武汉理工大学毕业设计（论文）

**基于Spring Cloud和Vue的家谱服务系统 的设计与实现**

学院（系）：计算机与人工智能学院

专业班级： 软件zy1901班

学生姓名： 李春雄

指导教师： 王云华

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：

2023年06月12日

学位论文版权使用授权书

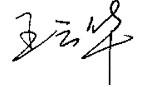
本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

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🗹。

（请在以上相应方框内打“√”）

作者签名： 2023年06月12日

导师签名： 2023年06月12日

# 摘 要

传统家谱文献以纸质制成，记录内容包含众多，编制格式复杂，随着世代增加，家谱中的人物呈指数级增长，使得家谱在传播、利用及查询上存在很大局限[1]。为了满足当代家族对家谱管理和文化传承的迫切需求，本文介绍了基于Spring Cloud和Vue框架实现的家谱服务系统。

该系统旨在设计和开发一款功能多样、操作简单的综合性数字化家族谱系服务平台。为了实现这一目标，系统被细分为移动端前台应用系统和桌面端后台管理系统。前台系统为家族成员提供了丰富的家族数据呈现功能，如家族成员树、世代成员列表、家族大事记和成员社交动态等，进而增进了家族成员之间的血缘联结。后台系统为管理员提供了管理家族信息的统一化平台，包括系统权限管理和家族成员管理等功能。

系统的前台用户页面是基于Vue框架实现的现阶段最流行的SPA（单页应用）。单页面应用的特点是所有页面的加载全部在一个窗口中进行，并通过路由实现页面跳转[2]。界面设计清晰简洁，交互逻辑性强，为用户提供友好的交互体验和界面展示。用户可以轻松地浏览家族成员信息和了解家族历史，同时还可以与其他家族成员进行分享和互动，良好的交互设计有助于增强家族成员之间的联系和认同感。

后台服务系统采用了面向服务的体系架构（SOA），分布式SOA架构的特点有：分布式、可复用、拓展灵活、松耦合[3]。通过微服务化的体系架构，实现了各个业务模块的松耦合，从而提高了系统的可维护性、稳定性和高可用性，这种分布式的架构使得系统更加稳定可靠，能够处理大量的数据和用户请求。

综上所述，本家谱服务系统为家族提供了一个全面的数字化管理平台。通过前台系统和后台系统的结合，家族成员可以轻松地了解和共享家族的历史与信息，家族管理员可以高效地管理和维护家族谱系，并且系统具备良好的用户体验、可扩展性和可维护性，为当下家族的发展和文化传承提供了有力支持。

关键词：家谱数字化；家谱系统；Spring Cloud；Vue；OAuth

# Abstract

Traditional genealogical documents are made of paper, containing a multitude of recorded information and complex formatting. As generations pass, the number of individuals recorded in the genealogy grows exponentially, resulting in significant limitations in terms of dissemination, utilization, and retrieval of information. In order to meet the urgent needs of contemporary families for genealogy management and cultural inheritance, this article introduces a genealogy service system implemented based on the Spring Cloud and Vue frameworks.

This system aims to design and develop a comprehensive digital family genealogy service platform with diverse functions and simple operations. To achieve this goal, the system is divided into a mobile frontend application system and a desktop backend management system. The frontend system provides family members with rich functions for presenting family data, such as family member trees, generational member lists, family memorabilia, and member social dynamics, thereby enhancing the bloodline connection between family members. The backend system provides administrators with a unified platform for managing family information, including system permission management and family member management functions.

The frontend user interface of the system is implemented using the Vue framework, which is currently the most popular SPA (Single-Page Application) framework. The characteristic of a single-page application is that all pages are loaded within a single window and page transitions are achieved through routing. The interface design is clear and concise, with strong logical interactivity, providing users with a friendly interactive experience and interface display. Users can easily browse family member information, learn about family history, and engage in sharing and interaction with other family members. The well-designed interactions help strengthen the connections and sense of belonging among family members.

The backend service system adopts a Service-Oriented Architecture (SOA), with distributed SOA architecture characterized by distribution, reusability, flexible scalability, and loose coupling. Through a microservices-based architecture, the system achieves loose coupling of various business modules, thereby improving system maintainability, stability, and high availability. This distributed architecture makes the system more stable and reliable, capable of handling large amounts of data and user requests.

In conclusion, this genealogy service system provides a comprehensive digital management platform for families. Through the combination of the frontend and backend systems, family members can easily understand and share the history and information of their family, while family administrators can efficiently manage and maintain the family genealogy. The system has a good user experience, scalability, and maintainability, providing strong support for the development and cultural inheritance of contemporary families.

**Key Words：**Genealogy digitization；Family tree system；Spring Cloud；Vue；OAuth

目 录

[摘 要 I](#_Toc137028073)

[Abstract II](#_Toc137028074)

[第1章 绪论 1](#_Toc137028075)

[1.1 研究现状和意义 1](#_Toc137028076)

[1.2 国内外现状分析 2](#_Toc137028077)

[1.2.1 国内研究现状 2](#_Toc137028078)

[1.2.2 国外研究现状 3](#_Toc137028079)

[1.3 研究内容和目标 3](#_Toc137028080)

[1.4 本文的组织结构 4](#_Toc137028081)

[第2章 相关技术 5](#_Toc137028082)

[2.1 系统技术栈 5](#_Toc137028083)

[2.2 Vue 5](#_Toc137028084)

[2.3 MySQL 6](#_Toc137028085)

[2.4 Spring Cloud 6](#_Toc137028086)

[2.5 Spring Security 7](#_Toc137028087)

[2.6 OAuth2 7](#_Toc137028088)

[2.7 本章小结 7](#_Toc137028089)

[第3章 系统分析 8](#_Toc137028090)

[3.1 功能性需求分析 8](#_Toc137028091)

[3.2 非功能性需求分析 9](#_Toc137028092)

[3.3 系统用例分析 10](#_Toc137028093)

[3.3.1 系统总用例 10](#_Toc137028094)

[3.3.2 用户模块用例 11](#_Toc137028095)

[3.3.3 家族人员模块用例 12](#_Toc137028096)

[3.3.4 家族相册模块用例 13](#_Toc137028097)

[3.3.5 社交动态模块用例 14](#_Toc137028098)

[第4章 系统设计 15](#_Toc137028099)

[4.1 架构设计 15](#_Toc137028100)

[4.1.1 前台应用系统 15](#_Toc137028101)

[4.1.2 后台管理系统 15](#_Toc137028102)

[4.1.3 后端服务系统 15](#_Toc137028103)

[4.2 模块设计 16](#_Toc137028104)

[4.2.1 前台应用系统 16](#_Toc137028105)

[4.2.2 后台管理系统 18](#_Toc137028106)

[4.2.3 后端服务系统 20](#_Toc137028107)

[4.3 数据库设计 20](#_Toc137028108)

[4.3.1 概念结构设计 20](#_Toc137028109)

[4.3.2 逻辑结构设计 22](#_Toc137028110)

[4.4 安全性设计 34](#_Toc137028111)

[4.5 核心算法流程描述 36](#_Toc137028112)

[第5章 系统实现 41](#_Toc137028113)

[5.1 前台应用系统 41](#_Toc137028114)

[5.1.1 用户模块 41](#_Toc137028115)

[5.1.2 主页模块 44](#_Toc137028116)

[5.1.3 家族模块 44](#_Toc137028117)

[5.2 后台管理系统 46](#_Toc137028118)

[5.2.1 权限管理 46](#_Toc137028119)

[5.2.2 数据管理 49](#_Toc137028120)

[5.2.3 用户管理 50](#_Toc137028121)

[5.3 后端服务系统 52](#_Toc137028122)

[5.3.1 系统文档 52](#_Toc137028123)

[5.3.2 工程结构 53](#_Toc137028124)

[5.4 系统测试 53](#_Toc137028125)

[第6章 结束语 57](#_Toc137028126)

[6.1 总结 57](#_Toc137028127)

[6.2 展望 57](#_Toc137028128)

[参考文献 58](#_Toc137028129)

[致 谢 60](#_Toc137028130)

# 第1章 绪论

## 1.1 研究现状和意义

“家之有谱，犹国之有史也”[4]。我国的家谱档案资源是中华民族历史文化的重要组成部分，在完善现代社会档案资源体系、填补历史空白、重构社会记忆等方面起着非常重要的作用[5]。随着互联网技术的飞速发展，传统纸质家谱的诸多缺点一览无余，这为数字化家谱系统的发展带来了全新的机遇。

首先，纸质族谱存在保存难、翻阅复杂、传播性弱等问题[6]。通过将数字化后的家族信息存储在可移动磁盘和数据库等存储设备中，即使发生火灾、水患或地震等意外，家族信息也能得到有效的保护。同时，数字化的家谱信息可以进行不限量备份和长期保存，并且容量不受纸张大小和数量的限制。与传统纸张易于磨损、容易变质和容量有限的缺点相比，这进一步保障了家族信息的完整性和持久性。

其次，数字化家谱系统进一步方便了信息的查阅、更新和维护。家族信息通常会随着时间的推移而发生变化，在纸质家谱中更新和修正信息是一项繁琐的任务，需要手动进行修改或添加，容易出现错误或遗漏。而通过数字化家谱系统，家族成员可以随时随地通过互联网及时更新个人信息和家庭情况等。这样，家族成员间的信息共享和传递变得更加便捷和高效，传统纸质家谱的信息查阅和维护能力与数字化家谱系统相比显得捉襟见肘。人们通过利用数字信息技术对家谱文献进行数据采集、集成和挖掘，拓展了家谱文献资源再生性保护与利用的渠道[7]。

此外，家谱数字化为中华传统家谱资源的整理和整合提供了新途径[8]。通过对家族成员的信息进行分类，可以按照不同的标准或属性进行组织和归档，比如按照家族分支、世系关系或职业等进行分类，使得家谱系统层次更加清晰。这样，无论是进行家族历史研究，还是针对具体的人物进行分析，都能够迅速定位到相关信息，提高了学术和历史研究的效率和准确性。通过对家族数据进行整理和分类，数字化家谱系统不仅提供了更加清晰和有序的家谱信息，而且还使得研究人员能够更加高效地进行历史研究和分析。无论是追溯家族历史发展，还是探索特定人物的生平和影响，研究人员都能够迅速定位到相关的信息，减少繁琐的查找工作，这对于家族研究和文化传承具有重要的意义。

最后，数字化家谱系统能够突破地理限制，方便信息的共享和传播，为家族成员带来了更多便利。传统的纸质家谱在家族成员分散在不同地区或国家时，面临着信息传递的困难，加之复制纸质家谱本身也是一个巨大挑战。而互联网的普及使得以上问题迎刃而解，家族成员可以通过网络平台分享和交流家族信息，无论家族成员分布在世界的哪个角落，或是与远离故乡的亲人，他们都能够通过家谱系统了解家族的历史和发展。数字化家谱系统通过利用互联网技术突破了地域限制，为家族成员提供了便捷的信息共享和传播方式。它促进了家族成员之间的紧密联系，加深了血缘的联结，同时也为家族历史的记录和传承提供了更好的平台。这种跨时空的信息传递，为家族的持续发展和文化传承注入了新的活力。

综上，越来越多的人选择将家谱数字化，以克服传统纸质家谱的诸多问题。故而开发一款数字化家谱服务系统势在必行，利用互联网对家谱进行数字化处理，不仅能够有效的帮助用户修编家谱，还能对家谱内容进行及时的修订和补充，对于家谱的保护和开发利用都起到极为重要的作用[9]。家谱系统不仅可以解决传统纸质家谱所面临的问题，还能为家谱的修订、补充和开发利用提供便利和机会。通过利用互联网和先进的技术手段，数字化家谱服务系统将为家族历史的传承和保护做出重要贡献。

## 1.2 国内外现状分析

在中国传统文献典籍中，家谱或称族谱、宗谱、家乘，是以记载祖先、世系和宗族制度为主要内容，反映中国社会与传统文化中的祖先崇拜、血缘意识和社会组织结构的一种历史文献[10]。随着数字化技术的快速发展和广泛普及，国内外数字家谱系统也应运而生，研究与发展已经取得了不小的进展，但仍存在着功能不够完善、页面不够美观和交互逻辑较差等问题。

### 1.2.1 国内研究现状

我国的家谱研究在移动端呈现出令人瞩目的发展趋势。随着智能手机和移动互联网的广泛普及，越来越多的互联网公司开始致力于开发功能完整且易于使用的数字化家谱应用，以满足用户在移动设备上查看、编辑和分享家谱信息的需求。通过在华为手机应用市场进行关键词检索，可以发现云族谱、族谱、时光家谱、百家有谱和藤云家谱等几个代表性的家谱APP。这些应用注重用户界面和交互逻辑的设计，使用户能够轻松地浏览和管理家族信息、建立完整的家族血统图，并实现对特定家族成员和人物关系的检索和定位等功能。

此外，国内的家谱研究也在电脑端网站方面取得了显著的发展。目前一些文化企业、资料收集机构、文化科研单位和社会组织纷纷建立家谱文化网站，弘扬家谱文化、开展家谱文化研究、存史集新、交流家谱资料和信息、寻根问祖。相关专门性的综合性的网站有“中国家谱网”、“家谱网”、“中国新家谱网”、“百姓家谱”、“家族族谱网”、“家谱寻根网”、“家谱信息”等[11]。家谱网、中国家谱网和中华寻根网等典型代表性网站为家谱研究领域提供了丰富的资源，在家族历史的展示和管理方面发挥着重要的作用。相较于移动设备有限的屏幕操作空间，这些家谱网站能够更充分地展示和管理家族的复杂关系和详细信息，为用户提供更为全面和便捷的研究平台。通过这些网站，用户可以更加方便地管理和展示家族的复杂关系和详细信息，同时也可以利用各种功能进行家族研究和寻根问祖。这些网站的发展和应用促进了家谱研究的进步，为广大用户提供了宝贵的家族历史资源。

### 1.2.2 国外研究现状

相较于国内，国外对于家谱档案利用十分看重，主要表现在相关机构对于家谱网站开发方面。国外档案馆的网站中大多设有“家族树”、“家族史”等栏目，其目的是为了更好的为公众提供服务，促进家族档案资源共享[12]。国外的数字化家谱系统研究起步更早，并更加注重技术的创新。一些科技公司利用图像处理技术修复传统家谱中已损坏的信息，让家谱更加完整。还有学者运用人工智能技术对家族中已故人员进行虚拟化处理，为用户提供更加丰富的家族文化体验。国外的代表性家谱网站包括Ancestry、MyHeritage和FamilySearch等。

国外的家谱网站更加注重用户隐私保护。它们采用严格的数据传输加密算法和访问权限控制，以确保家族信息的安全性和保密性。通过对用户分配不同的系统角色，并为每个角色分配一系列权限，实现了精细的用户访问权限控制，严格控制家族信息的可见范围。

同时，国外许多家谱网站非常重视多语言和跨文化支持。网站通常支持切换地区和语言，为用户提供定制化的服务，使他们可以根据自己熟悉的语言和操作习惯浏览和管理家族信息。有些网站还支持多种文化传承的家谱数据互联研究，为全球范围内的家族研究者提供了交流和合作的平台。

除此以外，国外家谱开发利用中还有“家族树（family tree）”的建立，家族树是指电子家谱的管理电子家谱的建立与管理。如Ancestry.com网站的注册用户可以“家族树”创建板块创建属于自己家族的电子家谱，并可以选择在家族成员之间或之外进行共享，上传并记录家族照片、影像资料，形成一个较大的家谱资源交流、共享社区，实现共同创建与维护[13]。

总体而言，数字化家谱服务系统在国内外都得到了广泛的关注和应用。国内主要集中在技术开发和数据资源建设，而国外则更加注重数据互联和跨学科合作。随着技术的进步和社会需求的增加，数字化家谱服务系统将继续发展，并在家族研究、遗传学和文化遗产保护等领域发挥更重要的作用。

## 1.3 研究内容和目标

本文基于以上分析的传统纸质家谱存在的诸多问题与不足，力求开发一个现代化、多功能的数字化家谱服务系统，分为移动端前台用户应用系统和桌面端后台管理员管理系统，以满足当下家族对家谱系统的多样化需求。主要包含的功能和特点有：

1. 全面的家族信息管理：系统将提供如家族成员树谱、家族相册、家族大事记和家族人物事迹等家族必要资源数据的新增、删除、编辑和修改功能。
2. 多样的数据可视化：基于现代绘图引擎的发展，系统将提供多种呈现家族资料的方式。如家族成员世系图、人际关系图、家族大事时间轴和各种统计图等。
3. 成员搜索与定位功能：系统将整合强大的搜索引擎，使得家族用户可以迅速地查询与定位到特定的家族成员，并且支持条件过滤和模糊查询等高级检索功能。
4. 多平台和跨设备支持：系统支持通过各种操作系统平台和设备浏览器进行登录访问，如用户可以使用手机、平板或电脑在各种Windows、IOS或Linux平台下访问系统。
5. 数据云端存储和备份：通过整合第三方云存储平台如七牛云和腾讯云等，为家族数据的安全性提供保障。系统还支持数据定时备份，防止资料的意外丢失和损坏。

通过开发这样一个全面且易用的数字家谱服务系统，将大力推动传统家谱的现代化转型，满足当下家族对家谱管理和文化传承的急切需求。

## 1.4 本文的组织结构

第1章是绪论，探讨了本研究的背景和意义，明确了研究的目标和内容，并分析总结了当前研究的国内外现状，为后续内容的展开做好铺垫。

第2章是相关技术介绍，主要对本研究设计到的相关技术背景和基础知识进行介绍，为系统需求分析和设计提供理论基础和技术支持。

第3章是系统需求分析，对系统的功能性和非功能性需求进行详细分析和规划，为系统的功能和性能要求提供明确的指导。

第4章是系统详细设计，先对系统的整体架构和模块进行概要设计，进一步对各个模块的数据结构、接口、数据库等进行详细设计，确保系统的可运行性。

第5章是系统功能实现，通过具体的编码和开发工作，实现系统各个模块的具体功能，保证系统的功能符合预期要求。

第6章是总结与展望，对整个研究工作进行总结与评价，指出系统开发中的创新点和存在的不足，为未来进一步开发提供方向。

# 第2章 相关技术

## 2.1 系统技术栈

如图2.1所示，该图清晰明了地展示了本研究系统开发过程中各个层次模块使用到的技术栈、开发工具和运行环境，以及各项技术和工具之间的关联与依赖关系，使得整个系统的技术层次一目了然，为系统开发提供参考和指导。这些技术和工具在构建现代化、安全和可靠的Web应用程序方面发挥了重要作用，下面将对其中极为典型的开发技术进行详细介绍。

