

学号：202110310230

上海海事大学

本科毕业论文（设计）



论文题目： 社区老人健康管理系统的设
计与实现

姓 名： 陈权

学 院 信息工程学院

专 业： 网络工程

班 级： 网络 211

指导教师： 韩玉娟

完成时间： 2025 年 4 月

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或者其他机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了感谢。

作者签名：_____日期：_____

论文版权使用授权书

本论文作者完全了解学校有关保留、使用论文的规定，即：学校有权保留并向国家有关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。本人授权上海海事大学可以将本论文的全部内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本论文。

本论文属于（请在以下相应方框内打“√”）：

☐ 保密，在 _____ 年解密后适用本使用授权书。

☐ 不保密。

作者签名：_____导师签名：_____日期：_____

摘 要

随着老龄化社会的到来，老年人健康管理问题日益突出，特别是在社区层面，老年人面临着慢性疾病、突发健康问题及日常生活管理等方面的需求。本项目设计并实现了一个社区老年人健康管理系统，旨在通过软件技术有效跟踪老年人的健康状况。系统通过数据库技术存储和处理老年人健康数据，并利用图数据库提供个性化的健康建议。系统的前端基于 Thymeleaf 框架开发，后端采用 SpringBoot 架构实现，结合 MySQL 数据库进行数据管理。数据上传功能仅作为接口，用于实现信息传输和管理。此外，系统支持家属信息查询和绑定功能，提供便捷的用户界面，确保老年人、家属和管理员能够实时监控老年人的健康状态和获取相关健康建议。该系统不仅能够提高老年人健康问题的早期发现率，还能减轻医疗负担，并优化社区老年人健康管理的服务质量。

关键词：老年人健康；权限分配；数据分析；图数据库

Abstract

With the aging society, the issue of elderly health management has become increasingly prominent, especially at the community level. Elderly individuals face various needs related to chronic diseases, sudden health issues, and daily life management. This project designs and implements a community elderly health management system, aiming to effectively track the health status of elderly individuals through software technology. The system uses database technology to store and process elderly health data, and utilizes a graph database to provide personalized health recommendations. The front-end is developed using the Thymeleaf framework, while the back-end is implemented with the SpringBoot architecture, combined with a MySQL database for data management. The data upload feature serves only as an interface for information transmission and management. Additionally, the system supports family member information query and binding functions, providing a user-friendly interface to ensure that elderly individuals, their families, and administrators can monitor the health status of the elderly in real time and receive relevant health advice. This system not only improves the early detection of health issues in elderly individuals but also alleviates medical burdens and enhances the quality of community-based elderly health management services.

KeyWords: Elderly health; Access control; Data analysis; Graph database

目 录

1 绪 论	1
1.1 研究背景	1
1.2 研究意义	2
1.3 相关研究	2
1.4 研究内容	3
2 系统概述	5
2.1 系统的特点	5
2.1.1 前后端分离架构	5
2.1.2 多数据源整合	5
2.1.3 安全性和权限管理	5
2.1.4 易扩展与维护	5
2.2 硬件环境	5
2.3 软件环境	6
2.4 开发工具	6
3 系统分析	7
3.1 系统总体要求	7
3.2 系统需求分析	7
3.3 系统模块划分	8
4 系统设计	10
4.1 MySQL 数据库设计	10
4.1.1 设计需求分析	10
4.1.2 数据表结构字段设计	10
4.2 图数据库设计	14
4.2.1 设计需求分析	14
4.2.2 数据获取与查询	15
4.3 系统功能设计	15
4.3.1 系统功能结构	15
4.3.2 各功能模块介绍	15
5 系统实现	17

5.1 数据库数据导入	17
5.1.1 MySQL 数据导入	17
5.1.2 Neo4j 数据导入	17
5.2 系统功能模块实现	19
5.2.1 用户角色模块	19
5.2.2 人员模块	21
5.2.3 指标模块	22
5.3 前端界面实现	23
5.3.1 认证模块	23
5.3.2 管理员界面	24
5.3.3 年轻人界面	25
5.3.4 老年人界面	27
6 结束语	28
6.1 研究结论	28
6.2 研究不足及展望	28
致 谢	29
参考文献	30

1 绪 论

随着全球老龄化进程的加速，老年人口比例逐年增加，尤其是在中国，老龄化问题日益严重。根据国家统计数据，到 2023 年底，中国 60 岁及以上的老年人口已达到 2.6 亿^[1]，占总人口的比例不断上升。伴随着老龄化问题的加剧，老年人的健康管理也成为社会关注的热点。老年人群体通常面临慢性疾病管理、突发健康问题、生活质量等多方面的挑战^[2]，如何有效地进行老年人健康管理，已成为社区医疗服务亟待解决的难题。

目前，尽管国内已有一些关于老年人健康管理的研究和实践^[3]，但大多集中在较大的城市和专业健康机构，而许多较小城镇和农村地区的老年人健康管理服务仍显不足。此外，现有的健康管理系统在数据整合、个性化服务等方面存在一定的不足，缺乏高效的数据处理和智能分析功能。因此，设计一款基于现代信息技术的社区老年人健康管理系统，能够实时监测老年人的健康状况并提供个性化建议，显得尤为重要。

本研究的主要目标是设计并实现一个社区老年人健康管理系统，利用图数据库技术处理和存储老年人健康数据，通过个性化的健康建议帮助老年人提高生活质量，减少因突发健康问题带来的医疗负担。该系统将结合前后端技术，提供便捷的用户界面，使老年人、家属及管理员能够有效管理和监控老年人的健康状态。

本论文的结构安排如下：第二章介绍系统概述，第三章进行系统分析，第四章详细阐述系统设计，第五章为系统功能模块的具体实现，最后在第六章总结研究成果，展望未来的发展方向。

1.1 研究背景

随着全球老龄化进程的加剧，老年人群体的健康管理问题日益成为社会各界关注的重点。据国家数据统计，全国 60 岁及以上的老年人口正在不断增加，预计到 2050 年，全国老年人口将达到 4.4 亿^[1]，占全国总人口的比例将从 2020 年的 17.4% 上升至 34.6%。

老年人群体常常面临慢性疾病、突发健康问题、心理健康和日常生活管理等多方面的健康挑战，而现有的医疗资源和健康管理服务在一些地区尤其是基层社区和农村地区依然匮乏，无法有效满足老年人健康管理的需求。

随着信息技术的快速发展，尤其是大数据、云计算、人工智能等技术的应用，为解决老年人健康管理提供了新的解决方案。许多国家已经开始尝试通过现代信息技术手段来改善老年人的健康管理。例如，美国和日本通过可穿戴设备收集老年人的健康数据，

远程监控并进行个性化健康管理^[4]，取得了初步成效。然而，尽管技术手段不断发展，国内针对老年人健康管理的系统仍存在许多问题，如数据整合不完善、个性化服务缺乏、用户体验差等。

在中国，虽然政府出台了《健康中国 2030 规划纲要》并不断推动社区健康管理的发展^[5]，但许多基层社区仍面临着健康管理服务不足、健康数据监控困难等问题。为了更好地帮助社区管理老年人健康，亟需一种高效、便捷、个性化的健康管理系统。因此，本研究旨在设计并实现一款基于图数据库的社区老年人健康管理系统，通过对老年人健康数据的采集、存储与智能分析，实现健康风险预警和个性化健康建议，提升老年人的生活质量和健康水平。

1.2 研究意义

本系统通过图数据库技术对老年人的健康数据进行智能分析，能够为老年人提供个性化的健康建议和预警，帮助早期发现潜在的健康问题。通过健康数据的监测和智能分析，能够提高老年人健康问题的早期发现率，有效降低突发健康事件的发生率和医疗费用^[6]。

其次，本研究还有望优化社区健康管理服务，推动基层医疗服务的数字化转型。通过健康数据的整合与分析，社区能够更加精准地了解老年人的健康状况，并在此基础上提供定制化的健康干预措施，提升老年人群体的整体健康水平。

此外，随着家庭结构的变化，许多子女由于工作原因无法时刻陪伴在老年人身边。通过本系统，家属可以实时查看老年人的健康数据和健康建议，减少了老年人健康问题未被及时发现的风险。同时，也能有效缓解子女对远离家乡的老年人的担忧，加强家庭成员之间的互动与关怀^[7]。

最后，本研究设计的老年人健康管理系统，不仅具有较高的实际应用价值，能够解决当前老年人健康管理中的许多实际问题，还能为未来的健康管理系统提供借鉴和参考，推动健康中国战略的实施。

1.3 相关研究

近年来，随着全球老龄化趋势的加剧，老年人健康管理成为学术界和社会各界关注的重要课题。国内外已有大量研究围绕老年人健康管理的不同技术和方法展开，尤其是在智能健康监测、健康数据分析以及信息技术的应用方面，取得了一定的进展。

在老年人健康管理的技术应用方面，许多研究探索了物联网和可穿戴设备在老年人

健康监测中的应用。例如，基于智能手环等可穿戴设备的健康监测技术，提出了通过实时数据采集和远程监控实现老年人健康管理的可行性^[8]。此外，IBM^[9]的 Watson 健康平台也在老年人健康管理中取得了一定的应用，利用人工智能分析老年人的健康数据，提供个性化的健康建议和预警。

然而，尽管这些技术在提升健康管理效率方面取得了一定成效，但目前的大多数系统仍存在一些局限性。首先，许多系统主要依赖硬件设备来采集数据，且往往忽略了数据整合和深度分析。其次，现有的老年人健康管理系统通常缺乏个性化服务，无法根据每个老年人的具体健康状况提供针对性的健康指导和干预。因此，如何通过高效的数据存储和分析技术，提供个性化、智能化的健康管理服务，仍是亟待解决的问题。

其次，图数据库在健康管理领域的应用也逐渐得到关注^[10]。图数据库因其在处理复杂关系数据方面相比关系型数据库具有天然的优势，已经被广泛应用于医学数据管理和疾病预测等领域。例如，neo4j 图数据库已在许多医学研究中被应用，用于疾病模型的构建和健康数据的关联分析。然而，针对老年人健康管理中如何利用图数据库技术进行个性化健康建议的研究尚较为有限，本项目旨在填补这一研究空白，通过图数据库技术优化健康数据存储与分析，提升社区老年人健康管理的精准度和实用性。

1.4 研究内容

本研究的主要目标是设计并实现一款基于图数据库的社区老年人健康管理系统，旨在通过现代信息技术手段提升老年人群体的健康管理效率和服务质量。具体研究内容如下：

系统需求分析与功能设计：明确老年人健康管理系统应具备的基本功能和需求，具体包括健康数据采集、存储、展示、分析、个性化健康建议和家属信息管理等。根据分析结果，设计系统的功能模块和技术架构，并确保系统能够满足社区老年人、家属以及管理员的多方需求。

