****

**本科毕业论文（设计）中期报告**

**题 目：**

**专 业**

**学 生**

**学 号**

**指导教师**

**日 期**

**哈尔滨工业大学教务处制**

一、论文（设计）工作是否按开题报告预定的内容及进度安排进行

开题报告中预定了如下研究内容：

运用现今广泛应用的技术资源，针对学校及学生群体设计并实现增益性网络应用。在此过程中，着重于提高系统的可扩展性，采用结构优化的编码策略，并进行代码重构，以期达到更高效的架构设计和实现。

（1）考虑提供哪些服务, 聆听学生面临的挑战与需求

（2）设计关系型数据库，注重效率和扩展性，进行精细化设计。

（3）编写单元测试，包括正确和错误的测试案例，以提高代码的准确性。

（4）编写并执行满足各种条件的功能，并监控其是否正常运行。为遵循面向对象设计，进行代码重构。

（5）通过开发用户界面来实现可实际服务的网络服务。分析请求数据，设计更高效的数据库索引，以提升数据检索效率。

目前毕业设计已按照开题报告预定的进度完成了要提供的服务，设计数据库，编写主要功能，并监控正常运行，代码重构，设计高校的代码结构

二、已完成的研究工作及成果

## 2.1 软件应用流程

### 2.1.1应用

用户在网上发送HTTP或HTTPS请求时，服务器将直接处理进入443端口的请求，并将进入80端口的HTTP请求重定向到443端口的HTTPS。前端的静态资源通过 / 请求进行处理，后端资源通过映射到/api的请求进行处理。Spring应用程序接收到请求后，处理相应的请求并向用户返回响应。请求流程如下图所示：

스크린샷, 도표, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

图 2-1 网站应用结构图

도표, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

图 2-2 网站应用技术栈图

### 2.1.2 登记和登录流程

登录过程通过第三方登录进行。用户点击登录按钮后，服务器会重定向登录请求到指定的认证页面，用户进行登录。登录成功，第三方服务器向服务器回应授权码，服务器随后向第三方服务器发送含该授权码的accessToken获取请求。若Code有效，则返还access Token，服务器再发送包含access Token的用户信息获取请求。若Access Token有效，第三方服务器会回应用户信息。服务器接收用户信息后，若数据库已存在该用户，则更新用户；若不存在，则存储用户信息。存储后，服务器使用JwtTokenProvider加密生成新的access\_token及refresh\_token回应用户。此时用户将access\_token与refresh\_token存储于本地