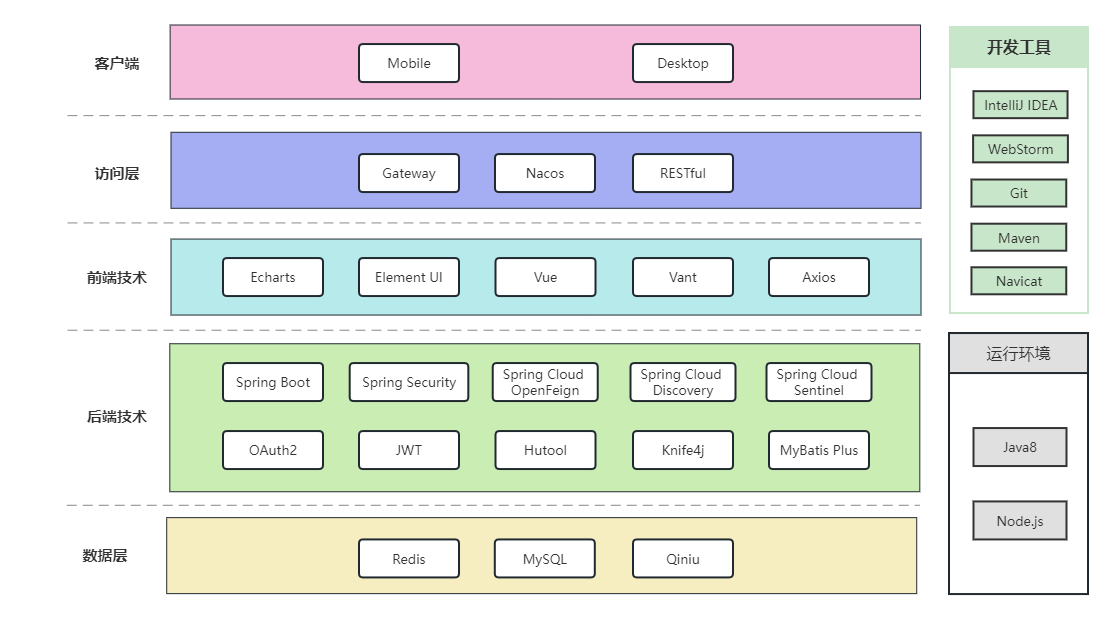


图2. 系统技术栈示意图

## 2.2 Vue

Vue.js是一款现代化、轻量级、用于快速构建用户页面的JavaScript框架，旨在简化Web应用的开发，常用于开发现阶段流行的单页应用（Simple Page Application）。具体来说，Vue.js 是一款依赖于模型-视图-视图模型（MVVM）的渐进式框架，通过其提供的一系列简单易上手的内置API，为开发者打造了优雅流畅的前端体验[14]。

Vue可以通过简单的API实现响应式双向数据绑定，快速构建用户界面。开发者只需要在Vue的实例中声明定制化的数据，在DOM元素上使用插值语法绑定数据，Vue便会自动追踪数据的变化，从而实时更新视图[15]。当然，页面数据的变化也将映射到定义的数据。

Vue组件化的开发思想极大程度上提高了代码的复用性和可维护性，同时也使得整个系统更加模块化和可扩展。组件化的开发方式使得组件的前端开发语言HTML、CSS、JS写在同一个文件里，同一个组件的资源就近维护，提升代码的可读性，使得程序结构更加清晰[16]，可以通过组合和嵌套的方式来构建复杂系统。

除此之外，Vue还拥有一个非常活跃的生态系统，提供了一系列非常实用的官方插件与工具。如Vue Router允许自定义路由规则用于构建可导航的SPA应用，Vuex状态管理库用于集中管理应用程序的状态。还有一些非常出色的开发工具如Vue CLI命令行工具、Vue Devtools 浏览器插件等，为开发者提供了易用、灵活和高效的开发体验。

## 2.3 MySQL

MySQL是一款可在多种操作系统平台上运行的开源RDBMS（Relation Database Management System）[17]。它以出色的性能、可靠性和灵活性备受企业和个人开发者喜爱，得到了开源社区的广泛支持。

MySQL较为全面地实现了结构化查询语言SQL定义的功能，使得开发者可以使用丰富的语法和操作符对数据进行高效的CRUD操作。当然，MySQL在支持SQL的同时也进一步扩展了SQL的功能，引入了诸如存储过程、触发器和视图等高级特性，使得开发者能够以更加灵活的方式处理数据。

MySQL具备良好的可扩展性，支持分布式部署。可以通过主从复制、分区等技术实现数据的水平和垂直扩展，从而保证了多个服务器之间数据的同步性和高可用。这些特性使得MySQL支持高负载场景，从而得到广大中小型网站以及相应企业的信赖。

MySQL提供了一系列工具和技术来优化数据库性能。例如，可以通过配置缓存、调整服务器参数和优化查询来提高性能。MySQL还提供了监控工具和性能分析器，用于识别性能瓶颈和优化数据库配置。

## 2.4 Spring Cloud

Spring Cloud技术框架并非是单一技术组成的框架，而是由其他一系列框架所组成的有序框架集合[18]。Spring Cloud中提供了一系列实用组件等，帮助开发人员更轻松地构建和管理分布式应用。其中Gateway、Discovery和OpenFeign是Spring Cloud最为核心的三个组件，在微服务架构中发挥着举足轻重的作用，也是本研究中用到的Cloud组件。

Spring Cloud Gateway是一个轻量级网关，用于构建和管控分布式系统中的请求路由和统一鉴权等[19]。通过配置具体的路由规则将客户端的请求映射到对应的微服务实例，同时支持负载均衡、流量控制和自定义过滤器等，为开发人员提供一个强有力的微服务开发工具。

Spring Cloud Discovery服务发现是构建分布式系统的关键组件，它使得各个微服务能过够自动注册服务并发现彼此，简化了微服务之间的通信和协调。Discovery支持多种服务注册与发现的实现，其中最常用的是Eureka、Zookeeper和Nacos。

Spring Cloud OpenFeign是一种基于声明式的REST客户端，旨在简化微服务之间的服务调用过程[20]，基于注解和接口定义的使用方式使得服务调用极为优雅。通过与Discovery集成，能够自动解析服务实例的地址和负载均衡策略，使得微服务间的通信变得更加简单和可靠。

## 2.5 Spring Security

Spring Security是一个开源的Java框架，提供高度灵活的身份验证、授权和访问控制解决方案。其通过一系列过滤器组成的过滤器链，为系统提供身份认证（Authentication）和授权（Authorization）功能，从而帮助开发者构建安全可靠的应用程序[21]。Spring Security 提供了一套简单且易于使用的API，用于管理用户身份验证、授权和其他安全性功能。它的主要目标是保护应用程序免受各种安全威胁，例如身份伪造、会话固定、跨站点请求伪造（CSRF）和跨站点脚本（XSS）攻击。

## 2.6 OAuth2

OAuth2是一种用于授权第三方应用程序访问用户资源的安全协议[22]。它允许用户让第三方应用访问用户在某个网站上存储的私密资源(例如照片、联系人通信录、视频等)，而不需要将用户名和密码提供给第三方应用[23]。许多大型互联网平台和服务商都采用OAuth2作为其授权框架，以确保用户数据的安全。开发者可以使用现有的OAuth2框架来简化实现，如Spring Security OAuth、Okta、Auth0等。

## 2.7 本章小结

通过前文系统技术栈架构设计和详细技术介绍，本研究拟采用Vue作为前台用户界面开发框架，选择MySQL作为系统的业务存储数据库，基于Spring Cloud Gateway、Spring Security以及OAuth2搭建分布式系统统一的认证和鉴权解决方案，实现认证服务只负责认证，网关负责校验认证和鉴权，其余微服务模块仅处理自己的业务，从而为用户提供良好的用户体验。

# 第3章 系统分析

## 3.1 功能性需求分析

一个现代化、功能全面且简单易用的数字化家谱服务系统应当包括但不限于以下功能：

1. 用户信息管理：系统应支持用户通过手机号、邮箱地址等进行账号注册与登录，并赋予用户修改个人信息和隐私信息的权限。同时，用户也可以进行找回密码、修改密码、注销账号以及查看账号登录记录等操作。
2. 家族信息管理：赋予家族创建者和修委会成员编辑家族基本信息，如家族名称、家族姓氏、家族地址及家族祖籍地址等，普通家族成员可以查阅家族信息。
3. 家族成员管理：家族修委会成员可以基于已经存在的家族成员进行添加亲人、移除成员、编辑成员信息以及查看家族成员树谱、家族世代成员列表等操作。而普通家族成员只能进行查阅操作，无权修改除亲人列表外的家族成员信息。
4. 家族公告管理：家族修委会成员可在后台管理系统中发布、编辑以及删除家族公告，家族成员可以在前台用户系统中浏览最新和历史家族公告。系统还应支持家族公告内容推送功能，即家族管理员发布家族公告后，每个家族成员用户应当能收到包含家族公告内容的系统消息私信。
5. 家族相册管理：每个家族成员可以新建家族相册，并上传图片至家族相册中，丰富家族文化内容。家族修委会成员可对家族相册进行审核、编辑等操作。
6. 家族大事管理：家族修委会可在后台管理系统中新增家族大事，如家族成员的婚丧嫁娶、家族聚会、庆祝活动、家族会议及决策等，记录家族多样化的发展历史。
7. 人物事迹管理：对家族发展做出极大贡献的家族成员，家族修委会可以发布人物事迹以纪念其对家族发展所做出的贡献，作为激励与鼓舞后代的范例。
8. 祖坟祠堂管理：家族修委会可通过实地采集收录，记录家族祖坟埋葬人员详细信息、准确地理位置以及坟墓图片等信息，后人通过系统提供的地图导航技术前往祖坟位置吊唁缅怀祖先。
9. 修谱记录管理：系统应记录家族成员变化情况以及家族访问记录。每位家族成员都能查看家族修谱日志，即何人在何时对家族成员信息做出了何种操作，如编辑、删除、新增家族成员信息等，以此保证家族信息的准确性。
10. 家族社交动态：在前台应用系统首页，可以查看当前家族所有成员发布的社交动态信息。这些动态信息内容丰富多样，具有很强的交互性，可以对其他成员发布的动态进行点赞、评论等互动操作，这进一步增加了家族成员之间的联系和纽带性。

## 3.2 非功能性需求分析

1. 性能需求：考虑到家族信息的复杂性与家族成员规模的庞大性，系统后台服务采用分布式架构，可以极为方便地进行水平扩展，具备较好的高并发和高负载能力，以满足用户对系统快速响应的速度需求。系统要求整体响应时间不大于5s，支持的并发用户数不低于500，CPU占用率不超过80%。对于某些性能达到瓶颈的功能，从用户体验上进行优化。如家族树谱页面数据量大，需要递归查询和展示成员数据时，系统弹出动图转移用户注意力，缓解用户焦虑。
2. 安全性需求：系统提供严格的访问权限控制和数据加密保护功能，并且记录用户的操作日志，确保用户数据的保密性和完整性。系统认证服务提供严格的身份验证和授权机制，网关服务负责校验认证和授权，确保了只有经过授权的用户才能访问和修改家族数据。同时用户敏感数据选用非对称加密算法进行加密，防止未经授权的访问和篡改行为，进一步保证用户隐私安全。
3. 可靠性需求：系统在实现可靠性方面采取了多种策略和技术，以确保系统的容错性和可靠性。首先，系统利用Spring Cloud负载均衡和熔断器等策略来实现请求的分发和高可用性，从而减轻单点故障的影响，并提供了容错机制，确保系统能够正常运行并处理请求。并且系统采用了可靠的第三方数据存储方案以及数据备份技术，通过选择可靠的数据存储方案，系统能够将数据存储在可靠的持久化介质上，并提供数据的高可用性和可靠性，从而防止数据丢失或损坏。
4. 易用性需求：系统应该操作简单，能够让用户快速上手[24]，快速理解系统的操作方式和功能。其次，系统应该保持一致的设计风格和操作逻辑，提供可预测的用户体验，使用户能够轻松地进行导航和操作。系统还应该具备良好的易学性和易上手性，提供明确的操作指导和帮助文档，使用户能够迅速学习和掌握系统的使用方法。此外，系统应该具备良好的错误处理和反馈机制，能够及时捕捉用户的错误输入并给出明确的错误提示和建议。
5. 可维护性与可扩展性需求：随着业务不断扩展的要求，保证系统具有较强的易维护性和可扩展性，能方便地进行功能调整以适应系统需求变化[25]，以方便系统管理员进行必要的技术升级和业务扩展。系统前台基于Vue组件化开发、后台基于Spring Cloud模块化开发的系统架构设计本身便提供了极好的可维护性和可扩展性，加之系统制定了统一的开发规范、进行了必要的单元和系统测试，使得系统能够很好地适应未来的需求变化和业务扩展，以及进行必要的维护和升级。

## 3.3 系统用例分析

### 3.3.1 系统总用例

如图3.1所示，其展示了本系统的各个模块主要角色涉及到的用例和用例关系。

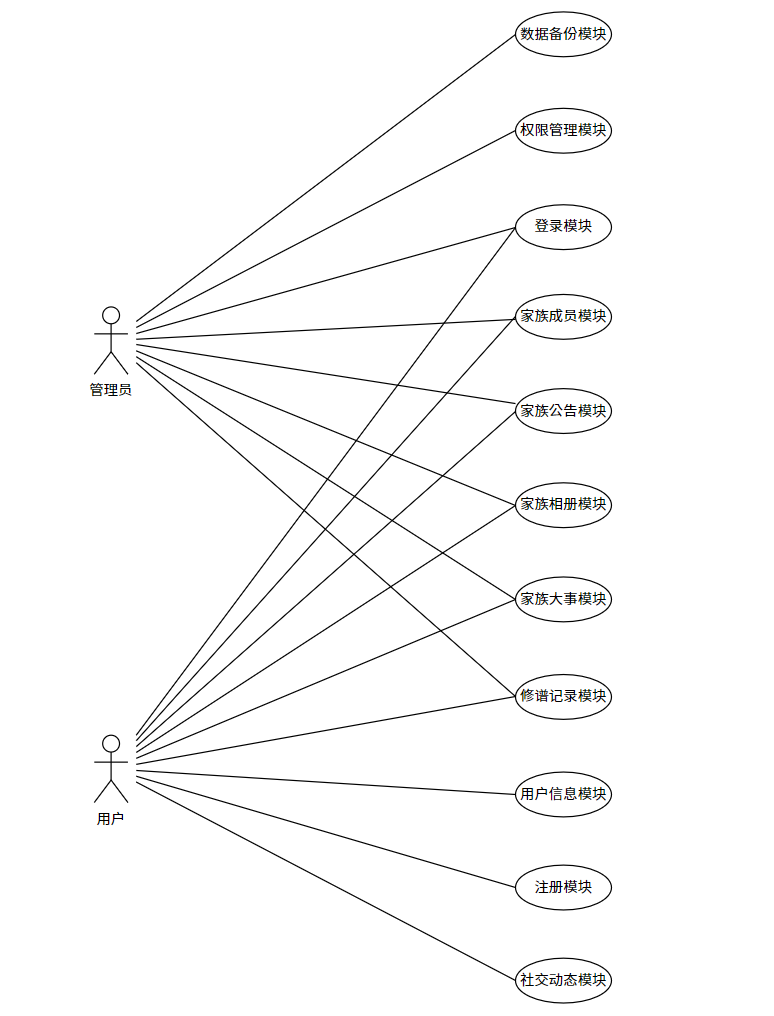


图3.1 系统总用例分析图

### 3.3.2 用户模块用例

如图3.2所示，该用例图描述了用户子系统与参与者之间的交互，系统的主要参与者是用户和管理员，每个参与者可以执行的操作如下图所示：

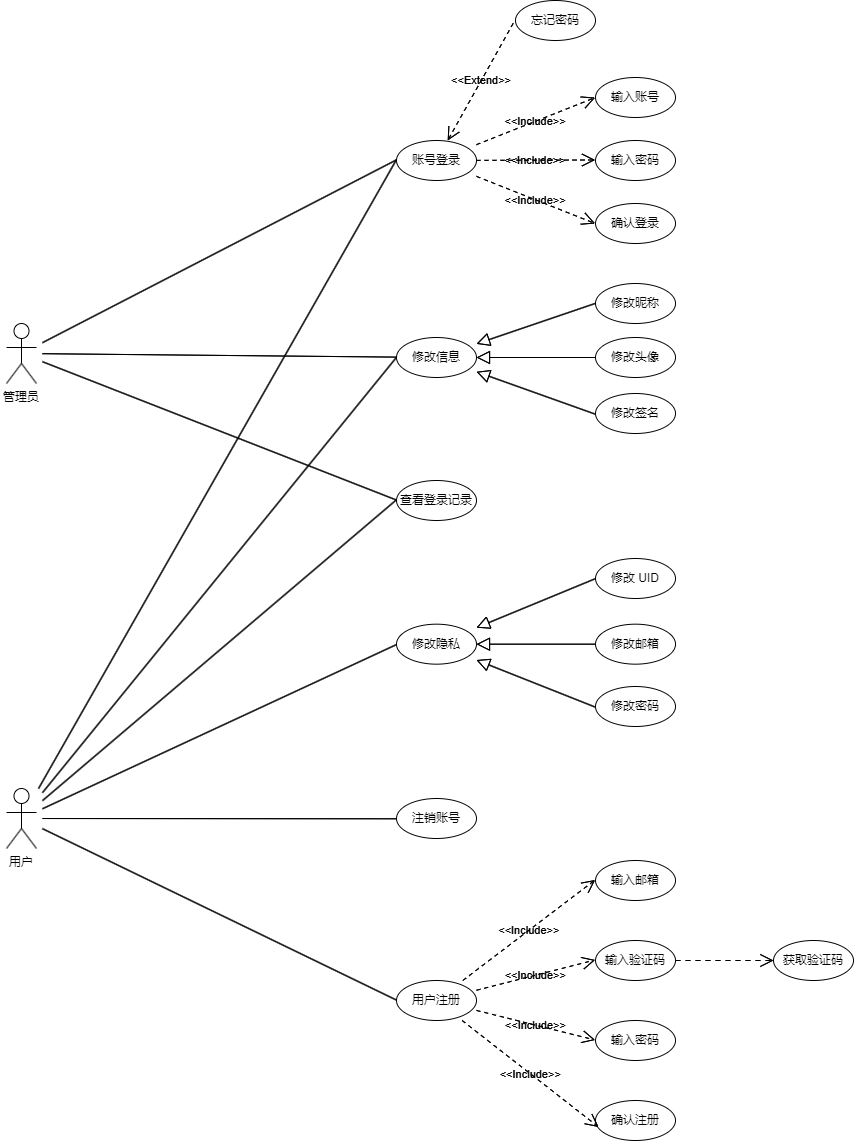


图3.2 用户模块用例分析图

### 3.3.3 家族人员模块用例

家族成员模块中包含的主要功能是对家族成员进行新增、移除、编辑和查看等操作。如图3.3展示了家族成员用例，图3.4展示了家族管理员用例，各参与者可以执行的具体操作如下：

