WeRun后端24暑期纳新第一阶段

1. 作业介绍

完成一个简易的日记网站：意语

技术栈：

- Java：springboot + Jpa/mybatis-plus + JWT + swagger + lombok + validation
- Golang：推荐 Gin + GORM + go-jwt

开发工具：IDEA / Goland, 接口测试工具（postman / apifox），数据库链接工具（navicat / datagrip）

1.1 功能点需求描述

1.1.1 登录注册

- **实现用户手机号密码一键注册登录接口。**

本系统的用户没有用户名，以用户id（主键）和手机号为标识，一个手机号码只能绑定一个用户。

一键登录注册的逻辑如下：

1. 输入手机号和密码
2. 用户此前没有注册登录过该系统（也就是数据库中没有该用户的记录），请求该接口会根据输入的手机号和密码在系统中注册该用户，并跳转第四步
3. 如果该用户此前注册登陆过该系统，根据用户输入手机号找到对应记录中的密码信息，与该用户输入密码比对，比对正确则登录成功
4. 登录成功，将用户基本信息（手机号，用户id）封装到token中，返回登录成功信息和token

- **实现认证拦截器，不带token或者token校验错误，拦截请求。**

ps：不是所有的接口都需要拦截，有些接口不需要登录并携带token就可以访问，在你的拦截器中配置这些接口的路径，遇到他们直接将他们放行

- **自行封装JWT工具类，实现基于JWT的token颁发和认证机制**

提示：JWT工具类需要有一个getToken方法颁发token，一个verify方法校验token，一个decodeUser方法解析token携带的用户信息，获得登录用户的id，手机号等信息

- **在需要用户信息的接口中，能够通过JwtUtils 解析携带的token获得用户信息**

1.1.2 日记模块

- 需求一：日记文本以文本形式持久化存储，需要实现**增删改**功能。
  - 每条日记除了有文本外，还需要有创建时间、最近修改时间、标签等属性。
  - 字体、排版、图片、超链接等无需实现。
  - 每个用户只能看到自己创建的日记。这里我们需要在接口内获得该请求用户的信息。
- 需求二：查询日记需要能够按照标签/内容/时间范围查询，能够按创建时间/最近编辑时间排序，能够分页查询。
  - 要求用**一个接口**实现，根据传参动态查询。
- 需求三：每条日记可以添加一个或多个标签。
  - 标签可以实现增删改。
  - 可以编辑标签名称，使得对应所有日记的标签都发生变化
  - 可以批量删除某一个标签下的所有日记

1.2 业务实现补充提示

提示1 数据库设计：

部分业务的实现可能需要用到数据库的一对多、多对多设计，可能需要用到连表查询等。个人认为这里有些难度，大家完成时要有耐心，做好注释。

提示2 接口设计：

对于每个用户而言，自己的日记都是不同的，所以我们要在**部分接口**内获得该用户的信息，然后根据用户信息对数据进行过滤就行啦。

提示3 假删除：

真删除： 指的就是**彻底地删除**。 从数据库表内将数据 进行移除 delete 。就是删除了数据库某一张表中的某一条记录，不但前端访问数据库的时候得不到这些数据,后台访问数据库的时候也看不到这条记录了.

假删除： 指的就是**逻辑上的删除**。数据库表内， 数据会包含一个标识flag字段 ， 例如： status（删除标识，0代表未删除，1代表删除，默认为0） ，执行假删时，只是将数据的删除标识 status从 0 改 1（update）。 本质上是数据的更新.后台访问数据库的时候仍然可以访问这写条数据,只不过**前端访问接口的时候得不到这些数据**了,相当于这些数据在用户的"眼"中被删除了.但**后台访问数据库的时候仍然能看到这条数据**。

本次要求使用假删除的方式删除~~好友和信件~~。（更正：删除tag或日记内容）

提示4 模糊查询：

大家平常在搜索信息的时候，通常不需要输入完整的关键词，就能查询到相应的信息。如搜索“微积分”可以搜索到“微积分（1）”和“微积分（2）”等。

模糊查询和直接展示可以用同一个接口实现，根据传参动态查询。

提示5 分页展示：

当查询的结果非常多的时候，一次性把大量的数据传给前端会造成前端压力过大。这种时候就需要使用分页功能，把数据分成很多页的，一页一页的传送到前端。

所需的技术栈不限制，大家可以自己找现成的模块也可以自己手写。

提示6 JWT工具类参考（Java）：

JWT工具类需要有一个getToken方法颁发token，一个verify方法校验token，一个decodeUser方法解析token携带的用户信息，获得登录用户的id，昵称等信息

