<项目名称>

软件设计文档

版本 <1.0>

修订版历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| < 2024/9/28 > | <1.0> | <版本1.0> | <55210808刘海峰 > |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 简介 4](#_Toc86704318)

[1.1 项目 4](#_Toc86704319)

[1.2 目的 4](#_Toc86704320)

[1.3 开发环境 4](#_Toc86704321)

[1.4 工件 4](#_Toc86704322)

[2. 需求分析 4](#_Toc86704323)

[2.1 用例描述和分析类 4](#_Toc86704324)

[2.2 用例实现（顺序图） 4](#_Toc86704325)

[2.3 参与类图 7](#_Toc86704326)

[3. 设计模型 8](#_Toc86704327)

[3.1 子系统与接口 8](#_Toc86704328)

[3.2 类图与类设计 8](#_Toc86704329)

1 介绍

1.1 项目

本项目是一个面向编程爱好者和开发者的在线论坛，旨在为用户提供一个互动平台，帮助用户解决编程中的疑难问题，并分享工作经验和面试题。用户可以通过论坛进行技术交流，发布和解答各种技术问题，并与其他用户共同成长。

1.2 目的

1. 明确项目目标和范围：

- 定义在线论坛的目标、功能和预期用户群体，确保所有团队成员了解项目的核心价值和预期成果。

2. 提供项目结构和架构：

- 描述系统架构，包括前端和后端的技术选型、数据库设计、API接口等，帮助团队理解系统的整体结构。

3. 指导开发过程：

- 为开发团队提供详细的开发规范、编码标准和流程，以确保代码的一致性和可维护性。

4. 促进团队沟通：

- 作为团队成员之间的沟通工具，便于各个角色（如开发人员、设计师、测试人员）之间的信息共享与协作。

5. 风险管理：

- 识别潜在的技术和项目风险，提供应对策略，确保项目按时、按质完成。

