目　　录

微视频网站的设计与实现

摘　要

在传统电视播放模式中，用户属于被动角色地位，只能收看节目提供者所播放的节目。而播放时间、播放资源和播放地点等限制使得用户在观看节目中无法享受到自由自在的观看体验。同时，随着社会生活节奏的加快，人们不再满足于冗长而繁琐的长视频，更渴望那些播放时间短，内容丰富，为大众喜闻乐见，能够迅速传递关键信息的短视频，所以人们对于“视频快餐”的需求越来越大。因此，开发一款专门针对人们的“视频快餐”需求的短视频网站很有必要。

基于 web 的微视频网站的设计与实现使用了前后端分离技术,前段采用了 reactjs,后端采用了 SpringCloud 的微服务架构，页面和 api 之间通过 ajax 通信。为用户提供稳定，高效，快速的“视频快餐”；在功能方面实现网上注册用户，在线检索视频，在线视频观看，人性化的视频推荐，在线查看和编辑个人信息，使得用户可以在有限的时间内获取内容丰富的信息，多样化的视频选择也提升了用户的兴趣。同时，微视频为同类软件提供可参考的经验。

Design and Implementation of Micro Video Website.

Abstract

In traditional TV play mode, users are passive roles, and can only watch programs broadcast by program providers. The restrictions such as playing time, playing resources and playing sites make it impossible for users to enjoy the free viewing experience in watching programs. At the same time, with the rapid pace of social life, people are no longer satisfied with long and tedious long video. They are more eager for short video with short time, rich content, good news for the public and quick transmission of key information. So people need more and more "video fast food". Therefore, it is necessary to develop a short video website which is aimed at people's demand for "video fast food".

The design and implementation of web based micro video website uses the front and back separation technology, the front segment uses the reactjs, the back end uses the SpringCloud micro service architecture, and the page and API communicate through Ajax. To provide users with stable, efficient and fast "video fast food", online registration users, online video retrieval, online video viewing, humanized video recommendation, online viewing and editing personal information, enabling users to obtain rich information and diversified video in a limited period of time. The choice also improves the interest of the user. At the same time, micro video provides similar experience for similar software.

第1章 前 言

1.1 项目的背景和意义

几十年前，传统的电视播放在市场中占据了主导地位，而刚发展起来的网络视频主要只是对传统的电视播放进行了补充，还没有形成自己的市场和观众，在运营模式，技术和功能方面存在许多不足。随着互联网络科技快速发展，大量的资本涌入网络信息市场，网络市场逐渐形成和壮大，网络应用技术根据市场需求也随之发生极大的变革和进步，特别是以 web2.0 为代表的新一代互联网技术，网上视频以起快捷，方便，内容丰富的特点逐渐壮大。

但是，随着社会生活节奏的加快，人们的空闲时间被压缩，获取信息的时间也被压缩，人们更渴望在有限的时间里获取更多的信息。所以人们不再局限于只能在网上看视频，更渴望那些播放时间短，内容丰富，为大众喜闻乐见，能够迅速传递关键信息的微视频，所以人们对于“视频快餐”的需求越来越大。

因此，就需要一种专门针对人们的“视频快餐”需求的短视频网站。

1.2 研究开发现状分析

视频网站行业作为最早进入互联网科技的行业之一，发展到现如今已经形成了一定规模。

自 2016 年，短视频行业进入黄金爆发期。陌陌、花椒、映客等直播平台“抢滩”短视频；互联网巨头 BAT 更是不甘落后，重金砸向短视频领域；新浪、搜狐等出台短视频扶持计划，欲在“乱战”中分一杯羹；秒拍“金栗子”奖现场看片会暨颁奖仪式吸引了来自国内短视频行业的优秀内容方、资本方和平台方齐聚一堂。可谓群雄逐鹿，精彩纷呈。

短视频行业发展现状：

1. 短视频发展历程,如图 1-1。

图 1-1 短视频发展历程

2. 短视频发展环境,如图 1-2。

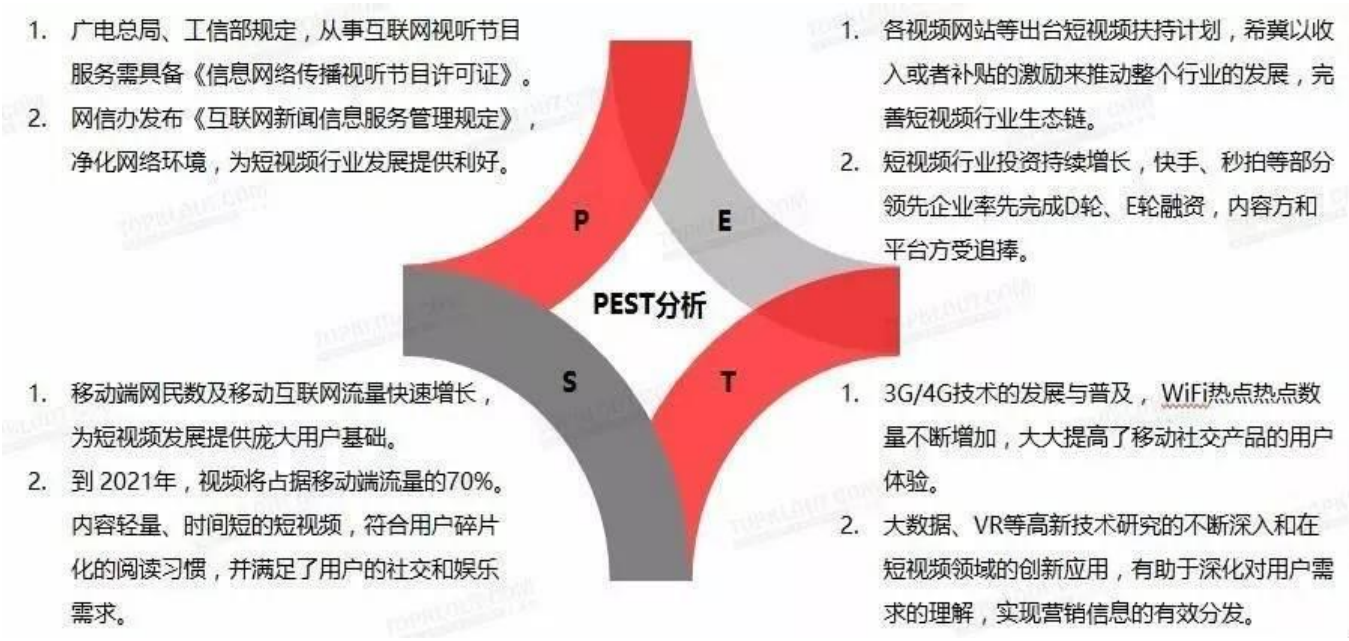


图 1-2 短视频发展环境

3. 短视频行业全生态图谱,如图 1-3。

图 1-3 短视频行业全生态图谱

通过研究发现，人们对于信息丰富、准确，内容简短的视频需求的需求越来越强烈，市场广阔，发展环境良好，微视频与传统的大电影行业的内容迭代周期

长，信息接受时间长，需要收取一定的费用等缺点相比拥有巨大的优势。

网站后端以 JAVA 语言开发，结合微服务分布式架构，将系统功划分为不同的微服务，一个为服务从数据库到界面都是独立于其他微服务，单个服务功能的故障不会影响其他服务的功能；同时，微服务能够在分布式环境下进行多实例的部署。这样一来，就构建一个拓展性强，性能稳定，高效的系统；前端运用了当前流行的 reactjs、webpack 等技术，开发效率高，运行效率高，界面美观，用户体验流畅。

1.3 项目的目标和范围

本论文的研究内容主要是开发一个微视频网站。志在于设计出基于微服务分布式架构和 reactjs，高效、稳定、拓展性强的系统。为广大用户提供信息丰富，内容简短，喜闻乐见的短视频服务。为了更好的帮助后面系统的设计与实现，本文将从以下几个方面进行探讨：

（1）研究了视频网站的设计思路及整体框架。

（2）分析了系统实现所需的关键技术。视频信息要尽量全面才能满足不同需要的用户，而且要简单易操作，系统稳定、高效。所以系统采用微服务架构和高效的 reactjs。

（3）平台要实现的基本功能：

前台：用户登录、注册； 检索视频；观看视频；查看视频信息等；

后台：管理员登录、注册；管理视频，角色，视频；基于 RBAC 的权限控制；在线状态实时监测等。

1.4 论文结构简介

本篇论文主要由七章构成，论文的头部为摘要部分，该部分主要记述的是本文的研究背景经济当前市场需求现状及解决后所产生的重大意义。接下来便是本文的重点部分。

第1章：主要阐述了项目的背景与意义、行业现状和发展趋势以及项目的研究范围与目标。

第2章：技术与原理，本章的主要内容为：简述本系统所涉及到的一些开发技术及特点，系统运行的响应流程。

第3章：需求建模，本章为全文比较重要的章节，本部分主要介绍系统的业务建模，系统部分功能采用了用例图给予说明，并详细论述了系统的主要模块。

第4章：本章是架构设计，主要是介绍了本系统的架构及原理，该系统主要使用的是微服务架构，简单介绍各部件及部件之间关系，业务用例的实现以及数据库的设计。

第5章：模块设计章节主要是介绍本系统的几个主要功能模块，模块的说明、接口及算法，同时展示该模块的界面设计。

第6章：部署和应用章节，主要是介绍本系统的运行环境，以及重要的系统输入和输出。

第7章：本章节是总结，对本论文和本系统进行概括性总结，总结做得好的和不足的地方。

第 2 章　技术与原理

本系统页面部分主要采用 reactjs 技术，在后台业务流程处理方面使用微服务架构。使用前后端分离开发的模式。整个系统的处理逻辑为：用户通过前台页面发起 ajax 请求，然后由 nginx 反向代理转发至 gateway(网关)，通过网关分发前台请求，请求对应的微服务，微服务返回数据到网关，网关返回响应至前台页面。至此，系统完成用户请求的一次完整响应。

2.1　开发技术

2.1.1　Mysql 数据库

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL 是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

2.1.2　Rabbitmq

MQ 全称为 Message Queue,消息队列（MQ）是一种应用程序对应用程序的通信方法。应用程序通过读写出入队列的消息（针对应用程序的数据）来通信，而无需专用连接来链接它们。消息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。排队指的是应用程序通过 队列来通信。队列的使用除去了接收和发送应用程序同时执行的要求。其中较为成熟的 MQ 产品有 IBM WEBSPHERE MQ 等等。

2.1.3　Springboot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot 致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。

2.1.4 Spring cloud 微服务

SpringCloud 是基于 SpringBoot 的一整套实现微服务的框架。他提供了微服务开发所需的配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等组件。最重要的是，跟 spring boot 框架一起使用的话，会让你开发微服务架构的云服务非常好的方便。SpringBoot 旨在简化创建产品级的 Spring 应用和服务，简化了配置文件，使用嵌入式 web 服务器，含有诸多开箱即用微服务功能，如图 2-1。

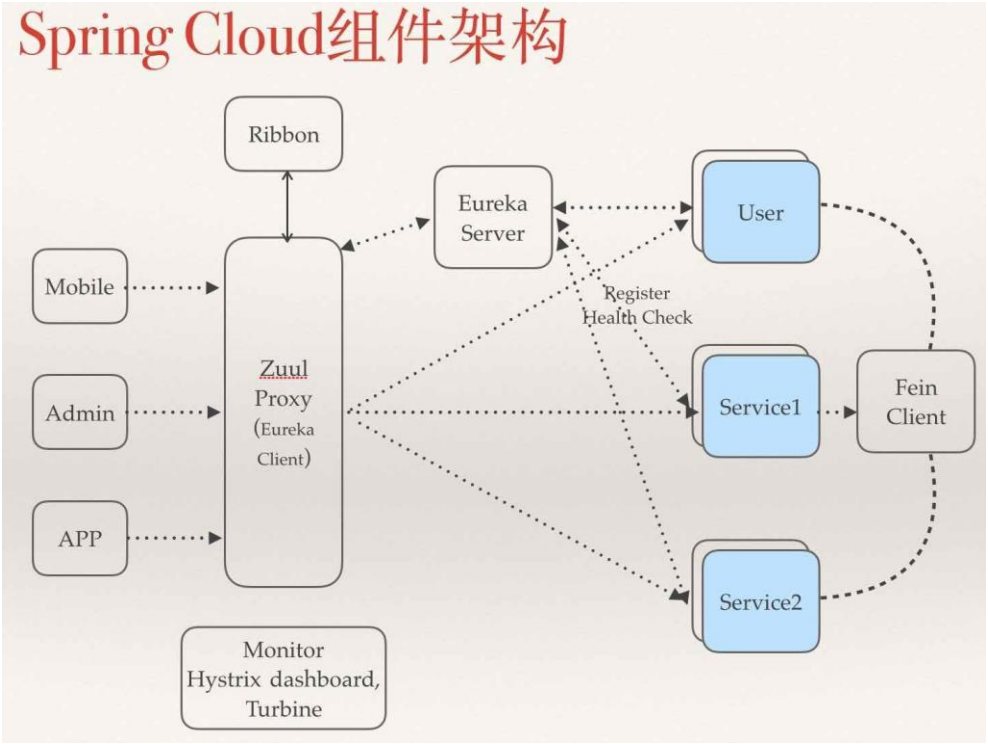


图 2-1 Spring Cloud 组件架构

2.1.5 Docker 容器

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

2.1.6 Docker-compose

Docker Compose 是 Docker 编排服务的最后一块，前面提到的 Machine 可以让用户在其它平台快速安装 Docker，Swarm 可以让 Docker 容器在集群中高效运转，而 Compose 可以让用户在集群中部署分布式应用。简单的说，Docker Compose 属于一个“应用层”的服务，用户可以定义哪个容器组运行哪个应用，它支持动态改变应用，并在需要时扩展。

