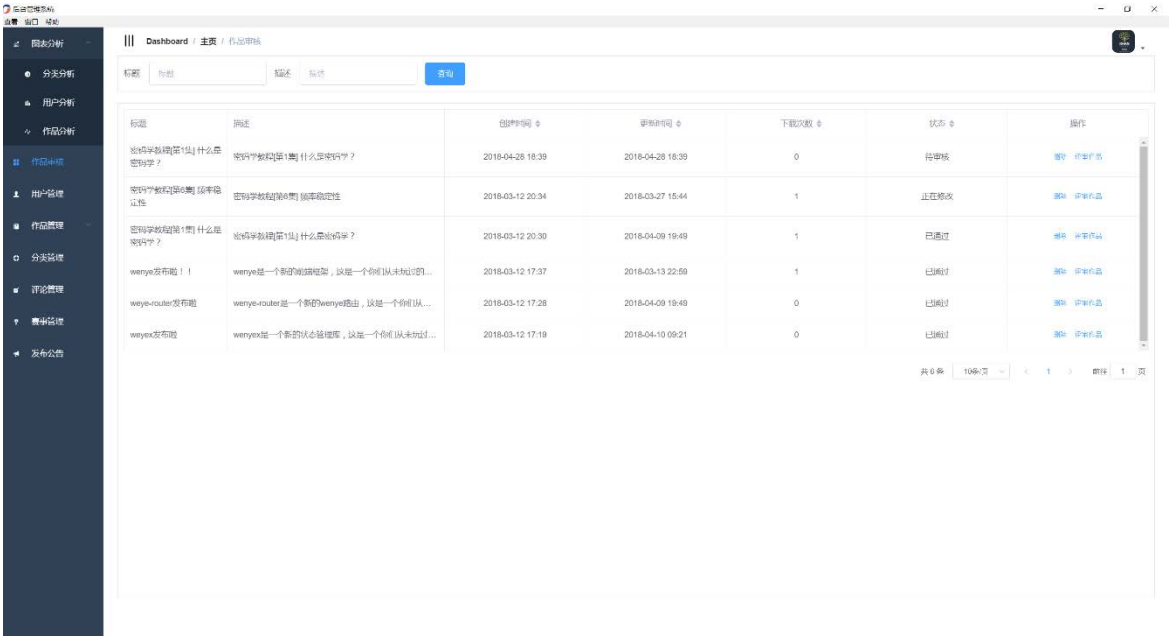

图 7-2 管理员首界面

7.3.2 作品展示界面

作品展示界面是上下布局，上边是作品的简单介绍，还有一些用户操作按钮，下

边是视频播放和内容展示和评论区，如图 7-3 所示。

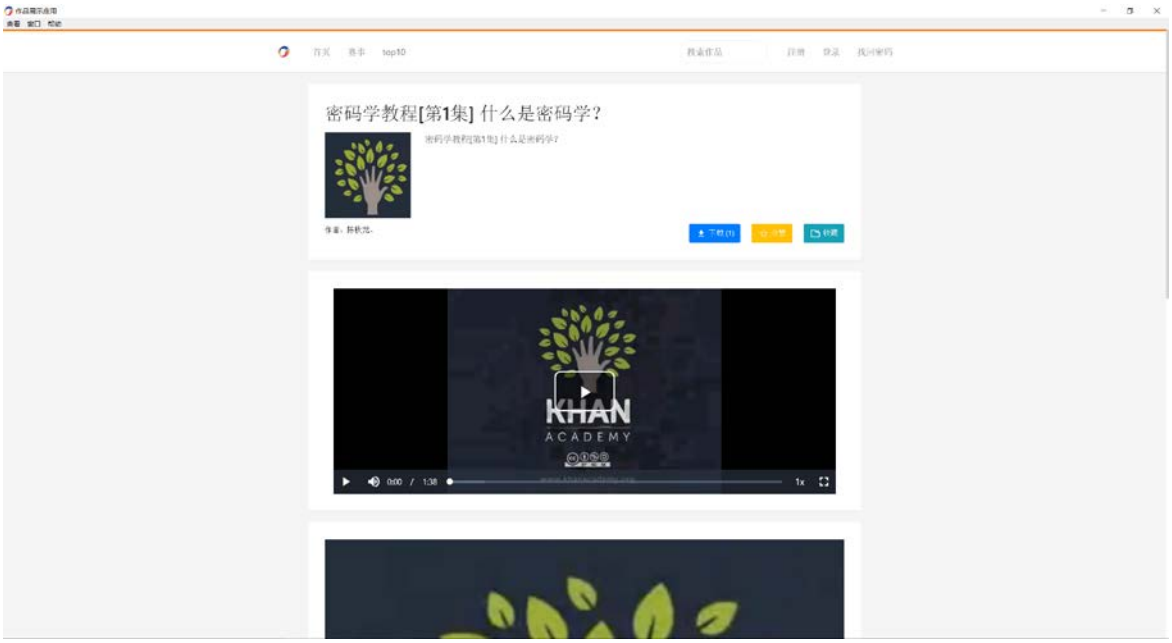

图 7-3 作品展示界面

7.3.3 作品评论区界面

评论界面由输入框和评论显示组合合成，评论显示的头像是可点击的，点击之后会跳转到用户的个人界面，如图 7-4 所示。

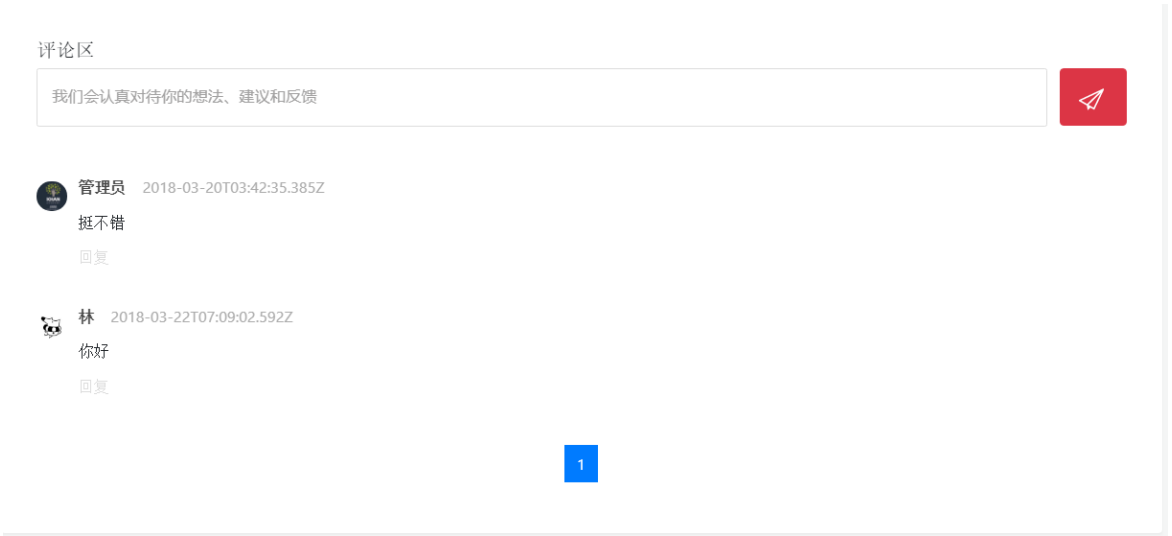

图 7-4 作品评论界面

7.3.4 作品收藏夹界面

