

# 本科毕业设计论文

**基于MVVM框架的线上图片编辑器**

**设计与实现**

院 系 管理学院管理科学与信息管理系

专业班级 信息管理与信息系统1802班

姓 名 迪力夏提·麦麦提敏

学 号 U201815616

指导教师 陈伟运 副研究员

2022 年 4月

# 学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月

**摘要**

随着数字技术的不断发展，我们与图片资源的关系变得越来越紧密。无论是在日常生活中，还是在学习或者工作当中，图片都发挥着举足轻重的作用，大众对图片处理的需求也在日益激增。随之出现了非常多样化的图片处理软件，例如PS、AI、CDR等。这些软件具有比较强大、全面的图片处理功能，可以胜任日常或者专业化的图片处理需求，因此也深受设计师等专业工作者的喜爱。但是它们作为专业化软件的缺陷也非常明显，最大的缺陷就是对大众不够友好。对于一个没有专业化需求、没有相关经验的普通用户来讲，使用这些软件进行一些简单的操作也需要付出较大的学习成本和时间。线上图片编辑器应运而生，它们编辑操作简单，不需要下载，学习成本较低，可以胜任普通用户对图片处理的一般性需求。

本文使用基于MVVM模式的三种前端框架实现了一种简单易用的在线图片编辑器。系统基于前后端分离的开发思想。前端利用HTML5新添加的Canvas元素提供的绘画能力，主要采用Angular、React、Vue等三种基于MVVM的前端框架，使用集成前端路由的方式实现单页面应用；后端使用Node.js服务端实现平台来提供数据存储等服务。首先，本文对在线图片编辑器项目进行了需求分析，在概要设计部分对系统的整体架构、包括前后端的逻辑架构、数据库设计等进行了高层描述，在系统的具体设计部分通过UML类图、时序图等方式对前后端结构进行详细设计。其次，在系统设计的基础上，本文基于Angular、React、Vue等三种框架分别实现了图片编辑器的服务端和前端，具体功能包括图片上传、图片剪裁、旋转、文字添加、画笔等功能。最后，鉴于线上图片编辑器对响应速度、视觉稳定性等用户体验的特定需求，本文基于Web Vitals对Angular、React、Vue等三种框架实现的图片编辑器进行了性能测试，并通过gzip压缩、减少HTTP请求、动态导入、图片懒加载、使用浏览器存储等方法对不同框架实现的图片编辑器进行性能测试与优化。测试结果表明，通过编写高复用性代码并使得不同框架系统实现所需代码量大致相当的条件下，以上优化方法能大大提高在线图片编辑器的性能。

关键词**：**Web；前端框架；Vue；React；Angular

**Abstract**

With the continuous development of digital technology, the relationship between us and image resources has become closer and closer. Whether in daily life, study or work, pictures play an important role, and the public demand for picture processing is also increasing. A variety of image processing software have emerged, such as PS, AI, CDR and so on. These software have powerful and comprehensive image processing functions, and can meet the daily or professional image processing needs. Therefore, they are also loved by designers and other professional workers. However, their defects as specialized software are also very obvious. The biggest defect is that they are not friendly to the public. For an ordinary user who has no professional needs and no relevant experience, using these software for some simple operations also needs to pay a large learning cost and time. Online image editors came into being. They are simple to edit, do not need to download, and have low learning costs. They can meet the general needs of ordinary users for image processing.

This paper uses three front-end frameworks based on MVVM pattern to implement a simple and easy-to-use image editor. The system is based on the development idea of front-end and back-end separation. The front-end uses the drawing ability provided by the newly added canvas element in HTML5. It mainly adopts three current mainstream MVVM based front-end frameworks, such as angular, react and Vue, and uses the integrated front-end routing method to realize single page applications. The backend uses node JS server implements a platform to provide data storage and other services. Firstly, this paper analyzes the requirements of the online image editor project, describes the overall architecture of the system, including the front and rear logical architecture, database design, etc. in the outline design part, and designs the front and rear structure in detail through UML class diagram, sequence diagram, etc. in the specific design part of the system. Secondly, on the basis of the system design, this paper implements the server and front end of the image editor based on the three frameworks of angular, react and Vue. The specific functions include image upload, image clipping, rotation, text addition, brush and so on. Finally, in view of the specific needs of online image editors for user experience such as response speed and visual stability, this paper tests the performance of image editors implemented by three frameworks, including angular, react and Vue, based on Web vitals, and tests and optimizes the performance of image editors implemented by different frameworks through gzip compression, reducing HTTP requests, dynamic import, lazy image loading, and using browser storage. The test results show that the above optimization methods can greatly improve the performance of online image editor under the condition of writing high reusability code and making the amount of code required by different framework systems roughly equal.

**Key Words:** Web; Front-end Framework; Vue; React; Angular

**目 录**

**[摘要](#_Toc1783)** [I](#_Toc1783)

**[Abstract](#_Toc5797)** [II](#_Toc5797)

**[1 绪论](#_Toc28792)** [1](#_Toc28792)

[1.1 研究背景及意义 1](#_Toc21146)

[1.2 相关技术发展脉络以及基于MVVM的技术框架 1](#_Toc32471)

[1.3 主要研究内容 4](#_Toc24038)

[1.4 主要工作及章节安排 4](#_Toc16980)

**[2 相关技术概览](#_Toc18790)** [6](#_Toc18790)

[2.1 Angular框架及其特点 6](#_Toc11151)

[2.2 React框架及其特点 7](#_Toc8610)

[2.3 Vue框架及其特点 8](#_Toc8627)

[2.4 TypeScript 8](#_Toc4823)

[2.5 Node.js及其特点 9](#_Toc25300)

[2.6 Express 10](#_Toc25166)

[2.7 MongoDB 10](#_Toc18561)

[2.8 Web Vitals 11](#_Toc3896)

**[3 线上图片编辑器需求分析](#_Toc6547)** [12](#_Toc6547)

[3.1 功能性需求分析 12](#_Toc16303)

[3.1.1 图片的上传 12](#_Toc7103)

[3.1.2 图片的裁剪 13](#_Toc659)

[3.1.3 添加文字 13](#_Toc27259)

[3.1.4 画笔 13](#_Toc30029)

[3.1.5 旋转 14](#_Toc1578)

[3.1.6 还原 14](#_Toc5321)

[3.2 非功能性需求分析 14](#_Toc26043)

[3.2.1 系统可维护性 14](#_Toc31525)

[3.2.2 系统易用性 15](#_Toc26135)

[3.2.3 系统安全性 15](#_Toc6113)

[3.2.4 系统性能 15](#_Toc3597)

[3.3 本章小节 15](#_Toc7838)

**[4 系统设计与实现](#_Toc4770)** [16](#_Toc4770)

[4.1 系统设计 16](#_Toc2209)

[4.1.1 数据库设计 17](#_Toc9971)

[4.1.2 服务端架构的具体设计 17](#_Toc10174)

[4.1.3 前端架构的具体设计 19](#_Toc15367)

[4.2 系统实现 20](#_Toc32091)

[4.2.1 服务端实现 20](#_Toc14400)

[4.2.2 前端实现 22](#_Toc24222)

[4.3 系统效果 30](#_Toc11088)

**[5 性能测试与优化](#_Toc30474)** [34](#_Toc30474)

[5.1 测试工具 34](#_Toc27355)

[5.2性能指标 34](#_Toc1307)

[5.2.1 LCP 34](#_Toc9521)

[5.2.2 CLS 34](#_Toc18590)

[5.2.3 FCP 35](#_Toc8896)

[5.2.4 TTFB 35](#_Toc6351)

[5.3 性能测试 35](#_Toc29744)

[5.4 性能优化 38](#_Toc2195)

[5.4.1 gzip压缩 38](#_Toc31303)

[5.4.2 减少HTTP请求 39](#_Toc28992)

[5.4.3 动态导入 39](#_Toc24072)

[5.4.4 图片懒加载 40](#_Toc16584)

[5.4.5 使用浏览器存储 40](#_Toc681)

[5.5 优化结果 40](#_Toc3670)

[5.6 框架横向比较 43](#_Toc2128)

**[6 总结与展望](#_Toc20623)** [45](#_Toc20623)

[6.1 总结 45](#_Toc28096)

[6.2 展望 46](#_Toc11027)

**[致谢](#_Toc4184)** [48](#_Toc4184)

**[参考文献](#_Toc13568)** [49](#_Toc13568)

# 

# 1 绪论

## 1.1 研究背景及意义

随着计算机技术的飞速发展和互联网的快速普及，电子图片成为了日常生活中最主要的信息载体之一。新闻、电子书和社交平台等都极度依赖于图片资源。腾讯企鹅智库的调查结果显示，近40%的图片类应用用户至少安装了一款图片处理类软件[[[1]](#endnote-0)]。随着人们对图片加工处理的需求在不断增长，出现了越来越多的图片处理软件。其中最有代表性的是Photoshop，它是一个功能十分强大的图片处理软件，可以实现一系列复杂的编辑操作，被广泛应用于许多的专业领域。但是对于非专业用户而言，它也存在着一些缺陷。首先该软件需要购买和安装，软件占用硬盘存储空间也较大，运行时占用大量计算机内存资源，因此对用户的电脑配置有较高的要求。其次该软件使用复杂，对普通用户不友好，需要用户较多的时间学习其操作方法。对于技术背景较弱的普通用户而言，他们更需要的是一个便捷、易用、操作简单的图片处理器。在此背景下，基于Web的简易线上图片编辑器已经成为了大多数用户的迫切需求。

国内诸多互联网应用（如微信、微博）为了让用户得到良好的图片编辑体验，投入了大量的技术研发力量，在这方面取得了卓越的成果。但还是有很多网站在Web图片处理技术上需要提高，他们只关注最终的编辑效果，不考虑用户的使用体验和用户需求的多样性，背弃了以用户为中心的原则。以图片的裁剪操作为例，有很多网站只能够裁剪图片固定宽高的部分，用户不能根据自己的需要调整裁剪大小。还有一部分在裁剪完成之后不相应地调整图片大小，使得裁剪之后的图片出现模糊、变形等等情况。其他一些编辑操作比如在添加文字时用户不能选择字体样式、字体大小和字体颜色等属性，导致出现字体被背景颜色所掩盖、编辑效果不美观等问题，使得用户对于图片编辑的基本需求得不到满足。随着图片处理技术不断向Web端迁移，Web图片处理领域中存在的一系列问题将越来越重要。针对用户对在线图片编辑的需求，实现基于浏览器的、只需要通过本地上传图片即可开始对图片进行多种编辑操作的在线图片编辑器具有很强的现实意义。

**1.2 相关技术发展脉络以及基于MVVM的技术框架**

早期的web应用主要是静态页面的浏览，而静态页面由HTML构成，浏览器端本身的图片处理能力非常弱。HTML作为一个标记语言，到现在为止依然是Web页面的基本骨架。这时的网页不具备与用户交互的能力，也没有任何动态效果。随着网络的发展，基于Internet的Web应用也变得越来越复杂，服务器上存放的静态资源已经无法满足用户的需求。更多的时候页面的展示取决于具体的用户，根据用户的请求来生成页面并发送给用户，这里面的涉及到了大量的数据库查询、后台计算等。相应的，用于搭建服务端的技术和语言也相继涌现，如ASP、JSP等等。PHP3的正式发布开启了真正的动态交互阶段，但PHP是嵌入到HTML页面的脚本语言，并没有前后端的明确分工，所以此时也没有专业的前端开发人员，页面的渲染工作也由后端工程师顺便完成。

Ajax的出现改变了传统的用户请求-等待-响应这种Web交互模式，采用异步交互机制避免了用户对服务器响应的等待，提供了更好的用户体验，也为在线图片处理提供了新思路。此外，它也改变了用户请求-服务器响应-页面刷新的用户体验方式，提供了页面局部刷新实现机制。Ajax开启了Web2.0时代。Ajax的出现使得前端开发在Web开发中扮演越来越重要的角色，随之也相继出现了很多前端框架和库。jQuery是其中最流行的一个，jQuery旨在简化前端的DOM操作，减少前端代码量，提升开发速度。当应用程序越来越的地需要采用交互模式，应用逻辑和页面效果开始逐渐变得越来越复杂，操作DOM的频率也越来越高。而DOM操作是所有Web应用性能开销最大的一部分，但jQuery这种命令式操作DOM的方式由于缺乏正规的组织形式，代码也会随着项目的增大而变得越来越难以维护。因此，实现基于Web端的在线图片处理在技术上还比较复杂。

HTML5的出现，再次改变了Web应用开发的发展格局，也为基于Web的图片处理提供了强大的技术支撑。在HTML5问世之前，大多数基于Web的图片处理器还是基于通过网络传输，在服务端完成图片的各类编辑操作，再将结果反馈给浏览器的模式，浏览器的图片处理能力非常弱。HTML5中新增的Canvas元素给浏览器提供了图片绘制的能力，利用Canvas元素的上下文对象可以在屏幕中进行线段、圆弧、四边形等等图形的绘制。Canvas也提供了单独的图片绘制方法，结合其他特性，可以对图片进行多样化的处理操作。鉴于Web网站相对于PC端软件更便捷灵活的优势，图片处理技术开始向Web端迁移。

同时，HTML5也使得前端能够实现的交互功能越来越多，相应代码的复杂度也在迅速提高。在此背景下，偏重于后端的MVVM模式开始出现在前端部分。从2010年10月出现的Backbone开始，基于MVVM模式的Angular、React和Vue等框架相继出现。这些框架的应用，使网站从Web Site进化成了Web App，开启了网站应用的SPA（单页应用程序）时代。为了应对jQuery这种命令式操作DOM方式中存在的各种问题，React、Angular和Vue等采用MVVM模式的前端框架相继涌现，并且迅速改变网络编程的格局。使用这些框架，程序员将无需再直接命令式地去操作DOM，而是直接使用声明式操作DOM的办法。所谓声明式操作指的是通过将状态与视图绑定，当状态发生变化时自动地通知视图，并将其更新。这为在线图片处理系统的实现提供了新的技术思路。

同时，组件化的设计理念也随着这些框架而开始出现，成为Angular、React和Vue等框架的核心理念之一，也大大提高了在线图片处理系统的前期开发和后期维护效率。这也是这些框架相比于jQuery最优越的地方之一。jQuery最大的缺点之一就是当项目变得越来越大时，代码就会越来越混乱，相互耦合度高，重复代码增多等等，使得项目变得及其难以维护。这些缺点在新的框架中得到很大的改善的最大原因就是组件化开发。组件是浏览器屏幕中的独立的UI片段，它允许我们将UI分解成可复用的代码片段，并对每个片段进行独立的设计。组件也让设计者可以将整个页面分解成相互独立的部分，各个UI片段之间没有紧密的逻辑耦合。整个页面就是几个独立的UI片段之间的拼接，同时因为各个UI片段之间相互独立，开发人员可以很轻松地实现代码复用，从而大大提高应用系统的开发效率。

基于MVVM的Web前端框架另外一个比较大的特性是包含了前端路由解决方案。传统的Web页面跳转路由一般都需要向后端发送请求，后端再向浏览器发送HTML文档，再将页面呈现至页面上，因此每次的路由跳转都需要将页面刷新一次。因为HTML文档的解析和渲染都需要时间，每次跳转都刷新页面显然会影响到用户体验。而在前端路由中这些问题都迎刃而解，不同视图（组建的模板）的内容都是在同一个页面中渲染，页面的跳转都是在浏览器端完成，所当路由跳转时也不再需要刷新页面。虽然，基于MVVM的Web前端框架在实现Web应用具有很多的优势，但在如何使用这些前端框架的实现在线图片编辑器的相关研究还比较少，特别是在控制各种实现方式的代码量的条件下对各种框架的性能进行比较的研究还不多。

