项目设计文档

文档修改历史

修改人员	日期	修改原因	版本号
楼澜	2022.2.27	创建了项目设计文档	v0.1
楼澜	2022.2.28	完成2.1 分层架构图和3.2 物理包图	v0.2
徐琪	2022.2.28	完成1.引言和2.2 逻辑包图	v0.3
徐琪	2022.3.1	完成4.1 模块的职责	v0.4
楼澜	2022.3.2	完成5 信息视角和4.2.3 网络数据模块的分解	v0.5
徐琪	2022.3.2	完成4.2.2 业务逻辑模块的分解	v0.6
蒲中正	2022.3.2	完成4.2.1 用户界面模块的分解	v0.7
徐琪	2022.3.4	完成迭代一项目设计文档	v1.0
徐琪	2022.3.20	根据迭代二需求，修改迭代一部分设计	v1.1
徐琪	2022.3.22	修改文档结构	v1.2
楼澜	2022.3.25	完善 4.2.1 用户界面模块的分解，添加 3.3 物理部署	v1.3
徐琪	2022.4.1	添加迭代二对应的剩余内容	v1.4
楼澜	2022.4.1	添加pipeline流水线	v2.0
徐琪	2022.5.8	根据迭代三需求，修改部分设计	v2.1
徐琪	2022.5.20	添加迭代三相关的业务逻辑层和网络数据层内容	v2.2
楼澜	2022.5.22	修改用户界面相关接口	v2.3
徐琪	2022.5.25	添加信息视角，完成迭代三项目设计文档	v3.0

1. 引言

1.1 编制目的

本文提供COLLECT的软件架构概览，采用若干架构师图描述系统的不同方面，以便表示构造系统所需要的重要架构决策。

1.2 对象与范围

本文档的读者是COLLECT团队内部的开发和管理人员，参考了RUP的《软件架构文档模版》，用于指导下一循环的代码开发和测试工作。

1.3 参考资料

《软件需求规格说明书》

《软件架构文档模版》

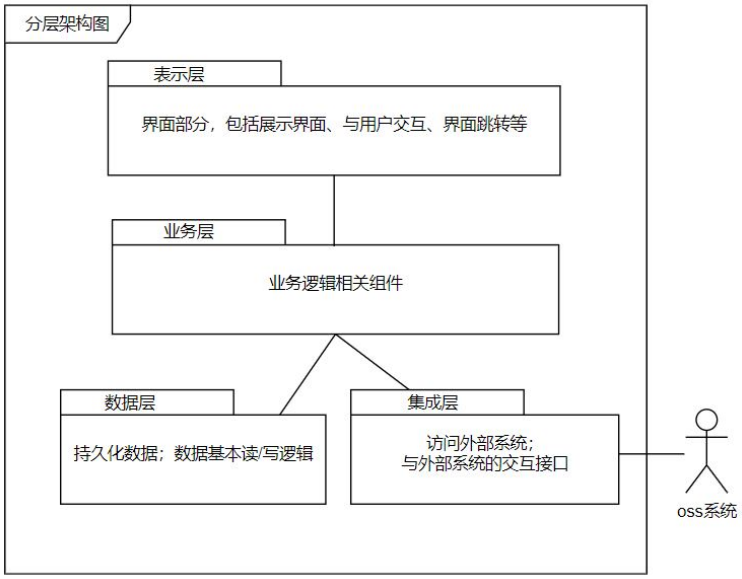
1.4 名词与术语

COLLECT：协作式众包测试平台（Collaborative Crowdsourced Testing Platform）

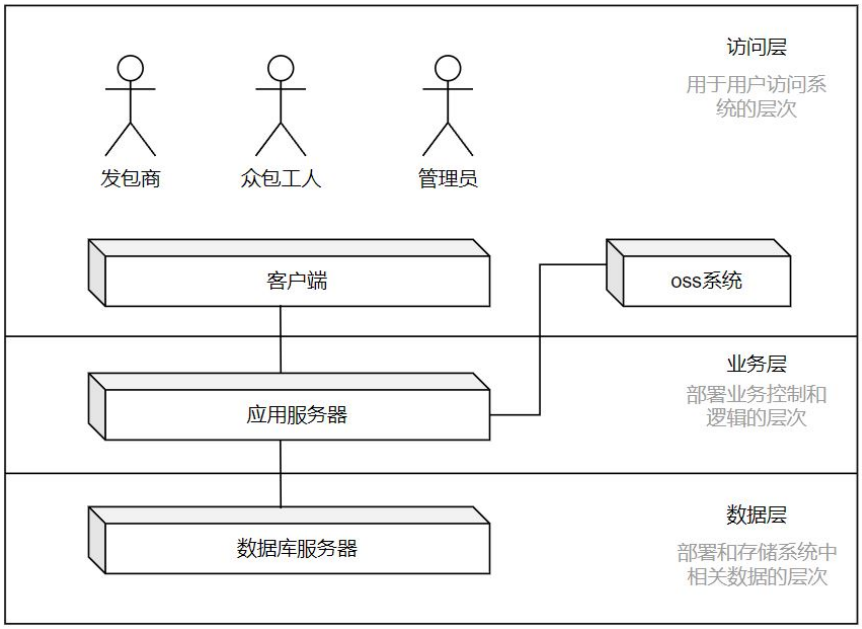
2. 逻辑视角

2.1 分层架构图

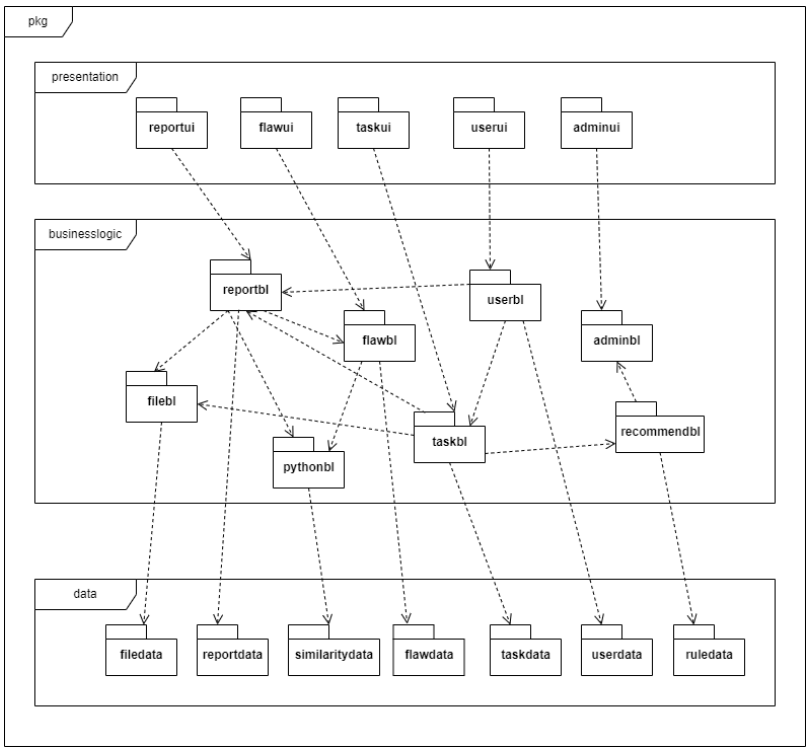
2.1.1 逻辑分层架构图



2.1.2 物理分层架构图



2.2 逻辑包图

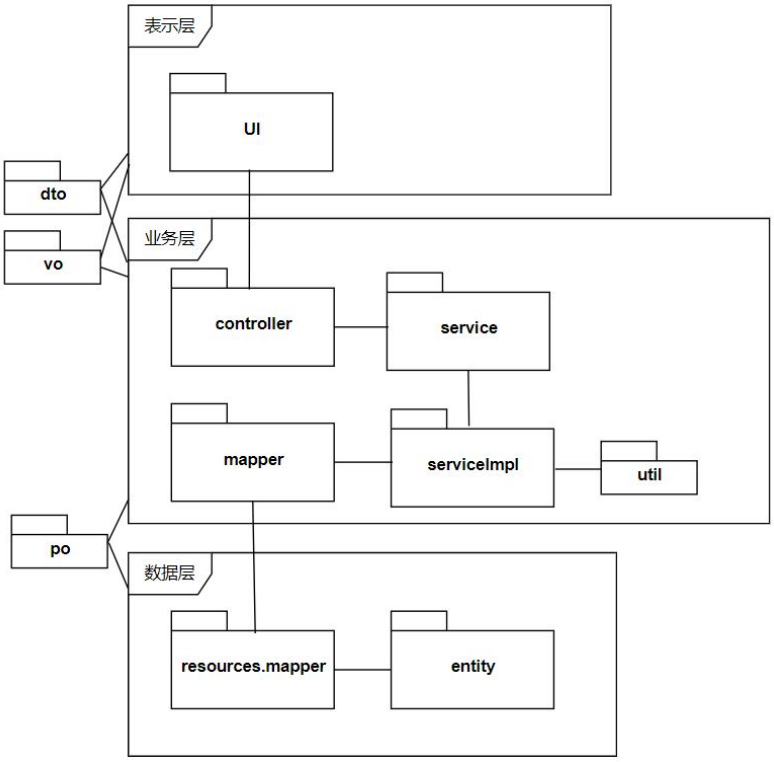


3. 组合视角

3.1 物理包的划分

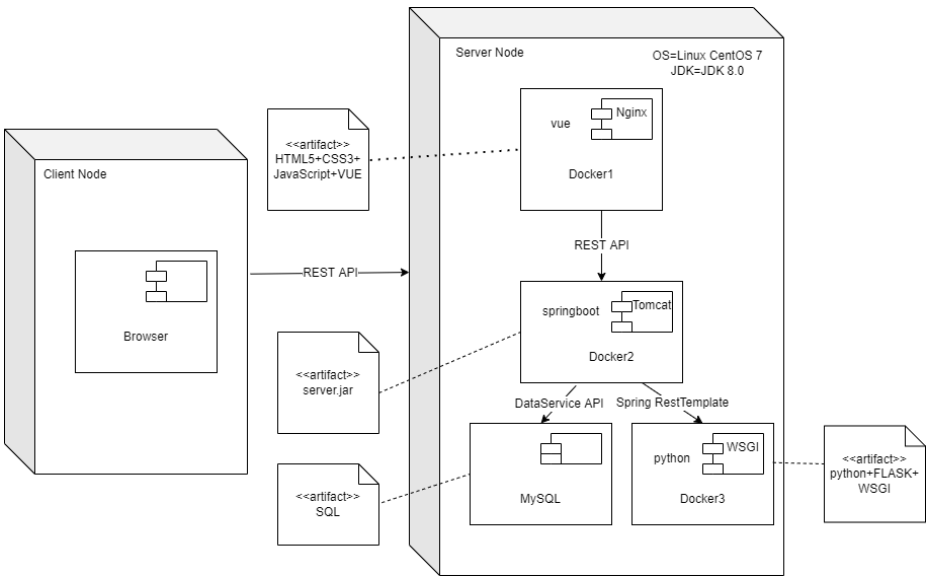
开发包	依赖的其他开发包
UI	controller, model.dto, model.vo
config	
model.dto	
model.vo	model.po
model.po	model.dto
controller	service
service	serviceimpl
serviceimpl	config, mapper, util
mapper	model.po, resources.mapper
resources.mapper	entity
entity	
util	

3.2 物理包图



- UI对象：负责处理系统数据的展现和用户的交互
- controller对象：负责获取用户输入，并调用service模块的服务
- service对象：负责提供服务的抽象接口，获取从数据端组装好的数据
- serviceImpl对象：负责对于抽象接口的实现模块
- mapper对象：负责封装从数据层获取批量数据的接口
- resources.mapper对象：负责与数据库实体交互，获取数据
- entity对象：数据库实体
- util对象：辅助业务层完成业务
- dto对象：表示层传输至业务层的数据对象
- vo对象：业务层传输至表示层的数据对象
- po对象：业务层和数据层之间传输的数据对象

3.3 物理部署



左边是 client 端，即用户的浏览器；右边是 server 端，即我们的产品。

我们的产品由四个服务组成：Docker1（前端）、Docker2（后端）、MySQL（数据库，未用docker起）、Docker3（python算法）。

Docker1 对外和 client 端交互，并且通过 REST API 调用 Docker2 。Docker2 会调用MySQL，并通过 Spring RestTemplate 与 Docker3 交互。

