项目设计文档

李大磊 191250067

曹璐 191250003

陆昱宽 191820133

熊智星 191250162

2022年4月25日

**摘要**

本文档为Volatile小组在2022年春学期《软件工程与计算三》课程作业迭代二中为项目所撰写的项目设计文档。

**目录**

[1 引言 3](#_Toc925706568)

[1.1 编写目的 3](#_Toc1978749508)

[1.2 对象与范围 3](#_Toc911223514)

[1.3 参考文献 3](#_Toc1227713041)

[1.4 名词与术语 4](#_Toc1150199711)

[2 逻辑视角 4](#_Toc1906236130)

[2.1 分层架构包图 4](#_Toc1949590964)

[2.2 逻辑包图 5](#_Toc469846022)

[3 组合视角 6](#_Toc404721735)

[3.1 物理包划分 6](#_Toc1077490096)

[3.2 物理包图 7](#_Toc1793931968)

[4 接口视角 8](#_Toc2091665943)

[4.1 模块的职责 8](#_Toc322202611)

[4.2 模块的接口规范 8](#_Toc1453008990)

[4.2.1 用户界面层分解 8](#_Toc1685544893)

[4.2.2 业务逻辑层分解 10](#_Toc1496229074)

[4.2.2.1 EntityService的接口规范 10](#_Toc88540348)

[4.2.2.2 RelationshipService的接口规范 11](#_Toc2038945112)

[4.2.2.3 DomainService的接口规范 12](#_Toc1153942205)

[4.2.2.4 FileService的接口规范 13](#_Toc381823378)

[4.2.2.5 UsersService的接口规范 14](#_Toc624376810)

[4.2.3 数据层分解 16](#_Toc1295946428)

[4.2.3.1 EntityMapper的接口规范 16](#_Toc1192467522)

[4.2.3.2 RelationshipMapper的接口规范 17](#_Toc1484248450)

[4.2.3.3 DomainMapper的接口规范 18](#_Toc593655598)

[4.2.3.4 TypeMapper的接口规范 19](#_Toc360611624)

[5 信息视角 20](#_Toc600712734)

[5.1 VO定义 20](#_Toc858295791)

[5.1.1 Result 20](#_Toc729702438)

[5.1.2 Entity 20](#_Toc1977251096)

[5.1.3 Relationship 21](#_Toc1497216794)

[5.1.4 Domain 21](#_Toc1656764859)

[5.1.5 DomainQuery 21](#_Toc974018211)

[5.1.6 User 22](#_Toc56231196)

[5.1.7 LoginVO 22](#_Toc184906492)

[5.1.8 RegisterVO 22](#_Toc314573835)

[5.2 数据库表 23](#_Toc2085189578)

[6 Pipeline脚本 23](#_Toc995602053)

# 1 引言

## 1.1 编写目的

本文档提供**COLLECT协作式众包测试平台**的软件架构概览，采用若干架构视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volatile团队内部的开发和管理人员，参考了RUP的《软件架构文档模板》，用于指导下一循环的代码开发和测试工作。

## 1.3 参考文献

《软件需求规格说明书》, Heap;

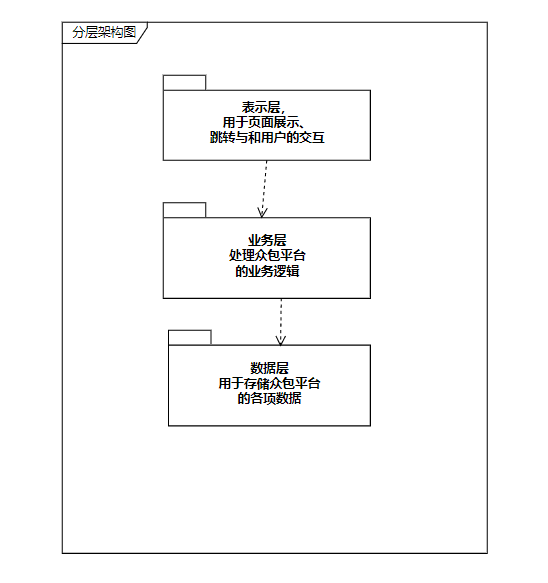
《软件架构文档模板》, Rational Software Corporation;

