

哈 尔 滨 工 业 大 学

本科毕业论文（设计）

说明：

规范中所引用的示例，只作为本科毕业论文（设计）书写格式的示范，并不代表本科毕业论文（设计）研究内容的示范。

本科毕业论文（设计）

基于 Springboot 面向对象设计的学校运动 场系统设计与实现

**"Design and Implementation of a School Sports Field
System Based on Object-Oriented Design with Spring
Boot" ↑**

郑民采

哈尔滨工业大学
2024 年 6 月

本科毕业论文（设计）

基于 Springboot 面向对象设计的学校运动
场系统设计与实现

本 科 生：郑民采

学 号：L170300502

指 导 教 师：骆功宁教授

专 业：计算机工程

学 院：计算机科学与技术

答 辩 日 期：2024 年 6 月

学 校：哈尔滨工业大学

摘 要

本文研究了基于 Spring Boot 和 Vue.js 的学校运动场管理系统的设计与实现。系统旨在促进学生参与体育活动，通过数字化管理运动场使用，以提高效率。为此，整合了最新的 Web 技术，提供了用户友好的界面，并在后端使用 Spring Boot 处理业务逻辑和数据管理。

系统的整体架构遵循客户端-服务器模型，前端使用 Vue.js 框架支持动态交互，后端则通过 Spring Boot 基础的 RESTful API 实现。数据库方面，采用 JPA 实现对象关系映射，使用 MySQL 数据库进行数据管理。系统的安全性通过 JWT 和 OAuth2 进行加强。

在实际实施阶段，系统的各项功能均得到了成功执行的验证。用户可以通过系统预约运动场，管理体育活动，学校管理员能轻松掌握并管理学校的运动场使用情况。

系统将多个功能模块化设计，并提供了每个模块的详细设计和实施方法。例如，用户信息管理功能允许修改和查询用户信息，运动场管理功能负责管理场地预约状态和调整比赛日程，比赛管理功能则负责创建和管理比赛结果。数据库设计包括用户、团队、运动场和比赛等信息，使用关系型数据库进行信息存储和管理。

总之，通过利用现代 Web 技术设计并实现学校运动场管理系统，不仅提高了学校体育活动管理的效率，还促进了学生的体育参与，为教育技术的进步提供了新的可能性，展示了在其他教育机构的广泛应用前景。

关键词: 运动场系统；面向对象；Spring Boot；

Abstract

This paper explores the design and implementation of a school sports field management system based on Spring Boot and Vue.js. The system aims to promote student participation in physical activities by digitally managing the use of sports fields to enhance efficiency. To achieve this, it integrates the latest web technologies, provides a user-friendly interface, and uses Spring Boot on the backend to handle business logic and data management.

The overall architecture of the system follows the client-server model, with the frontend using the Vue.js framework to support dynamic interaction, and the backend implementing RESTful APIs with Spring Boot. In terms of databases, it uses JPA for object-relational mapping and MySQL for data management. The system's security is enhanced through the use of JWT and OAuth2.

During the actual implementation phase, the functionality of the system was successfully validated. Users can reserve sports fields and manage physical activities through the system, while school administrators can easily monitor and manage the usage of sports facilities.

The system features a modular design for various functions, providing detailed design and implementation methods for each module. For instance, the user information management function allows for modification and querying of user data, the sports field management function is responsible for managing field reservation statuses and adjusting match schedules, and the match management function handles the creation and management of match results. The database design includes information on users, teams, sports fields, and matches, utilizing a relational database for information storage and management.

In summary, by utilizing modern web technologies to design and implement a school sports field management system, not only has the efficiency of managing school physical activities been improved, but student participation in sports has also been promoted, providing new possibilities for the advancement of educational technology and demonstrating broad application prospects in other educational institutions.

Keywords: Object Oriented, Playground System, Spring Boot

目 录

摘 要.....	I
ABSTRACT	II
目 录.....	III
第 1 章 绪 论.....	4
1.1 课题背景及研究的目的和意义	4
1.2 本文的主要研究内容及论文结构	5
第 2 章 学校运动场系统需求分析和关键技术.....	7
2.1 功能性需求分析	7
2.2 非功能性需求分析.....	11
2.3 关键技术	11
2.3.1 Nginx 技术	11
2.3.2 Vue 技术.....	12
2.3.3 JPA 技术.....	12
第 3 章 学校运动场系统设计	14
3.1 学校运动场系统整体设计	14
3.3.1 系统架构设计	14
3.3.1 系统功能结构设计	16
3.3.2 基于 JWT 令牌（Token）的无状态系统设计	17
3.3.3 第三方登录设计	19
3.3.4 系统错误处理设计	21
3.2 学校运动场系统详细设计	23
3.2.1 应用设计模式	23
3.2.2 用户管理模块	26
3.2.3 运动场管理模块	27
3.2.4 比赛管理模块	29
3.2.5 团队管理模块设计	31

3.2.6 请求管理模块设计	33
3.3 学校运动场系统数据库设计	34
3.3.1 用户和团队模型设计	34
3.3.2 场地模型设计	37
3.3.3 比赛模型设计	38
3.3.4 请求模型设计	41
3.3.5 数据库总体设计	43
第 4 章 学校运动场系统实现	45
4.1 用户信息管理功能实现	45
4.1.1 用户信息功能实现	45
4.1.2 用户比赛管理功能实现	46
4.2 运动场信息管理功能实现	47
4.2.1 运动场查看功能实现	47
4.2.2 运动场比赛查看功能实现	48
4.2.3 运动场添加功能实现	48
4.3 比赛管理功能实现	49
4.3.1 比赛创建功能实现	49
4.3.2 比赛结果功能实现	51
4.4 团队管理功能实现	52
4.4.1 团队创建功能实现	52
4.4.2 团队管理功能实现	53
4.4.3 团队查看功能实现	54
4.5 请求管理功能实现	54
4.5.1 请求发送功能实现	54
4.5.2 请求查看功能实现	56
第 5 章 学校运动场系统测试	57
5.1 系统性能测试	57
5.1.1 测试环境	57
5.1.2 性能测试	57
结论	60
参考文献	61

哈尔滨工业大学学位论文原创性声明和使用权限	63
致 谢	64

第 1 章 绪 论

1.1 课题背景及研究的目的和意义

在当前数字化和信息技术快速发展的时代背景下，大学生体育活动的促进与激励变得尤为重要。本研究旨在通过技术创新，特别是利用 Spring Boot^[14] 这一先进的开源开发框架，探索构建一个能够激发学生体育热情、促进其全面发展的数字平台。

随着国内体育文化的兴盛，发现有效的方法将学生通过网络技术聚集在一起，共同参与体育活动，不仅对学生的个人发展有益，也有利于增强团队协作能力和加强校园文化凝聚力。

本文提出通过科技手段，简化参与运动的流程，让学生易于接触并参与到运动中，同时，吸引更多学生的积极参与，为校园营造更加积极的体育氛围。本研究将深入讨论设计一个以学生福祉为核心的，以科技为支撑的体育活动平台。该平台不仅要满足学生的基本运动需求，还要通过创新技术手段，增加额外的价值，运动推荐和团队建立等服务。

研究表明，用户的参与意愿会受到平台的易用性、是否能为用户带来社交认可或奖励等因素的影响。因此，本研究在平台设计初期就考虑到这些因素，旨在创建一个符合学生习惯、简单易用并且富有社交特性的体育活动平台。通过整合社区功能和技术创新，解决学生在体育活动中遇到的问题，提供更优质的体验和便利，最终达到促进大学生体育活动的目的。在上文中，说明了构建一个以学生福祉为核心，以科技为支撑的体育活动平台的重要性和基本框架。

接下来，将深入探讨如何利用现代 Web 技术和高效的代码结构设计来优化这一平台，确保其不仅具有高度的用户友好性和可维护性，还能轻松适应未来的发展和变化。

1.2 本文的主要研究内容及论文结构

整体系统架构考虑了功能需求和性能要求，数据库和代码设计遵循规范化原则，并应用各种设计模式。功能模块包括用户管理、运动场管理、比赛管理、团队管理和请求管理，确保系统运行稳定、高效。

（1）整体架构设计

整体架构设计是研究和设计一个快速、易于维护且具有良好扩展性的系统架构。在设计过程中，需要考虑系统的功能需求、性能要求以及未来的扩展计划。

（2）数据库设计

合理的数据库表设计及遵循数据库规范化原则的数据库设计，尽量优化数据库。

（3）基于面向对象的代码设计

应用各种设计模式以设计更高效的代码结构对于开发过程中是非常重要的。这种方法可以提高代码的可用性，简化维护过程，并更有效地管理大型项目。例如，单例（singleton）模式，工厂方法（Factory method）模式，策略（Strategy）模式和观察者（Observer）模式等。

（4）功能模块设计

功能模块设计包括以下内容：信息安全令牌功能，第三方登录，用户管理包括用户信息的管理、修改、删除，以及加入和退出团队等功能，团队管理包括团队信息的创建、修改、删除，以及团队创新功能，运动场信息查询，比赛信息管理包括比赛的创建、提交、删除，以及即将开始的比赛、过去比赛和比赛结果的查看，请求管理包括对各种请求的处理和管理，以及排行榜功能。这些模块旨在确保系统的完整运作，为用户提供便捷、安全的服务。

（5）功能模块实现

用户管理功能:实现用户信息管理，包括用户资料的修改和查看，以及用户注册和注销功能。此外，还包括用户登录功能，确保用户可以安全地访问系统。最后，实现用户比赛管理功能，包括用户参与的比赛记录和结果查询。

运动场管理功能:实现运动场的基本管理，包括查看现有运动场的信息，以及添加新的运动场信息。此外，提供运动场比赛的查看功能，用户可以查看特定运动场的比赛信息和赛程安排。

比赛管理功能:实现比赛的创建和管理功能,包括创建新的比赛项目并设置相关参数,以及管理现有比赛的结果和成绩。此外,确保比赛结果的准确记录和及时更新。

团队管理功能:提供团队创建功能,允许用户创建新的团队并添加成员。此外,实现团队管理功能,包括团队成员的管理和权限分配,以及团队资料的修改和更新。最后,提供团队查看功能,允许用户查看其他团队的信息和成员列表。

请求管理功能:实现系统中的请求管理,包括处理用户提交的各种请求和反馈。确保对请求进行及时响应,并根据实际情况进行处理和调整。

(1) 绪论

本章首先概述了研究背景、目的及其重要性。总结了研究的主要内容和结构。

(2) 学校运动场系统需求分析和关键技术

本章分析体育场系统需要满足的功能性需求,并通过用例图来阐述系统功能需求。继续讨论系统需要满足的非功能性需求和相关性能指标,最后阐述了确保系统稳定运行的关键技术。

(3) 学校运动场系统设计

本章从系统的架构、设计策略、功能结构和数据库的整体设计。然后详细讨论每个模块的详细设计过程,并通过类图来展示设计结果。

(4) 学校运动场系统实现

本章主要介绍系统核心功能模块的界面图。介绍每个模块的使用方式,并描述它们的功能。

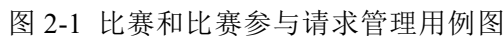
(5) 学校运动场系统测试

最后一章详细讨论了体育场系统的性能测试方法和结果。首先从测试目标和测试环境两个方面简述系统的性能测试计划,然后对实施的各个功能模块进行详尽的性能测试。

本章主要深入分析学校运动场系统的业务需求和功能性需求，非功能性需求，详细探讨并总结系统中所应用的关键技术。

本文分析和研究学生所需的学校运动场系统的运动比赛生成、参与、比赛日程管理、团队生成、参与等功能、个人信息、运动场信息、比赛信息和排名的功能。分析所有这些需求时,大致将平台分为4个功能模块。

比赛的主要参与者是比赛主办者和比赛参与者运动场管理者。图 2-1 为比赛和比赛参与请求管理的用例图。



比赛主持者可以参考特定运动场上的其他比赛的时间表,在该运动场存在的同一时间以外的其他时间创建比赛。管理者可以查看,删除创建的比赛,仅限于比赛结束的比赛能提交结果。参与者可以查询目前尚未开始的比赛,并可以发起比赛参加请求。比赛参与者可以看自己发起成功的比赛请求。

（2） 团队服务

团队服务主要参与者是团队队长和团队队员,团队管理者。图 2-2 为团队服务的用例图。

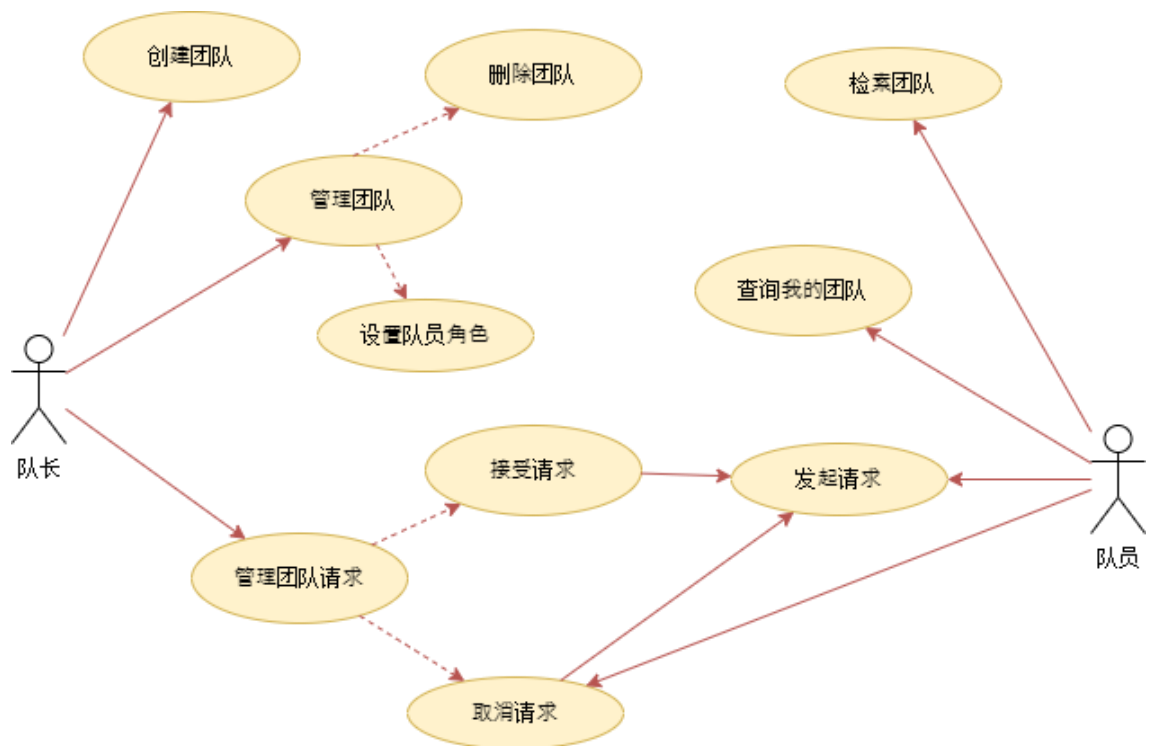


图 2-2 团队服务用例图

所有的用户都可以填写团队的信息创建团队,团队名称,头像,运动项目,团队介绍。团队成立后,创建团队的用于成为队长,可以删该团队,设置队员的角色。用户可以检索目前存在的团队或查看列表,并向想参与的团队发送团队参与请求。该团队的队长可以接受用户发送的请求,将可以接受该请求成为队员或拒绝。其请求接受并成为队员的用户可以在我的团队目录中看到自己所属的团队,此外队员可以退出自己所属的团队。

（3） 运动场服务

运动场服务主要参与者是用户和管理员。图 2-3 为运动场服务的用例图。

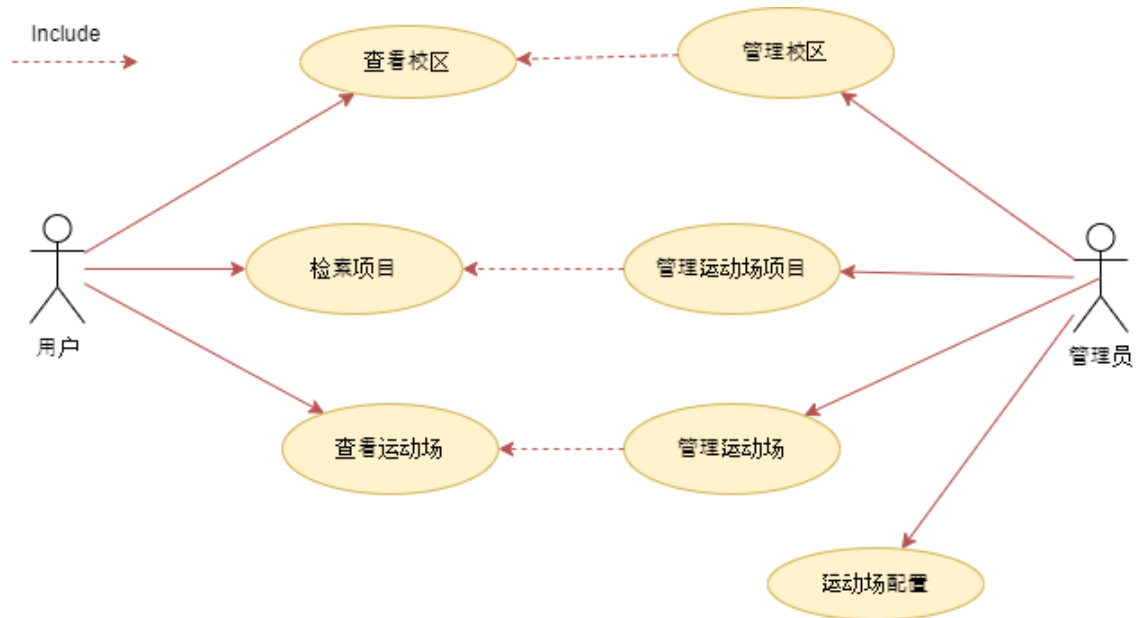


图 2-3 运动场服务用例图

用户可以选择想要的运动项目，可以选择整个校区或特定校区，并且可以查看该校区中管理员安排的运动场信息。管理员可以管理学校的校区，并且可以选择特定的项目来安排运动场。管理员负责管理运动场的状态。