收藏夹功能是弹窗效果，输入文字右边的添加按钮会被激活，已纳入收藏夹的作品，收藏夹会有绿色的标志，如图 7-5 所示。

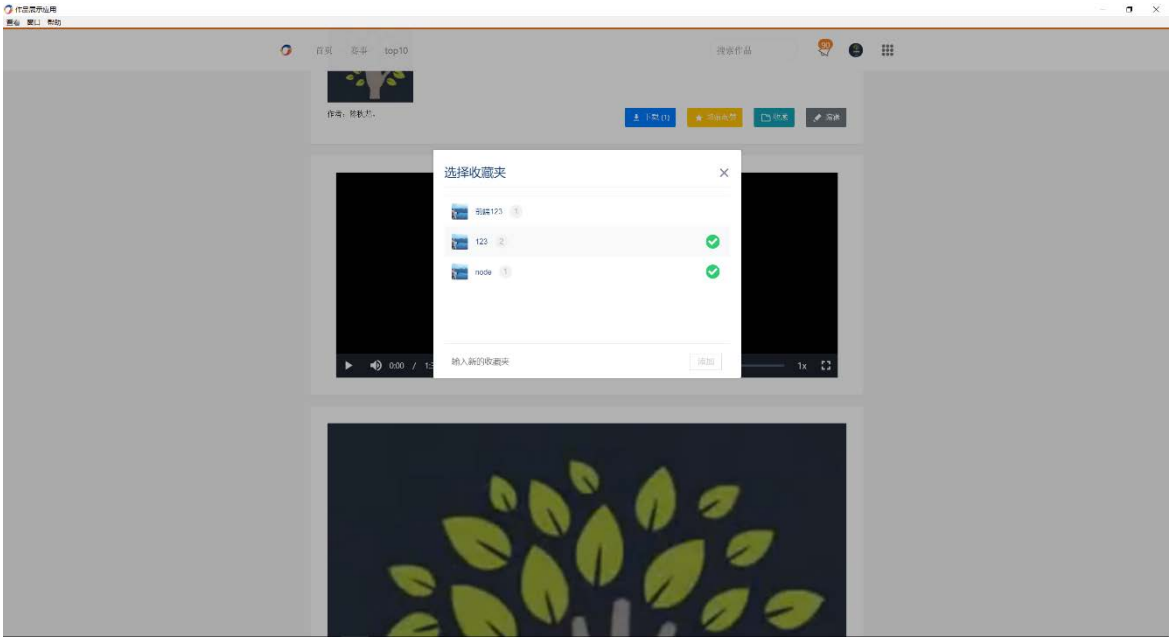

图 7-5 作品收藏夹界面

7.3.5 赛事作品界面

赛事作品界面是由标题和作品数量再加上一个渲染列表，渲染列表也自带自适应功能，如图 7-6 所示。

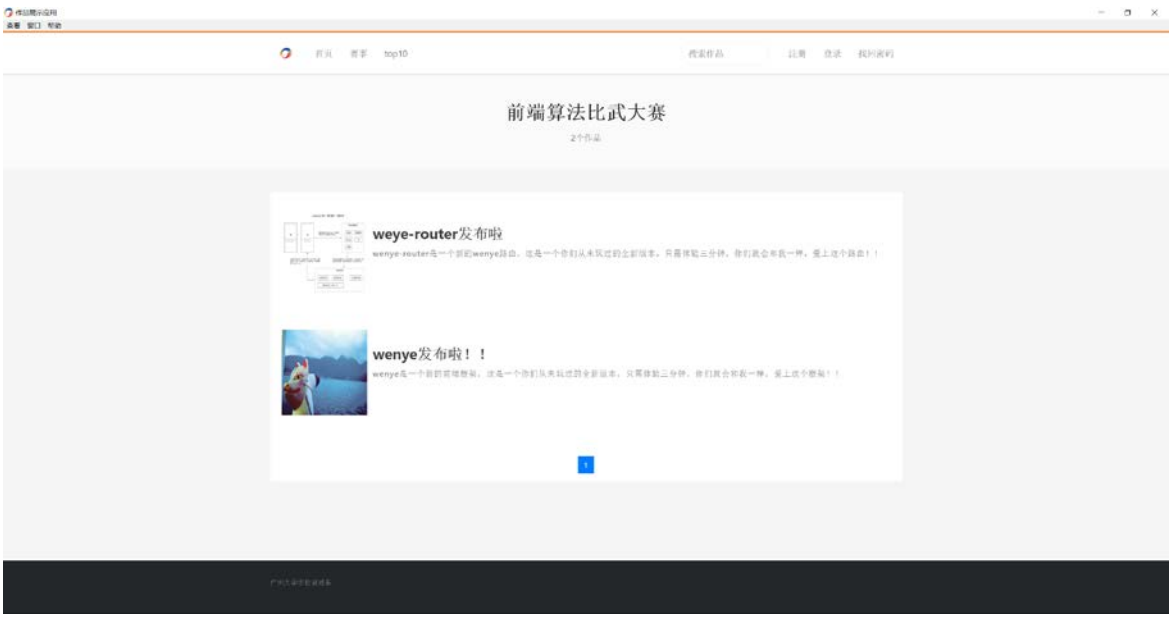

图 7-6 赛事作品界面

7.3.6 消息推送界面

消息推送先是系统弹框提醒用户，用户再到系统通知界面查看所有的消息，点击黄色的图标，会有一个模态框显示详情，如图 7-7 所示。

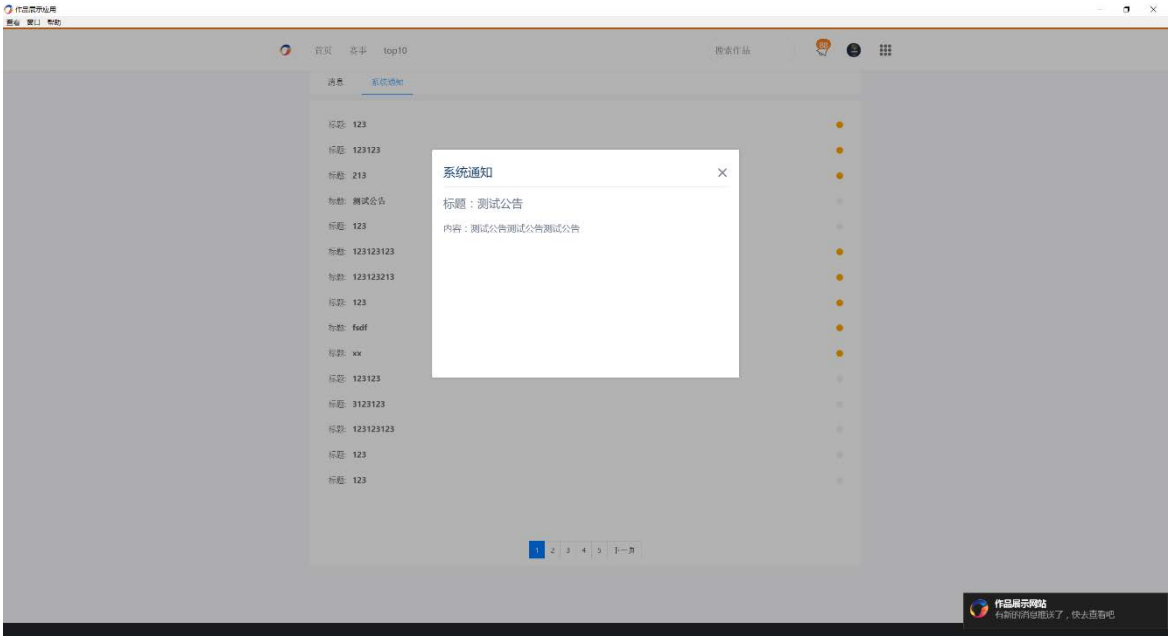