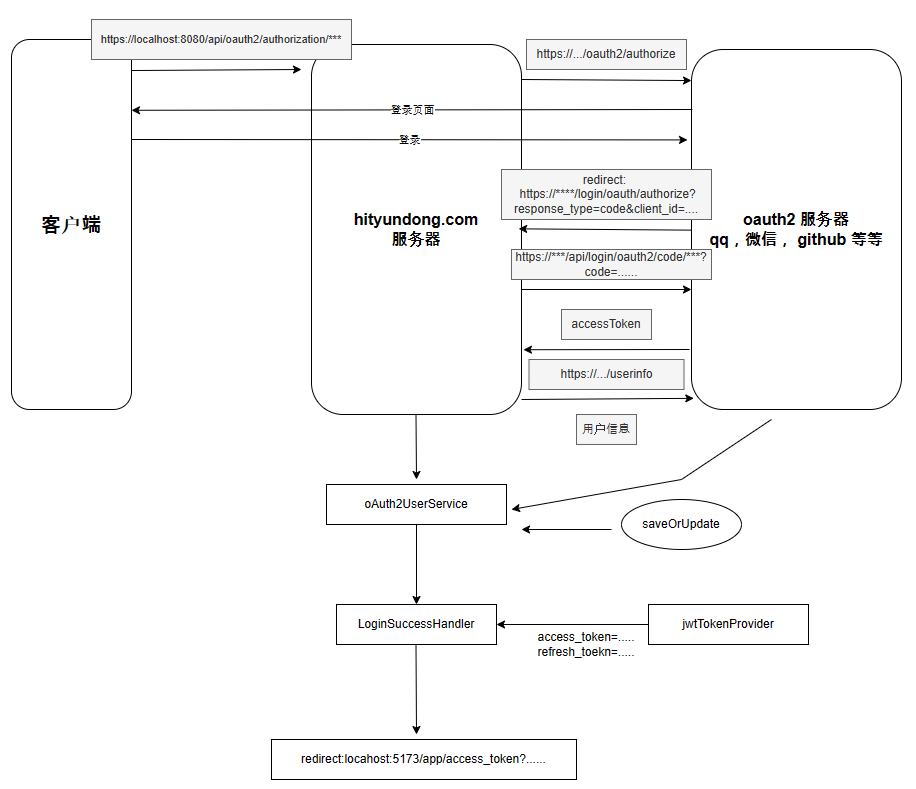


图 2-3 登记和登录流程图

### 2.1.3 请求过程

如果用户持有的访问令牌（access\_token）有效，则Jwt身份验证过滤器（JwtAuthenticationFilter）的令牌提供程序（JwtTokenProvider）将验证访问令牌的有效性。

若验证有效，则会在安全上下文（SecurityContext）中保存认证对象（Authentication对象），并通过过滤器（Filter）向控制器（controller）提交请求。

如果用户持有的访问令牌无效，但刷新令牌（refresh\_token）有效，则刷新令牌过滤器（JwtRefreshTokenFilter）的令牌提供程序将验证刷新令牌的有效性。若验证有效，令牌提供程序将发放新的令牌并返回给用户。

如果访问令牌和刷新令牌都无效，则Jwt身份验证过滤器不会通过，并返回403响应，并重定向到/login页面。

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

图 2-4 请求流程

## 2.2 代码重构

### 2.2.1 研究目的

在软件开发过程中，重构是提高系统维护性、可扩展性和可读性的重要步骤。本研究解释了比赛创建服务的初步设计如何通过使用策略模式和工厂模式进行重构，以及这种重构如何符合设计原则。具体来说，详细描述了比赛创建逻辑中固有的决策结构如何简化，并转换为增强了灵活性和可扩展性的设计 服务需要管理复杂和多变的业务逻辑，以支持不同类型的比赛。初步设计采用了在比赛类内部使用条件语句根据比赛类型来分支创建逻辑的方法。这种方法表明，每当添加新的比赛类型时，都需要修改类，从而限制了可扩展性和可维护性。为了克服这些限制，引入了策略模式和工厂模式进行重构。

### 2.2.2创建比赛功能重构前设计的问题

结合度增加：GameService 类型需要知道要创建的比赛类型，每当添加新的比赛类型时，都需要修改此类型。

可扩展性不足：添加新的比赛类型或更改现有逻辑时，需要对相关代码进行全面更改。

违反单一职责原则：GameService 类型负责处理多种比赛创建逻辑，承担了多个职责

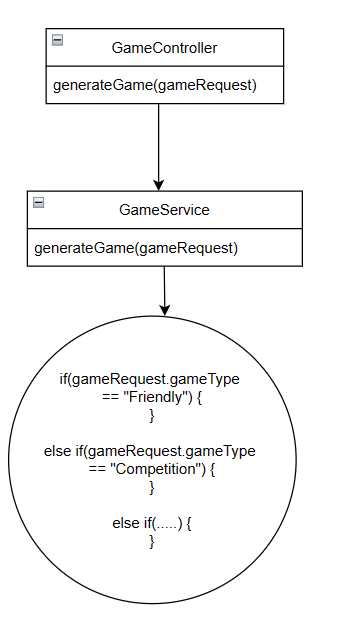


图2-5 重构前的创建比赛设计图

### 2.2.3 重构后的设计

工厂模式和策略模式的应用带来了以下优点：

灵活的创建：GameGeneratorFactor根据比赛类型生成相应的GameGenerator对象。

策略的可替换性：生成逻辑通过GameGenerator接口抽象化，可以用FriendlyGameGenerator和CompetitionGameGenerator等具体策略替换。

单一责任原则：每种比赛类型的生成逻辑被封装在相应的策略对象内，从而分离了GameService的责任。

开放/封闭原则：添加新的比赛类型时，不需修改现有代码，只需添加新的GameGenerator实现即可。

重构后的设计结构如下图:

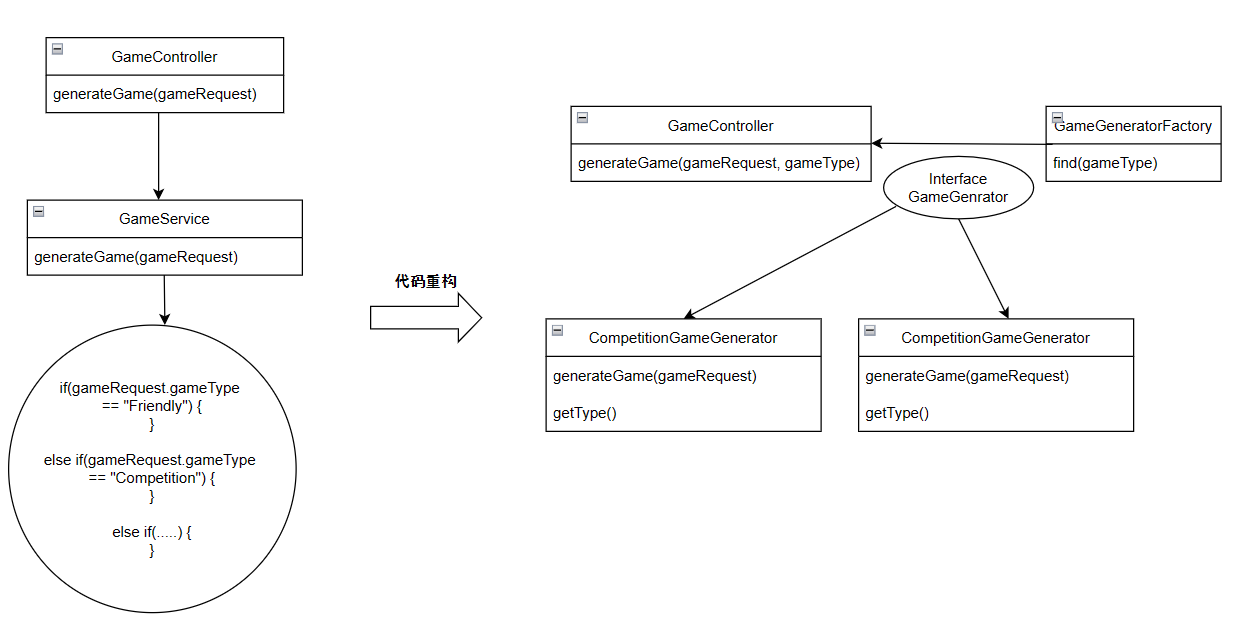


图2-6 GameGenerator代码重构图

### 2.2.4 请求管理功能设计

#### 2.2.4.1 重构前设计的局部性

在初始设计中，为了处理TeamGameJoinRequest和TeamJoinRequest，RequestService中使用了基于条件语句的逻辑，这带来了以下限制：

结合度：所有请求处理逻辑都绑定在一个类型中，添加新的请求类型时需要修改现有类型。

维护困难：处理不同请求类型的逻辑混合在一起时，使代码理解和修改变得困难。

缺乏可扩展性：每当添加或更改新的请求类型时，都需要对现有代码进行大量修改，

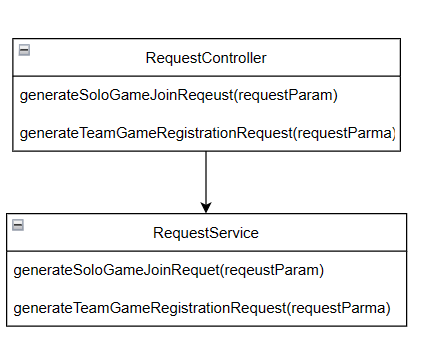


图2-7 重构前的请求功能设计图

#### 2.2.4.2 请求功能代码重构

通过引入RequestManagerFactory和RequestManager接口来解决这些问题。通过重构，将每个请求类型分离成单独的类型，并应用工厂模式，动态创建适当的请求处理器，改变了结构。本研究主要分析了该重构如何满足软件设计原则，特别是开放/封闭原则和单一责任原则。

