# 事件管理系统设计文档

## 1. 项目概述

### 1.1 项目目标

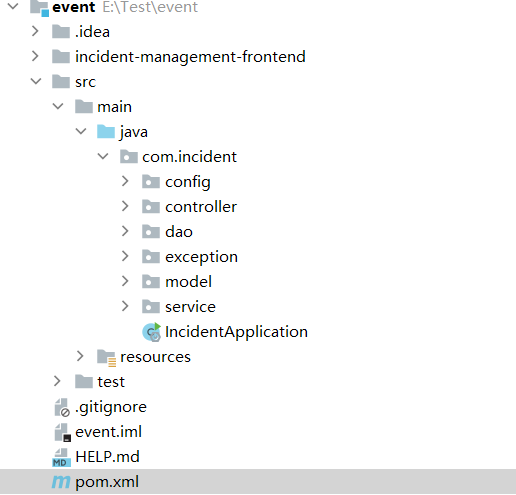
开发一个基于 Java 17 和 Spring Boot 的事件管理系统，实现事件的增删改查功能，并确保系统的性能、可靠性和可维护性。

### 1.2 技术栈选择

* 后端框架：Spring Boot 3.2.0
* 构建工具：Maven
* 容器化：Docker
* 缓存：Caffeine
* 前端框架：React
* 测试框架：JUnit 5, Mockito
* API文档：Springdoc OpenAPI

## 2. 系统架构

### 2.1 整体架构



### 2.2 核心组件

1. 事件管理模块
2. 缓存模块
3. 异常处理模块
4. 异步处理模块
5. 性能监控模块

## 3. 详细设计

### 3.1 数据模型

java

public class Incident {

private String id;

private String title;

private String description;

private Priority priority;

private Status status;

private LocalDateTime createdAt;

private LocalDateTime updatedAt;

}

### 3.2 API 设计

#### RESTful API 端点

bash

POST /api/incidents - 创建事件

GET /api/incidents - 获取所有事件

GET /api/incidents/{id} - 获取单个事件

PUT /api/incidents/{id} - 更新事件

DELETE /api/incidents/{id} - 删除事件

### 3.3 缓存策略

* 使用 Caffeine 实现内存缓存
* 缓存配置：
  + 最大容量：100条记录
  + 过期时间：60分钟
  + 刷新策略：写入时更新

### 3.4 异步处理

* 使用 Spring 的 @Async 注解
* 配置两个线程池：
  + taskExecutor：处理普通任务
  + asyncServiceExecutor：处理异步服务

## 4. 性能优化

### 4.1 缓存优化

### 4.2 并发处理

* 使用 ConcurrentHashMap 存储数据
* 使用线程池管理并发请求

## 5. 测试策略

### 5.1 单元测试

* Controller 层测试
* Service 层测试
* Performance层测试

### 5.2 集成测试

* API 端点测试
* 缓存机制测试
* 并发测试

### 5.3 性能测试

* 压力测试（使用 JMeter）
* 负载测试
* 并发测试

## 6. 部署方案

### 6.1 Docker 部署

dockerfile

FROM openjdk:17-jdk-slim

COPY target/\*.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","/app.jar"]

### 6.2 监控方案

* 使用 Spring Boot Actuator 进行应用监控

## 7. 安全性考虑

### 7.1 异常处理

* 统一异常处理机制
* 自定义业务异常
* 友好的错误响应

## 8. 项目依赖

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.0</version>  
 <relativePath/>  
 </parent>  
  
 <groupId>com.incident</groupId>  
 <artifactId>incident-management</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>incident-management</name>  
 <description>Incident Management System</description>  
  
 <properties>  
 <java.version>21</java.version>  
 <mybatis-plus.version>3.5.9</mybatis-plus.version>  
 <mysql.version>8.0.33</mysql.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Starters -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-validation</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-cache</artifactId>  
 </dependency>  
  
 <!-- MyBatis Plus -->  
 <dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-spring-boot3-starter</artifactId>  
 <version>${mybatis-plus.version}</version>  
 </dependency>  
  
 <!-- MySQL Driver -->  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <version>${mysql.version}</version>  
 </dependency>  
  
 <!-- Druid -->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid-spring-boot-starter</artifactId>  
 <version>1.2.20</version>  
 </dependency>  
  
 <!-- Lombok -->  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
  
 <!-- Caffeine Cache -->  
 <dependency>  
 <groupId>com.github.ben-manes.caffeine</groupId>  
 <artifactId>caffeine</artifactId>  
 </dependency>  
  
 <!-- Jackson for JSON -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.datatype</groupId>  
 <artifactId>jackson-datatype-jsr310</artifactId>  
 </dependency>  
  
 <!-- Testing Dependencies -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-junit-jupiter</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>32.1.3-jre</version>  
</dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <exclude>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

## 9. 开发时间线

* Day 1: 1.项目初始化和基础架构搭建,2.核心功能实现和单元测试
* Day 2: 1. 缓存实现和性能优化 2.文档完善

## 10. 项目交付件

1. 源代码
2. 文档

## 11. 注意事项

1. 确保代码质量和测试覆盖率
2. 遵循 RESTful API 设计规范
3. 做好异常处理和日志记录
4. 保证系统的可扩展性
5. 注重代码的可维护性