4. 接口视角

4.1 模块的职责

4.1.1 用户界面模块的职责

模块	职责
User	负责用户注册登录、查看个人信息等相关功能
Task	负责测试任务查看、发布、选取等相关功能
Report	负责测试报告提交、查看等相关功能
File	负责文件上传、下载等相关功能
Flaw	负责缺陷展示、完善、评分等相关功能
Admin	负责查看所有任务、编辑推荐规则相关任务

4.1.2 业务逻辑模块的职责

模块	职责
UserService	负责实现用户相关所需要的服务
AdminService	负责实现管理员相关所需要的服务
FileService	负责实现文件上传相关所需要的服务
ReportService	负责实现测试报告相关所需要的服务
TaskService	负责实现任务相关所需要的服务
FlawService	负责实现缺陷相关所需要的服务
PythonService	负责实现调用Python服务器提供的相关服务
RecommendService	负责实现推荐算法相关所需要的服务

4.1.3 网络数据模块的职责

模块	职责
UserMapper	持久化数据库的接口，提供用户相关所需要的服务
UserTagMapper	持久化数据库的接口，提供用户标签相关所需要的服务
TaskMapper	持久化数据库的接口，提供任务相关所需要的服务
TaskTagMapper	持久化数据库的接口，提供任务标签相关所需要的服务
TaskUserMapper	持久化数据库的接口，提供众包工人选择任务相关所需要的服务
ReportMapper	持久化数据库的接口，提供报告相关所需要的服务
FlawMapper	持久化数据库的接口，提供报告中缺陷相关所需要的服务
FlawPicMapper	持久化数据库的接口，提供缺陷对应的图片url相关所需要的服务
EvaluationMapper	持久化数据库的接口，提供缺陷评价相关所需要的服务
ScoreMapper	持久化数据库的接口，提供缺陷评分相关所需要的服务
SimilarityMapper	持久化数据库的接口，提供缺陷相似度相关所需要的服务
RecommendRuleMapper	持久化数据库的接口，提供推荐规则相关所需要的服务
RecommendRuleFactorMapper	持久化数据库的接口，提供推荐规则因子相关所需要的服务
UserSimilarityMapper	持久化数据库的接口，提供用户相似度相关所需要的服务
MultiObjectiveRecommendFactorMapper	持久化数据库的接口，提供多目标优化推荐因素所需要的相关服务
MultiObjectiveRecommendResultMapper	持久化数据库的接口，提供多目标优化推荐结果所需要的相关服务
TaskRecruitStopRecommendFactorMapper	持久化数据库的接口，提供停止招募所需要的相关服务
WorkerAbilityMapper	持久化数据库的接口，提供工人能力所需要的相关服务
WorkerContextMapper	持久化数据库的接口，提供工人测试环境所需要的相关服务
EvaLikeMapper	持久化数据库的接口，提供缺陷评论点赞或点踩所需要的相关服务

4.1.3 层之间调用接口

接口	服务调用方	服务提供方
AdminController	客户端展示层	服务端业务逻辑层
FileController	客户端展示层	服务端业务逻辑层
ReportController	客户端展示层	服务端业务逻辑层
TaskController	客户端展示层	服务端业务逻辑层
UserController	客户端展示层	服务端业务逻辑层
FlawController	客户端展示层	服务端业务逻辑层
UserMapper	服务端业务逻辑层	服务端数据层
UserTagMapper	服务端业务逻辑层	服务端数据层
TaskMapper	服务端业务逻辑层	服务端数据层
TaskTagMapper	服务端业务逻辑层	服务端数据层
TaskUserMapper	服务端业务逻辑层	服务端数据层
ReportMapper	服务端业务逻辑层	服务端数据层
FlawMapper	服务端业务逻辑层	服务端数据层
FlawPicMapper	服务端业务逻辑层	服务端数据层
EvaluationMapper	服务端业务逻辑层	服务端数据层
ScoreMapper	服务端业务逻辑层	服务端数据层
SimilarityMapper	服务端业务逻辑层	服务端数据层
RecommendRuleMapper	服务端业务逻辑层	服务端数据层
RecommendRuleFactorMapper	服务端业务逻辑层	服务端数据层
UserSimilarityMapper	服务端业务逻辑层	服务端数据层
MultiObjectiveRecommendFactorMapper	服务端业务逻辑层	服务端数据层
MultiObjectiveRecommendResultMapper	服务端业务逻辑层	服务端数据层
TaskRecruitStopRecommendFactorMapper	服务端业务逻辑层	服务端数据层
WorkerAbilityMapper	服务端业务逻辑层	服务端数据层
WorkerContextMapper	服务端业务逻辑层	服务端数据层
EvaLikeMapper	服务端业务逻辑层	服务端数据层

4.2 模块的接口规范

4.2.1 用户界面模块的分解

4.2.1.1 User

需要的服务(需接口)

服务名	服务
UserService.userRegister(UserDTO user)	用户注册
UserService.userLogin(String username, String password)	用户登录
UserService.viewUser(Integer userId)	获取用户详细信息
UserService.getUserInfo(Integer userId)	获取用户简略信息
UserService.getUserInfoList(List<Integer> userIds)	获取一组用户简略信息
UserService.viewFinishedTasks(Integer userId, Integer pageId)	获取用户已完成任务
UserService.viewUnfinishedTasks(Integer userId, Integer pageId)	获取用户未完成任务
UserService.viewExpiredTasks(Integer userId, Integer pageId)	获取用户已逾期任务
UserService.viewReportsFromTask(Integer userId, Integer taskId, Integer pageId)	获取某一任务的报告列表
UserService.viewWorkerAllReports(Integer userId, Integer pageId)	预览自己的所有报告
UserService.viewWorkerTaskReport(Integer userId, Integer taskId)	获取某一任务中自己的报告的缩略信息
UserService.addUserTag(Integer userId, Integer tag)	众包工人添加个人tag
UserService.deleteUserTag(Integer userId, Integer tag)	众包工人删除个人tag
UserService.getWorkerContext(Integer userId)	获取工人测试环境
UserService.updateWorkerContext(Integer userId, WorkerContextDTO workerContextDTO)	更新工人测试环境
UserService.getWorkerRadar(Integer userId)	获取工人雷达图
UserService.getWorkerAvgRadar()	获取工人平均雷达图
UserService.getWorkerActiveness(Integer userId)	获取工人活跃度
UserService.getWorkerWordCloud(Integer userId)	获取工人词云

4.2.1.2 Task

需要的服务(需接口)

服务名	服务
TaskService.viewTaskDetails(Integer taskId, Integer userId)	获取任务详情信息
TaskService.pickTask(Integer taskId, Integer userId)	众包工人接受任务
TaskService.createTask(TaskDTO taskDTO)	创建任务
TaskService.viewAllUnfinishedTasks(Integer pageId)	获取任务广场中的任务
TaskService.getReportId(Integer taskId, Integer userId)	当用户想从任务界面去完善报告时，需要知晓报告的id
TaskService.getRecommendElements()	发包方查看可选的带默认权重的推荐元素列表
TaskService.updateRecommendElementsWeights(Integer taskId,List<RecommendRuleFactorVO> recommendRuleFactorVOs)	发包方更新此任务的推荐权重
TaskService.getRecommendElementsWeights(Integer taskId)	发包方查看此任务的推荐权重
TaskService.getCurrentTaskRadar(Integer taskId)	发包方查看此任务的当前工人的雷达图值，包括工人能力、活跃度、相关性、多样性
TaskService.getTargetTaskRadar(Integer taskId)	发包方查看此任务的目标雷达图值，包括工人能力、活跃度、相关性、多样性
TaskService.stopTaskRecruit(Integer taskId)	发包方手动停止招募
TaskService.getFlawCurveAndNumPredicted(Integer taskId)	"发包方查看此任务的新缺陷发现曲线及预测的总缺陷数

4.2.1.3 File

需要的服务(需接口)

服务名	服务
FileService.policy()	获取向OSS服务器直传文件所需要的签名

4.2.1.4 Report

需要的服务(需接口)

服务名	服务
ReportService.viewReportDetails(Integer reportId);	用户选定任一报告查看报告详情
ReportService.createReport(ReportDTO reportDTO)	众包工人填写测试报告表单提交测试报告
ReportService.getToBeRefinedFlawLists(Integer reportId)	众包工人获取自己提交的报告所有待完善的缺陷列表

4.2.1.5 Admin

需要的服务(需接口)

服务名	服务
AdminService.getAllTasks(Integer pageId)	获取所有任务
AdminService.viewTaskDetails(Integer taskId)	查看任务详细信息
AdminService.getRecommendRules()	获取所有现存的推荐规则
AdminService.getRecommendRuleFactors()	获取影响推荐策略的因素
AdminService.deleteRecommendRule(Integer ruleId)	删除推荐规则
AdminService.addRecommendRule(RecommendRuleDTO recommendRuleDTO)	添加推荐规则
AdminService.selectRecommendRule(Integer ruleId)	选择推荐规则

4.2.1.6 Flaw

需要的服务(需接口)

服务名	服务
FlawService.refineFlaw(Integer flawId, Integer forkedFlawId, FlawDTO flawDTO)	完善缺陷
FlawService.noforkFlaw(Integer flawId)	完善缺陷时保留原缺陷
FlawService.getFlawTree(Integer flawId)	获取缺陷树
FlawService.getSimilarFlaws(Integer flawId)	获取与当前缺陷相似度高的缺陷列表
FlawService.getFlawScore(Integer flawId, Integer userId)	获取当前缺陷评分
FlawService.scoreFlaw(Integer flawId, Integer userId, Integer score)	为当前缺陷评分
FlawService.getTaskFlawMap(Integer taskId)	获取当前任务缺陷图
FlawService.getAllEvaluations(Integer flawId)	获取当前缺陷所有评论
FlawService.addEvaluation(Integer flawId, FlawEvaluationDTO flawEvaluationDTO)	给此缺陷增加评论
FlawService.addAppendContent(Integer flawId, String content)	对此缺陷进行补充说明

4.2.2 业务逻辑模块的分解

> 推荐算法更具体的业务逻辑接口实现在 com.seiii.collect.serviceimpl.recommend 中，详细设计见**算法解释文档**
>
> com.seiii.collect.scheduledtasks.recommend 下主要提供定时刷新的服务，详见**算法解释文档**

4.2.2.1 UserService

提供的服务(供接口)		
UserService.userRegister	语法	ResponseVO<Object> userRegister(UserDTO user);
	前置条件	提供正确和完整的用户注册信息
	后置条件	检查数据合法性（包括用户名是否重复），返回是否注册成功
UserService.userLogin	语法	ResponseVO<UserViewVO> userLogin(String username, String password);
	前置条件	提供用户名和密码
	后置条件	根据用户名和密码判断是否成功登陆，若成功登陆，则返回用户简略信息
UserService.getUserInfo	语法	ResponseVO<UserViewVO> getUserInfo(Integer userId);
	前置条件	用户已经登录
	后置条件	返回根据用户Id查询到的用户简略信息
UserService.getUserInfoList	语法	ResponseVO<List<UserViewVO>> getUserInfoList(List<Integer> userIds);
	前置条件	用户已经登录
	后置条件	返回根据用户Id列表对应的用户简略信息的列表
UserService.viewUser	语法	ResponseVO<UserVO> viewUser(Integer userId);
	前置条件	用户已经登录
	后置条件	返回根据用户Id返回用户详细信息

UserService.viewFinishedTasks	语法	ResponseVO<List<TaskViewVO>> viewFinishedTasks(Integer userId, Integer pageId);
	前置条件	用户已经登录
	后置条件	返回根据用户Id和页数查询的已完成任务
UserService.viewUnfinishedTasks	语法	ResponseVO<List<TaskViewVO>> viewUnfinishedTasks(Integer userId, Integer pageId);
	前置条件	用户已经登录
	后置条件	返回根据用户Id和页数查询的未逾期且未完成任务
UserService.viewExpiredTasks	语法	ResponseVO<List<TaskViewVO>> viewExpiredTasks(Integer userId, Integer pageId);
	前置条件	用户已经登录
	后置条件	返回根据用户Id和页数查询的逾期未完成任务
UserService.viewReportsFromTask	语法	ResponseVO<List<ReportViewVO>> viewReportsFromTask(Integer userId, Integer taskId, Integer pageId);
	前置条件	用户已经登录，并且拥有此任务
	后置条件	返回任务中的报告列表
UserService.viewWorkerAllReports	语法	ResponseVO<List<ReportViewVO>> viewWorkerAllReports(Integer userId, Integer pageId);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	根据用户id返回工人自己的所有报告列表
UserService.viewWorkerTaskReport	语法	ResponseVO<ReportViewVO> viewWorkerTaskReport(Integer userId, Integer taskId);