2.1.7 Nodejs

Node.js 是一个 Javascript 运行环境(runtime environment)，发布于 2009 年 5 月，由 Ryan Dahl 开发，实质是对 Chrome V8 引擎进行了封装。Node.js 对一些特殊用例进行优化，提供替代的 API，使得 V8 在非浏览器环境下运行得更好。

2.1.8 Websocket：

WebSocket 协议是基于 TCP 的一种新的网络协议。它实现了浏览器与服务器全双工(full-duplex)通信——允许服务器主动发送信息给客户端。

2.1.9 Sass

Sass 是一款强化 CSS 的辅助工具，它在 CSS 语法的基础上增加了变量(variables)、嵌套 (nested rules)、混合 (mixins)、导入 (inline imports)等高级功能，这些拓展令 CSS 更加强大与优雅。使用 Sass 以及 Sass 的样式库（如 Compass）有助于更好地组织管理样式文件，以及更高效地开发项目。

2.1.10 React

React 是近期非常热门的一个前端开发框架，其本身作为 MVC 中的 View 层可以用来构建 UI，也可以以插件的形式应用到 Web 应用非 UI 部分的构建中，轻松实现与其他 JS 框架的整合，比如 AngularJS。同时，React 通过对虚拟 DOM 中的微操作来实对现实际 DOM 的局部更新，提高性能。其组件的模块化开发提高了代码的可维护性。单向数据流的特点，让每个模块根据数据量自动更新，让开发者可以只专注于数据部分，改善程序的可预测性。

React 中最基础最重要的就是 Component 了，它的构造和功能相当于 AngularJS 里面的 Directive，或是其他 JS 框架里面的 Widgets 或 Modules。Component 可以认为是由 HTML、CSS、JS 和一些内部数据组合而成的模块。当然 Component 也可以由很多 Component 组建而成。不同的 Component 既可以用纯 JavaScript 定义，也可以用特有的 JavaScript 语法 JSX 创建而成。

2.1.11 Ant design

蚂蚁金服体验技术部经过大量的项目实践和总结，沉淀出设计语言 Ant Design，这可不单纯只是设计原则、控件规范和视觉尺寸，还配套有前端代码实现方案。也就是说采用 Ant Design 后，UI 设计和前端界面研发可同步完成，效率大大提升。目前有阿里、美团、滴滴、简书采用。Ant Design 有 Web 版和 Moblie 版。

2.1.12 Mybatis

MyBatis 本是 apache 源项目 iBatis, 2010 年这个项目由 apache software foundation 迁移到了 google code，并且改名为 MyBatis 。2013 年 11 月迁移到

Github。

iBATIS 一词来源于“internet”和“abatis”的组合，是一个基于 Java 的持久层框架。iBATIS 提供的持久层框架包括 SQL Maps 和 Data Access Objects（sDAO）。

2.2 开发工具

2.2.1 Git 版本控制工具

Git 是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

Git 是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

2.2.2 IntelliJ IDEA 工具

IDEA 全称 IntelliJ IDEA，是 java 语言开发的集成环境，IntelliJ 在业界被公认为最好的 java 开发工具之一，尤其在智能代码助手、代码自动提示、重构、J2EE 支持、各类版本工具(git、svn、github 等)、JUnit、CVS 整合、代码分析、 创新的 GUI 设计等方面的功能可以说是超常的。IDEA 是 JetBrains 公司的产品，这家公司总部位于捷克共和国的首都布拉格，开发人员以严谨著称的东欧程序员为主。它的旗舰版本还支持 HTML，CSS，PHP，MySQL，Python 等。免费版只支持 Java 等少数语言。

2.2.3 HeidiSql 数据库管理工具

HeidiSQL 是一款用于简单化迷你的 MySQL 服务器和数据库管理的图形化界面。HeidiSQL 提供了一个用于在数据库浏览之间切换 SQL 查询和标签带有语法突出显示的简单易用的界面。其它功能包括 BLOB 和 MEMO 编辑，大型 SQL 脚本支持，用户进程管理等。该软件资源开放。

2.2.4 Jenkins 持续集成

Jenkins 是一个开源软件项目，是基于 Java 开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能。

2.1.6 Centos 服务器

CentOS（Community Enterprise Operating System，中文意思是：社区企业操作系统）是 Linux 发行版之一，它是来自于 Red Hat Enterprise Linux 依照开放源代码规定释出的源代码所编译而成。由于出自同样的源代码，因此有些要求高度稳定性的服务器以 CentOS 替代商业版的 Red Hat Enterprise Linux

使用。两者的不同，在于 CentOS 并不包含封闭源代码软件。

2.1.12 maven

Maven 项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

第 3 章　需求建模

3.1　可行性研究

3.1.1　市场可行性分析

当前，人民日益增长的物质文化需求在某些方面已经得不到满足，在娱乐市场，尤其是微视频方面具有广阔的市场；

就开发费用而言，采用了像是 idea，mysql，springboot，heidisql 等开源免费的工具，无需任何费用，在开发费用方面不存在问题。

在后期维护方面，由于采用了像是 mysql，springboot 等轻量级的，简介开发框架，维护难度小，维护成本低。

3.1.2　技术可行性分析

本项目是一个独立且完整的系统，要求开发的严谨，因此掌握软件开发流程、项目管理、界面开发、事件处理以及用户输入逻辑的处理是非常重要的。为了能够全面的掌握知识，我阅读了很多与本系统相关的书籍，加强了自己对项目开发流程的理解。Vm 微视频网站采用了目前流行的微服务架构和 springcloud 框架设计。这些技术具有众多的商业项目成功案例，且体现出良好的性能，取得了傲人的成绩，得到了众多程序开发者的青睐。同时，在网络上也能找到众多技术博客，为本项目提供技术支持，所以说技术上是可行的。

3.1.3　运行可行性分析

本系统面向的使用对象要求低，只需要用户有基本的计算机使用能力就能够使用本系统，系统运行所需配置较低，用户可以在计算机端和移动端进行访问，极大的方便了用户的使用，只需要用户所使用设备连接上网络，则可以对本系统进行操作。

3.2　功能需求分析

3.2.1　前端用户需求分析

1. **登录**：前端用户通过 6-12 位的数字、字母账户，6-12 位的数字、字母、下划线、点密码登录。
2. **注册**：前端用户通过 6-12 位的数字、字母账户，6-12 位的数字、字母、

下划线、点密码登录。

3. **注销**：前端用户可以退出此次登录。

4. **浏览微视频列表**：可通过各种条件进行微视频检索。检索方式：分页检索，排序检索，标签检索，关键字检索；

5. **查看微视频信息**：用户可以查看微视频的各项信息，包括主图，标题，介绍，上映时间，最后更新时间，主演，播放次数，评分，查看微视频人。

6. **查看微视频人**：查看其基本信息，查看其微视频

7. **观看微视频**：用户可以在线观看微视频；查看推荐。

8. **个人信息管理**：包括对基本信息（账户[6-12 位的数字、字母账户]，出生日期，描述[255 位以内]）等的查看和修改；

3.2.2 后端管理员需求分析

1. **登录**：通过 6-12 位的数字、字母账户，6-12 位的数字、字母、下划线、点密码登录。

2. **用户日志管理**：管理用户日志。

3. **角色管理**：对系统角色进行管理。

4. **管理员管理**：对管理员进行管理。

5. **用户管理**：对用户进行管理。

6. **微视频管理**：对微视频图片。名称，演员等信息进行管理。

7. **微视频人管理**：对微视频人的基本信息进行管理。

8. **类型管理**：对微视频类型进行管理。

9. **管理员日志管理**：管理用户日志。

3.3 系统用例分析

3.3.1 前端用户用例分析

如图 3-1 所示。

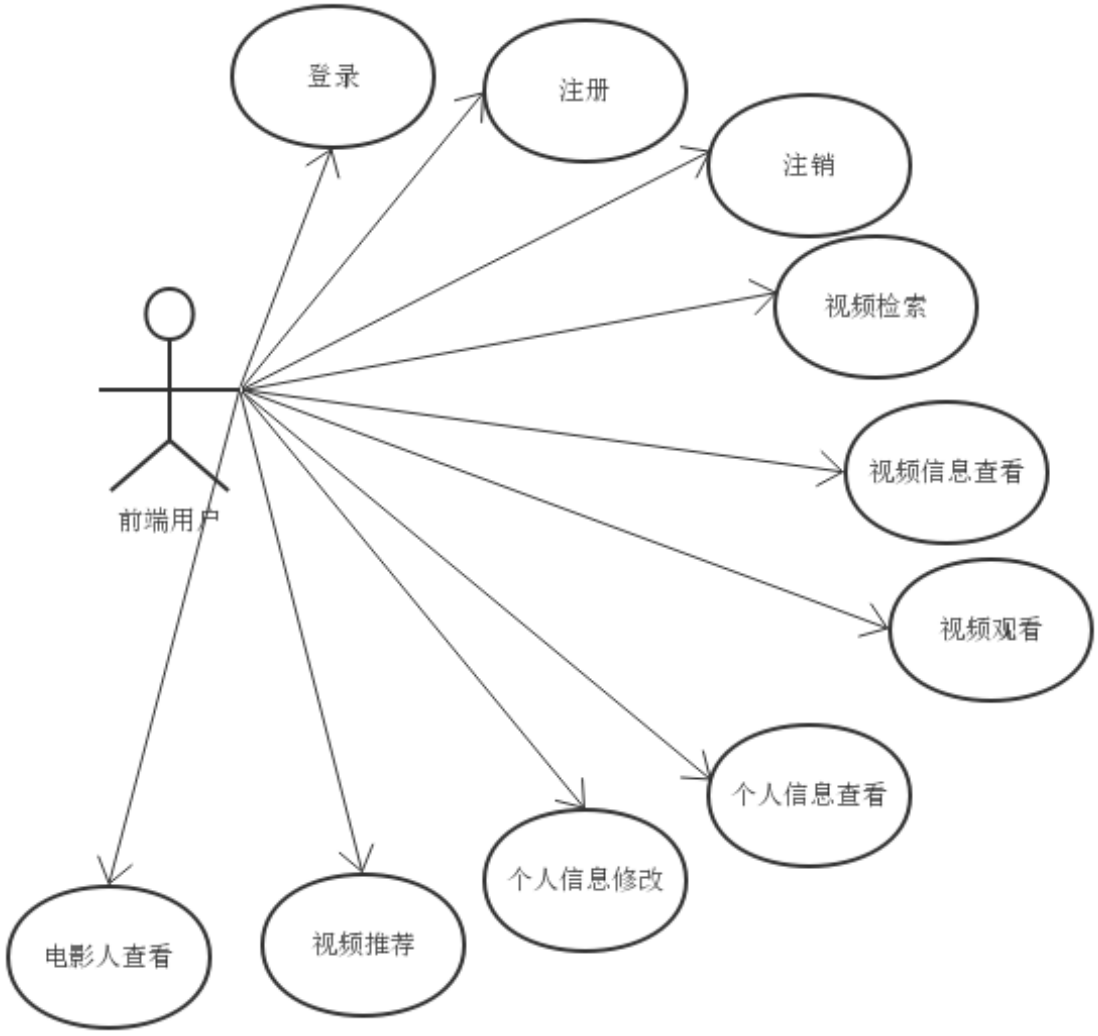


图 3-1 前端用户用例图

前端用户登录用例描述如表 3-1 所示。

表 3-1 前端用户登录用例描述说明

用例标识	U001	用例名称	前端用户登录
创建人	张可	创建日期	2r017/11/30
用例描述	此用例主要描述微视频网站的用户登录。由于微视频用户需要对个人信息进行管理，在用户访问敏感信息时，首先要通过合法的用户身份验证		
参与者	前端用户		
前置条件	用户使用在微网站		
后置条件	微视频用户信息的查看和修改		
基本操作流程	1.用户进入登录界面 2.用户输入正确的用户名、密码 3.用户点击登录按钮		

	4.系统校验用户输入的信息在后台数据库中是否存在，确认无误后，根据身份显示登录用户信息
可选操作流程	2a.用户输入用户名或密码错误 2b.系统提示用户重新输入，然后继续执行基本流的步骤 2。
被泛化的用例	无
被包含的用例	无

视频检索用例描述如表 3-2 所示。

表 3-2 视频检索用例描述说明

用例标识	U002	用例名称	视频检索
创建人	张可	创建日期	2017/11/30
用例描述	此用例主要描述微视频网站的视频检索。由于数量多，类型丰富，需要通过检索方式查找视频		
参与者	前端用户		
前置条件	用户进入检索页面		
后置条件	用户可以查看视频信息和观看视频		
基本操作流程	1.用户进入检索页面 2.用户正确输入关键字检索或者选择分类检索 4.用户点击搜索按钮 5.系统返回相关视频并且展示		
可选操作流程	2a.用户重复搜索关键字 2b.系统提示用户重新输入，然后继续执行基本流的步骤 2。		
被泛化的用例	无		
被包含的用例	无		

