

软件工程化(自动化、协作化)

开发环境与工具

开发工具

- 集成开发环境 (IDE)
 - 前端: Visual Studio Code
 - 后端: IntelliJ IDEA Ultimate
- 版本控制
 - Git

前端开发环境

- 主要依赖
 - Vue: 3.2.13
 - axios: 1.7.2
 - core-js: 3.37
 - vue-router: 4.3.3
 - element-plus: 2.7.5
- 构建工具: Webpack
- 包管理器: npm
- 主要开发依赖
 - eslint: 7.32.0
 - @babel/core: 7.12.16
 - @vue/cli-service: 5.0.0

后端开发环境

- 框架: Spring Boot v3.3.0
- 构建工具: Maven v4.0.0
- 数据库: H2 Database v2.2.220
- 主要的库
 - spring-boot-starter-web
 - spring-boot-starter-test
 - java-jwt: 4.4.0
 - hazelcast: 5.4.0
 - dev.langchain4j: 0.31.0

依赖管理

- 前端：package.json
- 后端：pom.xml

自动化流程

- 持续集成与持续部署 (CI/CD)
 - 使用GitHub Actions进行自动化build、release、deploy (与测试)
- 自动化测试
- 代码质量
 - ESLint用于前端代码规范检查

[!WARNING]

暂时还没写 测试 和 代码规范检测 的 xml

协作方式与规范

- 协作工具
 - 微信交流
- Pull Requests
 - 所有的代码变更都通过 PR 来进行，这样可以方便代码审查
- 代码评审
 - 在合并 PR 之前,每个Pull Request需要至少一名开发人员的审查
 - 使用GitHub的代码审查工具进行审查
- 文档
 - 保持 README 和其他文档的更新，包括如何设置开发环境、运行测试等
- 使用 Issues 进行任务管理
 - 使用 GitHub Issues 来跟踪功能请求、bug 和其他任务
- 开发规范
 - 遵循前端Vue.js风格指南和后端Spring Boot最佳实践
 - 使用规范的Git分支模型 (如Git Flow)

版本控制策略

- 分支策略
 - main 分支：稳定的生产代码
 - develop 分支：最新的开发代码
 - feature 分支：新功能开发
- 主要流程：
 - 从 develop 分支创建一个新的 feature/x 分支 (本地)
 - 在 feature/x 分支上开发新功能

- 完成功能开发后，将 `feature/x` 合并回 `develop` 分支
- 删除 `feature/x` 分支

部署流程

- 环境划分
 - 开发环境：本地开发和调试
 - 测试环境：用于集成测试和用户验收测试
 - 生产环境：正式上线的稳定版本
- 部署步骤
 1. 在GitHub上合并代码到 `main` 分支
 2. (github action) Jenkins触发构建任务，运行自动化测试
 3. 将应用打包 (release)
 4. 使用脚本进行部署
 5. 监控和日志记录确保部署成功