	前置条件	用户已经登录，并且类型为众包工人
	后置条件	根据用户id和任务id返回工人在该任务下的报告缩略图
UserService.addUserTag	语法	ResponseVO<Object> addUserTag(Integer userId, Integer tag);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	存储该用户选择添加的标签
UserService.deleteUserTag	语法	ResponseVO<Object> deleteUserTag(Integer userId, Integer tag);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	删除该用户选择的标签
UserService.getWorkerContext	语法	ResponseVO<WorkerContextVO> getWorkerContext(Integer userId);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	返回工人的测试环境
UserService.updateWorkerContext	语法	ResponseVO<WorkerContextVO> updateWorkerContext(Integer userId, WorkerContextDTO workerContextDTO);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	保存用户更新并返回
UserService.getWorkerRadar	语法	ResponseVO<WorkerRadarVO> getWorkerRadar(Integer userId);

	前置条件	用户已经登录，并且类型为众包工人
	后置条件	返回工人对应的雷达图
UserService.getWorkerAvgRadar	语法	ResponseVO<WorkerRadarVO> getWorkerAvgRadar();
	前置条件	无
	后置条件	返回所有工人的平均雷达图
UserService.getWorkerActiveness	语法	ResponseVO<WorkerActivationVO> getWorkerActiveness(Integer userId);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	返回工人对应的活跃度指数
UserService.getWorkerWordCloud	语法	ResponseVO<WorkerCloudVO> getWorkerWordCloud(Integer userId);
	前置条件	用户已经登录，并且类型为众包工人
	后置条件	返回用户对应的词云
UserService.refreshWorkerAbility	语法	ResponseVO<Object> refreshWorkerAbility();
	前置条件	无
	后置条件	定期刷新工人能力

需要的服务(需接口)

服务名	服务
userMapper.selectByPrimaryKey(Integer id)	系统根据关键字查询用户
userMapper.selectByUsername(String username)	系统根据用户名查找对应用户
userMapper.insert(User record)	系统记录用户的详细信息
taskMapper.selectByIds(List<Integer> taskIds)	系统根据关键字列表查询任务
taskMapper.selectByUserIdAndAfterEndTime(Integer userid, Date curtime, Integer start, Integer pagesize)	系统根据用户关键字、当前时间、起始条数、需要条数查询当前用户已经结束的部分任务
taskMapper.selectByUserIdAndBeforeEndTime(Integer userid, Date curtime, Integer start, Integer pagesize)	系统根据用户关键字、当前时间、起始条数、需要条数查询当前用户未结束的部分任务
taskUserMapper.selectByUserId(Integer userId)	系统根据用户关键字查询此用户领取的任务关键字列表
reportMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查找对应的报告列表
reportMapper.selectByUserIdAndPage(Integer userId,Integer start,Integer pagesize)	系统根据用户关键字、起始条数、需要条数查询当前用户的部分报告
reportMapper.selectByUserId(Integer userId)	系统根据用户关键字查询当前用户的所有报告
userTagMapper.insert(UserTag record)	系统记录用户和标签的关联记录
userTagMapper.selectByUserId(Integer userId)	系统根据用户关键字查询用户对应的标签组
userTagMapper.deleteByPrimaryKey(Integer userId, Integer tag)	系统根据用户关键字和标签删除关联记录
taskTagMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查询任务对应的标签组
flawMapper.selectByReportId(Integer reportId)	系统根据报告关键字查询对应的缺陷列表
workerContextMapper.selectByPrimaryKey(Integer userId)	系统根据用户关键字查询对应的测试环境
workerContextMapper.updateByPrimaryKey(WorkerContext record)	系统根据用户环境更新对应记录
pythonService.getTFIDF(List<String> list)	计算列表的TFIDF值
comprehensiveAbility.calAllAbilityValue(Integer userId)	计算工人的各个能力值
workerAbilityMapper.updateByPrimaryKey(WorkerAbility record)	系统根据工人能力更新对应记录

4.2.2.2 AdminService

提供的服务(供接口)		
AdminService.getAllTasks	语法	ResponseVO<List<TaskViewVO>> getAllTasks(Integer pageId);
	前置条件	管理员登陆
	后置条件	返回所有任务的简略信息列表
AdminService.viewTaskDetails	语法	ResponseVO<TaskVO> viewTaskDetails(Integer taskId);
	前置条件	选取某条任务
	后置条件	返回该任务的详细信息
AdminService.getRecommendRules	语法	ResponseVO<List<RecommendRuleVO>> getRecommendRules();
	前置条件	管理员已登陆
	后置条件	返回推荐规则的列表
AdminService.getRecommendRuleFactors	语法	ResponseVO<List<RecommendRuleFactorVO>> getRecommendRuleFactors();
	前置条件	管理员已登陆
	后置条件	返回推荐影响因素的列表
AdminService.deleteRecommendRule	语法	ResponseVO<Object> deleteRecommendRule(Integer ruleId);
	前置条件	选取需要删除某条推荐规则
	后置条件	删除选中的推荐规则

AdminService.addRecommendRule	语法	ResponseVO<List<RecommendRuleVO>> addRecommendRule(RecommendRuleDTO recommendRuleDTO);
	前置条件	正确填写推荐规则信息
	后置条件	添加该条推荐规则
AdminService.selectRecommendRule	语法	ResponseVO<Object> selectRecommendRule(Integer ruleId);
	前置条件	选取需要启用的某条推荐规则
	后置条件	启用该条推荐规则

需要的服务(需接口)

服务名	服务
taskMapper.selectAllByPageId(Integer start, Integer pagesize)	根据目前页数返回对应任务预览的记录
taskMapper.selectByPrimaryKey(Integer id)	系统根据关键字查询任务信息
taskUserMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查询选择该任务的用户列表
taskTagMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查询该任务的标签组
recommendRuleMapper.selectAll()	系统查询所有的推荐规则列表
recommendRuleMapper.deleteByPrimaryKey(Integer ruleId)	系统根据规则关键字删除对应的规则
recommendRuleMapper.selectByName(String name)	系统根据规则名称查找对应的规则
recommendRuleMapper.insert(RecommendRule record)	系统记录规则的详细信息
recommendRuleMapper.updateAllOff()	系统更新所有的规则状态为未被使用
recommendRuleMapper.updateOneOn(Integer ruleId)	系统根据规则关键字将该规则状态设为使用中
recommendRuleFactorMapper.selectByRuleId(Integer ruleId)	系统根据规则关键字查询该规则的影响因子列表

4.2.2.3 FileService

提供的服务(供接口)		
FileService.policy	语法	ResponseVO<OSSPolicyVO> policy();
	前置条件	无
	后置条件	返回上传policy和回调设置
FileService.callback	语法	ResponseVO<OSSCallbackResultVO> callback(String filename, String size, String mimeType, String width, String height);
	前置条件	oss服务器根据用户的回调设置发送回调请求
	后置条件	返回响应给oss服务器

需要的服务(需接口)

服务名	服务
OSSClient.generatePostPolicy(Date expiration, PolicyConditions conds)	生成Post请求的policy表单域
OSSClient.calculatePostSignature(String postPolicy)	根据Access Key Secret和policy计算签名，OSS依据该签名验证Post请求的合法性

4.2.2.4 ReportService

提供的服务(供接口)		
ReportService.viewReportDetails	语法	ResponseVO<ReportVO> viewReportDetails(Integer reportId);
	前置条件	选取某条报告
	后置条件	返回该报告对应的详细信息
ReportService.createReport	语法	ResponseVO<ReportVO> createReport(ReportDTO reportDTO);
	前置条件	提供正确和完整的测试报告信息
	后置条件	返回测试报告的详细信息
ReportService.getToBeRefinedFlawLists	语法	ResponseVO<List<TBAFlawVO>> getToBeRefinedFlawLists(Integer reportId);
	前置条件	已经通过请求体发布该报告
	后置条件	根据相似度，返回报告中待修改的缺陷列表和其相似缺陷列表

需要的服务(需接口)

服务名	服务
reportMapper.insert(Report record)	系统记录测试报告的详细信息
reportMapper.selectByPrimaryKey(Integer id)	系统根据关键字查询测试报告信息
reportMapper.selectByUserId(Integer userId)	系统根据用户关键字查询该用户提交的报告列表
flawMapper.selectByReportId(Integer reportid)	系统根据测试报告关键字查询缺陷列表
flawMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查询该任务下的所有缺陷列表
flawMapper.insert(Flaw record)	系统记录缺陷的详细信息
flawMapper.updateByPrimaryKey(Flaw record)	系统根据缺陷关键字更新缺陷信息
flawPicMapper.selectByFlawId(Integer flawid)	系统根据缺陷id查询图片列表
flawPicMapper.insert(FlawPic record)	系统记录缺陷和对应图片的关联信息
pythonService.getSimilarity(List<String> contents)	根据内容的列表返回第一个元素和所有元素的相似度值的列表
similarityMapper.insert(Similarity similarity)	系统记录缺陷之间的相似度

4.2.2.5 TaskService

提供的服务(供接口)		
TaskService.viewAllUnfinishedTasks	语法	ResponseVO<List<TaskViewVO>> viewAllUnfinishedTasks(Integer pageId)
	前置条件	无
	后置条件	按照pageId返回相应页数的任务简略信息的列表
TaskService.viewTaskDetails	语法	ResponseVO<TaskVO> viewTaskDetails(Integer taskId, Integer userId)
	前置条件	选取某条任务
	后置条件	返回该任务的详细信息
TaskService.createTask	语法	ResponseVO<TaskVO> createTask(TaskDTO taskDTO)
	前置条件	填写完整和正确的任务信息
	后置条件	返回任务的详细信息
TaskService.pickTask	语法	ResponseVO<TaskVO> pickTask(Integer taskId, Integer userId)
	前置条件	用户为众包工人
	后置条件	返回该用户选择的任务的详细信息
TaskService.getReportId	语法	ResponseVO<Integer> getReportId(Integer taskId, Integer userId);
	前置条件	用户为众包工人，且选择了该任务，并且有已经提交的报告
	后置条件	返回该用户在该任务下提交的报告的id

TaskService.getRecommendElements	语法	ResponseVO<List<RecommendRuleFactorVO>> getRecommendElements();
	前置条件	用户为发包方
	后置条件	返回该可选的带默认权重的推荐元素列表
TaskService.updateRecommendElementsWeights	语法	ResponseVO<List<RecommendRuleFactorVO>> updateRecommendElementsWeights(Integer taskId, List<RecommendRuleFactorVO> recommendRuleFactorVOs);
	前置条件	用户为发包方
	后置条件	更新此任务的推荐权重
TaskService.getRecommendElementsWeights	语法	ResponseVO<List<RecommendRuleFactorVO>> getRecommendElementsWeights(Integer taskId);
	前置条件	用户为发包方
	后置条件	返回此任务的推荐权重
TaskService.getCurrentTaskRadar	语法	ResponseVO<TaskRadarVO> getCurrentTaskRadar(Integer taskId);
	前置条件	用户为发包方
	后置条件	返回此任务的当前工人的雷达图值
TaskService.getTargetTaskRadar	语法	ResponseVO<TaskRadarVO> getTargetTaskRadar(Integer taskId);
	前置条件	用户为发包方
	后置条件	返回此任务的目标雷达图值

TaskService.stopTaskRecruit	语法	ResponseVO<Object> stopTaskRecruit(Integer taskId);
	前置条件	用户为发包方
	后置条件	该任务状态变为停止招募
TaskService.getFlawCurveAndNumPredicted	语法	ResponseVO<TaskFlawDetectionPredictionVO> getFlawCurveAndNumPredicted(Integer taskId);
	前置条件	用户为发包方
	后置条件	返回此任务的新缺陷发现曲线及预测的总缺陷数

需要的服务(需接口)