健康数据的采集与存储：系统将通过前端管理员界面进行健康数据的录入，并通过数据库技术将老年人的健康数据进行存储与管理。研究将重点解决数据的结构设计问题，确保数据的高效存储、查询和分析。为此，系统将结合 MySQL 数据库用于常规数据存储，同时采用图数据库技术来存储和分析老年人健康数据之间的复杂关系，提供个性化健康建议。

图数据库在健康管理中的应用：本研究的创新之一是引入图数据库技术，用于处理老年人健康数据之间的关系。例如，设计一套根据指标异常后催生的一些疾病，然后借助大模型训练得到一系列的建议数据，如食品和生活习惯建议，以此来构建健康图谱，

利用图数据库进行健康数据的分析和病症预测，为老年人提供个性化的健康管理方案。

前端和后端的开发与集成：研究将结合 Thymeleaf 框架开发系统前端界面，确保用户能够方便地查看老年人的健康数据和建议。同时，后端采用 SpringBoot 架构实现，结合 MyBatis-Plus 进行数据持久化处理，实现系统的功能逻辑。前后端数据交互通过 RESTful 接口进行，确保数据传输的稳定与安全。

家属信息管理与关系绑定功能：研究将实现家属信息管理模块，允许老年人家属与老年人之间进行关系绑定，通过系统查看老年人的健康数据和相关健康建议。这一功能旨在缓解子女因工作等原因无法时刻陪伴老年人的困境，加强家庭成员之间的联系与关怀。

系统测试与优化：在完成系统设计与实现后，本研究将对系统进行全面的功能测试、性能测试和用户体验评估。测试将重点评估系统在健康数据处理、图数据库查询与分析、用户界面交互等方面的表现，并对系统进行必要的优化，确保系统能够稳定运行并满足用户需求。

通过上述研究内容的实现，期望能够构建一个适用于社区的、便捷高效的老年人健康管理系统，推动老年人健康管理服务的数字化与智能化，为提高老年人群体的生活质量和健康水平提供有力支持。

2 系统概述

2.1 系统的特点

2.1.1 前后端分离架构

本系统采用前后端分离的架构设计，前端使用 Thymeleaf 框架进行页面渲染，后端基于 Spring Boot 框架构建，确保了系统的高效性和可扩展性。前后端通过 RESTful API 进行数据交互，确保了系统的灵活性和易于维护。

2.1.2 多数据源整合

系统不仅支持传统关系型数据库 MySQL 进行结构化数据存储，还结合图数据库（Neo4j）处理健康数据的复杂关系，充分发挥多数据源的优势，提升了数据整合和分析能力。另外在性能方面，系统集成了 Redis 进行缓存管理，提升了数据读取和处理的速度，特别是在高并发情况下，保证了系统的响应时间和稳定性。

2.1.3 安全性和权限管理

系统采用 Spring Security 实现用户权限管理，确保不同用户（管理员、老年人、年轻人家属）具有不同的操作权限并会根据权限来展示不同界面。系统同时采纳了 JWT（JSON Web Token）认证，进一步确保用户身份的安全性和数据访问的合规性和高效性。另外，对于用户密码的存储，后端采用了哈希算法的加密，避免了密码明文泄露，进一步的提高了用户的隐私性。

2.1.4 易扩展与维护

前后端均采用了 Spring Boot 框架的模块化设计，使得系统易于扩展和维护。新的功能模块或数据源的添加不会对现有系统造成影响，有助于后期的系统升级和功能扩展。另外前后端代码的编写采用了当下最流行的前后端分离管理，便于后续前端技术栈的迭代。另外，前后端间采用了 HTTP 请求进行沟通，简单高效易维护。

2.2 硬件环境

1. 处理器：Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

2. 机带：RAM 16.0 GB (15.8 GB 可用)
3. 系统类型：64 位操作系统, 基于 x64 的处理器

2.3 软件环境

1. 操作系统：Windows10 专业版 22H2, 版本号 19045.5487
2. JVM 环境(项目代码): java version "17.0.6" 2023-01-17 LTS Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190) Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)
3. MySQL: 版本 8.0.33, 数据库引擎 InnoDB, 字符集 utf8mb4
4. Neo4j: 版本 4.4.41, 字符集 utf8mb4。JVM 环境: java version "11.0.20" 2023-07-18 LTS Java(TM) SE Runtime Environment 18.9 (build 11.0.20+9-LTS-256) Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.20+9-LTS-256, mixed mode)
5. Redis: 版本 3.2.100, 字符集 utf8mb4

2.4 开发工具

1. 前端、后端：IntelliJ IDEA 2024.2.4
2. MySQL: DataGrip 2023.2.3
3. Neo4j: Google Chrome 版本 134.0.6998.89（正式版本）（64 位）
4. Redis: Another Redis Desktop Manager 1.5.8
5. 代码管理：Git-2.40.0-rc2-64-bit, 仓库 Github

3 系统分析

3.1 系统总体要求

本系统旨在构建一个社区老年人健康管理系统，通过前后端协同工作，实现老年人健康数据的管理、分析与展示，提升社区健康管理的效率。系统的总体要求如下：

1. 功能性要求

系统应具备完整的健康管理功能，包括老年人健康数据的录入、存储、查询、分析以及个性化健康建议。系统还需要支持家属绑定功能，使子女能够远程查看老年人健康状况。管理员可通过后台管理健康数据、用户信息以及角色分配管理。

2. 性能要求

系统应具备高并发处理能力，能够支持多用户同时访问，查询响应时间应控制在 1 秒以内。数据库需优化查询性能，保证在大规模数据存储的情况下仍能高效执行查询。缓存机制（如 Redis）应用于加速数据访问，提高系统整体性能。

3. 安全性要求

系统需提供严格的权限管理，确保不同角色（老年人、家属、管理员）只能访问各自授权的数据。用户认证应使用 JWT（JSON Web Token）进行身份验证，并支持加密数据存储，防止敏感信息泄露。

4. 兼容性要求

系统需兼容主流 Web 浏览器（如 Chrome、Firefox、Edge），确保跨平台访问的稳定性。同时，后端采用 RESTful API，方便第三方系统对接，便于数据共享和扩展。

3.2 系统需求分析

本系统旨在构建一个高效、便捷的社区老年人健康管理平台，整合健康数据存储、分析与展示功能，为老年人、家属及社区管理员提供健康数据的管理与监测能力。

首先根据用户系统进行需求分析，本系统主要涉及三类用户，每类用户的需求如下：

1. 老年人：能够查看个人健康数据，接收健康建议，了解自身健康状况，管理年轻人家属的绑定信息。

2. 年轻人家属：可通过系统远程查询自己绑定完毕的老年人健康数据，与老年人绑定关系，接收健康预警信息。

3. 管理员：负责系统用户管理、健康数据管理、权限分配，以及对健康数据进行整体分析和维护。

其次，根据功能来进行需求分析，分别为以下几种：

1. 用户管理：通过注册和登录系统，根据用户上传的身份证信息进行校验，通过后根据身份证来得到年龄并映射对应的角色信息。
2. 健康数据管理：由管理员来管理对应的老年人指标数据
3. 角色用户管理：通过管理员来完成角色的管理，并可实时查看用户数据，方便实时的禁用非法用户。
4. 数据可视化：后端在将接口代码和数据库搭建完毕后，需要搭建一个可视化的前端界面来封装接口并将应有的需求界面渲染展示。

最后再根据非功能来进行需求分析，分别为以下几种：

1. 性能需求：后端对一些需要常用的数据存储在内存中，方便后续数据的频繁获取，例如用户的令牌和指标内容，考虑将数据转储到 Redis 数据库中，并考虑到内存失效后造成的缓存击穿和雪崩问题。
2. 安全需求：对于用户的密码存储，考虑到 Spring Security 框架中的哈希值计算，将密码以密文存储于后端数据库，另外对于前后端交互的用户令牌，考虑摒弃传统的 cookies 传输，进而采用 JWT 令牌作为传输媒介，不仅保证了安全性还能方便数据的存储。

系统整体的需求分析的架构图如下图所示。

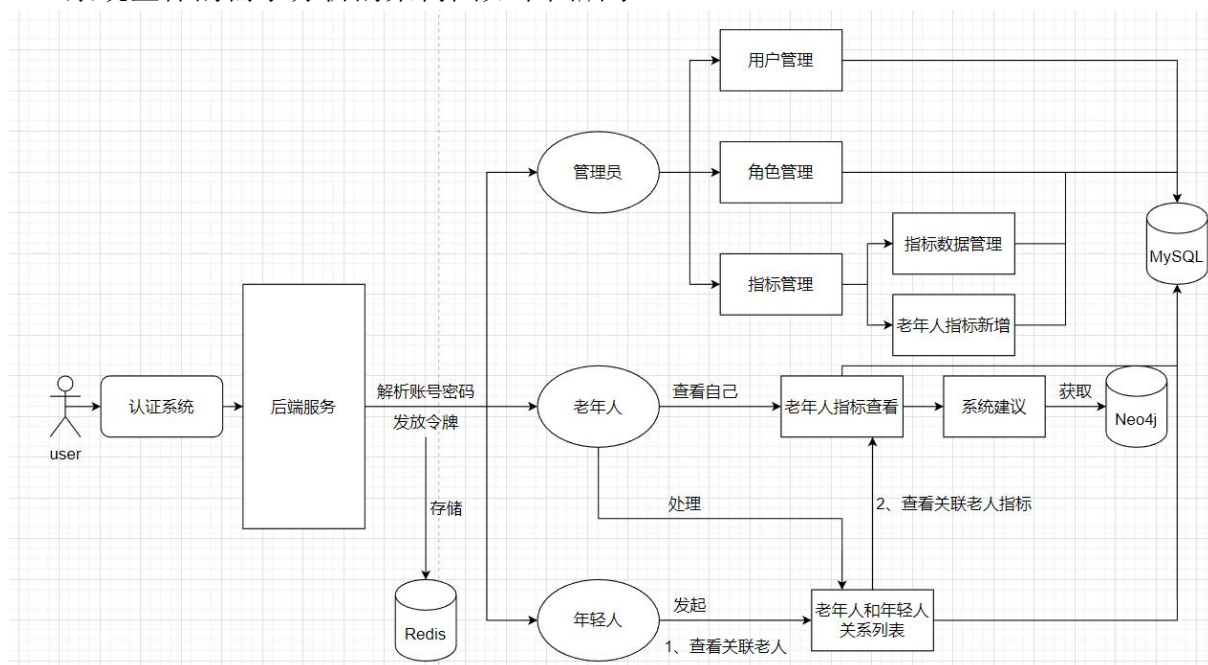


图 3.1 需求分析架构图

3.3 系统模块划分

系统分为前端和后端两大模块，前端主要以页面驱动开发，而后端主以功能驱动。

根据前端的界面设计，主要根据角色来分配界面，所以考虑根据角色来作为划分标准讲系统模块划分开，即主要分为认证模块、管理员模块、老年人模块和年轻人模块。认证模块主要负责根据用户角色将用户分流到对应的角色界面；管理员模块即对应着上图 3.1 的三个功能用户管理、角色管理和指标管理所展开；老年人模块则主要负责处理年轻人发起的关联申请和查阅自己的指标信息和对应的建议；年轻人模块则负责根据老年人的姓名和身份证号精准查阅指定的老年人来发起关联申请，在拥有关联老年人后可以查阅他的指标信息和建议。