## 1.3 主要研究内容

本文拟使用MVVM模式的Angular、React和Vue等三种前端框架，基于前端路由以前后端分离的模式实现基于Web的线上图片编辑器。首先，本文需要在线图片编辑器项目进行需求分析，在概要设计部分对系统的整体架构、包括前后端的逻辑架构、数据库设计等进行了高层描述，在系统的具体设计部分通过UML类图、时序图等等方式对前后端结构进行详细设计。其次，在系统设计的基础上，本文需要基于Angular、React、Vue等三种框架分别实现图片编辑器的服务端和前端，具体功能包括图片上传、图片剪裁、旋转、文字添加、画笔等功能，同时控制每种实现方案的代码行数，以保证能够对各种框架在同等代码量的条件下进行比较。项目开发需要遵循软件开发流程对系统进行需求分析、概要设计和详细设计，并使用Node.js服务端开发平台提供数据的上传和存储服务。鉴于线上图片编辑器对响应速度、视觉稳定性等用户体验的特定需求，在基本实现系统功能的前提下，本文拟基于Web Vitals对Angular、React、Vue等三种框架实现的图片编辑器进行性能测试，并通过测试研究不同框架实现的图片编辑器的性能优化方法，找出各个子系统存在的性能缺陷并进行针对性的优化，提高系统性能，为今后基于这3种框架开发类似的应用系统提供技术指导。

## 1.4 主要工作及章节安排

本文结构安排如下：

第一章为绪论，该部分简单概述本文的研究背景和意义，同时简单回顾了与本文研究目的相关的技术发展脉络以及MVVM模式和相关技术框架，简要地给出了本文的主要研究内容。

第二章对系统开发使用到的各种技术做介绍和汇总。其中包括Angular、React和Vue框架的基本特点以及除了框架之外的后端开发工具，数据库等。

第三章对系统进行详细的需求分析，从用户体验角度出发对系统各个模块的功能性需求和非功能性需求进行分析，为后面的系统设计于开发工作奠定基础。

第四章进行系统的设计与实现，包括系统架构的设计、前后端交互、功能的实现过程及原理，并详细介绍了开发中遇到的问题以及解决方式，并分析了相关问题解决方法在各个框架中的不同之处。

第五章是系统的性能测试与优化。首先，通过测试、数据收集和统计的方式对系统的性能做出具体的、量化的评价；然后，基于测试结果对系统进行性能优化，通过比较优化前后的性能数据来证实优化方式的有效性。

第六章对全文进行总结和展望，并指出将来实现线上图片编辑器需要改进提高的部分。

**2 相关技术概览**

本章主要针对系统开发所涉及的主要技术进行分析和研究，为后续具体的系统设计和实现打好基础。

## 2.1 Angular框架及其特点

AngularJs在2009年发布，之后被Google收购。它是一款基于MVVM模式的用于构建用户界面的前端框架，可以用于构建高性能、精致、复杂的单页面应用。也可以实现跨平台的移动端，桌面端应用。Angular最大的两个特点是依赖注入和服务，下面进行详细的介绍。

（1）依赖注入

依赖项是指某个类执行其功能所需的服务或者对象，依赖注入也是一种设计模式，具体的过程是类从外部请求所需的依赖，而不是在内部创建。此设计模式可以用来提升应用程序的灵活性和模块化程度。

通过angular提供的Injectible装饰器，可以很轻松的实现在指定位置上的依赖注入，Injectible装饰器的参数值为root，这表示这个服务在整个应用程序中是可见的，可以在程序任何地方注入这个服务来调用[[[2]](#endnote-1)]。代码需要重点关注的地方是构造函数constructor的参数。要想依赖某个服务，那么只需要在另外一个服务或者组件的构造函数中定义属性即可，之后就可以使用。此后服务的实例化等问题不再需要去关心，因为angular在注入服务时已经自动完成。

1. 服务

在angular中任何单一的功能都可以定义为服务，可以为这些功能生成单独的服务并作为依赖注入到其他的服务和组件中，在指定目录下运行生成服务命令，就会生成一个对应的服务文件和测试文件，图片编辑器在三个项目中的实现都受到了这种思想的启发。虽然React和Vue没有提供依赖注入此类功能，但是在项目的开发过程中，以实例化类并导出的方式来尽量模仿angular中的服务的概念。这确实在整个开发过程中带来了许多的便利。

## 2.2 React框架及其特点

React是用于构建[用户界面](https://baike.baidu.com/item/%E7%94%A8%E6%88%B7%E7%95%8C%E9%9D%A2/6582461" \t "https://baike.baidu.com/item/react/_blank)的[JavaScript](https://baike.baidu.com/item/JavaScript/321142" \t "https://baike.baidu.com/item/react/_blank)库，起源于Facebook的内部项目，著名的社交网站Instagram就是用React所构建，该项目于2013年5月开源，成为前端开发领域最流行的框架之一。React不断发展，从最初的一个UI引擎逐渐转变为了一套Web App前后端解决方案。衍生出的React Native可以实现一套代码，改代码能在在桌面、浏览器和手机上的同时运行。以下针对React最显著的特性做详细介绍。

（1）单向数据流

与Angular和Vue不同，React遵循单向数据流的设计理念，也被称为“数据瀑布”，没有表单数据的双向绑定。数据的传递只能是从数据到视图，它不提倡在子组件内改变父组件中的变量。单向数据流的好处是数据易于追踪，组件之间的关系不易混淆，利于维护。但是从需要编写的代码量来讲，表单数据的改变在React中需要手动监听并实时更新绑定数据，相比于其他两种框架来讲需要编写多一些代码，但是对整个开发流程和周期来讲影响不大。

（2）JSX语法

React框架另外一个鲜明的特点就是它使用JSX取代了模板语法。JSX是一个JavaScript的语法扩展，它具有JavaScript的所有功能。React认为渲染逻辑本质上与其他UI逻辑内在耦合，比如在UI中展示响应式数据、为DOM元素绑定事件等等。因此React并没有像Angular那样采取将逻辑和模板分离到不同文件中的组件形式，这种做法在开发时具有视觉上的辅助作用，还可以使React在开发时给出更多有用的警告和错误信息。

1. Hook API

Hook 是React16.8中推出的新特性，用于取代之前的class写法。Hook改变了在原先的class组件中代码复用困难的问题，函数式的写法使得React可以在不修改组件结构的情况下复用状态逻辑。除此之外，Hook还改善了在class组件中某些功能的逻辑被分散到不同的生命周期函数中的问题。随着项目体积的不断增大，被分散到各处的单一逻辑变得难以理解，随之而来的是开发人员的沟通和维护成本增大，Hook彻底解决了此问题[[[3]](#endnote-2)]。

## 2.3 Vue框架及其特点

Vue是一套用于构建[用户界面](https://baike.baidu.com/item/%E7%94%A8%E6%88%B7%E7%95%8C%E9%9D%A2/6582461" \t "https://baike.baidu.com/item/Vue.js/_blank)的渐进式[JavaScript](https://baike.baidu.com/item/JavaScript/321142" \t "https://baike.baidu.com/item/Vue.js/_blank)框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。当与现代化的工具链以及各种支持[类库](https://baike.baidu.com/item/%E7%B1%BB%E5%BA%93/3351433" \t "https://baike.baidu.com/item/Vue.js/_blank)结合使用时，Vue 也完全能够为复杂的单页应用（[SPA](https://baike.baidu.com/item/SPA/17536313" \t "https://baike.baidu.com/item/Vue.js/_blank)）提供驱动[[[4]](#endnote-3)]，下面详细介绍Vue的特性。

（1）渐进式框架

与其他两种框架不同，Vue被设计为自底向上的逐层应用，Vue的核心库只关心视图层，渐进式结构从里到外分别是声明式渲染、组件系统、前端路由、大规模状态管理和构建工具。开发者可以结合自身需求和项目规模来自由选择应该应用到渐进式结构的那一层，因此该框架可以较好地适配不同规模的项目[[[5]](#endnote-4)]。

（2）单文件组件

Vue与其他两种框架另外一个明显区别是单文件组件。所谓单文件组件就是在Vue项目中以.vue为扩展名的文件，开发者可以在此文件内分别编写组件模板、逻辑和样式。这种做法相比于React的一个优越点是样式表具有作用域，无需担心组件内的样式会渗透到全局中而与其他组件样式发生冲突。单文件组件还有助于逻辑聚合，因为每个组件的所有视图和逻辑都集中到了一个文件当中，有助于快速定位bug，改变特定组件的行为等。

## 2.4 TypeScript

TypeScript是JavaScript的类型超集。JavaScript作为解释性语言的一个特点是它是一个弱类型语言。所谓弱类型语言就是JavaScript不存在类型声明和类型判断。这种情况的一个弊端就是无法在编译阶段检测出潜在的类型错误，只有在代码运行报错时才可以察觉到代码中的问题，这在开发过程中无疑是一个很大的隐患。

TypeScript针对上面的问题提出了解决方案，TypeScript允许开发者在开发JavaScript程序时对类型进行静态检验，在开发过程中严格限制类型，使得程序更加健壮。

除了静态类型检验之外，TypeScript还有装饰器，通过指定JavaScript目标编译版本实现针对不同浏览器和平台的兼容等一系列功能，使JavaScript更加健全。

## 2.5 Node.js及其特点

Node.js是JavaScript的服务端运行环境，使得JavaScript也可以用于构建Web服务器。起初，Node.js的发明者Ryan Dahl目的就是开发一个简易的Web服务器，之后它变成了构建网络应用的一个基础框架。开发者可以在它的基础上构建更多的应用，诸如服务器、客户端、命令行工具等。Node成为不强制共享任何资源的单线程、单进程系统。同时，它包含非常适宜网络的库，为构建大型分布式应用程序提供基础设施[[[6]](#endnote-5)]。下面对Node.js的重要特性进行详细的介绍。

（1）单线程

Node.js采用单线程模型，保持了JavaScript在浏览器运行环境中的特点。单线程的好处就是不需要像其他多线程编程语言那样时刻关注多线程的同步、死锁问题，也没有多线程通信带来的性能上的开销。但是单线程模型也有它自己的问题，比如无法利用多核CPU、大量计算导致的CPU无法调用异步I/O等。Node也为此类问题提出了解决方案，像浏览器中的Web Worker那样，在Node中也可以使用child\_process来创建子进程，再通过事件的方式来与主线程通信。

1. 非阻塞异步I/O

Node.js采用非阻塞的异步I/O，与同步模型不同，Node.js中的某一个I/O流的执行不会阻塞程序中其他的代码的调用，在调用某一种异步操作时程序并不会等待其调用结果，而是立即开始执行后续代码。异步任务完成之后会通过事件回调的方式来执行对应的操作。图2-1给出了Ajax请求在Node中的处理过程。

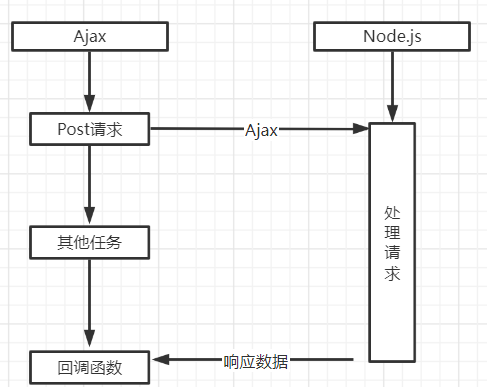


图2-1 Ajax请求在Node.js中的处理过程

图2-1可以看到像Ajax请求之类的异步操作在Node.js中是如何进行处理的，任何异步操作比如网络请求，文件的读写等都不会阻塞代码的运行，而是在异步操作处理完成时执行回调函数。回调函数的调用和执行顺序由事件轮询机制决定，事件轮询是指在执行异步操作时Node会将对应的回调函数推入任务队列，之后继续向下运行代码，之后不定期检查任务队列，如果某个异步操作已完成就执行对应的回调函数，之后将其从任务队列中删除。任务队列分为微任务队列和宏任务队列，Node会先检查微任务队列并执行，因此微任务的优先级要比宏任务高。这种处理方式使得Node能够一直处于高效并且随时能够做出快速响应的状态[[[7]](#endnote-6)]。

## 2.6 Express

Express 是一个保持最小规模的灵活的 Node.js Web 应用程序开发框架，为 Web 和移动应用程序提供一组强大的功能。它提供的工具使Web程序开发变得更容易。Express提供了统一的视图系统，可以在开发过程中使用任何模板引擎，各种中间件可以根据自己的需求随意添加到程序中以实现各种不同的功能。Express的主导思想是程序的需求和实现方式多变，主张使用轻量的框架实现需要的应用，非必要不引入任何多余的模块。Express和整个Node社区都致力于做出更轻量的、模块化程度更高的Web框架，而不是一个整体式框架。

Express具有非常高的扩展性，它可以根据需要任意配置中间件来轻易地实现Web服务器的所需功能。Node社区中提供了非常丰富的中间件，可以通过npm包管理器下载之后直接在程序中引用。这也是为什么Express被称为是一个轻量级的框架，因为在安装时它不会引入任何多余的插件，不会提供任何额外的功能，而是给开发者提供了更多的自由空间，让开发者来决定服务器的架构和需要实现的功能，高灵活度、高扩展性的特点使Express在Node社区迅速流行起来。

## 2.7 MongoDB

MongoDB是一个面向文档存储的NoSQL数据库，它是一种强大、灵活、可扩展的数据存储方式，它扩展了传统的关系型数据库的很多有用的功能，如范围查询、排序等等，所以它的功能非常的丰富。MongoDB致力于容易上手、便于使用，它的数据模型对开发者来说非常的友好，配置选项对数据库管理员来说也很轻松，并且有驱动程序和数据库shell提供的自然语言式的API，MongoDB可以为开发者扫清障碍，让开发者关注程序本身而不是数据的存储和读取。

为了提升开发效率，最大程度减少项目的复杂性，操作MongoDB数据库时使用了Mongoose。Mongoose框架是由美国Infor公司自主研发的软件开发应用框架，它是一种新型的技术平台，是一个高度集成的框架工具，可以让工程师更便捷的设计程序和部署应用，强化和扩展企业的核心解决方案。Mongoose框架提供了强大的支持，集成了服务端与客户端的通信机制、各种事件的处理、界面设计、访问数据库、License管理、用户管理等等功能，具有良好的适应性和可扩展性。

## 2.8 Web Vitals

当对一个Web站点进行性能优化的时候，绝大部分的技术人员都会首先想着从后端进行优化(后台程序优化、数据库优化)，但从Web站点的布局来看，前端才是真正与用户交互的模块。因此，除了从后端对Web站点进行优化外，还应重点在前端方面对其进行性能优化研究。据有关数据统计，有10%~20%的用户响应时间花费在了从服务器获取HTML文档上(后端优化部分)，剩下的20%-90%的用户响应时间都是花费在下载页面各种组件上(前端优化部分)。如果想要提高网站的整体性能，就必须将重点放在对前端部分的优化上。如果将后端响应的时间减少一半，那么网站的整体响应时间就会减少5%-10%；但如果将前端的响应时间减少一半，那么网站整的体响应时间就会减少40%-50%[[[8]](#endnote-7)]，这对Web站点的性能来说是一个巨大的提升。