3.3.2 后端管理员用例分析

后端管理员用例如图 3-2 所示

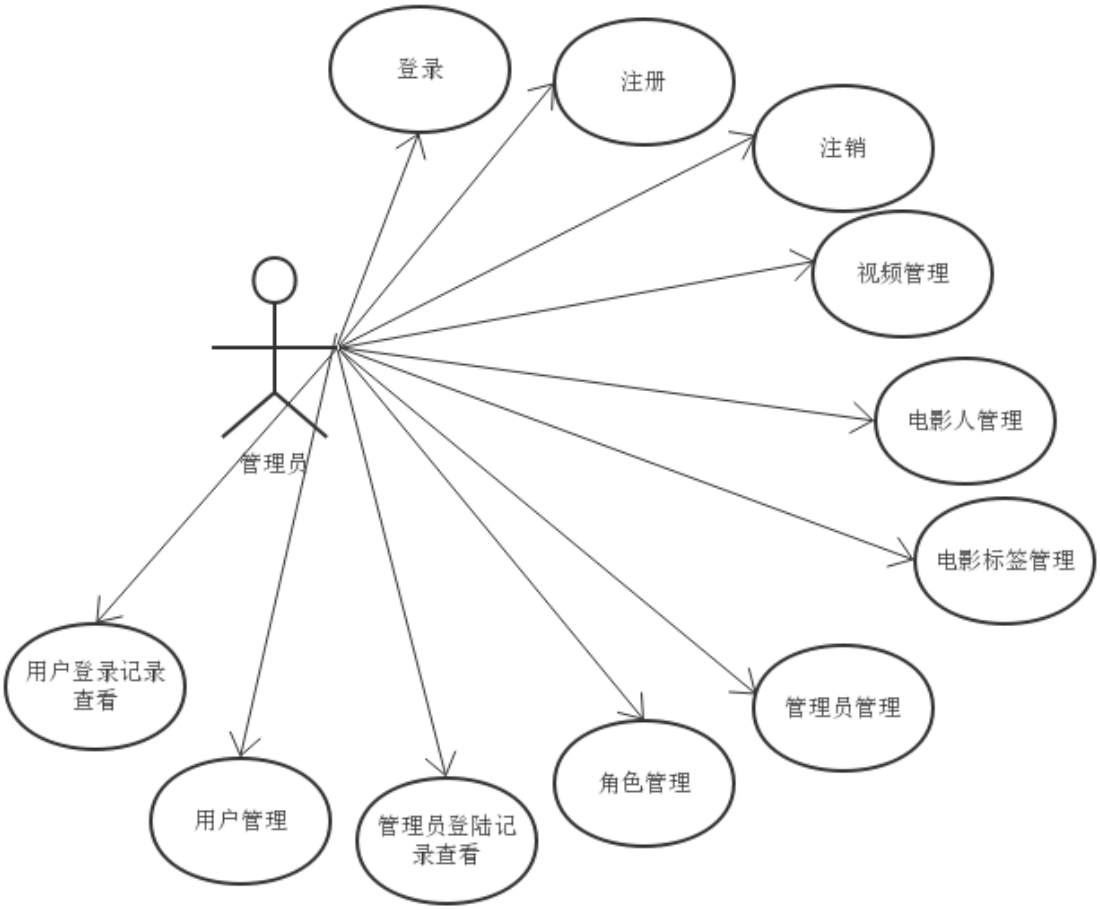


图 3-2　后端管理员用例图

后端管理员登录用例描述如表 3-3 所示。

表 3-3 后端管理员登录用例描述说明

用例标识	A001	用例名称	后端管理员登录
创建人	张可	创建日期	2017/11/30
用例描述	此用例主要描述微视频网站的管理员登录。由于微视频管理员需要对各种信息进行管理，在进行管理之前，首先要通过合法的用户身份验证		
参与者	后端管理员		
前置条件	用户进入管理员登录界面		
后置条件	管理员可以根据自身权限对信息进行管理		
基本操作流程	1.管理员进入登录界面 2.管理员输入正确的用户名、密码 3.管理员点击登录按钮 4.系统校验管理员输入的信息在后台数据库中是否存在，确		

	认无误后，根据身份显示登录管理员信息
可选操作流程	2a.管理员输入用户名或密码错误 2b.系统提示管理员重新输入，然后继续执行基本流的步骤 2。
被泛化的用例	无
被包含的用例	无

后端管理员登录用例描述如表 3-4 所示。

表 3-4 后端管理员管理用例描述说明

用例标识	A002	用例名称	管理员管理
创建人	张可	创建日期	2017/11/30
用例描述	此用例主要描述微视频网站的管理员对管理员的管理。可以对某个管理员的信息进行修改		
参与者	后端管理员		
前置条件	用户进入管理员管理界面且拥有相关操作权限		
后置条件	管理员可以根据自身权限对信息进行管理		
基本操作流程	1.管理员进入管理员管理界面 2.管理员修改相关信息 3.管理员点击保存按钮 4.系统校验管理员的权限，返回操作结果		
可选操作流程	4a.管理员没有相关权限 4b.系统提示管理员权限不足		
被泛化的用例	无		
被包含的用例	无		

第 4 章　架构设计

4.1　系统整体架构

4.1.1　系统整体设计

本系统页面部分主要采用 reactjs 技术，在后台业务流程处理方面使用微服务架构。使用前后端分离开发的模式。整个系统的处理逻辑为：用户通过前台页面发起 ajax 请求，然后由 nginx 反向代理转发至 gateway(网关)，通过网关分发前台请求，请求对应的微服务，微服务返回数据到网关，网关返回响应至前台页面。至此，系统完成用户请求的一次完整响应。系统整体架构如图 4-1 所示：

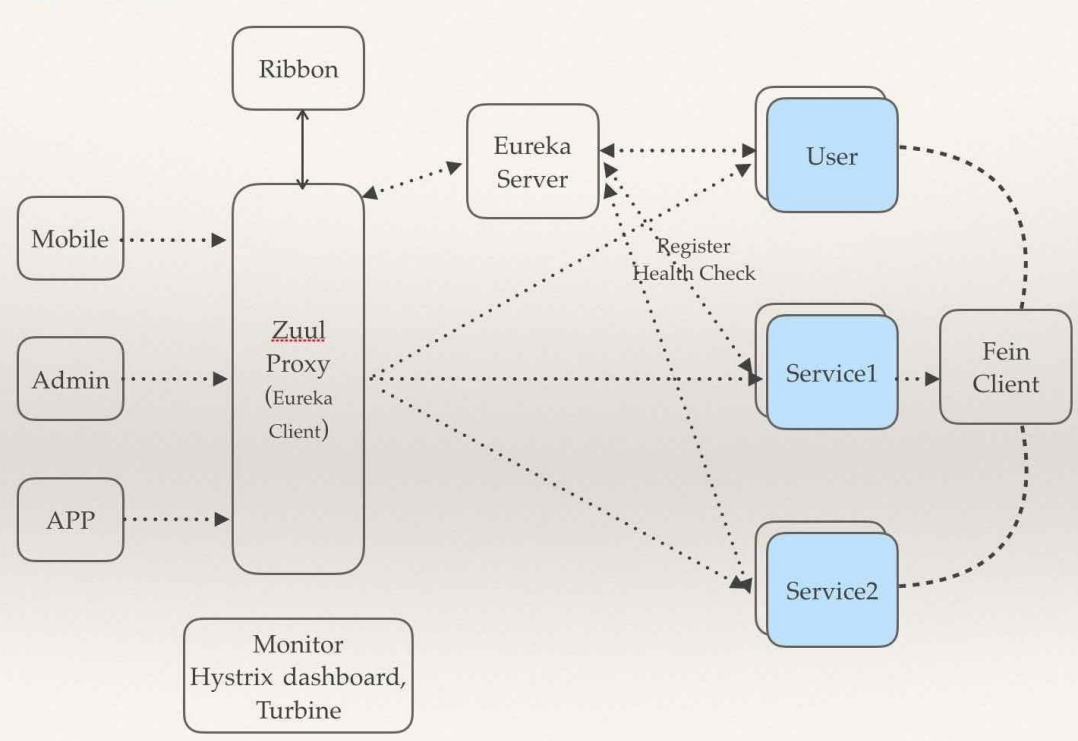


图 4-1 系统总体架构设计图

4.2　功能模块设计

微视频网站采用了前后端分离的开发方式，这决定了页面的开发方式将不同于传统的 jsp+servlet；在传统的 jsp+servlet 项目中，页面和服务 api 共在一个项目，而前后端分离的方式将会把页面和服务 api 拆分为两个或者多个项目。平行开发，提高了开发效率和项目可控性。

微视频也采用了微服务的架构，这决定了服务 api 将不会集中于一个服务实例中，这些 api 将会被拆分到不同的微服务中，在生产环境中进行分布式，多实

例的部署。

因此，从系统架构的角度划分模块，那么一个为服务就可以划分为子模块。

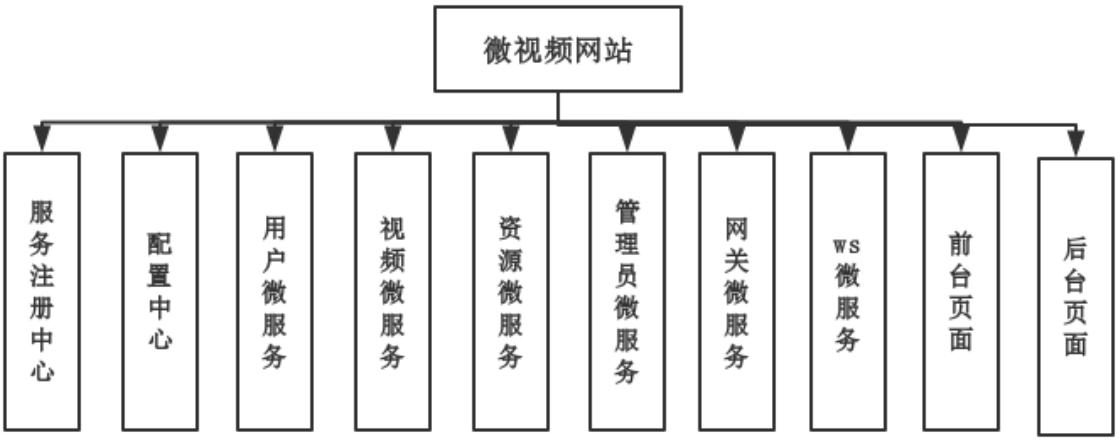


图 4-2 系统模块图

（1）用户微服务：

该模块负责处理与用户相关的请求，包括用户列表的获取，单个用户信息的查询，用户状态的保存（登录状态），用户登录和注册等。

（2）管理员微服务：

该模块负责处理与管理员相关的请求，包括管理员列表的获取，单个管理员信息的查询，管理员状态的保存（登录状态），管理员登录等。

（3）视频微服务：该模块负责处理与微视频相关的请求，包括微视频列表的获取，单个微视频信息的查询，微视频人的获取，微视频人的修改、更新等。

（4）资源微服务：该模块负责文件的储存与读取。

（5）网关微服务：采用不同的负载均衡策略分发前台请求，也具有熔断等功能。

（6）注册中心模块：为所有的微服务提供注册服务。

（7）配置中心：是项目的配置中心，储存配置文件，储存位置可以的本地，也可以是 git。

（8）ws 微服务：提供 websocket 的集群支持。

（9）前台页面：提供前台页面支持。

（10）后台页面：提供后台页面支持。

4.3 数据库设计

根据本系统的需求分析得到系统的表结构说明如下:

数据库名（vm），数据表：管理员表（vm_admins），管理员登录记录表（vm_admins_login_logs），国家地区表（vm_countrys），文件资源表（vm_files），微视频人信息表（vm_filmmakers），微视频表（vm_movies），微视频与微视频人关系表（vm_movies_filmmakers_realation），微视频版本表（vm_movies_src_version），微视频标签关系表（vm_movies_tags_realation），微视频标签表（vm_tags），微视频标签分组表（vm_tags_groups），用户表（vm_users），用户登录记录表（vm_users_login_logs）,管理员角色关系(vm_admins_roles_realation)，管理员权限表（vm_auths），管理员菜单（vm_menus），管理员角色（vm_roles），角色权限关系表（vm_roles_auths_realation），角色菜单关系表（vm_roles_menus_realation）。具体的设计如表 4-3 至表 4-21：

表 4-3 管理员表(vm_admins)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	username	唯一用户名	varchar	20		
3	password	用户密码，md5 加密	varchar	40		
4	description	用户密码，md5 加密	varchar	255		
5	create_time	创建时间	int	10,0		
6	update_time	更新时间	int	10,0		
7	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
8	status	状态,1 为正常,2 为冻结	tinyint	3,0		1
9	immutable	是否为内置不可变对象，1 为是，2 为否	tinyint	3,0		

表 4-4 管理员登录记录表(vm_admins_login_logs)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	admin_id	登录的管理员 id	bigint	20,0		
3	login_ip	登录 ip	varchar	20		
4	system	电脑系统	varchar	40		