6. 用户体验设计：

- 描述用户界面和用户交互的设计思路，确保论坛的可用性和用户体验。

7. 功能需求文档：

- 列出所有功能需求，确保开发团队和利益相关者对预期功能达成共识。

8. 测试与验证：

- 设定测试标准和验证方法，确保最终产品符合设计要求和用户需求。

9. 维护和扩展计划：

- 提供系统维护和未来扩展的建议，确保项目的可持续发展。

10. 文档化和参考：

- 为后续的维护和开发提供参考文档，方便新成员快速上手项目。

1.3 开发环境

为了确保项目的开发、测试、部署过程高效且稳定，本编程论坛项目选用了以下开发环境和工具：

1.编程语言

前端框架：Html js Css

后端框架：Spring Boot 2.x jdk17

2.开发工具与集成环境

IDE：

后端开发：IntelliJ IDEA

前端开发：Visual Studio Code

构建工具：

后端：Maven

前端：npm / Yarn（用于包管理和项目构建）

版本控制：Git

托管平台：GitHub 或 GitLab（用于代码管理和团队协作）

3.数据库

数据库类型：MySQL 8.x

ORM框架：MyBatis（用于后端数据持久化）

数据库管理工具：Navicat 或 DBeaver

4.服务器与部署环境

应用服务器：Apache Tomcat 9.x（内嵌于Spring Boot）

Web服务器：Nginx（用于负载均衡和静态资源的处理）

操作系统：Linux（开发和生产环境都使用Linux，如Ubuntu或CentOS）

5.其他工具

依赖管理：

后端：Spring Boot Starter（用于简化Spring依赖管理）

前端：Vue CLI 或 Vite（用于前端项目的初始化和配置）

容器化：Docker（用于开发环境的容器化部署，确保开发环境与生产环境一致）

6.调试与测试

单元测试框架：JUnit（用于后端功能的单元测试）

前端测试工具：Jest、Cypress（用于前端单元测试与集成测试）

API测试工具：Postman（用于后端API测试）