Web Vitals就是一个关注于测试和量化前端性能的工具，它由Google创立，提供了几种被认为对用户体验十分重要的性能指标，并且能够定位屏幕中对网站性能影响最大的模块，使得开发者可以专注于这一部分进行针对性的优化，由此提升用户体验、提高产品竞争力。

# 

# 3 线上图片编辑器需求分析

在本章将通过对线上图片编辑器进行需求分析，明确该项目要解决的核心问题，具体从功能性需求和非功能性需求两方面分析，详细说明该项目每一个模块要完成的功能和职责，为系统后续的设计和实现做好准备工作。

## 3.1 功能性需求分析

在本节中，为了合理的设计系统的功能集，对系统用户进行详细的功能性需求分析，并使用用例图和用例表的形式来具体说明。系统用例图如图3-1所示。

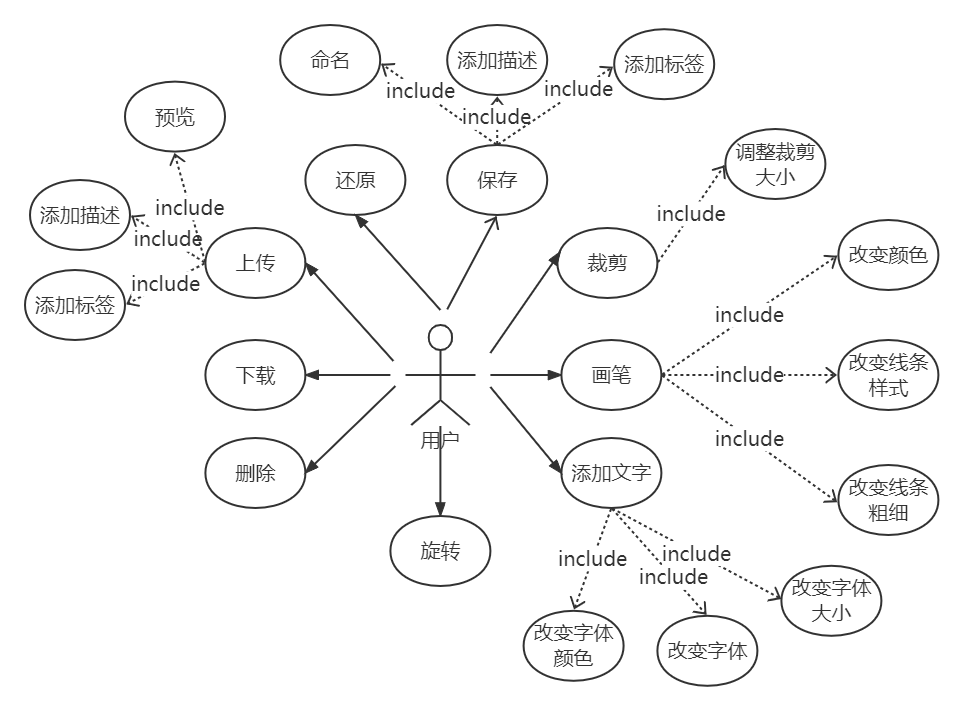


图3-1 用户用例图

本系统需要实现的功能包括图片的上传、下载、删除、裁剪、画笔、添加文字、旋转、还原、保存等，这些是用户需求最为广泛的功能。

**3.1.1 图片的上传**

图片的上传是Web图片处理器的必备功能，是对图片进行编辑操作的基础。出于用户体验考虑，在进行图片上传的过程中，用户可以对图片添加描述和标签，便于图片的分类和管理。除此之外，在用户确认上传之前需要展示出图片的预览效果，这可以作为一种反馈形式使得用户对上传效果更加有把握。图片的上传过程可能出现一系列的异常情况，比如图片格式、图片大小、网络异常，在此过程中都需要对用户给出及时的反馈。图片上传的模块用例如表3-1所示。

表3-1 图片上传用例描述

|  |  |
| --- | --- |
| ID | UC01 |
| 名称 | 图片上传 |
| 参与者 | 普通用户 |
| 优先级 | 高 |
| 用例描述 | 该用例用来描述用户的上传操作 |
| 前置条件 | 网络正常，设备正常，数据库正常 |
| 用例流程 | 1. 用户点击上传按钮，弹出图片上传弹框。 2. 用户点击弹框中上传按钮，打开资源管理器，选择图片并确认，预览图片的上传效果。 3. 用户输入图片描述和图片标签，点击确认。 4. 浏览网页查看图片是否上传成功。 |
| 异常事件流 | 1. 点击上传按钮无法弹出上传弹框。 2. 选择的图片不符合JPG或者PNG格式。 3. 选择的图片体积超过2MB。 4. 图片无法上传到数据库。 |
| 后置条件 | 无 |

**3.1.2 图片的裁剪**

图片的裁剪是需求量最大的图片编辑操作之一，广泛应用与现实生活中。比如一个社交平台的用户想要上传一个新的头像，如果原图比较大，直接上传会因为强制缩小的缘故导致图片内容展示的不够清晰，这是大多数社交平台用户都遇到过的一个场景。这时候用户就可以选择截取图片的重要部分之后进行上传，图像内容就可以更清晰地展示出来。出于用户体验和功能的灵活度考虑，在裁剪时还需要根据用户的操作实时地调整裁切大小，使得用户可以自由选择裁剪区域范围来适应各种不同的场景。

**3.1.3 添加文字**

给图片添加文字被广泛应用于社交平台、在线教育等领域中，这也是一个具备了良好用户体验的图片编辑器必备的功能。基于用户需求的多样性，在添加文字时需要让用户可以选择字体、大小、颜色等等属性，根据实际需求达到自己想要的处理效果。

**3.1.4 画笔**

画笔是所有的图片处理软件中非常重要的功能，在很多领域中被广泛应用，可以用于在线批改试卷、对图片进行艺术加工、自由绘画等操作。拿阅读电子书或者电子课件时的场景为例，在阅读时经常需要标记重点、做笔记等等操作。画笔功能在此类场景中非常的实用，可以利用调整画笔大小、颜色和线段类型等方式对内容和笔记按照重要程度进行细致的分类，以便强化学习记忆，非常有助于提升学习和工作效率。

**3.1.5 旋转**

在日常生活的照片拍摄、图片传输的过程中，很多人都遇到过图片需要倒转过来的情况，或者需要对图片在任意方向上旋转90度的情况。此时，如果不作调整，用户需要将设备或者头部调整过来才可以正常的阅览图片，这种情况会给用户造成很大的不便。因此设计一个图片旋转功能也是很有必要的，用户可以将发生偏转的图片上传到网站，调整到正常的角度之后进行阅览和下载。

**3.1.6 还原**

在用户进行图片编辑的过程中，可能会出现误操作或者对处理效果不满意的情况。这种情况下如果要求用户重新上传图片进行编辑无疑会大大降低用户满意度。因此设置一个图片还原功能是十分必要的。当用户对编辑效果不满意时可以一键还原图片并重新对其进行编辑操作，可以大幅优化系统的实用性、提升用户体验。

除了上面具体介绍的功能性需求，系统还需要具备图片的删除、下载和编辑之后的保存功能。这些功能对该系统的重要性不言而喻，一个健全的图片编辑器都需要具备这些功能才可以保证良好的用户体验。

## 3.2 非功能性需求分析

图片编辑器要满足良好的用户体验，除需要功能齐全和完备之外，非功能性需求也需要得到保障。因此在本节中将要对系统的非功能性需求进行详细的分析，从系统的易用性、可维护性、安全性和系统性能等角度出发进行详细的描述。

**3.2.1 系统可维护性**

系统的可维护性指的是系统的扩展性和系统出现故障后的可修复性。在本次的开发过程中，严格遵循了软件的开发流程，注重代码规范，采用当下最健全的技术来进行开发。并且在编写过程中始终关注模块之间耦合程度和逻辑的清晰程度，这样才能保障没有大的技术缺陷，有效保障了图片编辑器后期的可维护性。

**3.2.2 系统易用性**

简洁性和易用性是在该项目中的追求的最大特性，在构建页面时使用了特定组件库来尽量保证用户界面的整洁性。页面标签需要语义清晰，交互模式需要足够简单明了，这样用户才可以在较轻松的状态下完成各种图片编辑操作，没有经验或者经验不多的用户也可快速上手，对图片进行编辑。

**3.2.3 系统安全性**

在该系统中的安全性需求不是很高，图片的所有编辑操作都是在浏览器内部完成，在编辑阶段与后端数据库或者本地电脑资源的交互很少。如果涉及到处理一些个人照片，涉及到的个人隐私安全性则需要用户自己考虑，除此之外该系统对电脑不会产生任何损坏等情况。

**3.2.4 系统性能**

在提供各种图像处理的能力的同时，性能也是对用户体验至关重要的一部分。一般来讲，单个图片的编辑操作所耗费的性能资源不是很大，性能问题主要集中在需要加载和展示大量图片资源的首页中。达到快速的图片加载速度和保持页面滚动时不出现卡顿现象是需要重点处理的性能问题。因此在开发完成之后，对系统性能进行了严格的测试和优化，具体的过程会在第五章中详细描述。

## 3.3 本章小节

本章对线上图片编辑器项目做了详细的需求分析，从功能性需求和非功能性需求角度出发对系统需要具备的各种功能和其他特性做了详细的描述，明确了该项目需要达到的具体目标，为之后的系统设计和实现奠定了基础。

# 4 系统设计与实现

本章主要根据第三章的需求分析结果，对线上图片编辑器各个功能模块进行详细的设计与实现。主要包括：后端服务架构、数据库的设计、在浏览器中图片处理操作的具体实现流程。

## 4.1 系统设计

该项目基于B/S模式，分为浏览器端和服务器端。大部分编辑功能都放在浏览器端实现。服务器端负责图片的上传和存储工作。为了比较React、Angular和Vue等3种不同框架的性能，浏览器端需要使用这三种框架编写三套代码。

前端遵循组件化开发的思想，以组件的方式来构建整个页面，便于复用和维护。因为项目由多个页面构成，所以也需要在项目中集成前端路由。后端架构跟较多传统的后端架构方式不太一样，系统具体的架构如图4-1所示。

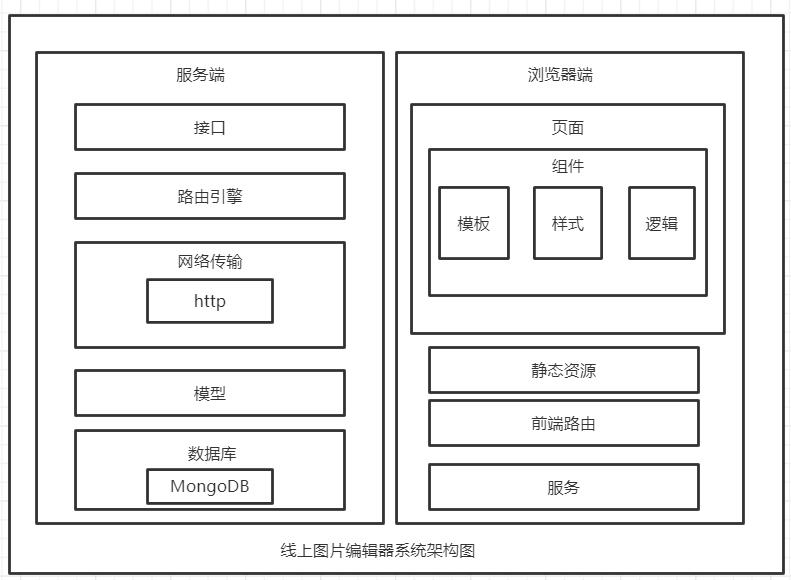


图4-1 系统架构图

该项目采用前后端分离的实现方式，前端注重于页面展示和实现编辑操作，后端则注重于提供接口和数据处理。前端分别采用Angular、React和Vue等框架，使用HTML、CSS、TypeScript等语言进行开发。前端的实现遵循组件化原则，一个组件包含了模板、样式和逻辑等部分，得以将整个页面切分成UI和逻辑上独立的块，降低整体项目结构的复杂性。在三种框架中都提供了非常丰富的组件通信方案。前端路由是前后端分离项目必备的模块，使得前后端的交互在不破坏用户体验的情况下异步完成。前端服务包含了网络服务和图片的编辑服务，这样做的目的是使得前端项目逻辑更加清晰、提升模块化程度、提升可维护性的同时在三个项目中实现代码的复用。

后端采用Node.js语言和Express框架开发，集成Compression、Cors等中间件来提供服务，并使用MongoDB作为数据库。模型是操作MongoDB数据库主要途径，一个模型对应了一个集合，要对集合中的数据进行任何操作都需要调用对应的模型上的API来实现。网络传输是一个Web服务端必备的，也是核心的模块，前后端的交互依赖于该模块。路由引擎给后端提供了集中式管理接口的能力，所有的接口都需要通过路由引擎来挂载到Express路由器对象上。接口是后端所有模块的服务对象，是后端向前端提供服务的唯一途径。

**4.1.1 数据库设计**

本次开发中数据库采用MongoDB实现，MongoDB是一个面向文档存储的NoSQL数据库，数据以文档的形式保存。文档由一个个的JSON字符串组成，阅读起来方便直观。Js、Java、Python等编程语言中都具备了很好的JSON数据支持。数据集合在JSON字符串中以键值对的形式存在，可以是任何数据类型，完全可以处理复杂对象的操作。具体的图片数据集合如表4-1所示。

表4-1 数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 属性 | 描述 |
| \_id | String | 主键 | 图片id |
| Image | String | 非空 | 图片地址 |
| Name | String | 非空 | 图片名称 |
| Description | String | 可为空 | 图片描述 |
| Tags | String | 可为空 | 图片标签 |

集合中的图片总共有5中属性，分别是图片的URL地址、图片的名字、描述以及标签。\_id属性由数据库自动生成，用于唯一标识一组数据，类似于关系型数据库中的主键。每张图片在集合中都已表4-1中的对象结构的形式保存下来，之后通过模型对其进行增删改查操作。

**4.1.2 服务端架构的具体设计**

如图3-1所示，服务端由数据库、模型、网络传输、路由引擎和接口等五大模块组成。接口和后端功能一一对应，如图4-2所示。

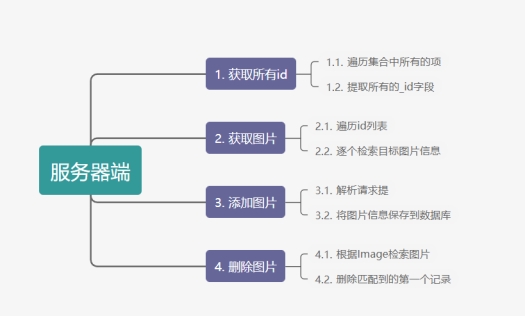


 图4-2 后端接口

后端实现了四个接口供前端调用，分别用于用户对图片的获取，添加和删除等操作。接口是前后端通信的唯一途径，后端架构中的所有模块都为接口服务，因此接口设计的合理性非常重要，关系到整个系统的运行效率。

路由引擎负责接口的挂载，集中式管理接口等工作。路由引擎提供了编写接口的统一思想，使得后端的接口和功能都具有了高度的可扩展性和可维护性。体现在于任何的功能或者接口的增加只需要新创建一个文件，编写接口逻辑之后在路由引擎中导入即可，不需要对整个系统的架构做出任何改变，接口的更新、添加或者是删除操作都被限制在了路由引擎模块内部。