5	dpi	电脑分辨率	varchar	40		
6	brower	使用的浏览器	varchar	1000		
7	country	登录国家	varchar	50		
8	province	登录省份	varchar	40		
9	city	登录城市	varchar	40		
10	login_time	登录时间	int	10,0		
11	result	登录结果，1 位成功，2 位失败	tinyint	3,0		
12	create_time	创建时间	int	10,0		
13	update_time	更新时间	int	10,0		
14	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
15	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-5 国家地区表(vm_countrys)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id		tinyint	3,0		
2	code		varchar	16		
3	name_chinese		varchar	128		
4	name_english		varchar	128		
5	create_time	创建时间	int	10,0		
6	update_time	更新时间	int	10,0		
7	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
8	status	状态,1 为正常，2 为冻结	tinyint	3,0		1

表 4-6 文件资源表(vm_files)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		

2	filename	图片名,如 a.png	varchar	255		
3	original_name	文件原名	varchar	255	√	
4	file_size	图片大小	bigint	20,0	√	
5	create_time	创建时间	int	10,0		
6	update_time	更新时间	int	10,0		
7	content_type	文件类型，如 video/mp4	varchar	20	√	
8	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
9	status	状态,1 为正常，2 为冻结	tinyint	3,0		1

表 4-7 微视频人信息表(vm_filmmakers)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	name	演员名	varchar	45		
3	alias	别名	varchar	45		
4	profession	职业	varchar	255		
5	blood_type	血型，A,B,AB,O,E,未知	tinyint	3,0		
6	sex	性别，1 为男，1 为女，3 未设置	tinyint	3,0		3
7	birthday	演员生日	int	10,0		
8	country	演员国家	varchar	255		
9	description	演员描述	varchar	255		
10	img_url	图片地址	varchar	255	√	
11	create_time	创建时间	int	10,0		
12	update_time	更新时间	int	10,0		
13	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
14	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-8 微视频表(vm_movies)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增 id	bigint	20,0		
2	name	标题	varchar	45		
3	alias	别名	varchar	45		
4	description	描述	varchar	1000		
5	director_id	导演 id（指向电影人表：vm_filmmakers）	bigint	20,0	√	
6	release_time	上映时间	int	10,0		
7	score	评分	float	3,1		
8	watch_num	观看数量	bigint	20,0		0
9	movie_time	电影时长	int	10,0		
10	poster_url	播放器显示的图片 url	varchar	100	√	
11	img_url	列表图片 url	varchar	100	√	
12	create_time	创建时间	int	10,0		
13	update_time	更新时间	int	10,0		
14	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
15	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-9 微视频与微视频人关系表(vm_movies_filmmakers_realation)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	movie_id	电影 id	bigint	20,0		
3	filmmaker_id	电影人 id	bigint	20,0		
4	create_time	创建时间	int	10,0		
5	update_time	更新时间	int	10,0		
6	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
7	status	状态,1 为正常，2 为冻结	tinyint	3,0		1

表 4-10 微视频版本表(vm_movies_src_version)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	create_time	创建时间	int	10,0		
3	update_time	更新时间	int	10,0		
4	sharpness	清晰度，1 代表标清，2 代表高清，3 代表超清	tinyint	3,0		
5	movie_id	电影 id	bigint	20,0		
6	weight	权重	tinyint	3,0		0
7	src_url	资源 url	varchar	100		
8	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
9	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-11 微视频标签关系表(vm_movies_tags_realation)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	movie_id	电影 id	bigint	20,0		
3	tag_id	类型 id	bigint	20,0		
4	create_time	创建时间	int	10,0		
5	update_time	更新时间	int	10,0		
6	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
7	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-12 微视频标签表(vm_tags)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		

2	name	类型名	varchar	45		
3	tag_group_id	所属标签组	bigint	20,0		
4	create_time	创建时间	int	10,0		
5	update_time	更新时间	int	10,0		
6	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
7	status	状态,1 为正常，2 为冻结	tinyint	3,0		1

表 4-13 微视频标签分组表(vm_tags_groups)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	name	标签组名	varchar	45		
3	create_time	创建时间	int	10,0		
4	update_time	更新时间	int	10,0		
5	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
6	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-14 用户表(vm_users)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	username	唯一用户名	varchar	20		
3	password	用户密码，md5 加密	varchar	40		
4	sex	性别，1 为男，2 为女，3 未设置	tinyint	3,0		3
5	birthday	用户生日	int	10,0	√	
6	description	用户描述	varchar	255	√	
7	create_time	创建时间	int	10,0		
8	update_time	更新时间	int	10,0		
9	img_url	用户头像 url	varchar	255	√	
10	is_deleted	状态,1 为 yes,2 为 no	tinyint	3,0		1
11	status	状态,1 为正常,2 为冻	tinyint	3,0		1

结

表 4-15 用户登录记录表(vm_users_login_logs)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	user_id	用户 id	bigint	20,0		
3	login_ip	登录 ip	varchar	20		
4	system	电脑系统	varchar	40		
5	dpi	电脑分辨率	varchar	40		
6	brower	使用的浏览器	varchar	1000		
7	country	登录国家	varchar	50		
8	province	登录省份	varchar	40		
9	city	登录城市	varchar	40		
10	login_time	登录时间	int	10,0		
11	result	登录结果，1 位成功，2 为失败	tinyint	3,0		
12	create_time	创建时间	int	10,0		
13	update_time	更新时间	int	10,0		
14	is_deleted	状态,1 为 yes,2 为 no	tinyint	3,0		1
15	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-16 管理员角色关系表(vm_admins_roles_realation)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	create_time	创建时间	int	10,0		
3	update_time	更新时间	int	10,0		
4	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
5	status	状态,1 为正常,2 为冻结	tinyint	3,0		1
6	role_id	角色 id	bigint	19,0		
7	admin_id	管理员 id	bigint	19,0		

表 4-17 管理员权限表(vm_auths)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	auth_name	资源名	varchar	20		
3	auth_code	权限码,如 user:add	varchar	255		
4	description	用户密码，md5 加密	varchar	255		
5	create_time	创建时间	int	10,0		
6	update_time	更新时间	int	10,0		
7	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
8	status	状态,1 为正常,2 为冻结	tinyint	3,0		1

表 4-18 管理员菜单表(vm_menus)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	menu_name	资源名	varchar	20		
3	description	用户密码，md5 加密	varchar	255	√	
4	create_time	创建时间	int	10,0		
5	update_time	更新时间	int	10,0		
6	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
7	status	状态,1 为正常,2 为冻结	tinyint	3,0		1
8	key_prop	菜单 key	varchar	255	√	
9	pid	上级菜单的 id	bigint	19,0	√	
10	is_leaf	是否为叶子节点	tinyint	3,0		
11	icon	图标	varchar	255	√	

表 4-19 管理员角色表(vm_roles)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		

2	role_name	唯一用户名	varchar	20	
3	description	用户密码，md5 加密	varchar	255	
4	create_time	创建时间	int	10,0	
5	update_time	更新时间	int	10,0	
6	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0	1
7	status	状态,1 为正常,2 为冻结	tinyint	3,0	1
8	immutable	是否为内置不可变对象，1 为是，2 为否	tinyint	3,0	

表 4-20 角色权限关系表(vm_roles_auths_realation)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	create_time	创建时间	int	10,0		
3	update_time	更新时间	int	10,0		
4	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
5	status	状态,1 为正常,2 为冻结	tinyint	3,0		1
6	role_id	角色 id	bigint	19,0		
7	auth_id	权限 id	bigint	19,0		

表 4-21 角色菜单关系表(vm_roles_menus_realation)

序号	字段名称	字段描述	字段类型	长度	允许空	缺省值
1	id	自增主键	bigint	20,0		
2	create_time	创建时间	int	10,0		
3	update_time	更新时间	int	10,0		
4	is_deleted	状态,1 为 no,2 为 yes	tinyint	3,0		1
5	status	状态,1 为正常,2 为冻结	tinyint	3,0		1
6	role_id	角色 id	bigint	19,0		
7	menu_id	菜单 id	bigint	19,0		

第 5 章　模块设计

5.1　模块概述

本系统的主要主要采用微服务的架构，将功能划分为一个个微服务，每个为服务否可作为一个单独的项目开发，服务之间呈现松耦合，互不关联，最终实现多实例的分布式部署。服务包括：网关服务，注册中心，配置中心，用户微服务，管理员微服务，视频微服务，资源微服务，ws 微服务，前端页面模块，后端页面模块。

5.2　管理员微服务

5.2.1　功能描述

此微服务提供与管理员相关的 api，供页面模块访问，包括管理员的登录，管理员的鉴权，对管理员的 CURD，对角色的 CURD。

5.2.2　核心功能实现代码

```
//权限拦截
@Around("declareJoinPointExpression()&&@annotation(requiredAuth)")
public Object doAroundAdvice(ProceedingJoinPoint joinPoint, RequiredAuth requiredAuth)
throws Throwable {
    List<String> requiredAuthCodes = Lists.newArrayList(requiredAuth.auths());
    ServletRequestAttributes attributes = (ServletRequestAttributes)
RequestContextHolder.getRequestAttributes();
    HttpServletRequest request = attributes.getRequest();
    String token = request.getHeader(OnlineConstants.KEY_OF_ACCESS_TOKEN);
    List<String> haveAuthCodes = AuthCacheManager.getAuthCodes(token);
    if (CommonUtil.isNullObject(haveAuthCodes)) {
        haveAuthCodes = Lists.newArrayList();
    }
    boolean haveAuth = true;
    for (String requiredAuthCode : requiredAuthCodes) {
        if (!haveAuthCodes.contains(requiredAuthCode)) {
            haveAuth = false;
            break;
```

```
        }
    }
    if (!haveAuth) {
        throw new VmCommonException("AuthValidateAop admin accessToken : " + token + "
is have not auth ! required auth codes is : " + requiredAuthCodes + " , admin have auth
codes is : " + haveAuthCodes,
                VmCommonException.ErrorCode.ADMIN_HAVE_NOT_AUTH.getCode(),
                VmCommonException.ErrorCode.ADMIN_HAVE_NOT_AUTH.getMsg());
    }
    return joinPoint.proceed();
}
```

5.3 视频微服务

5.3.1 功能描述

此微服务提供与视频相关的 api，供页面模块访问，包括对视频的 CURD。

5.3.2 核心功能实现代码

```
//获取电影信息
@Override
public List<VmMoviesDto> getBackendMovies(VmMoviesQueryBean query, PageBean page) {
    return customVmMoviesMapper.getBackendMovies(query,
page).stream().parallel().map(vmMovies -> {
        return makeBackendMoviesDto(vmMovies);
    }).collect(toList());
}
//po2dto
private VmMoviesDto makeBackendMoviesDto(VmMovies vmMovies) {
    VmMoviesDto vmMoviesDto = new VmMoviesDto();
    vmMoviesDto.setAlias(vmMovies.getAlias());
    vmMoviesDto.setDescription(vmMovies.getDescription());
    vmMoviesDto.setDirectorId(vmMovies.getDirectorId());
    vmMoviesDto.setId(vmMovies.getId());
    vmMoviesDto.setImgUrl(vmMovies.getImgUrl());
    vmMoviesDto.setMovieTime(vmMovies.getMovieTime());
    vmMoviesDto.setName(vmMovies.getName());
```

```
    vmMoviesDto.setPosterUrl(vmMovies.getPosterUrl());
    vmMoviesDto.setReleaseTime(vmMovies.getReleaseTime());
    vmMoviesDto.setWatchNum(vmMovies.getWatchNum());
    vmMoviesDto.setScore(vmMovies.getScore());
    vmMoviesDto.setCreateTime(vmMovies.getCreateTime());
    vmMoviesDto.setUpdateTime(vmMovies.getUpdateTime());
    vmMoviesDto.setStatus(vmMovies.getStatus());
    vmMoviesDto.setDescription(vmMovies.getDescription());
    return vmMoviesDto;
}
```

5.4 资源微服务

5.4.1 功能描述

对项目的资源文件进行保存和获取；多个微服务实例注册到注册中心，其他为服务可以通过 feign 调用其 api 接口，实现图片的保存、剪切，视频的保存、获取等。

5.4.2 核心功能实现代码

```
//裁剪图片
@Override
@Transactional
public Long cutUploadedImgFile(VmFilesDto vmFilesDto) throws Exception {
    logger.info("cutUploadedImgFile vmFilesDto is : {} !", vmFilesDto);
    VmFiles vmFiles = this.getUsableVmFilesById(vmFilesDto.getFileId());
    //delete temp file
    String filePath = vmBaseConfig.getSrcImgPath();
    String fileName = vmFiles.getFilename();
    final String canNotChangePathName = filePath + fileName;
    String copyFilePathName = filePath + "copy_" + fileName;
    IOUtil.copyFile(canNotChangePathName, copyFilePathName);
    IOUtil.deleteFiles(canNotChangePathName);
    //cut img
    try {
        ImageUtil.crop(copyFilePathName,
                canNotChangePathName,
```

```
                vmFilesDto.getX(),
                vmFilesDto.getY(),
                vmFilesDto.getWidth(),
                vmFilesDto.getHeight());
        logger.info("cutUploadedImgFile crop file is : {} !", canNotChangePathName);
    } catch (Exception e) {
        throw e;
    }
    //zoom img
    String[] versions = vmFilesDto.getVersions().split(",");
    int originalWidth = vmFilesDto.getWidth();
    int originalHeight = vmFilesDto.getHeight();
    Lists.newArrayList(versions).stream().parallel().forEach((version) -> {
        int intVersion = Integer.valueOf(version);//width
        String targetFilePathName = filePath + version + "_" + fileName;
        try {
            //get zoom
            Double zoom = originalHeight * 1.0 / originalWidth * 1.0;
            int height = (int) (zoom * intVersion);
            ImageUtil.resize(canNotChangePathName, targetFilePathName, intVersion,
height);
            logger.info("cutUploadedImgFile resize file is : {} !", targetFilePathName);
        } catch (Exception e) {
            e.printStackTrace();
        }
    });
    return vmFiles.getId();
}
```