服务名	服务
taskMapper.selectBeforeEndTime(Date curtime, Integer start, Integer pagesize)	系统根据页数、当前时间返回所有未过结束时间的部分任务预览
taskMapper.insert(Report record)	系统记录任务的详细信息
taskMapper.selectByPrimaryKey(Integer id)	系统根据关键字查询任务信息
userMapper.selectByPrimaryKey(Integer id)	系统根据关键字查询用户信息
taskUserMapper.selectByUserId(Integer userid)	系统根据用户关键字查询用户选取的任务列表
taskUserMapper.selectByTaskId(Integer taskId)	根据任务关键字查询选择该任务的所有用户列表
taskUserMapper.insert(TaskUser record)	系统记录任务和用户的关联信息
reportMapper.selectByUserId(Integer userid)	系统根据用户关键字查询用户提交的报告列表
taskTagMapper.selectByTaskId(Integer taskId)	系统根据任务关键字查询该任务的标签组
taskTagMapper.insert(TaskTag record)	系统记录任务和标签的关联记录
flawMapper.selectByReportId(Integer reportId)	系统根据报告关键字查询报告下的所有缺陷列表
recommendService.getAllRecommendedTasksForUser(Integer userId);	系统根据用户关键字返回推荐给该用户的任务列表
multiObjectiveRecommendFactorMapper.selectByPrimaryKey(Integer taskId)	系统根据任务关键字获取多目标优化推荐因素及其权重
multiObjectiveRecommendFactorMapper.updateByPrimaryKey(MultiObjectiveRecommendFactor record)	系统根据多目标优化推荐因素更新相关记录
multiObjectiveRecommendFactorMapper.insert(MultiObjectiveRecommendFactor record)	系统记录多目标优化推荐因素及其权重
taskRecruitStopRecommendFactorMapper.selectByPrimaryKey(Integer taskId)	系统根据任务关键字返回对应的停止招募推荐的相关因素

4.2.2.6 FlawService

提供的服务(供接口)		
FlawService.refineFlaw	语法	ResponseVO<Object> refineFlaw(Integer flawId, Integer forkedFlawId, FlawDTO flawDTO);
	前置条件	该缺陷存在
	后置条件	更新缺陷信息，存储缺陷的fork关系，更新相似度
FlawService.noforkFlaw	语法	ResponseVO<Object> noforkFlaw(Integer flawId);
	前置条件	该缺陷存在
	后置条件	存储缺陷关系，更新相似度
FlawService.getFlawTree	语法	ResponseVO<FlawTreeNodeVO> getFlawTree(Integer flawId);
	前置条件	对应的缺陷已经处理完毕
	后置条件	返回与该缺陷关联的缺陷树的根节点结构
FlawService.getSimilarFlaws	语法	ResponseVO<List<SimilarFlawVO>> getSimilarFlaws(Integer flawId);
	前置条件	对应的缺陷已经处理完毕
	后置条件	返回与该缺陷相似的缺陷列表
FlawService.getFlawScore	语法	ResponseVO<Integer> getFlawScore(Integer flawId, Integer userId);
	前置条件	对应的缺陷已经处理完毕
	后置条件	返回对应用户对该缺陷的评分
FlawService.scoreFlaw	语法	ResponseVO<Object> scoreFlaw(Integer flawId, Integer userId, Integer score);
	前置条件	对应的缺陷已经处理完毕
	后置条件	存储用户对该缺陷的评分值
FlawService.getTaskFlawMap	语法	ResponseVO<FlawMapVO> getTaskFlawMap(Integer taskId);
	前置条件	该任务存在
	后置条件	返回该任务下的缺陷图结构
FlawService.getAllEvaluations	语法	ResponseVO<List<FlawEvaluationVO>> getAllEvaluations(Integer flawId);
	前置条件	对应的缺陷已经处理完毕
	后置条件	返回与该缺陷下的所有评价
FlawService.addEvaluation	语法	ResponseVO<List<FlawEvaluationVO>> addEvaluation(Integer flawId, FlawEvaluationDTO flawEvaluationDTO);
	前置条件	该缺陷的作者不是该用户

	后置条件	给该缺陷中增加一条评价记录td>
FlawService.addAppendContent	语法	ResponseVO<Object> addAppendContent(Integer flawId, String content);
	前置条件	对应的缺陷已经处理完毕
	后置条件	给该缺陷增加或更新补充内容

需要的服务(需接口)

服务名	服务
flawMapper.selectByTaskId(Integer taskId)	系统根据任务关键字获取该任务下的所有缺陷列表
flawMapper.selectByPrimaryKey(Integer flawId)	系统根据缺陷关键字获取该缺陷的相关信息
flawMapper.updateByPrimaryKey(Flaw record)	系统根据缺陷关键字更新改缺陷的相关信息
flawPicMapper.selectByFlawId(Integer flawId)	系统根据缺陷的关键字查询该缺陷下的所有图片路径
flawPicMapper.deleteByPrimaryKey(Integer flawId, String url)	系统根据缺陷的关键字和图片的路径删除缺陷和图片的关联信息
flawPicMapper.insert(FlawPic flawPic)	系统记录缺陷和图片的关联信息
similarityMapper.insert(Similarity record)	系统记录缺陷相似度相关的信息
similarityMapper.selectByFlawIdCompared(Integer flawId)	系统根据当前缺陷的关键字，查找比此缺陷先插入且有相似度的记录
similarityMapper.selectByFlawIdSecondCompared(Integer flawId)	系统根据当前缺陷的关键字，查找比此缺陷后插入且有相似度的记录
similarityMapper.selectByPrimaryKey(Integer flawId1, Integer flawId2)	系统根据两个缺陷的关键字查询这两个缺陷间的相似度
scoreMapper.selectByPrimaryKey(Integer userId, Integer flawId)	系统根据用户关键字和缺陷关键字查找用户对该缺陷的评分
scoreMapper.insert(Socre record)	系统记录缺陷评分的相关信息
pythonService.getSimilarity(List<String> contents)	根据内容的列表返回第一个元素和所有元素的相似度值的列表
evaluationMapper.selectByFlawId(Integer flawId)	系统根据缺陷关键字获取缺陷的所有评价列表
evaluationMapper.insert(Evaluation record)	系统记录缺陷评价的相关信息
userMapper.selectByPrimaryKey(Integer userId)	系统根据用户关键字获取用户的信息

4.2.2.7 PythonService

提供的服务(供接口)		
PythonService.getSimilarity	语法	ResponseVO<List<Double>> getSimilarity(List<String> contents);
	前置条件	传入需要比较相似度的文本内容列表
	后置条件	返回列表第一个元素和其他元素的相似度值的列表
PythonService.getTFIDF	语法	ResponseVO<Map<String, Double>> getTFIDF(List<String> contents);
	前置条件	传入需要计算TFIDF的文本内容列表
	后置条件	返回计算结果的map表

需要的服务(需接口)

服务名	服务
PythonServer.getSimilarities()	Python微服务提供的获取相似度的服务
PythonServer.getTFIDF()	Python微服务提供的计算TFIDF的服务

4.2.2.8 RecommendService

提供的服务(供接口)		
RecommendService.refreshRecommendationResult	语法	List<Object> refreshRecommendationResult();
	前置条件	无
	后置条件	将推荐结果写入数据库
提供的服务(供接口)		
RecommendService.getAllRecommendedTasksForUser	语法	List<TaskViewVO> getAllRecommendedTasksForUser(Integer userId);
	前置条件	用户存在
	后置条件	根据推荐规则返回推荐给该用户的任务列表

需要的服务(需接口)

服务名	服务
userMapper.selectIdByType(Integer type)	系统查询特定类型的用户
taskRecruitStopRecommendFactorMapper.deleteAll();	系统删除所有任务的已报名众包工人参考信息。
taskRecruitStopRecommendFactorMapperinsertAll(List<TaskRecruitStopRecommendFactor> records);	系统存储所有任务的已报名众包工人参考信息。
multiObjectiveRecommendResultMapper.deleteAll();	系统删除所有的多目标优化结果
MultiObjectiveRecommendResultMapper.insertAll(List<MultiObjectiveRecommendResult> resultList);	系统记录所有的多目标优化结果

4.2.3 网络数据模块的分解

4.2.3.1 UserMapper

提供的服务(供接口)		
UserMapper.selectByPrimaryKey	语法	User selectByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照用户ID进行查找返回相应的User结果
UserMapper.insert	语法	int insert(User record);
	前置条件	相同id的User在Mapper中不存在
	后置条件	在数据库中增加一个User记录
UserMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照用户ID删除相应User记录
UserMapper.selectAll	语法	List<User> selectAll();
	前置条件	无
	后置条件	返回所有用户的记录
UserMapper.selectByUsername	语法	User selectByUsername(String username);
	前置条件	无
	后置条件	通过用户名查找返回用户记录
UserMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(User record);
	前置条件	数据库中已存在改条记录
	后置条件	根据关键字更新该条记录

4.2.3.2 UserTagMapper

提供的服务(供接口)		
UserTagMapper.selectByUserId	语法	List<Integer> selectByUserId(Integer userid);
	前置条件	该用户id存在
	后置条件	按照用户ID进行查找返回相应的tag的列表
UserTagMapper.insert	语法	int insert(UserTag record)
	前置条件	相同id的User在Mapper中不存在
	后置条件	在数据库中增加一个UserTag记录
UserTagMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer userid, Integer tag);
	前置条件	在数据库中存在该id
	后置条件	按照用户ID删除相应UserTag记录
UserTagMapper.selectAll	语法	List<UserTag> selectAll();
	前置条件	无
	后置条件	返回所有用户和标签的关联记录

4.2.3.3 TaskMapper

提供的服务(供接口)		
TaskMapper.selectByPrimaryKey	语法	Task selectByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照任务ID进行查找返回相应的Task记录
TaskMapper.insert	语法	int insert(Task record);
	前置条件	相同id的Task在Mapper中不存在
	后置条件	在数据库中增加一个Task记录
TaskMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id)
	前置条件	在数据库中存在该id
	后置条件	按照任务ID删除相应Task记录
TaskMapper.selectAll	语法	List<Task> selectAll();
	前置条件	无
	后置条件	返回所有任务的记录
TaskMapper.selectAllByPageId	语法	List<TaskView> selectAllByPageId(@Param("start") Integer start, @Param("pagesize") Integer pagesize);
	前置条件	无
	后置条件	根据目前页数返回对应任务预览的记录

TaskMapper.selectByIds	语法	List<TaskView> selectByIds(List ids);
	前置条件	ids一定不能为空，数据库中存在这些id
	后置条件	根据id集合返回相应的任务预览记录
TaskMapper.selectByUserIdAndAfterEndTime	语法	List<TaskView> selectByUserIdAndAfterEndTime(@Param("userid") Integer userid, @Param("curtime")Date curtime, @Param("start") Integer start, @Param("pagesize") Integer pagesize);
	前置条件	无
	后置条件	根据页数、用户ID、当前时间返回当前用户所有且已过结束时间的部分任务预览
TaskMapper.selectByUserIdAndBeforeEndTime	语法	List<TaskView> selectByUserIdAndBeforeEndTime(@Param("userid") Integer userid, @Param("curtime")Date curtime, @Param("start") Integer start, @Param("pagesize") Integer pagesize);
	前置条件	无
	后置条件	根据页数、用户ID、当前时间返回当前用户所有且未过结束时间的部分任务预览
TaskMapper.selectBeforeEndTime	语法	List<TaskView> selectBeforeEndTime(@Param("curtime")Date curtime, @Param("start") Integer start, @Param("pagesize") Integer pagesize);
	前置条件	无
	后置条件	根据页数、当前时间返回所有未过结束时间的部分任务预览

4.2.3.4 TaskTagMapper

提供的服务(供接口)		
TaskTagMapper.selectByTaskId	语法	List<Integer> selectByTaskId(Integer taskid);
	前置条件	该任务id存在
	后置条件	按照任务ID进行查找返回相应的tag的列表
TaskTagMapper.insert	语法	int insert(TaskTag record)
	前置条件	相同id的Task在Mapper中不存在
	后置条件	在数据库中增加一个TaskTag记录
TaskTagMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer taskid, Integer tag);
	前置条件	在数据库中存在该id
	后置条件	按照任务ID删除相应TaskTag记录
TaskTagMapper.selectAll	语法	List<TaskTag> selectAll();
	前置条件	无
	后置条件	返回所有任务和标签的关联记录