网络传输是一个Web服务器必备的功能，是服务器正常工作的前提。Express框架提供了网络传输和监听端口服务，用于快速搭建服务器。

模型是Mongoose中操作数据库的渠道。所有的数据操作都需要该模型来与数据库交互，因此也是服务端不可缺少的一部分。模型中定义了模板用于限制数据格式，每个模型都会映射到一个单独的集合中。

服务端使用Express框架实现，所有的接口最终都会被路由引擎挂载到Express路由对象当中，这时才可以接受浏览器发送的HTTP请求。接口接受收到HTTP请求之后调用定义好的数据库模型对数据库进行增删改查的操作，并将结果发送到浏览器。

**4.1.3 前端架构的具体设计**

前端项目总共由首页和图片编辑页两个页面组成。在需求分析结果中预期的主要功能有图片的上传、裁剪、添加文字、画笔、旋转、还原、保存以及图片的下载和删除等。整体分为交互模块、属性模块和编辑模块，具体的模块划分内容如图4-3所示。

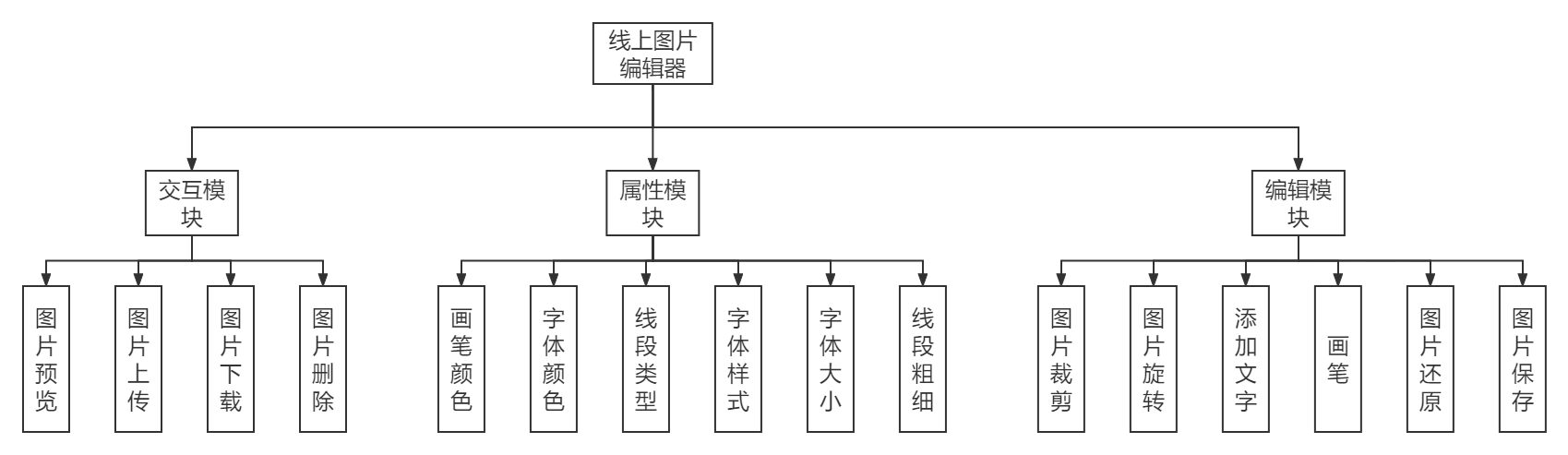


图4-3 前端功能模块

项目结构根据所基于的框架的不同各有差别。大体上的结构由页面、组件、网络服务、工具以及静态文件组成。页面由多个组件构成，图片的展示卡片、保存弹窗等都可以封装成单个独立的组件，每个组件都包含独有的模板，样式和逻辑等，以此来提高项目的模块化程度，便于项目的维护和功能的扩展。构建页面所需的各类图片，图标字体等静态资源也需要在单独的模块中集中式管理。首页和编辑页面之间的跳转还要求在项目中集成前端路由，前端路由可以使浏览器在不刷新页面的情况下跳转页面，页面所需的数据可以在页面加载完成之后利用Ajax异步的请求过来，因此用户不需要等待页面的跳转，点击的同时便可立即跳转到编辑页面。

前端的服务模块包含了网络服务和图片的编辑服务。开发者通过将各种功能抽象成服务从而提高项目的可维护性、复用性和扩展性。因为该系统要使用三种框架编写三次，所以在编写前端代码时需要格外注重代码的可复用性。虽然各个框架的代码组织方式和模板语法都不一样，但是一些特定的功能还是可以细致地提取到各种独立的工具函数里面。所以编写前端代码时，需要尽可能地将图片编辑的实现逻辑提取出来。这是一项很重要的工作。

当用户进入网站之后，首先会在首页向后端请求所有的图片信息，拿到数据之后会渐进式地在页面中渲染。用户点击侧边栏上的导入按钮会弹出图片上传弹框。用户点击上传按钮在资源管理器中选择图片之后会在弹框中展示图片预览，输入图片描述和图片标签并点击确认按钮就可以将图片发送到数据库保存起来。

## 4.2 系统实现

需求分析和系统架构设计完成之后就可以着手开始系统的具体实现，本节介绍整个系统的前后端具体实现方式。

**4.2.1 服务端实现**

根据前面的需求分析描述和架构设计，服务端是基于Express的搭建的项目，使用Node.js进行开发。下面详细讲述服务端各个模块的实现过程。

1. 接口的实现

本次项目中接口的编写形式采用类写法，每个接口都是一个单独的类，带有自己的私有和公共属性与方法。这种编写方式虽然比直接导出Express路由模块，比在该路由模块中直接挂载和定义接口稍微复杂一些，但是可以让代码变得更加规整、逻辑聚合、提高模块化程度。为了规范化不同接口的编写方式，可以使用TypeScript定义一个接口，TypeScript中的接口和服务器中定义的网络接口具有很大不同。TypeScript中的接口类似于Java、C++等面向对象的编程语言中的接口。通过该接口可以强制规定各个网络接口类必须实现的方法和功能，比如必须编写一个添加路由方法供路由引擎调用。后在每个类中实施该TypeScript接口，编写接口规定的添加路由方法。编写过程中遇到复杂的计算和逻辑时，将网络接口编写成类的做法就可以体现它的优越之处，因为可以将这部分复杂的逻辑提取出来，封装到该类的一个私有方法之中。除此之外这也给每个接口提供了维护自己独有状态的能力，使得接口功能更加强大和灵活。

1. 路由引擎的实现

路由引擎负责集中式管理所有接口，包括将接口类实例化、将其挂载到Express服务器等。在接口的实现过程中讲到，每个接口类必须提供一个添加路由的方法供路由引擎调用，路由引擎就是通过实例化接口类，并调用其添加路由方法实现路由的挂载。除此之外路由引擎还使用一个数组类型的私有属性来维护整个服务器中所有的接口类。

1. 数据库模型的实现

接口进行数据处理都需要依赖于模型，在单独的文件中编写一个数据库类，将数据库的链接函数作为公共方法导出，对连接时的状态和出现的错误等情况进行日志打印。同时在同一个文件中建立图片的数据模型，并导出Mongoose模板。之后在接口文件中引入，就可以在接口类对数据库进行操作。

1. 服务器类的实现

所有的主要模块实现完毕之后，就需要编写具体的服务器类用于启动服务。具体做法是将Express服务器、数据库类的实例和路由引擎的实例作为私有属性传递到服务器类中。之后在该类的启动方法中使用作为私有属性传递进来的各类实例进行连接数据库、启动路由引擎、配置中间件等操作。还需要将服务器的一些可选功能的配置代码封装到独立的方法中，便与在实例化时对服务器进行一些可选的配置，比如跨域、gzip压缩等功能。

此外，为了提高代码的复用性，服务器类需要编写成一个抽象类。并且将添加路由方法作为被保护的抽象方法交给子类实现。使得服务器类在别的项目中可以被复用，之后在具体的子类中添加自己的路由。

服务器基本功能实现完毕，服务器UML类图如图4-4所示。

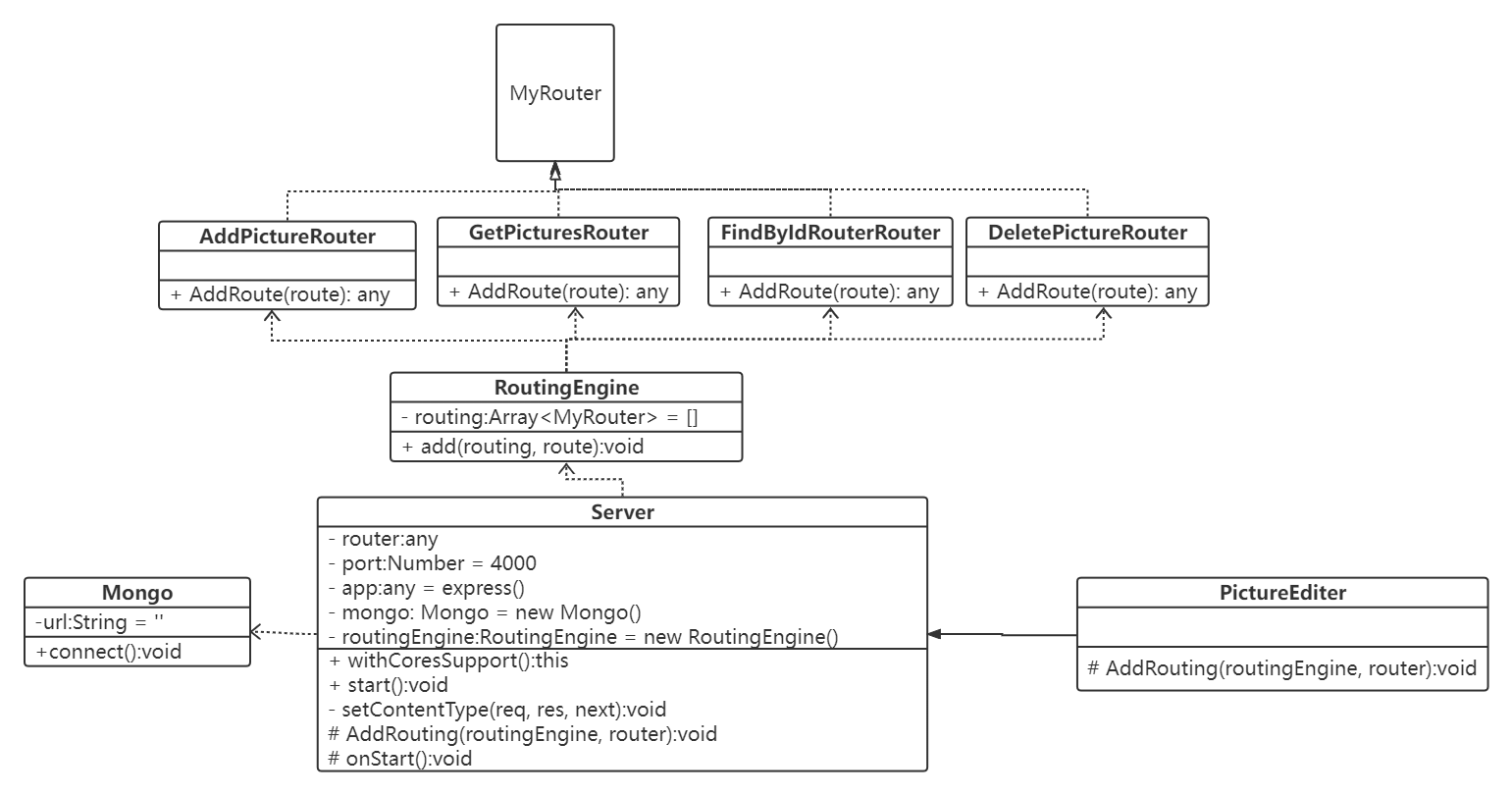


图4-4 服务端类图

如图4-4所示，在Server类中完成了基本的工作，比如配置跨域、解析请求体、连接数据库、启动服务器等等。应该注意的一点是，配置跨域的过程单独封装在了独立的方法中，通过此方法可以在实例化服务器时自由选择服务器是否接受跨域请求。因为并不是所有继承Server类的服务器都需要进行跨域通信，有时跨域请求会带来安全隐患。抽象函数AddRouting用于使用路由引擎挂载路由，需要子类来具体实现。onStart方法则负责监听服务器的启动事件，执行定制的副作用。

子类PictureEditer中实现了父类的抽象方法AddRouting，并添加了该项目所需的四个路由。在最后实例化子类，调用withCorsSupport方法配置接受跨域请求，最后调用start方法初始化并启动服务器。至此，我们实现了服务器端功能。

**4.2.2 前端实现**

该项目由首页和图片编辑页两个页面构成。首页负责展示图片、上传、下载和删除等功能。出于美观性和实用性的考虑，两个页面都是基于页头、侧边栏和内容区域的页面布局方式。

项目的难点在于实现图片的编辑。经过思考与收集资料，决定利用HTML5新推出的canvas元素完成这些功能。canvas俗称画布，给前端开发人员提供了在浏览器上绘制图形、文字的能力。通过在代码中保存编辑状态，在canvas元素上监听鼠标和键盘事件，可以合理地设计与用户的交互，完成各种有趣的图片编辑功能。

1. 上传

图片上传是用户进行图片编辑的前提，该功能可以结合HTML5的file标签和JavaScript的FileReader对象来完成。用户点击首页侧边栏上的上传按钮，在弹出来的弹框里点击按钮选择本地图片并确认，就可以在网页中预览将要上传的图片，整个上传过程的时序图如图4-5所示。除此之外，需要在图片的上传函数中对图片格式和图片大小加以限制。本项目中，上传图片的类型被限制为了PNG或者JPG格式，图片大小则是被限制为了2M，若不满足该限制，网页会以小弹窗的形式给出提示。

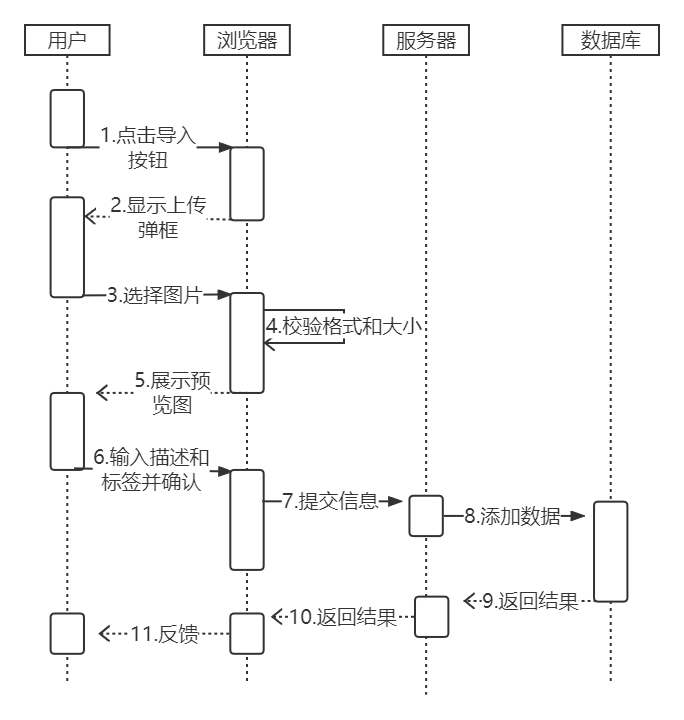


图4-5 图片上传时序图

因为图片的上传是异步操作，所以上传函数会返回一个Promise对象。Promise是js中处理异步操作的方式。将异步操作封装到Promise之后，就可以在then方法内部按顺序的拿到结果。此时，需要在Promise中读取图片并将图片的地址和描述等信息封装到图片模型中，并将其返回给调用者。图片的上传完成，在用户点击确认之后，就可以将图片发送到服务器，并根据结果给出反馈信息。