根据图 3.1 下的功能将项目代码主要分为指标模块、用户角色模块和人员模块。指标模块考虑主要负责编写老年人的指标数据管理和指标信息管理；用户角色主要负责认证、用户自我管理和管理员负责的用户角色数据管理；人员模块主要由老年人和年轻人二者相辅相成，主要是二者对老年人指标数据的查阅和二者的关系绑定。

4 系统设计

4.1 MySQL 数据库设计

4.1.1 设计需求分析

本系统的 MySQL 数据库设计遵循模块化原则，以提升数据存储的规范性和查询效率。具体表格设计依据系统的主要功能模块进行划分，包括指标模块、用户角色模块和人员模块。

由于用户模块与人员模块存在较高的耦合性，为避免数据冗余，提高系统的可维护性和扩展性，设计时将这两个模块合并至同一张表中，统一存储用户信息。此外，为了表示老年人与家属之间的关联关系，系统设计了一张关联表，用于存储二者的绑定信息。这种设计不仅保证了数据的完整性，还提高了数据查询的灵活性。

在指标模块方面，考虑到后续指标数据的可扩展性及查询性能，指标模块的表设计采用分表存储的方式。具体而言，系统将健康指标数据拆分为两张表：指标表与老年人指标表，二者通过指标表的主键进行关联。这种设计方式不仅方便后续新增或维护健康指标，还有效降低了查询时的性能损耗，同时减少了表与表之间的耦合度，使系统具备更好的可扩展性。

4.1.2 数据表结构字段设计

表格主要分为五张表格，分别为用户表，角色表，用户关系表，指标表和老年人指标表。具体的 ER 图如下所示：

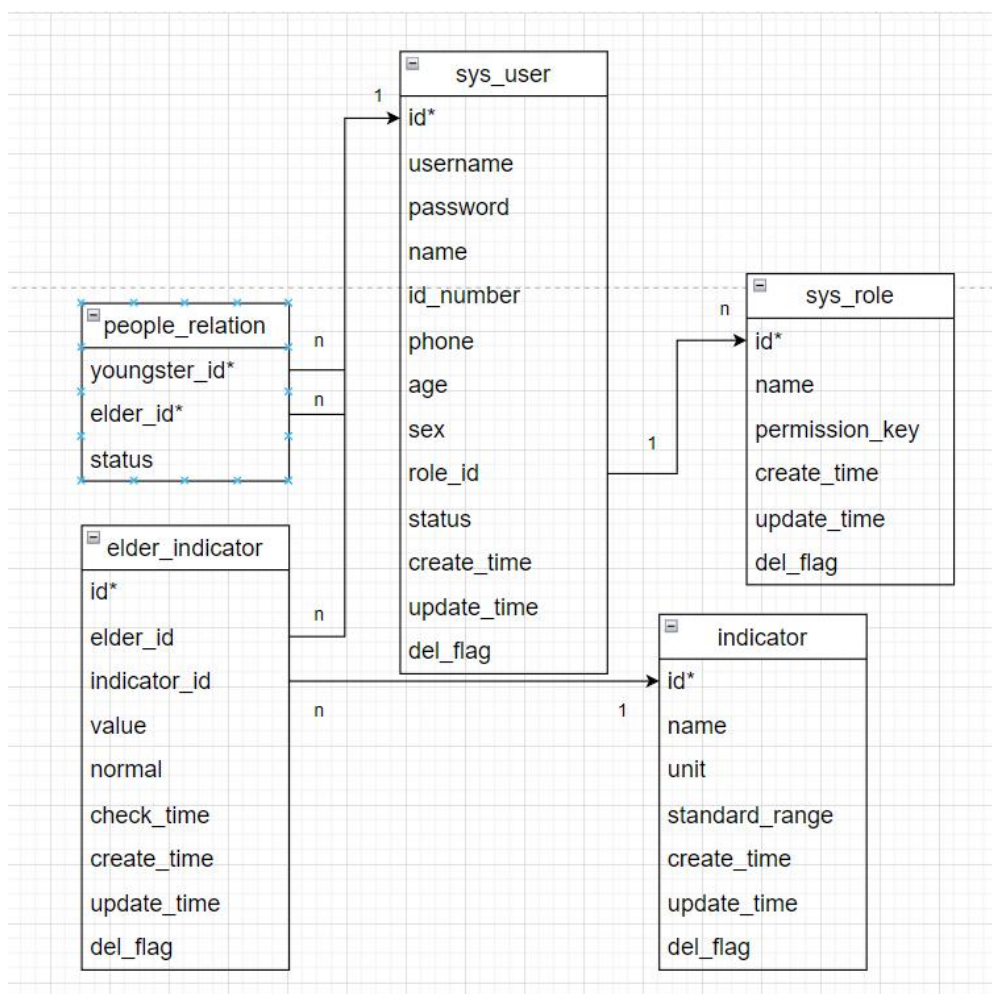


图 4.1 数据库 ER 图

1) 系统用户表如表 4.1 所示。

表 4.1 系统用户表

字段名	中文名	数据类型	Java 数据类型	备注
id*	ID	big int	java.lang.Long	主键 ID
username	用户名	varchar(64)	java.lang.String	用户名，唯一标识，可用于登录
password	密码	varchar(255)	java.lang.String	用户密码，以密文格式存储
name	姓名	varchar(64)	java.lang.String	用户真实姓名
id_number	身份证	varchar(18)	java.lang.String	用户身份证号
phone	手机号	varchar(11)	java.lang.String	用户手机号
age	年龄	int	java.lang.Integer	用户年龄
sex	性别	int	java.lang.Integer	用户性别，0 男 1 女

role_id	角色 ID	big int	java.lang.Long	角色 ID 外键
status	状态	int	java.lang.Integer	用户状态, 0 正常, 1
create_time	创建时间	datetime	java.util.Date	创建时间
update_time	更新时间	datetime	java.util.Date	更新时间
del_flag	删除标记	tinyint	java.lang.Integer	删除标记, 1 删除

2) 系统角色表如表 4.2 所示。

表 4.2 系统角色表

字段名	中文名	数据类型	Java 数据类型	备注
id*	ID	big int	java.lang.Long	主键 ID
permission_key	权限键	varchar(64)	java.lang.String	角色的权限键, 用于后端接口权限管理
name	角色名	varchar(64)	java.lang.String	角色名
create_time	创建时间	datetime	java.util.Date	创建时间
update_time	更新时间	datetime	java.util.Date	更新时间
del_flag	删除标记	tinyint	java.lang.Integer	删除标记, 1 删除

3) 人员关系表如表 4.3 所示。

表 4.3 人员关系表

字段名	中文名	数据类型	Java 数据类型	备注
youngster_id*	年轻人用户 ID	big int	java.lang.Long	年轻人的用户表外键, 和 elder_id 共同组成主键
elder_id*	老年人用户 ID	big int	java.lang.Long	老年人的用户表外键, 和 youngster_id 共同组成主键
status	关系状态	tinyint	java.lang.Integer	关系状态, 0 表示关系确认, 1 表示关系未确认, 默认是 1

4) 指标表如表 4.4 所示。

表 4.4 指标表

字段名	中文名	数据类型	Java 数据类型	备注
id*	ID	big int	java.lang.Long	主键 ID
unit	指标单位	varchar(64)	java.lang.String	指标单位
name	指标名	varchar(64)	java.lang.String	指标的医学名
standard_range	指标范围	varchar(64)	java.lang.String	指标的标准范围，若有男女之分将以;分隔
create_time	创建时间	datetime	java.util.Date	创建时间
update_time	更新时间	datetime	java.util.Date	更新时间
del_flag	删除标记	tinyint	java.lang.Integer	删除标记，1 删除

5) 老年人指标表如表 4.5 所示。

表 4.5 老年人指标表

字段名	中文名	数据类型	Java 数据类型	备注
id*	ID	big int	java.lang.Long	主键 ID
elder_id	老年人用户 ID	big int	java.lang.Long	老年人的用户表外键
indicator_id	指标 ID	big int	java.lang.Long	指标外键
value	指标值	decimal(10,5)	java.lang.Double	老年人的指标值
normal	指标正常	int	java.lang.Integer	指标是否正常，0 正常，1 异常
check_time	检查时间	date	java.util.Date	体检的时间
create_time	创建时间	datetime	java.util.Date	创建时间
update_time	更新时间	datetime	java.util.Date	更新时间
del_flag	删除标记	tinyint	java.lang.Integer	删除标记，1 删除

4.2 图数据库设计

4.2.1 设计需求分析

图数据库主要的责任是根据老年人的指标异常时，得到一些建议需求。因为指标异常时往往会引发一些并发症，所以将考虑将症状作为图数据库的入口，通过症状衍生到一些生活建议和饮食建议，最后将数据做好统一的规整返回给后端。其中，又考虑到食物具有一个食物类别的属性，同时，在一个类别中的食物也往往具有相同的功效，所以在饮食建议中插入一个食物类别的节点负责收纳众多食物统一管理，最后的图状结构如下图所示：