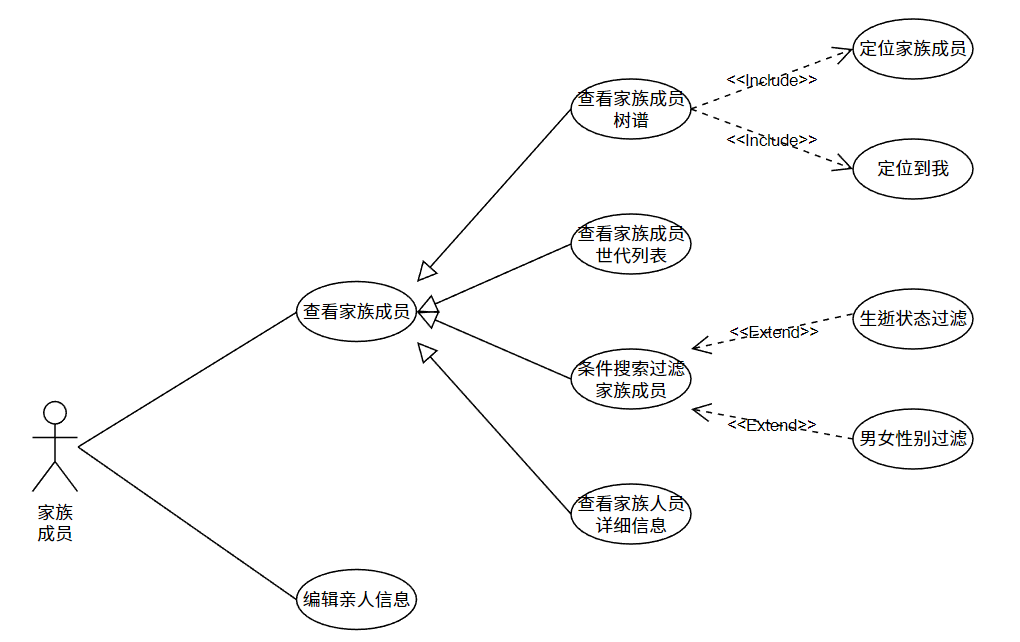


图3.3 家族人员模块家族成员用例分析图

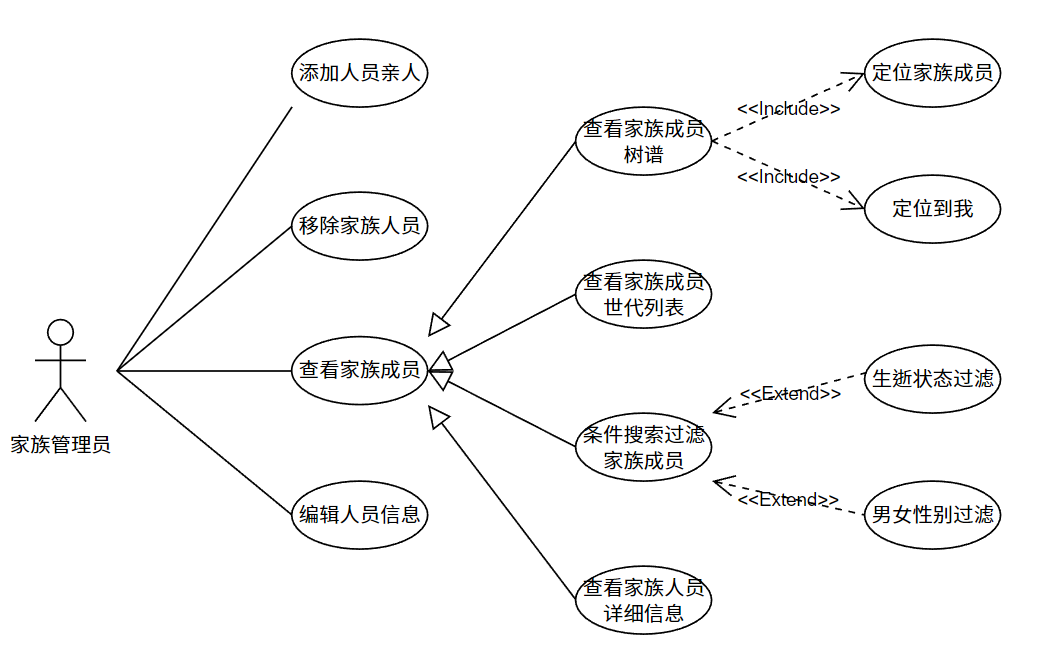


图3.4 家族人员模块家族管理员用例分析图

### 3.3.4 家族相册模块用例

1. 家族相册模块包含的主要功能是家族成员上传家族相关图片记录家族发展变化，如图3.5所示。家族管理员辅助管理家族相册内容，如管理图片等，如图3.6所示：

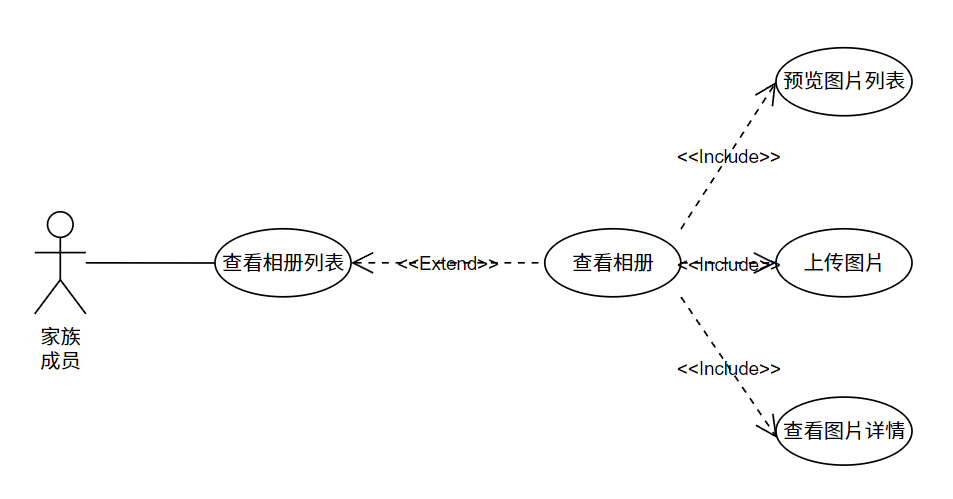


图3.5 家族相册模块家族成员用例分析图

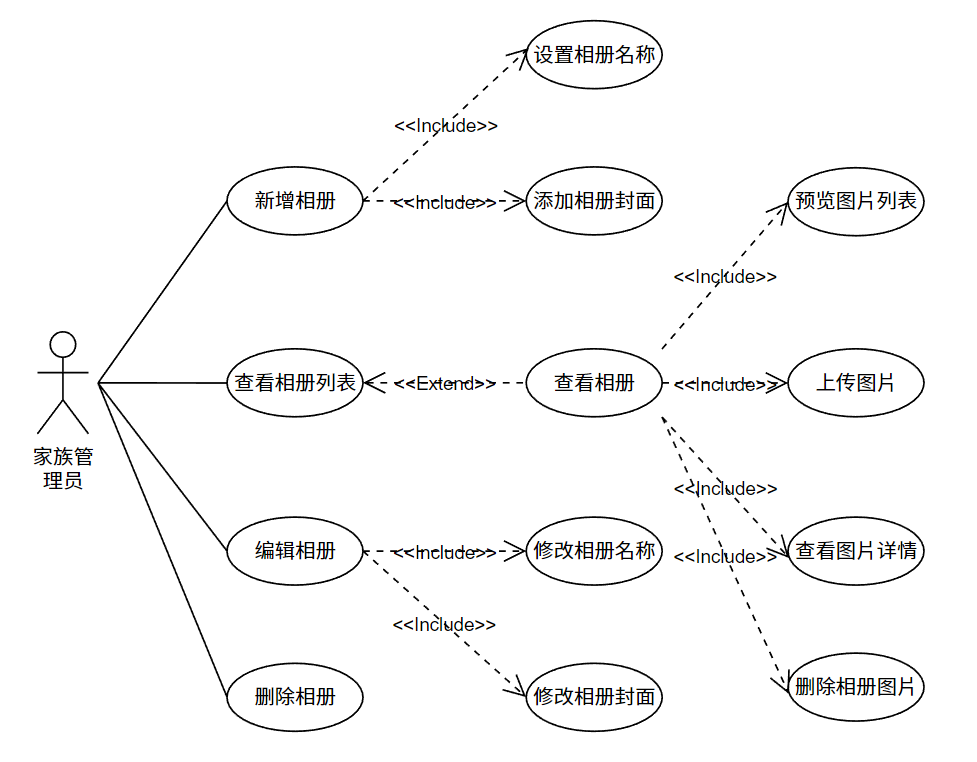


图3.6 家族相册模块家族管理员用例分析图

### 3.3.5 社交动态模块用例

如图3.7，家族社交动态模块的主要功能是家族成员发布包含图文消息的动态消息，其它家族成员可以进行点赞、评论、分享等操作。包含的参与者主要是家族成员，可以进行发布动态、编辑动态、删除动态、查看家族动态列表、查看个人动态列表等操作。

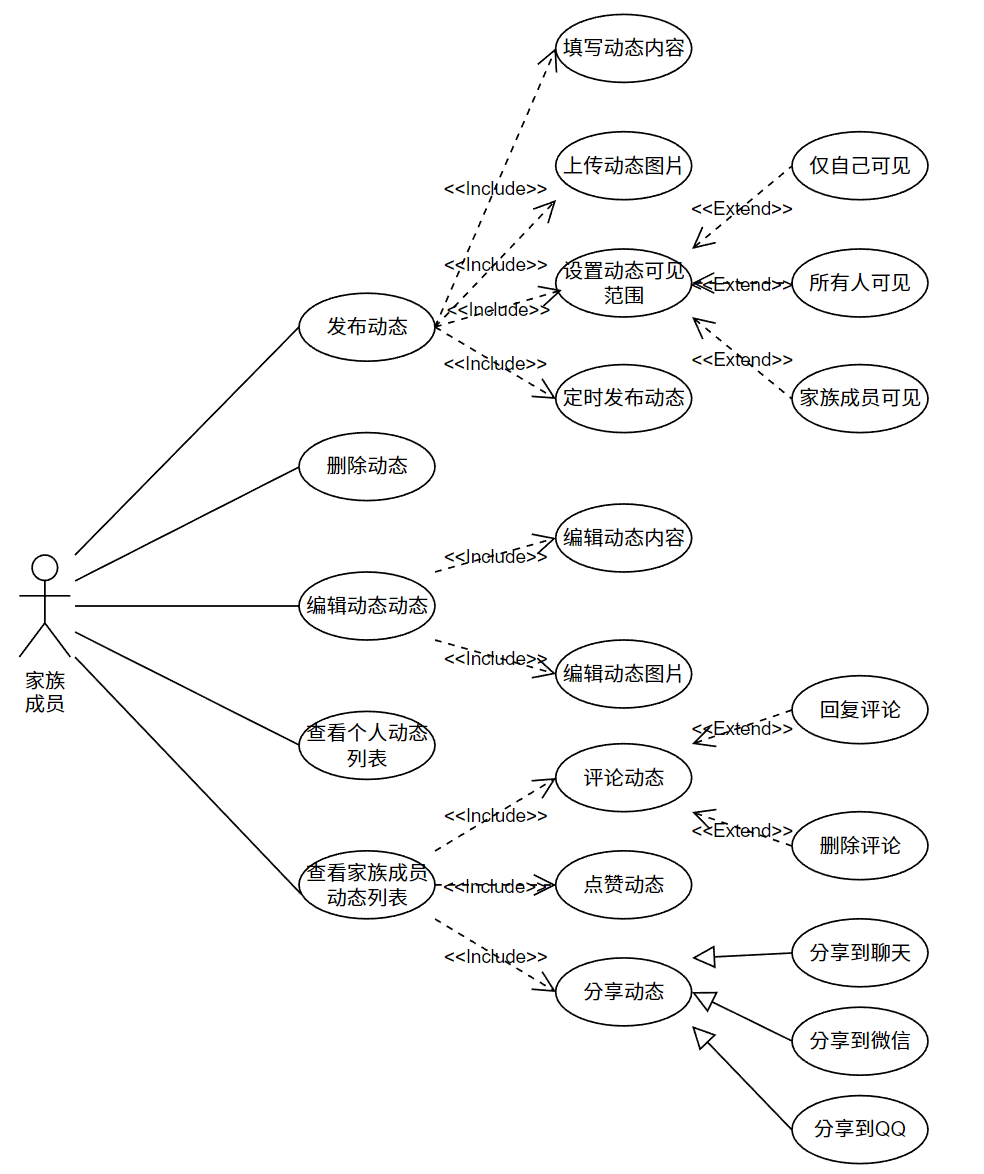


图3.7 家族社交动态模块家族成员用例分析图

# 第4章 系统设计

本系统（以下简称百家谱）是一个提供前台应用系统和后台管理系统的现代化、多功能和易上手的家谱服务系统。旨在帮助用户轻松地创建和管理自己的家族数据，提供一个便捷的平台来保存、共享和传承家族的历史，以满足当下家族对家谱管理和文化传承的迫切需求。系统具备良好的用户体验、灵活的界面定制能力和高可靠性的服务端架构。

## 4.1 架构设计

### 4.1.1 前台应用系统

百家谱移动端前台应用系统采用了Vue框架和Vant组件库，这使得系统能够以组件化的方式构建前端页面，提高了代码的可维护性和可复用性。通过使用移动端UI组件库Vant所提供的丰富组件和交互功能，百家谱移动应用实现了一致的页面风格，为用户提供了友好的界面和出色的交互体验。这大大提升了系统的开发效率，并使得用户能够更加轻松地使用和浏览应用。

### 4.1.2 后台管理系统

百家谱桌面端后台管理系统基于vue-element-admin模板库。vue-element-admin是一款基于Vue和Element UI的强大后台管理系统模板，它为开发者提供了众多的实用组件和灵活的布局选项，能够快速搭建功能完善和页面精美的现代化后台管理系统。通过对vue-element-admin路由和权限管理功能的定制与扩展，从而实现粒度极细的访问权限控制，最大程度上保证了系统的安全性和数据的保密性。

### 4.1.3 后端服务系统

百家谱后端服务系统基于Spring Cloud分布式微服务架构。这种架构将原有的大体量的单个应用拆分为多个独立的微服务，使得每个服务更专注于自己的业务[26]，能够很方便地进行水平扩展，从而使得整个系统进一步解耦。极大地提升了系统在高负载、高可用和高并发方面的能力，这样的架构设计为百家谱平台的稳定运行和未来发展奠定了坚实的基础。

通过Spring Cloud提供的Discovery，OpenFeign等实用组件实现微服务的动态注册、发现与调用，以及模块间的依赖解耦。此外，Gateway网关组件能够实现对外暴露统一的接口，完成统一的认证和鉴权，进一步提升系统的安全性和可靠性。

整个百家谱系统前端基于HTTP协议使用RESTful API与后端服务进行通行，完成数据的传输和交互。后端微服务之间通过Spring Cloud提供的服务注册与发现机制进行通信。