1.3 关于 Java / Golang 方向的介绍

Java：

- 编码较为规范，项目结构有严格约束
- Springboot 框架涉及很多中间件与设计模式，学习有一定难度
- 学习路线明确，互联网教程多
- 使用人数多，生态非常完善，目前以做业务逻辑居多

Golang：

- 教程相对较少，可参考的优秀项目稀缺
- 学习较为简单
- 编码与项目结构约束较少，简单灵活
- 目前岗位需求不如Java多，以做基础架构为主

2. 接口要求

本次作业要求**用户手机号密码一键注册登录接口格式。**

本次作业要求**该接口严格按照下面要求实现，其余接口不做要求：**

2.1 接口介绍

本接口用来实现用户一键注册登录功能，具体功能见上面的功能点需求描述。

2.2 接口基本信息

接口路径：/diary/login

接口方法：POST

2.3 接口参数

参数名	必填	类型	备注
phoneNumber	是	String	用户唯一身份标识
password	是	String	密码

参数要求使用@RequestBody

2.4 接口返回值

返回参数在json字符串中以键值对存在。如：

```
{
    "status":"1",
    "msg":"注册或登录成功！",
    "token":"你生成的token"
}
```

其中" status"为状态参数，0代表失败，1代表成功。

3. 项目要求（Java）

3.1 技术栈要求

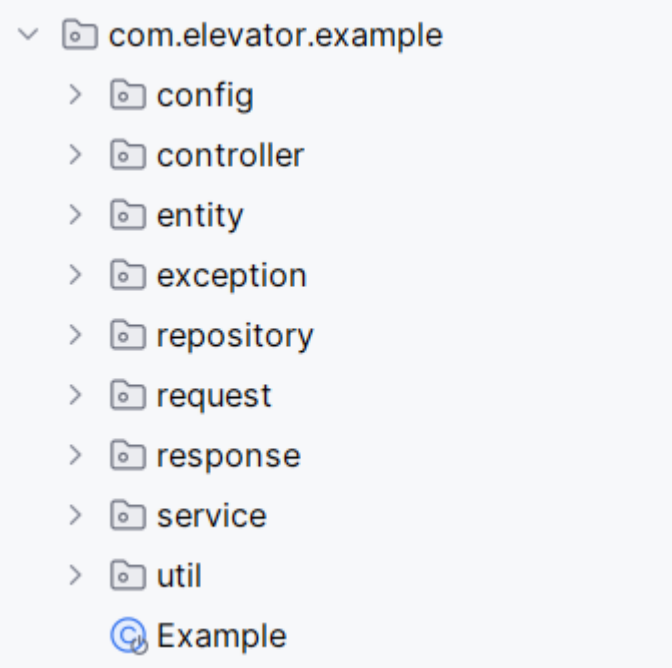
1. 整个项目为MVC架构，整体使用springboot框架；
2. 数据库要求使用MySQL；
3. 持久层框架要求使用mybatis-plus或JPA；
4. 需要使用**swagger生成在线接口文档**；
5. Jwt 和 过滤器/拦截器；
6. 实现全局异常处理，并合理向前端返回适当的报错信息；
7. 要求加上统一返回类