4.2.3.5 TaskUserMapper

提供的服务(供接口)		
TaskUserMapper.selectByUserId	语法	List<Integer> selectByUserId(Integer userid);
	前置条件	无
	后置条件	按照用户ID进行查找返回相应的任务ID
TaskUserMapper.selectByTaskId	语法	List<Integer> selectByTaskId(Integer taskid);
	前置条件	无
	后置条件	按照任务ID进行查找返回相应的用户ID
TaskUserMapper.selectByTaskIds	语法	List<TaskUser> selectByTaskIds(List<Integer> idList);
	前置条件	无
	后置条件	按照任务ID进行查找返回对应的TaskUser列表
TaskUserMapper.insert	语法	int insert(TaskUser record);
	前置条件	相同id的TaskUser在Mapper中不存在
	后置条件	在数据库中增加一个TaskUser记录
TaskUserMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(@Param("taskid") Integer taskid, @Param("userid") Integer userid);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应TaskUser记录
TaskUserMapper.selectAll	语法	List<TaskUser> selectAll();
	前置条件	无
	后置条件	返回所有任务-用户（众包工人选取任务）的记录

4.2.3.6 ReportMapper

提供的服务(供接口)		
ReportMapper.selectByPrimaryKey	语法	Report selectByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照报告ID进行查找返回相应的Report结果
ReportMapper.insert	语法	int insert(Report record);
	前置条件	相同id的Report在Mapper中不存在
	后置条件	在数据库中增加一个Report记录
ReportMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照报告ID删除相应Report记录
ReportMapper.selectAll	语法	List<Report> selectAll();
	前置条件	无
	后置条件	返回所有报告的记录
ReportMapper.selectByUserId	语法	List<ReportView> selectByUserId(Integer userid);
	前置条件	无
	后置条件	通过用户Id查找返回提交的报告记录
ReportMapper.selectByTaskId	语法	List<ReportView> selectByTId(Integer userid);
	前置条件	无
	后置条件	通过用户Id查找返回相应的报告记录
ReportMapper.selectByUserIdAndPage	语法	List<ReportView> selectByUserIdAndPage(Integer userid, Integer start, Integer pagesize);
	前置条件	无
	后置条件	通过用户Id查找返回相应的限定个数的报告记录
ReportMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(Report record);
	前置条件	无
	后置条件	根据关键字更新该条报告记录

4.2.3.7 FlawMapper

提供的服务(供接口)		
FlawMapper.selectByPrimaryKey	语法	Flaw selectByPrimaryKey(Integer id);
	前置条件	数据库存在相应id
	后置条件	按照缺陷ID进行查找返回相应的缺陷记录
FlawMapper.selectByReportId	语法	List<Flaw> selectByReportId(Integer reportid);
	前置条件	无
	后置条件	按照报告ID进行查找返回相应的缺陷列表
FlawMapper.insert	语法	int insert(Flaw record);
	前置条件	相同id的Flaw在Mapper中不存在
	后置条件	在数据库中增加一个Flaw记录
FlawMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应Flaw记录
FlawMapper.selectAll	语法	List<Flaw> selectAll();
	前置条件	无
	后置条件	返回所有缺陷记录
FlawMapper.selectByTaskId	语法	List<Flaw> selectByTaskId(Integer taskid);
	前置条件	无
	后置条件	按照任务ID进行查找返回相应的缺陷列表
FlawMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(Flaw record);
	前置条件	无
	后置条件	根据关键字更新对应的缺陷记录

4.2.3.8 FlawPicMapper

提供的服务(供接口)		
FlawPicMapper.selectByFlawId	语法	List<String> selectByFlawId(Integer flawid);
	前置条件	无
	后置条件	按照缺陷ID进行查找返回相应的图片url
FlawPicMapper.insert	语法	int insert(FlawPic record);
	前置条件	相同id的FlawPic在Mapper中不存在
	后置条件	在数据库中增加一个FlawPic记录
FlawPicMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(@Param("flawid") Integer flawid, @Param("pictureurl") String pictureurl);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应FlawPic记录
FlawPicMapper.selectAll	语法	List<FlawPic> selectAll();
	前置条件	无
	后置条件	返回所有缺陷-对应图片url的记录

4.2.3.9 EvaluationMapper

提供的服务(供接口)		
EvaluationMapper.selectByFlawId	语法	List<Evaluation> selectByFlawId(Integer flawid);
	前置条件	无
	后置条件	按照缺陷ID进行查找返回相应的评价
EvaluationMapper.insert	语法	int insert(Evaluation record);
	前置条件	相同id的Evaluation在Mapper中不存在
	后置条件	在数据库中增加一个Evaluation记录
EvaluationMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应Evaluation记录
EvaluationMapper.selectAll	语法	List<Evaluation> selectAll();
	前置条件	无
	后置条件	返回所有缺陷-对应评价的记录
EvaluationMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(Evaluation record);
	前置条件	无
	后置条件	根据关键字更新对应的评价记录
EvaluationMapper.selectByPrimaryKey	语法	Evaluation selectByPrimaryKey(Integer id);
	前置条件	数据库存在相应id
	后置条件	按照关键字进行查找返回相应的评价记录

4.2.3.10 ScoreMapper

提供的服务(供接口)		
ScoreMapper.insert	语法	int insert(Score record);
	前置条件	相同id的Score在Mapper中不存在
	后置条件	在数据库中增加一个Score记录
ScoreMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer userid, Integer flawid);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应Score记录
ScoreMapper.selectAll	语法	List<Score> selectAll();
	前置条件	无
	后置条件	返回所有缺陷-对应评分的记录
ScoreMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(Score record);
	前置条件	无
	后置条件	根据关键字更新对应的评分记录
ScoreMapper.selectByPrimaryKey	语法	Score selectByPrimaryKey(Integer userid, Integer flawid);
	前置条件	数据库存在相应id
	后置条件	按照关键字进行查找返回相应的评分记录

4.2.3.11 SimilarityMapper

提供的服务(供接口)		
SimilarityMapper.insert	语法	int insert(Similarity record);
	前置条件	相同id的Similarity在Mapper中不存在
	后置条件	在数据库中增加一个Similarity记录
SimilarityMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer flawid1, Integer flawid2);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应Similarity记录
SimilarityMapper.selectAll	语法	List<Similarity> selectAll();
	前置条件	无
	后置条件	返回所有缺陷相似度的记录
SimilarityMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(Similarity record);
	前置条件	无
	后置条件	根据关键字更新对应的相似度记录
SimilarityMapper.selectByPrimaryKey	语法	Score selectByPrimaryKey(Integer flawid1, Integer flawid2);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的相似度记录
SimilarityMapper.selectByFlawIdCompared	语法	List<Similarity> selectByFlawIdCompared(Integer flawid1);
	前置条件	数据库存在相应关键字的记录
	后置条件	用当前flawId查找比此flaw先插入且有相似度的记录
SimilarityMapper.selectByFlawIdSecondCompared	语法	List<Similarity> selectByFlawIdSecondCompared(Integer flawid2);
	前置条件	数据库存在相应关键字的记录
	后置条件	用当前flawId查找比此flaw后插入且有相似度的记录

4.2.3.12 RecommendRuleMapper

提供的服务(供接口)		
RecommendRuleMapper.insert	语法	int insert(RecommendRule record);
	前置条件	相同id的RecommendRule在Mapper中不存在
	后置条件	在数据库中增加一个RecommendRule记录
RecommendRuleMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer id);
	前置条件	在数据库中存在该id
	后置条件	按照ID删除相应RecommendRule记录
RecommendRuleMapper.selectAll	语法	List<RecommendRule> selectAll();
	前置条件	无
	后置条件	返回所有推荐规则的记录
RecommendRuleMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(RecommendRule record);
	前置条件	无
	后置条件	根据关键字更新对应的推荐规则记录
RecommendRuleMapper.selectByPrimaryKey	语法	RecommendRule selectByPrimaryKey(Integer id);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的推荐规则记录
RecommendRuleMapper.selectByName	语法	RecommendRule selectByName(String name);
	前置条件	无
	后置条件	查找规则名称对应的推荐规则
RecommendRuleMapper.updateAllOff	语法	int updateAllOff();
	前置条件	无
	后置条件	让所有的规则的isUsing字段都变为false
RecommendRuleMapper.updateOneOn	语法	int updateOneOn(Integer id);
	前置条件	数据库中存在对应的id
	后置条件	让对应id的规则状态变为使用中

4.2.3.13 RecommendRuleFactorMapper

提供的服务(供接口)		
RecommendRuleFactorMapper.insert	语法	int insert(RecommendRuleFactorMapper record);
	前置条件	相同id的RecommendRuleFactor在Mapper中不存在
	后置条件	在数据库中增加一个RecommendRuleFactor记录
RecommendRuleFactorMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(String factorname);
	前置条件	在数据库中存在改关键字对应的信息
	后置条件	按照影响因素的名称删除相应RecommendRuleFactor记录
RecommendRuleFactorMapper.selectAll	语法	List<RecommendRuleFactor> selectAll();
	前置条件	无
	后置条件	返回所有推荐规则和影响因子关联的记录
RecommendRuleFactorMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(RecommendRuleFactorMapper record);
	前置条件	无
	后置条件	根据关键字更新对应的推荐规则和影响因子的关联记录
RecommendRuleFactorMapper.selectByPrimaryKey	语法	RecommendRuleFactor selectByPrimaryKey(String factorname);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的推荐规则和影响因子的关联记录
RecommendRuleFactorMapper.selectByRuleId	语法	List<RecommendRuleFactor> selectByRuleId(int ruleId);
	前置条件	无
	后置条件	查找规则对应的影响因素列表
RecommendRuleFactorMapper.deleteByRuleId	语法	int deleteByRuleId(int ruleId);
	前置条件	数据库中存在该规则id
	后置条件	删除数据库中该条规则对应的记录
RecommendRuleFactorMapper.selectByUsingRule	语法	List<RecommendRuleFactor> selectByUsingRule();
	前置条件	无
	后置条件	查找状态为使用中的RecommendRuleFactor记录

4.2.3.14 UserSimilarityMapper

提供的服务(供接口)		
UserSimilarityMapper.selectSimilarUserVectorInfo	语法	List<UserVectorComponent> selectSimilarUserVectorInfo(Integer userId, Integer limit);
	前置条件	无
	后置条件	根据用户关键字，算出相似用户向量信息的列表
UserSimilarityMapper.selectJaccardSimilarity	语法	List<JaccardUserSimilarity> selectJaccardSimilarity(Integer userId, Integer limit);
	前置条件	无
	后置条件	根根据用户关键字，使用杰卡德距离选择出相似用户列表

4.2.3.15 MultiObjectiveRecommendFactorMapper

提供的服务(供接口)		
MultiObjectiveRecommendFactorMapper.insert	语法	int insert(MultiObjectiveRecommendFactorMapper record);
	前置条件	相同id的记录在Mapper中不存在
	后置条件	在数据库中增加一个MultiObjectiveRecommendFactor记录
MultiObjectiveRecommendFactorMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer userid);
	前置条件	在数据库中存在改关键字对应的信息
	后置条件	按照关键字删除相应MultiObjectiveRecommendFactor记录
MultiObjectiveRecommendFactorMapper.selectAll	语法	List<MultiObjectiveRecommendFactorMapper> selectAll();
	前置条件	无
	后置条件	返回所有MultiObjectiveRecommendFactor的记录
MultiObjectiveRecommendFactorMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(MultiObjectiveRecommendFactor record);
	前置条件	无
	后置条件	根据关键字更新对应的MultiObjectiveRecommendFactorMapper记录
MultiObjectiveRecommendFactorMapper.selectByTaskIds	语法	ListMultiObjectiveRecommendFactor> selectByTaskIds(ListInteger> ids)
	前置条件	无
	后置条件	返回关键字位于给定列表中的记录

4.2.3.16 MultiObjectiveRecommendResultMapper

提供的服务(供接口)		
MultiObjectiveRecommendResultMapper.insertAll	语法	void insertAll(<MultiObjectiveRecommendResult> resultList);
	前置条件	相同id的MultiObjectiveRecommendResult在Mapper中不存在
	后置条件	在数据库中增加多个MultiObjectiveRecommendResult记录
MultiObjectiveRecommendResultMapper.deleteAll	语法	int deleteAll();
	前置条件	无
	后置条件	删除表中所有记录
MultiObjectiveRecommendResultMapper.selectAll	语法	List<MultiObjectiveRecommendResult> selectAll();
	前置条件	无
	后置条件	返回所有MultiObjectiveRecommendResult的记录
MultiObjectiveRecommendResultMapper.selectByUserId	语法	MultiObjectiveRecommendResult selectByUserId(Integer userid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的MultiObjectiveRecommendResult记录

