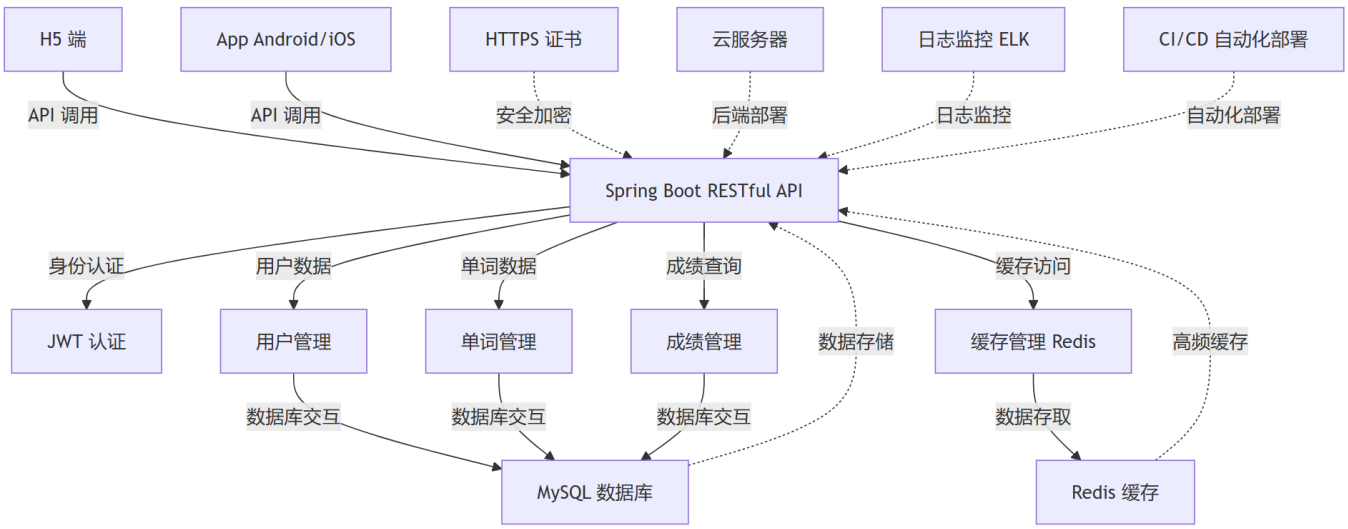
项目设计方案

一、项目概述

本项目是一款基于 uni-app 构建的跨平台单词学习应用，主要面向微信小程序及其他多端（H5/App）运行。前端提供单词练习、翻译、记录与成绩统计等功能；后端采用 Spring Boot 开发 RESTful API，负责数据持久化、业务逻辑处理和安全认证。系统支持微信登录、用户数据同步以及高频数据缓存，旨在为用户提供便捷、高效、跨端一致的单词学习体验。



二、前端技术栈与架构设计

1. 前端技术栈

- **uni-app**
  - 跨平台开发框架，支持微信小程序、H5、App 等多端编译
  - 基于 Vue.js 开发，组件化管理页面
- **Vue.js**
  - 利用 .vue 文件组织业务组件
  - 数据绑定与组件通信提升开发效率
- **uView UI**

    - 企业级、跨平台的 UI 组件库，提供导航栏、列表、弹窗、输入框等组件
    - 统一页面风格，减少重复开发
- **SCSS**
    - 通过 uni.scss 实现全局样式管理和模块化样式编写
    - 提高样式的可维护性和扩展性
- **微信小程序 API**
    - 使用 wx.request、wx.getStorageSync、wx.navigateTo 等接口实现网络请求、本地缓存与页面导航
    - 结合 uni-app 封装，确保跨平台兼容性

2. 项目目录结构

- **pages/**
    - 按功能模块划分（如 beiwords、beiwords1、beiwords2、common 等），各模块均采用 .vue 单文件组件格式编写
- **static/**
    - 存放图片等静态资源
- **配置文件**
    - 包含 `manifest.json`、`pages.json`、`project.config.json` 等，用于配置多端运行、路由和打包参数

3. 主要功能模块

(1) 单词练习模块（pages/beiwords/）

- 提供选择题形式的单词练习，支持单词发音、翻译展示、选项生成和计时答题
- 通过本地存储缓存单词数据与用户记录，实现练习记录的持久化

(2) 单词记录模块（pages/beiwords1/）

- 展示用户记录的单词笔记与错题
- 支持列表展示及详情查看

(3) 成绩展示模块（pages/beiwords2/）

- 展示用户的练习成绩、得分、错题数及记录日期
- 支持查看错题详情，以便针对性复习

(4) 数据交互

- 使用 wx.request 与后端 RESTful API 交互，数据格式均为 JSON
- 通过 wx.getStorageSync 与 wx.setStorageSync 实现本地数据缓存