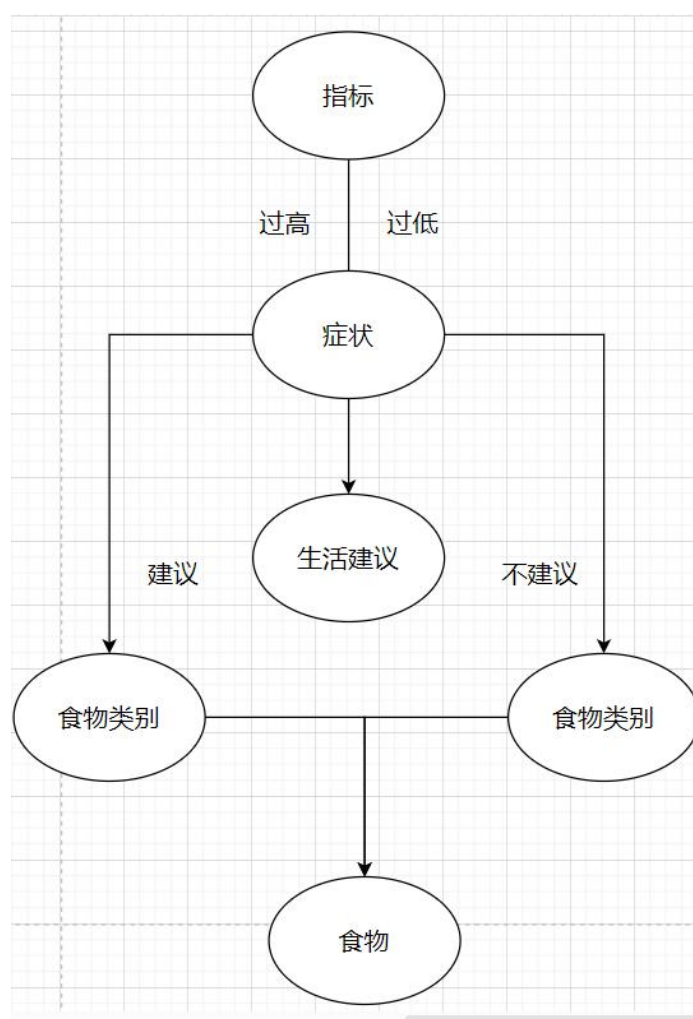


图 4.2 图结构

4.2.2 数据获取与查询

基于历史数据和大模型训练，系统能够识别症状与老年人健康状况之间的潜在关系。大模型通过对大量健康数据的学习，能够预测在特定症状下，哪些生活和饮食建议对老年人更为适宜。模型通过学习已有的健康数据、症状和建议之间的关系，得出针对健康异常的精准建议。系统根据症状与饮食建议之间的关系，生成具体的饮食推荐。每个饮食建议与特定食物类别相关联，如“低脂”、“高钠”等。在图数据库中，食物类别作为一个独立节点，与具体食物，如苹果、胡萝卜等关联。根据大模型的训练结果，系统会根据症状选择对应的食物类别，并从每个类别中提取最合适的食物进行推荐。最终，通过图数据库的查询和关系推理，系统将从症状节点衍生出相关的生活建议、饮食建议和食物类别节点，形成完整的数据结构。这个图状结构将统一整理后，返回给后端，以便生成可视化界面，供老年人和家属查看。

4.3 系统功能设计

4.3.1 系统功能结构

根据系统需求，项目需要具备这些功能。老年人健康数据和异常指标的建议的查阅，年轻人也该具备同样的功能，二者互相该具备有一个双向的关系绑定。管理员则独立于前面的功能，主要掌控用户、角色、指标等数据的管理权限。对于图数据库的维护暂时交由开发人员来维护。

4.3.2 各功能模块介绍

通过用户访问前端系统来讲模块依次展开介绍。

首先，用户会先访问系统的认证模块。该模块主要由注册和登录两个子模块组成，为控制注册的用户数量，考虑通过身份证和姓名作为认证的第一关，接着在后端逻辑中会通过用户身份证来完成信息的采集，如性别和年龄，通过年龄即可判断用户为老年人还是年轻人。登录模块中主要通过用户唯一标识和密码来完成登录，其中唯一标识考虑可以输入手机号、身份证和账户。登录成功后服务端会将用户的信息和令牌返回前端。在用户登录成功后，前端会根据用户的身份信息来跳转对应角色的界面。大致流程图如图 4.3 所示。

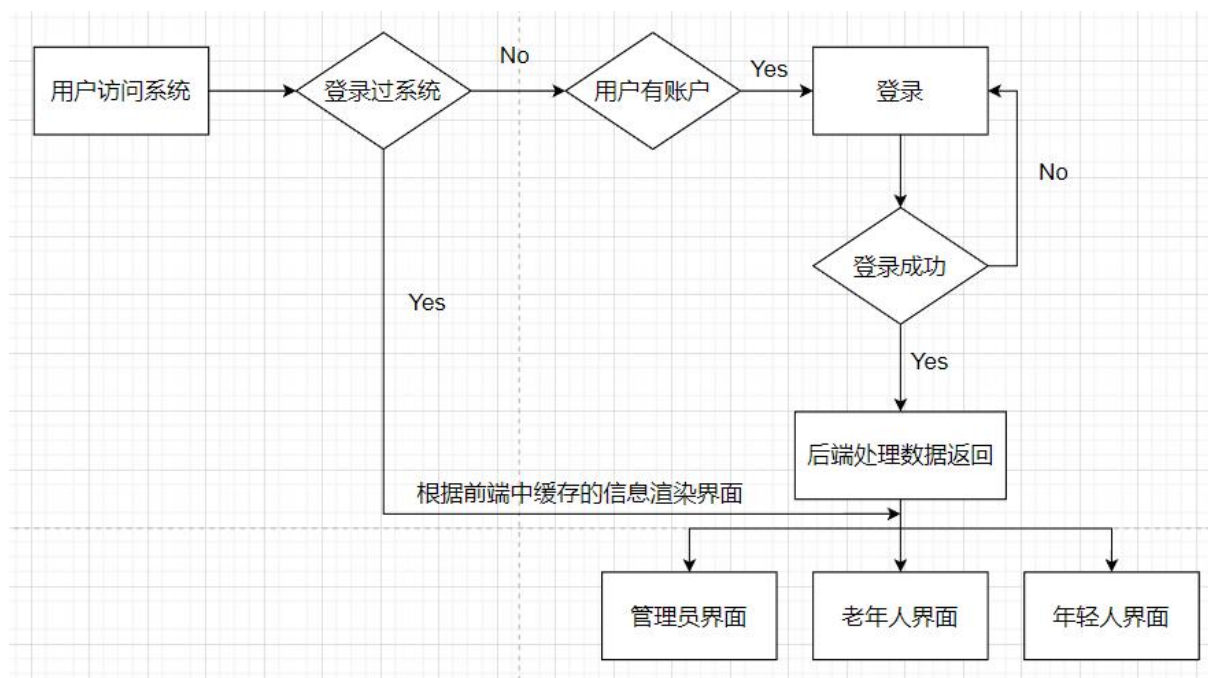


图 4.3 认证流程图

老年人界面主要由自己的健康指标数据和建议的查阅和年轻人管理两个菜单模块，指标模块负责展示该用户的过往的指标信息，年轻人管理模块主要是管理年轻人发送的绑定请求来完成绑定和解绑；年轻人界面是老年人管理菜单模块，年轻人通过身份证和姓名来完成老年人的唯一搜索，搜索完毕后可以申请绑定关系，在老年人同意了关系绑定后可以供该年轻人查阅老年人的身体健康指标；管理员界面拥有用户、角色和指标管理权限，对于用户模块而言，主要负责根据条件检索用户并可以禁用和删除用户，还可以新增老年人的指标数据，角色模块则是可以管理角色，指标模块主要是管理一些指标数据。界面的系统图如图 4.4 所示。

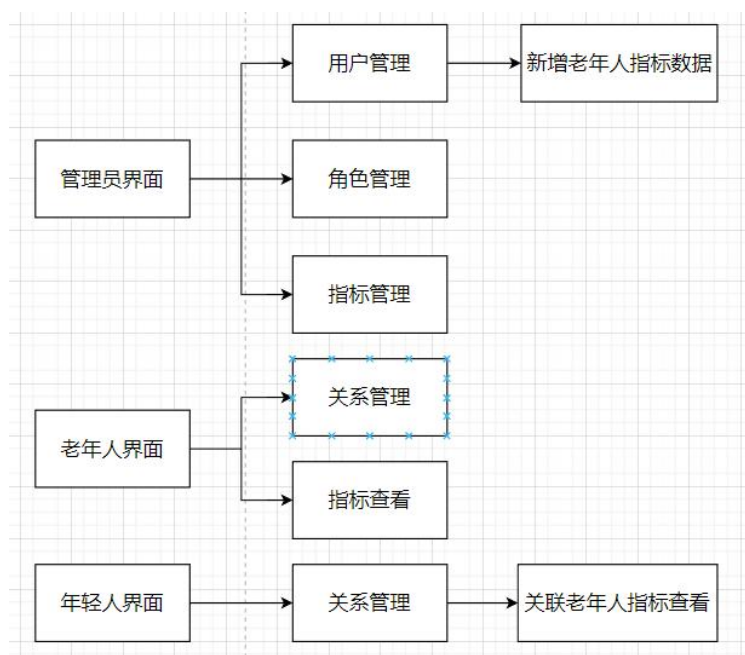


图 4.4 系统模块图

5 系统实现

5.1 数据库数据导入

5.1.1 MySQL 数据导入

通过系统设计完毕的数据库字段，编写 sql 语句，完成表格创建。创建表格完毕后，可默认创建一些数据，如角色和管理员用户，接着根据网上现有权威材料，可以获取到老年人健康指标的数据，本系统以一些血液指标为例展开研究。最后再查阅网上一些公开的相关数据，完成老年人指标数据的创建。

5.1.2 Neo4j 数据导入

根据大模型训练出的数据，编撰成一个格式化的文本，接着根据代码解析文本，得到一个完整的数据集，最后再利用框架中集成的接口连接 Neo4j 数据库，完成数据的导入。

首先需要根据大模型训练得到一个格式化的文本。

接着编写代码，将文本读取并将数据写入数据库中。系统使用的框架 SpringBoot 中具备有集成 Neo4j 的一个依赖库，可以通过配置的方式来连接数据库并以面向对象的模式将数据导入到数据库中。配置数据库的地址和连接的协议等。

最后编写对应的代码完成数据的导入。根据 Spring 官方文档中建议我们使用 dsl 方式来完成 neo4j 的 cql 语句编写。

接着就可以编写对应的代码来将数据写入了。先编写一个 Neo4j 的实体类，并利用框架的注解来完成后续的 cql 语句数据的填充。如下图 5.1 所示。

```

@Node(FOOD_NODE)
public class FoodNode extends BaseNode {
    public final static String NAME = "name";

    @Id
    @Property(NAME)
    private String name;

    @Override
    public String getIdKey() { return NAME; }

    @Override
    public String getIdValue() { return name; }
}