（4） 用户服务

用户服务主要参与者是用户和管理员。图 2-4 为用户服务的用例图。

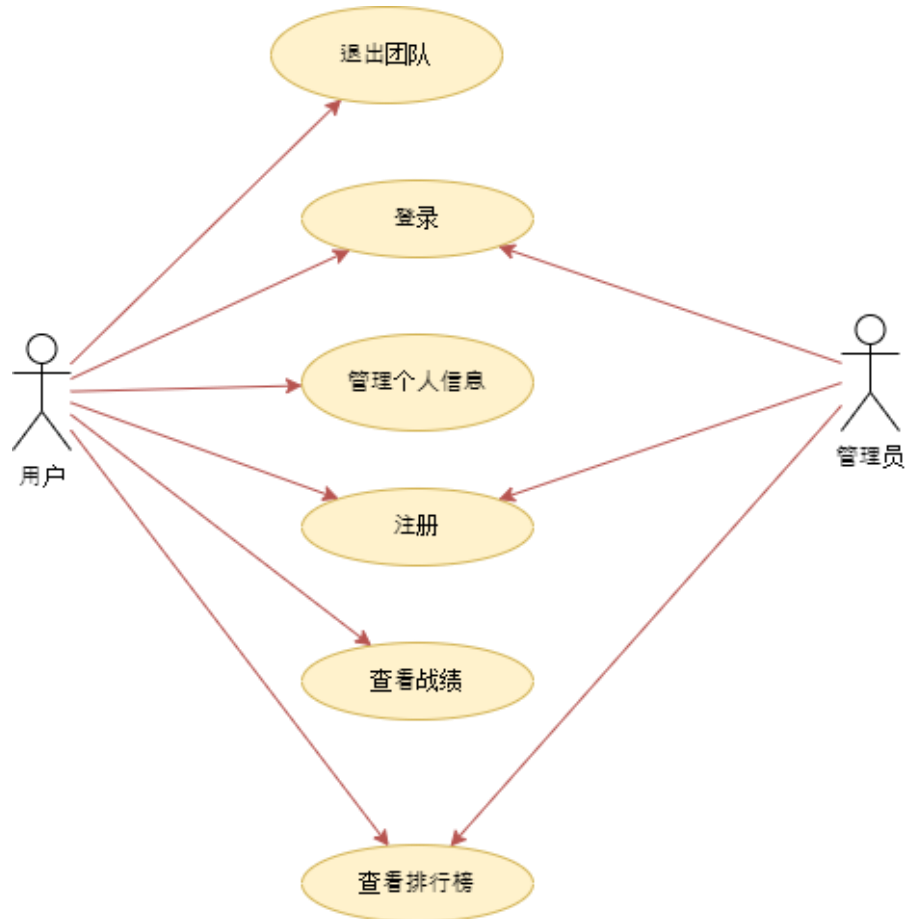


图 2-4 用户服务用例图

用户可以退出自己所在的团队，并且可以管理用户信息，如更改昵称、更改个人资料等。用户可以查看已经申请参加并成功被接受的比赛，这些比赛分为已经结束的比赛和即将开始的比赛。在即将开始的比赛中，用户可以退出该比赛。

2.2 非功能性需求分析

由于体育场系统被众多用户使用，系统的高效运行非常重要。同时，由于在系统操作过程中可能会添加许多要求，代码的维护性和可扩展性也是非常重要的方面。最后，为了让学生们能更容易使用，易用性也是重要的需求之一。

（1）可用性

系统应至少支持每秒查询处理量（QPS）1000 次以上，以保证在高强度使用环境下能够稳定提供服务。同时，系统应保持快速响应，对用户的请求在 2000 毫秒内完成响应。

（2）维护性和可扩展性

系统应设计为便于添加新功能或修改现有功能。通过应用模块化设计，可以独立更新或替换各个模块。此外，代码易于理解和修改。

（3）易用性

界面应直观且易于使用。通过采用以用户为中心的设计方法，应使学生能够迅速理解系统并顺畅使用。主要功能应明确显示，易于用户访问，常用功能应放在显眼位置，便于用户快速找到。

2.3 关键技术

Nginx 的应用是为了提高网站的性能和稳定性，Vue.js 和 JPA 的应用则是为了快速开发和灵活的数据管理。

2.3.1 Nginx 技术

Nginx^[10]（“Engine-X”）是一种开源软件，用于网页服务、反向代理、负载均衡和 HTTP 缓存。它由伊戈尔·赛索耶夫创建，旨在提高网页服务的性能和稳定性。Nginx 是异步事件驱动的架构，这使其在处理高并发连接时非常高效。因其资源消耗低，且能够在保持低延迟的同时处理大量网络流量，而被广泛用于优化网站的响应速度和可靠性。Nginx 的配置文件采用简洁的结构，易于理解和修改。Nginx 支持多种类型的代理，如 HTTP、HTTPS、WebSocket 等，也支持各种连接的负载均衡策略。在现代网络架构中，Nginx 常被用作前端服务器，处理客户端请求，并将请求转发到后端的应用服务器。此外，Nginx 的模块化设计允许开发者添加新的功能，如流媒体、安全性增强等。

由于其高性能和灵活性,Nginx 已成为世界上最受欢迎的网页服务器之一,尤其是在面对动态和静态内容时,它提供的性能优势使得大量的互联网公司选择使用 Nginx 作为他们的解决方案。

2.3.2 Vue 技术

Vue.js^[11]是一种轻量级的开源 JavaScript 框架,旨在简化高级 Web 界面的开发。该框架由 Evan You 于 2014 年首次推出,旨在便于构建单页应用(SPA)和界面组件。Vue.js 以其数据绑定和组件化架构而著称,提供了快速的性能和卓越的可扩展性。此外,Vue.js 利用虚拟 DOM、响应式以及可配置的组件系统,实现了高效的更新和渲染。Vue.js 的一个主要优点是其轻量级和快速上手的特性。它易于初学者学习,并且可用于从小型项目到大型企业应用的各种环境。Vue.js 生态系统由 Vuex(状态管理)、Vue Router(路由)、Vuetify(UI 组件库)等多种库和插件组成,使开发人员能够根据需要轻松添加和扩展功能。Vue.js 在全球范围内广泛使用,并且得益于社区的持续贡献而快速发展。其易用性、灵活性和强大的功能使其在前端开发者中非常受欢迎,对塑造 Web 开发的未来发挥着重要作用。

2.3.3 JPA 技术

Java Persistence Application API^[17](JPA)为 Java 应用提供了一种与关系型数据库数据进行交互的标准化方法。此技术利用对象-关系映射(ORM)策略,允许开发者将数据库表转化为对象,进而无需复杂 SQL 查询即可实现与数据库的互动。JPA 旨在简化应用程序数据访问层的开发,使得开发者能够更加集中于面向对象的编程。JPA 兼容众多 ORM 框架,Hibernate 为其广泛应用的实现之一。通过提供标准 API 进行数据库操作,JPA 提升了应用程序对数据库的独立性及代码的可移植性。使用 JPA 时,可以将 Java 类(称为实体)映射到数据库表。每个实体实例对应表中的一行,实体类的字段映射到表的列。开发人员可以通过实体管理器来管理实体的生命周期。这简化了对数据库的 CRUD(创建、读取、更新、删除)操作,并允许抽象化复杂的查询和数据库操作。JPA 技术的一个显著优势在于它支持开发者以面向对象的思维进行数据库操作,这

提高了应用程序的可维护性与扩展性，同时也提升了开发效率。然而，要充分利用 JPA 的潜力。

第 3 章 学校运动场系统设计

3.1 学校运动场系统整体设计

本章介绍学校运动场系统的整体设计。首先介绍了系统的架构设计，阐释了如何通过各组件的结构设计来实现高效的设计。随后，详细展示了系统功能的结构，以及基于 JWT 令牌的无状态设计，这种设计支持安全的用户身份验证和授权过程。然后讨论了第三方登录的方式，以及系统如何有效地识别和处理运行中的错误，确保用户体验的连续性和系统的稳定性。通过这些设计，本系统旨在提供一个既安全又易于管理的运动场管理解决方案。

3.3.1 系统架构设计

该系统设计分为三个主要层次：客户端层，核心服务层，数据层。系统架构如下图 3-1 所示。

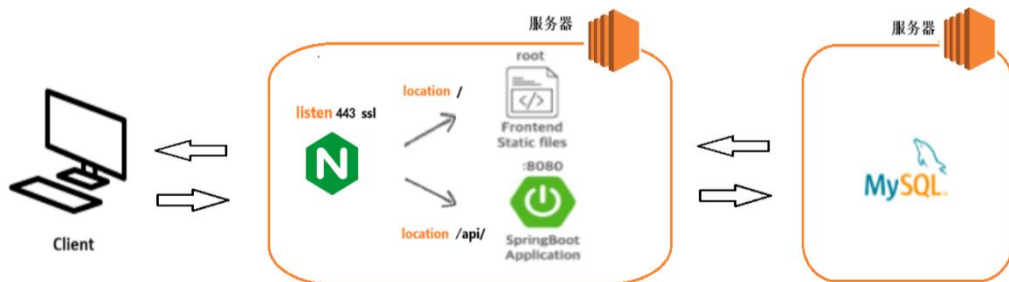


图 3-1 系统整体架构图

如图所示，网站应用通过 HTTP 或 HTTPS API 接口向服务器发送请求。在本系统中，采用了 Nginx 的反向代理技术来首先接收并处理用户的请求。对于定位在“/”的请求，Nginx 服务器从“/var/www/html”目录下提供静态资源响应。对于定位在“/api/”的请求，则从核心服务层提供资源。HTTPS 协议是安全通信协议，只有在实际系统中正确实施，才能有效防止意图破坏协议安全的攻击者对用户的侵害。此外，所有通过 HTTP 80 端口的请求都会被 Nginx 服务器重定向到 HTTPS 443 端口。图 3-2 所示 nginx 的设置。

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    return 301 https://hityundong.com$request_uri;
}

root /var/www/html/dist;

# Add index.php to the list if you are using PHP
index index.html;

server_name hityundong.com;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.