三、后端设计方案

1. 后端技术栈

- **Spring Boot**
  - 基于 Spring Boot 快速构建 RESTful API 服务
  - 内置自动配置和大量开箱即用的功能，便于快速开发和部署
- **MySQL**
  - 关系型数据库，负责存储用户信息、单词数据、成绩记录等结构化数据
  - 设计合理的表结构，支持事务和复杂查询
- **Redis**
  - 内存缓存系统，用于缓存高频访问的数据，如单词列表和用户会话信息
  - 降低数据库压力，提高系统响应速度
- **JWT**
  - JSON Web Token，用于实现无状态的安全认证
  - 结合微信 UnionID（或其他第三方登录信息）实现跨端用户身份认证

2. 后端系统架构

(1) 系统总体架构

- **控制层（Controller）**
  - 提供 RESTful API 接口，供前端调用，支持单词练习、成绩查询、用户登录等功能
- **业务层（Service）**
  - 负责业务逻辑处理，例如单词数据查询、用户登录验证、成绩统计等
- **数据访问层（Repository/DAO）**
  - 通过 MyBatis 或 Spring Data JPA 等持久化框架与 MySQL 数据库交互

- **缓存层**
  - 通过 Redis 缓存频繁访问的数据，降低数据库访问压力

(2) **模块设计**

- **用户管理模块**
  - 用户注册、微信登录（或第三方登录）
  - JWT 生成与验证，确保接口安全性
- **单词管理模块**
  - 单词数据的增删改查操作
  - 数据存储在 MySQL 中，并结合 Redis 缓存常用的单词列表
- **成绩记录模块**
  - 保存用户练习成绩、错题记录及练习时间
  - 提供成绩查询接口，支持分页加载
- **数据同步与缓存模块**
  - 使用 Redis 缓存常用数据，如单词列表、用户会话信息
  - 通过差分同步方式，前端仅传输变更数据，降低网络传输量

3. 数据库设计

- **用户表（user）**
  - 字段示例：id、union_id、用户名、头像、注册时间等
- **单词表（word）**
  - 字段示例：id、单词名称、发音、释义、标签、分类、创建时间等
- **成绩表（score）**
  - 字段示例：id、用户ID、分数、错题数、记录日期等
- **练习记录表（practice_record）**
  - 存储用户每次练习的详情，包括正确/错误的单词记录

4. API 设计

- **用户接口**
  - `POST /api/auth/login`：用户登录（微信登录后传递 UnionID）并返回 JWT
  - `GET /api/user/{id}`：查询用户信息
- **单词接口**
  - `GET /api/words`：分页查询单词数据（结合 Redis 缓存）
  - `POST /api/words`：新增或更新单词信息
- **成绩接口**
  - `GET /api/scores/{userId}`：获取用户练习成绩记录（支持分页加载）
  - `POST /api/scores`：提交练习成绩记录

5. 安全认证

- 采用 JWT 实现无状态认证，后端在用户登录后生成 JWT，前端后续请求中携带该令牌
- 配置 Spring Security，对接口进行权限控制，确保只有经过认证的请求才能访问核心 API

6. 性能优化措施

- **Redis 缓存**
  - 缓存高频访问数据（如单词列表、用户会话），提高查询速度
  - 设置合理的过期时间，确保数据新鲜度与一致性
- **分页加载**
  - 对单词列表和成绩记录接口采用分页加载，避免一次性返回大量数据
- **异步处理**
  - 对部分复杂计算（如统计分析）使用异步任务或消息队列处理，减轻主线程负担
- **日志与监控**
  - 集成日志记录和监控系统，及时捕获异常并优化系统性能