```

图 5.1 neo4j 的实体类

接着编写 dsl 的代码来自动生成 cql 语句，完成数据的写入。样例代码如下图 5.2 所示。

```

@Transactional(rollbackFor = Exception.class, transactionManager = "neo4jTransactionManager")
public int addNodes(Set<? extends BaseNode> nodes) {
    return neo4jTemplate.saveAll(nodes).size();
}

@Transactional(rollbackFor = Exception.class, transactionManager = "neo4jTransactionManager")
public int addRelationsById(BaseNode parentNode, Set<? extends BaseNode> childrenNodes, String relationName) {
    Node parent = node(parentNode.getNodeName()).named(newSymbolicName: "parent")
        .withProperties(Map.of(parentNode.getIdKey(), parentNode.getIdValue()));
    List<String> cql = childrenNodes.stream().map(childrenNode -> {
        Node child = node(childrenNode.getNodeName()).named(newSymbolicName: "child")
            .withProperties(Map.of(childrenNode.getIdKey(), childrenNode.getIdValue()));
        return match(parent).match(child) OngoingReadingWithoutWhere
            .create(parent.relationshipTo(child, relationName)) OngoingUpdate
            .build().getCypher();
    }).collect(Collectors.toList());
    return executeCql(cql);
}

```

图 5.2 dsl 编写 cql 语句

5.2 系统功能模块实现

在本系统的后端实现中，采用了 Spring Boot 框架进行开发，支持快速开发和自动化配置，十分方便后续项目迭代。为了确保系统的安全性，本系统采用了 Spring Security 进行用户身份验证和权限控制。Spring Security 提供了一套强大的认证和授权机制，可以有效保护系统免受未经授权访问，确保数据的安全性。通过集成 JWT，系统为每个用户生成有效的认证令牌，并通过令牌进行身份验证，确保不同角色的用户访问权限得到严格控制。数据持久层使用了 MyBatisPlus 作为 ORM 框架，不仅继承了 MyBatis 通过简单的 XML 配置的方式提供了灵活的 SQL 查询和执行能力，还支持复杂查询语句和动态 SQL 生成，极大减少了对 sql ddl 语句的编写。

5.2.1 用户角色模块

该模块主要负责用户认证和管理员对普通用户和角色的管理。用户模块主要有注册、登录、退出登录、更新个人信息和管理员的用户管理接口。角色模块主要是负责对角色表的管理，和给普通用户更改角色。后端通过引入 Spring MVC 来便捷地采用 RESTful 的模式编写接口代码，样例代码如下图 5.3 所示。

```
@RestController
public class UserController extends BaseController {
    @Autowired
    private SysUserService userService;

    @PostMapping("/register")
    public ResponseEntity<RegisterVO> register(@RequestBody RegisterDTO register) {
        return ok(userService.register(register));
    }

    @PostMapping("/login")
    public ResponseEntity<UserInfoVO> login(@RequestBody LoginDTO login) {
        return ok(userService.login(login));
    }

    @DeleteMapping("/logout")
    public ResponseEntity<Void> logout() {
        userService.logout();
        return ok();
    }
}
```

图 5.3 controller 层代码

其中，在用户注册时，后端会检测用户传递的信息是否符合后端逻辑条件，如身份

证、手机和用户名等的格式。在完成格式校验后会接着查询数据库唯一标识是否已经存在，唯一标识会用在登录界面，主要是手机号、身份证和用户名三个组成，因为登录时可选其一作为账户，所以在注册业务中必须对其做唯一性校验。最后，完成了所有的校验后会根据用户的身份证号来获取用户的年龄和性别，最后保存数据库，完成用户创建。具体注册逻辑代码如下图 5.4 所示。

```
@VerifyRequestBody
public @NonNull RegisterVO register(RegisterDTO register) {
    // 校验格式
    AssertUtils.isTrue(StringUtils.isMatch(register.getPhone(), PHONE_FORMAT), AppHttpCode.PHONE_FORMAT_ERROR);
    AssertUtils.isTrue(StringUtils.isMatch(register.getUsername(), USERNAME_FORMAT), AppHttpCode.USERNAME_FORMAT_ERROR);
    AssertUtils.isTrue(StringUtils.isMatch(register.getPassword(), PASSWORD_FORMAT), AppHttpCode.PASSWORD_FORMAT_ERROR);
    AssertUtils.isTrue(StringUtils.isMatch(register.getIdNumber(), ID_NUMBER_FORMAT), AppHttpCode.ID_NUMBER_FORMAT_ERROR);
    AssertUtils.isEquals(register.getPassword(), register.getRepeatPassword(), AppHttpCode.PASSWORD_NOT_EQUALS_ERROR);
    // todo 校验名字和身份证，需要企业用户才能调用接口

    // 校验用户名、手机、身份证号是否已存在
    LambdaQueryWrapper<SysUser> userWrapper = new LambdaQueryWrapper<>();
    userWrapper.eq(SysUser::getUsername, register.getUsername());
    AssertUtils.isTrue(!count(userWrapper) == 0, AppHttpCode.USERNAME_EXISTS);
    userWrapper.clear();
    userWrapper.eq(SysUser::getPhone, register.getPhone());
    AssertUtils.isTrue(!count(userWrapper) == 0, AppHttpCode.PHONE_EXISTS);
    userWrapper.clear();
    userWrapper.eq(SysUser::getIdNumber, register.getIdNumber());
    AssertUtils.isTrue(!count(userWrapper) == 0, AppHttpCode.ID_NUMBER_EXISTS);

    // 保存用户
    int bornYear = Integer.parseInt(register.getIdNumber().substring(6, 10));
    int nowYear = new GregorianCalendar().get(Calendar.YEAR);
    int age = nowYear - bornYear;
    SysUser user = BeanCopyUtils.copyBean(register, SysUser.class);
    user.setPassword(passwordEncoder.encode(register.getPassword()));
    user.setAge(age);
    user.setRoleId(age >= ELDER_AGE_BOUNDARY ? ELDER_ROLE_ID : YOUNGSTER_ROLE_ID);
    user.setSex(register.getIdNumber().substring(16, 17).matches(regex: "[13579]") ? USER_SEX_MAN : USER_SEX_WOMAN);
    save(user);
    return BeanCopyUtils.copyBean(user, RegisterVO.class);
}
```

图 5.4 注册逻辑代码

登录业务主要依托于 Spring Security 框架来完成，通过用户传递的账户密码，搜索数据库成功会创建用户令牌，在 Redis 中缓存用户数据，最后返回用户信息，等待前端下一步请求。具体的逻辑代码如下图 5.5 所示。

```

@VerifyRequestBody
public @NonNull UserInfoVO login(LoginDTO login) {
    UsernamePasswordAuthenticationToken authentication = new
        UsernamePasswordAuthenticationToken(login.getIdentification(), login.getPassword());
    Authentication auth = manager.authenticate(authentication);
    AssertUtils.nonNull(auth, AppHttpCode.UNAUTHORIZED_ERROR);
    LoginUser loginUser = (LoginUser) auth.getPrincipal();
    String userId = String.valueOf(loginUser.getUser().getId());

    Map<String, Object> payload = new HashMap<>();
    String token = JwtUtils.createJWT(userId, payload);
    redisCache.setWithExpire(key: RedisConstants.REDIS_TOKEN_KEY + userId,
        loginUser,
        RedisConstants.REDIS_TOKEN_EXPIRE);
    UserInfoVO userInfo = BeanCopyUtils.copyBean(loginUser.getUser(), UserInfoVO.class);
    userInfo.setToken(token);
    userInfo.setRoleName(loginUser.getRole().getName());
    return userInfo;
}

```

图 5.5 登录逻辑代码

5.2.2 人员模块

该模块主要负责老年人和年轻人的数据交互，以老年人视角来看，人员模块有关联的用户查询、新增和删除接口，年轻人同样拥有类似的接口，同时考虑到老年人对于智能化的接受努力低于年轻人，所以将关系的发起者交到年轻人身上，因此年轻人还有一个根据身份证和名字来查询的接口。如下图 5.6 的添加关系的逻辑代码和图 5.7 的根据用户的姓名和身份证号获取到指定老年人。

```

public @NonNull UserInfoVO addPersonRelation(@NonNull Long relationUserId) {
    SysUser relationUser = this.getById(relationUserId); // 根据id获取用户
    AssertUtils.nonNull(relationUser, AppHttpCode.USER_NOT_FOUND_ERROR);
    LoginUser loginUser = SecurityUtils.getLoginUser();
    SysRole userRole = loginUser.getRole(); // 获取发起请求的用户的角色信息
    SysUser user = loginUser.getUser();
    if (YOUNGSTER_ROLE_ID.equals(userRole.getId())) {
        // 如果是年轻人，则只能添加老人
        AssertUtils.isTrue(ELDER_ROLE_ID.equals(relationUser.getRoleId()), AppHttpCode.RELATION_USER_ROLE_ERROR);
        int addCount = this.baseMapper.addRelation(user.getId(), relationUserId);
        AssertUtils.isTrue(ret: addCount > 0, AppHttpCode.RELATION_EXISTS_ERROR);
    } else if (ELDER_ROLE_ID.equals(userRole.getId())) {
        // 如果是老人，则只能添加年轻人
        AssertUtils.isTrue(YOUNGSTER_ROLE_ID.equals(relationUser.getRoleId()), AppHttpCode.RELATION_USER_ROLE_ERROR);
        int addCount = this.baseMapper.affirmRelation(relationUserId, user.getId());
        AssertUtils.isTrue(ret: addCount > 0, AppHttpCode.RELATION_NOT_FOUND_ERROR);
    } else {
        throw new IllegalArgumentException("用户的角色错误");
    }
    return BeanCopyUtils.copyBean(relationUser, UserInfoVO.class);
}

```

图 5.6 添加关系逻辑代码


```

public @NonNull PersonVO getExactPerson(String idNumber, String name) {
    AssertUtils.isTrue(StringUtils.isMatch(idNumber, ID_NUMBER_FORMAT), AppHttpCode.ID_NUMBER_FORMAT_ERROR);
    LambdaQueryWrapper<SysUser> wrapper = new LambdaQueryWrapper<>();
    wrapper.eq(SysUser::getIdNumber, idNumber).eq(SysUser::getName, name);
    SysUser user = this.getOne(wrapper);
    AssertUtils.nonNull(user, AppHttpCode.USER_NOT_FOUND_ERROR);
    AssertUtils.isTrue(ELDER_ROLE_ID.equals(user.getRoleId()), AppHttpCode.USER_NOT_FOUND_ERROR);
    return BeanCopyUtils.copyBean(user, PersonVO.class);
}