在重构过程中，应用了以下变化：

引入工厂模式：使用RequestManagerFactory根据请求类型动态生成相应的处理器（RequestManager）。

应用策略模式：为每种请求类型创建对应的单独类（TeamGameJoinRequestManager，TeamJoinRequestManager），这些类型都实现了RequestManager接口。

重构的优点分析:

重构后的系统获得了以下优势：

减少结合度：引入RequestManagerFactory后，即使添加新的请求类型，也无需修改RequestService。

提高可维护性：通过将每种请求类型的特定逻辑分离到不同的类型中，清晰了每个类型的职责，并提高了代码的可读性。

增加扩展性：新的请求类型的添加对现有系统的影响最小化，并且通过仅添加新类型就可以轻松扩展请求类型。

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

图2-8 重构后的请求功能设计图

#### 2.2.2.6 重构后实际实例分析

软件设计模式的应用对软件的可扩展性和可维护性有着重大影响。通过实际案例分析策略模式的应用。案例包括最近添加的团队参与功能和友谊赛申请功能。

具体而言，每种比赛申请类型（竞技游戏、友谊赛、团队参与）都被定义为实现GameRequestManager接口的独立类型，依据申请类型可以应用不同的处理方法。

通过采用策略模式，分别为每种比赛申请类型实施了 FriendlyGameReuqest、TeamGameJoinRequest等。这种设计提供了灵活性，即每当新的比赛申请类型被引入时，可以通过添加新类型而无需修改现有代码来满足需求。 团队参与和友好比赛申请功能的加入很好地展示了策略模式的好处。每增加一个功能，无需更改现有代码，仅通过增加新的策略类即可满足需求。这遵循了开放/关闭原则意味着可以在不更改现有代码的情况下扩展系统功能。此外，因为每个比赛申请处理逻辑被分离到不同的类型中，因此增强了可维护性和代码的可读性。

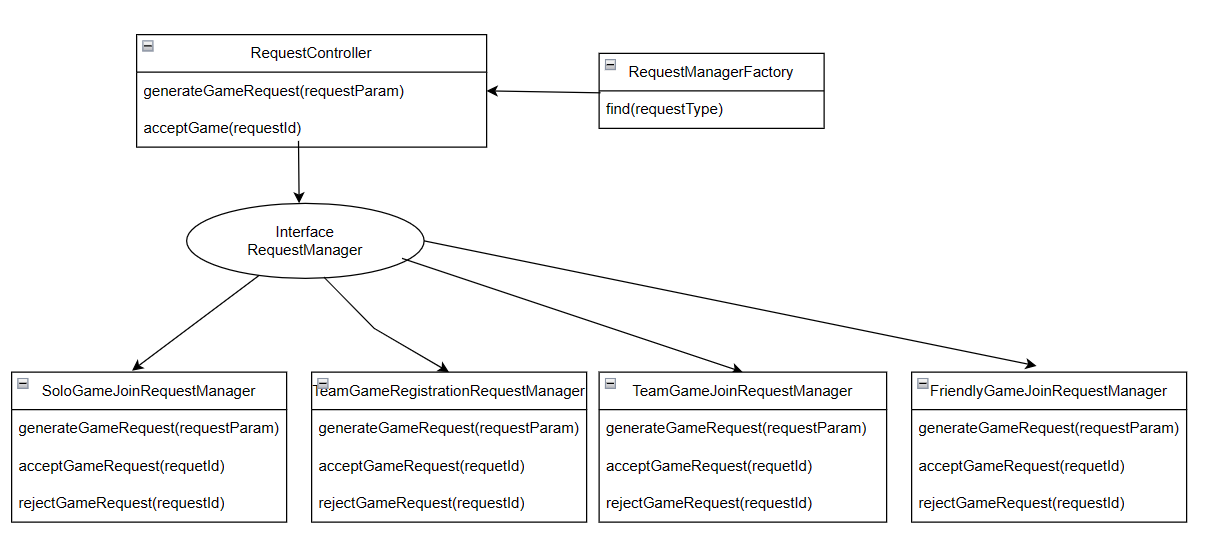


图2-9 重构后实际实例结构图

三、后期拟完成的研究工作及进度安排

四、存在的问题与困难

五、论文按时完成的可能性