5.5 用户微服务

5.5.1 功能描述

此微服务提供与用户相关的 api，供页面模块访问，包括对用户的 CURD。

5.5.2 核心功能实现代码

```
//获取指定 id 的用户基本信息
```

```
@Override
public VmUsersDto getUserBasicInfo(Long userId) {
    //获取指定 id 的 user
    VmUsers dbUser = this.getUsableUserById(userId, BasePo.Status.NORMAL,
BasePo.IsDeleted.NO);
    if (isNullObject(dbUser) || VmUsers.IsDeleted.isDeleted(dbUser.getIsDeleted())) {
        throw new VmUsersException("getUserBasicInfo user is not exits! userId is : "
+ userId,
                VmUsersException.ErrorCode.USER_IS_NOT_EXITS.getCode(),
VmUsersException.ErrorCode.USER_IS_NOT_EXITS.getMsg());
    }
    return makeVmUsersDto(dbUser);
}
```

5.6 配置中心

5.6.1 功能描述

提供其他服务配置文件，实现集中配置

5.6.2 核心功能实现代码

```
//服务入口
@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
public class ConfigApplication {
    public static void main(String[] args) {
        SpringApplication springApplication = new
SpringApplication(ConfigApplication.class);
        springApplication.run(args);
    }
}
```

5.7 注册中心

5.7.1 功能描述

提供其他服务实例注册服务，类似于 zookeeper 等

5.7.2 核心功能实现代码

```
//服务入口
@EnableEurekaServer
@SpringBootApplication
public class EurekaApplication {
    public static void main(String[] args) {
        SpringApplication springApplication = new
SpringApplication(EurekaApplication.class);
        springApplication.run(args);
    }
}
```

5.8 网关服务

5.8.1 功能描述

转发前台请求到指定的微服务，并且具有熔断，负载均衡等功能.

5.8.2 核心功能实现代码

```
/**
 * iphash 算法实现负载均衡
 */
public class IpHashRule extends AbstractLoadBalancerRule {
    private static Logger log = LoggerFactory.getLogger(IpHashRule.class);
    public Server choose(ILoadBalancer lb, Object key) {
        if (lb == null) {
            log.warn("no load balancer");
            return null;
        }
        Server server = null;
        int count = 0;
        while (server == null && count++ < 10) {
            List<Server> reachableServers = lb.getReachableServers();
            List<Server> allServers = lb.getAllServers();
            int upCount = reachableServers.size();
            int serverCount = allServers.size();

            if ((upCount == 0) || (serverCount == 0)) {
```

```
                log.warn("No up servers available from load balancer: " + lb);
                return null;
            }
            int nextServerIndex = ipUserHash(serverCount);
            server = allServers.get(nextServerIndex);
            if (server == null) {
                /* Transient. */
                Thread.yield();
                continue;
            }
            if (server.isAlive() && (server.isReadyToServe())) {
                return (server);
            }
            // Next.
            server = null;
        }
        if (count >= 10) {
            log.warn("No available alive servers after 10 tries from load balancer: "
                    + lb);
        }
        return server;
    }
    private int ipUserHash(int serverCount) {
        String userIp = getRemoteAddr();
        try {
            userIp = InetAddress.getLocalHost().getHostAddress();
        } catch (UnknownHostException e) {
        }
        int userHashCode = Math.abs((userIp).hashCode());
        return userHashCode % serverCount;
    }
    private String getRemoteAddr() {
        RequestContext ctx = RequestContext.getCurrentContext();
        HttpServletRequest request = ctx.getRequest();
```

```
        String remoteAddr = "0.0.0.0";
        if (request.getHeader("X-FORWARDED-FOR") != null) {
            remoteAddr = request.getHeader("X-FORWARDED-FOR");
        } else {
            remoteAddr = request.getRemoteAddr();
        }
        return remoteAddr;
    }
    @Override
    public Server choose(Object key) {
        return choose(getLoadBalancer(), key);
    }

    @Override
    public void initWithNiwsConfig(IClientConfig clientConfig) {
        // TODO Auto-generated method stub
    }
    public static void main(String[] args) {
        String localIp = "127.0.0.1";
        System.out.println(Math.abs((localIp).hashCode()) % 5);
    }
}
```

5.9 ws 微服务

5.9.1 功能描述

提供其他服务 websocket 通信支持

5.9.2 核心功能实现代码

```
//声明 mq 管道
public interface AdminReceiverChannel {
    String ADMIN_INPUT = "adminInput";
    @Input(AdminReceiverChannel.9ADMIN_INPUT)
    SubscribableChannel adminInput();
}
```

5.10 前台界面

5.10.1 功能描述

该模块属于MVC中属于View层，界面采用reactjs+webpack实现，通过nodejs服务器框架express部署。但是不提供api。页面需要通过发送ajax请求其他微服务提供的api获取数据，然后展示。

5.10.2 核心代码

```
render: function () {
    //set now page's props
    return (
        <div id="index">
            <HashRouter>
                <div>
                    {
                        /*头部*/
                    }
                    <Head/>
                    <Switch>
                        <Route exact path='/'
                               render={() => (
                                   <MovieListPage tagGroupSource="/tagGroup/list"
                                                  movieSource="/movie/list"/>
                               )}/>
                        <Route exact path='/movie/list'
                               render={() => (
                                   <MovieListPage tagGroupSource="/tagGroup/list"
                                                  movieSource="/movie/list"/>
                               )}/>
                        <Route exact path='/movie/:movieId' component={MovieInfoPage}/>
                        <Route exact path='/filmmaker/:filmmakerId'
component={FilmmakerInfoPage}/>
                        {/*包括/user/online/basicInfo,/user/online/resetPwd 等*/}
                        <Route path='/user/online' component={UserPage}/>
                    </Switch>
                    {
```

```
                    /*尾部*/
                }
                <Tail/>
                {
                    /*信息弹出框*/
                }
                <MsgDialog ref="msg_dialog"/>
                {
                    /*信息弹出框*/
                }
                <Loading/>
            </div>
        </HashRouter>
    </div>
  );
}
```

5.10.3 核心界面实现

界面包括：登录界面，浏览界面，视频详情/播放界面，电影人详细界面，如图 5-1 至图 5-4。

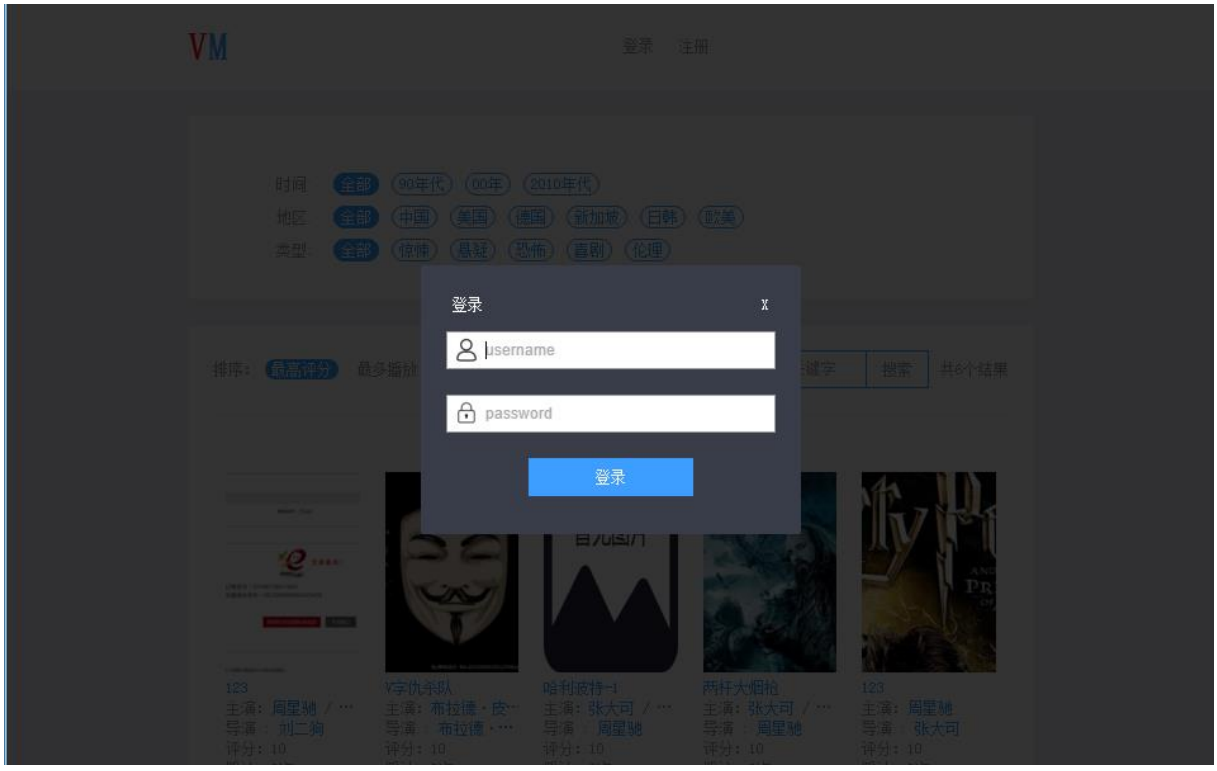
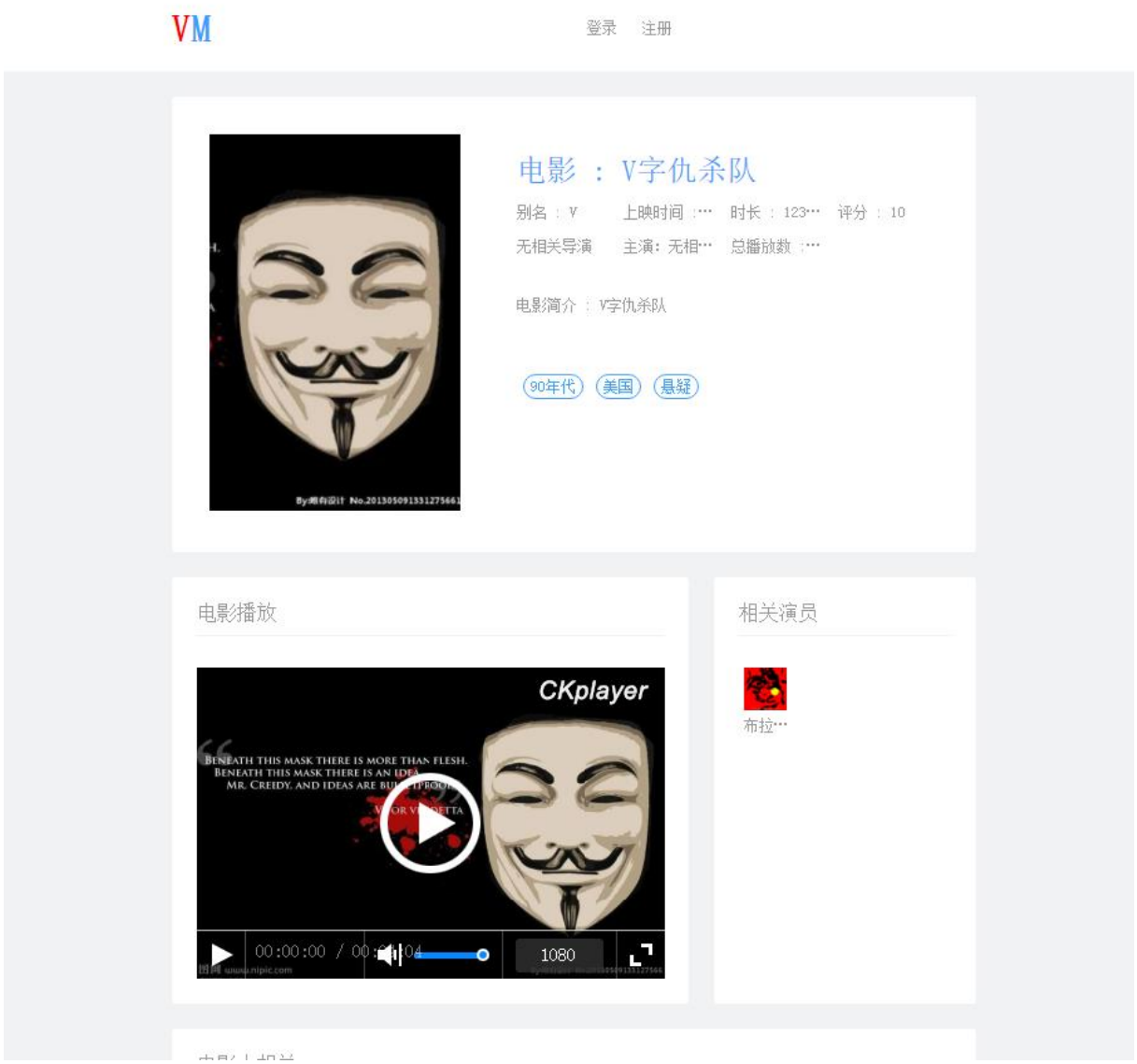
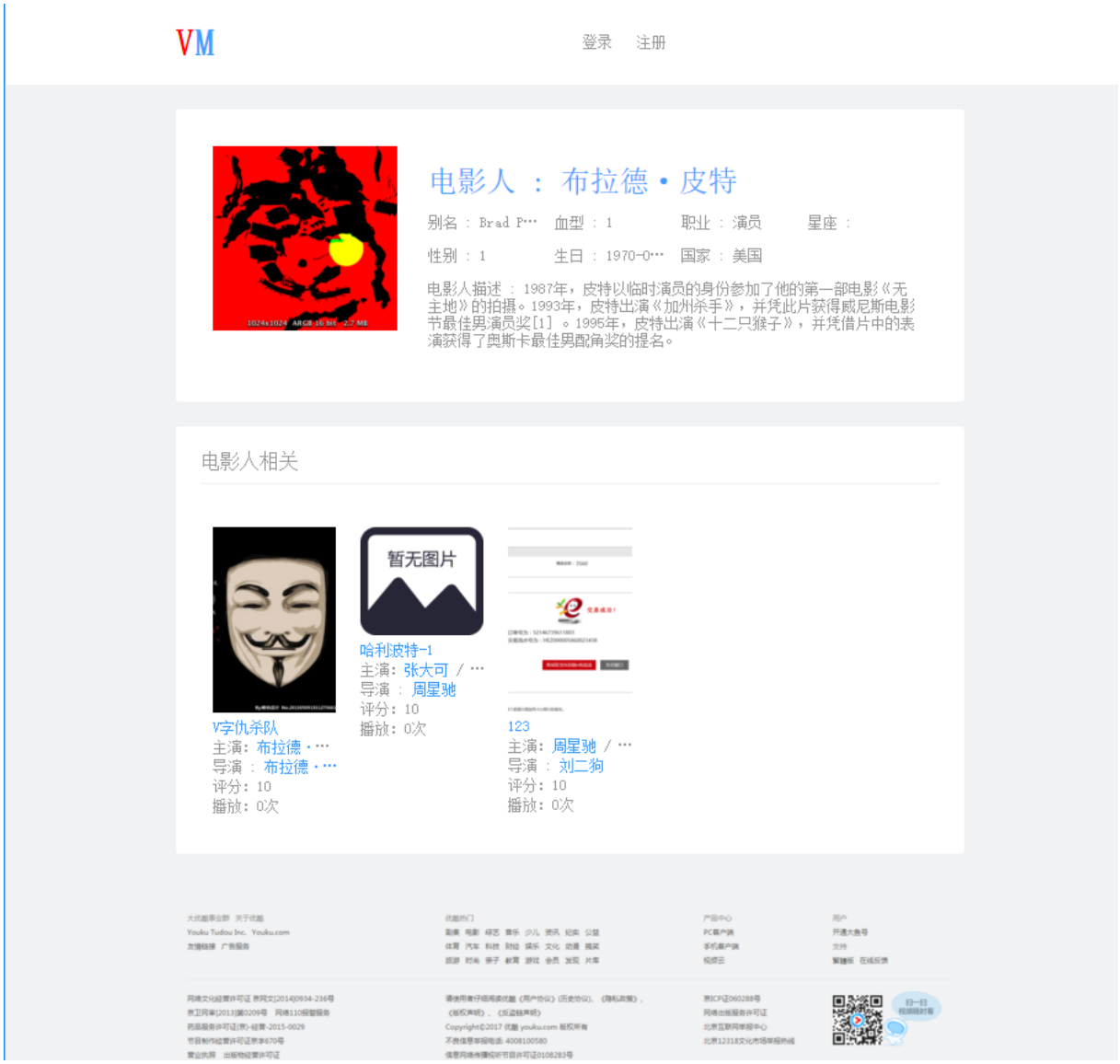