```

图 5.7 查询老年人逻辑代码

5.2.3 指标模块

指标模块是系统的核心模块，主要分为指标数据模块和老年人指标模块。

指标数据模块由管理员和开发人员来管理，通过大模型技术等完成数据的填充，是项目的核心数据模块，为老年人指标模块提供数据源，主要由管理指标接口组成。

老年人指标模块负责连通指标数据模块和人员模块，通过获取指标数据模块中已训练完毕的数据，传递给指定的用户，完成数据的展示。接口由老年人指标获取、年轻人指标获取和管理员上传老年人指标组成。

老年人数据指标上传时，系统接收来自老年人的健康指标数据，根据数据库查询标准范围，如果检测到健康指标超出正常范围（过高或过低），系统即认为健康状态异常，会通过 MySQL 数据库的对应字段进行存储。如下图 5.8 所示。

```

@VerifyRequestBody
@Transactional(value = "transactionManager", rollbackFor = Exception.class)
public @NonNull Integer addElderIndicators(PatchElderIndicatorDTO indicators) {
    SysUser user = sysUserMapper.selectById(indicators.getElderId());
    AssertUtils.nonNull(user, AppHttpCode.USER_NOT_FOUND_ERROR);
    List<ElderIndicator> elderIndicators = new ArrayList<>();
    for (ElderIndicatorDTO elderIndicator : indicators.getElderIndicators()) {
        AssertUtils.nonNull(elderIndicator.getIndicatorId(), AppHttpCode.REQUEST_DATA_FIELD_IS_NULL);
        AssertUtils.nonNull(elderIndicator.getValue(), AppHttpCode.REQUEST_DATA_FIELD_IS_NULL);
        // 检测是否超出正常范围，记录进数据库
        Integer normalStatus = indicatorService.getNormalStatus(user.getSex(), elderIndicator);
        ElderIndicator indicator = BeanCopyUtils.copyBean(elderIndicator, ElderIndicator.class);
        indicator.setElderId(indicators.getElderId());
        indicator.setCheckTime(indicators.getCheckTime());
        indicator.setNormal(normalStatus);
        elderIndicators.add(indicator);
    }
    return this.saveBatch(elderIndicators) ? elderIndicators.size() : 0;
}

```

图 5.8 新增老年人指标逻辑代码

老年人指标获取的业务流程有两步，第一步系统会先返回简单的指标信息给用户，并带上了检测时间和指标的异常标记，如果用户对指标表示有所疑虑，可以点击查看详情。点击后进入第二步，系统会根据检测时间统一返回对应指标，并会根据异常信息查询知识图谱完成建议的填充，最后封装完毕返回前端渲染。

5.3 前端界面实现

在本系统的前端实现中，采用了 Thymeleaf 作为模板引擎，并结合 Spring Boot 框架进行开发，方便后续扩展。其允许开发者在 HTML 中直接嵌入动态数据，通过简洁的语法控制页面的显示效果。它支持表单处理、URL 链接、条件判断等功能，使得前端编写更加灵活多变。另外，Thymeleaf 代码的编写十分适配后端代码，能让后端开发程序员快速上手。在前后端的接口对接上，不仅可以通过 JavaScript 代码来实现，还可以通过 Java 的控制层代码来完成控制，进一步的提高数据的私密性。另外在前后端的交互上还可以使用 Java 的 OkHttp 来完成 http 请求。

对于用户认证模块，前端主要负责利用表单来完成数据的传输，并利用 CSS 样式来完成界面的美化。一些数据交互场景，还可以利用动态的界面生成技术来自动填充数据表单和浮动窗口，方便用户处理数据，如角色管理和老年人指标数据的添加时均采用了浮动窗口，完美规避了界面跳转的弊端，实现便捷的数据交互。

界面设计主要根据用户类型来完成了分隔，管理员、年轻人和老年人各有各的界面，每一个菜单目录即是一个代码文件，完美避免了在交互时出现界面跳转错误的场景。

5.3.1 认证模块

认证模块在前端界面分为登录和注册两个界面，如下图 5.9 和图 5.10 所示。

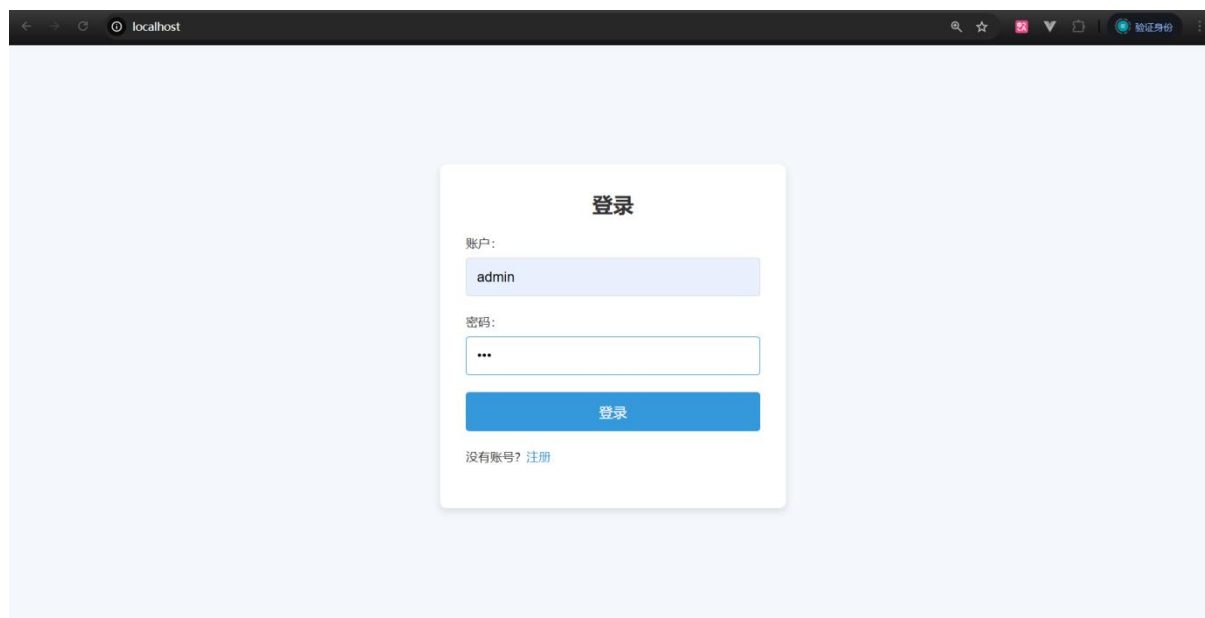
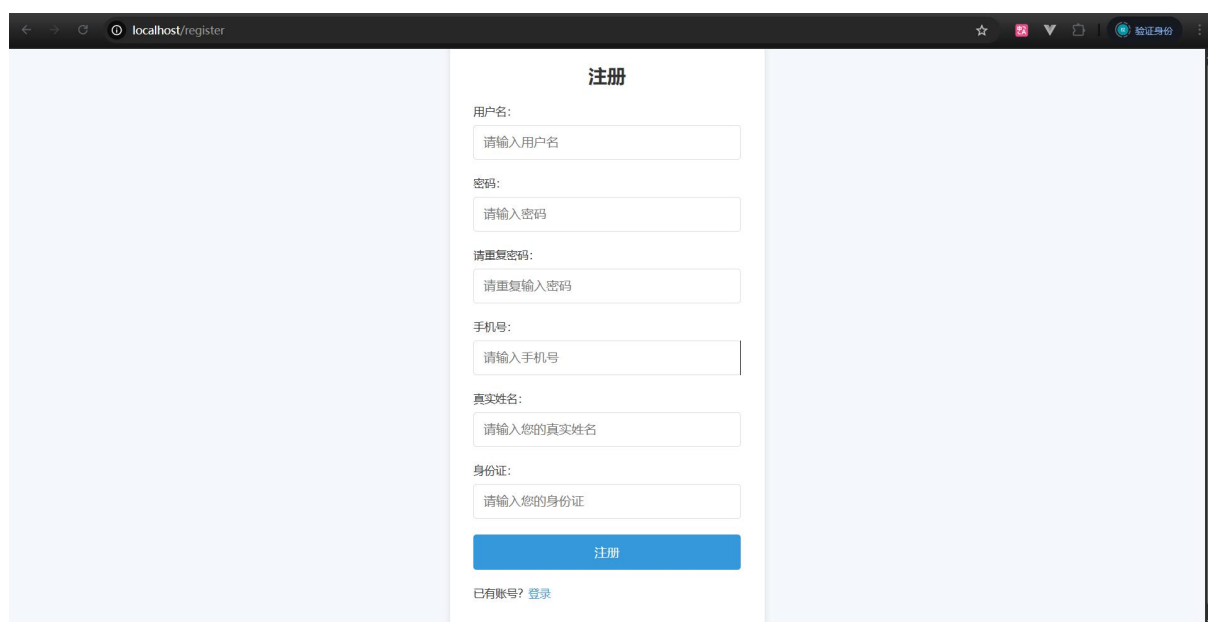


图 5.9 登录界面



The registration interface (注册) is displayed in a browser window at localhost/register. It features a central form with the following fields and labels:

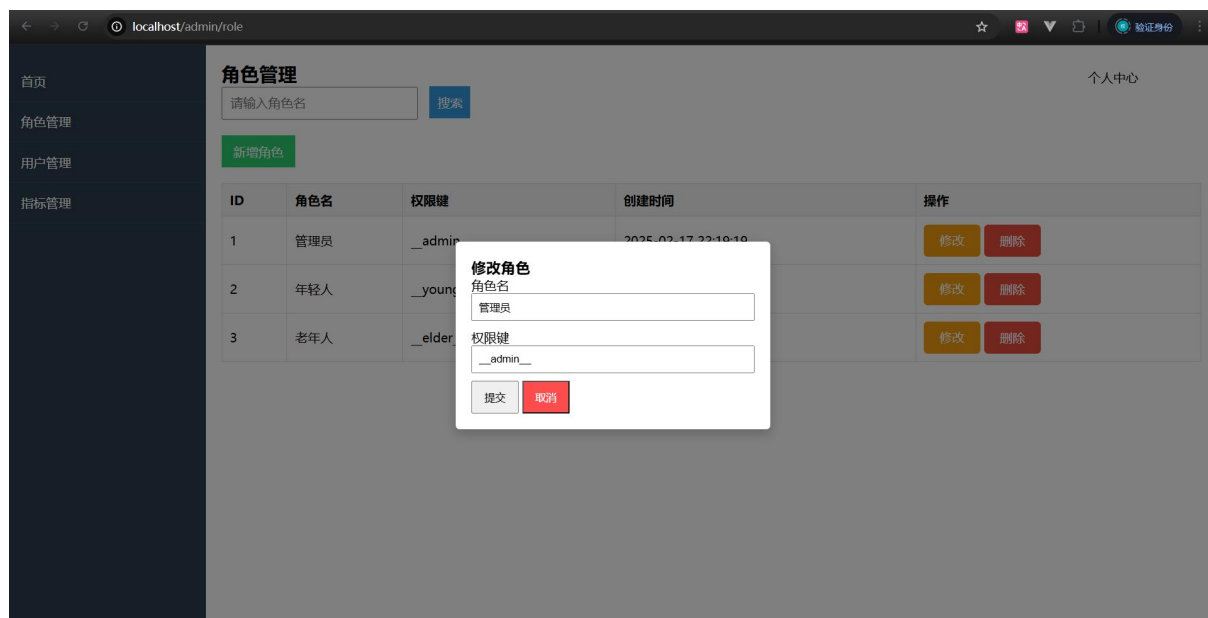
- 用户名: 请输入用户名
- 密码: 请输入密码
- 请重复密码: 请重复输入密码
- 手机号: 请输入手机号
- 真实姓名: 请输入您的真实姓名
- 身份证: 请输入您的身份证

At the bottom of the form is a blue button labeled "注册". Below the button is a link: "已有账号? 登录".