性能测试：JMeter（用于性能测试，确保系统能处理高并发请求）

开发环境配置

1.后端开发环境配置：

安装并配置Java 17 SDK

设置Maven构建工具

使用Spring Boot项目模板启动后端开发

MySQL数据库的本地安装与配置

2. 前端开发环境配置：

安装Node.js及npm（或Yarn）

2 需求分析

2.1用例描述和分析类

用例“发布需求”的基本流程内容：

1.准备发布需求：用户通过登录论坛账号，进入“发布需求”页面。

2.填写需求信息：用户在输入框中输入需求标题、需求描述，并选择合适的分类标签（如“编程问题”“项目协助”等）。

3.检查信息：系统检查用户输入的信息，确保标题和描述不为空且符合格式要求。

4.确认发布：当用户点击“发布”按钮后，系统再次确认需求信息是否完整且符合发布规则。

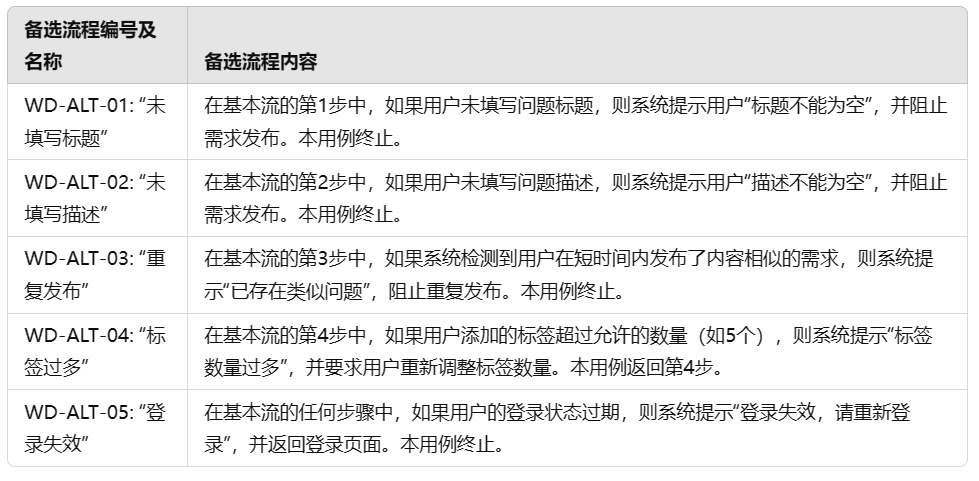
5.发布成功：系统将需求发布到相应的需求列表中，并显示给所有用户。

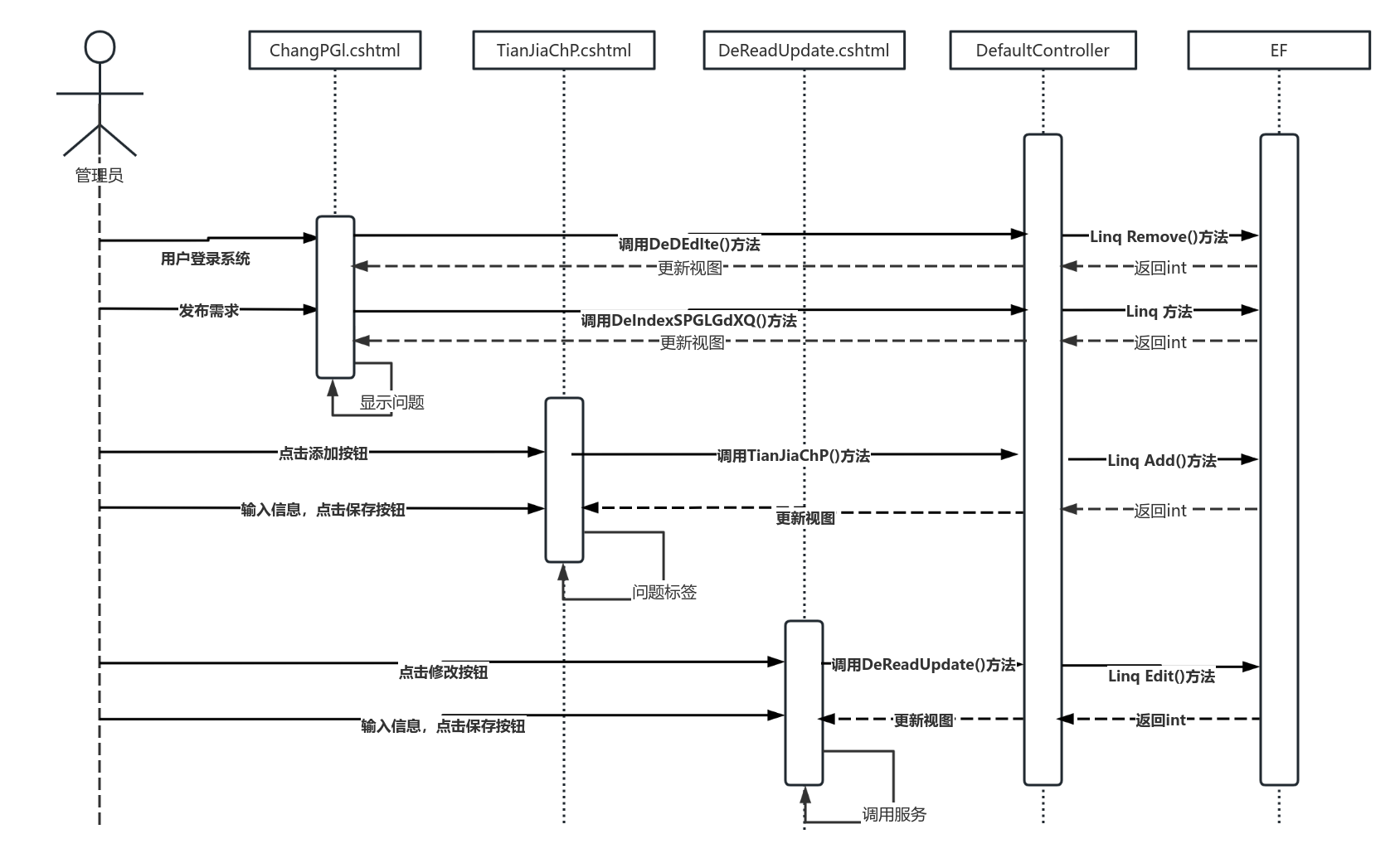
6.通知用户：系统通知用户发布成功，并引导用户查看自己发布的需求。

7.需求展示：其他用户可以在论坛页面查看并响应该需求，系统根据需求的标签和内容进行展示排序。

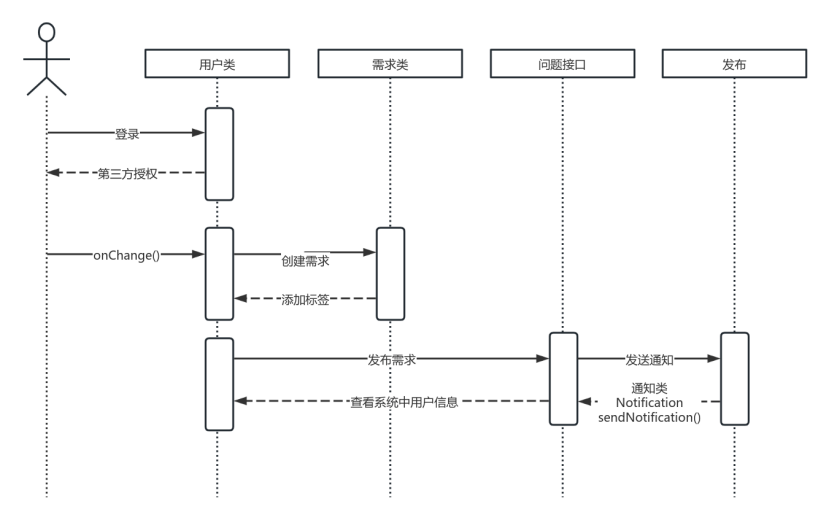
8.更新记录：系统将此需求信息存储在数据库中，并更新相关日志记录。

|  |  |
| --- | --- |
| 参与对象类型 | 用例“论坛发布”的参与对象 |
| 用例本身 | 论坛发布 |
| 常用名词 | 注册、登录、发布需求、评论需求、添加标签发送通知 |
| 现实世界实体 | Csdn |
| 现实世界过程 | 用户注册流程、用户登录流程、需求发布流程、评论添加流程、标签管理流程、通知发送流程 |
| 应用领域术语 | 用户、需求、评论、标签、通知 |
| 分析类的类型 | 用例“论坛发布”的分析类 |
| 实体类 | 用户信息、需求信息、评论信息、标签信息 |
| 边界类 | 用户界面、系统接口 |
| 控制类 | 论坛控制器 |