4.2.3.17 TaskRecruitStopRecommendFactorMapper

提供的服务(供接口)		
TaskRecruitStopRecommendFactorMapper.deleteAll	语法	int deletAll();
	前置条件	无
	后置条件	删除所有记录
TaskRecruitStopRecommendFactorMapper.selectAll	语法	List<TaskRecruitStopRecommendFactor> selectAll();
	前置条件	无
	后置条件	返回所有TaskRecruitStopRecommendFactor的记录
TaskRecruitStopRecommendFactorMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(TaskRecruitStopRecommendFactor record);
	前置条件	无
	后置条件	根据关键字更新对应的TaskRecruitStopRecommendFactor记录
TaskRecruitStopRecommendFactorMapper.selectByPrimaryKey	语法	TaskRecruitStopRecommendFactor selectByPrimaryKey(Integer userid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的TaskRecruitStopRecommendFactor记录
TaskRecruitStopRecommendFactorMapper.insertAll	语法	void insertAll(<TaskRecruitStopRecommendFactor> records);
	前置条件	相同id的TaskRecruitStopRecommendFactor在Mapper中不存在
	后置条件	在数据库中增加多个TaskRecruitStopRecommendFactor记录

4.2.3.18 WorkerAbilityMapper

提供的服务(供接口)		
WorkerAbilityMapper.insert	语法	int insert(WorkerAbility record);
	前置条件	相同id的WorkerAbility在Mapper中不存在
	后置条件	在数据库中增加一个WorkerAbility记录
WorkerAbilityMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer userid);
	前置条件	在数据库中存在改关键字对应的信息
	后置条件	按照关键字删除相应WorkerAbility记录
WorkerAbilityMapper.selectAll	语法	List<WorkerAbility> selectAll();
	前置条件	无
	后置条件	返回所有WorkerAbility的记录
WorkerAbilityMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(WorkerAbility record);
	前置条件	无
	后置条件	根据关键字更新对应的WorkerAbility记录
WorkerAbilityMapper.selectByPrimaryKey	语法	WorkerAbility selectByPrimaryKey(Integer userid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的WorkerAbility记录

4.2.3.19 WorkerContextMapper

提供的服务(供接口)		
WorkerContextMapper.insert	语法	int insert(WorkerContext record);
	前置条件	相同id的WorkerContext在Mapper中不存在
	后置条件	在数据库中增加一个WorkerContext记录
WorkerContextMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(Integer workerid);
	前置条件	在数据库中存在改关键字对应的信息
	后置条件	按照关键字删除相应WorkerContext记录
WorkerContextMapper.selectAll	语法	List<WorkerContext> selectAll();
	前置条件	无
	后置条件	返回所有WorkerContext的记录
WorkerContextMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(WorkerContext record);
	前置条件	无
	后置条件	根据关键字更新对应的WorkerContext记录
WorkerContextMapper.selectByPrimaryKey	语法	WorkerContext selectByPrimaryKey(Integer workerid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的WorkerContext记录

4.2.3.20 EvaLikeMapper

提供的服务(供接口)		
EvaLikeMapper.insert	语法	int insert(EvaLike record);
	前置条件	相同id的EvaLike在Mapper中不存在
	后置条件	在数据库中增加一个EvaLike记录
EvaLikeMapper.deleteByPrimaryKey	语法	int deleteByPrimaryKey(@Param("evaid") Integer evaid, @Param("userid") Integer userid);
	前置条件	在数据库中存在该关键字对应的信息
	后置条件	按照关键字删除相应EvaLike记录
EvaLikeMapper.selectAll	语法	List<EvaLike> selectAll();
	前置条件	无
	后置条件	返回所有EvaLike的记录
EvaLikeMapper.updateByPrimaryKey	语法	int updateByPrimaryKey(EvaLike record);
	前置条件	无
	后置条件	根据关键字更新对应的EvaLike记录
EvaLikeMapper.selectByPrimaryKey	语法	EvaLike selectByPrimaryKey(@Param("evaid") Integer evaid, @Param("userid") Integer userid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回相应的EvaLike记录
EvaLikeMapper.selectCountByEva	语法	EvaLikeStatistic selectCountByEva(@Param("evaid") Integer evaid);
	前置条件	数据库存在相应关键字的记录
	后置条件	按照关键字进行查找返回点赞点踩相关的统计值

5. 信息视角

5.1 VO定义

5.1.2 ReponseVO

```
public class ResponseVO<T> {
    public int code;
    public String msg;
    public T data;
}
```

5.1.3 UserVO

```
public class UserVO {
    @ApiModelProperty("用户唯一标识码")
    private Integer id;

    @ApiModelProperty(value = "用户类型，0对应发包方，1对应众包工人，2对应管理员")
    private Integer type;

    @ApiModelProperty("用户名")
    private String username;

    @ApiModelProperty("密码")
    private String password;

    @ApiModelProperty("邮箱")
    private String email;

    @ApiModelProperty("手机号")
    private String phone;

    @ApiModelProperty("标签组")
    private List<Integer> tagGroups;
}
```

5.1.4 UserViewVO

此VO对应用户的缩略信息，后期迭代会包括用户头像等。

```
public class UserViewVO {
    @ApiModelProperty("用户唯一标识码")
    private Integer id;

    @ApiModelProperty(value = "用户类型，0对应发包方，1对应众包工人，2对应管理员")
    private Integer type;

    @ApiModelProperty("用户名")
    private String username;
}
```

5.1.5 TaskVO

```
public class TaskVO {
       @ApiModelProperty("任务唯一标识码")
    private Integer id;

    @ApiModelProperty("发包方的id")
    private Integer userId;

    @ApiModelProperty(value = "0代表功能测试，1代表性能测试")
    private Integer type;

    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("描述")
    private String description;

    @ApiModelProperty("开始日期")
//    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date startTime;

    @ApiModelProperty("结束日期")
//    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date endTime;

    @ApiModelProperty("创建日期")
//    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
```

```
    private Date createTime;

    @ApiModelProperty("工人数量")
    private Integer workerNum;

    @ApiModelProperty("已选工人数量")
    private Integer pickedWorkerNum;

    @ApiModelProperty("可执行文件url")
    private String exeFileName;

    @ApiModelProperty("需求文档url")
    private String reqDocName;

    @ApiModelProperty(value = "用来描述任务的状态。0表示未领取或者请求该任务详情的用户不是众包工人，1表示进行中，2表示已完成")
    private Integer state;

    @ApiModelProperty("标签组")
    private List<Integer> tagGroups;
}
```

5.1.6 TaskViewVO

此VO对应任务的缩略信息。

```
public class TaskViewVO {
    @ApiModelProperty("任务唯一标识码")
    private Integer id;

    @ApiModelProperty("发包方的id")
    private Integer userId;

    @ApiModelProperty(value = "0代表功能测试，1代表性能测试")
    private Integer type;

    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("描述")
    private String description;

    @ApiModelProperty("开始日期")
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date startTime;

    @ApiModelProperty("结束日期")
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date endTime;

    @ApiModelProperty("工人数量")
    private Integer workerNum;

    @ApiModelProperty("标签组")
    private List<Integer> tagGroups;
}
```

5.1.7 ReportVO

```
public class ReportVO {
    @ApiModelProperty("报告唯一标识码")
    private Integer id;

    @ApiModelProperty("任务的id，可服务于众包工人直接浏览所有测试报告")
    private Integer taskId;

    @ApiModelProperty("众包工人的id，为了让发包方看到")
    private Integer userId;
```

```
    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("报告评分")
    private Double score;

    @ApiModelProperty("创建日期")
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date createTime;

    @ApiModelProperty("缺陷列表")
    private List<FlawVO> flaws;
}
```

5.1.8 ReportViewVO

此VO对应报告的缩略信息。

```
public class ReportViewVO {
    @ApiModelProperty("报告唯一标识码")
    private Integer id;

    @ApiModelProperty("任务的id，可服务于众包工人直接浏览所有测试报告")
    private Integer taskId;

    @ApiModelProperty("众包工人的id，为了让发包方看到")
    private Integer userId;

    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("报告评分")
    private Double score;

    @ApiModelProperty("缺陷个数")
    private Integer flawNum;

}
```

5.1.9 FlawVO

```
public class FlawVO {
@ApiModelProperty("缺陷唯一标识码")
    private Integer id;

    @ApiModelProperty("报告id")
    private Integer reportId;

    @ApiModelProperty("错误截图")
    private List<String> pictureURLs;

    @ApiModelProperty("描述")
    private String description;

    @ApiModelProperty("复现步骤描述")
    private String stepDes;

    @ApiModelProperty("设备信息")
    private String deviceInfo;

    @ApiModelProperty("缺陷平均分")
    private Double score;

    @ApiModelProperty("评分人数")
    private Integer scoreNum;

    @ApiModelProperty("缺陷是否已被处理")
```

```
    private boolean state;

    @ApiModelProperty("对已提交的报告的补充内容")
    private String appendcontent;
}
```

5.1.10 TBAFlawVO

此VO对应一个待完善的缺陷和相应相似度高的历史缺陷。

```
public class TBAFlawVO {
    @ApiModelProperty("有待完善的缺陷")
    private FlawVO tbaFlaw;

    @ApiModelProperty("相似缺陷列表")
    private List<SimilarFlawVO> similarFlaws;
}
```

5.1.11 SimilarFlawVO

此VO前端用于展示某缺陷的相似缺陷列表

```
public class SimilarFlawVO implements Comparable<SimilarFlawVO>, Cloneable {
    @ApiModelProperty("缺陷相似度")
    private double similarity;

    @ApiModelProperty("缺陷")
    private FlawVO flaw;
}
```

5.1.12 FlawTreeNodeVO

此VO前端用于展示缺陷树

```
public class FlawTreeNodeVO {
    @ApiModelProperty("缺陷树子节点")
    private List<FlawTreeNodeVO> children;

    @ApiModelProperty("当前缺陷")
    private FlawVO current;
}
```

5.1.13 FlawMapVO

此VO前端用于展示缺陷图

```
public class FlawMapVO {
    @ApiModelProperty("所有根节点的flawId")
    private List<Integer> flawIds;

    @ApiModelProperty("矩阵中i行j列的元素代表flawIds中索引为i和j的缺陷的相似度")
    private List<List<Double>> similarityMatrix;
}
```

5.1.14 FlawEvaluationVO

此VO前端用于展示某缺陷的用户评论

```
public class FlawEvaluationVO {
    @ApiModelProperty("评价用户的简略信息")
    private UserViewVO user;

    @ApiModelProperty("评价内容")
    private String content;

}
```

5.1.15 OSSPolicyVO

```
public class OSSPolicyVO {

    @ApiModelProperty("访问身份验证中用到用户标识")
    private String accessKeyId;

    @ApiModelProperty("用户表单上传的策略,经过base64编码过的字符串")
    private String policy;

    @ApiModelProperty("对policy签名后的字符串")
    private String signature;

    @ApiModelProperty("上传文件夹路径前缀")
    private String dir;

    @ApiModelProperty("oss对外服务的访问域名")
    private String host;

    @ApiModelProperty("上传成功后的回调设置")
    private OSSCallbackParamVO callback;

}
```

5.1.16 OSSCallbackResultVO

```
public class OSSCallbackResultVO {

    @ApiModelProperty("文件名称")
    private String filename;

    @ApiModelProperty("文件大小")
    private String size;

    @ApiModelProperty("文件的mimeType")
    private String mimeType;

    @ApiModelProperty("图片文件的宽")
    private String width;

    @ApiModelProperty("图片文件的高")
    private String height;
}
```

5.1.12 OSSCallbackParamVO

```
public class OSSCallbackParamVO {
    @ApiModelProperty("请求的回调地址")
    private String callbackUrl;

    @ApiModelProperty("回调时传入request中的参数")
    private String callbackBody;

    @ApiModelProperty("回调时传入参数的格式，比如表单提交形式")
    private String callbackBodyType;
}
```

5.1.13 WorkerActivationVO

```
public class WorkerActivationVO {

    @ApiModelProperty("最后一次提交时间")
    private Date lastSubmitTime;

    @ApiModelProperty("近三天提交的报告数")
    private Integer lastThreeDaySubmitNum;
```

```
    @ApiModelProperty("近两周提交的报告数")
    private Integer lastTwoWeekSubmitNum;

    @ApiModelProperty("近一个月提交的报告数")
    private Integer lastOneMonthSubmitNum;

    @ApiModelProperty("近半年提交的报告数")
    private Integer lastHalfYearSubmitNum;

    @ApiModelProperty("活跃度评分")
    private Integer score;

}
```