图 5-1 前台用户登录界面



5-2 前台视频详情/播放界面



5-3 前台电影人详情界面

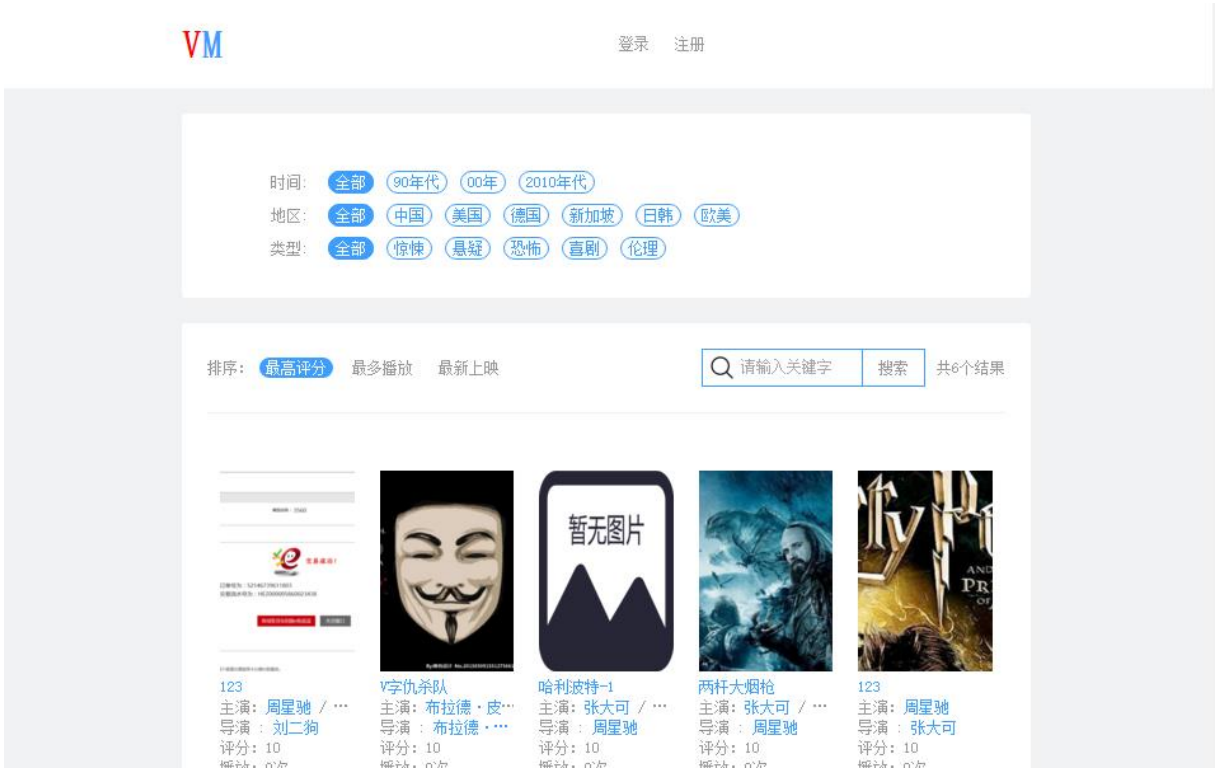


图 5-4 前台用户浏览界面图图

5.11 后台界面

5.11.1 功能描述

该模块属于 MVC 中属于 View 层，该模块属于 MVC 中属于 View 层，界面采用 reactjs+webpack 实现，通过 nodejs 服务器框架 express 部署。但是不提供 api。页面需要通过发送 ajax 请求其他微服务提供的 api 获取数据，然后展示。

5.11.2 核心代码

```
render: function () {
    //set now page's props
    const {collapsed, loading,siderCurrtWidth, collapsedWidth} = this.state;
    const antIcon = <Icon type="loading" style={{fontSize: 24}} spin/>;
    return (
        <HashRouter>
            <Spin indicator={antIcon}
                  spinning={loading}>
                <Layout>
                    {/*登录框*/}
                    <LoginDialog ref="login_dialog"/>
```

```
                    <Sider
                        collapsible
                        trigger={null}
                        collapsed={collapsed}
                        onCollapse={this.onCollapse}
                        width={siderCurrtWidth}
                        collapsedWidth={collapsedWidth}
                        style={{overflow: 'auto', height: '100vh', zIndex: "999",
position: 'fixed', left: 0}}
                    >
                        {/*nav*/}
                        <Nav/>
                    </Sider>
                    <Layout style={{marginLeft: siderCurrtWidth}}>
                        <Header style={{background: '#fff', padding: 0}}>
                            {/*head*/}
                            <Head/>
                        </Header>
                        <Content style={{margin: '0 16px', overflow: 'initial'}}>
                            <Breadcrumb style={{margin: '16px 0'}}>
                                <Breadcrumb.Item>User</Breadcrumb.Item>
                                <Breadcrumb.Item>Bill</Breadcrumb.Item>
                            </Breadcrumb>
                            <div style={{paddingLeft: 24, paddingRight: 24, background:
'#fff'}}>
                                <Routes/>
                            </div>
                        </Content>
                        <Footer style={{textAlign: 'center'}}>
                            Vm backend ©2016 Created by Zhangke
                        </Footer>
                    </Layout>
                </Layout>
            </Spin>
```

```
        </HashRouter>
    );
}
```