如图4.1所示，其清晰明了地展示了百家谱应用服务系统的系统架构设计。

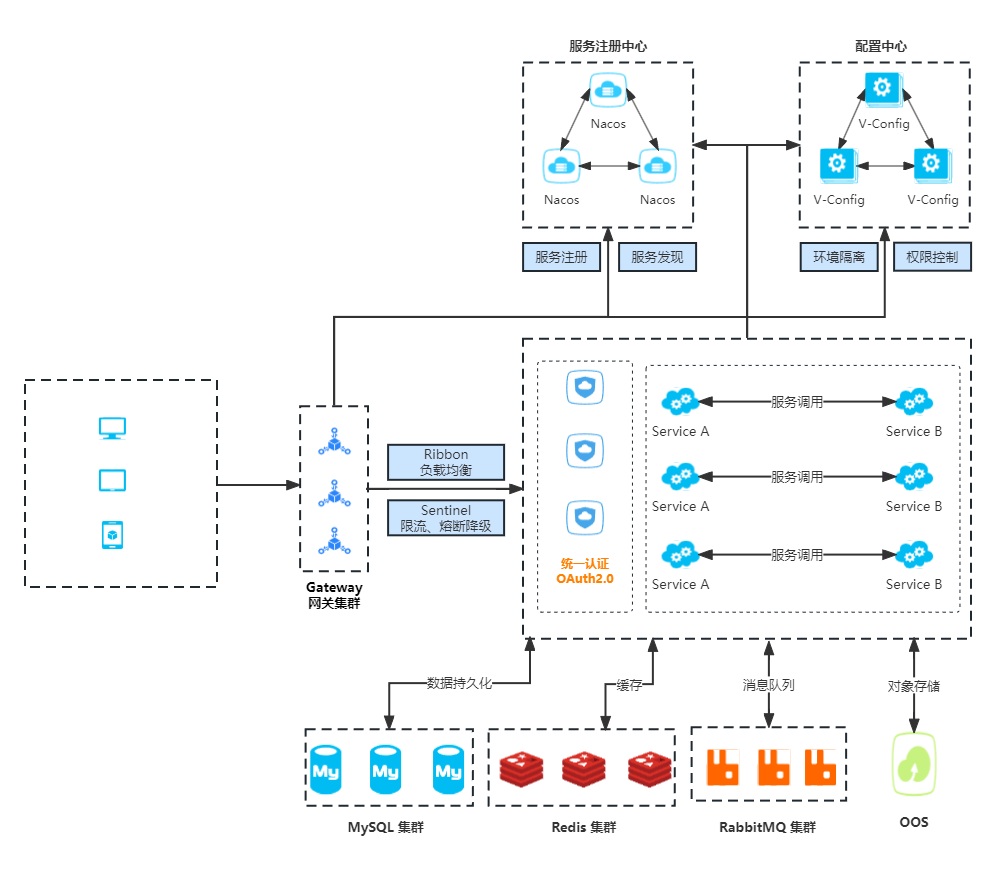


图4.1 系统架构设计图

## 4.2 模块设计

### 4.2.1 前台应用系统

百家谱移动端前台应用系统主要包含主页、家族和用户三大模块，各个模块包含的主要功能如图4.2所示。下面将从每个模块中各选取一个典型功能进行详细设计说明：

1. 新用户注册：游客用户可以通过正确的手机号或邮箱地址申请注册为百家谱用户，注册过程中需要获取并填写验证码、设置账号登录密码，注册成功后将自动跳转到系统首页，同时接收并保存后台服务器返回的用户令牌、刷新令牌、令牌前缀以及令牌有效期等信息，可用于后续用户免密的登录系统，接着用户便可享受百家谱前台系统提供的一系列家族谱系服务。
2. 家族成员管理：家族创建者或修委会成员（创建者设置的家族管理员）可对家族成员进行新增、移除、编辑和查看操作。其中进行新增操作时系统设置了规则限制以保证

家族信息的完整性，新增家族成员操作必须是基于已经存在的家族人员进行添加亲人操作，可以添加该人员的生父、生母、配偶、子女和同胞，具体添加限制规则如下：

1. 添加生父：只有该人员的生父信息不存在时方可添加生父信息，并且只有该人员不存在丈夫时才能添加生父信息。
2. 添加生母：只有该人员存在生父、不存在生母信息，且不存在丈夫时方可添加生母信息。
3. 添加配偶：不存在丈夫时方可添加配偶信息，实际上等价于只能给家族男性成员添加妻子信息。
4. 添加子女：只有存在配偶信息方可添加子女信息。当家族男性成员的妻子数量大于1时系统提示必须通过配偶添加子女，从而明确子女的生母关系。
5. 添加同胞：当且仅当不存在丈夫、存在生父和生母信息时方可添加同胞信息。

通过以上新增操作限制，进一步保证了家族血统的纯正性和家族成员信息的完整性。当然，移除家族成员时需要移除该成员的配偶和子女信息，因移除家族成员对家族成员层次结构影响较大，所以当前的百家谱系统实现为逻辑删除家族成员信息，家族修委会成员可通过后台管理系统查看已经逻辑删除的家族成员列表和彻底删除该家族成员后对家族造成的影响，从而决定是否彻底删除该家族人员。

1. 成员社交动态：家族成员用户可在用户模块中发布个人社交动态，包含填写动态内容、上传动态图片、设置动态可见范围、定时发布时间等操作。只有当家族成员用户将动态的可见范围设置为家族成员或所有人可见时，其他的家族成员在主页模块中才能查看到对应的动态图文消息，可以进行点赞、评论、回复评论等互动操作。

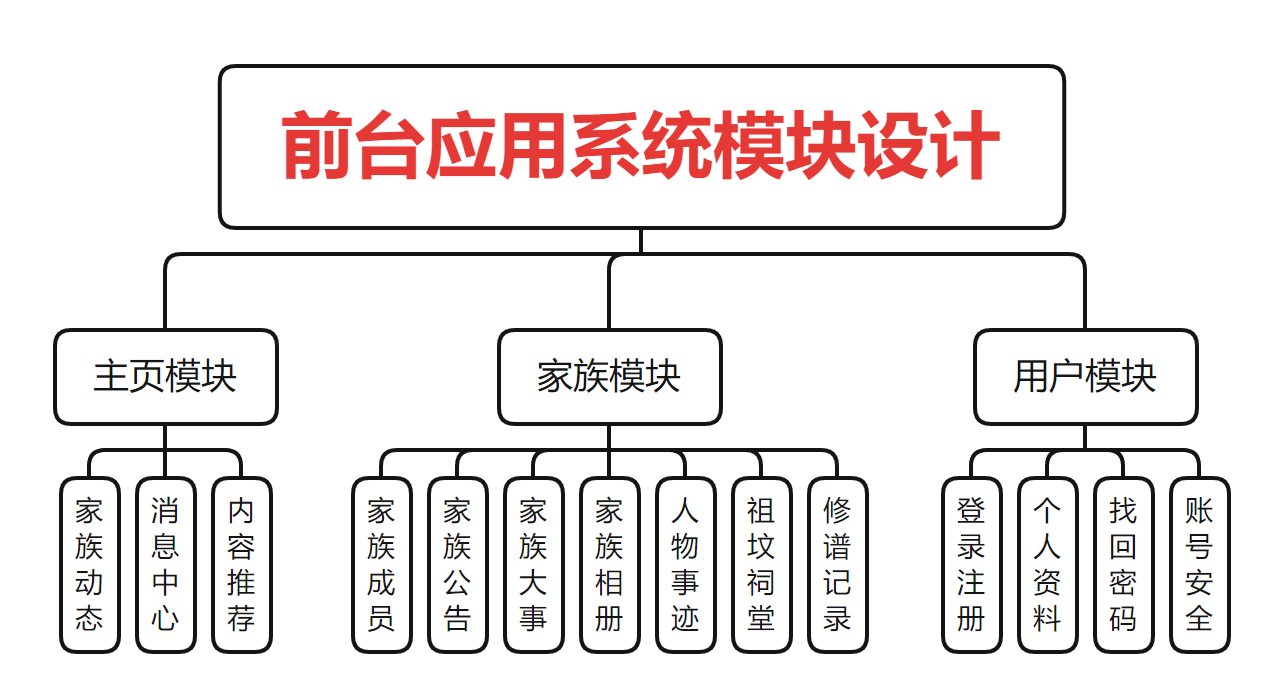


图4.2 前台应用系统模块设计图

### 4.2.2 后台管理系统

百家谱后台管理系统主要包含了家族、用户、数据、日志和配置五大模块。各个模块包含的具体功能如图4.3所示。下面将选取几个典型功能进行详细设计说明：

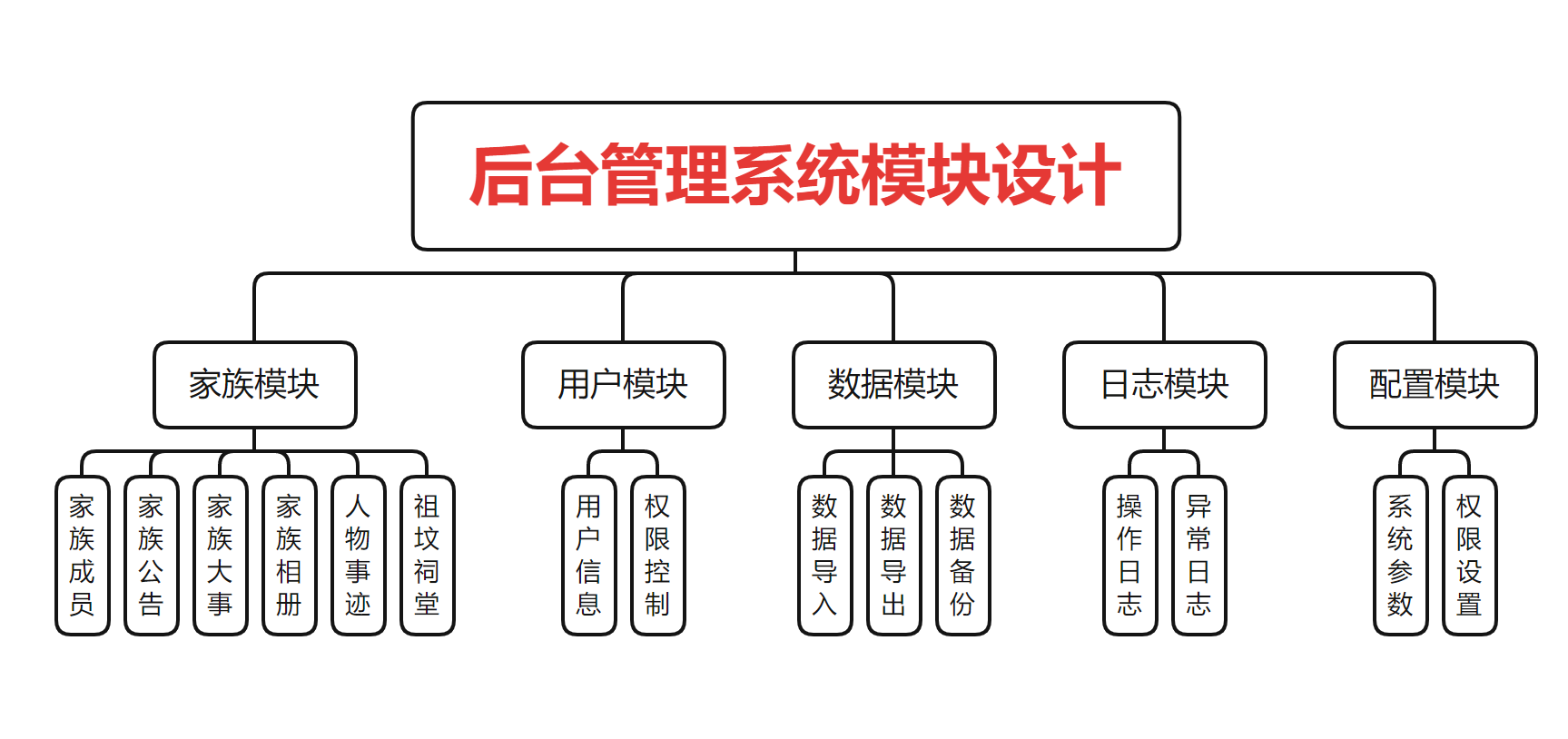


图4.3 后台管理系统模块设计图

1. 管理员登录：百家谱应用系统登录认证采用了现阶段最流行的OAuth2认证技术，摒弃了原有的Session认证方式，更好地满足了分布式应用系统的认证授权需求。登录请求认证流程如下所述：
2. 首先客户端调用登录接口传入用户名和密码，之后服务端请求身份认证中心，若验证用户名和密码正确，服务端创建JWT，返回给客户端[27]，前台将服务器返回的令牌和刷新令牌信息保存都本地存储中。保存的令牌和刷新令牌存在有效期，过期后管理员需要重新输入用户名和密码登录系统，获取新的令牌信息。
3. 随后，前端每次请求都会在请求头中携带令牌信息。服务器通过解析HTTP请求头中的令牌信息来获取请求用户的详细信息，然后服务器根据这些信息来处理用户的相关请求。
4. 最后，使用OAuth2认证的方式可以实现用户免密登录。当管理员本次操作结束关闭页面再次访问百家谱系统时，系统将从本地存储中取出令牌信息携带在请求头中，服务器通过解析请求头中的令牌即可完成用户身份认证识别，无需管理员输入用户名、密码再次登录。而当用户的令牌过期时，系统将从本地存储中取出刷新令牌请求服务器生成新令牌，进一步完善了用户免密登录逻辑，优化用户体验的同时进一步保障了用户敏感数据的安全。

相关的登录请求认证流程如图4.4所示：

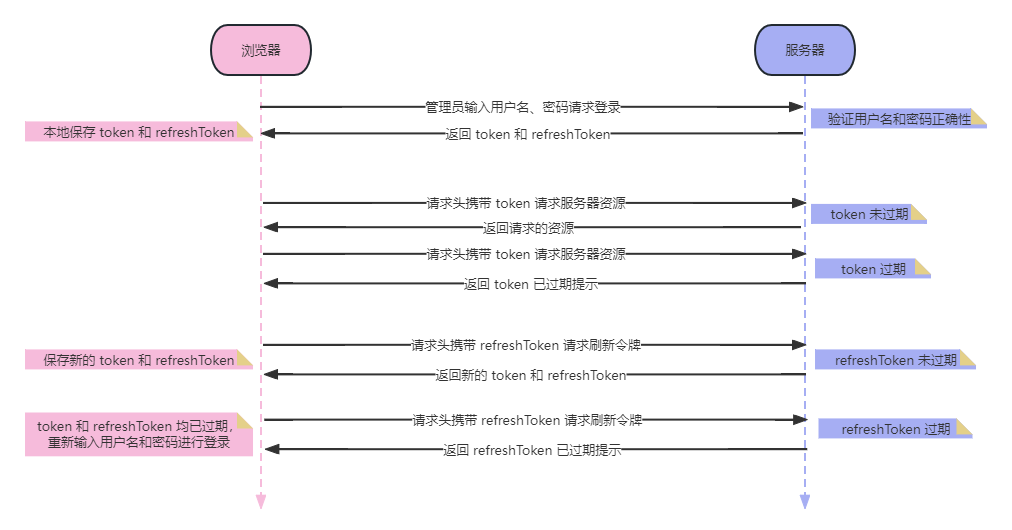


图4.4 登录请求认证流程示意图

1. 系统权限设置：百家谱系统权限包括前台路由菜单资源权限（Menu）和后端服务资源权限（Permission）。用户权限管理的核心是基于RBAC（Role-Based Access Control）模型，即每个用户对应一个或多个系统角色（Role），而每个Role对应一系列的Menu和Permission，从而做到粒度极细的系统访问权限控制，很好地保障了系统的资源安全。系统角色设计为家族成员、家族管理员、家族创建者、平台管理员、超级管理员五类角色，每种角色可以执行的操作如表4.1所示。

表4.1 系统角色与权限描述对应表

|  |  |
| --- | --- |
| 角色名称 | 权限描述 |
| 家族成员 | 仅能查看家族相关信息 |
| 家族管理员 | 在家族成员权限基础上，可新增、编辑、删除家族信息 |
| 家族创建者 | 在家族管理员权限基础上，可为家族修委会指定管理员 |
| 平台管理员 | 协助管理所有的家族信息，可操作所有家族资料 |
| 超级管理员 | 拥有系统所有权限 |

### 4.2.3 后端服务系统

百家谱后端服务系统主要包含了通用、认证、网关和业务四大模块。模块功能不一却又相互依赖、相辅相成，共同协作为前台应用系统和后台管理系统提供业务数据支持。系统工程架构模块图如图4.5所示：

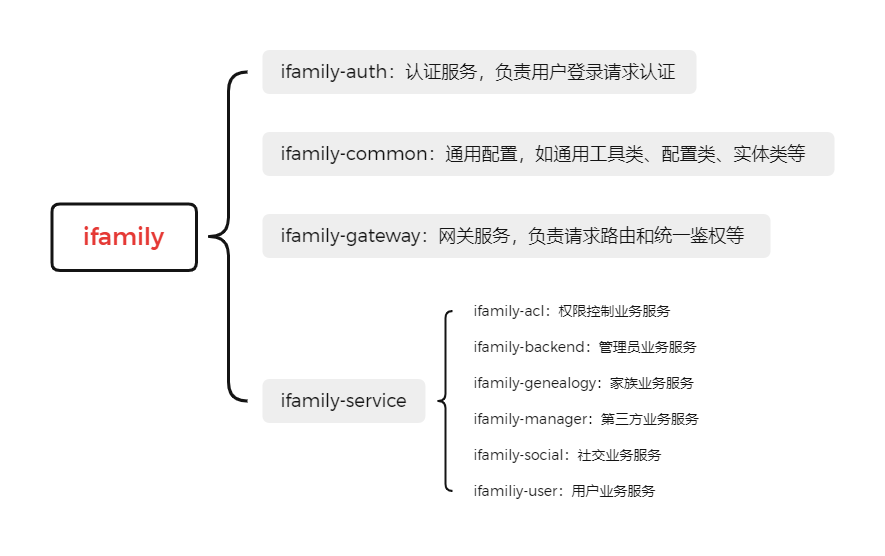


图4.5 后端服务系统工程模块设计图

## 4.3 数据库设计

百家谱是一个微服务架构的家谱服务系统，每个微服务模块ifamily-service仅仅关注处理与自己业务逻辑有关的业务，拥有自己独立的数据库。每个业务模块数据库表众多，以下数据库概念结构设计将抽取功能典型的数据库表进行设计说明。

### 4.3.1 概念结构设计

百家谱数据库系统包含的主要业务实体及数据项如下，各实体间关系如图4.6所示：

1. User（用户）：ID、用户名、密码、手机号、邮箱、昵称、头像图片地址、个性签名
2. UserLoginLog（用户登录记录）：ID、IP地址、IP归属地、登录设备、用户ID
3. Moment（用户动态）：ID、动态内容、动态可见范围、发布时间、用户ID、家族ID
4. MomentComment（动态评论）：ID、评论内容、动态ID、动态发布者用户ID、动态被回复者用户ID、父级评论ID
5. Genealogy（家族信息）：ID、家族封面图片地址、家族名称、家族姓氏、家族地址、祖籍地址、家族简介、家族字辈歌、创建者用户ID
6. GenealogyPeople（家族人员）：ID、人物肖像图片地址、姓氏、姓名、性别、世代、字辈、排行、手机、常住地、出生日期、出生地、逝世日期、埋葬地、家族特殊关系备注、生父ID、生母ID、丈夫ID、家族ID、用户ID
7. GenealogyRevisionLog（家族修谱日志）：ID、操作类型、操作日期、被操作人员姓名、操作者用户ID、被操作者用户ID、家族ID
8. Admin（管理员）：ID、用户名、密码、昵称、签名、头像图片地址、禁用状态
9. Menu（前台菜单）：ID、路由路径、菜单名称、菜单描述、禁用状态
10. Permission（后台权限）：ID、请求路径、权限名称、权限描述、禁用状态
11. Role（系统角色）：ID、角色名称、角色描述、禁用状态