5.1.14 WorkerCloudVO

```
public class WorkerCloudVO {

    @ApiModelProperty("关键字")
    private String name;

    @ApiModelProperty("权重")
    private Integer value;
}
```

5.1.15 WorkerContextVO

```
public class WorkerContextVO {
    @ApiModelProperty("工人id")
    private Integer id;

    @ApiModelProperty("设备类型")
    private String deviceType;

    @ApiModelProperty("os信息")
    private String osInfo;

    @ApiModelProperty("内存大小")
    private String ramSize;

    @ApiModelProperty("网络环境")
    private String netEnv;

}
```

5.1.16 WorkerRadarVO

```
public class WorkerRadarVO {

    @ApiModelProperty("报告协作能力")
    private Integer cooperateAbility;

    @ApiModelProperty("报告审查能力")
    private Integer reviewAbility;

    @ApiModelProperty("创新能力")
    private Integer creativeAbility;

    @ApiModelProperty("寻找bug的能力")
    private Integer findBugAbility;

    @ApiModelProperty("语言表达能力")
    private Integer languageAbility;

    @ApiModelProperty("综合能力")
    private Integer avgAbility;
```

```
}
```

5.1.17 TaskFlawDetectionCurveVO

```
public class TaskFlawDetectionCurveVO {
    @ApiModelProperty("当前时间（以天为单位）")
    @JsonFormat(pattern = "yyyy-MM-dd")
    private LocalDate time;

    @ApiModelProperty("截止当天发现缺陷总数")
    private Integer num;
}
```

5.1.18 TaskFlawDetectionPredictionVO

```
public class TaskFlawDetectionPredictionVO {
    @ApiModelProperty("任务预测的flaw数")
    private Integer numPredicted;

    @ApiModelProperty("任务已发现缺陷曲线列表")
    private List<TaskFlawDetectionCurveVO> curves;
}
```

5.1.19 TaskRadarVO

```
public class TaskRadarVO {
    @ApiModelProperty("能力")
    private Integer ability;

    @ApiModelProperty("活跃度")
    private Integer activation;

    @ApiModelProperty("相关性")
    private Integer revelance;

    @ApiModelProperty("多样性")
    private Integer diversity;

    @ApiModelProperty("平均目标值")
    private Integer avgTarget;

}
```

5.1.20 FlawEvaLikeVO

```
public class FlawEvaLikeVO {
    @ApiModelProperty("评价的点赞数")
    private Integer likeNum;

    @ApiModelProperty("评价的点踩数")
    private Integer dislikeNum;

    @ApiModelProperty("评价的状态")
    private Integer status;
}
```

5.3 DTO视角

DTO主要用于前端传递数据给后端。

5.3.1 UserDTO

用户注册时，前端传给后端的数据

```
public class UserDTO {
    @ApiModelProperty(value = "用户类型，0对应发包方，1对应众包工人，2对应管理员")
    private Integer type;
```

```
    @ApiModelProperty("用户名")
    private String username;

    @ApiModelProperty("密码")
    private String password;

    @ApiModelProperty("邮箱")
    private String email;

    @ApiModelProperty("手机号")
    private String phone;

    @ApiModelProperty("标签组")
    private List<Integer> tagGroups;

}
```

5.3.2 UserFormDTO

用户登录时，前端传给后端的数据

```
public class UserFormDTO {
    @ApiModelProperty("用户名")
    private String username;

    @ApiModelProperty("密码")
    private String password;
}
```

5.3.3 TaskDTO

发布任务时，前端传给后端的数据

```
public class TaskDTO {
    @ApiModelProperty("发包方的id")
    private Integer userId;

    @ApiModelProperty(value = "0代表功能测试，1代表性能测试")
    private Integer type;

    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("描述")
    private String description;

    @ApiModelProperty("开始日期")
//    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date startTime;

    @ApiModelProperty("结束日期")
//    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
    private Date endTime;

    @ApiModelProperty("工人数量")
    private Integer workerNum;

    @ApiModelProperty("可执行文件url")
    private String exeFileName;

    @ApiModelProperty("需求文档url")
    private String reqDocName;

    @ApiModelProperty("标签组")
    private List<Integer> tagGroups;

}
```

5.3.4 ReportDTO

提交报告时，前端传递给后端的数据

```
public class ReportDTO {
    @ApiModelProperty("任务的id，可服务于众包工人直接浏览所有测试报告")
    private Integer taskId;

    @ApiModelProperty("众包工人的id，为了让发包方看到")
    private Integer userId;

    @ApiModelProperty("标题")
    private String title;

    @ApiModelProperty("缺陷列表")
    private List<FlawDTO> flaws;
}
```

5.3.5 RecommendRuleDTO

管理员增加规则时，前端传递给后端的数据

```
public class RecommendRuleDTO {
    @ApiModelProperty("推荐规则名称")
    private String name;

    @ApiModelProperty("推荐影响因素列表")
    private List<RecommendRuleFactorVO> factors;
}
```

5.3.6 FlawDTO

用户提交报告时，前端传递给后端的flaw数据

```
public class FlawDTO {
    @ApiModelProperty("错误截图")
    private List<String> pictureURLs;

    @ApiModelProperty("描述")
    private String description;

    @ApiModelProperty("复现步骤描述")
    private String stepDes;

    @ApiModelProperty("设备信息")
    private String deviceInfo;
}
```

5.3.7 FlawAppendDTO

用户对自己的缺陷进行补充描述时，前端传递给后端的数据

```
public class FlawAppendDTO {
    @ApiModelProperty("缺陷id")
    private Integer flawId;

    @ApiModelProperty("补充内容")
    private String appendContent;
}
```

5.3.8 FlawEvaluationDTO

用户对他人缺陷进行评价时，前端传递给后端的数据

```
public class FlawEvaluationDTO {
    @ApiModelProperty("评论的用户id")
    private Integer userId;

    @ApiModelProperty("评价内容")
    private String content;
}
```

5.3.9 FlawScoreDTO

用户对他人缺陷进行评分时，前端传递给后端的数据

```
public class FlawScoreDTO {
    @ApiModelProperty("评分用户id")
    private Integer userId;

    @ApiModelProperty("缺陷id")
    private Integer flawId;

    @ApiModelProperty("评分")
    private Integer score;
}
```

5.3.10 WorkerContextDTO

```
public class WorkerContextDTO {
    @ApiModelProperty("设备类型")
    private String deviceType;

    @ApiModelProperty("os信息")
    private String osInfo;

    @ApiModelProperty("内存大小")
    private String ramSize;

    @ApiModelProperty("网络环境")
    private String netEnv;
}
```

5.4 PO视角

po数据往往和数据库的字段一一对应。

5.4.1 User

```
public class User implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "phone")
    private String phone;

    @ApiModelProperty(value = "", name = "type")
    private Integer type;

    @ApiModelProperty(value = "", name = "password")
    private String password;

    @ApiModelProperty(value = "", name = "username")
    private String username;

    @ApiModelProperty(value = "", name = "email")
    private String email;
}
```

5.4.2 UserTag

```
public class UserTag implements Serializable {
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "tag")
    private Integer tag;
}
```

5.4.3 JaccardUserSimilarity

用于存储两用户的相似度，不用每次都计算

```
public class JaccardUserSimilarity {
    @ApiModelProperty(value = "", name = "userid1")
    Integer userid1;
    @ApiModelProperty(value = "", name = "userid2")
    Integer userid2;
    @ApiModelProperty(value = "", name = "jaccardsim")
    Double jaccardsim;
}
```

5.4.4 Task

```
public class Task implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "type")
    private Integer type;

    @ApiModelProperty(value = "", name = "title")
    private String title;

    @ApiModelProperty(value = "", name = "description")
    private String description;

    @ApiModelProperty(value = "", name = "starttime")
    private Date starttime;

    @ApiModelProperty(value = "", name = "endtime")
    private Date endtime;

    @ApiModelProperty(value = "", name = "workernum")
    private Integer workernum;

    @ApiModelProperty(value = "", name = "exefilename")
    private String exefilename;

    @ApiModelProperty(value = "", name = "reqdocname")
    private String reqdocname;

    @ApiModelProperty(value = "", name = "createtime")
    private Date createtime;
}
```

5.4.5 TaskView

有时只需要task的部分信息，增加此PO是为了减少数据库返回数据，提高效率

```
public class TaskView implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;
```

```
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "type")
    private Integer type;

    @ApiModelProperty(value = "", name = "title")
    private String title;

    @ApiModelProperty(value = "", name = "description")
    private String description;

    @ApiModelProperty(value = "", name = "starttime")
    private Date starttime;

    @ApiModelProperty(value = "", name = "endtime")
    private Date endtime;

    @ApiModelProperty(value = "", name = "workernum")
    private Integer workernum;
}
```

5.4.6 TaskTag

```
public class TaskTag implements Serializable {
    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "tag")
    private Integer tag;
}
```

5.4.7 TaskUser

此PO存储众包工人承接了哪些任务的信息

```
public class TaskUser implements Serializable {
    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;
}
```

5.4.8 Report

```
public class Report implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "title")
    private String title;

    @ApiModelProperty(value = "", name = "createtime")
    private Date createtime;
}
```

5.4.9 ReportView

有时只需要report的部分信息，增加此PO是为了减少数据库返回数据，提高效率

```
public class ReportView implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "title")
    private String title;
}
```

5.4.10 RecommendRule

```
public class RecommendRule implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "name")
    private String name;

    @ApiModelProperty(value = "", name = "isusing")
    private Boolean isusing;
}
```

5.4.11 RecommendRuleFactor

用于存储`5.4.10 RecommendRule`的具体权重

```
public class RecommendRuleFactor implements Serializable {
    @ApiModelProperty(value = "", name = "factorname")
    private String factorname;

    @ApiModelProperty(value = "", name = "ruleid")
    private Integer ruleid;

    @ApiModelProperty(value = "", name = "factorweight")
    private Double factorweight;
}
```

5.4.12 UserVectorComponent

属于`recommend`模块，用于存储推荐相关的用户信息

```
public class UserVectorComponent {
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;
    @ApiModelProperty(value = "", name = "tag")
    private Integer tag;
    @ApiModelProperty(value = "", name = "score")
    private Double score;
}
```

5.4.13 Flaw

```
public class Flaw implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "reportid")
    private Integer reportid;
```

```
    @ApiModelProperty(value = "", name = "description")
    private String description;

    @ApiModelProperty(value = "", name = "stepdes")
    private String stepdes;

    @ApiModelProperty(value = "", name = "deviceinfo")
    private String deviceinfo;

    @ApiModelProperty(value = "", name = "score")
    private Double score;

    @ApiModelProperty(value = "", name = "state")
    private boolean state;

    @ApiModelProperty(value = "", name = "path")
    private String path;

    @ApiModelProperty(value = "", name = "scorenum")
    private Integer scorenum;

    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "appendcontent")
    private String appendcontent;
}
```

5.4.14 FlawPic

flaw的缺陷截图

```
public class FlawPic implements Serializable {
    @ApiModelProperty(value = "", name = "flawid")
    private Integer flawid;

    @ApiModelProperty(value = "", name = "pictureurl")
    private String pictureurl;
}
```

5.4.15 (Flaw)Similarity

存储相似度，避免二次计算。

```
public class Similarity implements Serializable, Comparable<Similarity> {
    @ApiModelProperty(value = "", name = "flawid1")
    private Integer flawid1;

    @ApiModelProperty(value = "", name = "flawid2")
    private Integer flawid2;

    @ApiModelProperty(value = "", name = "similarity")
    private Double similarity;
}
```

5.4.16 (Flaw)Score

```
public class Score implements Serializable {
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "flawid")
    private Integer flawid;

    @ApiModelProperty(value = "", name = "socre")
    private Integer socre;
}
```

5.4.17 (Flaw)Evaluation

缺陷的所有评论

```
public class Evaluation implements Serializable {
    @ApiModelProperty(value = "", name = "id")
    private Integer id;

    @ApiModelProperty(value = "", name = "flawid")
    private Integer flawid;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "content")
    private String content;
}
```

5.4.18 WorkerAbility

```
public class WorkerAbility implements Serializable {
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "collabval")
    private Double collabval;

    @ApiModelProperty(value = "", name = "reviewval")
    private Double reviewval;

    @ApiModelProperty(value = "", name = "creatval")
    private Double creatval;

    @ApiModelProperty(value = "", name = "detval")
    private Double detval;

    @ApiModelProperty(value = "", name = "expval")
    private Double expval;

    @ApiModelProperty(value = "", name = "comprehensiveval")
    private Double comprehensiveval;
}
```