图 7-7 消息推送界面

7.3.7 个人界面

个人界面包含头像、信息、收藏夹、作品、点赞组成，设置按钮在非本人登陆的情况下是不会显示出来的，如图 7-8 所示。

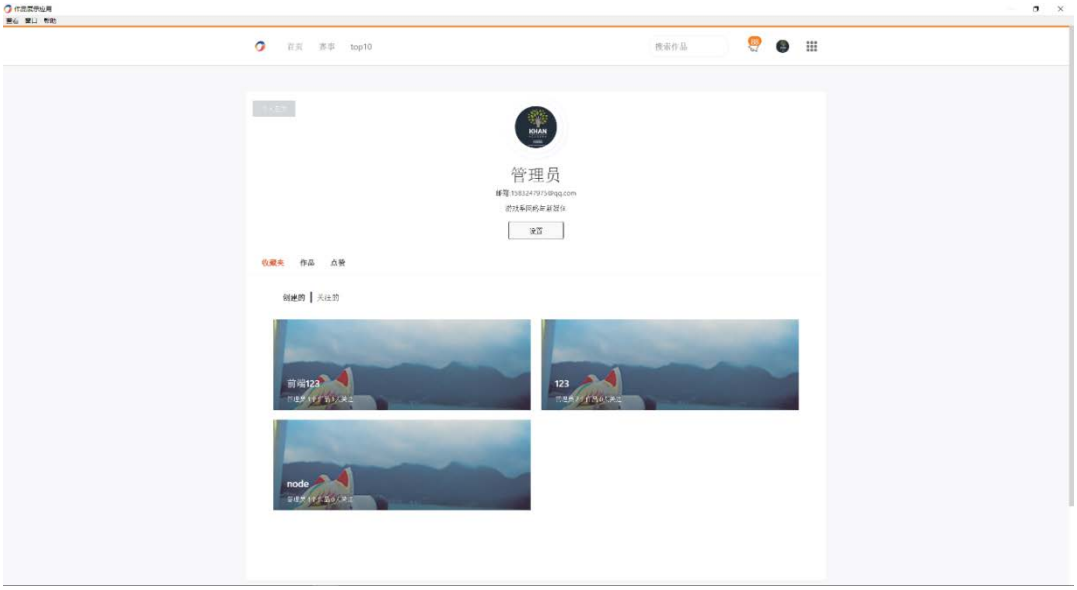

图 7-8 个人界面

7.3.8 个人信息修改界面

个人信息修改可以修改用户的头像，系别、名字、性别、专业、邮箱，如图 7-9 所示。

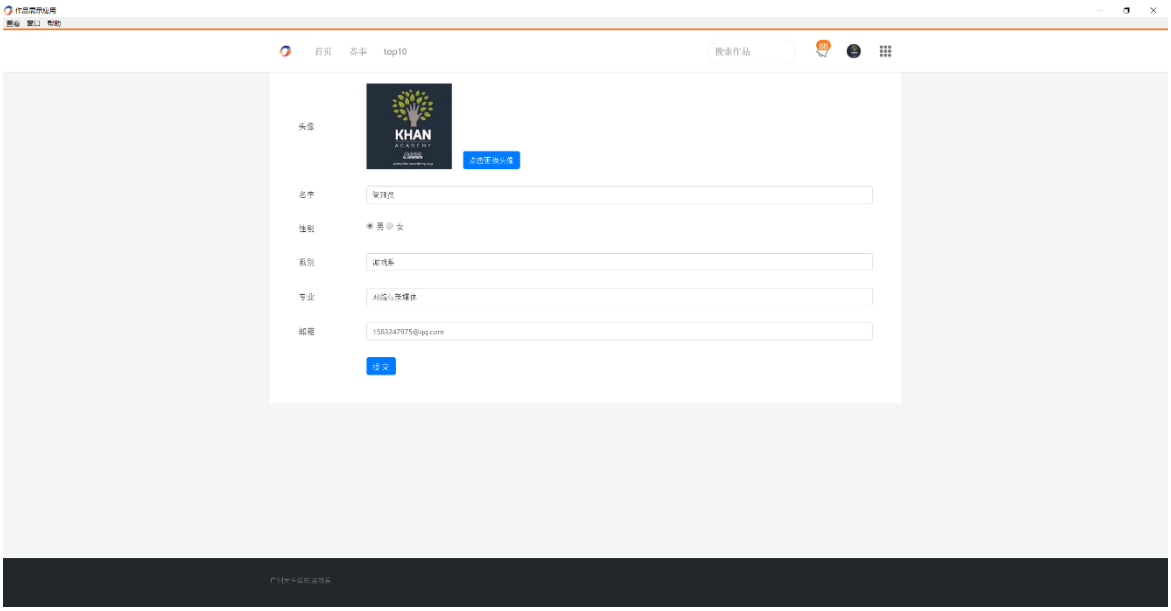

图 7-9 个人信息修改界面

7.3.9 批量上传作品界面

批量上传作品，填写完毕所有数据后，会自动的校验输入信息是否合法，否则不能上传，上传的时候作品右侧会出现进度条，如图 7-10 所示。