3.2 项目规范

此处作者 王梓 来自 WeRun2024 Java 22级大作业

项目分包

JPA的项目分包：



config为配置类

exception全局异常处理

request请求类

response请求返回类

util工具类包

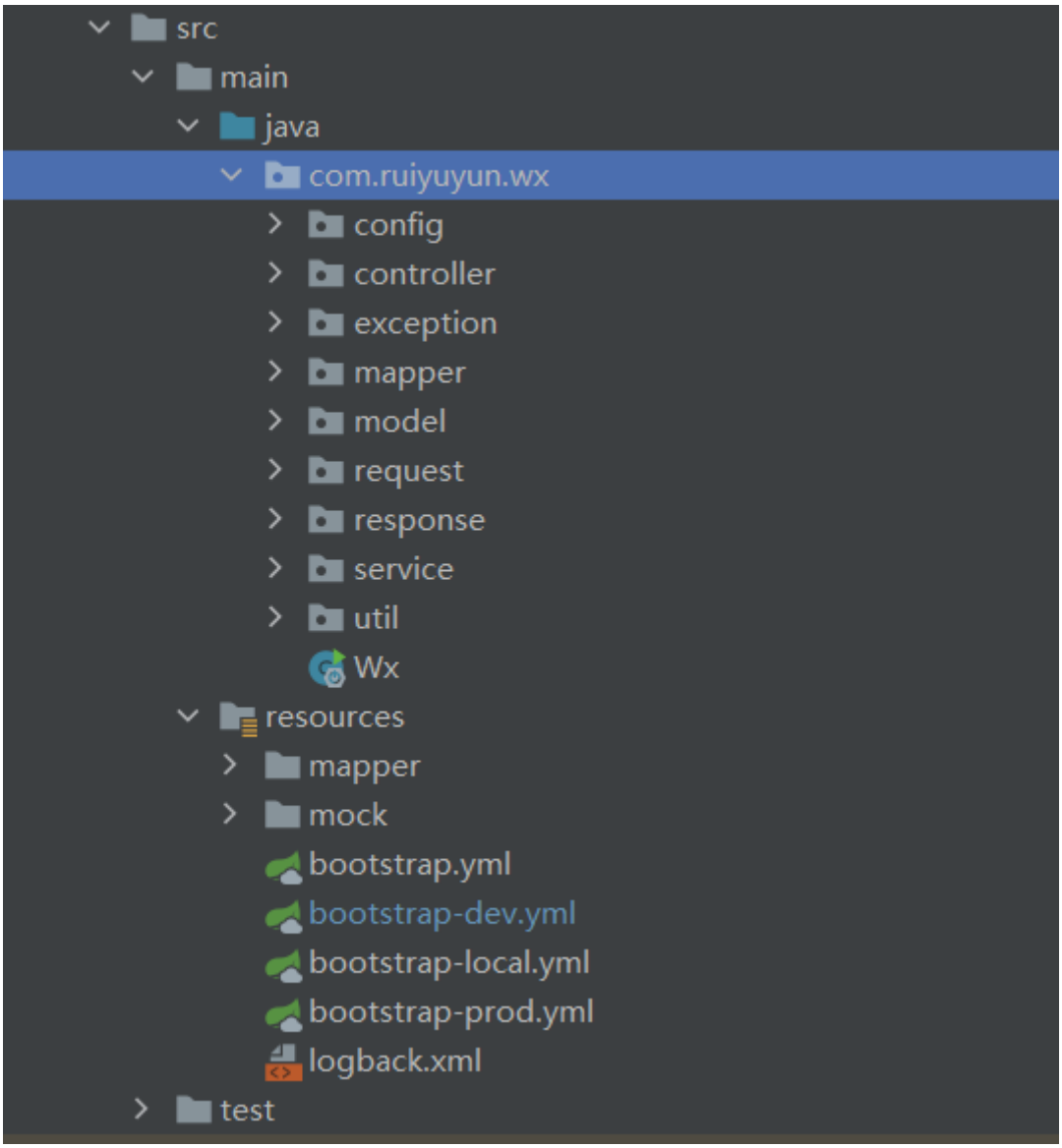
controller为控制层

entity实体类

repository数据访问层

service业务逻辑层

Mybatis-Plus的分包：



与JPA分包不同的地方：repository更名为mapper，entity更名为model

Controller:

整体开发遵循restful规范

固定的注解:
@RestController
@RequestMapping("/")
@RequiredArgsConstructor(onConstructor_ = {@Autowired})

注意：不要在controller层写过多的业务逻辑！！ 业务逻辑要封装在service层中

request:

请求类

命名规范:行为+对象+Request 如UpdateExampleRequest

response:

命名规范:对象+Response /对象+Page+Response /对象+List+Response

接口统一返回baseresponse类

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class BaseResponse<T> {
    private String msg;
    private Boolean success;
    private T data;

    public BaseResponse<T> construct(String msg, Boolean success){
        this.msg = msg;
        this.success = success;
        return this;
    }

    public BaseResponse<T> construct(String msg, Boolean success, T data){
        this.msg = msg;
        this.success = success;
        this.data = data;
        return this;
    }
}
```

T是泛型，可以接受你自定义的返回类对象,比如：

```
public BaseResponse<List<ListProdSkuResponse>>
```

ListProdSkuResponse是我们自定义的返回类对象，我们把它的list作为参数T传入baseresponse统一返回类中，然后再把baseresponse返回给前端

exception:

在这里存放全局异常处理类和自定义的异常类

config:

这里存放配置类

比如可以存放swagger的配置类：

```java

/**
 * Swagger 配置
 */
@Configuration
@EnableOpenApi
@EnableKnife4j
@Slf4j
public class SwaggerConfig {

    /**
     * 用于读取配置文件 bootstrap.yml 中 swagger 属性是否开启
     */
    @Value("${swagger.enabled}")
    Boolean swaggerEnabled;

    /**
     * 请求地址前缀 /service-name
     */
    @Value("${swagger.prefix}")
    private String prefix;

    @Bean
    public Docket docket() {
        return new Docket(DocumentationType.OAS_30)
                // 是否开启swagger
                .enable(swaggerEnabled)
                .select()
                // 过滤条件，扫描指定路径下的文件

.apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                // 指定路径处理，PathSelectors.any()代表不过滤任何路径
                .paths(PathSelectors.any())
                .build()
                .securityContexts(Arrays.asList(securityContext()))
                .securitySchemes(Arrays.asList(securityScheme()))
                .apiInfo(apiInfo())
                //请求URL都会加上/project前缀
                .pathMapping(prefix);


    }

    private AuthorizationScope[] scopes() {
        return new AuthorizationScope[]{
                new AuthorizationScope("all", "all scope")
```

```
46          };
47      }
48
49      private SecurityScheme securityScheme() {
50          return new ApiKey("Authorization", "Authorization", "header");
51      }
52
53      private SecurityContext securityContext() {
54          return SecurityContext.builder()
55                  .securityReferences(Arrays.asList(new
    SecurityReference("Authorization", scopes())))
56                  .operationSelector(o ->
57                          o.requestMappingPattern().matches("/.*")
58                  )
59                  .build();
60      }
61
62      private ApiInfo apiInfo() {
63          /*作者信息*/
64          Contact contact = new Contact(
65                  "",
66
    "https://zqmfb0zeawm.feishu.cn/docx/LAModLTGZoTiF3xqptdc1r3nnJf",
67                  ""
68          );
69          return new ApiInfo(
70                  "xxx模块",
71                  "xxx模块测试接口文档",
72                  "v1.0",
73
    "https://zqmfb0zeawm.feishu.cn/docx/LAModLTGZoTiF3xqptdc1r3nnJf",
74                  contact,
75                  "Apache 2.0",
76                  "http://www.apache.org/licenses/LICENSE-2.0",
77                  new ArrayList()
78          );
79      }
80
81      /**
82       * 解决 swagger 不支持高版本 springboot 问题
83       * https://fullstackcode.dev/2022/06/12/swagger-is-not-working-with-spring-
    boot-2-6-x/
84       */
85      @Bean
86      public static BeanPostProcessor springfoxHandlerProviderBeanPostProcessor()
     {
87          return new BeanPostProcessor() {
```

