项目实现规范报告

项目概述

本项目采用 Spring Boot 作为基础框架，结合 MyBatis-Plus 操作 Mysql 数据库，同时利用 Redis 进行缓存优化，旨在构建一个高效、稳定且易于维护的系统。

代码规范

1. 遵循 Java 编码规范，包括**命名**约定、**注释**规范等。
    - 类名采用大驼峰命名法，方法名和变量名采用小驼峰命名法。
    - 为关键代码提供详细的注释，提高代码的可读性。
2. Spring Boot 相关**代码分层**规范
    - Controller 层负责接收请求和响应处理，参数校验和异常处理遵循统一规范。
    - Service 层负责业务逻辑处理，保持事务的一致性和完整性。
    - Mapper 层（MyBatis-Plus Mapper）专注于数据库操作，保证数据访问的准确性和高效性。
3. MyBatis-Plus 代码的**映射**和**SQL书写**规范。
    - Mapper接口使用与对应的数据库表相关且有意义的名称，例如 UserMapper 对应 user 表。（*接口名与表名对应*）
    - Mapper 接口的方法命名清晰反映数据库操作的意图。（*方法名与意图对应*）
    - XML文件的 namespace 要与对应的 Mapper 接口全路径一致。（*XML的路径与Mapper对应*）
    - XML 映射文件中的 SQL 语句书写规范，避免复杂和难以维护的查询。
4. Redis 相关数据**命名**和客户端**操作**规范
    - 键的命名遵循统一的命名空间和规则，便于管理和识别。
    - 对 Redis 操作进行封装，提供统一的接口，避免直接在业务代码中操作 Redis 客户端。

数据库设计规范

1. 数据类型选择
    - 为每个列选择合适的数据类型，以节省存储空间并提高性能。
2. 索引设计

- 在 Username 等经常用于查询的列上创建索引。
- 避免过度创建索引，以免影响插入和更新操作的性能。

3. 范式遵循
   - 尽量遵循第三范式，减少数据冗余。
   - 为了提高查询性能，在teacher表违反了第三范式。
4. 主键和外键
   - 为每张表定义主键，确保数据的唯一性和完整性。
   - 在存在关联关系的表之间建立外键，以维护数据的一致性。

缓存设计规范

1. 明确缓存的更新策略，如主动更新、定时更新或基于数据变更的被动更新。
   - 本项目设计为修改了表就直接删除缓存。
2. 设置合理的缓存过期时间，根据数据的热度和更新频率进行调整。

测试规范

1. 编写单元测试、集成测试和接口测试，确保代码的质量和功能的正确性。
   - 对数据库操作和缓存操作进行充分的测试。