**任务管理系统架构设计说明书**

**技术栈**：Spring Boot + MySQL + Redis + Vue.js/React  
**版本**：V1.0  
**日期**：2025年XX月XX日

**一、系统架构总览**

本系统采用**垂直分层+水平解耦**的五层架构设计，各层职责明确，技术选型以Spring Boot生态为核心。

**二、五层架构详解**

**1. ​展现层（Presentation Layer）​**

* ​**作用**：用户直接交互的界面，负责数据展示和操作接收。
* ​**技术实现**：
  + ​**前端框架**：Vue.js/React（二选一）
    - 组件化开发（如任务卡片、进度条）
    - 响应式布局（适配PC/移动端）
  + ​**通信协议**：RESTful API（通过HTTP与后端交互）
  + ​**实时功能**：WebSocket（任务状态实时更新）
* ​**示例代码**：

vue

复制

<!-- Vue组件示例：任务列表 -->

<template>

<div v-for="task in tasks" :key="task.id">

<h3>{{ task.title }}</h3>

<button @click="updateTask(task)">完成</button>

</div>

</template>

**2. ​应用层（Application Layer）​**

* ​**作用**：处理业务逻辑，协调各组件交互。
* ​**技术实现**：
  + ​**核心框架**：Spring Boot
    - 控制器（@RestController）：接收前端请求
    - 服务层（@Service）：封装业务逻辑（如任务分配算法）
    - 数据传输对象（DTO）：隔离前后端数据模型
  + ​**关键功能**：
    - 用户认证（Spring Security）
    - 文件上传（Spring Boot Actuator监控）
    - 第三方集成（邮件通知通过JavaMailSender）
* ​**示例代码**：

java

复制

*// Spring Boot Service层示例：创建任务*

@Service

public class TaskService {

@Autowired

private TaskRepository taskRepository;

public Task createTask(TaskRequest request) {

Task task = new Task(request.getTitle(), request.getDueDate());

return taskRepository.save(task);

}

}

**3. ​业务逻辑层（Domain Layer）​**

* ​**作用**：定义核心业务规则和实体关系。
* ​**技术实现**：
  + ​**领域模型**：使用JPA（Java Persistence API）映射数据库表

java

复制

@Entity

public class Task {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

@ManyToOne

private User assignee; *// 关联用户表*

private LocalDateTime dueDate;

}

* + ​**规则引擎**：通过Spring AOP实现权限校验（如仅管理员可删除任务）

**4. ​数据访问层（Data Access Layer）​**

* ​**作用**：封装数据库操作，提供数据持久化支持。
* ​**技术实现**：
  + ​**数据库选择**：MySQL（关系型数据库存储任务、用户信息）

sql

复制

*-- 任务表结构示例*

CREATE TABLE tasks (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL,

due\_date DATETIME,

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

* + ​**缓存优化**：Redis（存储高频访问数据，如在线用户列表）

java

复制

@Cacheable("users")

public User getUserById(Long id) {

return userRepository.findById(id).orElse(null);

}

**5. ​基础设施层（Infrastructure Layer）​**

* ​**作用**：提供系统运行所需的底层支持。
* ​**技术实现**：
  + ​**部署方案**：Docker容器化 + Kubernetes集群管理
  + ​**云服务**：阿里云ECS（弹性计算服务）
  + ​**安全防护**：Spring Boot Security + JWT Token（防止未授权访问）
  + ​**监控工具**：Prometheus + Grafana（实时监控CPU/内存使用率）

**三、关键技术实现**

**1. ​用户认证与授权**

* ​**流程**：
  1. 用户登录 → 密码加密存储（BCryptPasswordEncoder）
  2. 生成JWT Token → 前端携带Token访问受保护接口
* ​**代码示例**：

java

复制

*// 生成Token*

public String generateToken(UserDetails userDetails) {

return Jwts.builder()

.setSubject(userDetails.getUsername())

.setIssuedAt(new Date())

.signWith(SignatureAlgorithm.HS256, "secret-key")

.compact();

}

**2. ​实时协作功能**

* ​**技术选型**：
  + WebSocket + STOMP协议（通过Spring Boot的@MessageMapping实现）
  + 前端库：Socket.IO（兼容多浏览器）
* ​**场景**：多人编辑任务时，实时同步进度

**3. ​自动化提醒**

* ​**实现方式**：
  + Quartz定时任务：每天凌晨扫描即将到期的任务
  + 邮件通知：通过Spring Boot Mail发送提醒

**四、开发与部署建议**

**1. 开发流程**

| **阶段** | **技术工具** | **交付物** |
| --- | --- | --- |
| 需求分析 | Axure RP / 墨刀 | 原型图（含任务分配、甘特图） |
| 后端开发 | IntelliJ IDEA + Maven | Spring Boot RESTful API |
| 前端开发 | VS Code + Chrome DevTools | 响应式Web界面 |
| 测试 | Postman + JUnit | 接口测试报告 + 单元测试覆盖率 |

**2. 预算优化**

* ​**免费资源**：
  + MySQL：阿里云学生版（1核2G免费）
  + Redis：RedisInsight（开源可视化工具）
  + 云服务器：腾讯云轻量应用服务器（学生认证享折扣）

**五、架构优势总结**

1. ​**分层清晰**：各层职责明确，便于分工协作（如前端组专注Vue.js，后端组专注Spring Boot）
2. ​**扩展性强**：通过模块化设计（如单独拆分任务服务、用户服务）支持未来功能扩展
3. ​**学习成本低**：技术栈均为大学生常用工具，代码示例可直接复用

**备注**：完整项目需结合《软件需求规格说明书》和《数据库设计说明书》，本报告聚焦架构设计层面的技术决策。