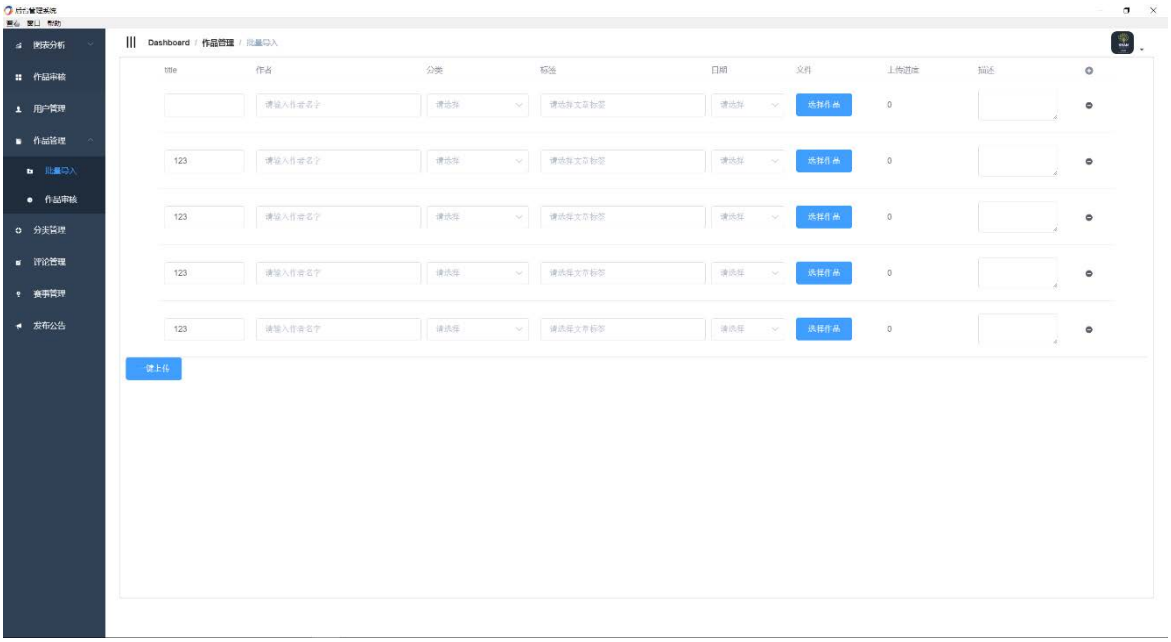

图 7-10 批量上传作品界面

7.3.10 赛事管理界面

赛事管理界面也是由表单组成，管理员须填写信息完毕后才能发布赛事，生成赛事码，如图 7-11 所示。

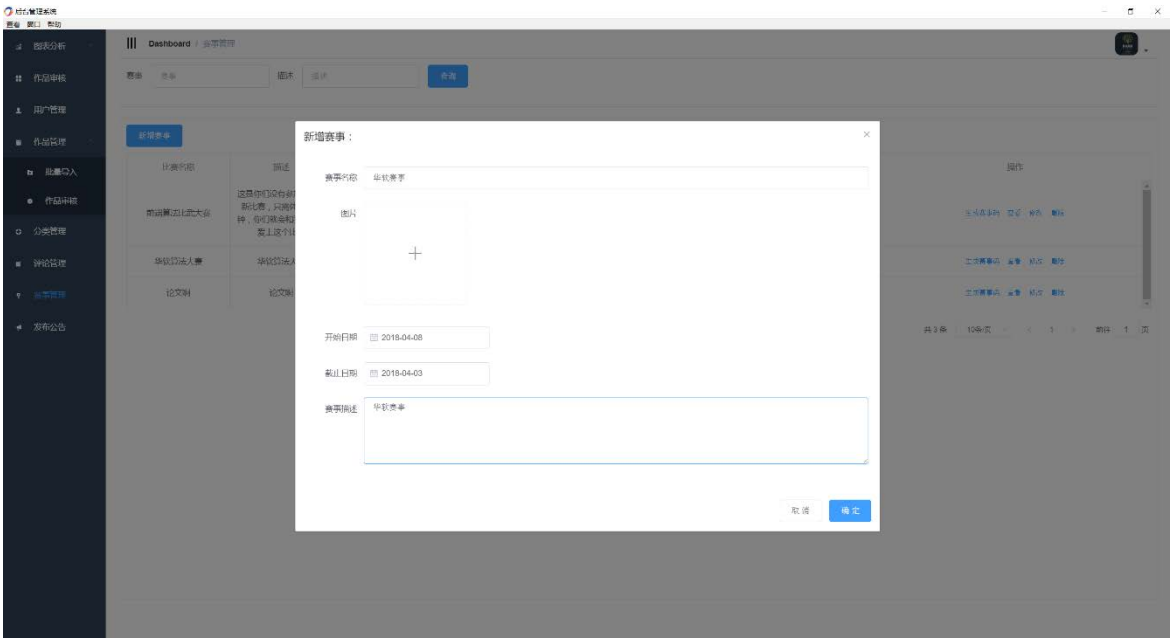

图 7-11 赛事管理界面

7.3.11 分类管理界面

赛事管理界面，管理员可以点击新建分类按钮新增顶级分类，也可以点击分类树右边的按钮进行当前层级分类的增删改查，如图 7-12 所示。

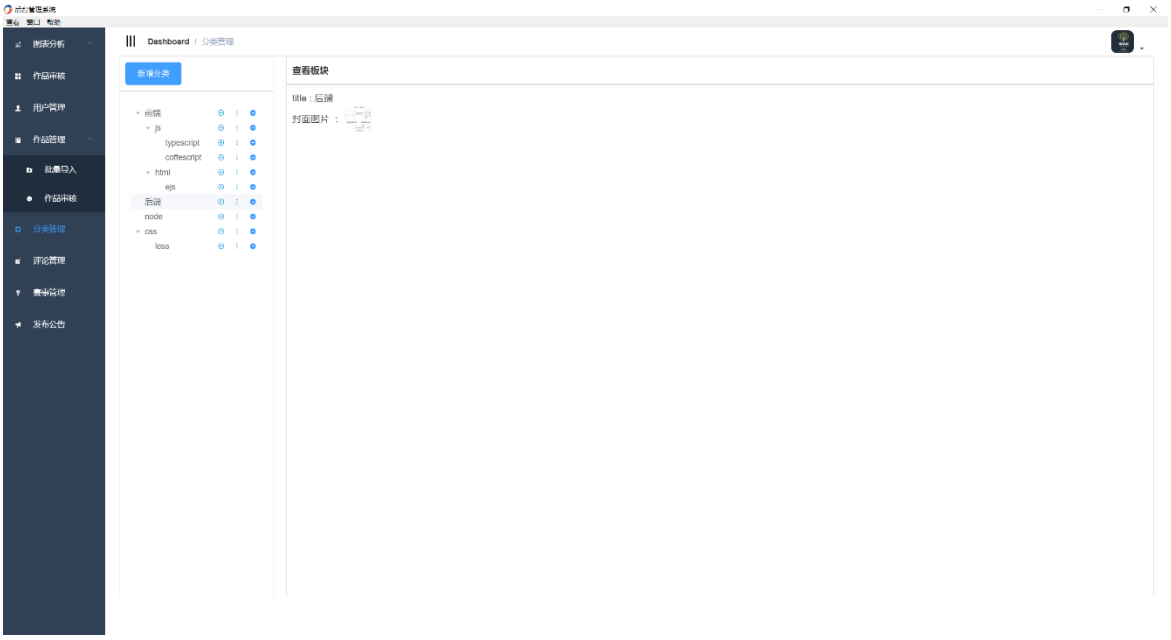

图 7-12 赛事管理界面

7.3.12 用户管理界面