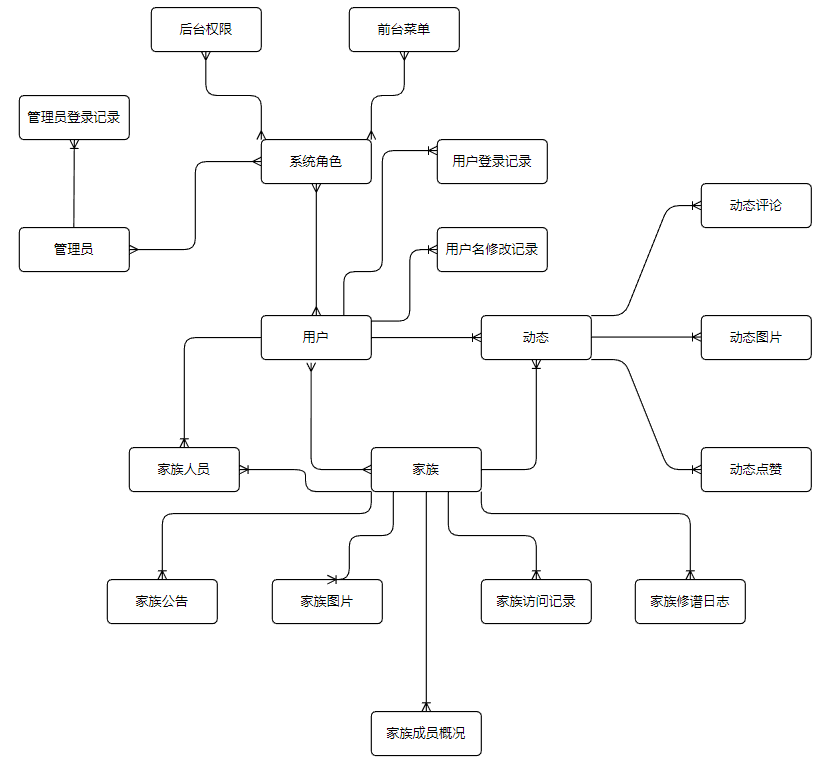


图4.6 数据库逻辑设计实体间关系 E-R 图

### 4.3.2 逻辑结构设计

通过4.3.1中对数据库的概念结构设计以及对各实体间关系的分析，结合MySQL数据的语法特点和系统模块设计，百家谱服务系统的各模块数据库逻辑结构表设计如下：

用户模块（ifamily-user）：用户模块主要包含基本信息、登录记录、用户名修改记录共3张数据库表，各数据库表包含的所有字段和数据类型如下所示。

1. 用户表（user）：保存用户基本信息。

表4.2 用户表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| username | varchar(32) | 用户名，唯一约束 |
| phone | varchar(32) | 手机号，唯一约束 |
| email | varchar(128) | 邮箱，唯一约束 |
| password | varchar(64) | 密码，加密存储 |
| nickname | varchar(32) | 用户昵称 |
| avatar | varchar(128) | 用户头像图片地址 |
| signature | varchar(32) | 个性签名 |
| status | tinyint | 禁用状态：[0]启用 [1]禁用 |
| last\_login | datetime | 上次登录时间 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |

1. 用户登录记录表（user\_login\_log）：保存用户账号登录记录。

表4.3 用户登录记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| ip | char(16) | IP 地址 |
| location | varchar(128) | IP 归属地 |
| device | varchar(128) | 登录设备名称 |
| login\_datetime | datetime | 登录时间 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| user\_id | bigint | 用户ID，外键 |

1. 用户用户名修改记录表（username\_update\_log）：保存用户用户名修改记录，用户名作为系统的唯一ID标识，每个用户一年内只能修改一次。

表4.4 用户用户名修改记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| old\_username | varchar(32) | 旧用户名 |
| new\_username | varchar(32) | 新用户名 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| user\_id | bigint | 用户ID，外键 |

社交模块（ifamily-social）：社交模块主要包含动态、动态评论、动态点赞、动态图片共4张数据库表，各数据库表包含的所有字段和数据类型如下所示。

1. 社交动态表（moment）：保存用户发表的社交动态。

表4.5 用户社交动态表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| content | varchar(1024) | 动态内容 |
| who\_can\_see | tinyint | 动态可见范围 |
| schduled | datetime | 定时发表时间 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| user\_id | bigint | 用户ID，外键 |
| genealogy\_id | bigint | 家族ID，外键 |

1. 动态评论表（moment\_comment）：保存用户社交动态对应的评论。

表4.6 用户社交动态评论表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| content | varchar(1024) | 评论内容 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| moment\_id | bigint | 动态ID，外键 |
| source\_user\_id | bigint | 评论者用户ID，外键 |
| target\_user\_id | bigint | 被回复者用户ID，外键 |
| parent\_id | bigint | 父级评论ID，外键 |

1. 动态点赞表（moment\_like）：保存用户社交动态对应的点赞人员。

表4.7 用户社交动态点赞表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| username | varchar(32) | 点赞者用户名 |
| avatar | varchar(128) | 点赞者头像图片地址 |
| nickname | varchar(32) | 点赞者昵称 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| user\_id | bigint | 点赞者用户ID，外键 |
| moment\_id | bigint | 动态ID，外键 |

1. 动态图片表（moment\_photo）：保存用户社交动态对应的图片链接。

表4.8 用户社交动态图片表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| url | varchar(128) | 图片地址 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| moment\_id | bigint | 动态ID，外键 |
| uploader\_user\_id | bigint | 上传者用户ID，外键 |

第三方模块（ifamily-manager）：第三方模块主要包含验证码发送记录、七牛云令牌获取记录共2张数据库表，各数据库表包含的所有字段和数据类型如下所示。

1. 验证码发送记录表（code\_send\_log）：保存用户注册时发送验证码的限制信息，每个用户每天发送注册验证码的次数不能超过5次[28]。

表4.9 验证码发送记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| receiver | varchar(128) | 收件人 |
| code | char(6) | 验证码 |
| type | tinyint | 验证码类型 |
| is\_success | tinyint | 发送成功：[0]否 [1]是 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |

1. 七牛云令牌获取记录表（qiniu\_token\_log）：保存用户上传图片时获取的七牛云令牌记录。

表4.10 七牛云令牌获取记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| bucket | varchar(32) | 存储桶名称 |
| file\_key | varchar(128) | 文件名 |
| cdn | varchar(128) | CDN 域名 |
| token | varchar(512) | 令牌 |
| is\_success | tinyint | 获取成功：[0]否 [1]是 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| user\_id | bigint | 用户ID，外键 |

家族模块（ifamily-manager）：家族模块主要包含家族信息、家族大事、家族公告、家族人员、家族图片、家族成员概况、家族修谱日志、家族访问记录、用户家族关系表共9张数据库表，各数据库表包含的所有字段和数据类型如表所示。

1. 家族信息表（genealogy）：保存家族名称、地址、祖籍地址等基本信息。

表4.11 家族信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| cover | varchar(256) | 家族封面图片地址 |
| title | varchar(32) | 家族名称 |
| surname | varchar(32) | 家族姓氏 |
| address | varchar(256) | 家族地址 |
| ancestry\_address | varchar(256) | 祖籍地址 |
| introduction | varchar(1024) | 家族简介 |
| generation\_song | varchar(1024) | 家族字辈歌 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| creator\_user\_id | bigint | 创建者用户ID，外键 |

1. 家族大事表（genealogy\_memorabilia）：保存家族大事信息。

表4.12 家族大事表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| title | varchar(128) | 家族大事标题 |
| occurred\_year | date | 发生年份 |
| cover | varchar(256) | 大事配图图片地址 |
| content | varchar(1024) | 家族大事内容 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 家族ID，外键 |
| publisher\_user\_id | bigint | 发布者用户ID，外键 |

1. 家族公告表（genealogy\_notice）：保存家族公告信息。

表4.13 家族公告表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| content | varchar(1024) | 公告内容 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 家族ID，外键 |
| announcer\_user\_id | bigint | 发布者用户ID，外键 |

1. 家族人员信息表（genealogy\_people）：保存家族人员信息。

表4.14 家族人员信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| portrait | varchar(256) | 人员肖像图片地址 |
| surname | varchar(32) | 姓氏 |
| name | varchar(32) | 姓名 |
| gender | tinyint | 性别：[0]男 [1]女 |
| generation | smallint | 世代 |
| generation\_name | varchar(32) | 字辈 |
| seniority | tinyint | 排行 |
| phone | varchar(32) | 电话 |
| residence | varchar(256) | 常住地 |
| birthdate | date | 出生日期 |
| lunar\_birthdate | tinyint | 农历出生日期：[0]否 [1]是 |
| birthplace | varchar(256) | 出生地 |
| death\_date | date | 逝世日期 |
| lunar\_death\_date | date | 农历逝世日期：[0]否 [1]是 |
| burial\_place | varchar(256) | 埋葬地 |
| remark | varchar(256) | 特殊家庭关系备注 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| father\_id | bigint | 生父ID，外键 |
| mother\_id | bigint | 生母ID，外键 |
| husband\_id | bigint | 丈夫ID，外键 |
| genealogy\_id | bigint | 家族ID，外键 |
| user\_id | bigint | 用户ID，外键 |

1. 家族图片（genealogy\_photo）：保存家族图片信息。

表4.15 家族图片表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| url | varchar(256) | 家族图片地址 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 家族ID，外键 |
| uploader\_user\_id | bigint | 上传者用户ID，外键 |

1. 家族成员概况信息表（genenalogy\_profile）：保存家族成员概况信息。

表4.16 家族成员概况表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| total | bigint | 总人数 |
| male | bigint | 男性人数 |
| female | bigint | 女性人数 |
| alive | bigint | 健在人数 |
| death | bigint | 已逝人数 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 家族ID，外键 |

1. 家族修谱日志表（genealogy\_revision\_log）：保存家族成员变动日志。

表4.17 家族修谱日志表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| operation\_type | tinyint | 操作类型 |
| operation\_date | date | 操作日期 |
| operated\_peopole\_name | varchar(32) | 被操作者姓名 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| operator\_user\_id | bigint | 操作者用户ID，外键 |
| operated\_people\_id | bigint | 被操作者人员ID，外键 |
| genealogy\_id | bigint | 家族ID，外键 |

1. 家族访问记录表（genealogy\_visitor\_log）：保存家族信息查看人员记录。

表4.18 家族访问记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| visited\_date | date | 访问日期 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 被访问家族ID，外键 |
| visitor\_user\_id | bigint | 访问者用户ID，外键 |

1. 用户家族关系表（user\_genealogy\_relation）：保存用户、家族多对多关系。

表4. 19 用户-家族关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| is\_default\_genealogy | tinyint | 是否用户默认家族 |
| is\_genealogy\_admin | tinyint | 是否家族管理员 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| genealogy\_id | bigint | 家族ID，外键 |
| user\_id | bigitn | 用户ID，外键 |

管理员模块（ifamily-backend）：管理员模块主要包含管理员信息、管理员登录日志共2张数据库表，各数据库表包含的所有字段和数据类型如下所示。

1. 管理员表（admin）：保存管理员签名、头像和禁用状态等信息。

表4.20 管理员表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| username | varchar(32) | 用户名 |
| password | varchar(64) | 密码 |
| nickname | varchar(32) | 昵称 |
| signature | varchar(32) | 个性签名 |
| avatar | varchar(256) | 头像图片地址 |
| status | tinyint | 禁用状态：[0]启用 [1]禁用 |
| last\_login | datetime | 上次登录时间 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |

1. 管理员登录记录表（admin\_login\_log）：保存管理员账号登录日志。

表4. 21 管理员登录记录表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| ip | char(16) | IP 地址 |
| location | varchar(128) | IP 归属地 |
| device | varchar(128) | 登录设备名称 |
| login\_datetime | datetime | 登录时间 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |
| is\_deleted | tinyint | 是否删除：[0]否 [1]是 |
| admin\_id | bigint | 管理员ID，外键 |

访问控制模块（ifamily-acl）：访问控制模块主要包含前台菜单、后台权限、系统角色、管理员角色关系、用户角色关系、角色权限关系、角色菜单关系共7张数据库表，各数据库表包含的所有字段和数据类型如下所示。

1. 前台菜单表（menu）：保存前台路由菜单信息。

表4.22 前台菜单表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| path | varchar(64) | 路由路径，唯一约束 |
| title | varchar(64) | 菜单名称 |
| description | varchar(128) | 菜单描述 |
| status | tinyint | 禁用状态：[0]启用 [1]禁用 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |

1. 后台权限表（permission）：保存后台资源权限信息。

表4.23 后台权限表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| path | varchar(64) | 权限路径 |
| name | varchar(64) | 权限名称 |
| description | varchar(128) | 权限描述 |
| status | tinyint | 禁用状态：[0]启用 [1]禁用 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |

1. 系统角色表（role）：保存系统角色信息。

表4.24 系统角色表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| id | bigint | 主键 |
| name | varchar(64) | 角色名称 |
| description | varchar(128) | 角色描述 |
| status | tinyint | 禁用状态：[0]启用 [1]禁用 |
| created | datetime | 创建时间 |
| modified | datetime | 修改时间 |

1. 管理员角色关系表（admin\_role\_relation）：保存管理员和角色多对多关系。

表4.25 管理员-角色关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| admin\_id | bigint | 管理员ID，联合外键 |
| role\_id | bigint | 角色ID，联合外键 |

1. 用户角色关系表（user\_role\_relation）：保存用户和角色多对多关系。

表4.26 用户-角色关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| user\_id | bigint | 用户ID，联合外键 |
| role\_id | bigint | 角色ID，联合外键 |

1. 角色菜单关系表（role\_menu\_relation）：保存角色和菜单多对多关系。

表4. 27角色-菜单关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| role\_id | bigint | 角色ID，联合外键 |
| menu\_id | bigint | 菜单ID，联合外键 |

1. 角色权限关系表（role\_permission\_relation）：保存角色和权限多对多关系。

表4.28 角色-权限关系表

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 备注 |
| role\_id | bigint | 角色ID，联合外键 |
| permission\_id | bigint | 权限ID，联合外键 |

## 4.4 安全性设计

通过在后端服务中整合Spring Cloud Gateway、Spring Security、OAuth2三大组件，为百家谱服务系统提供了微服务权限管理的最终解决方案。所有的登录认证请求交由认证服务器ifamily-auth处理，所有的权限和路由功能交由网关服务ifamily-gateway处理，ifamily-service专注于处理自己的业务，这样的架构使得各个服务可以独立演进，同时保证了系统的安全性和可扩展性。下面将阐述百家谱后端服务中的项目安全控制的具体设计。

1. 业务服务（ifamily-service）：处理用户业务逻辑，无需进行任何安全逻辑校验。所有的安全认证和鉴权工作已经由认证服务器和网关服务完成。业务API服务可以专注于处理与自身业务相关的逻辑，提供相应的接口供客户端调用。
2. 认证服务（ifamily-auth）：作为系统的统一认证中心，处理所有的登录请求认证。通过给客户端提供Token（令牌）和刷新令牌（Refresh Token）的方式来完成客户端身份的识别和认证[29]。实现时可以通过集成Spring Security、OAuth2、JWT等已有的安全控制框架来实现认证功能，核心业务逻辑伪码如下：

|  |
| --- |
| // 从请求头中获取客户端 ID  String clientId = this.httpServletRequest.getParameter("client\_id");  // 根据不同的客户端调用不同远程服务验证用户信息正确性  if ("ADMIN\_CLIENT".equals(clientId)) {      userDTO = this.adminFeignService.loadAdminByUsername(username);  } else if ("MOBILE\_CLIENT".equals(clientId)) {      userDTO = this.userFeignService.loadUserByUsername(username);  }  // 将通过认证的用户转换为 Spring Security 安全框架的认证用户对象 UserDetails  SecurityUser securityUser = new SecurityUser(userDTO);  // 用户登录请求身份信息验证通过，签发 token 返回给客户端  OAuth2AccessToken token = this.tokenEndpoint.postAccessToken(request.getUserPrincipal(), parameters).getBody(); |

1. 网关服务（ifamily-gateway）：作为微服务架构的唯一入口，负责路由和鉴权。所有的请求首先会经过网关服务进行鉴权，然后再路由到相应的业务API服务。在鉴权方面，通过结合Spring Security和OAuth2来实现对访问令牌的验证和权限校验，确保只有通过了鉴权的请求才会被转发到具体的业务API服务。具体的鉴权流程如图4.7所示：