```
88              @Override
89              public Object postProcessAfterInitialization(Object bean, String
    beanName) throws BeansException {
90                  if (bean instanceof WebMvcRequestHandlerProvider || bean
    instanceof WebFluxRequestHandlerProvider) {
91
    customizeSpringfoxHandlerMappings(getHandlerMappings(bean));
92                  }
93                  return bean;
94              }
95
96              private <T extends RequestMappingInfoHandlerMapping> void
    customizeSpringfoxHandlerMappings(List<T> mappings) {
97                  List<T> copy = mappings.stream()
98                          .filter(mapping -> mapping.getPatternParser() == null)
99                          .collect(Collectors.toList());
100                 mappings.clear();
101                 mappings.addAll(copy);
102             }
103
104             @SuppressWarnings("unchecked")
105             private List<RequestMappingInfoHandlerMapping>
    getHandlerMappings(Object bean) {
106                 try {
107                     Field field = ReflectionUtils.findField(bean.getClass(),
    "handlerMappings");
108                     field.setAccessible(true);
109                     return (List<RequestMappingInfoHandlerMapping>)
    field.get(bean);
110                 } catch (IllegalArgumentException | IllegalAccessException e) {
111                     throw new IllegalStateException(e);
112                 }
113             }
114         };
115     }
116 }
```

或者是validation的配置类：

```
1 @Configuration
2 public class ValidatorConfig {
3
4     /**
5      * 配置快速失败模式，遇到一个不匹配直接返回，而不是扫描所有是否匹配：
6      * 1. 普通模式（默认模式）：会校验完所有的属性，然后返回所有的验证失败信息
```

```
     * 2. 快速失败模式：只要有一个验证失败，则返回
     */
    @Bean
    public Validator validator() {
        ValidatorFactory validatorFactory =
Validation.byProvider(HibernateValidator.class)
                .configure()
                // 快速失败模式
                .failFast(true)
                .buildValidatorFactory();
        return validatorFactory.getValidator();
    }

    /**
     * 设置 Validator 模式为快速失败返回
     */
    @Bean
    public MethodValidationPostProcessor methodValidationPostProcessor() {
        MethodValidationPostProcessor postProcessor = new
MethodValidationPostProcessor();
        postProcessor.setValidator(validator());
        return postProcessor;
    }