    try_files $uri $uri/ /index.html;
}

location /api/ {
    proxy_pass http://localhost:8080/;
    proxy_redirect off;
}
```

图 3-2 nginx 的设置图。

该系统的所有 HTTP 或 HTTPS 请求和响应均通过 Rest API 进行。^[1]REST 建立在高效的 Web 技术之上，它提供了简单性，并且还确保了所有平台的互操作性和可用性，REST API 的优点是灵活，并且能够非常轻松地与其他系统和应用程序交互。Anshu 和 Virender^[2]进行的一项研究发现，与 SOAP 相比，REST Web 服务在处理请求方面要快得多。^[3]在接口方式上 Melnichuk、Kornienko 和 Boytsova 对“网络服务”进行了研究。“Restful Architecture”的结论是，需要特定 Web 服务器数据的开发者不需要重新编写服务器。这是采用 REST 的原因。

本项目的后端数据库为 MySQL，项目使用了 JPA 作为数据层框架。JPA 是一个标准化的 API，旨在简化 Java 应用程序中使用关系型数据库的方式。它帮助开发者更容易且一致地处理数据库操作。通过 JPA，开发者可以利用对象关系映射 (ORM) 来配置 Java 对象与数据库表之间的映射，从而缩小面向对象编程与关系型数据库之间的差距。JPA 为 CRUD（创建、读取、更新、删除）操作提供了标准 API，允许开发者以简单直观的方式管理数据，而无需编写复杂的数据库查询^[7]。

3.3.1 系统功能结构设计

用户模块：用户作为系统的核心参与者，拥有个人信息管理、参与比赛管理以及安全登录等功能，实现个性化的用户体验。

运动场模块：为用户提供详细的运动场信息查看和比赛日程查询功能，让用户轻松了解运动场的活动和赛事情况。

比赛模块：用户可通过此模块创建个人比赛或加入已有比赛，并方便地管理比赛结果，促进赛事信息的全面管理和更新。

团队模块：提供团队创建、成员管理和团队信息查看等功能，促进团队之间的协作与交流，推动团队活动的顺利进行。

请求模块：用户可提交各类请求，如赛事申请、信息修改等，系统将对请求进行管理和处理，确保用户需求得到及时响应。错误管理模块：负责捕获和处理系统运行中的错误，保障系统稳定性和用户体验，为用户提供流畅可靠的服务。运动场系统功能结构。如图 3-3 系统功能结构图所示。

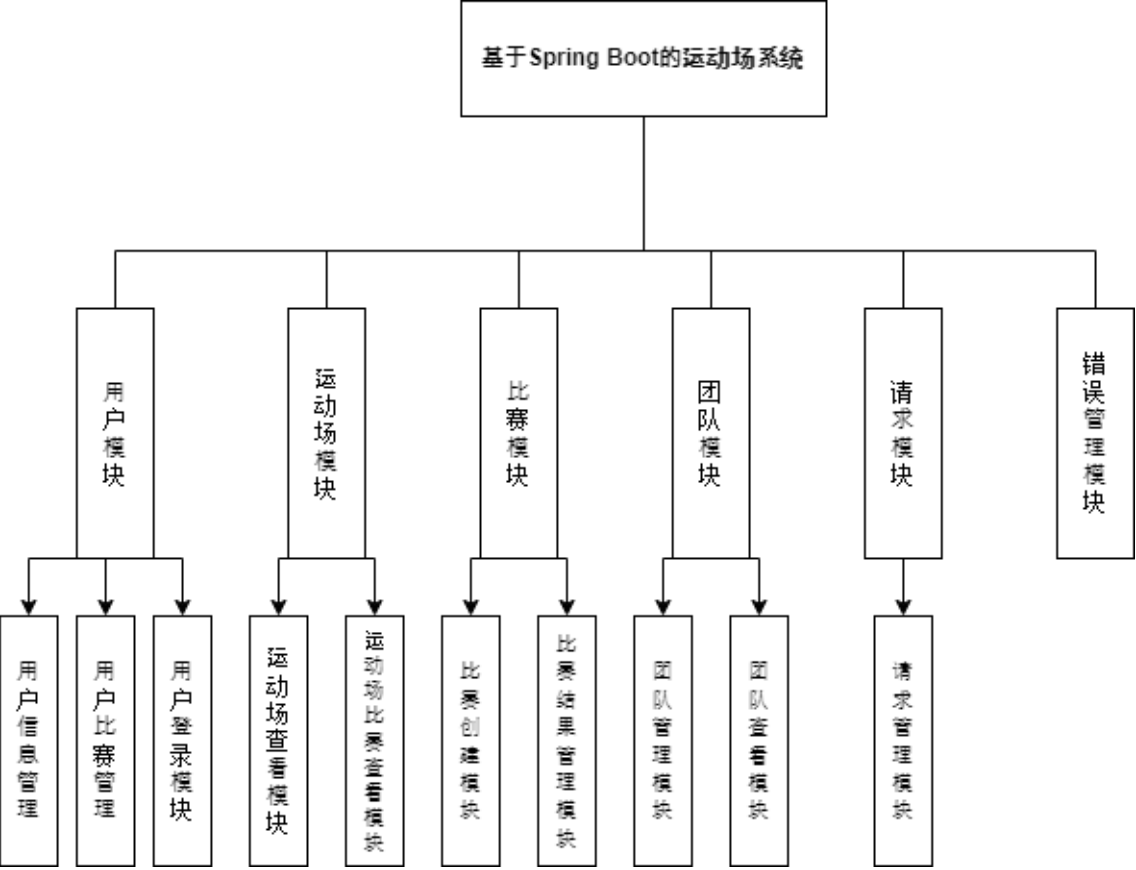


图 3-3 系统功能结构图

3.3.2 基于 JWT 令牌 (Token) 的无状态系统设计

JWT 令牌(Token)^[4]将作为 JSON 对象的数据通过将其表示为字符串来编码, 并使用 HMAC(基于哈希的消息认证码)等算法进行数字签名以确保数据的安全。Aldy, Alam 和 Muhammad 的研究表明^[8], JWT 是一种无状态身份验证形式, 这意味着 JWT 不是存储在服务器上, 而是存储在客户端中。

(1) 前端 JWT 无状态系统设计

除登录过程外, 所有要求都将在 header 上包括 JWT Token 进行安全检查。前端在发起所有 http 请求之前, 会检查当前用户的 Local Storage 中是否存在 JWT Token。如果没有任何 JWT token, 任何请求都不能包括 JWT token, 因此将直接重定向到登录页面。如果存在 JWT Token, 将对核心服务层进行 Access Token 的有效性检查。如果有效, 正常进行请求, 如果无效, 就从用户的 Local Storage 传送核心服务层 Refresh Token。再次要求检查 Refresh Token 的有效性, 如果有效, 从核心服务层接收新的 Access Token, 并储存在 Local Storage 中。如果它无效, 它将直接重定向到登录页面。图 3-4 所示前端 JWT 无状态系统设计。

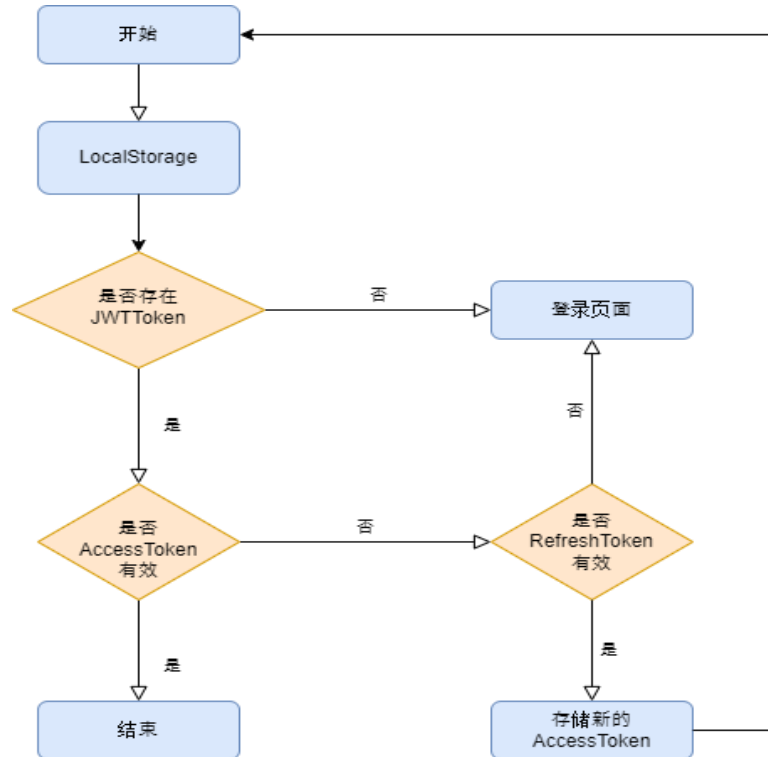


图 3-4 前端 JWT 无状态系统流程图

（2） 后端 JWT 无状态系统设计

在后端，除了登录过程以外，所有请求都将通过 JwtAuthenticationFilter 过滤器。此过滤器负责检查 Access Token 的有效性，如果有效，则允许请求通过所有过滤器，并通过 Controller 进行处理。如果无效，则向用户请求 Refresh Token。接收到的 Refresh Token 将在 JwtRefreshTokenFilter 过滤器中进行有效性检查，如果有效，将生成新的 Access Token。如果无效，则返回 HTTP 403 错误。图 3-5 所示前端 JWT 无状态系统设计。

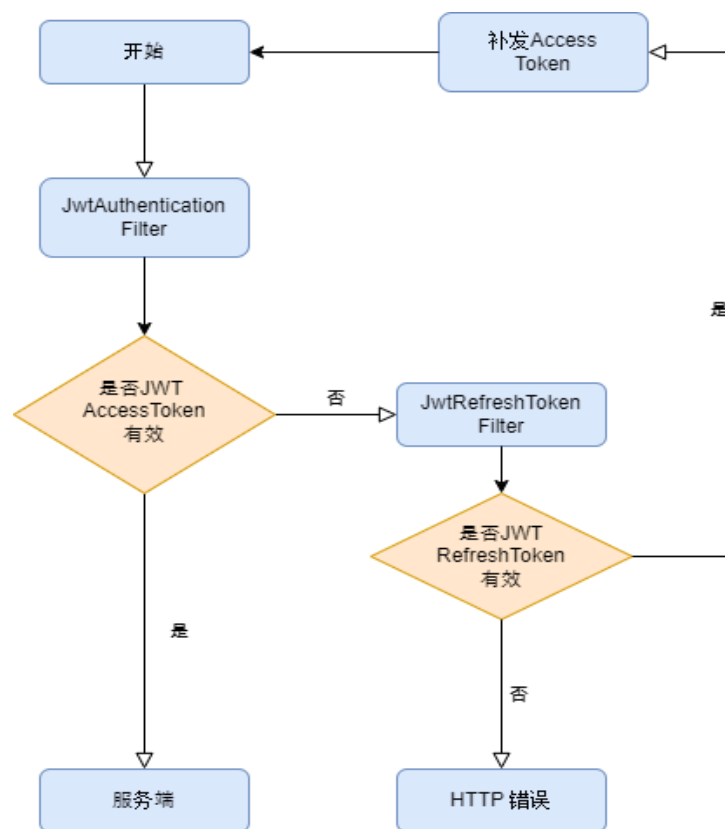


图 3-5 后端 JWT 无状态系统流程图

学校运动场系统将利用 JWT 来提高安全性，并在 JWT 中包含权限或有效期，以为学生提供灵活的服务。此外，通过在每次请求时使用用户的 JWT 令牌来检查身份验证权限，无需在服务器上存储用户信息，因此可以实现无状态系统，即使在处理大量学生流量时，服务器负载也将较低。

3.3.3 第三方登录设计

Spring Security 和 OAuth2^[15] 是该系统使用的登录框架。Spring Security 是一个功能强大的身份验证和访问控制框架，用于保护基于 Java 的应用程序。它提供了一种灵活的方式来管理用户认证、授权和安全性，并与 Spring 框架紧密集成。OAuth 2.0 是一种用于授权的框架，其主要目的是为第三方应用程序提供受限访问，而无需用户提供其凭据。它使用户能够向第三方应用程序提供对其资源的有限访问权限，同时保护用户的个人凭据和隐私信息。用户首先通过客户端应用发起特定服务的登录请求，引导至服务器。服务器重定向到登录界面，用户输入账户信息后，第三方认证服务器验证身份。验证成功后，第三方认证服务器将用户重定向回服务器，并提供一个授权码。服务器应用使用该授权码，向第三方服务器请求访问令牌。第三方认证服务器审查授权码及客户端凭据后，发放访问令牌。服务器凭借访问令牌，向第三方认证服务器请求用户信息。第三方认证服务器确认令牌无误后，返回用户信息。通过这一机制，用户能够安全地授权客户端应用访问其在服务提供方的部分信息。图 3-6 所示第三方认证过程时序图。

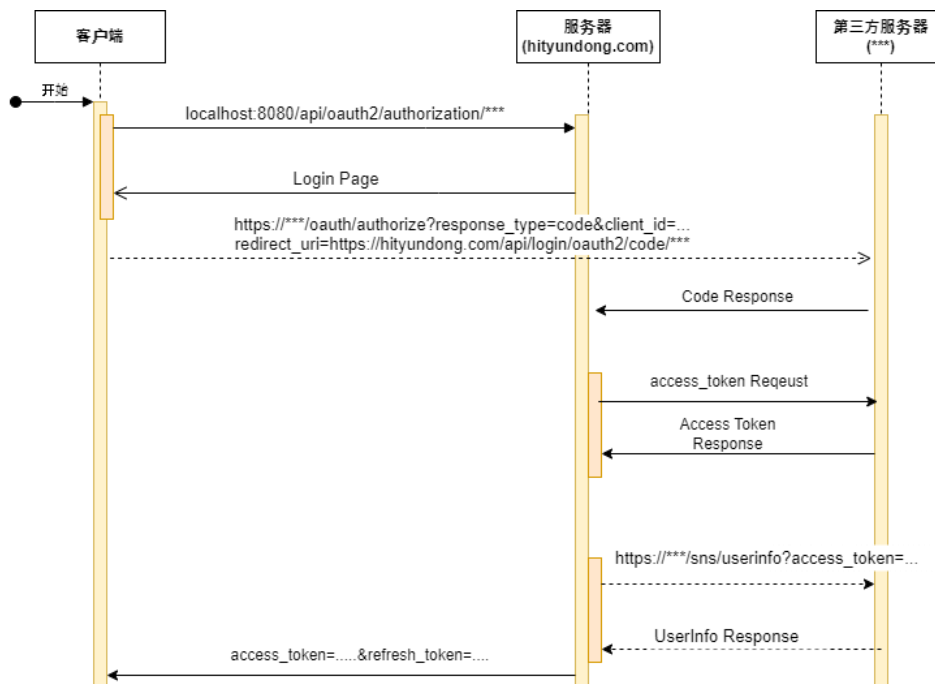


图 3-6 第三方认证过程时序图