1. 图片编辑的基础工作

在开始实现任何编辑功能之前，需要先结合框架特性构思好整个基础框架，假设实现流程以及可能出现的问题。

在需要实现的几个功能中，有几处需要在编辑过程中以动画的形式给用户反馈。在canvas中的动画制作需要不停的清除和重绘，因为图片的加载需要一定的时间，所以这很有可能会导致图片在动画过程出现闪烁。该问题的解决方案是将两个canvas元素重叠。上方canvas作为封面透明的展示在主要画板canvas之上，当需要动画时将其呈现到封面canvas中，主画板不需要进行任何操作，图片闪烁的问题就得到了解决，如图4-6所示。

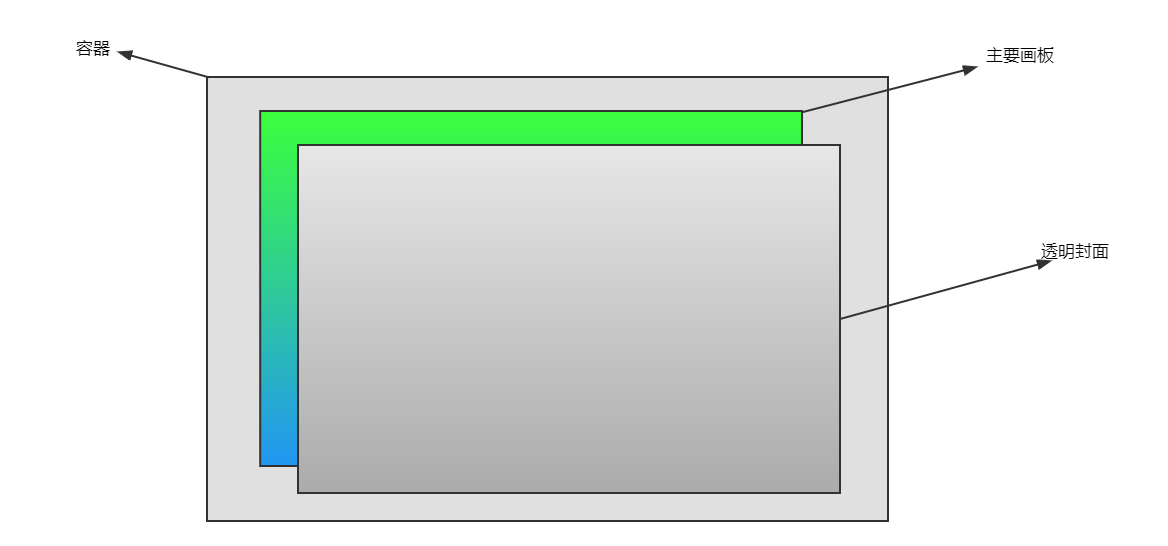


图4-6 画板层次结构

编辑属性需要特定的状态变量来保存，无论是React、Angular或者Vue都可以定义响应式变量来管理这些状态。除此之外，为了方便在以后再次拓展编辑器的功能，除了更新实时状态，还需要保存和维护画板的所有历史状态。因此需要在每一种编辑操作完成之后，要及时地更新实时状态和历史状态。历史状态是一个栈结构，它不需要是响应式的，因为视图只跟实时状态有关。

1. 图片裁剪

所有的编辑操作都需要调用Canvas特定的API，但是在每个操作函数内部都去重复调用Canvas的方法不仅会降低代码的可维护性，而且还会增加重复代码增加项目体积，从而影响性能表现。因此将所有的canvas操作都封装到单独的文件和单独的类中、之后在各项编辑功能的实现过程中都去引用这个类是比较好的做法。

用户进行裁剪的正常流程是：点击裁剪按钮、画布蒙上阴影、开始裁剪、标记出裁剪范围、停止裁剪。就像之前讲到的，在用户放开鼠标停止裁剪之前的标志裁剪范围的操作是在上层的封面canvas中完成的，否则重复的渲染和清除会造成图片的闪烁问题。

在实现裁剪的过程中需要考虑的一个问题是裁剪方向。因为canvas的裁剪API用到的参数是裁剪的起始位置和宽高，直接用鼠标开始裁剪的起始位置和结束位置来调用是不行的。因为裁剪方向可以有四种，分别是从左上到右下、右上到左下、右下到左上、左下到右上。要正确调用canvas的裁剪API，就需要判断用户的裁剪方向对并对起始坐标和结束坐标做出适当的调整。之后便可以使用调整完成后的坐标进行裁剪。图4-7所示是裁剪处理时序图。

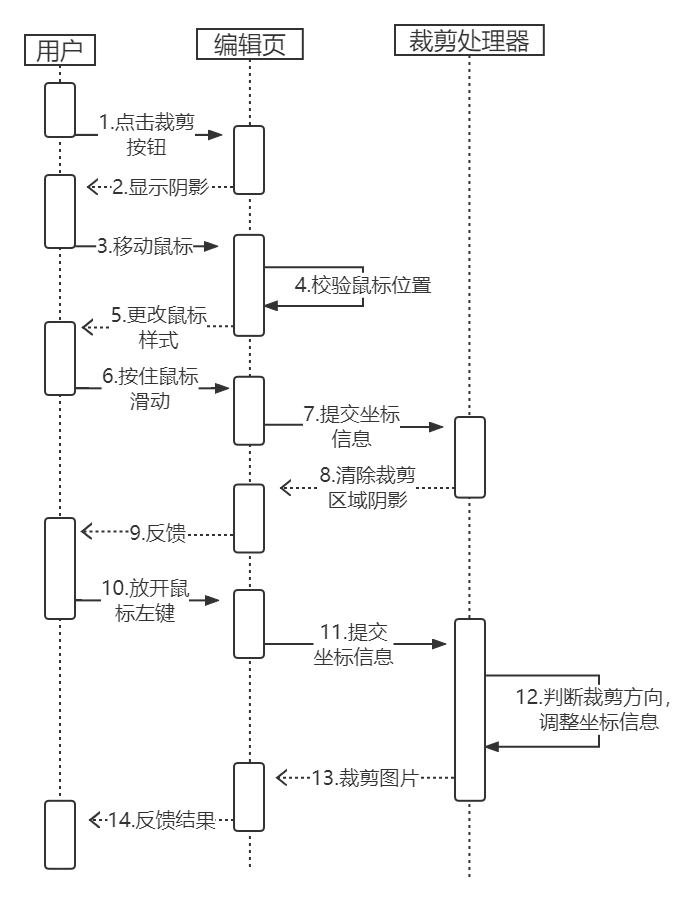


图4-7 裁剪处理时序图

1. 添加文字

canvas也提供了绘制文字功能，该功能主要难点在于与用户的交互形式，该项目中实现的最终交互模式为：点击文字按钮、点击画板、点击位置上显示输入框、输入文字、按下回车键、添加文字。添加文字操作具体流程图如图4-8所示。

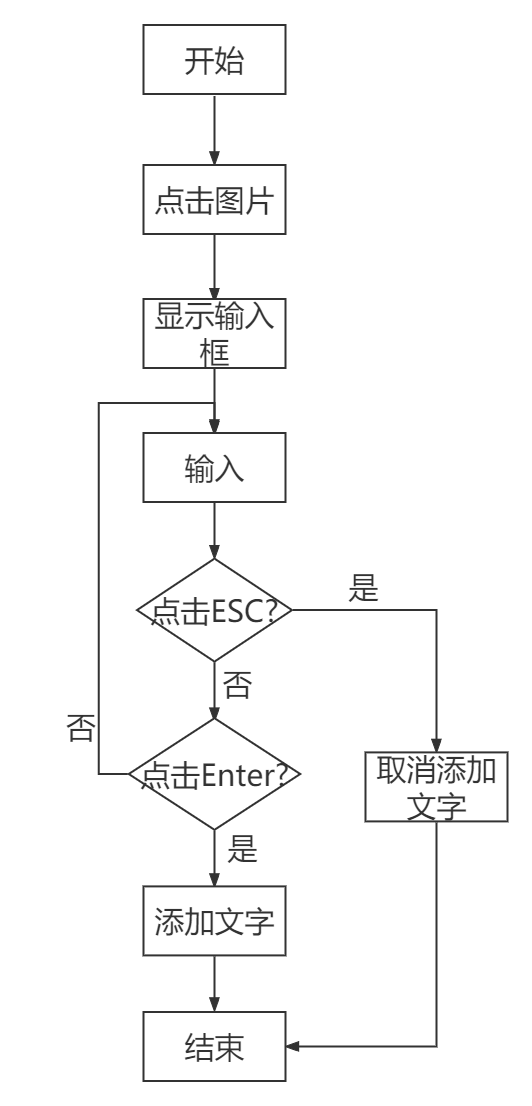


图4-8 添加文字流程图

该过程中需要解决的一个问题是输入框的显示隐藏和坐标处理。在三种框架中DOM元素的显示都可以使用响应式变量来控制，而坐标则需要根据容器的定位和鼠标点击位置来确定。具体的做法是对包裹canvas和输入框的容器开启相对定位，对输入框开启绝对定位，这样鼠标坐标和输入框相对与canvas画板的坐标就会保持一致。

1. 画笔

画笔功能需要注意的点是，画笔需要一系列点的坐标。当鼠标移动时需要快速的确定线段的起始坐标和结束坐标。解决的方案是在画笔类的类的私有属性上维护一个坐标值，当函数每一次结束调用时，将结束时的点坐标保存下来，并在下一次调用开始时将起始坐标移动到上一次的结束坐标再开始画线。这样无数个小段的直线组合到一起可以画出任何形状，就像一支笔一样随便涂鸦。然后在用户放开鼠标时将起始坐标调整为0，再清除监听器即可。整个流程时序图如图4-9所示。

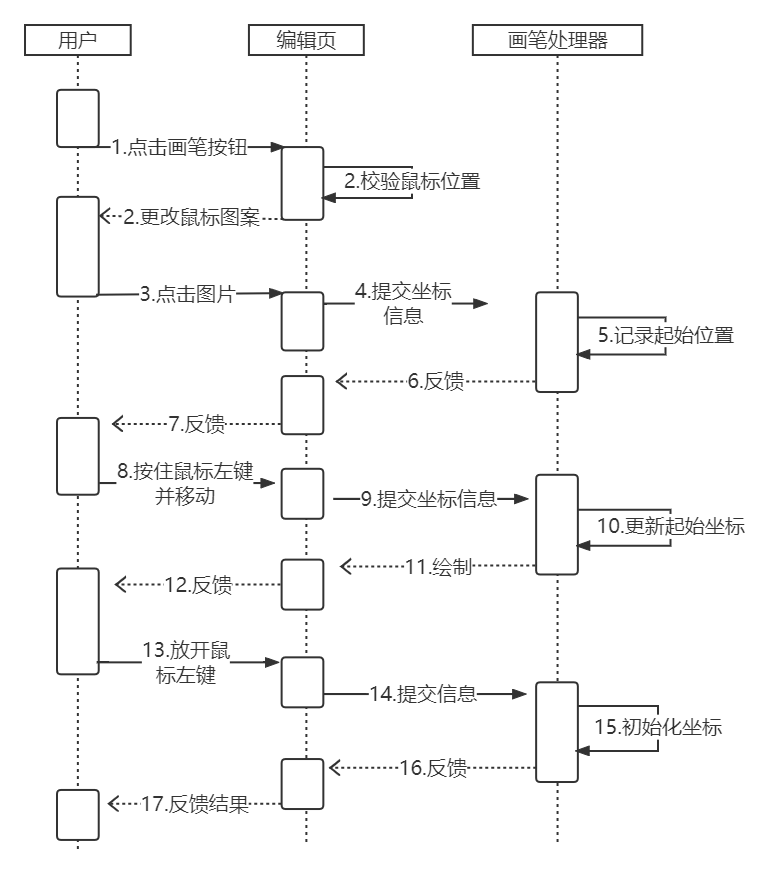


图4-9 画笔时序图

1. 旋转

图片的旋转功能主要依赖于canvas的变换坐标原点的功能，实现原理是当用户点击旋转按钮时，先把坐标原点调整到对应的四个顶点中的一个，具体的顶点需要根据旋转类中维护的坐标信息来确定，之后再将图片在新的坐标系下重新绘制一遍。

旋转图片的难处在于再裁剪完成之后再次旋转会出现由于canvas的宽高不均造成的图片的部分缺失或者重叠情况。解决方案是在每次旋转之后都交换一次canvas画布的宽和高的值。但是坐标变换带来的又一个问题是，当旋转完成一定的角度之后，因为原点坐标的变换，再次去添加文字或者使用画笔、裁剪图片，都会出现用户点击和移动鼠标之后的坐标紊乱的情况，使得整个系统完全不可用。对此的解决办法是在每次旋转完成之后都迅速将左边原点恢复到初始位置，也就是左上角。幸运的是canvas提供了特定的API完成这个操作。分别是canvas上下文对象的save方法和restore方法，save 方法的作用就是将画布当前的状态推入一个栈保存起来，然后在调用restore方法时推出栈顶的项，返回到上次调用save方法时的状态。具体的解决方法时在旋转函数开始执行时调用一次save方法，之后在变换坐标并且绘画完成之后，再次调用了restore方法来将坐标系恢复到原点，该问题就得到了解决。

1. 属性的状态

所有属性的状态，例如颜色、字体、大小等等，都有对应的响应式变量来维护，由页面中的表单控件来控制。当用户通过表单改变属性时，更新之后的属性值会以参数的形式来传递给canvas类的各个方法，以此来实现在不同状态下的编辑。