5.11.3 核心页面

界面包括：管理员登录界面，管理统计界面，用户管理界面，电影管理界面，管理员管理界面，修改界面如图 5-5 至图 5-10。

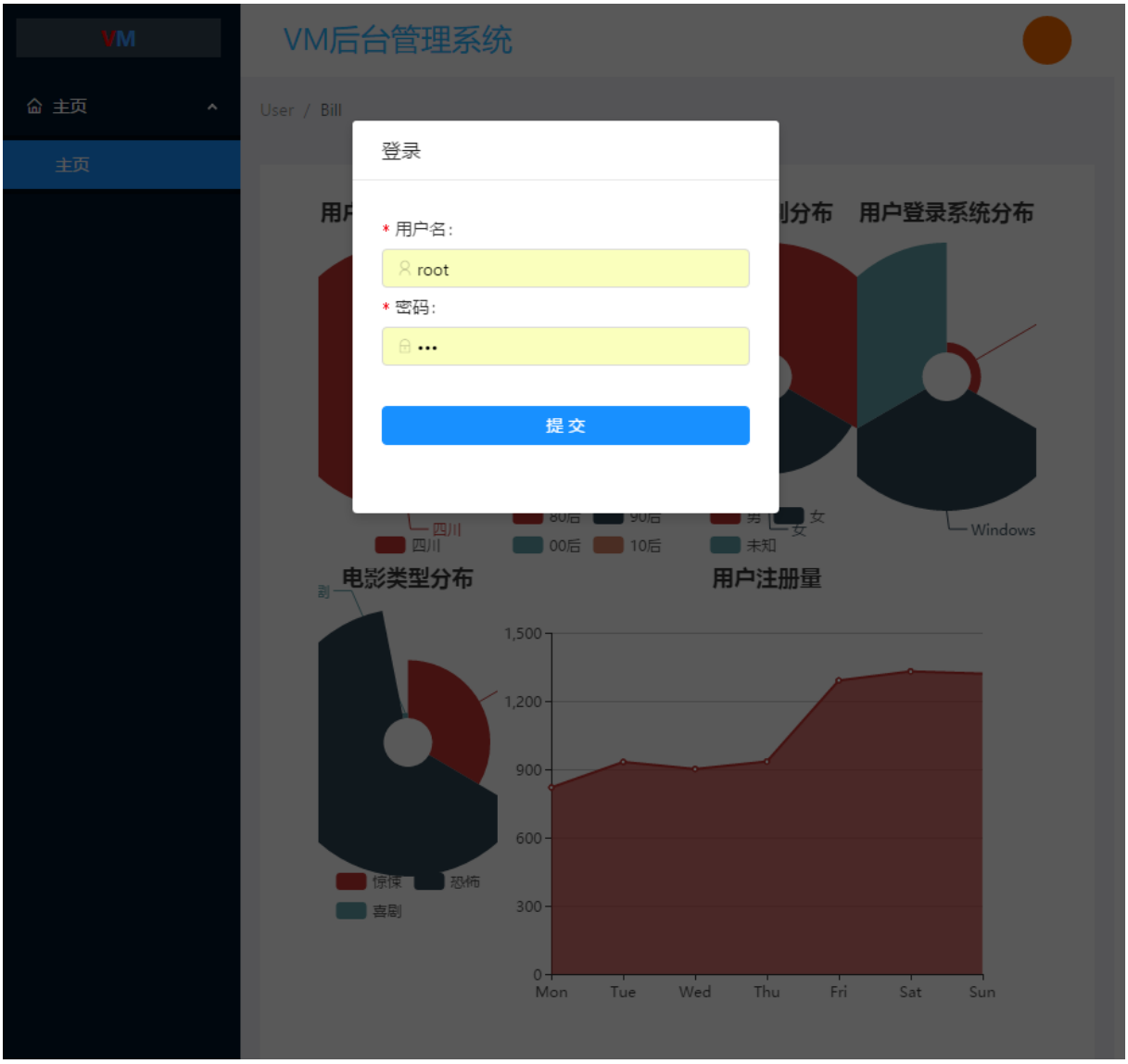


图 5-5 后台管理员登录界面

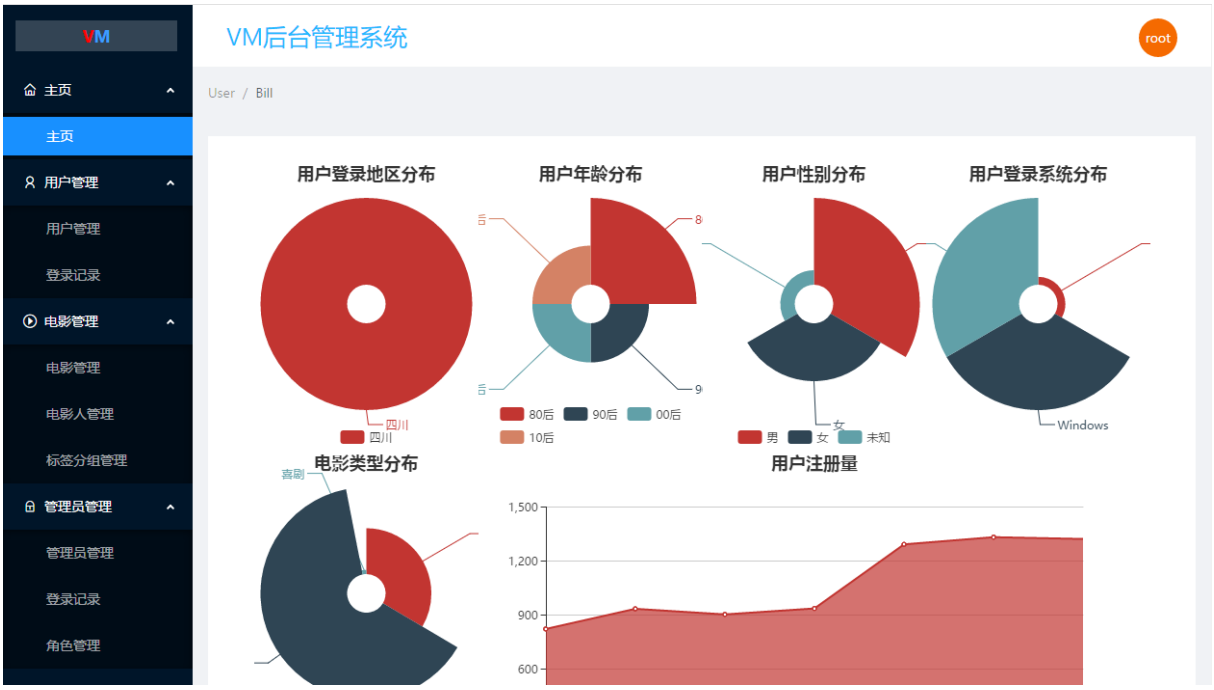


图 5-6 后台管理统计界面

图 5-7 后台用户管理界面

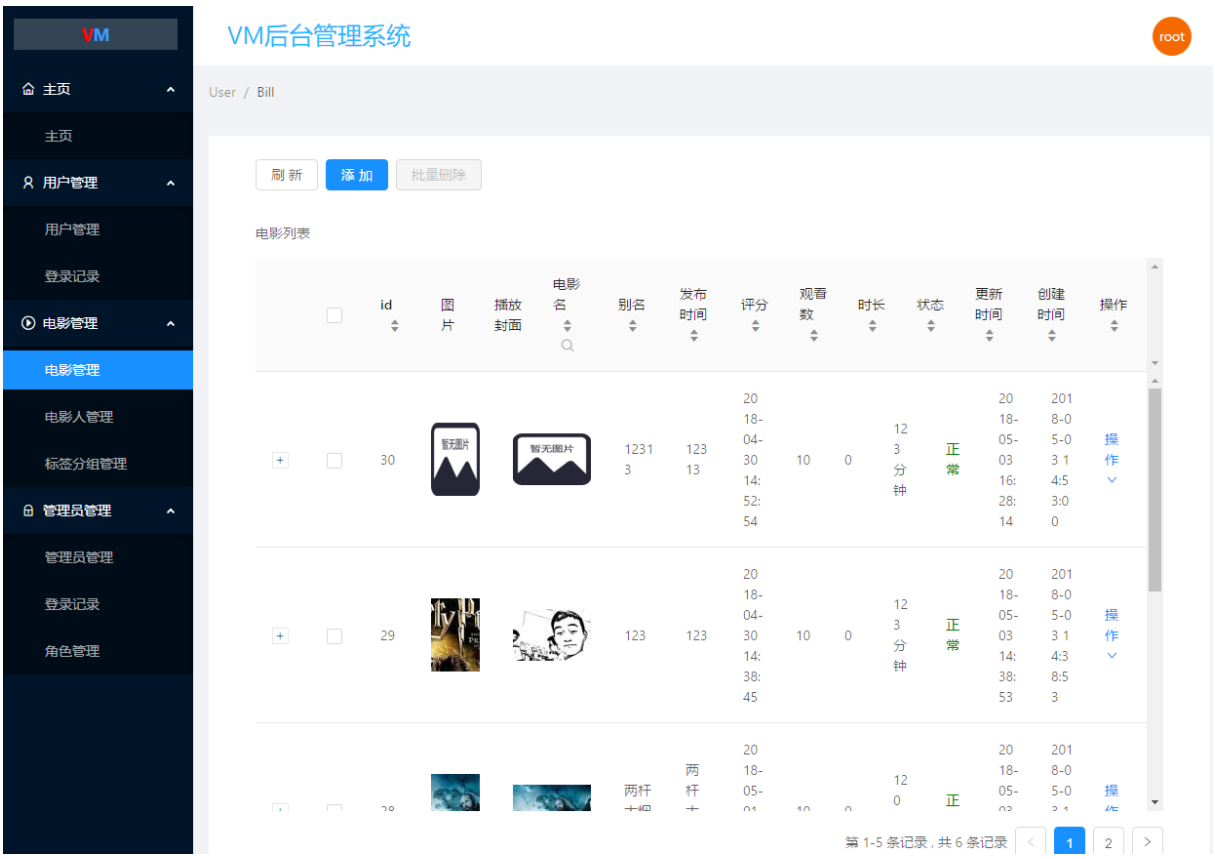


图 5-8 后台电影管理界面

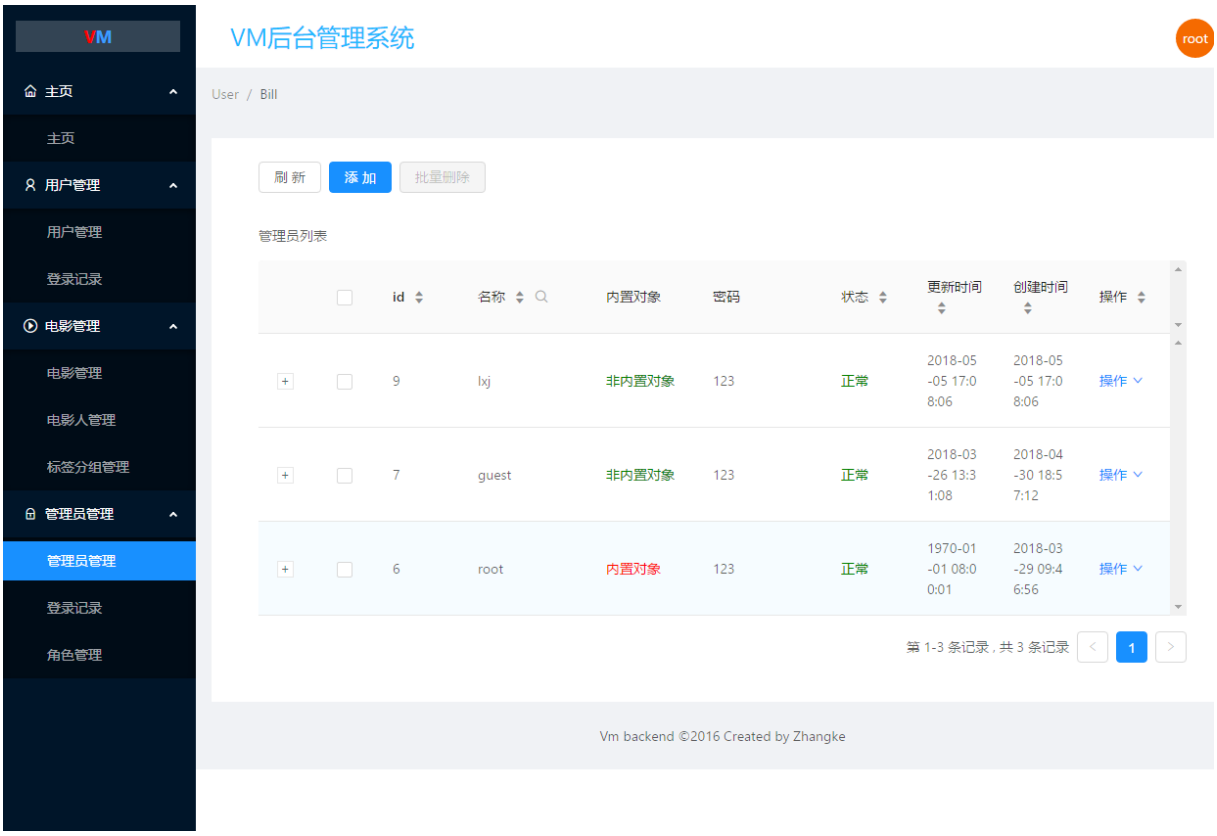


图 5-9 后台管理员管理界面


修改用户信息
id:
49
* 生日:
2018-05-09
* 状态:
冻结
* 用户名:
root1
* 密码:
123
* 性别:
女
创建时间:
2018-05-04
最后更新时间:
2018-05-04
* 简介:
asd
提交

图 5-10 后台修改界面

第 6 章　部署与应用

本系统部署在 centos 系统之上，需要多台服务器构建一个微服务集群，保证服务器的稳点和高效；测试采用自动化测试和单元测试的方式，自动测试使用的测试工具为 loadrunner11，单元测试采用 Mock。对系统的用户端和管理端分别进行了测试。测试项目包含了常规详细信息，工作负载特性，性能概述，HTTP 响应概要，最差 URL(根据平均响应时间)，每秒点击次数，吞吐量，事务摘要，平均事务响应时间，LoadRunner 对象等。

6.1　系统测试

本系统采用 loadrunner11 自动化测试，测试内容包括：每秒事物总数，事务摘要，最大 URL，消耗 URL 的大多数资源，业务流程，工作负载特性，最差 URL，可见图 6-1 至 6-7。

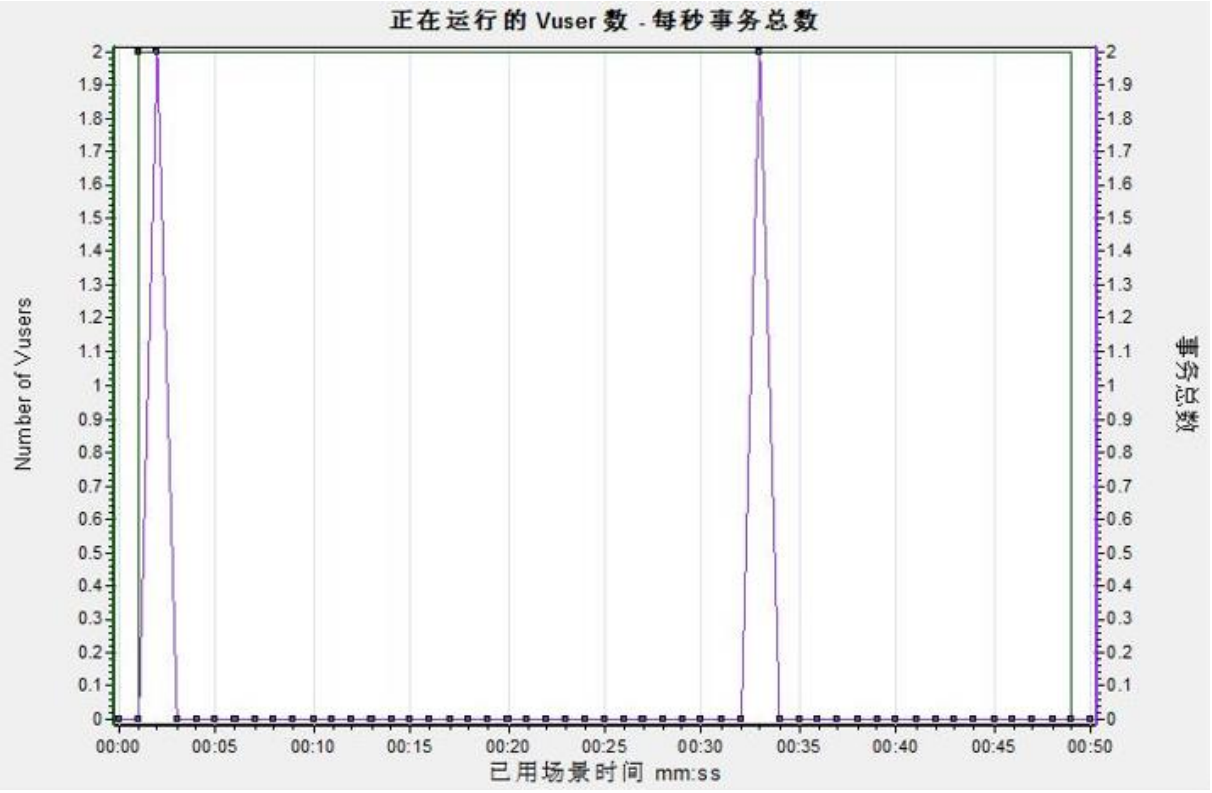
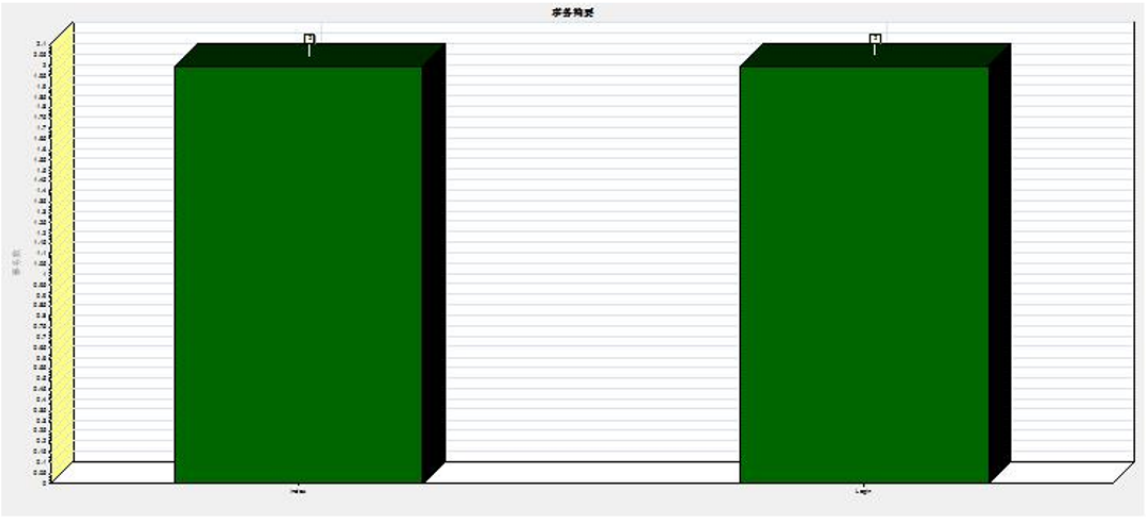