当从第三方认证服务器获得认证令牌后，后端服务器的UserServiceImpl将向第三方认证服务器请求用户信息。当从第三方认证服务器获得用户信息的attributes后，后端服务器将提取所需的信息，username、provider等，然后创建User Class。之后，如果User已经存在于数据库中，则更新信息，如果不存在于数据库中，则保存，以存储用户信息。成功保存用户信息后，服务器将调用LoginSuccessHandler。在此之后，根据新的用户信息生成新的JWT令牌，access_token的有效期自生成时起为 1 小时，而refresh_token的有效期自生成时起为 30 天。并将用户状态更新为'isLoggedIn = true'。同时，为了防止之前的refreshToken再次使用，系统会在数据库中更新一个新的refreshToken，并将新生成的access_token和refresh_token通过URL发送给用户。图 3-7 所示第三方认证过的用户信息处理程结构图。

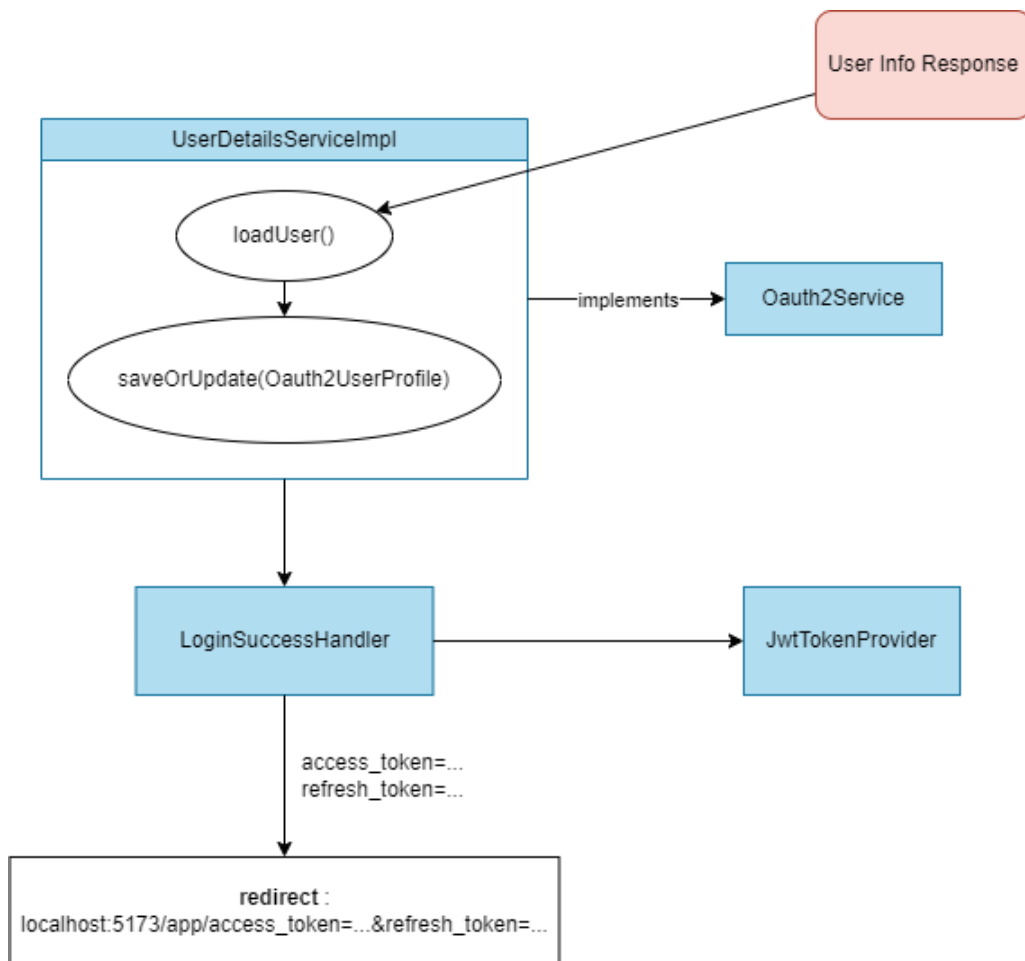


图 3-7 第三方登录用户信息处理结构图

3.3.4 系统错误处理设计

Controller Advice^[16]提供了一种管理 Spring Framework 中全局错误处理和数据绑定机制。这允许在特定控制器或整个应用中应用共同的错误处理。例如，可以通过 Controller Advice 为应用程序中可能发生的每种错误类型定义特定操作，或者在特定条件下自动将数据添加到模型中。这将减少代码的重复，并允许在整个应用中保持一致的错误处理和数据管理策略。Controller Advice 是将@Controller Advice 使用与@ExceptionHandler 一起使用，可以定义各种情况下的处理逻辑。图 3-8 所示 Spring 框架中，当异常发生时 DispatcherServlet 如何将异常传递给正在处理请求的控制器。如果控制器内部能处理异常，则 Controller Exception Handler 处理错误。如果不行，则通过 ControllerAdvice 处理异常。如果所有这些步骤都未能处理异常，则最终客户端将返回 HTTP 错误响应。这种结构使得集中式异常处理成为可能，从而帮助更有效地管理异常处理逻辑。

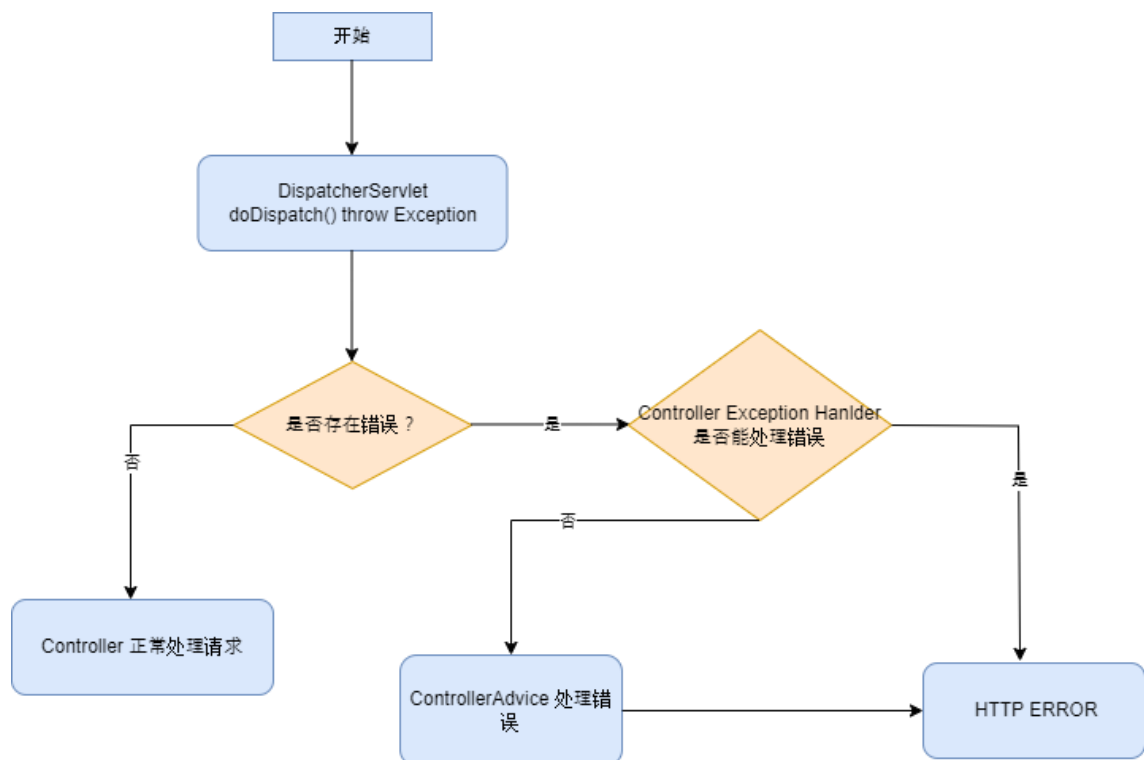


图 3-8 错误处理流程图

图 3-9 为 ControllerAdvice 的结构。当 Controller 执行逻辑时出现错误，ControllerAdvice 将捕获该错误，并判断是否有相应的错误处理方法。如果能够处理该错误，则在 ControllerAdvice 的相应函数中进行处理，然后返回响应的 HTTP 错误响应。

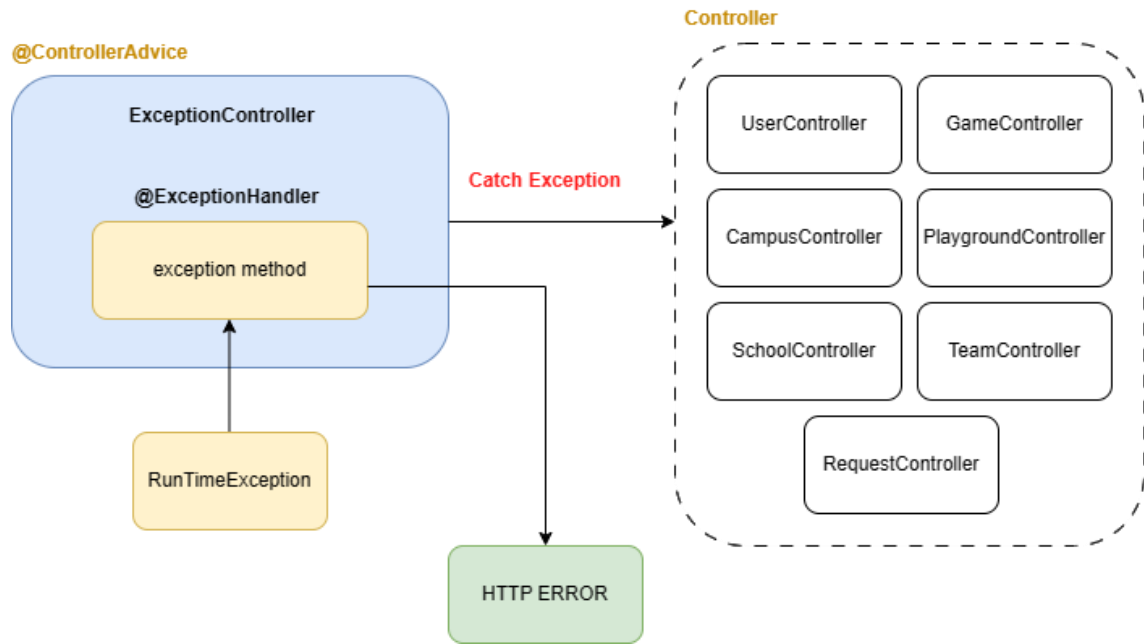


图 3-9 错误处理结构图

3.2 学校运动场系统详细设计

3.2.1 应用设计模式

设计模式是针对软件设计中常见问题的通用解决方案。通过设计模式，开发人员可以利用已经被证实有效的经验来解决软件设计中的各种挑战。设计模式不是具体的实现，而是一种通用的设计思想，可以应用于不同的情景中。设计模式可以帮助开发人员提高代码的可维护性、可扩展性和重用性，从而提高软件的质量和开发效率。在软件设计过程中，合理地运用设计模式可以使系统更加灵活、可靠、易于理解和维护。

3.2.2.1 工厂模式（Factory pattern）和 策略模式（Strategy method）

工厂模式和策略模式^[12]是一种设计模式，工厂模式作为一种创建型设计模式，在软件工程中，通过定义一个用于创建对象的接口，允许子类决定实例化哪一个类型。这种方法减少了客户端代码与对象创建过程的耦合，提高了系统的可扩展性与灵活性。

在 Java 对象导向编程框架内，策略模式使得开发者可以定义一系列的算法，并将它们封装成类，通过继承一个共同的接口来保证它们可以互换使用。这种设计模式让算法的变更和客户端的调用解耦合，增强了软件系统的灵活性。具体实现时，可以通过在客户端代码中设置不同的策略对象来动态改变对象的行为。这样，软件在扩展新行为时不需要修改现有代码，只需添加新的策略类即可。

本文通过利用这些设计模式，设计了更具可用性和扩展性的结构^[13]。在最初阶段，用户的请求类型相对较少，大约只有两种。然而，随着比赛种类的增加，请求类型也相应增多，这导致了函数内部需要处理更多的分支，从而需要修改代码。图 3-10 所示利用设计模式前的请求模块图。

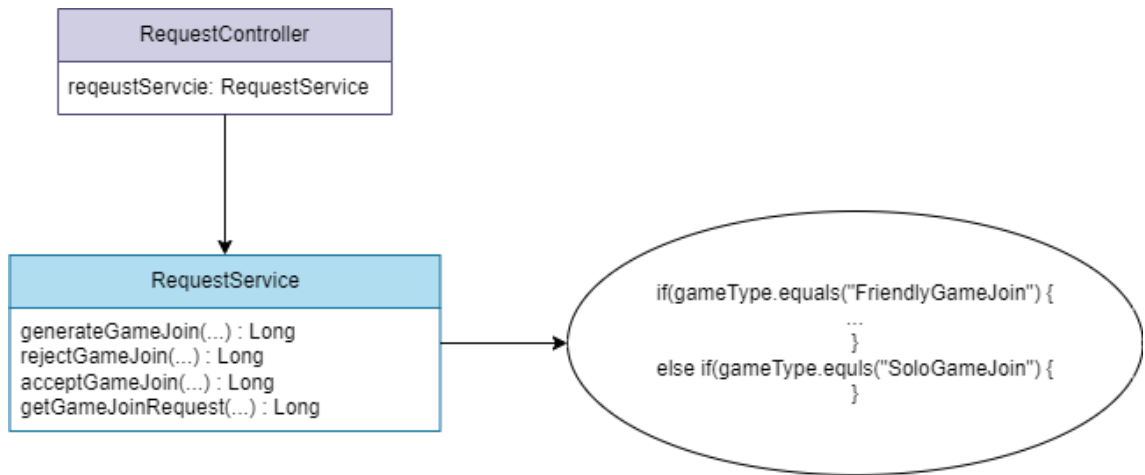


图 3-10 采用设计模式前的请求模块图

这种情况下，原有的面向对象设计原则受到挑战，最终导致了难以维护的代码结构。为了应对这一挑战，需要一种能够有效管理增长中的请求种类、同时保持代码的可维护性和扩展性的设计策略。通过引入灵活的设计模式，如策略模式和工厂模式，可以在不直接修改现有代码的基础上，添加新的行为和处理逻辑，从而避免了代码结构的僵化和维护难度的增加。这样，即便面对不断变化和增加的用户需求，也能保持软件架构的灵活性和可扩展性。

在用户请求增长和多样化的情况下，采用策略模式能够提高代码的可维护性和扩展性。例如，在一个比赛服务请求处理系统中，可以定义一个请求服务接口 RequestService，然后针对不同类型的比赛请求，实现多个具体服务类，如 TeamJoinRequestService、SoloGameJoinRequestService 等。每个服务类都实现了 RequestService 接口，根据不同的请求类型来执行相应的逻辑。这种设计有效地将请求类型与请求处理逻辑解耦，使得在新增请求类型时，只需添加新的服务类而不需修改现有代码，从而保持了架构的灵活性和代码的可维护性。这样的对象导向设计，通过多态性原则实现了行为的动态绑定，并促进了高内聚与低耦合的设计目标。图 3-11 所示采用策略模式的请求模块。

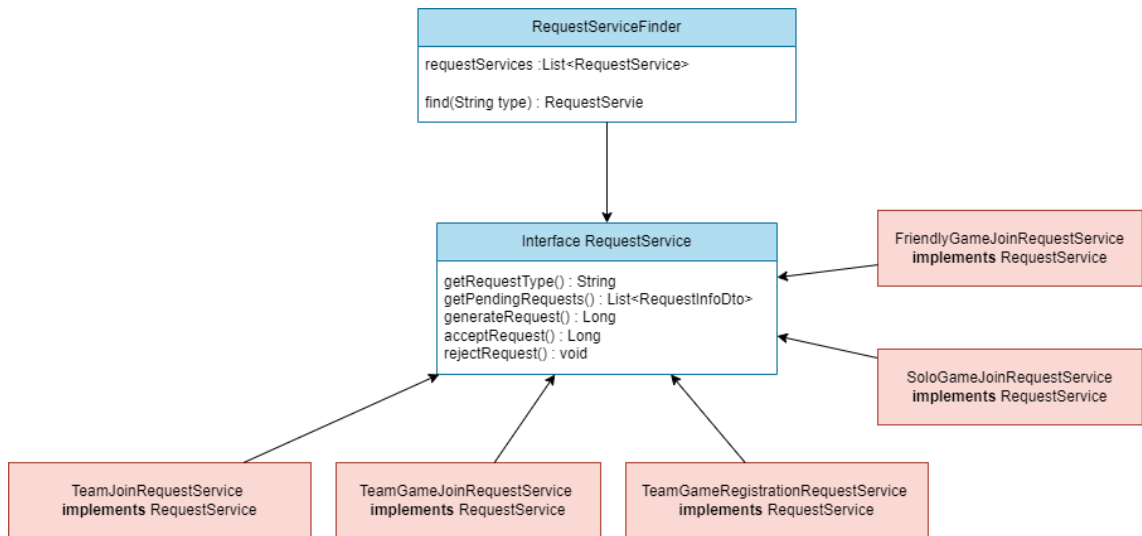


图 3-11 采用设计模式的请求模块图

在策略模式成功实施的基础上，通过引入工厂模式的概念，对请求服务对象的创建过程进行了优化。在这个设计模式中，添加了一个新的组件 RequestServiceFinder，它的职责是根据请求的具体类型，动态地寻找并返回相应的 RequestService 实现类的实例。这种方法不仅保留了策略模式的灵活性，使得可以在运行时切换不同的处理策略，同时也简化了客户端代码，因为客户端不再需要了解具体的服务实现细节，只需通过 RequestServiceFinder 来获取所需的服务实例。此种设计使得系统架构能够以开放封闭原则应对未来的变动，即在不修改现有代码的前提下扩展新的功能。图 3-12 所示 RequestServiceFinder 的寻找函数图。