所有的编辑功能已实现完成，以裁剪功能类图为例对所有的编辑操作进行简单的梳理，如图4-10所示。

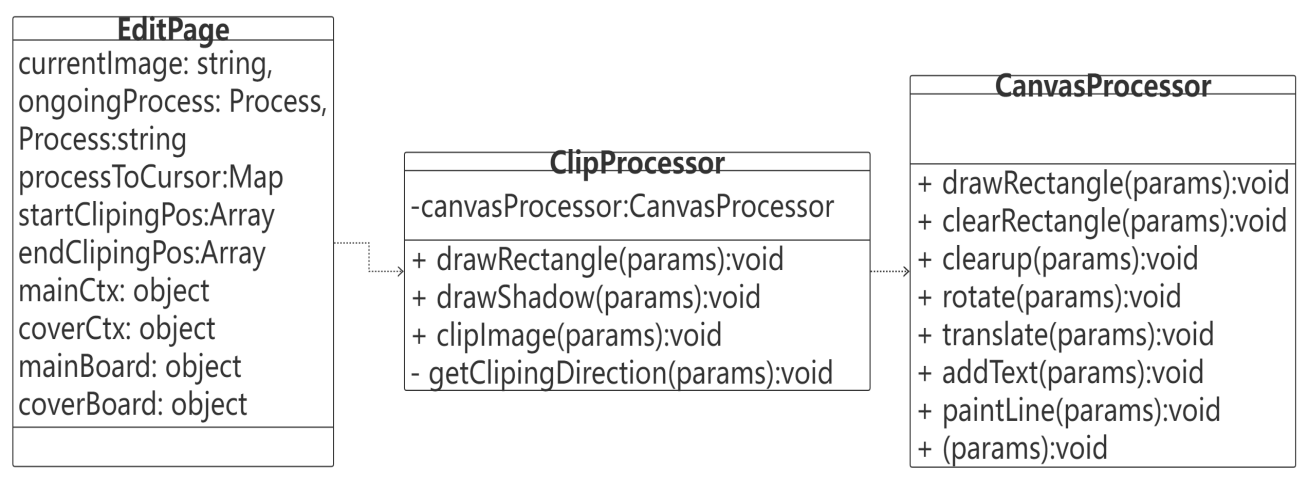


图4-10 编辑功能类图

上图中可以看到，所有的编辑类都依赖了CanvasProcessor，它是直接用于对canvas画布进行绘画操作的类，在编辑页面不会直接引用这个类，而是会通过某种具体的编辑操作对应类来操作canvas。以ClipProcessor举例，它负责处理添加裁剪功能，它会在对编辑页面传递进来的参数做适当的处理之后再调用CanvasProcessor来向进行图片的裁剪。这么做的原因是可以实现逻辑的松耦合、减少重复的代码、并且易于维护。因为所有的canvas操作都聚集到了单独的模块中，之后在单独的编辑类中加以引用提高了模块化程度、所有的操作逻辑边缘清晰、互不越界。

## 4.3 系统效果

美观简洁的UI界面也是用户体验的关键点之一。因此构建页面分别用到了angular/material、atn-design等等UI库。因为编写页面UI是在Web页面开发中很重要而且比较繁琐的一部分，选择一个好的UI库可以减少工作量，能够快速的搭建出简洁、美观的用户界面。图4-11是React图片编辑器的首页效果。

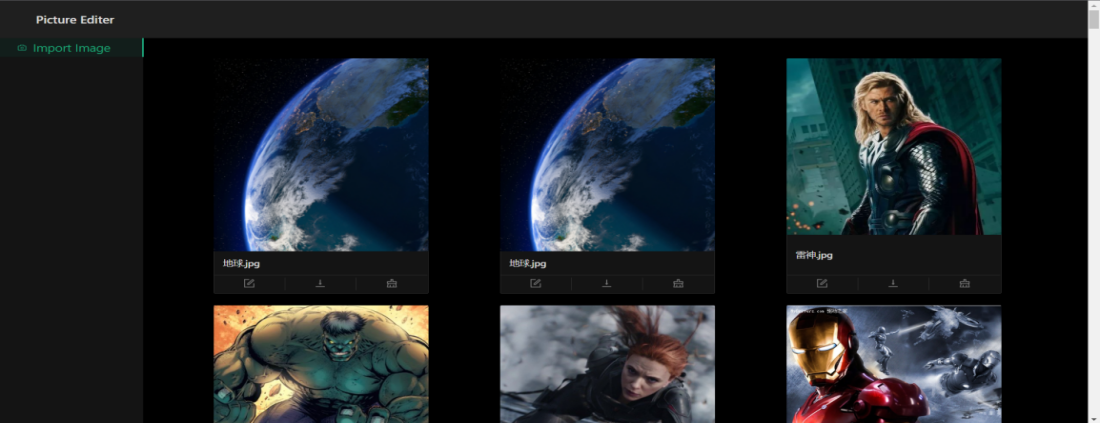


图4-11 React图片编辑器首页

如上图所示，每张图片展示卡片下方都有三个按钮，分别对应编辑、下载和删除。点击页面侧边栏上的Import Image会弹出图片上传框，即可在上传框里预览图片、输入描述和标签、进行上传。上传过程如图4-12所示。

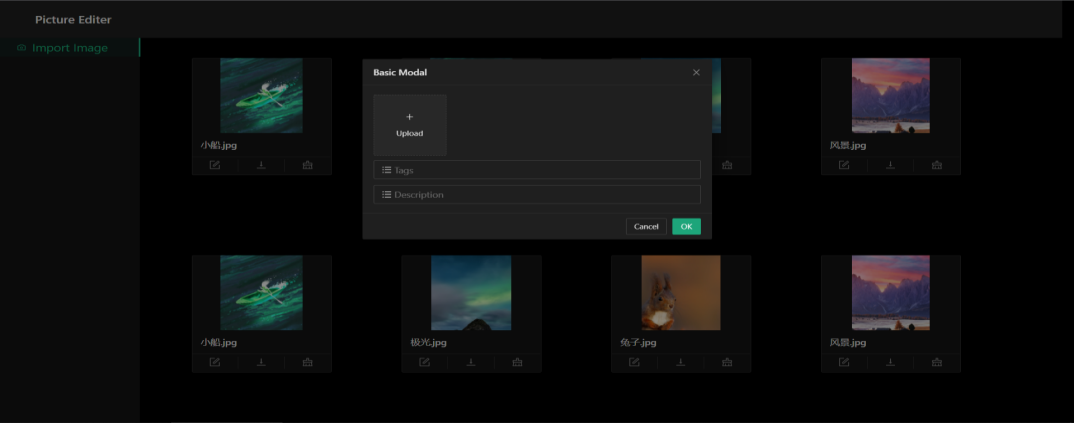


图4-12 React图片编辑器上传图片

如上图所示，点击左侧侧边栏点的按钮之后，就会弹出图4-12所示的上传弹框，点击upload按钮之后即可打开本地资源管理器，选中某个图片后，系统会判断上传的文件是不是PNG或者JPG格式，如果不是则会向用户反馈上传文件不符合格式。如果上传的图片的大小超过了2MB也会给予用户对应的提示。都符合要求即可在upload按钮上预览上传的图片。在下方的两个输入框中分别选择性地输入图片描述和图片标签之后点击确定，即可将图片发送到服务器中保存起来，整个上传操作便完成了。

在点击某个图片下方的编辑按钮之后，即可跳转到对应的编辑页面，编辑页面效果如图4-13所示。

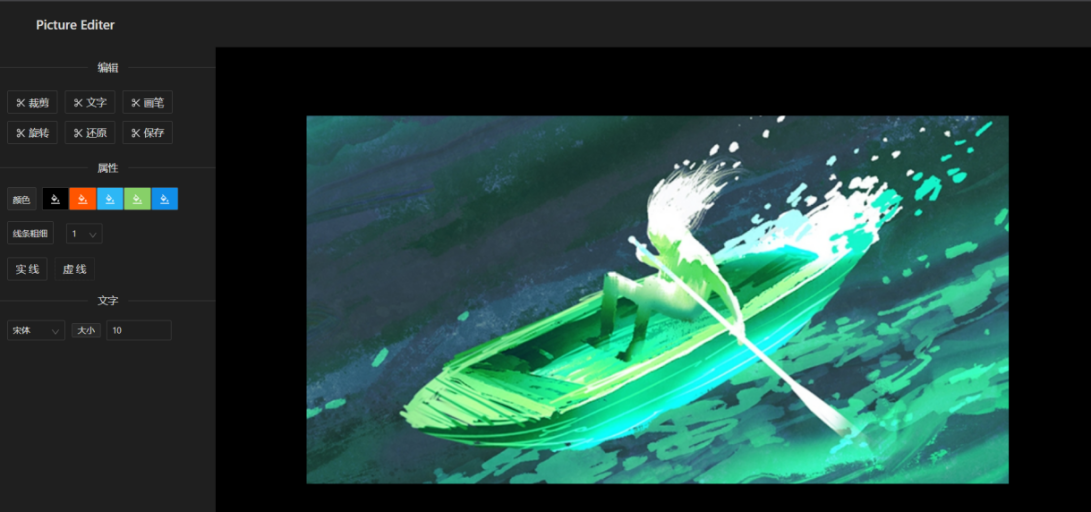


图4-13 React图片编辑器编辑页面

上图可以看到，编辑页面由图片展示区和侧边栏构成。侧边栏包含编辑操作区域、属性区域和文字区域等三个部分。编辑栏是是用户对图片进行的操作的主要活动区域，所有的编辑功能都集中在这一栏。点击裁剪按钮之后图片上方会蒙上一层阴影，当鼠标移动到图片上方时鼠标图案会变成累次瞄准镜的图案用于提示用户正在进行的是裁剪操作，按住鼠标左键在图片上方滑动会实时清理掉画出四边形上方的阴影，使得用户可以更好地控制裁剪效果，放开鼠标左键即可完成裁剪，如图4-14所示。

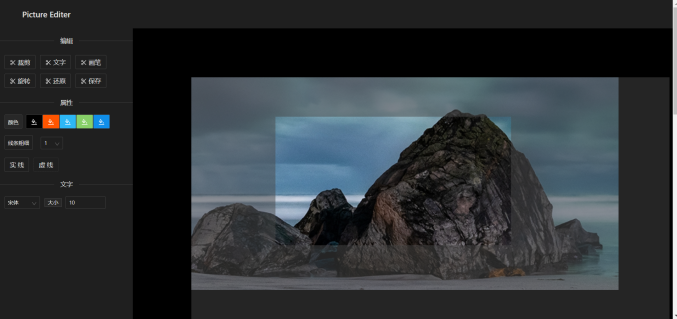


图4-14 React图片编辑器对图片进行裁剪

在用户点击添加文字时，鼠标在图片上方会变成一道竖线来提示用户正在进行添加文字操作，点击图片会在鼠标位置显示一个输入框，输入想要添加的文字按下回车键即可将文字绘制到图片地对应位置中，如果用户想终止操作，按下键盘的退出按键即可结束该操作。画笔操作也很方便，用户只需要点击画笔按钮之后，在图片上方按住鼠标左键开始绘制即可。旋转不需要任何额外的操作，按下旋转按钮之后图片便会向右旋转90度。如果用户对编辑效果不满意，点击还原按钮，图片便会恢复到开始编辑前的状态，用户可以重新开始编辑。

属性栏可以选择画笔和文字颜色，提供了五种颜色作为备选。线条粗细可以改变画笔线段，用户可以在点击下拉选择框之后对其进行调整。除此之外也可以选择线段类型为实线还是虚线。最下方的文字栏可以选择字体和大小，提供了大部分浏览器支持的11种字体供用户选择。



图4-15 处理之前的原图



图4-16 处理之后的效果

如图4-15和4-16所示分别是一张图片在使用图片编辑器进行处理前后的对比。在图4-16中对图片进行了裁剪、旋转、添加文字、使用画笔绘制等操作，并且在操作过程中改变了字体，颜色，大小，线段类型等等属性。可以看出系统运行正常，实现了良好的编辑效果。

**5 性能测试与优化**

## 5.1 测试工具

性能测试工具选择由Google发起的 Web Vitals。Web Vitals 提供了对用户体验影响最大的几个指标。可以在每次进入页面时打印出性能指标详细的报告，并指示出页面中对单一指标影响最大的模块。在使用脚手架创建的React项目中从新版本开始已经默认包含了Web Vitals 工具，其他两种框架则需要手动配置。配置的第一步是从npm包管理器中将Web Vitals 下载到项目中，之后在项目的入口文件中导入，然后以控制台输出语句为参数分别调用各种指标的获取函数即可。

## 5.2性能指标

**5.2.1 LCP**

LCP（Largest Contentful Paint）中文名称最大内容绘制。LCP是测量网站用户体验的重要指标之一。该项指标会在页面的主要内容加载完成时标记出页面中的最大模块，并给出加载该模块所耗用的加载时间。网页的最大模块展示事件时间无疑是用户最容易感知到的部分。只有尽可能快速地呈现出网页的主要模块，才可以尽可能地留住用户，而不是让用户在漫长的页面加载等待时间后退出页面，去打开竞争对手的网站。

页面中的最大元素的大小一般是在屏幕可视区域内的大小，如果一个元素溢出屏幕延伸到了用户可视区域之外，这一部分不会被计入LCP指标的计算中。除此之外，图片在网页中的大小通常是经过了一定程度上的调整的，在计算LCP时通常以调整前后的照片尺寸较小者为准。

Google建议网页的最大内容绘制事件应该在2.5秒内完成，在2.5秒到4秒的区间内属于待优化，大于4秒会严重影响用户体验，使产品失去竞争力。

**5.2.2 CLS**

CLS（Cumulative Layout Shift）中文名称累计布局偏移。CLS是测量视觉稳定性的一个以用户为中心的重要指标。很多人都经历过在网页中进行阅读、或者正准备要点击某个按钮时网页的布局瞬间发生偏移的情况，这时我们要重新寻找内容或者已经错误点击了别的按钮。页面内容的移动通常是由于资源的异步加载或者动态地对现有页面上方添加DOM元素所造成的。布局偏移的计算是通过偏移元素占据可视区域的大小百分比和移动距离占据可视区域的大小百分比相乘而来。

为了保证良好的用户体验，CLS指数应该保持在0.1之下，如果超过了0.25就可以视为已经严重影响了用户体验。

**5.2.3 FCP**

FCP（First Contentful Paint）中文名称首次内容绘制。FCP标志出了页面从加载到渲染出第一个模块所耗费的事件。与LCP不同，FCP不关心模块的大小，优秀的FCP指标有助于及时给予用户反馈，让用户知道页面正在努力加载。

为了提供良好的体验FCP指标应该保持在1.8秒以下，如果超出3.0秒则被认为影响了用户体验。

**5.2.4 TTFB**

TTFB（Time to First Byte）中文名称首字节事件。TTFB是用于测量Web服务器响应的基本指标，有助于识别Web服务器相应太慢而影响用户体验的情况。统计时间包括重定向、DNS查询、握手和建立链接到第一个字节到达浏览器等过程所耗费的时间。

基于网络和应程序的差异和复杂性，并不能对TTFB提出具体的数字标准。Web Vitals 建议TTFB的速度可以让75%的用户体验到优秀的FCP。

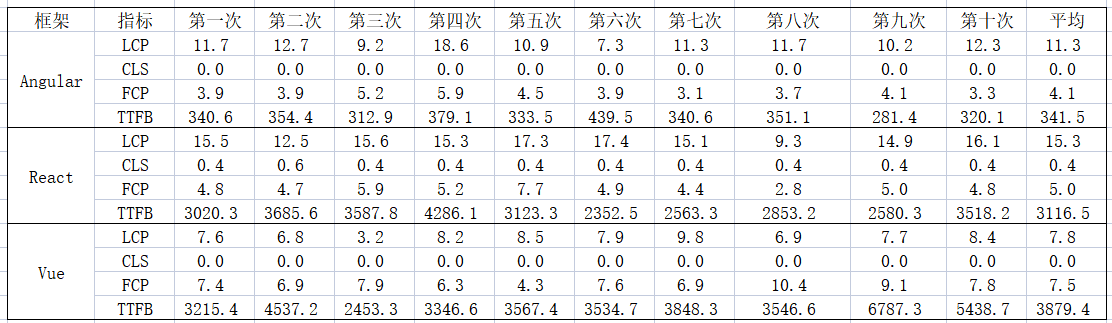
## 5.3 性能测试

性能测试第一步是将测试工具集成到项目中，用React脚手架创建的项目已经默认包含了Web Vitals 。在Angular和Vue中需要手动安装和配置。配置完成之后便可以在网页加载时在浏览器控制台看到具体的测试结果。测试环境如表5-1所示。