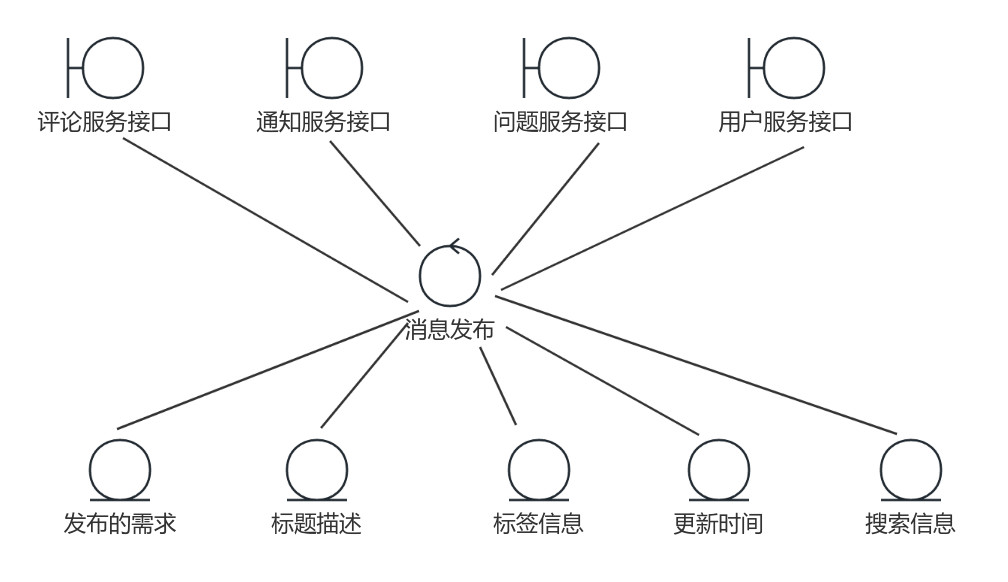


2.2用例实现顺序图

续



2.3参与类图



3 设计模型

3.1子系统与接口

子系统设计

用户管理子系统 (User Management Subsystem)

负责用户的注册、登录、权限验证等功能。

包括以下模块：

注册模块：负责用户的注册流程，验证信息并创建用户。

登录模块：负责用户的认证，提供登录和登出功能。

权限验证模块：验证用户是否具备特定的权限进行操作。

需求管理子系统 (Question Management Subsystem)

负责需求的创建、修改、删除、查询等功能。

包括以下模块：

需求发布模块：负责用户发布新需求，包含需求标题、内容、标签等。

需求修改模块：允许用户修改已发布的需求。

需求查询模块：提供通过标题、标签、时间等条件查询需求的功能。

需求统计模块：统计需求的浏览次数、点赞、评论等信息。

评论与反馈子系统 (Comments and Feedback Subsystem)

负责用户对需求的评论、反馈与评分功能。

包括以下模块：

评论模块：用户可以对需求进行评论和回复。

反馈模块：用户可以给需求提出建议或反馈。

评分模块：提供对需求或用户的评分功能，便于后续排序。

搜索与筛选子系统 (Search and Filter Subsystem)

负责基于用户输入的关键字进行需求的搜索和筛选。

包括以下模块：

搜索模块：允许用户通过关键词搜索需求。

筛选模块：根据发布时间、用户、标签等进行筛选需求。

通知与消息子系统 (Notification and Messaging Subsystem)

负责系统内的消息推送及通知管理，确保用户及时收到需求的相关更新。

包括以下模块：

消息推送模块：推送需求的评论、反馈及系统消息。

通知中心模块：提供需求相关的系统通知与更新提示。

接口设计

为了让这些子系统协同工作，需要设计各个子系统之间的接口，以下是一些主要的接口：

1. 用户管理接口

注册接口 (Register API)

描述：提供用户注册功能。

请求类型：POST

路径：/api/user/register

参数：

username (字符串) - 用户名

password (字符串) - 密码

email (字符串) - 邮箱

响应：

成功：200 OK + 用户信息

失败：400 Bad Request + 错误信息

登录接口 (Login API)

描述：提供用户登录功能。

请求类型：POST

路径：/api/user/login

参数：

username (字符串) - 用户名

password (字符串) - 密码

响应：

成功：200 OK + 用户Token

失败：401 Unauthorized + 错误信息

2. 需求管理接口

发布需求接口 (Publish Question API)

描述：提供用户发布需求的功能。

请求类型：POST

路径：/api/question/publish

参数：

title (字符串) - 需求标题

description (字符串) - 需求描述

tags (数组) - 需求标签

响应：

成功：200 OK + 发布成功提示

失败：400 Bad Request + 错误信息

修改需求接口 (Edit Question API)