图 6-1 每秒事物总数

事务摘要

标题	事务摘要
当前结果	c:\Users\Public\0330\0330.lrr
筛选器	(未 包括思考时间)
分组方式	



颜色	比例	度量
	1	Pass

描述: 显示通过、失败、停止或因为出错而结束的事务的数目。

图 6-2 事务摘要

最大 URL(按平均 KB 值计算)

URL 名称	父事务名称	平均大小	计数	总计
http://192.168.0.101:8080/backend/index	index		1	873.849
http://192.168.0.101:8080/users/img/d45b43b1-ae90-4d5a-ba00-12cfa1941a43	Login		1	41.103
http://192.168.0.101:8080/users/img/fb5a8dc3-ecbb-4bf7-af84-9224eedecbcd	Login		1	41.103
http://192.168.0.101:8080/users/img/a8978e74-7356-491a-8774-b2f3bb7867a2	Login		1	41.103

图 6-3 最大 URL

消耗 URL 的大多数资源

URL 名称	父事务名称	服务器总时间	计数	总计
http://192.168.0.101:8080/admins/login	Login		1	19.442
http://192.168.0.101:8080/backend/index	index		1	0.396
http://192.168.0.101:8080/users/list	Login		1	0.129
http://192.168.0.101:8080/users/img/fb5a8dc3-ecbb-4bf7-af84-9224eedecbcd	Login		1	0.028
http://192.168.0.101:8080/users/img/c9209589-f43a-414a-9466-1dc556900fad	Login		1	0.024
http://192.168.0.101:8080/users/img/1545006b-573b-4965-a9f7-adac9d6df47e	Login		1	0.018
http://192.168.0.101:8080/users/img/d45b43b1-ae90-4d5a-ba00-12cfa1941a43	Login		1	0.017
http://192.168.0.101:8080/backend/fonts/lato/lato-regular.woff	index		1	0.017
http://192.168.0.101:8080/users/img/65307c85-bad3-4308-99bd-b1bc61ccf020	Login		1	0.016
http://192.168.0.101:8080/users/img/d8c231ef-a800-4f12-afed-e5e5274e8fe5	Login		1	0.015

图 6-4 消耗 URL 的大多数资源

业务流程

组名	脚本名称	并发 Vuser 数	总 Vuser 数的 %	每小时事务数	开始时间
zhangke0330	zhangke0330	2	100	288	2018\3\30 星期五 16:16
	总计:	2	100%		

图 6-5 业务流程

工作负载特性

度量	
最大运行 Vuser 数	2
每秒平均点击次数	1.686
总点击次数	86
每秒通过的事务总数	0.078
每分钟通过的事务总数	4.68
事务总数	2

图 6-6 工作负载特性

最差 URL(根据平均响应时间)

URL 名称	父事务名称	URL 的层细分	计数	总计
http://192.168.0.101:8080/admins/login	Login		1	19.444
http://192.168.0.101:8080/backend/index	index		1	0.812
http://192.168.0.101:8080/users/list	Login		1	0.13
http://192.168.0.101:8080/users/img/fb5a8dc3-ecbb-4bf7-af84-9224eedecbcd	Login		1	0.039
http://192.168.0.101:8080/users/img/c9209589-f43a-414a-9466-1dc556900fad	Login		1	0.038
http://192.168.0.101:8080/users/img/d45b43b1-ae90-4d5a-ba00-12cfa1941a43	Login		1	0.031
http://192.168.0.101:8080/users/img/65307c85-bad3-4308-99bd-b1bc61ccf020	Login		1	0.03
http://192.168.0.101:8080/users/img/d8c231ef-a800-4f12-afed-e5e5274e8fe5	Login		1	0.03
http://192.168.0.101:8080/backend/fonts/lato/lato-regular.woff	index		1	0.03

图 6-7 最差 URL

6.2 部署与应用环境

6.2.1 应用开发平台

本系统是在 Windows 系统环境下开发,使用 B/S 架构，主要开发语言为 JAVA 和 jsx，并使用目前流行的微服务架构+前后端分离。系统部署在 centos 操作系统中，并且采用分布式的部署，开发环境配置清单见表 6-2 所示。

表 6-2 开发环境软件配置一览表

序号	软件名称	软件版本	备注
1	IntelliJ IDEA	V2017.1	IDE 开发工具
2	Sun JDK	V1.8.x	提供编译及执行 Java 程序
3	Apache Tomcat	V8.x	提供 J2EE 应用运行 Web 环境
4	Mysql	V6.x	数据库软件
5	Reactjs	V15.x	前端页面编写语言
6	Webpack	V3.x	前端代码编译工具
7	Centos	V7.x	Linux 服务器

8	Jenkins	V2.7.x	CI 工具
9	Npm	V5.5.x	前端包管理工具
10	Springboot	V1.5.7	开箱即用
11	Springcloud	VDalston.SR3	分布式成套解决方案
12	Rabbitmq	V3.6.12	消息队列
13	Redis	V4.0.2	高速缓存
14	Docker	V18.03	容器
15	Docker-compose	V1.9.0	容器编排
16	Websocket	-	双工通信

系统部署步骤说明:

1、开通服务器 A，B （可以开通 aliyun 的 Ecs 服务，至少需要两台服务器，配置为 1 核+2g+50g 硬盘）。

2、构建 A,B 服务器内网环境（如果是 aliyun 服务，且是经典网络，需要开通高速内网）。

3、安装环境：mysql，docker，docker-compose，nodejs，npm，rabbitmq，redis，git，jdk。

4.配置环境：如配置/etc/profile；/etc/hosts；各个服务的配置文件

5.A 服务器启动服务：mysql，docker，rabbitmq， redis

6.测试机打包微服务为 docker 镜像，并且推送至 aliyun 镜像仓库。

7.服务器 A 通过 docker 拉取镜像 eureka,config-server,provider-movie，provider-ws

8.通过 git 拉取分支 prodServer00

9.通过 docker-compose 配置文件启动相关微服务

10.待服务器的 eureka 和 config-server 服务启动成功后，服务器 B 通过 docker 拉取镜 provider-user, provider-admin, provider-src, gateway

10.通过 git 拉取分支 prodServer01

11.通过 docker-compose 配置文件启动相关微服务

12.测试机通过 webpack 打包前端代码,并且上传 vm-frontend 和 vm-backend 的 dist 至服务器 B。如上传到目录/usr/local/src/vm-web/vm-frontend/dist

13.通过 npm 或者 yarn 安装 express 服务器所需包（安装在各自的 dist 目录下）。

14.安装 forever（可使服务器后台可靠的运行，且为热部署）。

15.执行 forever server.config.js 运行前端页面。

12.执行访问：

http://{A.IP}:1110 访问 eureka 注册中心

http://{B.IP}:5550/XX 访问相关接口

http://{B.IP}:3999 访问后端页面

http://{B.IP}:3000 访问前端页面

第 7 章　结论

网上微视频系统的实现过程充分体现了软件工程的开发原理。从最初的项目可行性研究到最后的项目部署测试，每一个步骤都需要投入大量的工作和技术准备。这其中包括大量的阅读与软件开发相关的书籍文献、了解整个编码过程中需要的开发技术和学习多种软件用图的绘画。经过自己的不懈努力和导师的悉心教导，网上微视频系统已经开发完成，基本实现了系统的各项功能，能给用户带来良好的操作体验。

一个软件项目的设计与实现过程，是一个不断学习和完善的过程。我们需要扎实的理论基础，还要有过硬的实践能力。通过本次的毕业设计，我不仅充分发挥和巩固了书本上所学的专业知识，还了解了之前没有接触过的开发技术，比如利用 springcloud 构建了微服务的架构，在多个服务器中分布式的部署微服务的实例，比但单节点部署的系统，在系统稳定性，可拓展性等方面有很大的提升；利用了 docker 对微服务进行了编排部署，提高了开发环境部署到生产环境的能力，减少了由于开发环境和生产环境差异造成的部署困难，实现了快速部署；通过 aop 的方式实现严密的权限控制和登录控制，提高了系统的安全性；利用 reactjs 编写前段页面，实现了组件化的开发，降低了开发难度和维护成本；利用 webpack 对前段代码进行打包混淆，代码压缩，多环境打包等；aop 实现异常的统一捕获和日志的统一输出；充分利用 cdn 加速的优势，将页面延时从 10s 降低至 2s。

虽然我的设计已经完成，但是还有很多不尽如人意的地方。比如视频的检索方式上，可以使用 solr 全文检索的方式；在跨微服务操作数据库时没有找到可行的分布式事务解决方案。这当然和我自身的能力有关，但我内心仍然充满着成功的喜悦。能够独立完成一套设计，就是一次成功，也是我进入职场前一次宝贵的软件开发经验。

参考文献

[1] 陈屹.深入 React 技术栈[M].人民邮电出版社,2016.
[2] 周立.SpringCloud 与 Docker 微服务架构[M].电子工业出版社,2017.
[3] (美)沙鲁巴·夏尔马(Sourabh Sharma) 著；卢涛 译.Java 微服务[M].电子工业出版社, 2017.
[4] Sanjay，Patni 著；郭理勇 译.RESTful API 开发实战[M].清华大学出版社, 2018.
[5] Adrian Mouat. Docker 开发指南[M].人民邮电出版社,2017.
[6] 吴浩麟.深入浅出 Webpack [M].电子工业出版社,2018.
[7] 金蓓弘.分布式系统 概念与设计[M].机械工业出版社,2018.
[8] (美)Flavio Junqueira. ZooKeeper.分布式过程协同技术详解[M].机械工业出版社, 2016.
[9] (美)埃克尔. Java 编程思想第 4 版[M].机械工业出版社,2007.
[10] 结城浩.图解设计模式[M].人民邮电出版社，2018.
[11] 徐超. React 进阶之路[M].清华大学出版社, 2018.
[12] 兰洋，温迎福.测试实践丛书 持续集成实践[M].电子工业出版社,2018.
[13] CSDN. Spring Cloud综合实战 - 基于TCC补偿模式的分布式事务[EB/OL]. https://blog.csdn.net/solarison/article/details/68061157,2017-03-29.
[14] 博客园. 微服务架构[EB/OL]. https://www.cnblogs.com/imyalost/p/6792724.html,2017-05-1.
[15] CSDN.MQ选型对比 RabbitMQ RocketMQ ActiveMQ Kafka. https://blog.csdn.net/oMaverick1/article/details/51331004,2016-05-06.
[16] 51CTO博客.docker知识全面讲解. http://blog.51cto.com/10546390/1855652,2017-09-23.
[17] 博客园.分布式系统的架构思路. https://www.cnblogs.com/chulung/p/5653135.html,2016-07-08.
[18] 天码营. Spring Boot——开发新一代Spring Java应用. https://www.tianmaying.com/tutorial/spring-boot-overview,2017-08-05.