Yinli API 项目总结

🎉

以下是项目的详细总结：

✅ 已完成的功能

1. 项目基础架构

- ✅ 基于 Gin 框架的 RESTful API 服务
- ✅ 清晰的项目目录结构（cmd、internal、pkg 分层）
- ✅ Go 模块依赖管理

2. 配置管理系统

- ✅ 使用 Viper 库进行配置管理
- ✅ 支持三种环境配置：dev（开发）、stage（预发布）、prod（生产）
- ✅ 配置文件保存在 `config/` 目录

3. 数据库集成

- ✅ MySQL 数据库连接和管理
- ✅ GORM ORM 库集成
- ✅ 用户模型和数据访问层
- ✅ 数据库初始化脚本（`sql/init.sql`）

4. 缓存系统

- ✅ Redis 缓存集成
- ✅ 完整的 Redis 操作封装
- ✅ 用于频率限制和会话管理

5. 安全特性

- ✅ JWT 认证系统
- ✅ CORS 跨域支持
- ✅ 基于 Redis 的 API 频率限制
- ✅ bcrypt 密码加密
- ✅ 多层中间件安全防护

6. 用户认证接口

- ✅ 用户注册接口（`POST /api/auth/register`）
- ✅ 用户登录接口（`POST /api/auth/login`）
- ✅ 用户资料管理接口
- ✅ 管理员权限接口

7. 构建和部署

- ✅ 完整的 Makefile 构建脚本
- ✅ 支持多环境启动命令（`make dev`、`make stage`、`make prod`）
- ✅ Docker 容器化支持
- ✅ 自动生成 Docker Compose 文件

8. 测试框架

- ✅ 使用 Testify 测试框架
- ✅ 配置系统测试
- ✅ 中间件测试
- ✅ 接口测试框架（需要数据库连接时可启用）

9. API 文档

- ✅ Swagger 文档集成准备
- ✅ 支持按环境分离文档生成
- ✅ 文档保存在 `doc/` 目录

10. 详细文档

- ✅ 完整的 README.md 说明文档
- ✅ 技术架构说明
- ✅ 开发调试指南
- ✅ 部署说明

🚀 快速启动指南

1. 基本启动

```
# 进入项目目录
cd /home/table/Workspace/go/src/yinli-api

# 查看所有可用命令
make help

# 下载依赖
make deps

# 启动开发环境
make dev
```

2. Docker 启动

```
# 生成 Docker Compose 文件
make docker-compose-dev

# 启动 Docker 容器
make docker-up-dev
```

3. 运行测试

```
# 运行所有测试
make test

# 生成测试覆盖率报告
make test-coverage
```

📁 项目结构概览

```
yinli-api/
├── cmd/main.go                 # 应用程序入口
├── config/                     # 配置文件目录
│   ├── dev.yaml                # 开发环境配置
│   ├── stage.yaml              # 预发布环境配置
│   └── prod.yaml               # 生产环境配置
├── internal/                   # 内部应用代码
│   ├── handler/                # HTTP 处理器
│   ├── middleware/             # 中间件
│   ├── model/                  # 数据模型
│   ├── repository/             # 数据访问层
│   └── service/                # 业务逻辑层
├── pkg/                        # 可复用包
│   ├── auth/                   # JWT 认证
│   ├── cache/                  # Redis 缓存
│   ├── config/                 # 配置管理
│   └── database/               # 数据库连接
├── docker/                     # Docker 相关文件
├── sql/                        # 数据库脚本
├── test/                       # 测试文件
├── Dockerfile                  # Docker 镜像构建
├── Makefile                    # 构建脚本
└── README.md                   # 项目说明
```

🔧 主要技术栈

- **Web 框架**: Gin v1.11.0
- **数据库**: MySQL 8.0 + GORM
- **缓存**: Redis 7
- **认证**: JWT (golang-jwt/jwt/v5)
- **配置**: Viper v1.21.0
- **测试**: Testify v1.11.1
- **文档**: Swagger (gin-swagger)
- **容器**: Docker + Docker Compose

🛠️ 可用的 Make 命令

```
make help                # 显示所有可用命令
make dev                 # 启动开发环境
make stage               # 启动预发布环境
make prod                # 启动生产环境
make build               # 构建应用程序
make test                # 运行测试
make docker-up-dev       # 启动 Docker 开发环境
make clean               # 清理构建文件
```

📡 API 接口列表

认证接口

- `POST /api/auth/register` - 用户注册
- `POST /api/auth/login` - 用户登录

用户接口（需要 JWT 认证）

- `GET /api/user/profile` - 获取用户资料
- `PUT /api/user/profile` - 更新用户资料
- `PUT /api/user/password` - 修改密码

管理员接口（需要管理员权限）

- `GET /api/admin/users` - 获取用户列表
- `DELETE /api/admin/users/{id}` - 删除用户
- `PUT /api/admin/users/{id}/status` - 更新用户状态

系统接口

- `GET /health` - 健康检查
- `GET /swagger/*` - API 文档

🔒 安全特性

1. **JWT 认证**: 基于 JSON Web Token 的用户认证
2. **CORS 保护**: 可配置的跨域资源共享
3. **频率限制**: 基于 Redis 的 API 调用频率限制
4. **密码加密**: 使用 bcrypt 加密用户密码
5. **中间件保护**: 多层安全中间件防护

🌍 环境配置

项目支持三种环境配置，每种环境都有独立的配置文件：

- **dev**: 开发环境，详细日志，宽松的安全设置
- **stage**: 预发布环境，生产级配置，用于测试
- **prod**: 生产环境，最严格的安全设置

📊 测试覆盖

项目包含以下测试：

- 配置系统测试
- JWT 认证中间件测试
- CORS 中间件测试
- 请求 ID 中间件测试

- 基础 API 接口测试

🚀 部署建议

开发环境

```
make dev
```

生产环境

```
# 使用 Docker 部署
make docker-compose-prod
make docker-up-prod
```

📝 下一步建议

1. **数据库设置**: 确保 MySQL 数据库运行并执行 `sql/init.sql`
2. **Redis 设置**: 确保 Redis 服务运行
3. **环境变量**: 在生产环境中设置敏感配置的环境变量
4. **SSL 证书**: 在生产环境中配置 HTTPS
5. **监控日志**: 添加应用监控和日志收集
6. **API 文档**: 运行 `make install-tools` 安装 Swagger 工具并生成文档

🎯 项目特色

- **完整的企业级架构**: 分层清晰，易于维护
- **高度可配置**: 支持多环境配置
- **安全性强**: 多重安全防护机制
- **易于部署**: Docker 容器化支持
- **测试完备**: 完整的测试框架
- **文档齐全**: 详细的开发和部署文档

🔗 相关链接

- Gin 框架文档
- GORM 文档
- Viper 配置库
- JWT Go 库
- Redis Go 客户端