图 5.10 注册界面

5.3.2 管理员界面

管理员界面主要由三个部分组成，分别是角色、用户管理和新增指标。角色管理界面需要有能够满足角色数据的增删改查功能，搜索功能支持角色名模糊搜索，还需要支持表单浮窗来给用户填充数据。具体界面如下图 5.11 所示。



The role management interface (角色管理) is shown in a browser window at localhost/admin/role. It includes a sidebar with navigation links: 首页, 角色管理, 用户管理, and 指标管理. The main content area has a search bar and a "搜索" button. Below the search bar is a green button labeled "新增角色".

The main table displays the following data:

ID	角色名	权限键	创建时间	操作
1	管理员	_admin	2025-02-17 22:16:19	修改 删除
2	年轻人	_young		修改 删除
3	老年人	_elder		修改 删除

A modal window titled "修改角色" (Edit Role) is open, showing the following fields:

- 角色名: 管理员
- 权限键: _admin_

At the bottom of the modal are two buttons: "提交" (Submit) and "取消" (Cancel).

图 5.17 角色管理界面

用户管理同样需要满足 crud，对于用户的搜索需要支持对用户名、真实姓名的模糊搜索和手机号、角色的精准匹配，此外需要对老年人和年轻人进行区分，老年人允许点击后添加指标数据，具体的界面如下图 5.12 所示。

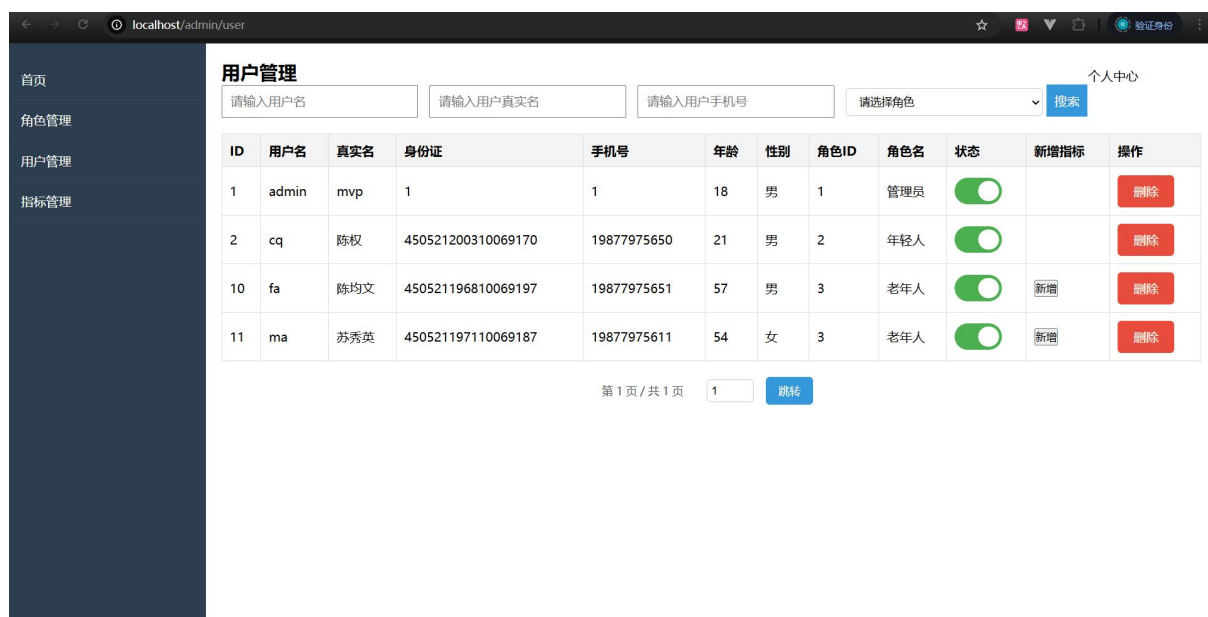


图 5.12 用户管理界面

在管理员点击了新增指标后，会搜索 MySQL 数据库获取到所有正常的指标，返回给管理员，后续需要管理员输入检查时间和指标的数据，最后完成数据的提交。如下图 5.13 所示。将来若是打通第三方的数据源，例如医院，可以借助管理员的身份，完成数据的自动化收集。

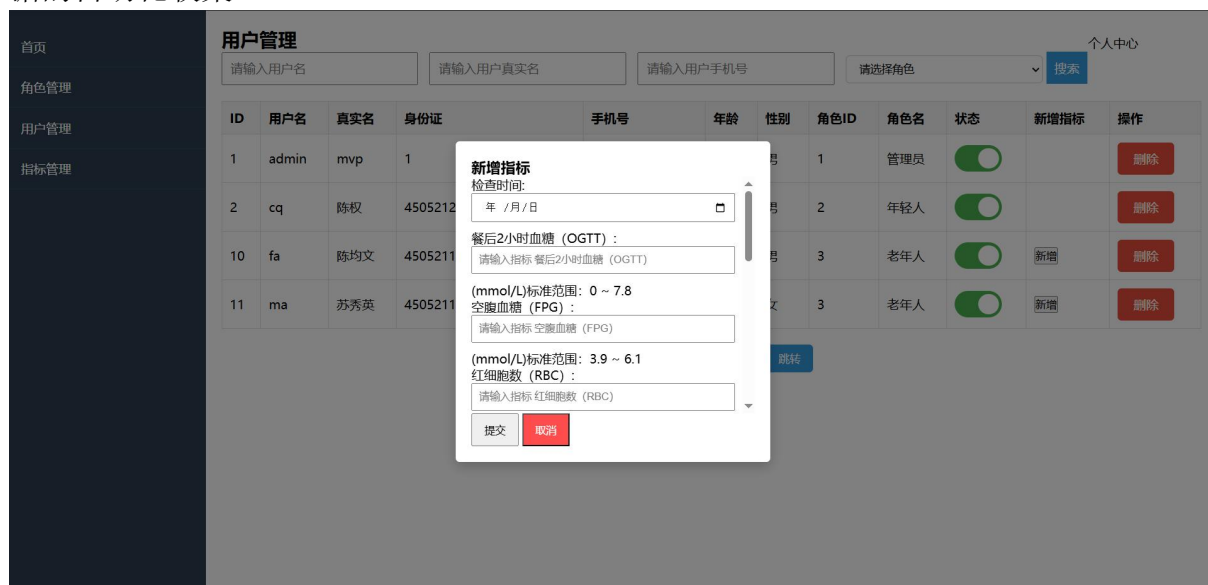


图 5.13 新增老年人指标界面

5.3.3 年轻人界面

年轻人界面主要通过和老年人进行关联，然后查询关联老年人的指标数据，所以界面只需要一个长辈管理即可，通过查询长辈来查询老年人的指标数据。另外还需要一个可以新增和删除长辈的一个功能。具体的界面如下图 5.14 所示。

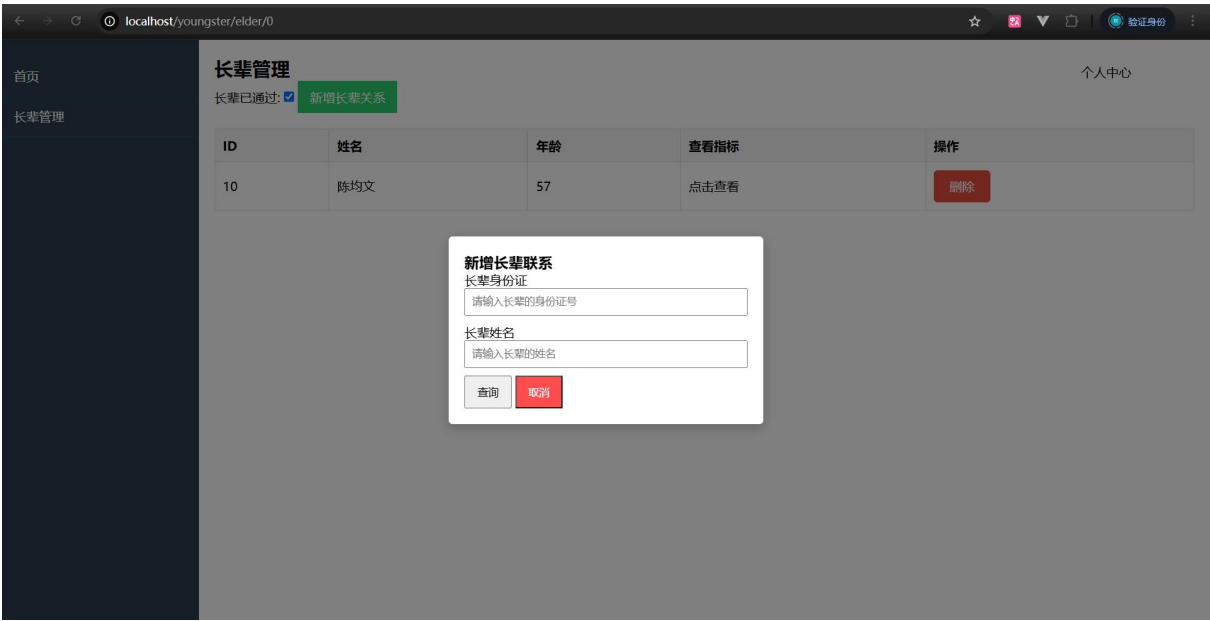


图 5.14 年轻人长辈管理界面

另外，年轻人可以通过点击查看来访问老年人的指标数据，由于同一个时间中可能会有多个指标，例如在一个时间内可能会检测有红细胞数和血红蛋白两个指标，如果后端直接将所有的指标都整合返回，会降低服务器处理性能，并且可能对于一些过早的检测时间可能并不关系，所以为了提高性能，考虑先查阅一个大致的数据，接着可以继续查看指标详情数据。具体的界面如下图 5.15 所示。