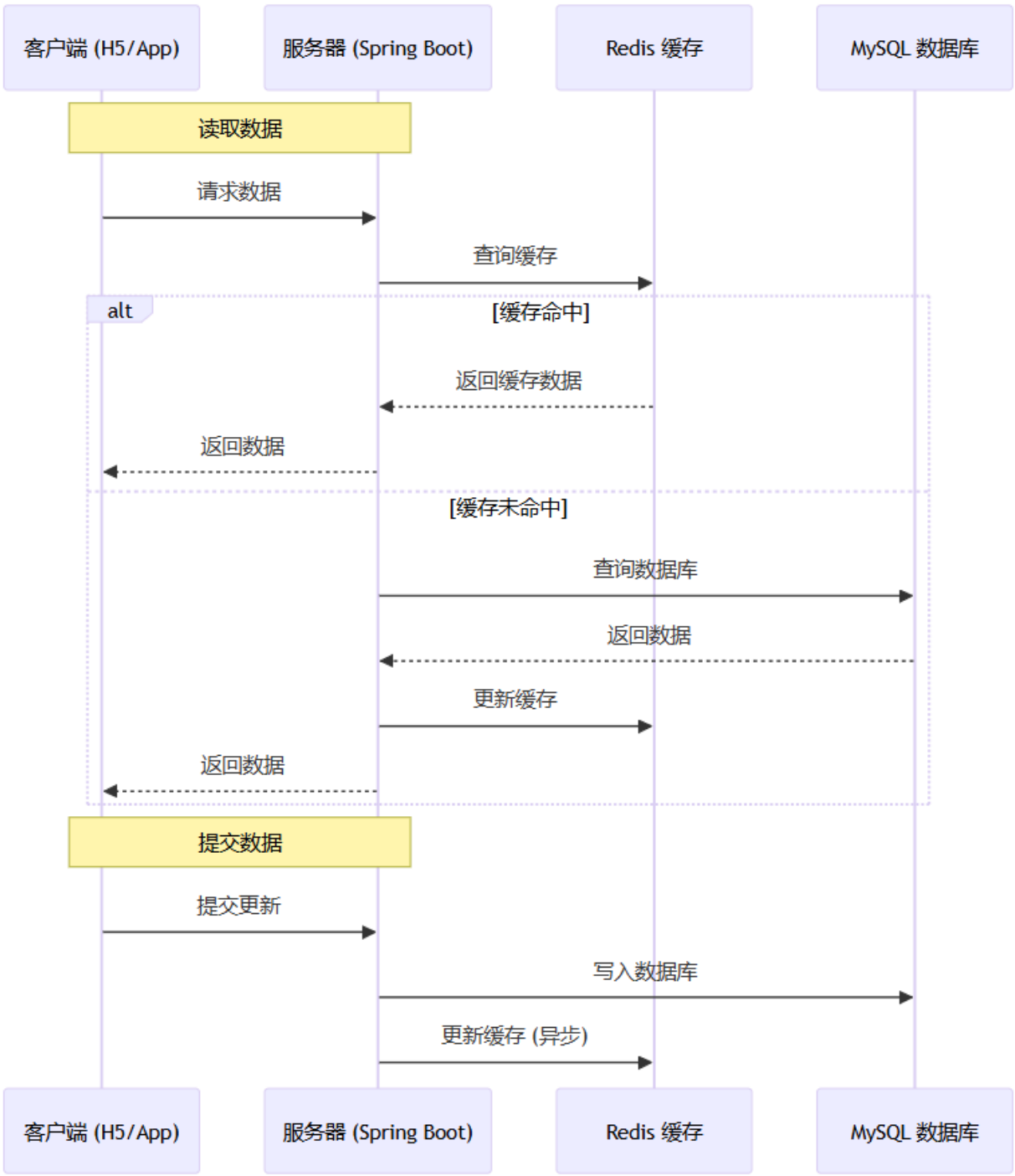
四、前后端交互与部署

1. 前后端交互

- 前端通过 wx.request 发起 HTTP 请求调用后端 RESTful API
- 数据格式采用 JSON，接口地址统一配置（可通过环境变量管理）
- 前端在登录、数据加载等关键操作中使用加载提示和错误提示，提升用户体验

2. 部署架构

- **后端部署**
  - 采用 Spring Boot 部署在云服务器或容器平台（如 Docker/Kubernetes）
  - MySQL 数据库与 Redis 缓存分别独立部署，确保数据持久化与高性能
- **前端部署**
  - 微信小程序代码通过微信开发者工具上传
  - H5 版本部署至 CDN，加速访问
- **持续集成与自动化部署**
  - 利用 CI/CD 流程自动化构建、测试与部署，提高开发效率与代码质量



五、总结

该项目前端基于 uni-app 与 Vue.js 构建跨平台单词学习应用，后端采用 Spring Boot 构建 RESTful API，整合 MySQL 存储结构化数据与 Redis 提高数据访问性能。通过 JWT 实现安全认证，前后端采用统一 JSON 数据格式交互，整体架构具备高扩展性、高性能与良好用户体验。未来可根据业务需求进一步扩展功能模块、优化性能和完善数据同步机制。