5.4.19 WorkerContext

```
public class WorkerContext implements Serializable {
    @ApiModelProperty(value = "", name = "workerid")
    private Integer workerid;

    @ApiModelProperty(value = "", name = "devicetype")
    private String devicetype;

    @ApiModelProperty(value = "", name = "osinfo")
    private String osinfo;

    @ApiModelProperty(value = "", name = "ramsize")
```

```
    private String ramsize;

    @ApiModelProperty(value = "", name = "netenv")
    private String netenv;
}
```

5.4.20 TaskRecruitStopRecommendFactor

```
public class TaskRecruitStopRecommendFactor implements Serializable {
    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "abilityactual")
    private Double abilityactual;

    @ApiModelProperty(value = "", name = "abilityexpected")
    private Double abilityexpected;

    @ApiModelProperty(value = "", name = "activenessactual")
    private Double activenessactual;

    @ApiModelProperty(value = "", name = "activenessexpected")
    private Double activenessexpected;

    @ApiModelProperty(value = "", name = "relevanceactual")
    private Double relevanceactual;

    @ApiModelProperty(value = "", name = "relevanceexpected")
    private Double relevanceexpected;

    @ApiModelProperty(value = "", name = "diversityactual")
    private Double diversityactual;

    @ApiModelProperty(value = "", name = "diversityexpected")
    private Double diversityexpected;

    @ApiModelProperty(value = "", name = "avgtargetactual")
    private Double avgtargetactual;

    @ApiModelProperty(value = "", name = "avgtargetexpected")
    private Double avgtargetexpected;
}
```

5.4.21 EvaLike

```
public class EvaLike implements Serializable {
    @ApiModelProperty(value = "", name = "evaid")
    private Integer evaid;

    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "support")
    private Integer support;

    @ApiModelProperty(value = "", name = "oppose")
    private Integer oppose;
}
```

5.4.22 EvaLikeStatistic

用于统计点赞点踩的个数。

```
public class EvaLikeStatistic implements Serializable  {
    @ApiModelProperty(value = "", name = "supportcnt")
    private Integer supportcnt;

    @ApiModelProperty(value = "", name = "opposecnt")
    private Integer opposecnt;
}
```

5.4.23 MultiObjectiveRecommendFactor

```
public class MultiObjectiveRecommendFactor implements Serializable {
    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;

    @ApiModelProperty(value = "", name = "workerAbility")
    private Double workerAbility;

    @ApiModelProperty(value = "", name = "activeness")
    private Double activeness;

    @ApiModelProperty(value = "", name = "workerDiversity")
    private Double workerDiversity;

    @ApiModelProperty(value = "", name = "taskRelevance")
    private Double taskRelevance;

    @ApiModelProperty(value = "", name = "workerCost")
    private Double workerCost;
}
```

5.4.24 MultiObjectiveRecommendResult

```
public class MultiObjectiveRecommendResult {
    @ApiModelProperty(value = "", name = "userid")
    private Integer userid;

    @ApiModelProperty(value = "", name = "taskid")
    private Integer taskid;
}
```

5.5 数据库表

> 说明：数据库表中的字段和PO中属性一一对应，故在此处只列出表和PO的对应关系，详细设计见 5.4 PO视角

数据库表	PO	说明	备注
user_info	User	存储用户信息	
user_tag	UserTag	存储用户和标签的关联信息	
worker_ability	WorkerAbility	存储工人能力信息	**迭代三新增**
worker_context	WorkerContext	存储工人测试环境信息	**迭代三新增**
task_info	Task	存储任务信息	**迭代三增加部分字段**
task_tag	TaskTag	存储任务和标签的关联信息	
task_user	TaskUser	存储任务和选取任务的用户之间的关联信息	
report_info	Report	存储报告信息	
flaw	Flaw	存储缺陷信息	
flaw_pic	FlawPic	存储缺陷和图片url的关联信息	
flaw_evaluation	(Flaw)Evaluation	存储缺陷的评价信息	
flaw_eva_like	(Flaw)EvaLike	存储缺陷评价的点赞和点踩信息	**迭代三新增**
score	(Flaw)Score	存储缺陷的评分信息	
similarity	(Flaw)Similarity	存储缺陷之间的相似度，避免对py微服务的多次重复请求	
rule	RecommendRule	存储推荐规则信息	
rule_factor	RecommendRuleFactor	存储推荐规则和影响因子的关联信息	
multi_objective_recommend_factor	MultiObjectiveRecommendFactor	存储多目标优化推荐因素的信息	**迭代三新增**

数据库表	PO	说明	备注
multi_objective_recommend_result	MultiObjectiveRecommendResult	存储多目标优化推荐结果的信息	**迭代三新增**
task_recruit_stop_recommend_factor	TaskRecruitStopRecommendFactor	存储任务停止招募的因素信息	**迭代三新增**

6. 附录——pipeline脚本与Makefile

详细解释见`部署文档--使用Jenkins流水线进行CI/CD集成`，建议两者结合阅读

6.1 前端项目

6.1.2 pipeline脚本

```
pipeline {
    agent {label 'slave'}

    stages {

        stage('pull') {
            steps {
                retry(3){
                    timeout(time:10,unit:'SECONDS'){
                        git branch: 'develop', credentialsId: 'ssh_key', url:
'git@git.nju.edu.cn:monian/frontend-collect.git'
                    }
                }
                sh 'pwd'
                sh 'ls -al'
            }
        }

        stage('prepare') {
            steps {
                sh 'make prepare'
            }
        }

        stage('clean') {
            steps {
                script{
                    try{
                        sh 'make clean'
                    }catch(err){
                    }
                }
            }
        }

        stage('build') {
            steps {
                script{
                    try{
                        retry(3){
                            sleep(time: 10, unit: 'SECONDS')
                            sh 'make build'
                        }
                    }catch(err){
                        sh 'make end'
                        throw err
                    }
                }
            }
        }
```

```
        stage('test') {
            steps {
                sh 'make test'
            }
        }

        stage('deploy') {
            steps {
                sh 'make deploy'
            }
        }

        stage('end'){
            steps{
                sh 'make end'
            }
        }


    }
    post {
        success {
            updateGitlabCommitStatus name: 'CICD', state: 'success'
        }
        failure {
            updateGitlabCommitStatus name: 'CICD', state: 'failed'
        }
    }


}
```

6.1.3 Makefile

```
NPM := npm
IMG_NAME := frontend-collect
CON_NAME := Frontend-Collect
BACK_CON := Backend-Collect

all: prepare clean build test deploy end

prepare:
    @echo "==Prepare=="
    ln -sf /usr/local/node/bin/node /usr/bin/node
    rm -f ./package-lock.json

clean:
    @echo "==Clean=="
# ifneq ($(shell docker ps | grep $(BACK_CON)),)
#    docker stop $(BACK_CON)
# endif
ifneq ($(shell docker ps | grep $(CON_NAME)),)
    docker stop $(CON_NAME)
endif
ifneq ($(shell docker ps -a | grep $(CON_NAME)),)
    docker rm $(CON_NAME)
endif
ifneq ($(shell docker images | grep $(IMG_NAME)),)
    docker rmi $(IMG_NAME)
endif

build:
    @echo "==Build=="
    $(NPM) install
    $(NPM) run build
```

```
test:
    @echo "==Test=="

deploy:
    @echo "==Deploy=="
    docker build -t $(IMG_NAME) .
    docker run -p 8082:80  -v /export/nginx/nginx.conf:/etc/nginx/nginx.conf -d --name $(CON_NAME)
$(IMG_NAME)

end:
# ifeq ($(shell docker ps | grep $(BACK_CON)),)
#   docker restart $(BACK_CON)
# endif
```

6.2 后端项目

6.2.2 pipeline脚本

```
pipeline {
    agent {label 'slave-python'}

    stages {

        stage('pull') {
            steps {
                retry(3){
                    timeout(time:10,unit:'SECONDS'){
                        git branch: 'develop', credentialsId: 'ssh_key', url:
'git@git.nju.edu.cn:monian/backend-collect.git'
                    }
                }
                sh 'pwd'
                sh 'ls -al'
            }
        }

        stage('clean') {
            steps {
                sh 'make clean'
            }
        }

        stage('build') {
            steps {
                sh 'make build'
            }
        }

        stage('test') {
            steps {
                script{
                    try{
                        sh 'make test'
                        jacoco()
                    }catch(err){
                        sh 'make deploy'
                        throw err
                    }
                }
            }
        }

        stage('deploy') {
            steps {
                sh 'make deploy'
            }
        }
```

```
    }
    post {
        success {
            updateGitlabCommitStatus name: 'CICD', state: 'success'
        }
        failure {
            updateGitlabCommitStatus name: 'CICD', state: 'failed'
        }
    }


}
```

6.2.3 Makefile

```
MVN  := mvn
MVN_ARG := -P prod
IMG_NAME := backend-collect
CON_NAME := Backend-Collect

all: clean build test deploy

clean:
    @echo "==Clean=="
ifneq ($(shell docker ps | grep $(CON_NAME)),)
    docker stop $(CON_NAME)
endif
ifneq ($(shell docker ps -a | grep $(CON_NAME)),)
    docker rm $(CON_NAME)
endif
ifneq ($(shell docker images | grep $(IMG_NAME)),)
    docker rmi $(IMG_NAME)
endif
    $(MVN) clean $(MVN_ARG)

build:
    @echo "==Build=="
    $(MVN) package $(MVN_ARG) -DskipTests

test:
    @echo "==Test=="
    $(MVN) test $(MVN_ARG)

deploy:
    @echo "==Deploy=="
    docker build -t $(IMG_NAME) .
    docker run -d -p 8080:8080 --name $(CON_NAME) $(IMG_NAME)
MVN  := /usr/local/maven/bin/mvn
MVN_ARG := -P prod
IMG_NAME := backend-collect
CON_NAME := Backend-Collect

all: clean build test deploy

clean:
    @echo "==Clean=="
ifneq ($(shell docker ps | grep $(CON_NAME)),)
    docker stop $(CON_NAME)
endif
ifneq ($(shell docker ps -a | grep $(CON_NAME)),)
    docker rm $(CON_NAME)
endif
ifneq ($(shell docker images | grep $(CON_NAME)),)
    docker rmi $(IMG_NAME)
endif
    $(MVN) clean $(MVN_ARG)
```

```
build:
    @echo "==Build=="
    $(MVN) package $(MVN_ARG) -DskipTests

test:
    @echo "==Test=="
    $(MVN) test $(MVN_ARG)

deploy:
    @echo "==Deploy=="
    docker build -t $(IMG_NAME) .
    docker run -d -p 8080:8080 --name $(CON_NAME) $(IMG_NAME)
```

6.3 python项目

6.3.2 pipeline脚本

```
pipeline {
    agent {label 'slave-python'}

    stages {

        stage('pull') {
            steps {
                retry(3){
                    timeout(time:10,unit:'SECONDS'){
                        git branch: 'master', credentialsId: 'ssh_key', url:
'git@git.nju.edu.cn:monian/python-collect.git'
                    }
                }
                sh 'pwd'
                sh 'ls -al'
            }
        }

        stage('clean') {
            steps {
                script{
                    try{
                        sh 'make clean'
                    }catch(err){
                    }
                }
            }
        }

        stage('deploy') {
            steps {
                sh 'make deploy'
            }
        }



    }
    post {
        success {
            updateGitlabCommitStatus name: 'CICD', state: 'success'
        }
        failure {
            updateGitlabCommitStatus name: 'CICD', state: 'failed'
        }
    }


}
```

6.3.3 Makefile

```
IMG_NAME := python-collect
CON_NAME := Python-Collect

all: clean deploy

clean:
    @echo "==Clean=="
ifneq ($(shell docker ps | grep $(CON_NAME)),)
    docker stop $(CON_NAME)
endif
ifneq ($(shell docker ps -a | grep $(CON_NAME)),)
    docker rm $(CON_NAME)
endif
ifneq ($(shell docker images | grep $(IMG_NAME)),)
    docker rmi $(IMG_NAME)
endif

deploy:
    @echo "==Deploy=="
    docker build -t $(IMG_NAME) .
    docker run -p 5000:5000 -v /pythonProject:/pythonProject -d --name $(CON_NAME) $(IMG_NAME)
```