```

1 usage
private final List<RequestService> requestServices;

8 usages
public RequestService find(String type) {
    return requestServices.stream() Stream<RequestService>
        .filter(requestService -> requestService.getRequestType().equals(type))
        .findFirst() Optional<RequestService>
        .orElseThrow(RuntimeException::new);
}

```

图 3-12 RequestServiceFinder 函数图

为了适应不断增长和多样化的功能，同时保证系统架构的可扩展性，本文的几部分模块应用了设计模式来加强面向对象编程的效能。通过这一系列策略，即使在功能种类不断扩大的情况下，系统设计仍然保持了清晰的逻辑和稳固的架构，确保了系统处理的效率和适应性。

3.2.2 用户管理模块

3.2.2.1 用户信息功能设计

用户能够在系统中访问和查看自己的个人资料，其中包含他们的个人资料照片、昵称、参与过的比赛记录以及他们所在的团队信息。在用户的个人信息页面上，他们有权更改自己的资料头像和昵称。图 3-13 所示用户信息服务类图。

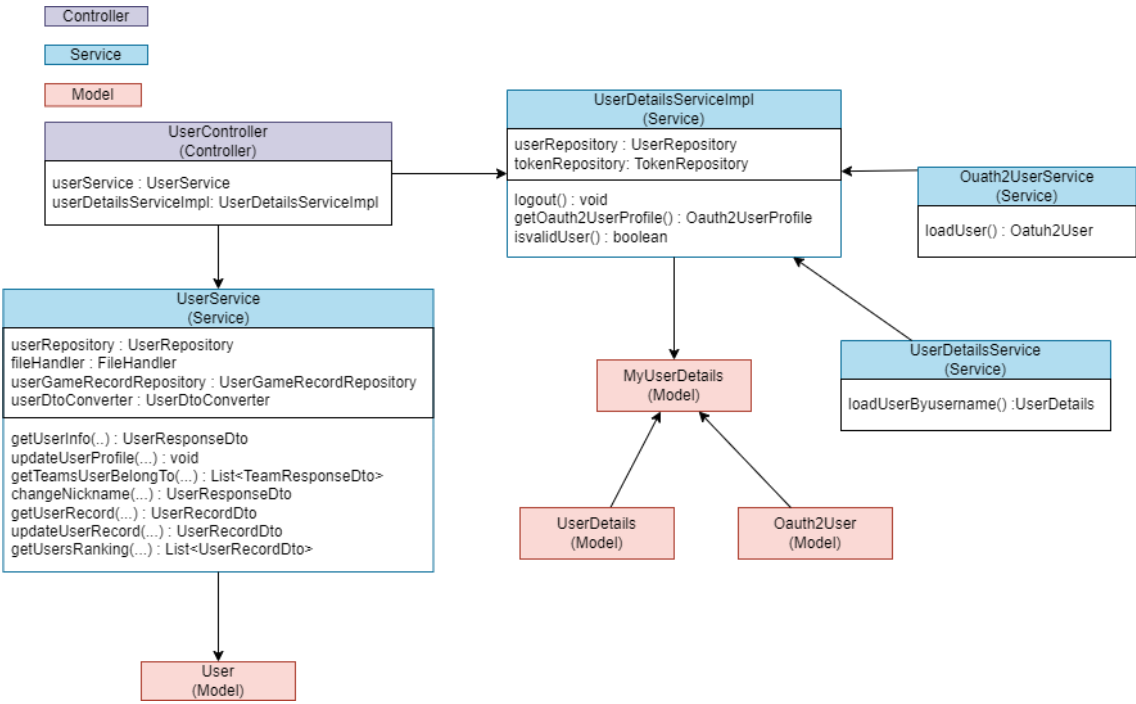


图 3-13 用户信息服务类图

3.2.2.1 用户比赛管理服务设计

图 3-14 所示用户比赛管理服务类图。该服务关于用户参与或主办的比赛的模块。当用户请求其参与的比赛时，UserJoinGameService 将回应用户当前参与的比赛信息，包括比赛名称、开始时间、比赛类型和位置等。而 UserHostGameService 则回应用户主持的比赛。查找比赛时使用了 GameFinder 模块，并且使用 GameSorting 模块按日期对比赛进行排序，从而实现了职责分离。这些模块都只执行一个功能，因此满足单一职责原则。

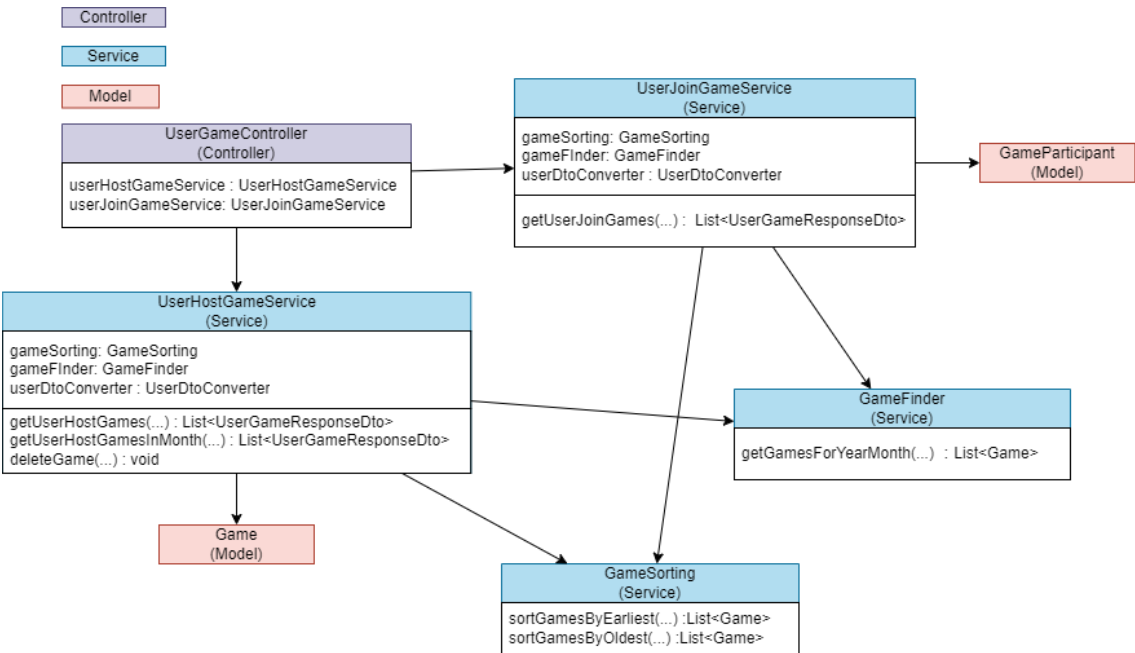


图 3-14 用户比赛管理服务类图

3.2.3 运动场管理模块

3.2.3.1 运动场查看功能设计

用户在特定项目中选择学校的某个或所有校区。如果所选校区中有即将开始比赛，系统将显示最多 n 场比赛。查找运动场时使用了 PlaygroundFinder 模块。同时，还会显示该校区内当前激活的运动场列表。运动场查看功能的类图如 3-15 所示。

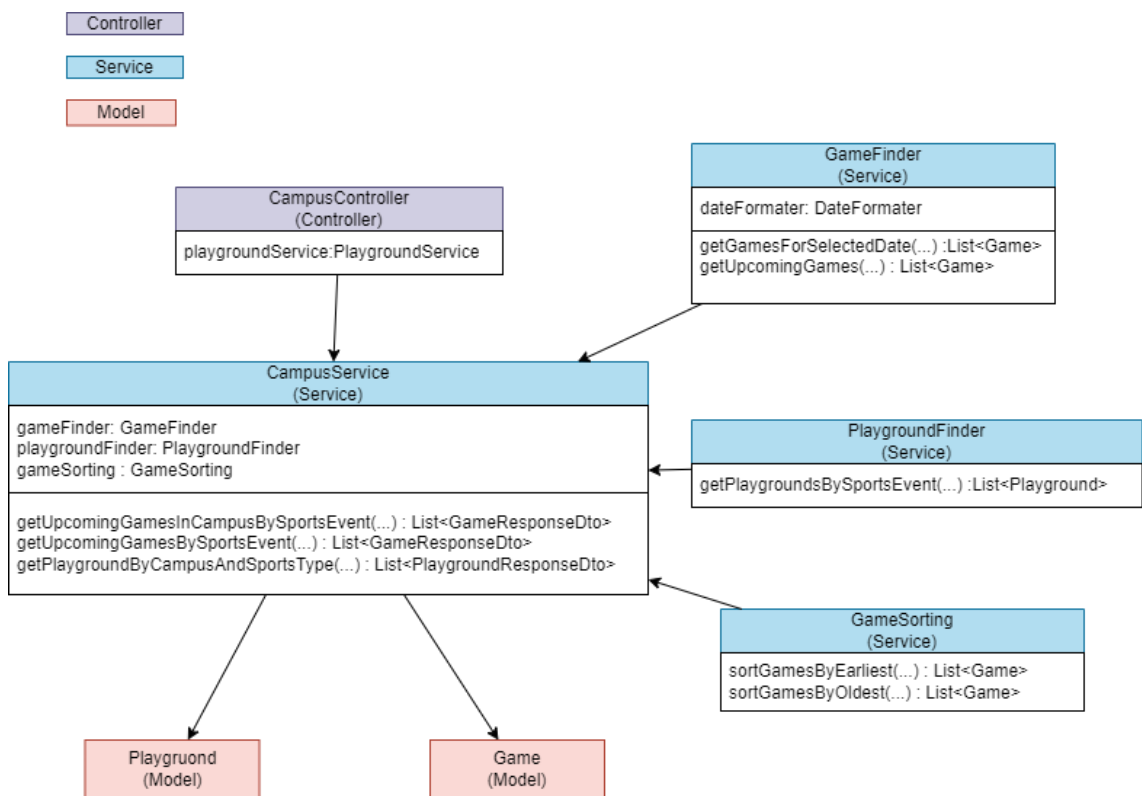


图 3-15 运动场查看服务类图

3.2.3.2 运动场比赛查看功能设计

用户选择运动场后，可以查看有关运动场的信息。运动场信息包括运动场名称、运动场招牌你和位置信息。此外，还可以查看当前在该运动场进行的比赛以及即将开始的比赛信息。该运动场还提供特定一天内比赛的时间线。运动场比赛查看功能的类图如 3-16 所示。

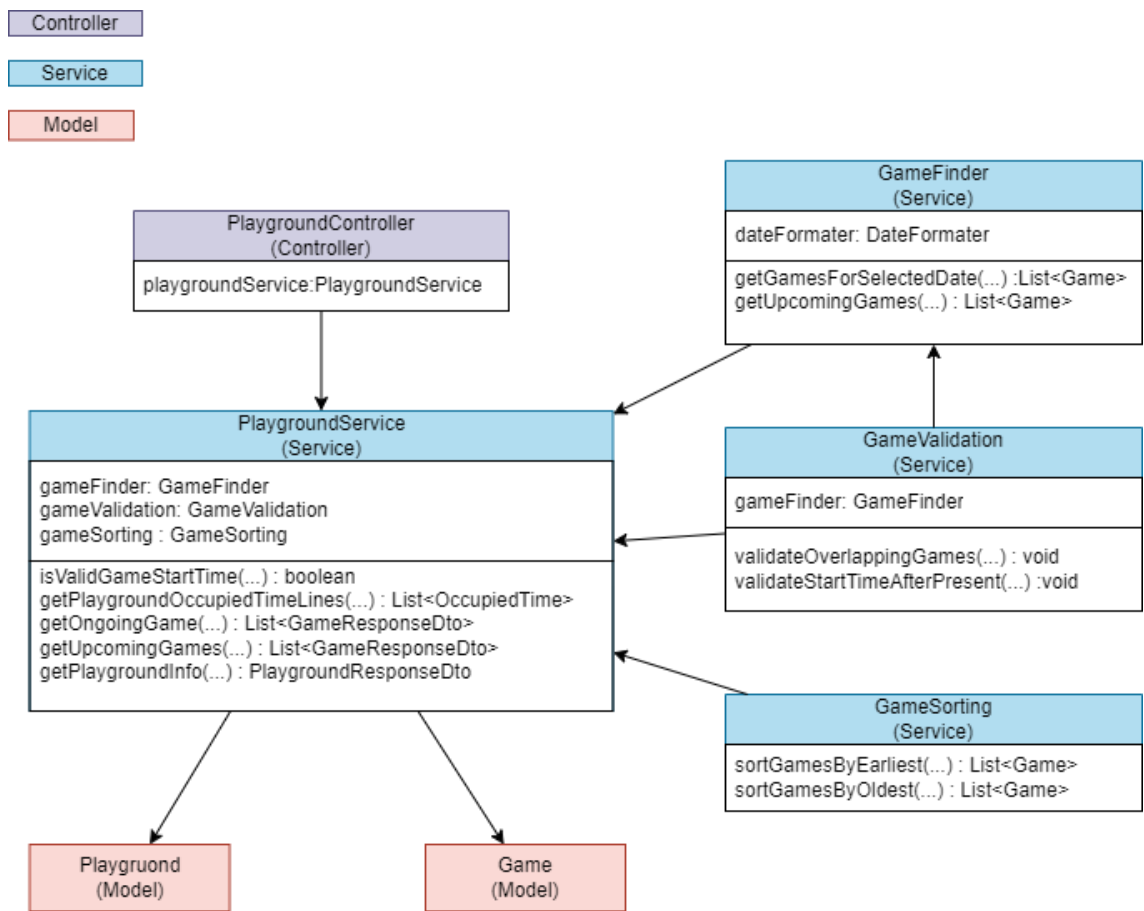


图 3-16 运动场比赛查看服务类图

3.2.4 比赛管理模块

3.2.4.1 比赛创建功能设计

创建比赛的过程由希望创建比赛的用户启动，用户首先选择希望进行比赛的运动场。在创建比赛时，用户需设定比赛名称、比赛的开始时间以及预计持续的时间。在选择比赛开始时间时，系统会检查所选运动场在该时间是否已有比赛安排，如果没有，比赛则被创建。在创建比赛的界面上，会显示选定日子内的所有比赛时间。比赛有多种类型，目前仅支持选择友谊赛和竞赛两种类型。比赛创建功能的类图如 3-17 所示。

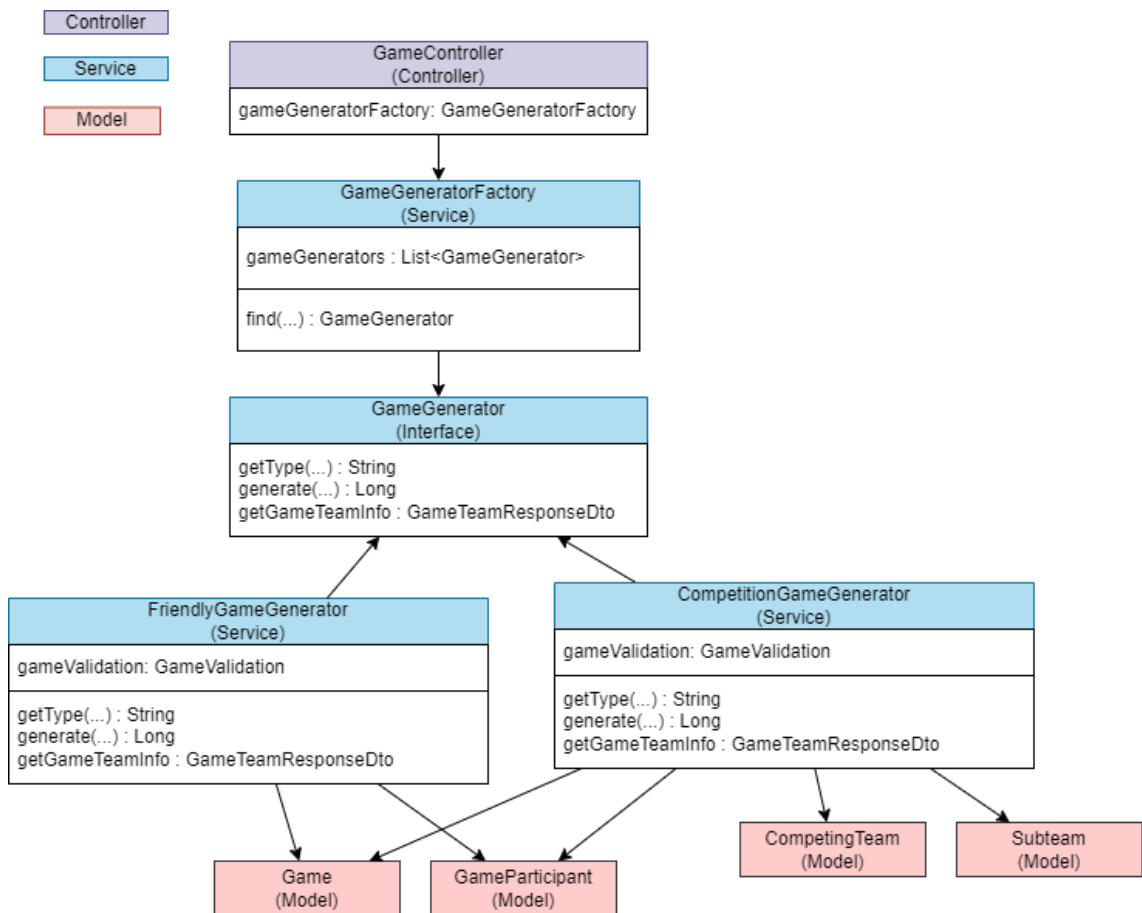


图 3-17 比赛创建功能的类图

3.2.4.1 比赛结果功能设计

比赛结束后，比赛的主持者可以提交比赛结果。如果是友谊赛，将进行得分输入，但胜负不会对用户产生影响。如果是竞赛，则在输入比分后确定胜负，并且比赛的胜负结果会影响参赛者之前的比赛记录。一旦输入了比分，便可以进行更改，若进行更改，则之前的记录会被删除。比赛结果功能的类图如 3-18 所示。

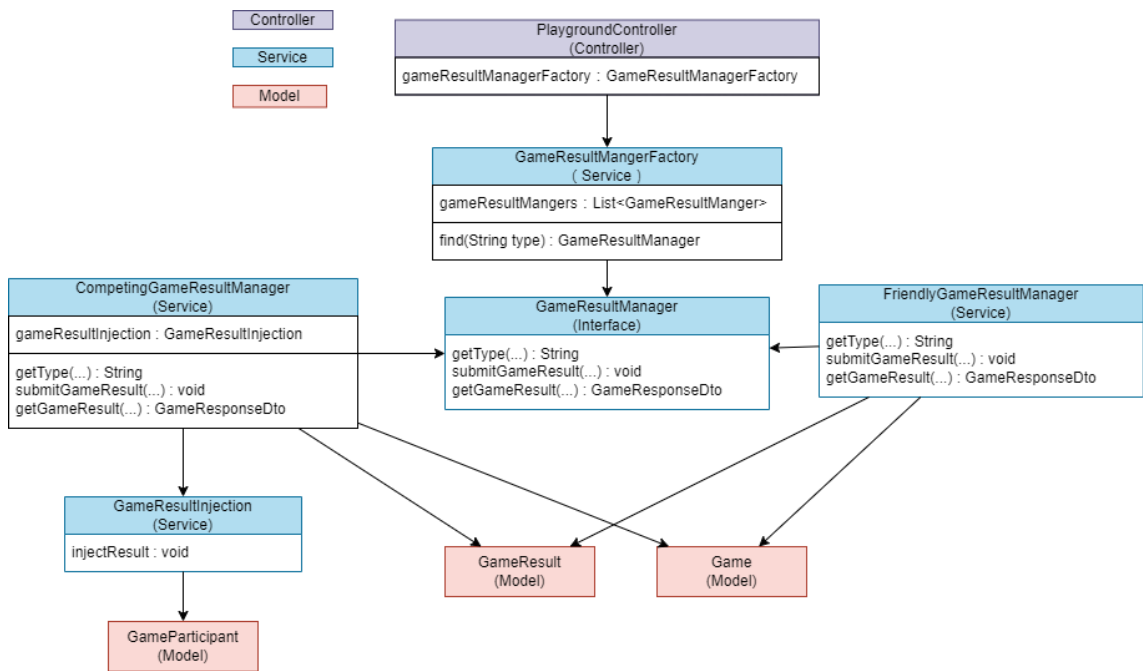


图 3-18 比赛结果功能的类图

3.2.5 团队管理模块设计

3.2.5.1 团队管理功能设计

用户可以创建团队，并在创建时必须输入团队名称、团队头像和团队参与的项目。团队介绍是可选项。最初，创建团队的用户将成为团队队长。团队队长有权限删除团队并处理加入团队的请求。团队成员可以申请加入或退出团队。加入团队后，在比赛中将作为团队的一员参与比赛。团队管理功能的类图如 3-19 所示。

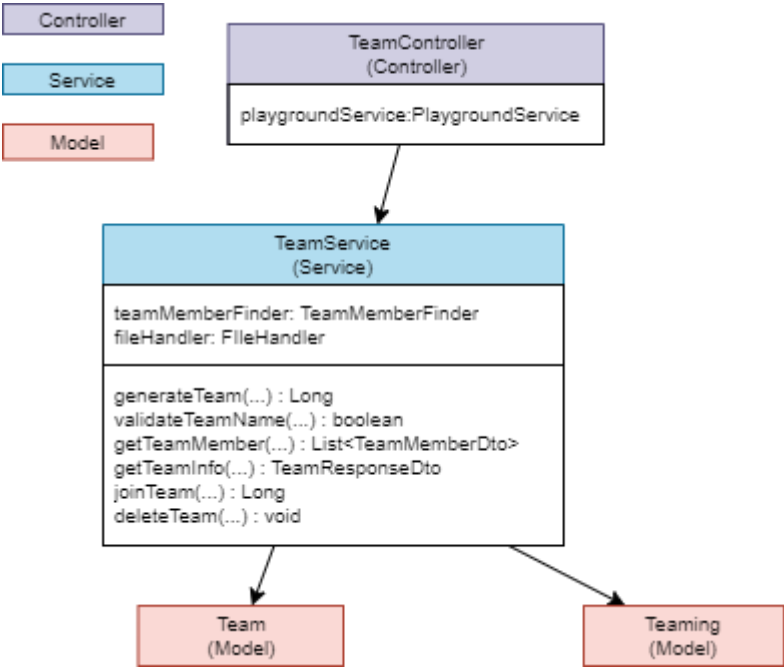


图 3-19 团队管理功能服务类图

3.2.5.2 团队查看功能设计

用户可以查看当前存在的所有团队列表。列表最初显示学校中所有项目的团队，选择特定项目后，系统会列出该项目的所有团队。团队信息展示包括团队头像、团队名称和参与的项目。团队查看功能的类图如 3-20 所示。

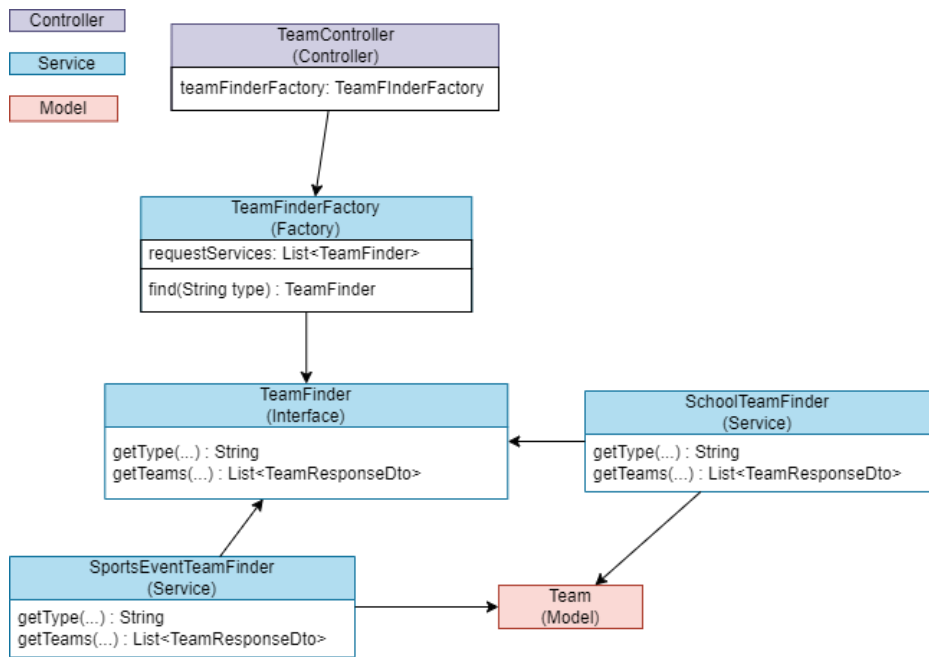


图 3-20 团队查看功能服务类图

3.2.6 请求管理模块设计

3.2.6.1 请求管理功能设计

请求管理由 **RequestController** 执行。简单的请求处理通过 **RequestProcessor** 来处理如删除请求等逻辑。请求分为五种类型：团队加入请求、友谊赛加入请求、竞赛个人参加请求、竞赛小队创建请求、竞赛小队加入请求。每种请求都在 **RequestFinder** 中输入想要处理的请求后，由实现了 **RequestService** 的实现体处理创建请求、接受请求、拒绝请求、查看请求等逻辑。图 3-21 所示请求管理功能类图。

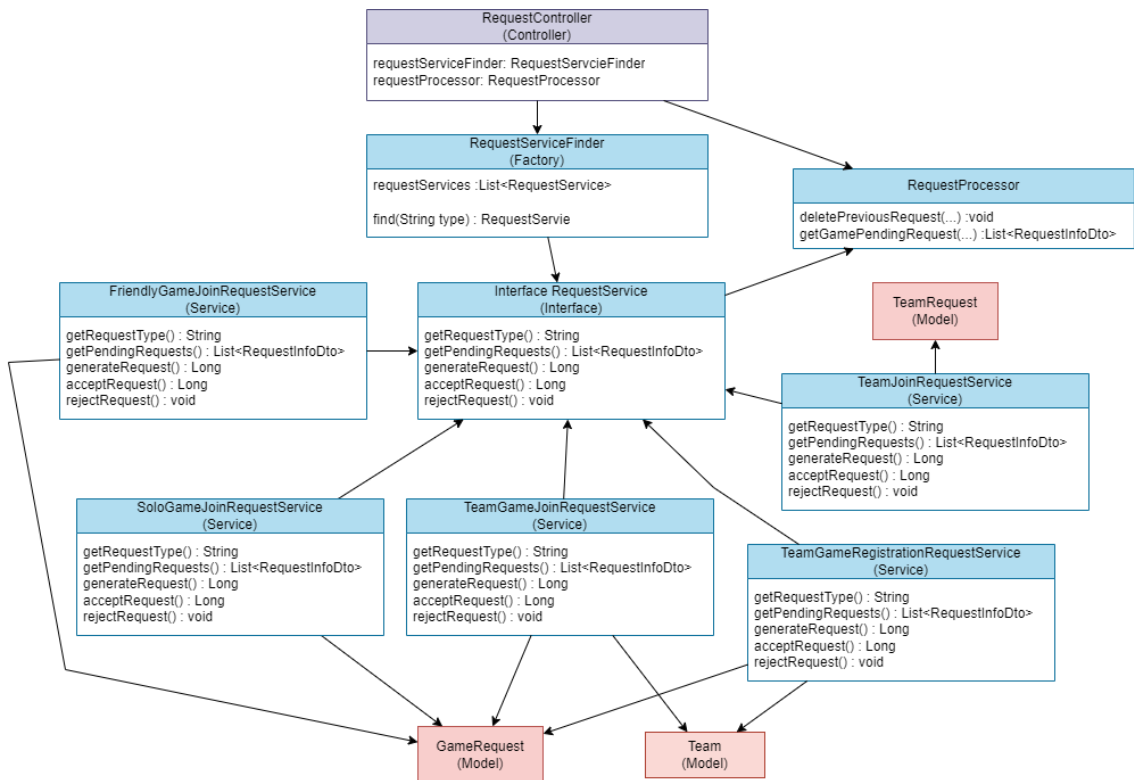


图 3-21 请求管理功能类图

3.3 学校运动场系统数据库设计

3.3.1 用户和团队模型设计

用户和团队的关系模型如图 3-22 所示，用户和团队关系模型有用户表，团队表，用户团队关系表，文件表。用户表和团队表分别用外键将具有头像信息的 `upload_File` 一对一关系。一个用户可以加入多个团队，而一个团队可以包括多名用户。每个团队有一个队长，该队长与用户表具有一对一关系。该关系模型满足 BCNF 范式。

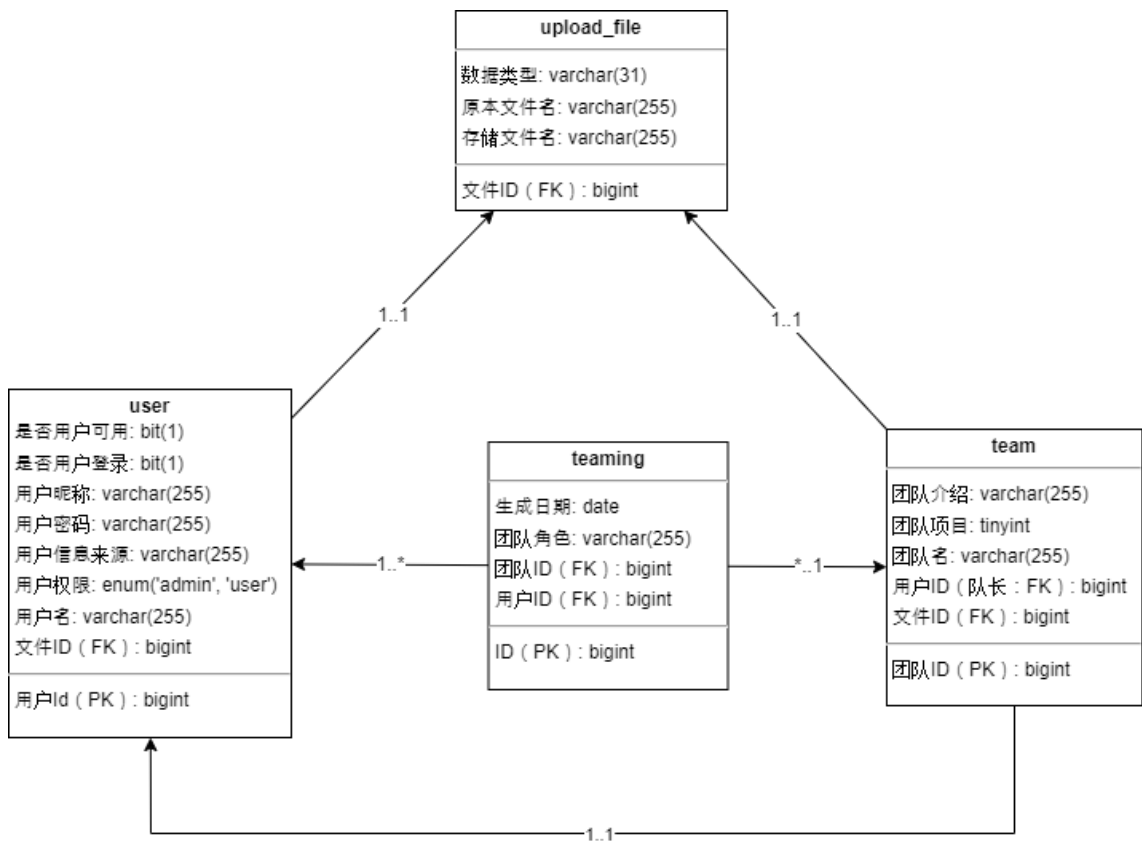


图 3-22 用户和团队实体关系图

用户和团队数据库主要表格式如下。

表 3-1 用户表

字段名称	类型	空否	说明
id	bigint	否	主键
is_enable	bit	否	
is_logged_in	bit	否	
nickname	varchar	否	
password	varchar	否	
provider	varchar	否	
role	enum	否	
username	varchar	否	
profile_img_id	bigint	是	外键

表 3-2 文件表

字段名称	类型	空否	说明
id	bigint	否	主键
org_file_name	varchar	否	
store_file_name	varchar	否	
dtype	varchar	是	

表 3-3 团队表

字段名称	类型	空否	说明
id	bigint	否	主键
description	varchar	是	
sports_event	tinyint	否	
team_name	varchar	否	
leader_id	bigint	否	外键
team_pic_id	bigint	是	外键

表 3-4 用户和团队关系表

字段名称	类型	空否	说明
id	bigint	否	主键
role	varchar	是	
team_id	bigint	否	外键
user_id	bigint	否	外键
create_time	date	否	

3.3.2 场地模型设计

场地的关系模型如图 3-23 所示，一所学校可以拥有多个或多个校区，每个校区可以有多个运动场。在系统结构中，校区表将使用学校的 ID 为外键，而运动场表则使用相应校区的 ID 作为外键。该关系模型满足 BCNF 范式。

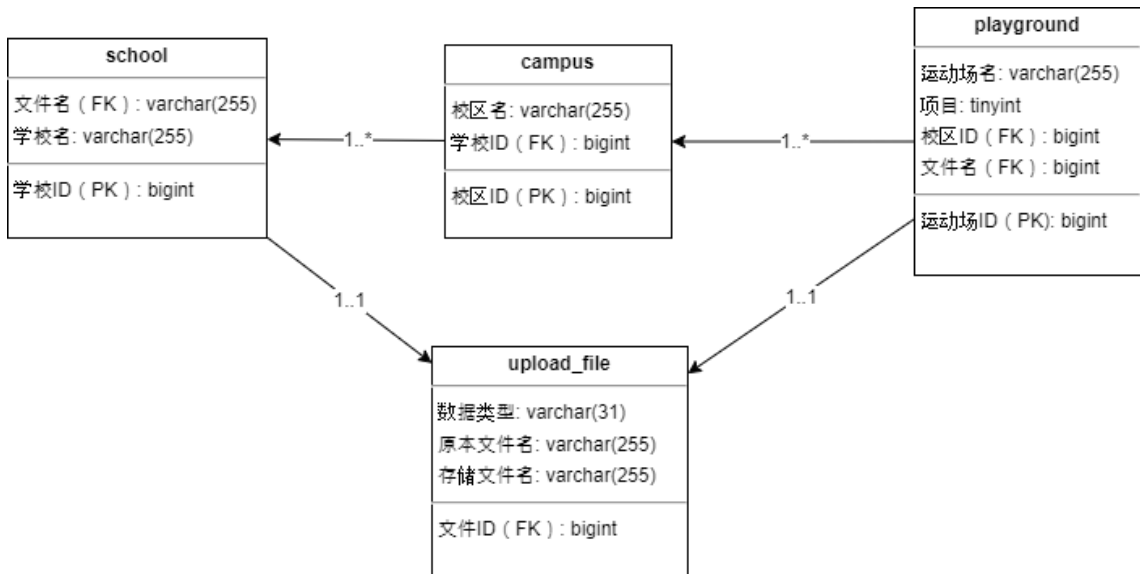


图 3-23 场地实体关系图

场地数据库主要表格式如下。

表 3-5 运动场表

字段名称	类型	空否	说明
id	bigint	否	主键
name	varchar	否	
sports_event	tinyint	否	
campus_id	bigint	否	外键
img_id	bigint	是	外键

表 3-6 学校表

字段名称	类型	空否	说明
id	bigint	否	主键
School_name	varchar	否	