表5-1 测试环境

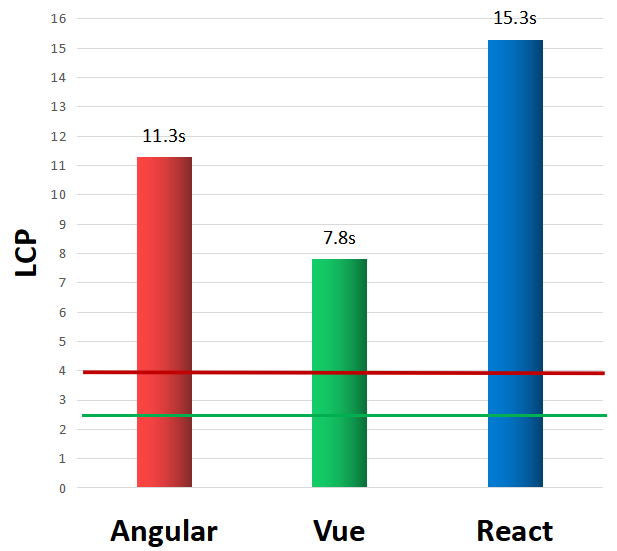
|  |  |
| --- | --- |
| 测试环境 | Chrome浏览器 |
| 操作系统 | Windows10 |
| 系统类型 | 64 位操作系统 |
| 处理器 | AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz |
| 运行内存 | 8.00 GB |

在进行优化之前，在三个项目中分别进行10次测试，并记录下每项指标数据。数据收集完成之后，每项数据去除两个极值再求平均。统计数据如图5-1所示。



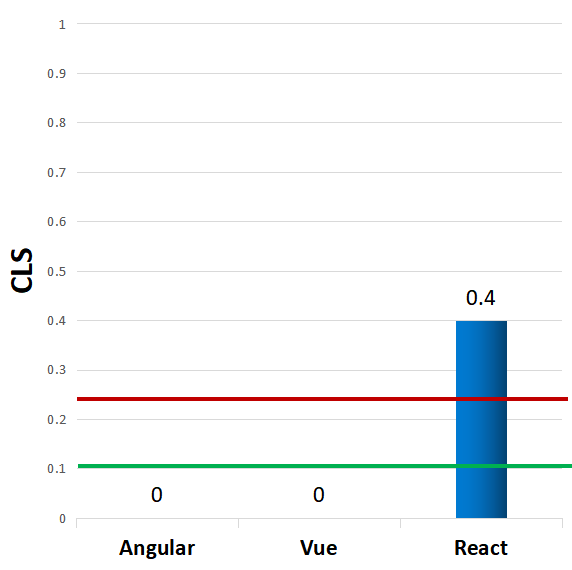
**图**5-1 **性能统计数据**

统计完成，为了能够更直观地体现三个项目在优化之前的性能表现，以柱状图的形式来呈现各个指标的平均数据。LCP指标结果如图5-2所示。



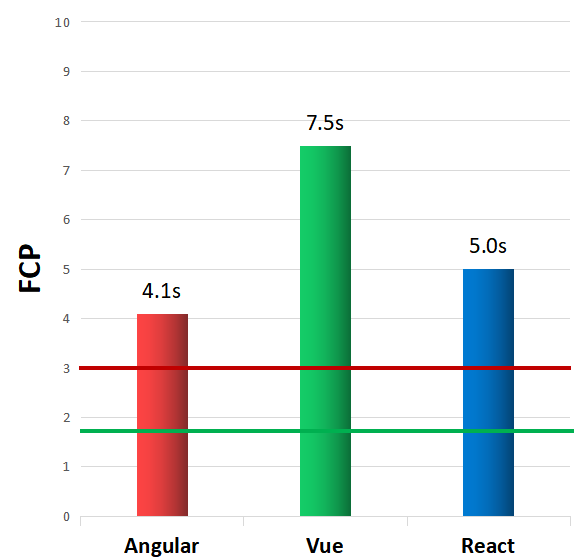
**图**5-2 LCP**指标优化前**

图5-2中，绿色横线代表LCP指标的优秀线2.5秒，红色横线代表该指标的最高限度4秒。图中可以清晰地看到三个项目的LCP指数已经严重超标。Angular，Vue和React的最大内容绘制时间分别在11.3秒、7.8秒和15.3秒，意味着用户需要等待这么长的时间才可以看到页面中的最大模块，这对一个合格的产品来讲是不可接受的，需要进行大幅度的优化。



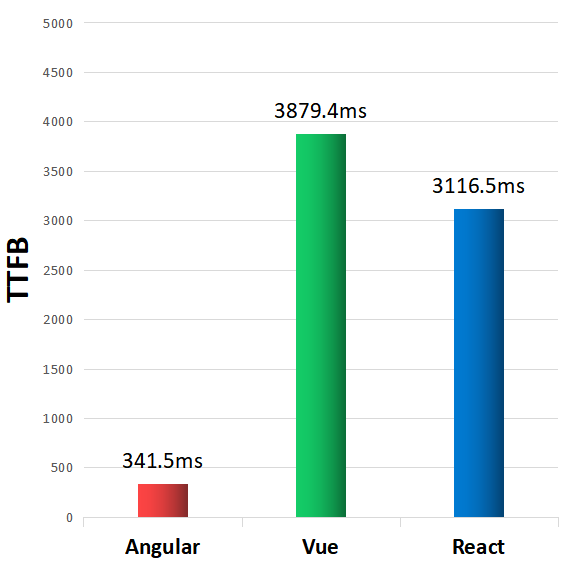
**图**5-3 CLS**指标优化前**

图5-3是CLS指标的测试结果，可以看到Angular和Vue项目中页面布局并未发生偏移，CLS指数均为0，是一个很理想的结果，但是在React中该项指标超过了0.25，表明页面元素在渲染完成之后出现了明显的布局偏移，因此需要对React图片编辑器进行相应的优化。



**图**5-4 FCP**指标优化前**

图5-4是FCP指标在优化前的性能测试结果，可以看到在该项指标三个项目均超标严重。鉴于FCP是统计首次内容绘制时间，超标原因可能是因为过多的网络请求导致的缓慢的加载速度，后续可以基于该出发点进行优化。



**图**5-5 TTFB**指标优化前**

图5-5是TTFB指标在优化前的测试结果，TTFB虽然没有像其他三个指标那样有具体的量化评价区间，但是它可以反应出其他指标的测试结果。基于其他指标的测试结果和Vue和React中3秒多的首字节到达时间，我们可以判断TTFB指标的测试结果也并不理想。TTFB主要反映的是网络层面的问题，量化前后端交互的效率，优化时可以从这一点出发进行改善。

## 5.4 性能优化

以上是优化前的性能测试和数据统计。接下来我们开始对三个项目分别进行优化。该项目最大的性能消耗点在于图片资源过多导致的网络请求负载过大以及首屏渲染时间过长等问题。因此优化工作也应该从这几点入手。

**5.4.1 gzip压缩**

大量的图片从后端请求过来会造成巨大的网络负载，减缓此情况的一个有效途径是对请求数据进行gzip压缩处理。gzip是http协议中的一种文件压缩算法，当客户端向应用程序发出请求后，应用程序一般会将页面文件或者其他资源返回给客户端，客户端加载后渲染呈现。如果在服务器端开启gzip，并且客户端浏览器支持gzip压缩，那么服务器端响应之前，会将一些文本文件或者其他文件通过压缩算法来进行压缩。传递到客户端后，由客户端的浏览器进行解压。在客户端与服务器之间使用gzip传输方式可以大大节省传输时间，加快浏览器的加载速度，改善用户的浏览体验。当然，在服务器端对文件进行压缩和在浏览器端进行解压都会产生一定的CPU开销，因此需要进行权衡。影响的因素有很多，比如浏览器和服务器之间的距离、带宽、文件大小等等。这些因素很多时候难以发现和量化，就算可行的话也会有很多的变量需要去考虑。根据经验，我们一般认为对大于1KB或者2KB的文件进行压缩是比较好的选择。Compression中间件为Express服务器提供了数据压缩的能力，安装和配置中间件之后重启服务器即可开启gzip压缩。

**5.4.2 减少HTTP请求**

减少HTTP请求在很多场景中都是非常适用的性能优化方法。但是具体的操作方法需要根据具体的场景来决定。在此项目中从前后端交互形式入手来达到该目的。

优化之前从后端请求一张图片的过程是，在页面初始化时将所有的图片id请求过来，所有的id以数组的形式交给前端之后，在前端遍历该数组从后端请求该id对应的图片。整个过程中涉及到检索所有id、根据id查询图片等数据库操作。

新的交互形式简单许多，在页面初始化时直接请求图片。不需要向id接口发送请求，不仅减少了网络请求，还降低了检索id、根据id查询图片等数据库操作，使得服务器的响应速度变得更快。

**5.4.3 动态导入**

首页需要加载的资源过多时，除了对资源进行压缩还可以进行按需导入。将首页不需要展示的部分切分出来，当用户特定的行为发生时再导入，为此Webpack提供了import函数。import函数可以将对应的资源切分成代码块，在页面初始化时不予导入。本项目中对路由模块进行了动态导入处理，在React中对首页弹窗也进行了动态导入，使得首次请求资源的体积减少、提升加载速度。

**5.4.4 图片懒加载**

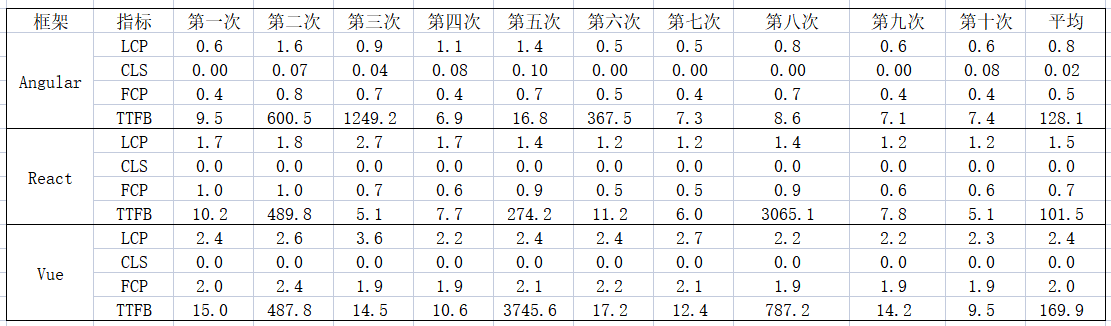
三个项目的性能指标都比较差的一个重要原因是首页图片资源过多，要解决此问题最好的方法是图片懒加载。在页面初次加载时只加载少部分图片，之后根据页面的滚动高度再加载更多对应数量的图片，测试结果也证明这是最有效的优化手段。为了实现该功能，除了前端的请求图片函数之外，还需要对后端接口进行适当的调整，使得接口可以在后端维护图片的请求进度，在下一次请求被接受时从对应位置开始发送图片。

**5.4.5 使用浏览器存储**

在用户点击编辑按钮跳转到编辑页面时需要根据图片id向服务端请求图片，使用浏览器存储可以省略这一步。将对应图片在跳转时存储到localstorage中，返回到首页或者关闭网页时再清除存储可以提升编辑页面的加载速度。

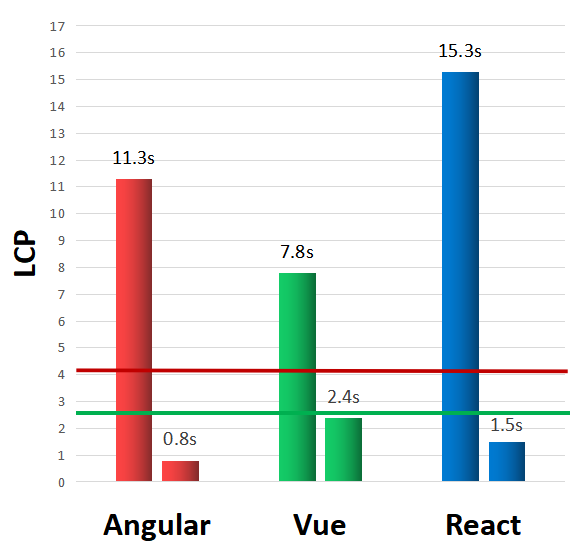
## 5.5 优化结果

优化完成，再次重复4.3节的测试步骤收集数据并再次统计得到的测试结果如图5-6所示。



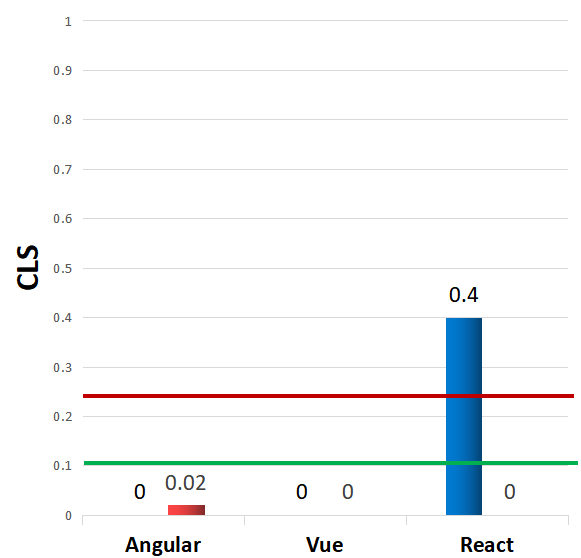
**图**5-6 **优化后的数据表**

图5-6是优化完成之后的统计表，表格很难看出来优化效果如何，图5-7是各个框架LCP指标优化之前和优化完成之后的对比柱状图。



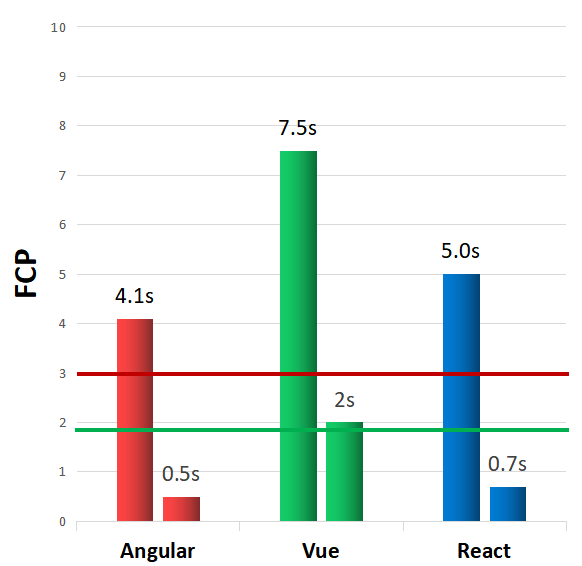
**图**5-7 **优化后的**LCP**指标**

可以看到LCP指标下降巨大，三个框架的LCP指标都降到了优秀线2.5秒以下，页面内容的加载时间显著缩短，最大模块呈现给用户的速度变得更快，优化效果明显。



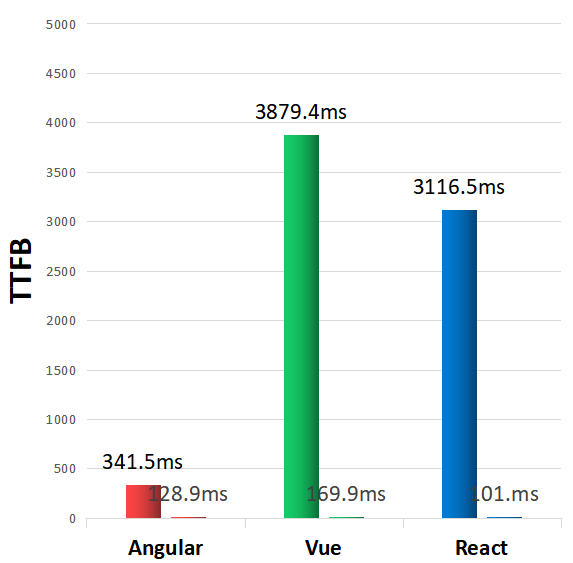
**图**5-8 **优化后的**CLS**指标**

图5-8是CLS指标的优化结果。我们看到React项目的CLS指标由0.4降到了0，视觉稳定性变得更优秀。Vue依然保持在了0。但是一个小瑕疵是前面进行的优化操作给Angular项目引入了少量的布局偏移，从0上升到了0.02。可能是因为为了降低严重超标的LCP指标而对DOM元素进行的调整导致的，不过0.02的累计布局偏移依然在优秀线0.1之下，是可以被接受的，后续需要找出问题的根源，将其重新降低到0.