描述：允许用户修改已经发布的需求。

请求类型：PUT

路径：/api/question/edit/{id}

参数：

id (整数) - 需求ID

title (字符串) - 需求标题

description (字符串) - 需求描述

tags (数组) - 需求标签

响应：

成功：200 OK + 更新成功提示

失败：404 Not Found + 错误信息

3. 评论与反馈接口

添加评论接口 (Add Comment API)

描述：允许用户对需求进行评论。

请求类型：POST

路径：/api/comment/add

参数：

questionId (整数) - 需求ID

content (字符串) - 评论内容

响应：

成功：200 OK + 评论成功提示

失败：400 Bad Request + 错误信息

4. 搜索与筛选接口

需求搜索接口 (Search Question API)

描述：根据用户提供的关键词进行需求的搜索。

请求类型：GET

路径：/api/question/search

参数：

keyword (字符串) - 搜索关键词

响应：

成功：200 OK + 需求列表

失败：404 Not Found + 错误信息

5. 通知接口

消息推送接口 (Notification API)

描述：推送用户的需求更新及相关消息。

请求类型：GET

路径：/api/notification

响应：

成功：200 OK + 消息列表

失败：404 Not Found + 错误信息

3.2类图与类设计

主要类：

User类（用户类）

Question类（需求类）

Comment类（评论类）

Tag类（标签类）

Notification类（通知类）

System类（系统类）

类图关系

User 类和 Question 类是一对多的关系。一个用户可以发布多个需求，但一个需求由一个用户发布。

Question 类和 Comment 类也是一对多的关系。一个需求可以有多个评论。

Question 类和 Tag 类是多对多的关系。一个需求可以包含多个标签，一个标签可以属于多个需求。

User 类和 Notification 类是一对多的关系。一个用户可以接收到多条系统通知。

System 类将作为全局管理类，负责处理需求发布、评论管理、通知推送等。

类设计

User类

属性：

id (整数) - 用户ID

username (字符串) - 用户名

password (字符串) - 用户密码

email (字符串) - 邮箱

questions (列表<Question>) - 用户发布的需求列表

comments (列表<Comment>) - 用户发布的评论列表

notifications (列表<Notification>) - 用户收到的通知列表

方法：

register() - 用户注册

login() - 用户登录

publishQuestion(Question question) - 发布需求

addComment(Comment comment) - 添加评论

Question类

属性：

id (整数) - 需求ID

title (字符串) - description (字符串) - 需求描述

tags (列表<Tag>) - 需求标签列表

author (User) - 需求发布者

comments (列表<Comment>) - 需求的评论列表

views (整数) - 浏览次数

likes (整数) - 点赞次数

方法：

addTag(Tag tag) - 添加标签

removeTag(Tag tag) - 删除标签

addComment(Comment comment) - 添加评论

getDetails() - 获取需求详情

incrementViews() - 增加浏览次数

Comment类

属性：

id (整数) - 评论ID

content (字符串) - 评论内容

author (User) - 评论发布者

question (Question) - 所评论的需求

timestamp (日期时间) - 评论时间

方法：

editContent(String newContent) - 编辑评论

deleteComment() - 删除评论

Tag类

属性：

id (整数) - 标签ID

name (字符串) - 标签名

questions (列表<Question>) - 关联的需求列表

方法：

addQuestion(Question question) - 关联需求

removeQuestion(Question question) - 解除需求关联

Notification类

属性：

id (整数) - 通知ID

content (字符串) - 通知内容

recipient (User) - 接收者

timestamp (日期时间) - 发送时间

方法：

markAsRead() - 标记为已读

deleteNotification() - 删除通知

System类

属性：

users (列表<User>) - 系统中的用户列表

questions (列表<Question>) - 系统中的需求列表

comments (列表<Comment>) - 系统中的评论列表

方法：

registerUser(User user) - 注册用户

loginUser(String username, String password) - 用户登录

publishQuestion(User user, Question question) - 发布需求

addComment(User user, Question question, Comment comment) - 添加评论

sendNotification(User user, String content) - 发送通知

类之间的互动场景

用户注册与登录

用户通过 User 类的 register() 方法注册，并通过 login() 方法登录。

发布需求

登录用户通过 publishQuestion(Question question) 方法发布需求，系统将需求对象添加到系统的 questions 列表中。

评论需求

用户可以通过 addComment(Comment comment) 方法对需求进行评论，评论将被关联到相应的需求。

标签管理

用户在发布需求时，可以通过 addTag(Tag tag) 方法为需求添加标签。

通知系统

系统通过 System 类的 sendNotification(User user, String content) 方法向用户发送通知。