在这个界面，管理员可以禁止某个账号的使用，也可以批量上传用户，可以根据条件筛选用户，如图 7-13 所示。

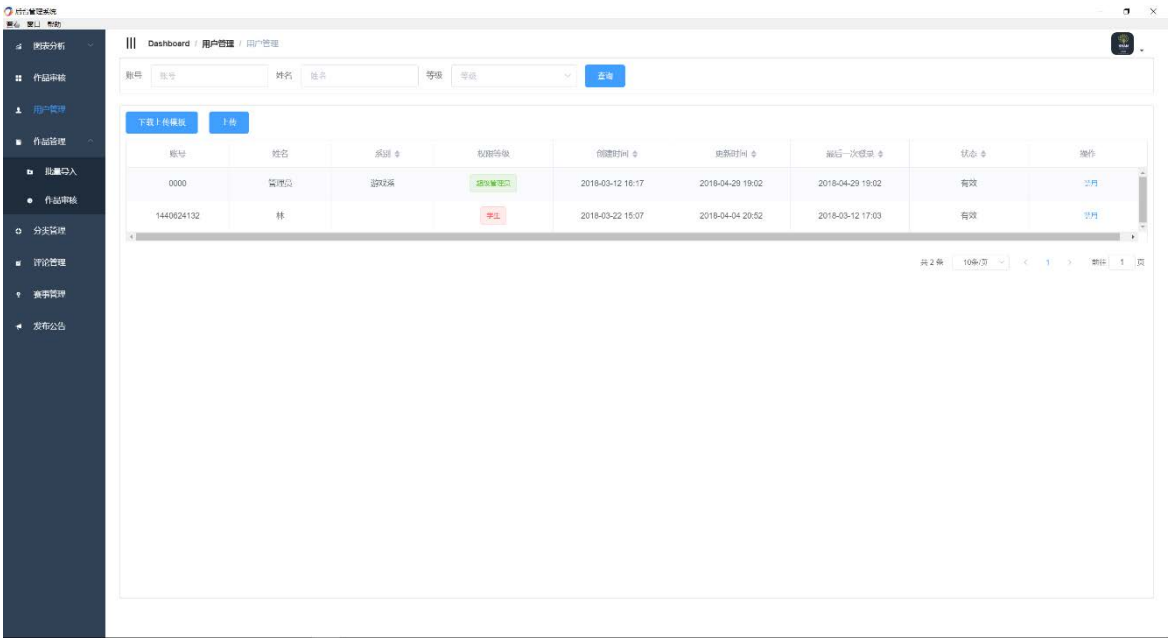

图 7-13 用户管理界面

7.3.13 数据图表界面

数据分析界面，管理员可以很直观的查看数据，支持柱状图，折线图，数据视图，也支持数据导出下载功能，如图 7-14 所示。

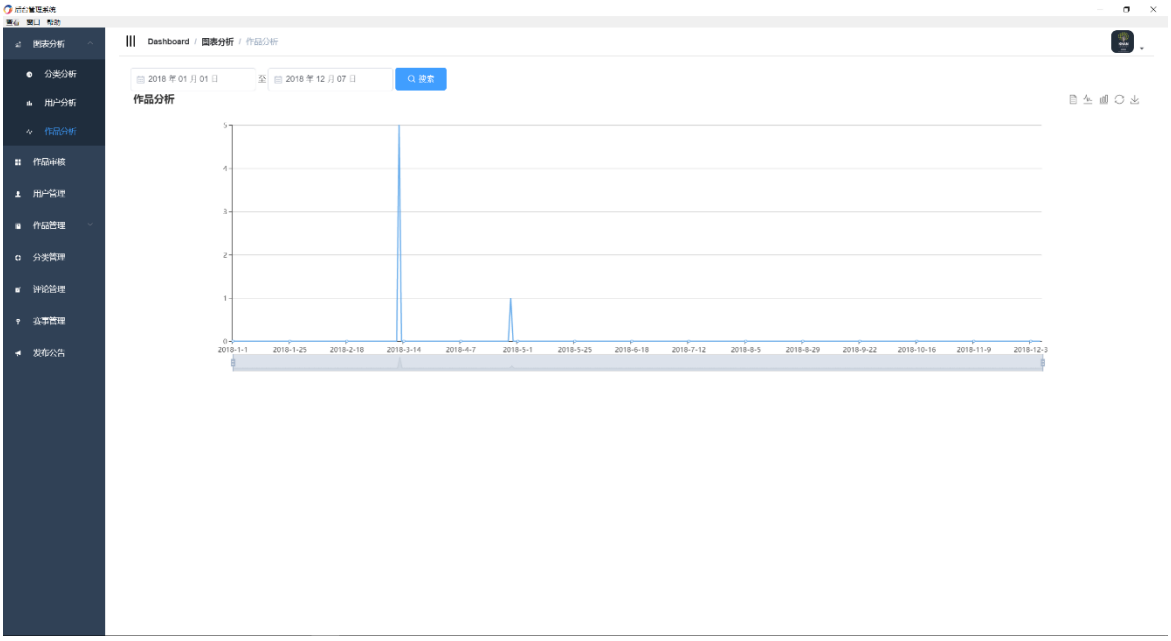

图 7-14 数据图表界面

参考文献

[1] 张佳伟,张涛,周叶. 基于 Electron 的跨平台客户端技术[J]. 智能计算机与应用,2017,7(03):120-122.

[2] Muhammed Jasim. Building Cross-Platform Desktop Applications with Electron