图 5.15 年轻人查阅关联老年人指标界面

接着可以点击异常中的查看建议，可以得到后端返回的系统建议，包含有指标异常可能会诱发的症状，生活建议和食物建议，还有大模型给到的一些建议。具体界面如下图 5.16 所示。

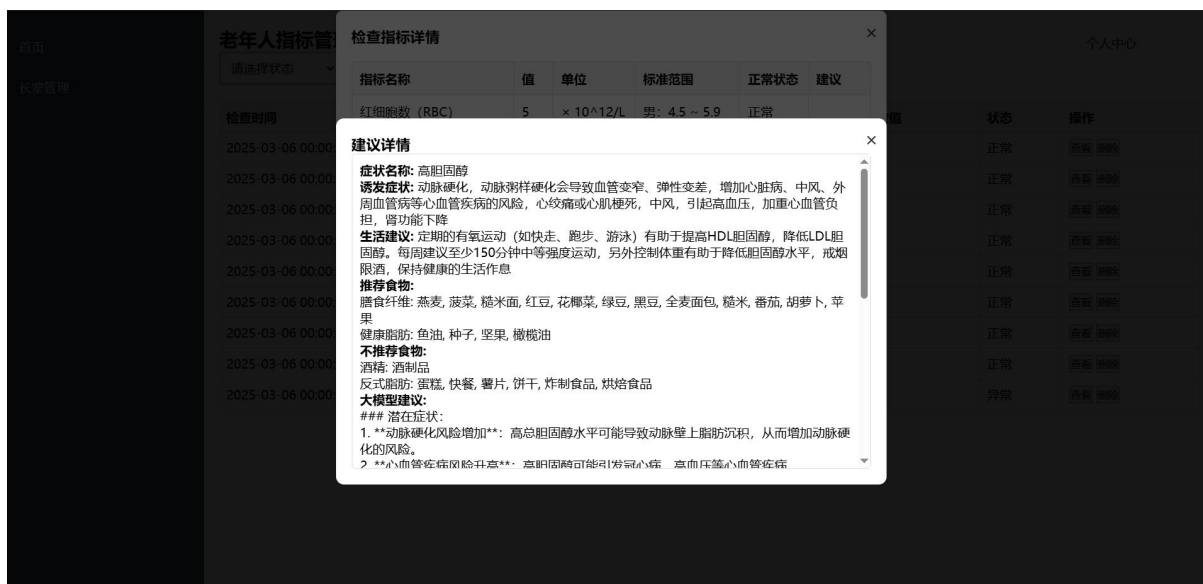


图 5.16 建议查询界面

5.3.4 老年人界面

老年人界面主要分为子女管理和指标的查看两大模块, 子女管理负责管理年轻人发起的关联请求, 指标的查看和年轻人的界面逻辑是一致的。老年人的界面如下图所示。

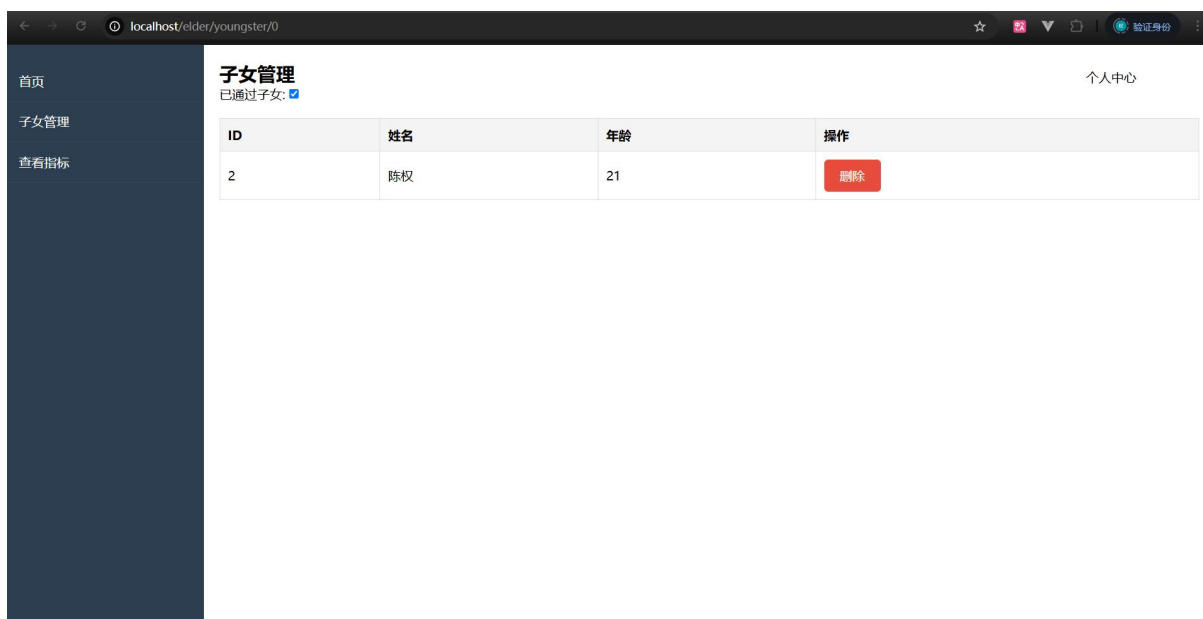


图 5.17 老年人界面

6 结束语

6.1 研究结论

本研究设计并实现了一个基于图数据库的社区老年人健康管理系统，旨在通过实时健康数据的监测与分析，并结合了大模型训练和健康数据分析，为老年人提供精准的健康建议。大模型通过对历史数据的学习，能够预测在特定症状下，哪些健康建议更为合适，进一步提升了系统的智能化水平。通过多维度的数据分析，系统能够实时监控健康指标的异常情况并及时提供预警，帮助老年人及其家属做出早期干预，从而有效减少疾病的发生和健康风险。

6.2 研究不足及展望

尽管本研究已经初步实现了健康管理的智能化和个性化，但仍存在一些不足之处。首先，健康数据的采集和处理依赖于用户的主动输入，如何进一步提高数据采集的准确性和实时性仍是一个挑战。其次，尽管系统设计了多层次的建议生成机制，但在面对复杂的多症状组合时，如何提高推理的精度和效率仍需进一步研究。

未来的研究可以考虑引入更多的传感器和实时数据接口，提高数据采集的自动化和实时性。此外，随着机器学习和人工智能技术的发展，未来可以进一步增强系统的智能分析能力，提高建议生成的个性化和准确性。通过集成更多医疗大数据和基于患者历史健康记录的分析，系统可以为老年人提供更加全面和精准的健康管理服务。

致 谢

行文至此，笔触稍顿，四载求学时光恍如昨日。回首本科生涯，从懵懂求知到独立完成毕业设计，其间每一步成长皆离不开师长、同窗与亲友的倾力支持。在此，谨以最诚挚的谢意，致谢所有给予我指引与温暖之人。

首谢恩师，传道授业，润物无声。感谢我的导师韩玉娟教授。从选题构思到实验设计，从数据纠偏到论文成稿，您始终以渊博的学识与严谨的态度为我指明方向。尤记深夜收到您批注的论文稿，字句圈点皆见匠心；实验瓶颈时，您一句“数据求真比结果完美更重要”令我豁然开朗。您不仅是学术的引路人，更以躬身治学的风范教我恪守科研初心。

次谢同窗挚友，并肩同行，风雨共济。感谢姜添翼、涂凯南等同学，无数个调试代码的深夜、反复验证数据的周末，因你们的协作与鼓励而不再枯燥。特别感谢室友邴雨飞在我撰写论文期间分担琐事，以乐观豁达消解我的焦虑。同窗之谊，既似战友，亦如家人，此间情义必将长存心间。

再谢父母家人，舐犊情深，默默守候。求学二十余载，父母从未以生计艰辛扰我分毫。每遇困顿，母亲电话中的“慢慢来，家里都好”总能令我重拾勇气；父亲虽寡言，却总在我归家时备好一桌热饭。寸草之心，难报春晖，惟愿早日立业，反哺亲恩。

终致岁月，以梦为马，不负韶华。本科生涯将尽，而求知之路未止。这段旅程中，我曾为实验失败彻夜难眠，亦为微小突破欢欣雀跃，更在团队协作中领悟“独行快，众行远”的真谛。这些经历终将凝成铠甲，助我在未来学术或职场中从容前行。

纸短情长，辞浅恩深。唯怀感恩之心，脚踏实地，以今日所学回报社会。此致，敬礼！

参考文献

- [1] 杜鹏, 翟振武, 陈卫. 中国人口老龄化百年发展趋势[J]. 人口研究, 2005, 29(6): 92.
- [2] 张嵩浩, 王丽芹. 老年慢性病患者社区健康管理模式研究进展[J]. *Advances in Clinical Medicine*, 2024, 14: 775.
- [3] 葛振兴, 李晓光, 王慧. 老年数字化健康管理研究进展[J]. 生命科学, 2023, 35(8): 984-993.
- [4] Saxena V, Kandpal S D, Goel D, et al. Health status of elderly a community based study[J]. *Indian journal of community health*, 2012, 24(4): 269-274.
- [5] 曾钊, 刘娟. 中共中央, 国务院印发《“健康中国 2030”规划纲要》[J]. 中学政史地: 高中文综, 2016 (12): 9-11.
- [6] 李丹, 张美琴, 唐诗. 基于物联网的远程医疗在老年健康管理中的应用研究进展[J]. 医学信息学杂志, 2022, 43(9): 47-53.
- [7] 马宁. 城市空巢老人心理健康状况的影响因素分析[J]. 决策与信息, 2016 (14): 88-88.
- [8] 刘亚男, 丛杉. 可穿戴技术在人体健康监测中的应用进展[J]. 纺织学报, 2018, 39(10): 175-179.
- [9] Ziegler J F, Curtis H W, Muhlfeld H P, et al. IBM experiments in soft fails in computer electronics (1978–1994)[J]. *IBM journal of research and development*, 1996, 40(1): 3-18.
- [10] 吴江, 黄晓, 董克. 基于知识图谱的在线医疗研究综述[J]. 信息资源管理学报, 2016, 6(2): 4-12, 21.