School_img	bigint	是	

表 3-7 校区表

字段名称	类型	空否	说明
id	bigint	否	主键
Campus_name	varchar	否	
school_id	bigint	否	

3.3.3 比赛模型设计

比赛关系模型如图 3-24 所示，每个运动场可以拥有多场比赛，每场比赛都会有运动场 ID 作为外键，与运动场形成一对多关系。每场比赛都有一个结果，该结果拥有比赛的主键作为外键。参与比赛可以是个人或者小队，一个小队有一个团队的主键，如果以团队形式参与，将通过小队的比赛参与者名单来建立比赛和团队之间的关系。比赛参与者名单是一个表，用于建立比赛和用户之间的参与关系，一个团队或比赛中可以有多个参与者。

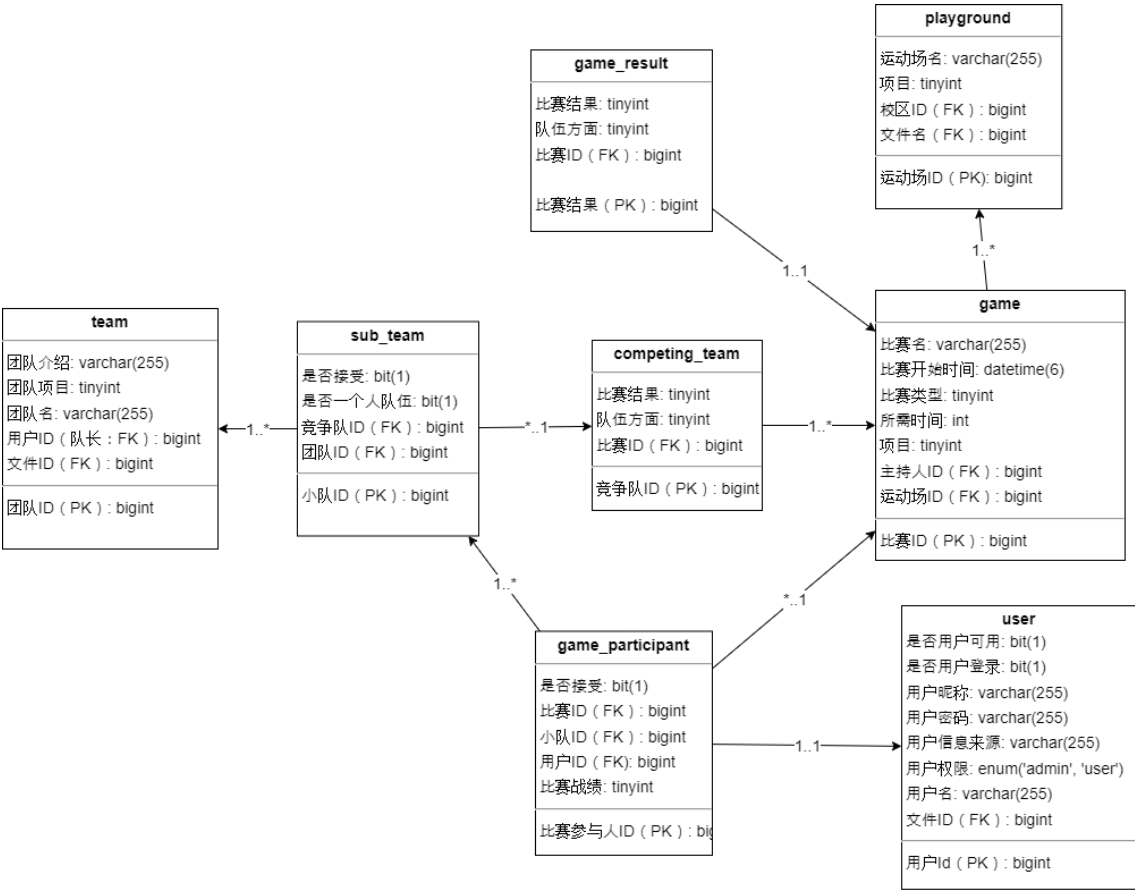


图 3-24 比赛实体关系图

比赛数据库主要表格式如下。

表 3-8 比赛表

字段名称	类型	空否	说明
id	bigint	否	主键
game_name	varchar	否	
game_start_date_time	datetime	否	
Game_type	tinyint	否	
Running_time	int	否	
sports_event	tinyint	否	
host_id	bigint	否	外键

playground_id	bigint	否	外键
---------------	--------	---	----

表 3-9 比赛参与者表

字段名称	类型	空否	说明
id	bigint	否	主键
is_accepted	bit	否	
game_id	bigint	否	外键
sub_team_id	bigint	是	外键
user_id	bigint	否	外键
game_record	tinyint	是	

表 3-10 参赛队伍表

字段名称	类型	空否	说明
id	bigint	否	主键
game_result	tinyint	是	
game_team_side	bigint	否	
game_id	bigint	否	外键

表 3-11 小队表

字段名称	类型	空否	说明
id	bigint	否	主键
is_accept	bit	是	
is_solo_team	bit	否	
competing_team_id	bigint	否	外键
team_id	bigint	否	外键

3.3.4 请求模型设计

请求关系模型如图 3-25 所示，一个用户对一个比赛只能发送一个请求，比赛请求根据类型分为不同的表。组队参加、创建小队参加、单独参加、参加友谊赛等都继承自 game_request，继承的每个类按 game_type 区分。团队参加请求也是一个用户只能向一个团队发送一个请求，并且继承自 team_request。

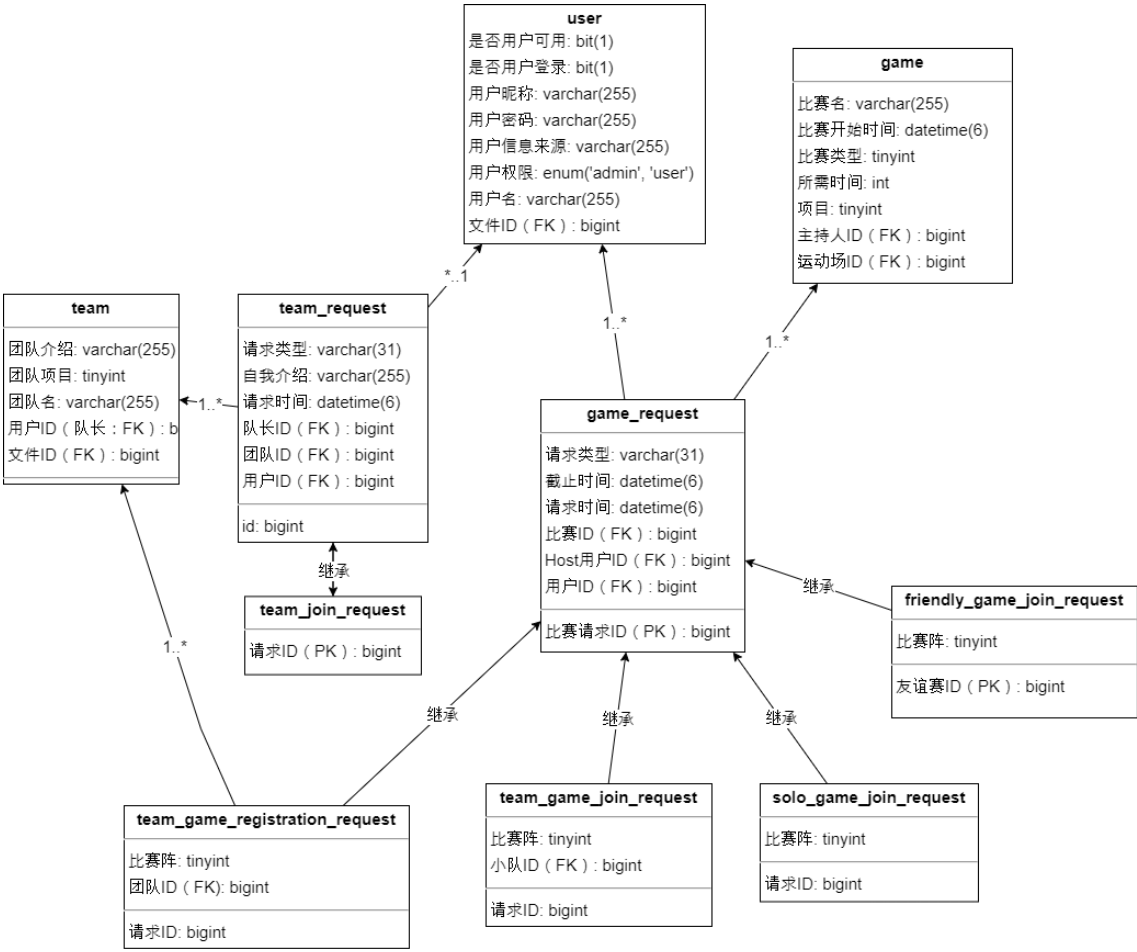


图 3-25 请求实体关系图

表 3-12 比赛请求表

字段名称	类型	空否	说明
------	----	----	----

id	bigint	否	主键
type	varchar	否	
expired_time	datetime	否	
request_time	datetime	否	
game_id	bigint	否	外键
host_id	bigint	否	外键
user_id	bigint	否	外键

表 3-13 团队请求表

字段名称	类型	空否	说明
id	bigint	否	主键
type	varchar	否	
introduction	datetime	否	
request_time	datetime	否	
game_id	bigint	否	外键
host_id	bigint	否	外键
user_id	bigint	否	外键

3.3.5 数据库总体设计

图 3-26 为整个数据库架构图，详细展示了一个基于 Web 技术的学校运动场管理系统的数据库设计。该设计涵盖了多个实体和关系，包括用户（User）、团队（Team）、运动场（Playground）、比赛（Game）等核心组件。每个实体都通过具体的属性来定义，显示了系统如何存储和关联数据。

在此数据库模型中，运用了多种关系型数据库的特性^[19]，如外键（Foreign Key）关联和实体的主键（Primary Key）定义。例如，比赛结果（Game_Result）与运动场关联，团队与比赛通过队伍请求（Team_Request）相连。此外，系统支持复杂查询，如根据用户的参与查询比赛结果，或是团队的所有比赛记录。整个数据库设计旨在优化查询效率和数据完整性，确保系统运行的高效和稳定。

第 4 章 学校运动场系统实现

本章详细阐述了学校运动场系统的实现，包括用户信息管理、运动场信息管理、比赛管理和团队管理等核心功能。用户信息管理功能允许用户查看和更新个人资料，参与比赛管理。运动场信息管理提供了查看和添加运动场的功能，使用户能够轻松获取运动场的详细信息。比赛管理功能则包括比赛的创建和结果记录，支持用户有效地组织和跟踪比赛活动。团队管理功能则涉及到团队的创建、查看和管理。

4.1 用户信息管理功能实现

4.1.1 用户信息功能实现

用户信息页面包含用户的昵称，个人头像照片和战绩，以及一个可以更新战绩的更新按钮。点击昵称更改按钮后，会弹出一个可以输入新昵称的输入框，点击确认后会进行重复检查，然后可以更改昵称。存在一个可以退出的按钮，还可以查看当前所属团队的完整列表。图 4-1 为用户页面。



图 4-1 用户信息界面

4.1.2 用户比赛管理功能实现

用户当前参与的比赛可以在比赛 Menu 中选择。所参与的比赛会展示该月的比赛，已经结束的比赛显示在左侧，尚未开始的比赛显示在右侧。显示比赛的名称、时间、比赛类型和比赛开始时间。图 4-2 所示用户参与的比赛界面。

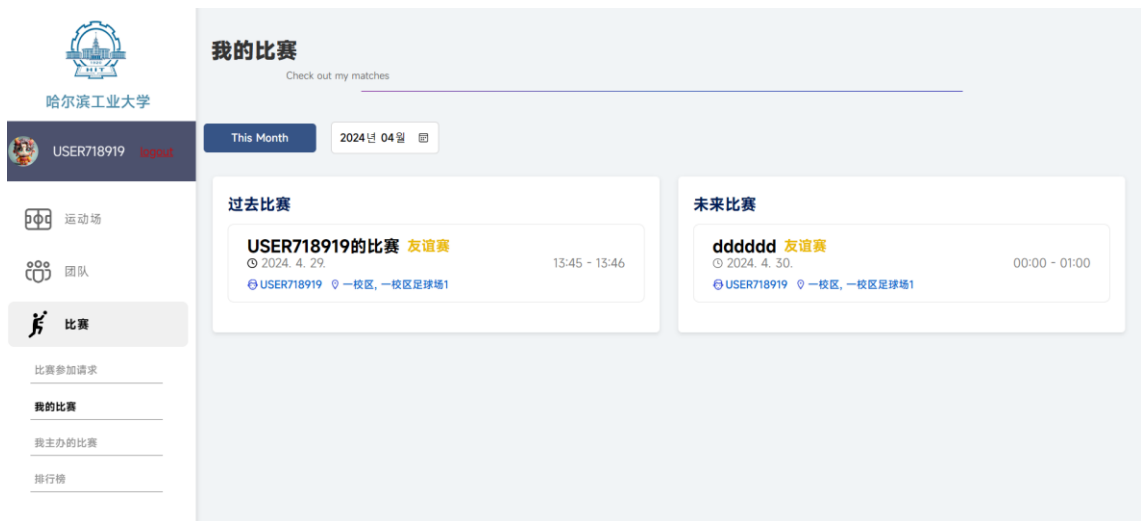


图 4-2 用户参与的比赛界面

图 4-3 显示用户主办的比赛界面。尚未开始的比赛显示在上方，已经结束的比赛显示在下方。最初显示所有比赛，用户可以选择年份和月份，以查看当月主办的比赛。

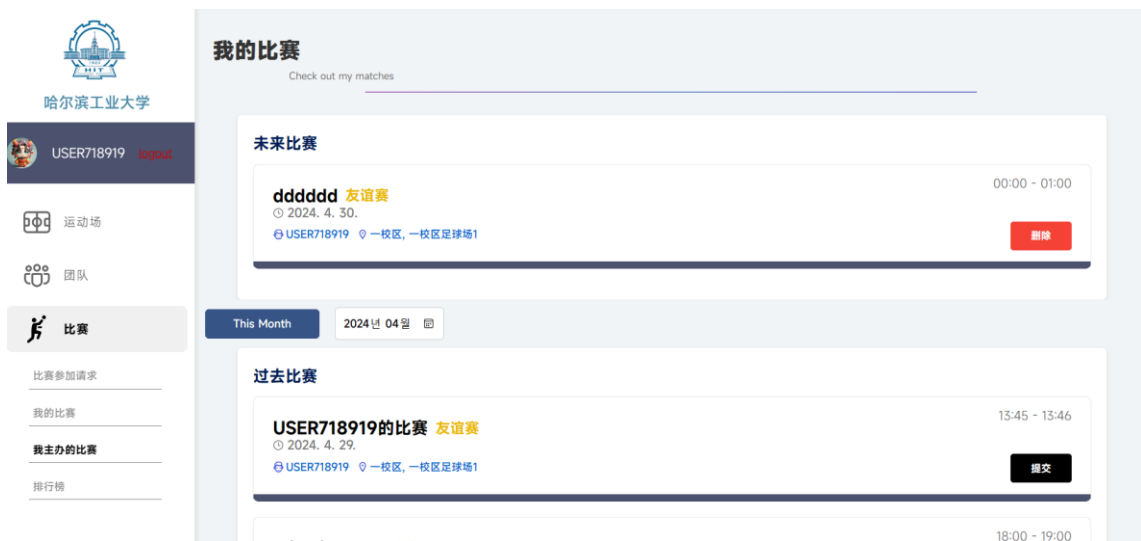


图 4-3 用户主办的比赛界面图

4.2 运动场信息管理功能实现

4.2.1 运动场查看功能实现

运动场信息页面可以根据项目选择查看学校所有运动场或按校区查看。选择项目后，图 4-4 所示可以在顶部看到该项目即将开始的比赛信息，以及该项目的运动场列表。在运动场列表中，可以看到运动场信息和即将开始的比赛数量。点击运动场即可进入运动场信息页面。图 4-5 为运动场信息列表界面。



图 4-4 即将开始的比赛界面图

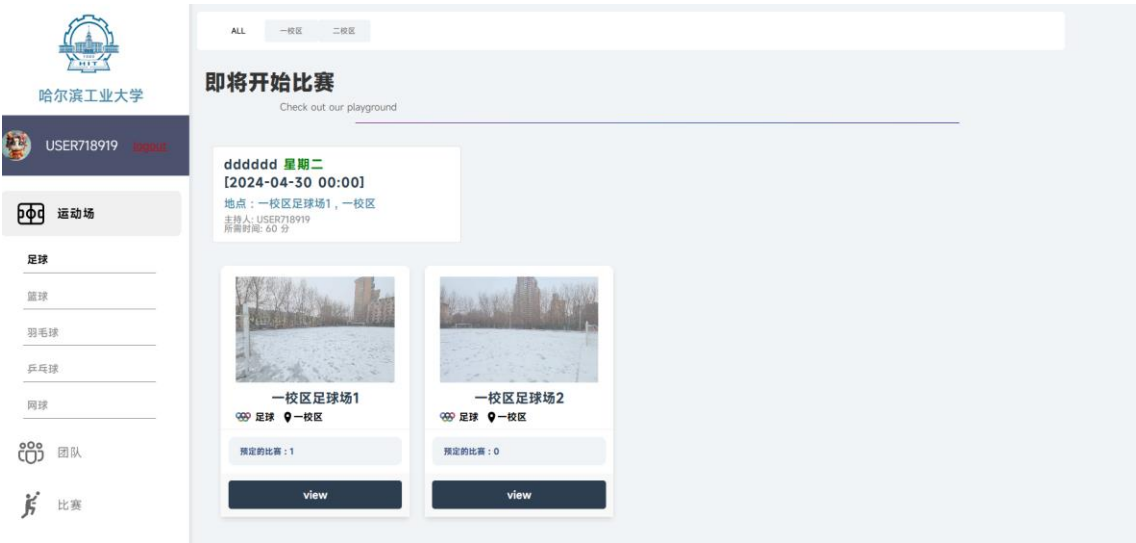


图 4-5 运动场信息界面图

4.2.2 运动场比赛查看功能实现

当点击运动场时，如下图 4-6 所示，显示运动场的头像以及当前进行中的比赛，未来将开始的比赛信息显示在右侧。点击即将开始的比赛，可以看到参赛的选手和团队。

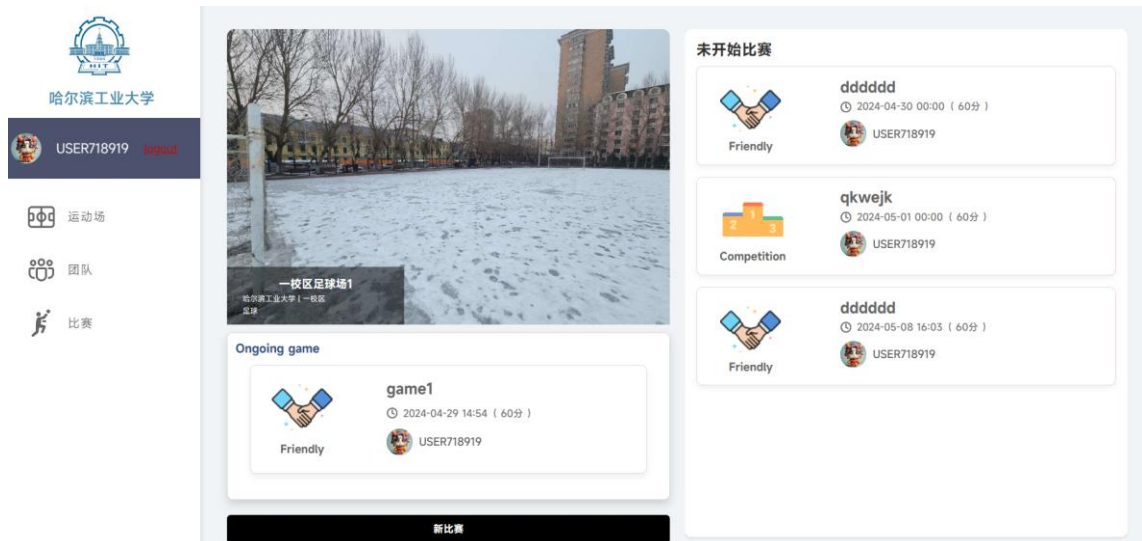


图 4-6 运动场内比赛列表

4.2.3 运动场添加功能实现

系统管理员可以添加要运营的运动场。在界面上选择运动场的头像，设置运动场的名称，然后选择项目后，可以添加运动场。图 4-7 所示添加运动场的界面。



图 4-7 添加运动场界面

4.3 比赛管理功能实现

4.3.1 比赛创建功能实现

图 4-8 所示，用户可以创建比赛。比赛名称需要用户填写，项目会根据该运动场的项目自动选择。点击比赛开始时间，可以指定比赛开始时间。图 4-9 所示。比赛开始时间可以选择年月日时分，不能选择过去的时间。选择日期后，会显示当天已存在的比赛时间，不能选择与现有比赛时间重叠的时间。选择比赛时间后，可以选择相应比赛的类型并创建比赛。

创新比赛

请输入比赛事详情

比赛名

game1

项目

足球

开赛时间

2024 年 04 月 26 日 15:20

☒ 友谊赛

☐ 竞赛赛

Cancel

Save

图 4-8 创建比赛页面

选择时间

请输入比赛开始时间

2024-04-24

⌚ 开始时间

⌚ 16:16

⌚ 所需时间

-

60

+

取消

确定

占用的时隙

Select the start time and duration for your participation.

⌚ 14:37 ~ 15:37

⌚ 20:13 ~ 21:13

图 4-9 选择比赛时间页面

- 50 -

4.3.2 比赛结果功能实现

主办比赛的用户可以查看自己主办的比赛。图 4-10 为用户主办的比赛。可以查看尚未开始的比赛和该月的过去比赛。对与尚未开始的比赛可以删除该比赛，对于已经结束的比赛可以输入结果。如图 4-11 所示，输入比分后提交。



图 4-10 用户主办的比赛页面



图 4-11 输入比赛结果的页面

4.4 团队管理功能实现

4.4.1 团队创建功能实现