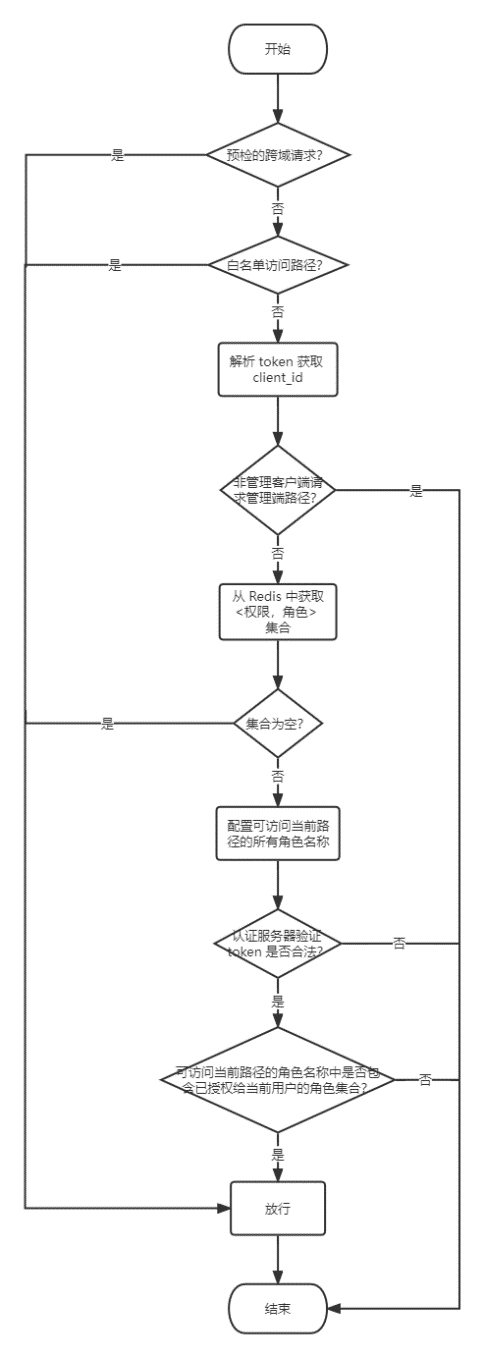


图4.7 网关鉴权流程图

当然，除了在访问权限控制逻辑上对系统资源进行安全保护外，还可以使用加密技术对系统资源进行安全保护，包括使用对称加密算法和非对象加密算法保护家族信息等。除此之外，定期的安全审计和漏洞扫描也是确保系统资源安全的关键步骤。通过以上安全措施的使用将大大降低未经授权访问和恶意攻击对系统资源造成的风险，提高系统的整体安全性和稳定性。

## 4.5 核心算法流程描述

家族成员树谱的查询和展示需要用到递归算法。以下将详细分析后端服务如何构建出家族成员树型结构数据，以及前台应用系统如何解析展示递归数据，从而实现家族树功能。

首先是后端服务如何实现从MySQL数据库中查询出当前家族的成员家谱树数据。根据4.4中家族成员数据库表结构设计可知，每个成员均存在一个生父ID、生母ID以及丈夫ID，生成树型结构数据的关键在于使用深度优先算法查询当前家族人员的孩子、查询孩子的孩子，如此递归，直到某个家族成员不存在孩子节点，在递归的过程中依次查询当前节点人员的妻子列表，从而实现了完整的家族人员树型数据查询，算法流程伪码描述如下：

|  |
| --- |
| // 查询家族人员树谱数据  public MemberTreeNodeBO memberTree(Long genealogyId) {      // 根据当前家族的最小世代查询人员信息，此即为家族的祖先  MemberTreeNodeBO ancestor = this.getOne(queryWrapper);      // 获取祖先的妻子      ancestor.setMates(this.listWives(ancestor.getId()));      // 获取祖先的孩子      this.listChildren(ancestor);      return ancestor;  }  // 查询丈夫的妻子列表  private List<MemberTreeNodeBO> listWives(Long husbandId) {  // 从数据库家族人员信息表中查询出丈夫 ID 为 husbandId 的人员即为当前人员的妻子  }  // 查询父亲的孩子列表  private void listChildren(MemberTreeNodeBO parent) {      // 查询父亲下的所有孩子列表      List<PeopleDO> childList = this.list(queryWrapper);      if (childList == null || childList.isEmpty()) {          return;      }      List<MemberTreeNodeBO> children = new ArrayList<>();      // 遍历所有孩子，依次查询其妻子和孩子  childList.forEach(item -> {          // 创建孩子 BO          MemberTreeNodeBO child = new MemberTreeNodeBO();          BeanUtils.copyProperties(item, child);          // 查询孩子的妻子列表          child.setMates(this.listWives(child.getId()));          // 递归查询孩子的孩子列表          listChildren(child);          // 将孙子添加到孩子结果列表中          children.add(child);      });      // 将孩子列表设置为父节点的子节点      parent.setChildren(children);  } |

最后是前台应用系统如何展示后端递归生成的家族成员树型JSON数据，如图4.8所示。



图4.8 家族成员树型 JSON 数据示意图

诚然，递归生成的树型结构数据自然需要使用递归算法来解析展示。前台系统家族树谱数据的展示较之后端服务更为复杂之处在于需要控制当前根节点的孩子节点的展开状态，以及根据当前节点的人员性别来展示默认人物肖像图片和姓名颜色，从而做到人员性别的辨别一目了然，以及需要在递归生成的树型结构中定位到某个家族成员，实现逻辑比后端服务更为繁杂。具体期望的页面展示效果如图4.9所示：



图4.9 家族成员树谱效果展示图

经过对后端响应数据的深层次分析，并结合预期的效果展示图，前台系统的算法实现依旧是使用深度优先遍历算法实现数据遍历，在遍历的过程中结合Vue动态给DOM元素添加样式名和类名的方式实现根据性别展示不同的默认人物肖像和姓名颜色。算法代码实现如下：

|  |
| --- |
| <template>    <div class="chart">      <table v-if="tree.name" class="single">        <tr>          <td :colspan="Array.isArray(tree.children) ? tree.children.length \* 2 : 1"              :class="{'extend': Array.isArray(tree.children) && tree.children.length && tree.extend}">            <div :class="{'couple': true, 'mate': tree.mates}">              <!-- 当前节点 -->              <div class="person" @click="$emit('click-node', tree)"                   :class="[Array.isArray(tree.class) ? tree.class : [], `pid-${tree.id}`, tree.gender === 0 ? 'male' : 'female']">                <div class="avatar">                  <img :src="tree.portrait || defaultPortrait(tree.gender)" alt="portrait"/>                </div>                <div class="name">{{ tree.name }}</div>              </div>              <!-- 伴侣节点 -->              <template v-if="Array.isArray(tree.mates) && tree.mates.length">                <div class="person" v-for="(mate, mateIndex) in tree.mates" :key="tree.name+mateIndex"                     :class="[Array.isArray(mate.class) ? mate.class : [],`pid-${mate.id}`, mate.gender === 0 ? 'male' : 'female']"                     @click="$emit('click-node', mate)">                  <div class="avatar">                    <img :src="mate.portrait || defaultPortrait(mate.gender)" alt="portrait"/>                  </div>                  <div class="name">{{ mate.name }}</div>                </div>              </template>            </div>            <!-- 展开/折叠箭头 -->            <div class="arrow" v-if="Array.isArray(tree.children) && tree.children.length"                 @click="tree.extend = !tree.extend; $emit('center-node', $event.target.previousSibling)">            </div>          </td>        </tr>        <!-- 递归生成孩子节点 -->        <tr v-if="Array.isArray(tree.children) && tree.children.length && tree.extend">          <td v-for="(children, index) in tree.children" :key="index" colspan="2" class="child">            <family-tree-branch :tree="children" :single="tree.children.length === 1"                                @click-node="$emit('click-node', $event)"                                @center-node="$emit('center-node', $event)"            />          </td>        </tr>      </table>    </div>  </template> |

通过在后端服务和前台系统中使用改良的深度优先算法，较好地展示了家族成员的树谱数据，使用如此直观和易于理解的方式展示家族成员之间的关系和层级，丰富了家族成员列表的展示方式，很好地提升了用户的体验感和家族信息的丰富性。

# 第5章 系统实现

## 5.1 前台应用系统

### 5.1.1 用户模块

用户注册：用户通过正确的邮箱地址申请注册为百家谱用户，注册过程中需要验证邮箱验证码是否正确以及用户输入的邮箱地址是否已被其它用户占用，前后端还将共同校验用户输入的密码难度是否符合系统要求，从而保证用户的账号安全。相关页面如图5.1和图5.2所示：

|  |  |
| --- | --- |
| 图5.1 用户注册 | 图5. 邮箱验证码 |

用户登录：用户可以通过输入UID、手机或邮箱和登录密码登入系统，在登录系统的过程中系统会验证账户名和密码的正确性。相关页面如图5.3~图5.5所示：

|  |  |  |
| --- | --- | --- |
| 图5.3 用户信息不存在 | 图5. 账号密码错误 | 图5. 账户已被禁用 |

用户信息修改：用户可修改个人昵称、签名以及上传头像。页面如图5.6和图5.7所示：

|  |  |
| --- | --- |
| 图5.6 修改用户昵称 | 图5. 上传头像图片 |

用户隐私修改：可修改用户UID、手机号或邮箱以及用户登录密码等。UID作为百家谱系统的唯一用户标识，一年内只允许修改一次。在修改用户隐私信息的过程中需要对用户身份进行安全校验以保护用户账号安全，如修改用户名时需要输入登录密码、修改登录密码时需要获取验证码等。相关页面如图5.8~图5.10所示：

|  |  |  |
| --- | --- | --- |
| 图5.8 修改用户UID | 图5. 修改邮箱地址 | 图5. 修改登录密码 |

账号安全：包含注销账号和查看用户登录设备功能，相关页面如图5.11和图5.12所示：

|  |  |
| --- | --- |
| 图5.11 登录设备 | 图5. 注销账号 |

### 5.1.2 主页模块

主页模块功能包含查看百家谱轮播图和浏览成员动态功能，家族成员动态支持下拉刷新、点赞、评论以及点击头像查看家族成员用户信息等操作。相关页面如图5.13~图5.15所示：

|  |  |  |
| --- | --- | --- |
| 图5.13 点赞动态 | 图5. 评论动态 | 图5. 查看用户 |

### 5.1.3 家族模块

家族信息：包含家族创建者新增家族、家族修委会成员编辑家族信息以及家族成员浏览家族信息等功能。相关页面如图5.16~图5.18所示：

|  |  |  |
| --- | --- | --- |
| 图5.16 创建家族 | 图5. 编辑家族 | 图5. 查看家族 |

家族成员查看：包含查看家族成员树谱、在家族树中定位某个家族成员以及查看家族成员世代成员列表过滤展示功能。相关页面如图5.19~图5.21所示：

|  |  |  |
| --- | --- | --- |
| 图5.19 家族树谱 | 图5. 定位成员 | 图5. 世代成员列表 |

家族成员信息管理：家族修委会可基于已经存在的家族成员添加成员亲人、编辑家族成员信息、移除家族成员，家族成员可查看家族人员信息。相关页面如图5.22~图5.24所示：

|  |  |  |
| --- | --- | --- |
| 图5.22 查看人员 | 图5. 添加亲人 | 图5. 编辑人员 |

家族修谱记录：包含家族修谱日志和家族访问记录。家族修谱日志中主要展示了家族修委会管理员对家族成员进行的新增、编辑和移除操作。相关页面如图5.25~图5.26所示：

|  |  |
| --- | --- |
| 图5.25 家族修谱日志 | 图5. 家族访问记录 |

## 5.2 后台管理系统

### 5.2.1 权限管理

管理员登录：管理员登录后台管理系统。根据账号对应的系统角色不同所展示的路由菜单也不尽相同。登录页面如图5.27所示：



图5.27 管理员登录

后台权限资源管理：超级管理员可在后台管理系统中管理后端服务所有权限资源。如图5.28所示，当新增或编辑后端服务权限时，系统将自动查询出后端服务中所有控制器方法的请求路径（此即为后端服务权限资源），并使用Ant匹配规则自动勾选已经配置过的权限资源，使得管理员可以对已配置和未配置的系统权限资源一目了然。再配合数据库对权限路径的唯一性约束，在极大程度上保证了后端服务资源的有效配置，进一步保障了资源有效性。

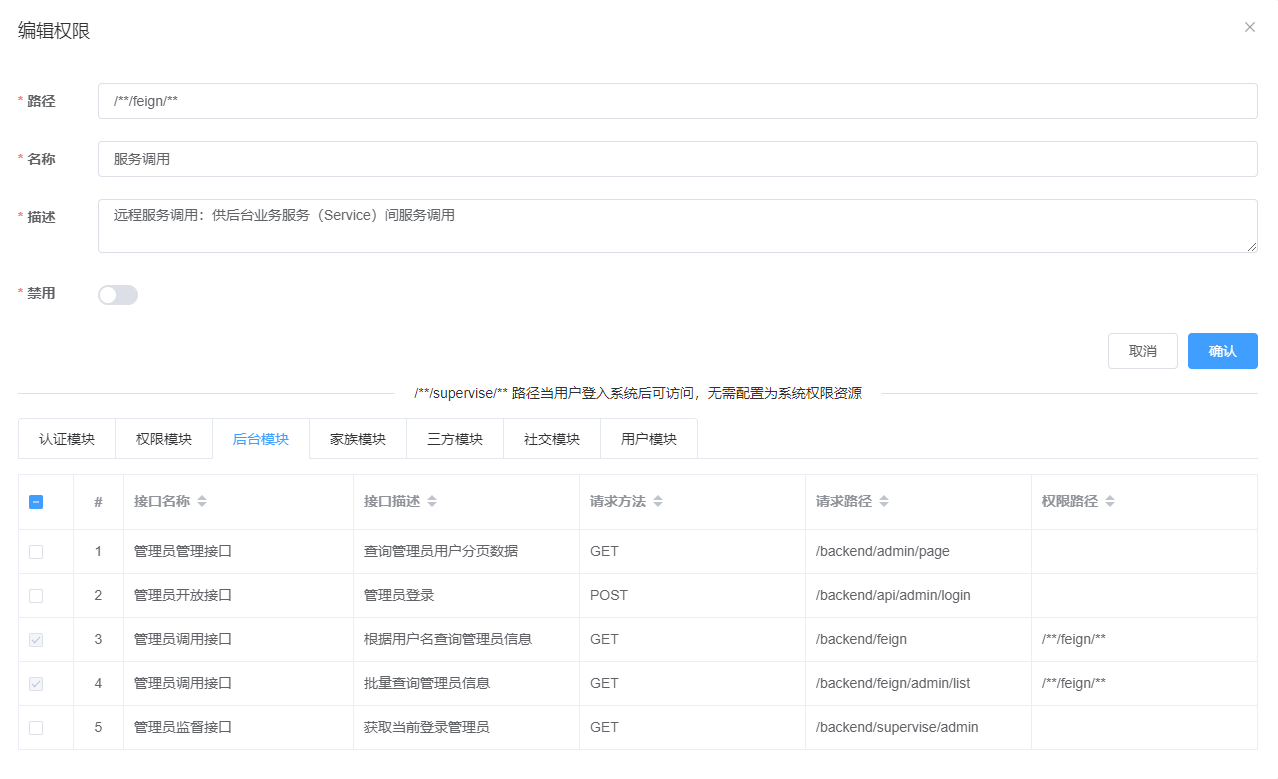


图5.28 系统权限资源管理

前台路由菜单管理：当超级管理员在后台管理系统中配置路由菜单权限时，系统将自动过滤出当前路由配置中所有需要访问控制的路由路径，并以树型结构数据展示，同后台权限资源管理一样，也将自动勾选已配置过的路由路径，相关功能具体实现如图5.29所示：

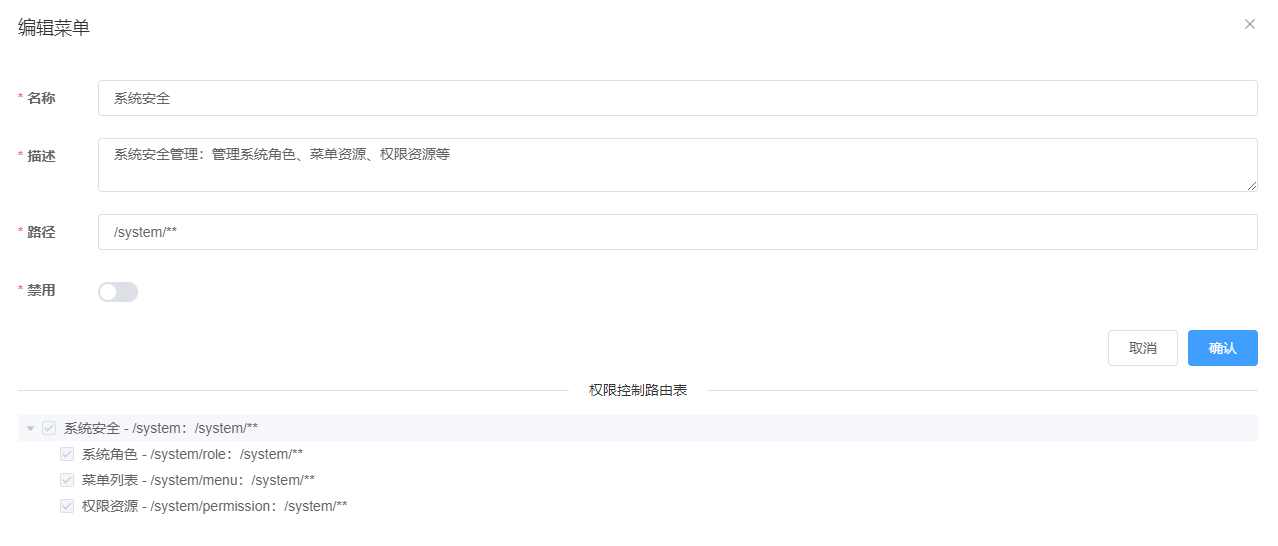


图5.29 前台路由菜单管理

系统角色管理：超级管理员在后台管理系统中配置管理系统角色时，系统将查询出所有的前台菜单（Menu）资源和后台权限（Permission）资源，管理员通过勾选当前角色拥有的菜单权限和资源权限即可为当前角色分配系统权限。前台菜单权限意味着用户对路由菜单的可见性，后台权限资源意味着用户能否请求某个API接口。通过两层细粒度的用户权限控制，极大程度上保证了家族数据的保密性和系统的安全性。功能界面实现如图5.30所示：