}
```

lombok自动注入：

使用@RequiredArgsConstructor(onConstructor_ = {@Autowired})注解代替@Autowired

比如上图的service注入就用lombok注解代替了@Autowired

命名规范：

数据库表命名规范_数据库表名-CSDN博客

cloud.tencent.com

3.3 其它加分项内容

该部分内容可在做完上述作业和加分项后进行学习，也可在做项目时提供帮助：

1. 单元测试：可以单独测试单一模块
2. ER图：在数据库设计和实体类设计上有帮助
3. 反射与AOP
4. Redis：可以使用redis存储登陆注册时的token
5. 前端页面等

3.4 补充说明

关于后端第一次作业的几点补充说明：

1. 日记的文本内容用数据库字段存储即可。
2. Java 的持久层框架可以用 mybatis / mybatis-plus / jpa 任意一种均可
3. Java 登录接口需要的返回内容 和 参考的 BaseResponse类不太一样，以前者为准，后者仅供参考。
4. 没有交报名表的同学也可以交作业参与后续纳新，若到截止时间来不及提交也可以联系我适当延期。
5. Java ER图的参考画法：https://blog.csdn.net/WHEgqing/article/details/108998283

4. 项目要求（Golang）

此处作者 @21-刘梓竣 来自 Golang方向第二次作业

4.1 数据库与ORM

选择一种数据库（推荐MySQL或PostgreSQL）在本机安装。

https://www.mysql.com/cn/

MySQL

Skip to Main Content 全球广受欢迎的开源数据库 联系 MySQL | 登录 | 注册 MySQL.com 下载 文档 开发人员专区 开发人员专区 文档 下载 MySQL.com 产品 云 服务 合作伙伴 客户 为何选择 MySQL? 新闻和活动 如何购买 下载 文档 开发人员专区 Section Menu…

https://www.postgresql.org

PostgreSQL

The world's most advanced open source database.

任选一种数据库可视化工具（推荐DBeaver Community〔无需破解〕，其他有Navicat〔破解方法请参考Java作业及群文件〕、DataGrip等）尝试连接数据库。

https://dbeaver.io

dbeaver.io

任选一种Go语言ORM框架（推荐GORM）尝试连接到数据库，并了解用法（下方**GORM指南**链接是安装教程，**连接到数据库**链接是测试指南）。

https://gorm.io/zh_CN/docs/index.html

GORM 指南

The fantastic ORM library for Golang aims to be developer friendly. 特性 全功能 ORM 关联 (Has One，Has Many，Belongs To，Many To Many，多态，单表继承) Create，Save，Update，Delete，Find 中钩子方法 支持 Preload、Joins 的预加载 事···

GORM https://gorm.io/zh_CN/docs/connecting_to_the_database.html

连接到数据库

GORM 官方支持的数据库类型有：MySQL, PostgreSQL, SQLite, SQL Server 和 TiDB MySQLimport ("gorm.io/driver/mysql" "gorm.io/gorm")func main() { // 参考 https://github.com/go-sql-driver/mysql

4.2 Web框架

任选一种Go语言HTTP Web框架（推荐Gin）并尝试运行一个HTTP服务（下方**快速入门**链接是安装教程及测试指南）。

https://gin-gonic.com/zh-cn/docs/quickstart/

快速入门

Gin Web Framework

任选一种HTTP接口测试工具进行安装并测试HTTP接口（不做推荐，即使使用终端命令测试也是可以的）。

https://www.postman.com

Postman API Platform | Sign Up for Free