## **1.4 名词与术语**

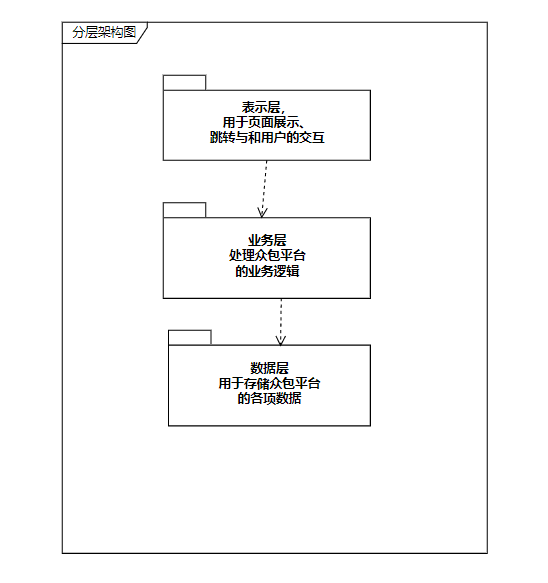
无

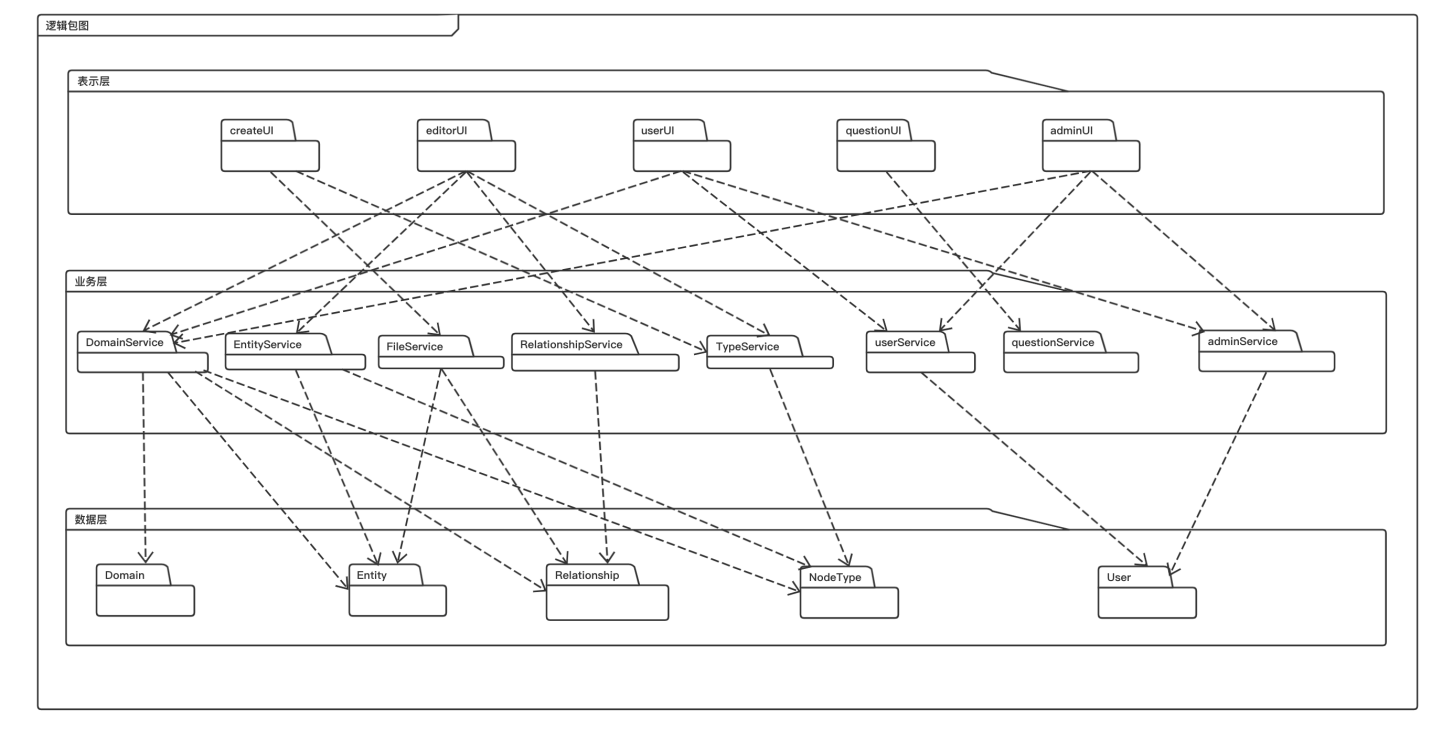
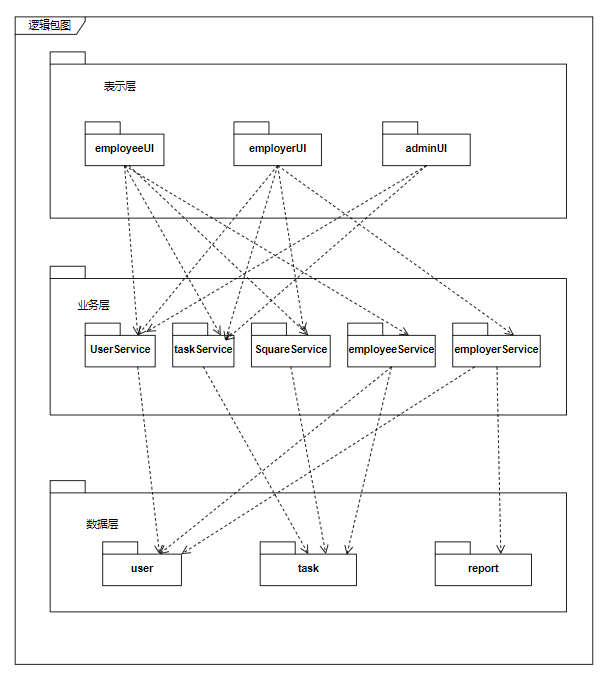
# 2 逻辑视角