图5.30 系统角色管理

当超级管理员查看角色的详情信息时，系统将弹窗展示当前角色的详细信息、角色拥有的菜单权限和权限资源列表，以及当前角色已经分配给的用户和管理员。如图5.31所示：



图5.31查看角色详情

### 5.2.2 数据管理

如图5.32所示，在后台管理系统的数据管理模块中管理员可操作备份百家谱系统所有数据库SQL脚本文件或导出对应数据库表的Excel表数据。

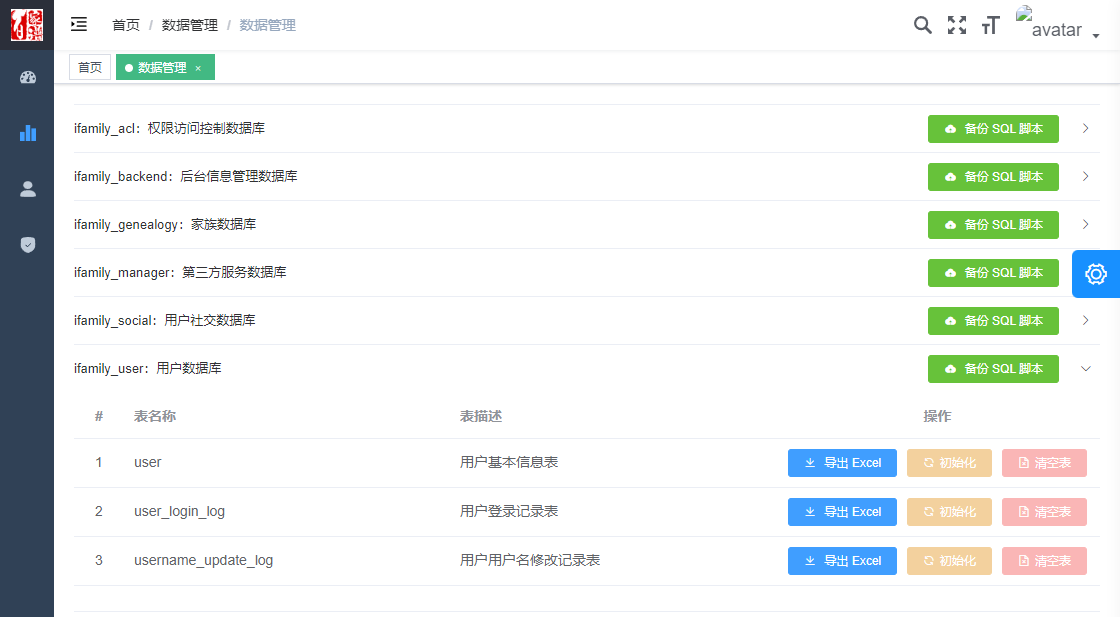


图5.32 备份数据库或导出数据库表

对应的数据库SQL脚本文件内容如图5.33所示，包含了当前数据库中的所有表结构以及表数据，当数据库发生故障时管理员可通过此备份脚本快速恢复数据库数据。

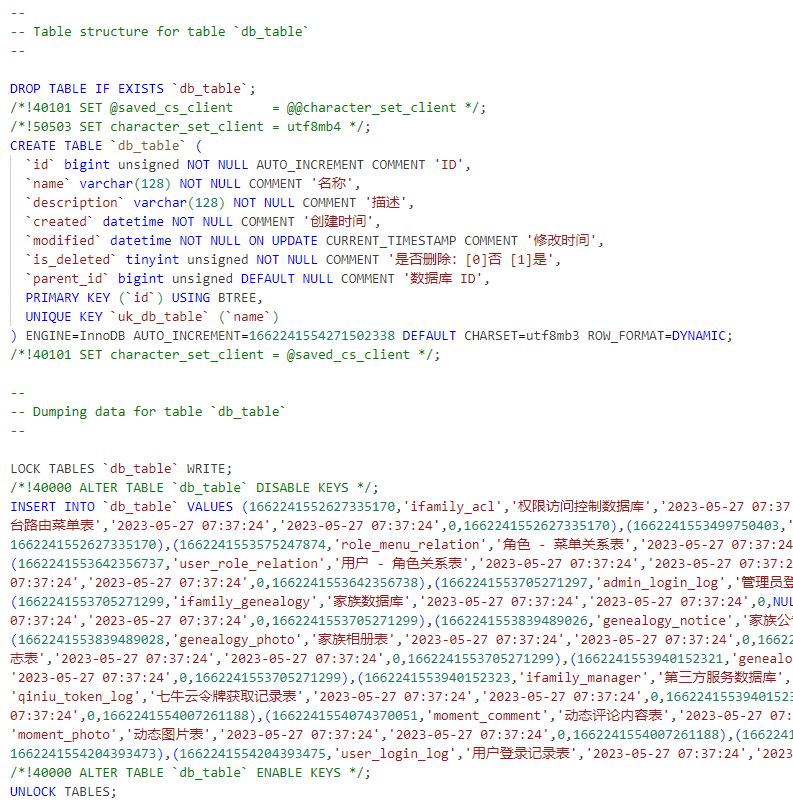


图5.33 数据库SQL脚本文件内容

导出对应的数据库表的Excel文件数据如图5.34所示，工作表中包含了表头以及各行记录数据，更利于非专业开发人员对数据进行分析。

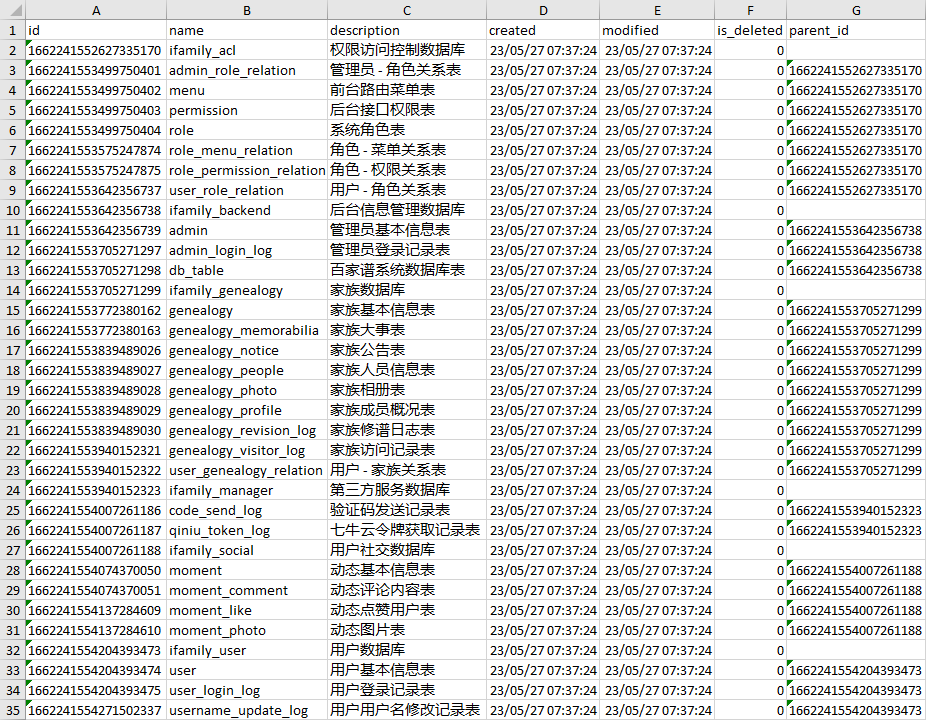


图5.34 数据库表Excel文件内容

### 5.2.3 用户管理

如图5.35所示，系统管理员可在后台管理系统的用户管理模块中管理百家谱前台用户和后台管理员账户，包括设置账户基本信息、禁用账户和分配角色等。

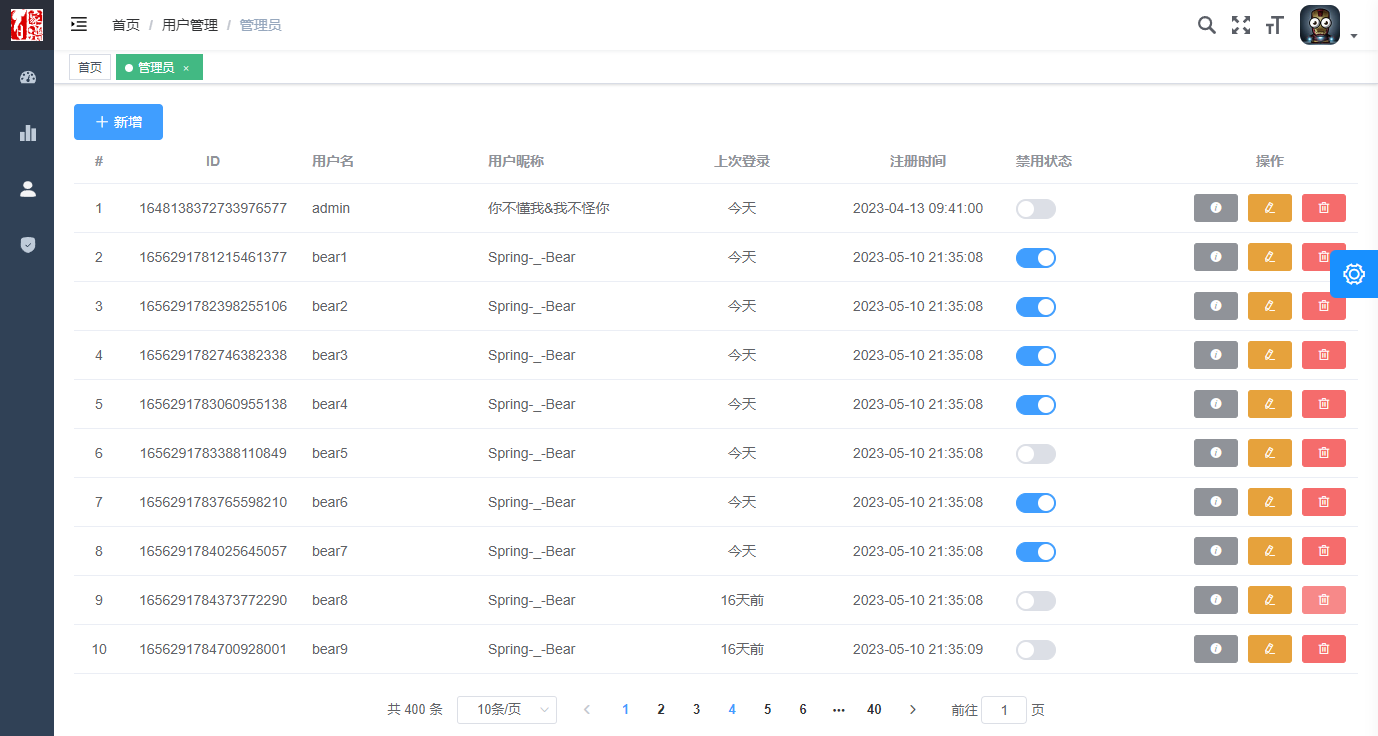


图5.35 管理系统账户

如图5.36所示，系统管理员通过点击用户列表中某个用户的禁用状态按钮即可设置某个百家谱账户的禁用状态，其中0标识为启用中，1标识为已禁用。

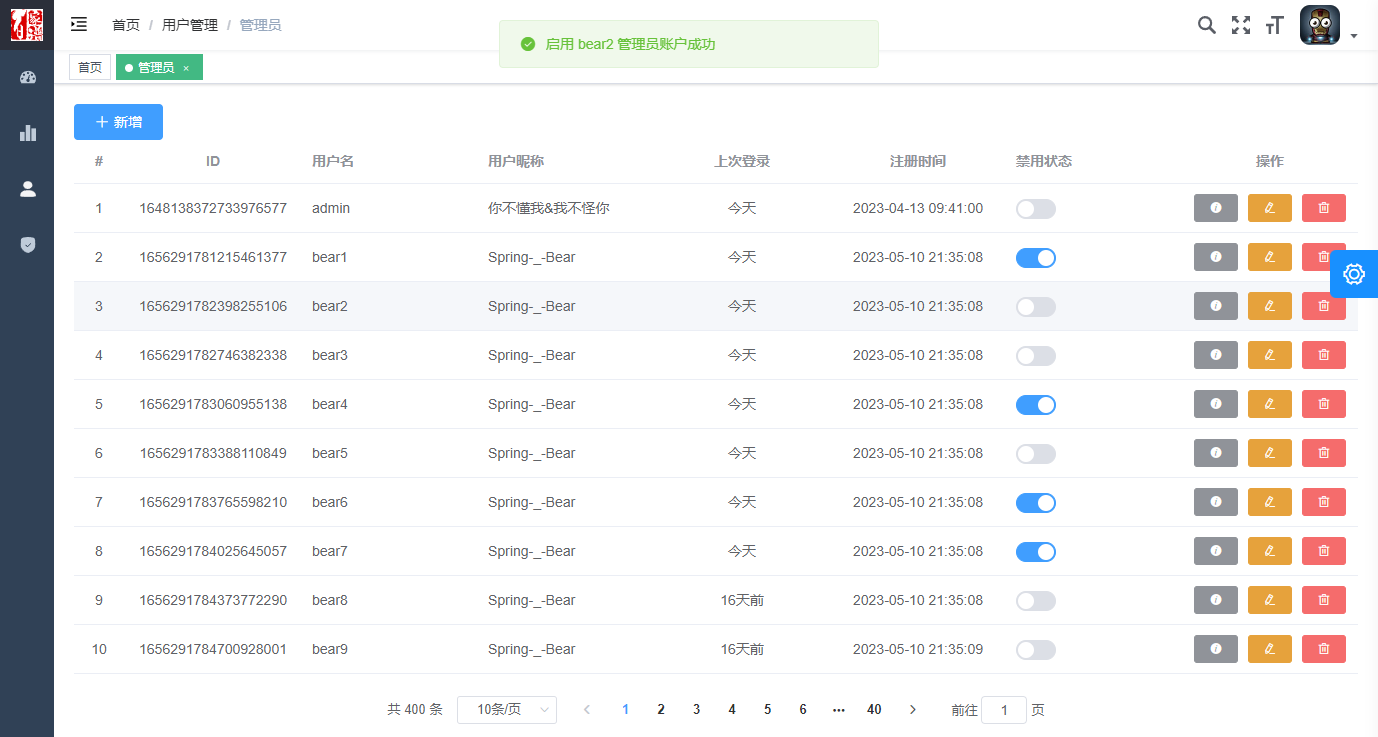


图5.36 启用管理员账户

当管理员查看当前账户详情时，将弹窗展示当前账户的详细信息，例如注册时间、上次登录时间、对应的系统角色等，并且可以为当前账户添加或移除系统角色。详情如图5.37所示。



图5.37 添加或移除系统角色

## 5.3 后端服务系统

### 5.3.1 系统文档

后端服务系统统一接口文档：通过在后端服务系统的网关模块中引入Knife4j文档依赖的相关UI包，在网关处聚合所有业务API接口，从而实现通过网关访问系统接口文档。效果如图5.38所示：



图5.38 网关统一接口文档

如图5.39所示，可以通过在接口文档的UI界面中完成对后端服务所有API接口的测试，相较于使用Postman客户端测试更加便捷，并且支持接口访问权限控制。

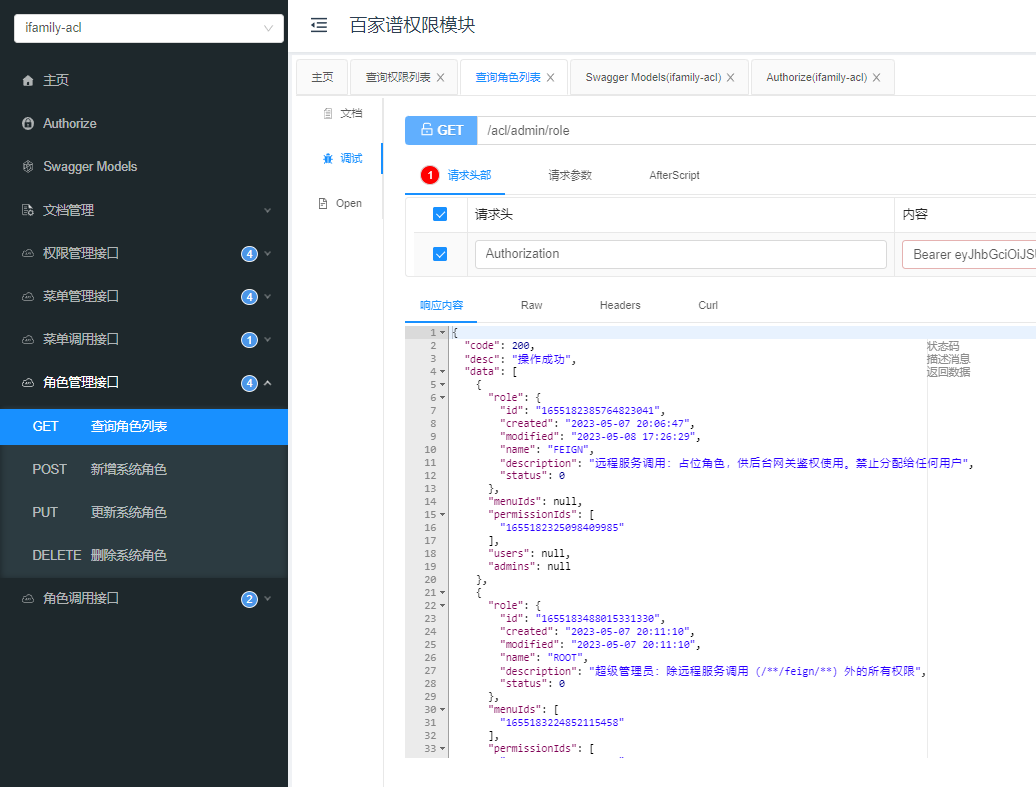


图5.39 网关API接口测试

### 5.3.2 工程结构

如图5.40所示，其为后端微服务架构模块工程结构图。下面将对各个模块的主要功能进行说明：ifamily-server模块作为系统的顶级父工程管理项目依赖版本；ifamily-auth作为分布式系统的统一认证服务器；ifamily-common作为其余所有模块的依赖，包含常用的配置类、工具类以及实体类等，部署时不需要打包运行；ifamily-gateway作为系统网关完成请求路由和统一认证鉴权，对外暴露API接口；ifamily-service作为其余具体业务模块的父工程管理业务模块的通用依赖，其下的子工程各自提供不同的业务服务，如ifamily-acl对系统的访问权限控制进行管理、ifamily-genealogy为用户提供家族服务等。各个模块间互相独立却又相辅相成，共同为百家谱家族服务系统提供数据业务支持。