**图**5-9 **优化后的**FCP**指标**

图5-9所示是优化前后FCP指标的变化，可以看到FCP在三个项目中的降低都很明显，Angular和React都降低到了优秀线1.8秒以下，Vue虽然略微超出了1.8秒，但它5秒的降幅也是最大的，说明优化过程有效，后续可以进行更深度的优化。

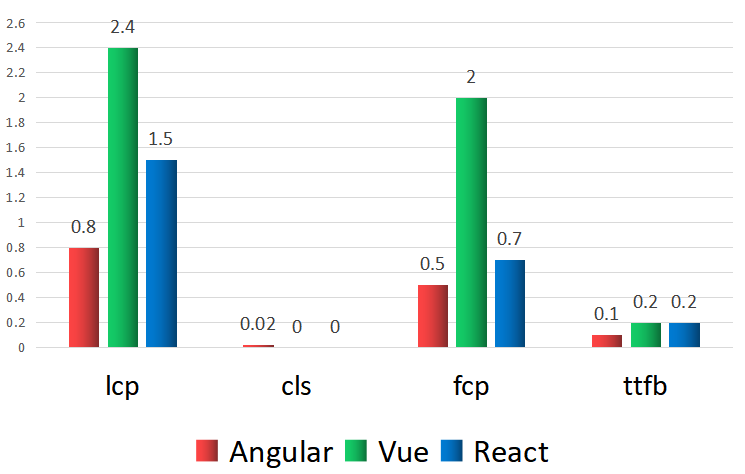


**图**5-10 **优化后的**TTFB**指标**

三个框架的各项性能指标提升明显、优化效果显著。性能优化除了遵循准则和理念，还需要根据实际场景来创造性的解决问题。比如上面的优化过程之中，除了gzip压缩和图片懒加载之外都需要对项目结构做出适当的改变使得代码更高效的运行。这次的测试结果也证明了项目不仅仅是功能实现完成、代码跑起来就可以，还需要在开发完成之后对项目进行测试，找到不足之处并针对性地进行优化，才可以创造出一个高性能的、能够提供良好的用户体验的、有竞争力的产品。

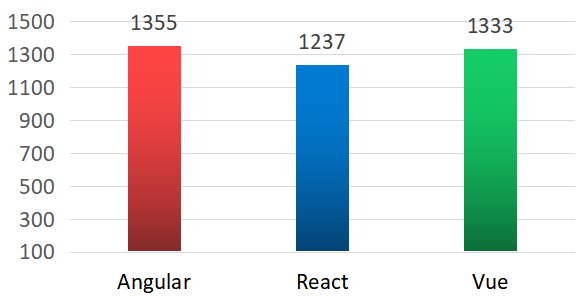
## 5.6 框架横向比较

同一个项目分别用三种框架实现三次，经过性能的测评和优化之后，为框架之间的横向比较提供了前提条件，如图5-11是三种框架在本次项目中的总体性能表现。



**图**5-11 **性能汇总**

可以看到，总体来讲在本次项目中Angular的性能表现最优，React次之。除了性能，出于开发体验和周期考虑，还需要比较系统使用三种框架实现所需的代码行数，如图5-12所示。



**图**5-12 **代码行数**

从图5-12中可以看到，三个项目编写的代码行数都在1300行左右，由于在编写代码的过程中，很难去量化和控制代码质量保持一致，所以一些小的上下波动是完全正常的，可以忽略不计，可以得出框架的简洁性在同一个水平上的结论。

**6 总结与展望**

本章主要内容是对整篇论文所做的工作进行梳理，对线上图片编辑器整体的结构，主要解决的问题和项目现状进行总结。描述目前存在的问题，对需要改进的方面进行梳理，便于对自己以后的工作和学习提出更高的要求。下面对本章总结与展望进行详细说明。

## 6.1 总结

随着浏览器功能变得越来越强大和完善，具备庞大用户需求量的图片处理技术开始往Web上迁移，基于这种趋势和实际需求，本文提出了完全基于浏览器的图片处理解决方法，针对广大普通用户的基本需求开发了三套基于MVVM模式前端框架的线上图片编辑器，为用户提供便捷、灵活的图片处理服务。同时，本文通过各种方法针对四种对用户体验极其重要的性能指标进行性能优化，并对比性能指标优化前后的值来证明优化方法的有效性。下面是论文所作工作的总结。

1. 介绍了课题的研究背景以及意义、框架技术的发展现状。
2. 对系统开发所使用到的主要技术做了详细的说明，介绍了这些当下流行的技术的基础特性，用于解决的问题、使用方法等，并阐述了使用这些技术的原因。
3. 论文按照软件开发流程，首先进行了用户需求分析，阐述了线上图片编辑器的特性以及所需实现的功能、功能模块的划分和发挥的作用。根据需求分析结果进行系统架构的设计，阐述线上图片编辑器架构模型，说明了系统架构各个模块所发挥的作用和所能解决的问题，说明了前后端交互模式和分别需要完成的工作。对前后端各个单独的模块在整个项目开发中的功能和作用进行详细的阐述。
4. 根据需求分析和系统设计结果，对线上图片编辑器进行具体的代码实现，详细阐述前后端实现过程和开发过程中遇到的问题。对系统各个功能的实现做详细的介绍，描述了图片的裁剪、画笔、添加文字等等功能在浏览器中的实现原理。
5. 系统开发完成之后，介绍使用的性能测试工具和对用户体验极其重要的四种性能指标以及它们所代表的意义和计算方式等。并且说明它们是如何对用户体验产生影响，论证针对这四种指标进行优化的必要性。
6. 对三个项目中的四种性能指标各自进行性能测试，收集数据并统计，并以柱状图的形式直观地呈现出测试结果。基于三个项目在四种性能指标中的性能表现，本文根据测试结果提出了针对性的优化措施。
7. 详细介绍所采取的懒加载、动态导入等等优化方法和性能提升原理并编码实现。优化结果显示三个项目的四种性能指标都有了巨大的提升。除性能表现之外，为了比较各个框架的开发速度和简洁性，分别统计了三个框架实现线上图片编辑器所需的代码行数。

## 6.2 展望

论文所有工作已经完成，但是研究过程和线上编辑器的一些系统功能的实现，交互模式和功能完整度上还存在一些不足，后续还需要进行不断地完善和改进。研究方法和系统实现需要改进的内容如下：

1. 由于系统的复杂性，框架特性和第三方库之间的差异，在三个项目的实现过程中出现了不同程度上的差异，可能导致测试结果和开发体验方面的结论可能出现偏差。后续需要继续完善代码、尽量地控制开发条件来提升结论的准确性。
2. 线上图片编辑器需要使用三种框架实现三次，由于时间的关系导致单个系统的功能不够丰富。后续需要不断地进行完善，添加更多的高级功能来提升系统的可用性。
3. 系统的性能测试都是在本地开发环境之下进行，所以可能没能测试一些线上生产环境之下的问题，比如网络堵塞、本地缓存等。这些问题需要在系统功能更加完善、项目结构更加健全、发布到线上投入使用之后继续进行测试和优化。

在论文的编写和整个系统的开发过程中，本人的个人能力也得到了巨大的提升，无论是在写作还是在开发能力方面都变得更加成熟，也接触到了此前没有使用过的技术，比如Angular等。这次毕业设计为本人以后的学习和工作生涯奠定了更加坚实的基础。我希望自己以后可以更加严以律己，继续保持对知识的渴望，不断地提高自己，在学习和工作中变得更加优秀。

**致谢**

非常感谢我的导师陈伟运老师，从开题以来老师对我的论文工作一直非常的关心，非常的负责，给予了我非常多的建议和帮助。每次都对我的论文做出细致的评价和改进建议，指出我的不足，让我可以不断完善自己的论文。每当我在写作或者在研究思路方面遇到困难，老师都会很认真的给我指导。

大学四年即将结束，我就要离开学校走上工作岗位，感谢这四年来给予我帮助的所有老师和同学，祝愿老师和同学们身体健康，工作顺利！

**参考文献**

1. 单苏强. 图片处理APP的使用行为及使用意愿影响因素研究[D]. 南京师范大学, 2017.
2. react官方[EB/OL]. [2022-5-7]. <https://react.docschina.org/.>
3. 刘博文. 深入浅出Vue.js[M]. 人民邮电出版社, 2019-3 :1-282.
4. 孙鑫. Vue.js3.0从入门到实践[M]. 中国水利水电出版社, 2021-04-01 :1-448.
5. 朴灵. 深入浅出node.js[M]. 人民邮电出版社, 2013-12-1 :1-332.
6. Cantelon M. Node.js实战[M]. 人民邮电出版社, 2014-5.
7. Souders S. High Performance Web Sites[M]. O'Reilly Media, 2007-9-21 :1-170.
8. Peter O’Hanlon. Advanced TypeScript Programming Projects[M]. 1. Packt Publishing, 2019-7.
9. Paul Krill. React 18 brings concurrent renderer, automatic batching[J]. InfoWorld.com,2022.
10. Paul Krill. React 17 makes upgrades easier[J]. InfoWorld.com,2020.
11. 吴煜洲. 基于React的可视化编辑平台的设计与实现[D]. 北京交通大学, 2021.
12. 胡芸. 基于React和Node.js的中台开发框架设计与实现[D]. 华中科技大学, 2019.
13. 辛红,基于HTML5+CSS3交互式网页布局的研究[J].考试周刊, 2017(103):195-195.
14. 许家胤. 基于MVVM的仓储物流管理系统设计与实现[D]. 北京邮电大学, 2019.
15. 戴添祥. 图片社交系统前端性能优化[J]. 中国新通信, 2018,20(02):39-40.
16. 冯瑞成. 基于React的设备管理系统前端框架的设计[D].华中科技大学, 2018.
17. 刘艳伟. 适用于web需求的图片处理系统的设计与实现[D]. 北京工业大学, 2013.
18. 朱云峰. Ajax页面局部刷新性能优化[J].常州信息职业技术学院学报, 2011, 10(04):16-18.
19. 徐俊. 基于Vue框架的阅读平台管理系统研究与设计[D]. 北京邮电大学, 2021.
20. 孟艳齐. 基于vue.js的英语核心素养平台的设计与实现[D]. 北京邮电大学, 2021.
21. 陈捷.网站前端技术及其对网站性能的影响研究[J]. 电脑知识与技术, 2019,15(15):68-69.
22. 石冠洲. 基于Vue的Web系统前端性能优化研究与应用[D].长安大学, 2020.
23. 曾卓. 基于CDN的流媒体缓存优化策略研究[D]. 江汉大学, 2021.
24. 张林. Web应用防火墙关于gzip文件的检测研究[J].电子设计工程, 2020,28(19):113-117+125.
25. 王琪. 基于缓存命中率的DNS隧道检测方法研究与实现[D]. 北京邮电大学, 2020.
26. 白斌. 基于Mongoose的制造执行系统设计与实现[D]. 上海交通大学, 2017.
27. 郭蕊,赵元苏.Web前端框架技术综述[J]. 北京工业职业技术学院学报, 2021,20(03):24-27.
28. 刘启伟. 基于Vue.js框架的Web前端开发工具的设计与实现[D]. 北京邮电大学, 2021.

1. 

   **本科毕业设计（论文）任务书**

   题 目 基于MVVM的线上图片编辑器设计与实现

   （任务起止日期：2021年11月2日～2022年6月5日）

   院 系 管理学院

   专业班级 信管1802班

   姓 名 迪力夏提·麦麦提敏

   学 号 U201815616

   指导教师 陈伟运

   教研室（系、所）负责人 20 年 月 日审查

   院（系）负责人 20 年 月 日批准

   |  |
   | --- |
   | 课题内容：  1. 叙述研究背景，论证基于Web的图片编辑器的重要性。 2. 对系统进行需求分析，明确要解决的问题。 3. 对图片编辑器前后端各个模块进行具体的设计与实现。 4. 测试系统性能并优化，给出性能优化指导建议。 5. 对整篇论文所做的工作进行梳理和总结。 |
   | 课题任务要求：  论文主旨明确，结构合理；  系统设计合理，开发出实用的图片编辑器；  关注系统性能，进行深度优化，提供的良好的用户体验； |
   | 主要参考文献（由指导教师选定）：  1. 单苏强. 图片处理APP的使用行为及使用意愿影响因素研究[D]. 南京师范大学, 2017. 2. react官方[EB/OL]. [2022-5-7]. <https://react.docschina.org/.> 3. 刘博文. 深入浅出Vue.js[M]. 人民邮电出版社, 2019-3 :1-282. 4. 孙鑫. Vue.js3.0从入门到实践[M]. 中国水利水电出版社, 2021-04-01 :1-448. 5. 朴灵. 深入浅出node.js[M]. 人民邮电出版社, 2013-12-1 :1-332. 6. Cantelon M. Node.js实战[M]. 人民邮电出版社, 2014-5. 7. Souders S. High Performance Web Sites[M]. O'Reilly Media, 2007-9-21 :1-170. 8. 吴煜洲. 基于React的可视化编辑平台的设计与实现[D]. 北京交通大学, 2021. 9. 胡芸. 基于React和Node.js的中台开发框架设计与实现[D]. 华中科技大学, 2019. 10. 许家胤. 基于MVVM的仓储物流管理系统设计与实现[D]. 北京邮电大学, 2019. 11. 冯瑞成. 基于React的设备管理系统前端框架的设计[D].华中科技大学, 2018. 12. 刘艳伟. 适用于web需求的图片处理系统的设计与实现[D]. 北京工业大学, 2013. 13. 徐俊. 基于Vue框架的阅读平台管理系统研究与设计[D]. 北京邮电大学, 2021. 14. 孟艳齐. 基于vue.js的英语核心素养平台的设计与实现[D]. 北京邮电大学, 2021. 15. 石冠洲. 基于Vue的Web系统前端性能优化研究与应用[D].长安大学, 2020. 16. 曾卓. 基于CDN的流媒体缓存优化策略研究[D]. 江汉大学, 2021. |
   | 同组设计者： |
   | 指导教师签名：  年 月 日 |

   [↑](#endnote-ref-0)
2. [↑](#endnote-ref-1)
3. [↑](#endnote-ref-2)
4. [↑](#endnote-ref-3)
5. [↑](#endnote-ref-4)
6. [↑](#endnote-ref-5)
7. [↑](#endnote-ref-6)
8. [↑](#endnote-ref-7)