用户可以在创建团队界面中创建自己的团队。首先选择团队的头像照片，然后输入团队名称和选择团队的运动项目。这三项是必须输入的。最后，可以添加团队的介绍。图 4-12 所示创建团队的页面。

建立团队

组队



点击换头像

团队名

项目



队长



USER718919

介绍

确定

图 4-12 创建团队页面

4.4.2 团队管理功能实现

图 4-13 所示团队信息和管理界面。团队信息页面展示了团队名称、团队头像、团队介绍以及团队成员。在团队右侧有一个按钮，如果是队长，可以删除团队，如果是队员，可以退出团队，如果是未加入团队的用户，可以发送加入申请。

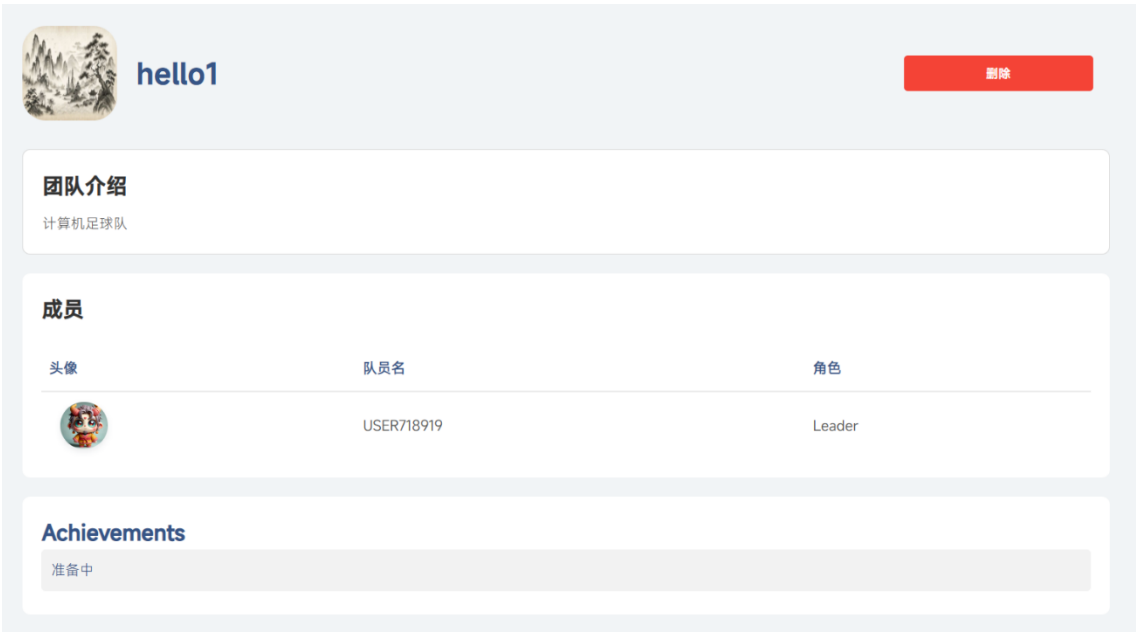


图 4-13 团队管理界面

4.4.3 团队查看功能实现

图 4-14 所示团队列表界面。显示当前存在的团队列表。最初显示了所属学校的所有团队列表，点击每个项目的按钮，可以只显示属于该项目的团队。团队只展示头像和名称，点击即可进入该团队信息页面。

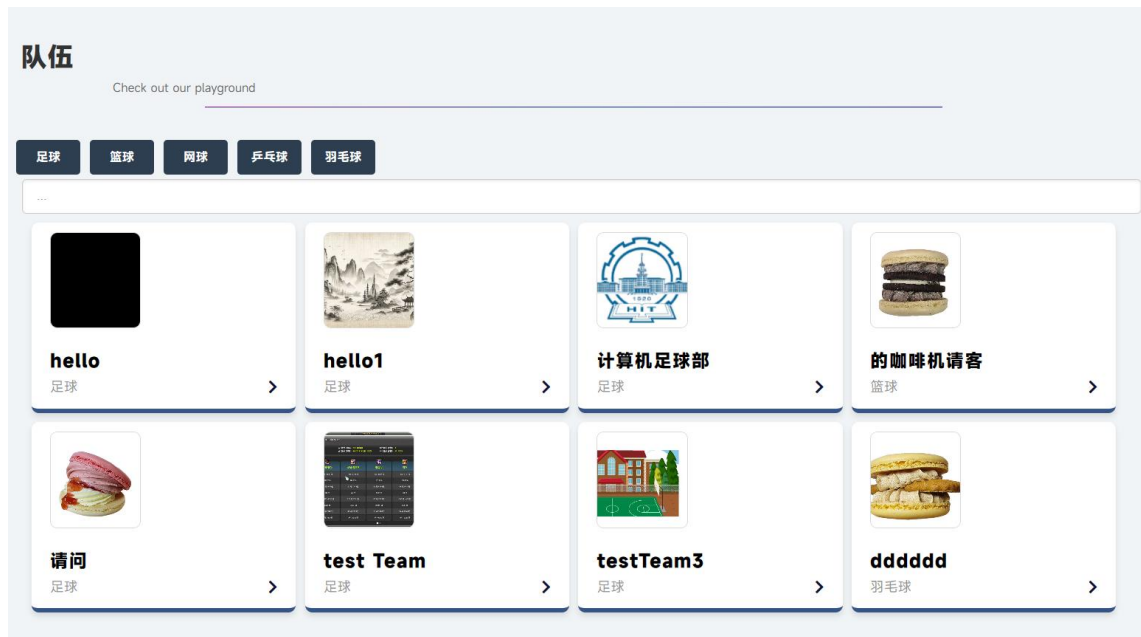


图 4-14 团队列表界面

4.5 请求管理功能实现

4.5.1 请求发送功能实现

各种请求分为友谊赛参与请求、竞争比赛参与请求和团队参与请求。友谊赛参与请求是用户想要参加友谊赛时可以通过点击按钮来发送的，由于友谊赛不涉及团队，因此只发送一次请求。图 4-15 所示友谊赛请求界面图。



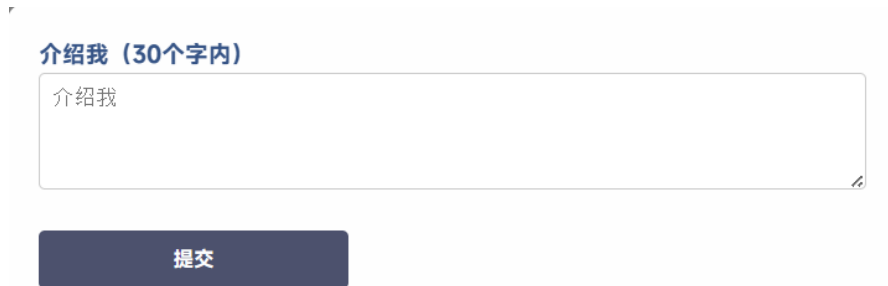
图 4-15 友谊赛请求界面图

竞争比赛请求，参与者可以选择加入主队或客队。在该团队内，参与者既可以以个人参与，也可以创建自己的小团队发送请求，或者如果已有小团队存在，团队成员可以向该小团队发送加入请求。竞技比赛请求由比赛主办方管理。图 4-16 所示竞争赛请求界面图。



图 4-16 竞争赛请求界面图

团队参与请求可以在相应的团队页面发出。发送团队参与请求时，可以附带一份不超过 30 个字符的个人简介。图 4-17 所示团队参与请求界面图。

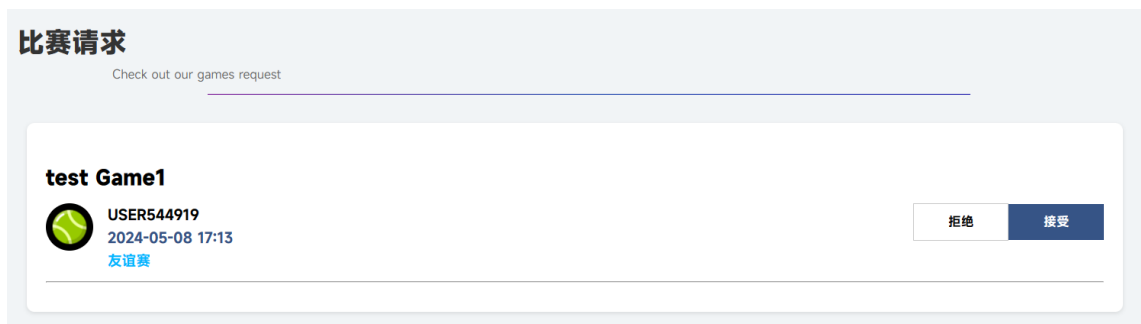


The form consists of a title '介绍我 (30个字内)' in blue, followed by a text input field with the placeholder '介绍我'. Below the input field is a dark blue button with the white text '提交'.

图 4-17 团队参与请求界面图

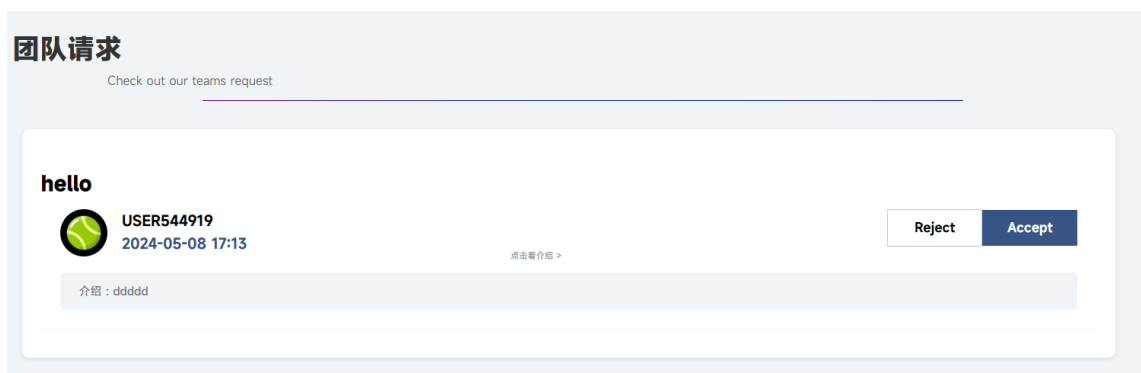
4.5.2 请求查看功能实现

比赛主办方可以查看所有关于比赛的请求，每个请求包括比赛名称、请求者的个人资料信息、请求发送时间以及比赛类型。比赛主办方可以选择接受这些请求或拒绝并删除请求。图 4-18 所示比赛请求查看界面图。团队请求包括团队的名称、请求者的信息、请求时间以及请求者的个人简介。图 4-19 所示团队参与请求查看界面图。



The interface has a header '比赛请求' with a subtitle 'Check out our games request'. Below is a card for 'test Game1' featuring a green circular icon, the text 'USER544919', the timestamp '2024-05-08 17:13', and the category '友谊赛'. To the right of the card are two buttons: '拒绝' (Reject) and '接受' (Accept).

图 4-18 比赛请求查看界面图



The interface has a header '团队请求' with a subtitle 'Check out our teams request'. Below is a card for 'hello' featuring a green circular icon, the text 'USER544919', the timestamp '2024-05-08 17:13', and a link '点击看介绍 >'. Below the card is a text input field with the placeholder '介绍 : ddddd'. To the right of the card are two buttons: 'Reject' and 'Accept'.

图 4-19 团队参与请求查看界面图

第 5 章 学校运动场系统测试

5.1 系统性能测试

在本节中，将介绍使用开源性能测试工具 JMeter 进行的网络和应用性能测试过程。测量性能并生成详尽的测试报告。

5.1.1 测试环境

服务器配置见表 5-1 所示。

表 5-1 服务器配置

名称	环境
操作系统	Window 10
CPU	Intel i-7 1165G
内存	16GB
硬盘	512GB

5.1.2 性能测试

JMeter^[18]是开源性能测试工具，专为网络和应用性能测试设计。其支持多种协议，如 HTTP、FTP、JDBC 等，可模拟各种请求，分析结果并生成详细报告。每个测试会测量发送 HTTP 并发请求到收到响应所平均时间。测试会测量在 1 秒内由 1000 个线程发送每个请求时的平均响应时间。每个测试进行 4 次，总共发送了 4000 个请求。

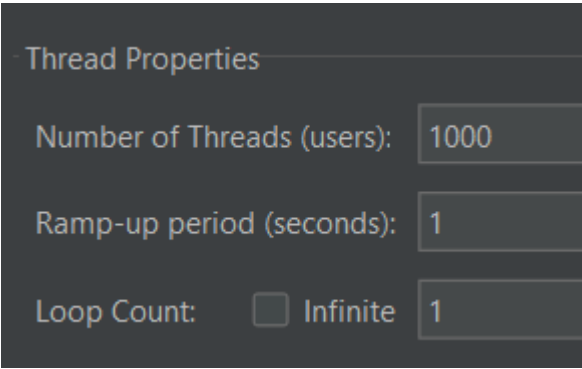


图 5-2 并发测试配置

以下是系统主要功能的 GET 或 POST 请求的响应时间测量。

(1) 运动场和比赛信息页面

图 5-3 所示，最短时间为 3ms，最长时间为 931ms，平均时间为 96ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	4000	96	3	931	153.65	0.00%	323.8/sec	162.21	125.53	513.0
TOTAL	4000	96	3	931	153.65	0.00%	323.8/sec	162.21	125.53	513.0

图 5-3 运动场信息和比赛信息页面测试结果图

(2) 团队列表页面

图 5-4 所示，最短时间为 30ms，最长时间为 3311ms，平均时间为 1389ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	4000	1389	30	3311	738.55	0.00%	221.1/sec	430.29	94.13	1993.0
TOTAL	4000	1389	30	3311	738.55	0.00%	221.1/sec	430.29	94.13	1993.0

图 5-4 团队列表页面测试结果图

(3) 用户主办的比赛页面

图 5-5 所示，最短时间为 21ms，最长时间为 2525ms，平均时间为 1025ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	4000	1025	21	2525	516.45	0.00%	239.7/sec	658.10	84.05	2811.0
TOTAL	4000	1025	21	2525	516.45	0.00%	239.7/sec	658.10	84.05	2811.0

图 5-5 用户主办的比赛页面测试结果图

(4) 主页面

图 5-6 所示，最短时间为 26ms，最长时间为 3234ms，平均时间为 1528ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	5000	1528	26	3234	821.32	0.00%	178.0/sec	258.52	62.93	1487.0
TOTAL	5000	1528	26	3234	821.32	0.00%	178.0/sec	258.52	62.93	1487.0

图 5-6 主页面测试结果图

(5) 运动场列表页面

图 5-7 所示，最短时间为 23ms，最长时间为 2947ms，平均时间为 1313ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	4000	1313	23	2947	657.53	0.00%	201.3/sec	282.25	72.53	1436.0
TOTAL	4000	1313	23	2947	657.53	0.00%	201.3/sec	282.25	72.53	1436.0

图 5-7 运动场列表页面测试结果图

（6） 排行榜页面

图 5-8 所示，最短时间为 20ms，最长时间为 2775ms，平均时间为 1004ms。

Label	# Samples	Average	Min	Max	Std. Dev. ↑	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Reque...	4000	1004	20	2775	548.83	0.00%	264.6/sec	148.05	100.51	573.0