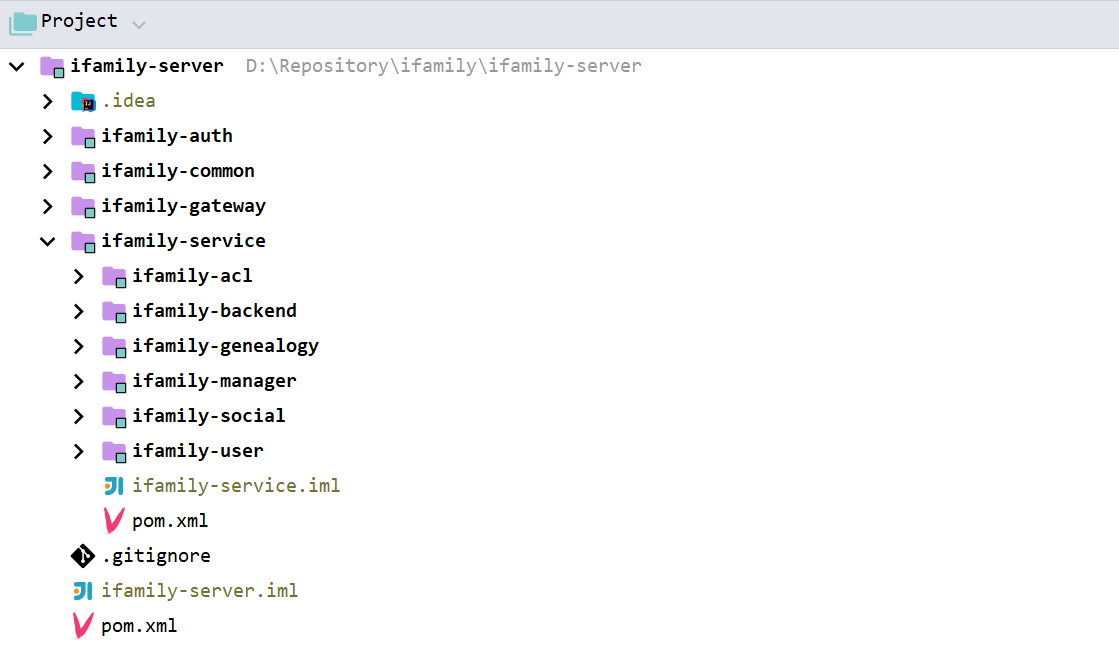


图5.40 后端服务系统工程结构图

## 5.4 系统测试

1. 单元测试：单元测试是软件开发中的一种测试方法，用于验证程序中的最小功能模块（通常是一个函数或方法）是否按预期工作。它是软件测试中的基本组成部分，有助于提高代码质量、发现和修复潜在的错误[30]。如图5.41所示，该单元测试方法实现对权限访问控制数据库ifamily\_acl的数据库表信息进行初始化：



图5.初始化acl数据库单元测试方法

在第75行中使用了断言方法对单元测试结果进行判定，若save方法返回结果为false则断言失败，意味着单元测试未通过。测试结果如图5.42所示：

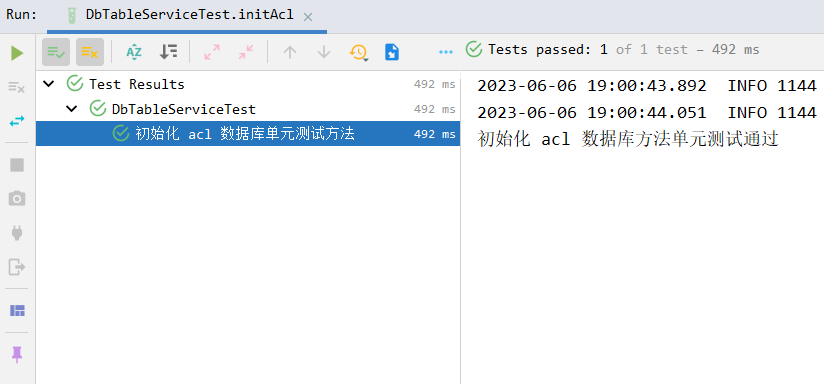


图5.42 初始化acl数据库单元测试通过

1. 远程过程调用（RPC）测试：RPC是一种用于不同计算机或进程之间进行通信和调用的技术。在进行RPC远程过程调用测试时，主要目标是验证远程调用的正确性和可靠性[31]。如图5.43所示，其通过Postman工具实现对百家谱后端服务系统的HTTP请求接口进行测试，具体测试了管理员登录接口，需携带服务器必要请求参数，测试通过则返回用户令牌和刷新令牌等信息，测试结果如下所示：

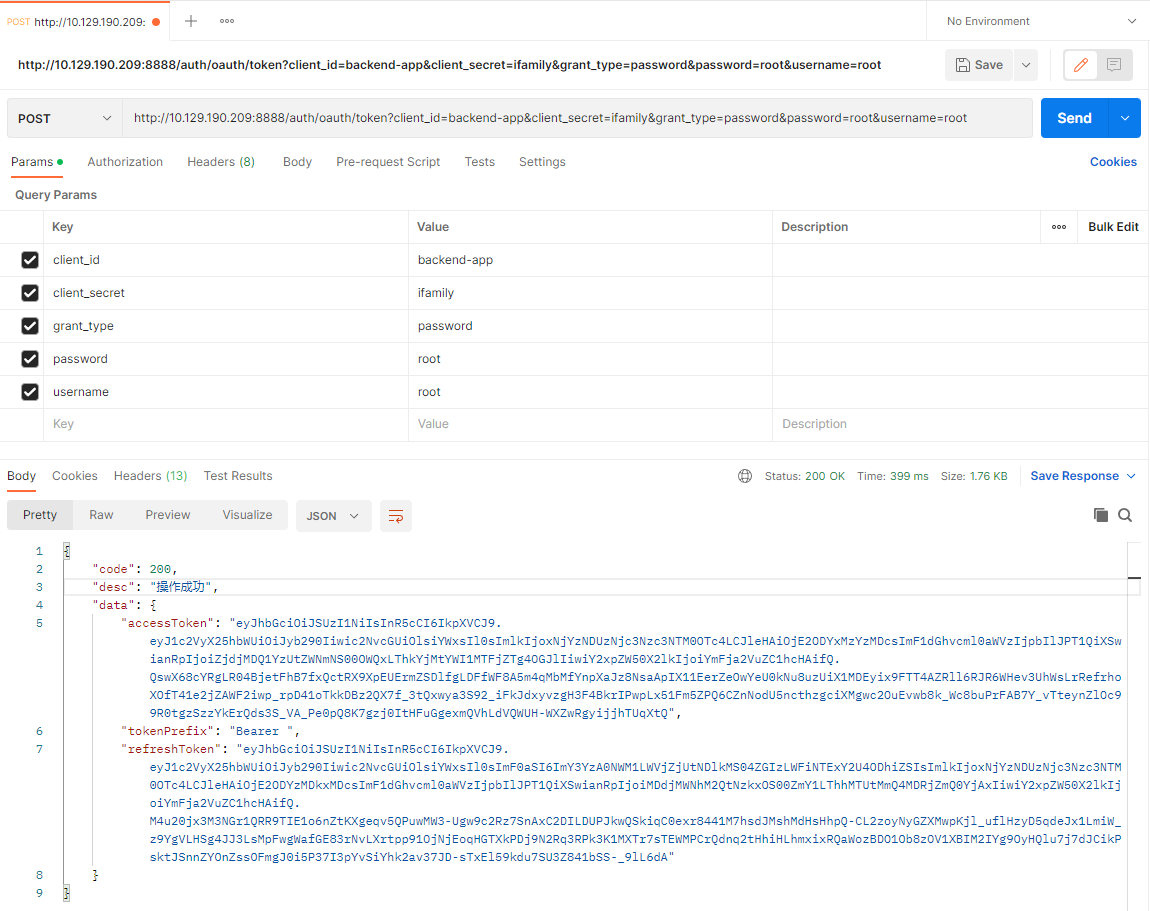


图5.43 管理员登录RPC接口测试

1. 压力测试：压力测试是一种测试方法，用于评估系统在预期工作负载下的性能表现。它主要关注系统在高负载条件下的稳定性、吞吐量、响应时间和资源利用率等指标。进行压力测试的目的是确定系统的性能极限、找出瓶颈和性能问题，并为系统提供容量规划和优化建议。如图5.44所示，使用JMeter系统测试工具对百家谱后端服务系统家族模块家族树谱核心功能进行了压力测试，模拟100个用户并发访问系统，共访问100次，总共发出10000次请求。



图5.44 家族树谱接口压力测试

测试结果如图5.45所示，家族树谱接口平均响应时间为736毫秒，吞吐量为132.7次/秒，90%的请求在1065毫秒内完成，完成一万次请求错误请求次数为0，详细的压力测试信息如下图所示：



图5.45 家族树谱接口压力测试结果

# 第6章 结束语

## 6.1 总结

本研究通过分析传统纸质家谱存在的诸多缺点与不足，如易丢失损毁、纸张容量有限、存储内容不够丰富等，发现当下越来越多的家族选择将家族信息数字化。然而，现有的家族APP或网站不能很好地满足家族的需求，要么功能过于单一，要么页面设计过于冗杂。故而本研究旨在设计和开发一款现代化、多功能、易上手的综合性数字化家谱服务系统，以满足当下家族对家族管理和文化传承的迫切需求。

研究过程中通过用户需求调研、同类竞品对比等方式分析得出本系统应该包含的具体功能，结合现阶段企业和组织最流行的开发技术，对系统的前后端架构、工程模块、数据库和核心算法等进行了详细的设计，并基于已有的分析设计对系统进行了实现。细分为移动端前台应用系统和桌面端后台管理系统。前台应用系统给家族成员提供了丰富的家族数据展示功能，如家族成员树、世代成员列表、家族公告、家族相册、家族大事记和成员社交动态等，促进了家族成员之间的联系和互动。后台管理系统给家族管理员提供了管理家族相关信息的平台，包括但不限于管理家族成员、家族公告、人物事迹等。通过对两端家谱系统的具体实现，将大力推动传统家谱的现代化转型，满足当代家族对家谱管理和文化传承的需要。

## 6.2 展望

虽然本系统已经满足了当下家族对家谱系统的绝大部分功能需求，但随着技术的不断发展和用户需求的变化，当前家谱服务系统有望进一步发展和完善。以下是一些可能的发展方向和改进点：

1. 引入智能化功能：结合人工智能算法，为家族成员提供智能化的功能和个性化体验。例如，系统可以实现对家族失考人员的人物肖像虚拟、对纸质家谱中的已损信息进行智能化修复或智能化推荐相关的家族历史和传统知识等。
2. 扩展社交互动功能：当前的系统实现中仅仅实现了家族成员的动态展示和交互功能，未来系统可以扩展成员在线聊天、视频通话以及举办线上家族聚会等功能。
3. 多端的应用支持：当前系统仅支持浏览器进行访问，随着业务和用户需求的变化，未来系统应提供更广泛的访问支持，如微信小程序、移动端APP、桌面端应用支持等。

总之，数字化家谱服务系统在满足家族管理和文化传承需求方面具有巨大潜力。未来的发展应注重智能化功能、社交互动及多端支持等方面的改进。通过不断创新和完善，数字化家谱服务系统将为家族管理和文化传承提供更全面、便捷和持续的解决方案。

# 参考文献

1. 杨志丹，蔡跃进．面向内容整理的家谱数字化系统分析与设计[J]．泉州师范学院学报，2018，第36卷(4)：30-34．
2. 顾梦蝶．基于Vue的著作权预约登记微平台系统研发[D]．北京：北京印刷学院，2022．
3. 胡绍轩．基于Spring Cloud的教务管理系统的设计与实现[D]．吉林：吉林大学，2022．
4. 祝振媛．我国传统家谱档案的可视化构建——以浙江仙居高迁《吴氏西宅宗谱》为例[J]．档案学通讯，2020，(2)：72-79．
5. 沈煦然．家谱档案资源整合平台建设研究[D]．黑龙江：黑龙江大学，2022．
6. 张颖莉．中国传统族谱的数字化信息设计研究——以湘西《朱氏族谱》为例[D]．江苏：南京师范大学，2021．
7. 刘莉．浅谈文化生态视角下家谱数字资源的开发与建设[J]．大学，2021，(45)：125-127．
8. 任明，李俊杰，刘雪薇，等．中华传统家谱数据可视化研究[J]．数字人文研究，2021，第1卷(4)：61-76．
9. 姜丰．乡村振兴话家谱-家谱数字化服务平台构建与视觉设计研究[D]．湖北：华中师范大学，2021．
10. 汪超．徽州家谱数据库建设现状、设计与建议[J]．大学图书情报学刊，2021，第39卷(5)：55-60．
11. 谢琳惠．家谱文献资源数字化开发与利用[J]．河南图书馆学刊，2012，第32卷(6)：125-128．
12. 刘晴．数字时代馆藏家族档案资源建设研究[D]．河南：郑州航空工业管理学院，2019．
13. 夏妍．档案馆馆藏家谱档案开发利用研究——以我国省级综合档案馆为调研对象[D]．上海：上海师范大学，2021．
14. Li，Nian；Zhang，Bo．The Research on Single Page Application Front-end development Based on Vue[J]．Journal of Physics: Conference Series，2021，Vol.1883(1)：012030．
15. Song，JH(Song，Junhui)；Zhang，M(Zhang，Min)；Xie，H(Xie，Hua)．Design and Implementation of a Vue.js-Based College Teaching System[J]．International Journal of Emerging Technologies in Learning，2019，Vol.14(13)：59-69．
16. 顾梦蝶．基于Vue的著作权预约登记微平台系统研发[D]．北京：北京印刷学院，2022．
17. Sufyan bin Uzayr．Introduction to MySQL[J]．Mastering MySQL for the Web，2022，1-40．
18. Banu Parasuraman．Spring Cloud Function and IoT[J]．Practical Spring Cloud Function，2022，:253-283．
19. Marten Deinum & Iuliana Cosmina．Spring Applications in the Cloud[J]．Pro Spring MVC with WebFlux，2021，:521-561．
20. 张益．基于微服务的物联网平台的设计和容器部署研究[D]．北京：北京邮电大学，2020．
21. August Lilleaas．Enterprise Authentication Using Spring Security[J]．Pro Kotlin Web Apps from Scratch，2023，: 261-275．
22. Chatterjee，Ayan；Prinz，Andreas．Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study[J]．Sensors，2022，Vol.22(5): 1703．
23. 汪昱．应用程序认证机制安全研究[D]．陕西：西安电子科技大学，2018．
24. 陈红菊．新疆城乡道路客运成品油价格补助管理系统的设计与实现[D]．山东：山东大学，2015．
25. 李颖．基于工作流技术的政府办公自动化系统的设计与实现[D]．天津：天津大学，2010．
26. 彭炳江．基于政府人才云平台的申报审批系统研究与实现[D]．贵州：贵州大学，2021．
27. 郑俊秋．基于微服务的土石方施工调配平台设计与实现[D]．中国石油大学(华东)，2019．
28. 李梅文．爱奇艺会员系统的设计与实现[D]．北京：北京交通大学，2013．
29. 陈军．基于Oauth协议的智慧社区平台的设计与研究[D]．苏州大学，2015．
30. August Lilleaas．Automated Tests with jUnit 5[J]．Pro Kotlin Web Apps from Scratch，2023，:119-139．
31. Peter Michalek；Jakub Kralovanec and Jan Bujnak.Composite Steel and RPC Testing[J]．Pollack Periodica，2020，Vol.15(3)：144-149

# 致 谢

“世事一场大梦，人生几度秋凉”。行文至此，百感交集，思绪纷乱，心潮起伏，感慨万千。一路走来，对于为我提供过帮助的贵人，谨以此篇诚表我最真挚的谢意。

“饮其流者怀其源，学其成时念吾师”。由衷地感谢我的论文指导老师王云华先生，从论文的选题到定稿，导师用渊博的专业知识、严谨的治学态度和丰富的学术经验给予我细致入微的指导和宝贵的建议，让我得以顺利完成研究。在与老师的讨论中，我收获了许多洞见和思考的方法，这对我未来的学习和工作将产生重要影响。同时，我也要感谢学习生涯中每一位教授过我知识的恩师，感谢老师们的悉心教诲和无私奉献，让我拥有了迈向未来的勇气和能力。希望老师们万事胜意，桃李芬芳！

“父母之爱子，则为之计深远”。真切地感谢我的父母二十余载的养育和支持，日复一日年复一年地辛勤劳作，含辛茹苦将我抚养成人，护我饱暖、供我求学，养育之恩无以为报，欲报之德天地难穷。此外，我也要感谢我的家人们，无论处于人生的哪个阶段，家人们始终都是我坚实的后盾和无私的支持者，给予我无微不至的爱和关怀。尤其需要特别感谢已古稀之年的爷爷，二十多年来爷爷始终将我视作自己的掌上明珠，无私地给予我无尽的关爱与教导，没有爷爷的付出就没有今天的我。祝愿家人们身体健康，平安喜乐！

“人生自古谁无死，留取丹心照汗青”。深深地感谢党、国家和社会，以及生我养我的这片土地。从高中到本科毕业，精准扶贫政策给我这个来自寒门的学子带来了无尽的温暖与希望，让我有机会接受优质的教育，追逐属于自己的梦想，安定的求学环境离不开党和国家为教育事业的持续投入和努力。游目八荒，河清海晏，生于盛世，长于盛世，未来盛世更将由我们谱写，待之有为，必报中华。诚盼国家国泰民安，风调雨顺！

“平生感知己，方寸岂悠悠”。诚挚地感谢一直陪伴在我身边的朋友们。解我烦忧与孤苦，予我欣喜与心安。相伴至今，不离不弃，知音难觅，心有灵犀，愿岁并谢，与友长兮！

“路漫漫其修远兮，吾将上下而求索”。感谢一直以来未曾放弃的自己，坚持追求自己的目标和梦想。凡心所向，素履所往，生如逆旅，一苇以航！

四载始，四载终，往事无归途，幸相遇，感相濡，历历在心头。