编程规范文档

目录

1. 概述
2. 通用规范
3. 前端规范
    - 文件结构
    - 代码风格
    - CSS/Sass规范
    - JavaScript/TypeScript规范
    - React规范
    - Material-UI规范
4. 后端规范
    - 文件结构
    - 代码风格
    - API设计规范
5. 数据库规范
6. 版本控制规范
7. 代码评审规范
8. 文档规范

1.概述

本编程规范文档旨在确保Code Cube项目的开发过程中，开发团队遵循统一的编程标准，提高代码的可读性、可维护性和一致性。

2.通用规范

编码风格：所有代码应遵循一致的编码风格。

注释：代码应包含适当的注释，解释复杂的逻辑和算法。注释应简明扼要，避免冗长。

命名约定：使用有意义的命名。变量名、函数名和类名应使用驼峰命名法（camelCase），常量名使用全大写字母和下划线分隔（UPPER_SNAKE_CASE）。

代码复用：避免代码重复，提取公共代码到模块或函数中进行复用。

3.前端规范

3.1 文件结构

所有前端代码应放置在`/frontend/src`目录下。

组件应放置在`/src/component`目录下。

页面应放置在`/src/page`目录下。

前后端交互应放置在`/src/service`目录下。

3.2 JavaScript/TypeScript规范

强制使用TypeScript进行类型检查。

避免使用any类型，尽量明确类型定义。

所有函数和方法应有明确的返回类型。

3.3 React规范

组件文件名应使用大写开头的驼峰命名法（PascalCase）。

每个组件应尽量保持简洁，单一职责，避免太多的逻辑堆积在一个组件中。

使用函数式组件和React Hooks，避免使用类组件。

状态管理尽量使用React的Context API或状态管理库（如Redux、Recoil等）。

避免直接操作DOM，尽量使用React的ref和受控组件。

所有组件应有PropTypes或TypeScript类型定义，确保类型安全。

3.4 Material-UI规范

使用Material-UI的主题功能，统一管理颜色、字体和其他样式变量。

避免直接修改Material-UI组件的样式，尽量通过主题或CSS-in-JS解决方案（如 makeStyles）进行样式定制。

使用Material-UI的Grid系统进行布局，保持布局的一致性和响应性。

4.后端规范

4.1 文件结构

所有后端代码应放置在`/backend/src/main/java`目录下。

控制器文件应放置在`/src/main/java/com/example/backend/controller`目录下。

服务文件应放置在`/src/main/java/com/example/backend/service`目录下。

存储库文件应放置在`/src/main/java/com/example/backend/repository`目录下。

DAO文件应放置在`/src/main/java/com/example/backend/dao`目录下。

实体文件应放置在`/src/main/java/com/example/backend/domains`目录下。

配置文件应放置在`/src/main/resources`目录下。

4.2 代码风格

使用最新的Java版本，避免使用过时的语言特性。

遵循单一职责原则，每个类和方法应只负责一个功能。

使用Spring Boot提供的依赖注入和AOP特性，减少代码耦合。

所有的异常应处理并记录日志，避免程序崩溃。

4.3 API设计规范

所有API应遵循RESTful风格。

使用HTTP状态码表示请求结果。

所有API应返回统一格式的响应数据，包括成功和错误信息。

使用Spring Boot的`@RestController`和`@RequestMapping`注解定义控制器和路由。

使用Swagger或OpenAPI生成API文档，保持文档与代码一致。

5.数据库规范

使用MySQL作为数据库。

数据库表名应使用小写字母和下划线分隔（snake_case）。

每个表应有主键，推荐使用自增ID。

字段名应使用有意义的命名，避免缩写。

所有表应有适当的索引，以提高查询性能。

使用JPA进行数据库操作，避免直接书写SQL语句。

6.版本控制规范

使用Git进行版本控制。

每个功能或修复应创建一个新的分支，分支命名应简洁明了，如`feature/功能描述`或`bugfix/问题描述`。

所有代码应提交到远程仓库，避免本地代码丢失。

提交信息应简洁明了，描述本次提交的内容。

7.代码评审规范

所有代码在合并到主分支之前应经过代码评审。

代码评审应关注代码的逻辑、可读性、性能和安全性。

评审者应提供建设性的反馈，提出改进建议。

8.文档规范

所有公共API应有详细的接口文档，描述请求方法、参数、返回值和示例。

每个模块应有README文件，描述模块的功能、使用方法和注意事项。

文档应及时更新，保持与代码一致。