TOTAL	4000	1004	20	2775	548.83	0.00%	264.6/sec	148.05	100.51	573.0

图 5-8 排行榜页面测试结果图

测试中 1000 次并发请求的平均响应时间约为 1 秒。起初响应时间超过 2 秒，因为在验证 JWT Token 的过程中，需要从数据库中查询用户信息。识别到这个问题后，我们将用户信息直接存储在 JWT Token 中，而不是查询数据库，这带来了显著的时间缩短。超过 2000 次的并行请求时，平均响应时间再次超过 2 秒，因此发现有必要进一步优化数据库访问过程的时间。

结论

在系统设计与实现方面，采取了模块化的开发策略，将系统分为多个独立但协作的模块。这些模块包括用户管理、场地管理、比赛管理和安全管理等，每个模块负责处理特定的功能需求。这种设计使得系统具有高度的灵活性和可扩展性，便于未来的功能添加和系统升级。

在前端，通过 Vue.js 实现，能够自适应各种屏幕尺寸和设备。Vue.js 的组件化特性极大地提高了前端代码的重用性和维护性。同时，利用 Vue Router 进行页面路由管理，实现了单页面应用（SPA）的流畅体验。

在后端，使用 Spring Boot 框架简化了企业级应用开发，通过依赖注入和面向切面编程减少了组件之间的耦合。利用 Spring Data JPA 简化了数据访问层的代码，自动化地处理了数据库交互，提高了开发效率并减少了潜在的错误。

安全方面，系统集成了 JWT 和 OAuth2，实现了基于 Token 的身份验证和授权。通过 OAuth2，系统支持第三方登录，增强了用户的便捷性和系统的开放性。所有的数据传输都采用 HTTPS 协议，确保了传输过程中的数据安全。

数据库设计方面，采用了关系型数据库管理系统，并对数据模型进行了规范化设计，以支持系统的高效运行和扩展。

系统实现部分包括用户信息管理、运动场信息管理、比赛管理等功能模块的详细实现过程。此外，确保了系统的易用性。整个系统的开发和实现过程充分体现了现代 Web 技术的强大功能和高效性。

总之，通过这些设计和实现策略，本研究成功地构建了一个高效、安全且用户友好的学校运动场管理系统，满足了现代教育机构对体育设施管理的需求。

参考文献

- [1] A. Shahi, A. Barge, A. Butala and S. Patil, "Rest API based Web Interface for Blogging Application," 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2023, pp. 673-677, doi: 10.1109/ICSCSS57650.2023.10169662.
- [2] A. Soni and V. Ranga, "API Features Individualizing of Web Services: REST and SOAP", Int. J. of Innovative Technol. and Exploring Eng., vol. 8, no. 9S, pp. 664-671, 2019.
- [3] M. Melnichuk, Yu. Kornienko and O. Boytsova, "Web-Service. Restful Architecture", Automat. Technological and Business Process, vol. 10, no. 1, 2018.
- [4] J. Bradley, N. Sakimura and M. B. Jones, "JSON Web Token (JWT)", [online] Available: <https://tools.ietf.org/html/rfc7519>.
- [5] S. I. Adam, J. H. Moedjahedy and J. Maramis, "RESTful Web Service Implementation on Unklab Information System Using JSON Web Token (JWT)," 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), Manado, Indonesia, 2020, pp. 1-6, doi: 10.1109/ICORIS50180.2020.9320801.
- [6] S. Chen, Z. Mao, Y. -M. Wang and M. Zhang, "Pretty-Bad-Proxy: An Overlooked Adversary in Browsers' HTTPS Deployments," 2009 30th IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2009, pp. 347-359, doi: 10.1109/SP.2009.12.
- [7] 欧黎源, 邱会中, 白亚茹. 基于 JPA 的数据持久化模型设计与实现[J]. 计算机工程, 2009, 35(20): 76-77+80.
- [8] A. P. Aldya, A. Rahmatulloh and M. N. Arifin, "Stateless Authentication with JSON Web Tokens using RSA-512 Algorithm", *Jurnal INFOTEL*, vol. 11, no. 2, pp. 36-42, Jun. 2019.
- [9] 范展源, 罗福强. JWT 认证技术及其在 WEB 中的应用[J]. 数字技术与应用, 2016(02): 114. DOI: 10.19695/j.cnki.cn12-1369.2016.02.087.
- [10] 刘金秀, 陈怡华, 谷长乐. 基于 Nginx 的高可用 Web 系统的架构研究与设计[J]. 现代信息科技, 2019(11): 94-97.
- [11] You, Evan. Vue.js. Vue.js Developers. <https://vuejs.org/>.

- [12] 张向征. 设计模式在 Web 系统开发中的应用研究[D]. 兰州大学, 2006.
- [13] Riaan Nel. "Design Patterns: The Strategy and Factory Patterns." (n.d.). DZone. Retrieved from Design Patterns: The Strategy and Factory Patterns - DZone
- [14] Efficient Backend Development with Spring Boot: A Comprehensive Overview. (n.d.). International Journal for Research in Applied Science & Engineering Technology (IJRASET). Retrieved from www.ijraset.com
- [15] Lee, Y.-H. (2019). Study on Security Enhancement of Android Applications Using Spring Security and OAuth 2.0. Journal of Information Security, 25(3), 67-78. DOI: 10.2345/6789012345
- [16] Controller Advice. Retrieved from Controller Advice :: Spring Framework.
- [17] Javatpoint. (n.d.). JPA Introduction. Retrieved from <https://www.javatpoint.com/jpa-introduction>
- [18] Apache Software Foundation. (2024). JMeter Documentation. Retrieved from <https://jmeter.apache.org/usermanual/index.html>
- [19] Harrington, J. L. (n.d.). Relational Database Design and Implementation (4th ed.). O'Reilly Media. Retrieved from <https://www.oreilly.com>

致 谢

衷心感谢导师骆功宁教授对本人的精心指导。即使在我因事故突然休学的情况下，教授您仍然长时间地给予我指导，细心地指导我关于多篇论文的事宜，对此表示衷心的感谢。

感谢朱占仕老师，在我对论文方向感到困惑时，您帮我正确把握了多种方向，并帮助我更好地完成写作，对此表示感谢