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better ···

https://www.apipost.cn

Apipost-API 文档、设计、调试、自动化测试一体化协作平台

Apipost是集API设计、API调试、API文档、自动化测试为一体的API研发协同平台,支持grpc,http,websocket,socketio,socketjs类型接口调试,支持私有化部署。

最后使用选择好并测试过的框架搭建一个HTTP Web服务完成本次作业。

你可以自行选择加入其他可以方便代码编写的框架，例如配置管理框架Viper。

https://www.liwenzhou.com/posts/Go/viper/

Go语言配置管理神器——Viper中文教程 - 李文周的博客

李文周的Blog viper 配置管理 go config yaml json etcd Consul toml 配置 热加载

4.3 安全框架

了解JWT的原理，任选一种安全框架（推荐jwt-go），实现使用JWT进行用户登录并保证接口安全（例：用户A只能修改用户A的数据，不能修改用户B的数据，这需要调用接口时传入登录时生成的token）。

https://golang-jwt.github.io/jwt/

Getting Started

Getting started with golang-jwt/jwt

https://www.liwenzhou.com/posts/Go/jwt_in_gin/

在gin框架中使用JWT - 李文周的博客

李文周的Blog JWT OAuth json web token 前后端分离 认证 Bearer middleware 中间件

4.4 接口文档

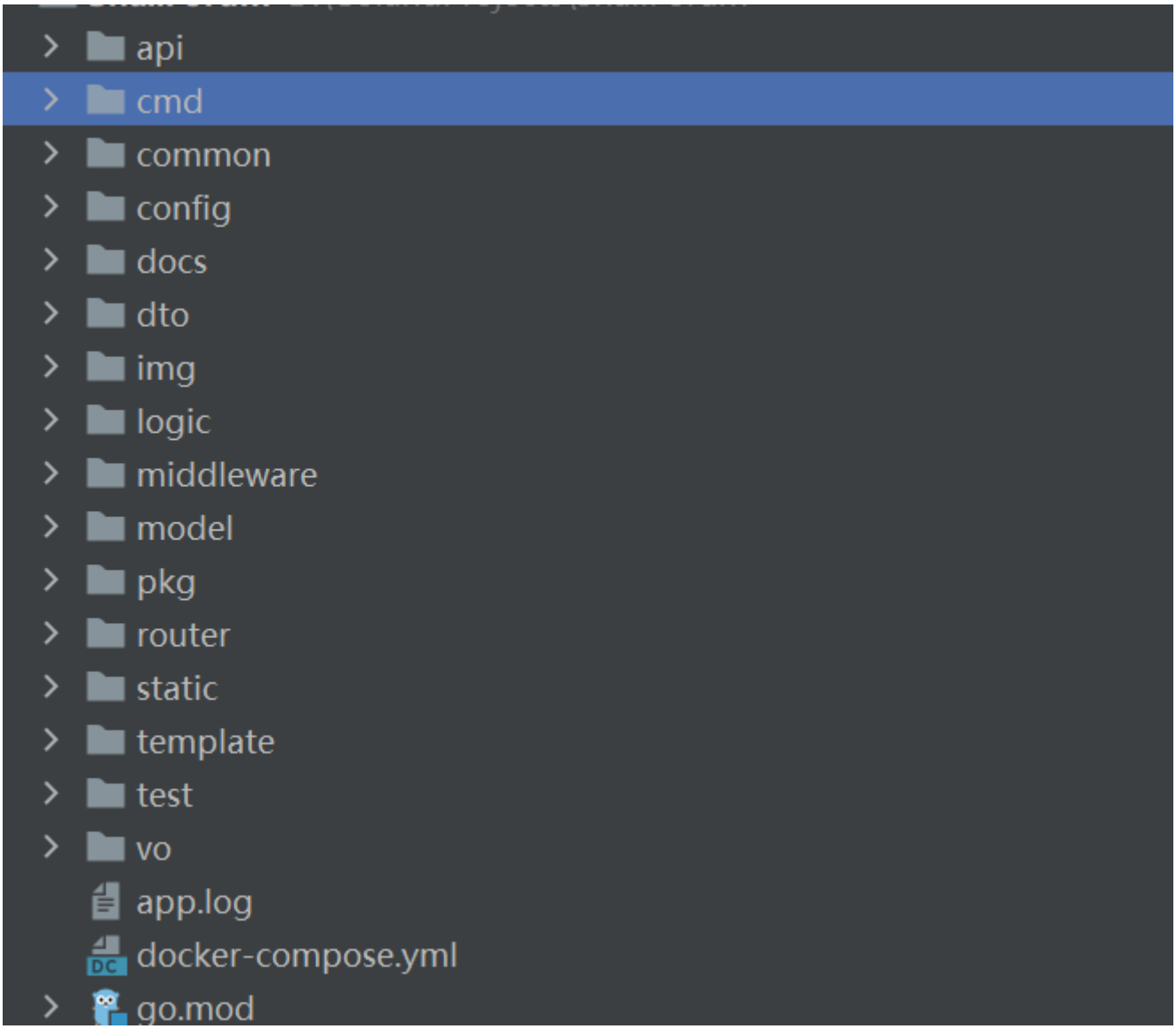
可以选用任意一种方式实现接口文档。可以自己手写，也可以使用Postman生成，或者使用Swagger等。以便于后续与前端的接口对接。

4.5 项目目录结构规范

go的项目没有像java项目使用springboot框架对项目结构有着严格的分层要求，但项目最好还是要有清晰的分层并最好能够遵循规范。

参考链接：https://makeoptim.com/golang/standards/project-layout/

举例：



4.6 test函数

完善的测试体系，能够提高开发的效率，当项目足够复杂的时候，想要保证尽可能的减少 bug，有两种有效的方式分别是代码审核和测试，Go语言中提供了 testing 包来实现单元测试功能。可以在项目中适当添加一些test函数，当然这不作为强制要求。

4.7 Mock(加分项)

不作为强制要求，可以了解一下因为在实际生成环境中最耗时与麻烦的是对代码进行测试。

学习链接：https://geektutu.com/post/quick-gomock.html

4.8 学习资料

Golang方向第一次作业

Golang方向第二次作业

地鼠文档：http://www.topgoer.cn

知识星球：https://www.golangroadmap.com/

Go程序员面试宝典：https://golang.design/go-questions/

Golang修养之路：https://www.yuque.com/aceld/golang

Go effective:https://go.dev/doc/effective_go

Go guide:https://github.com/uber-go/guide/blob/master/style.md

不知名博客 by 魏鹏宇 ：http://bearsattack.top:8090/archives/go

另一个不知名博客：http://zxlmdonnie.cn/archives/goIntro

微信公众号：小徐先生的编程世界

极客兔兔：https://geektutu.com/

书籍：

《Go并发编程实战》（搜不到可以把pdf发群里）

5. 提交要求

提交：项目文件 + 数据库sql文件 + 接口文档（Java需要内含**Swagger截图**）+ QQ号；

> 数据库sql文件如导出有困难，可以截图“设计表”部分查看数据库结构，如表名、字段名、类型、长度、注释等。

将程序源文件以及附录通过**打压缩包**发送到邮箱 werun_backend@163.com，邮件名为 学号-姓名-第一次作业-Java/Golang语言

截止时间：小学期第四周结束（8月4日）