[3] Node.js 6 紧跟 V8 引擎更新,提高了速度和安全性[J]. 电脑编程技巧与维护,2016,(09):4.

[4] 朱丽英. 基于 Node-Webkit 平台的 JavaScript 工具集研究与实现[D].电子科技大学,2016.

[5] 王姝文. 嵌入式浏览器跨平台服务组件研究与设计[D].电子科技大学,2011.

[6] 戴翔宇. Web 前端工程组件化的分析与改进[D].吉林大学,2016.

[7] 闪森林,张玺. Weex 跨平台开发方案研究与应用[J]. 信息通信,2017,(04):112-113.

[8] 唐成. 基于 Blink 的 WebCL 基础模块的设计和实现[D].中山大学,2015.

[9] 黄雄. 基于 HTML5 的视频音频传输技术的研究与设计[D].广东技术师范学院,2014.

[10] 常闻宇. HTML5 跨平台技术在视频点播系统中的研究与应用[D].东华大学,2013

[11] 王玉平. 视频、音频文件跨平台播放的研究与实现[D].中国地质大学(北京),2016.

[12] 麦冬,陈涛,梁宗湾. 轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版),2017,(07):58-59.

[13] 王胜,张靖. 基于 Vue.js 高速路政管理系统的设计与实现[J]. 电脑知识与技术,2017,13(21):86-88+101.

[14] 徐頔,朱广华,贾瑶. 基于 VueJs 的 WEB 前端开发研究[J]. 科技风,2017,(14):69.

致 谢

本论文在袁冠远导师的悉心指导下完成。老师以负责，耐心，求实的治学态度、对我产生非常重要影响。无论是在思想上、学习上，还是生活上，袁冠远导师都对我严格要求，热情鼓励和殷切关怀，也使我能够在挫折面前重拾信心，就算在取得小小成绩时也能够保持自我意识的清醒，看到自己的不足。袁冠远导师是我人生方向的灯塔和动力，导师给我的谆谆教诲必让我终生受益。因此在此谨向我的导师以最朴实无华的话语表示我最诚挚的感谢以及最崇高的敬意！

在此同时我也非常感谢刘生建老师和我在广州大学华软软件学院上的所有的任课老师和同学，我深深地感谢大学所有在学习和生活上给予过关心和帮助的人们，感谢你们使我进步以及坚强，谢谢你们。