## 2.1 分层架构包图



## 2.2 逻辑包图



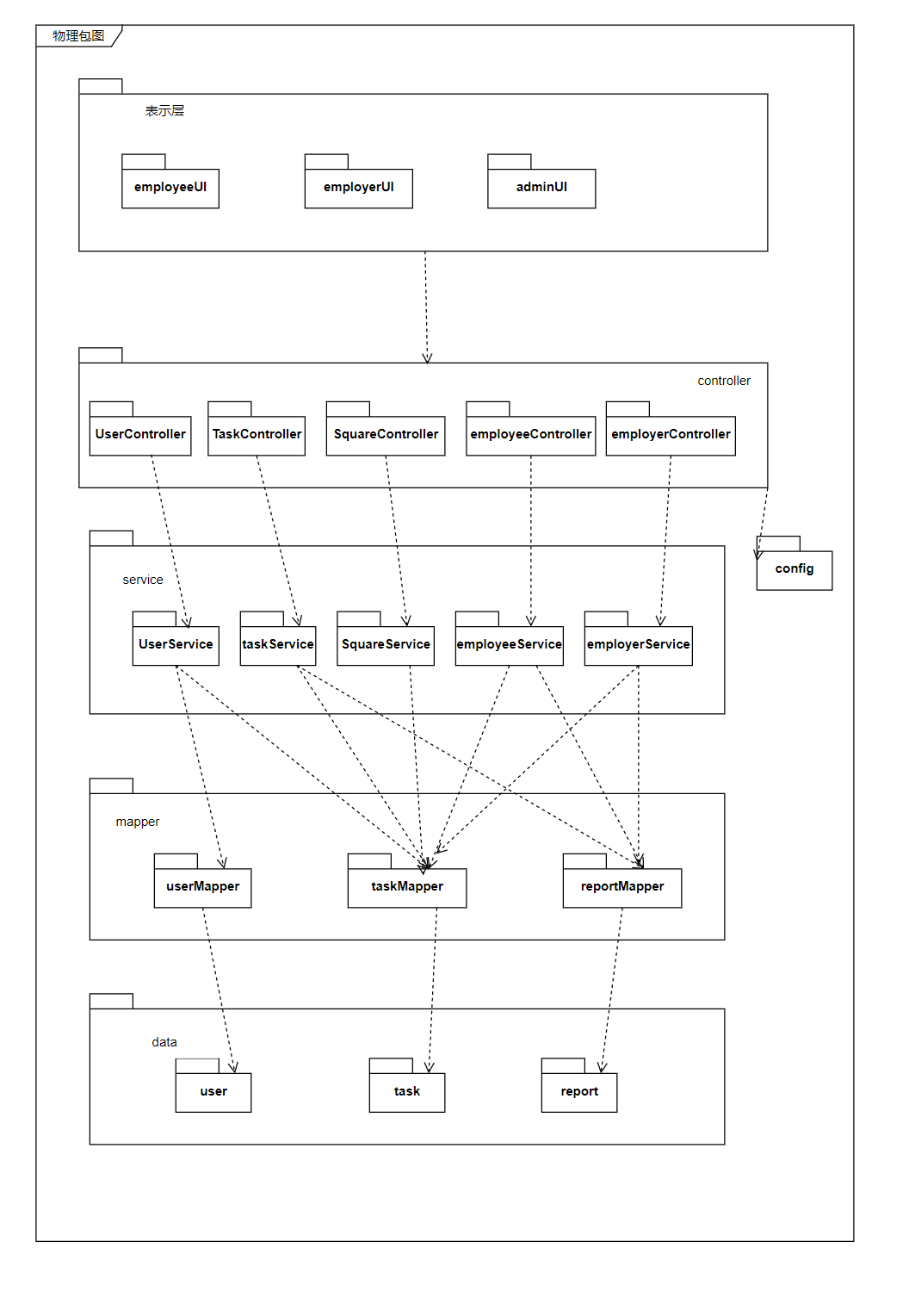


# 3 组合视角

## 3.1 物理包划分

|  |  |
| --- | --- |
| 开发包 | 依赖的其他开发包 |
| commonUtils |  |
| config |  |
| controller | service, commonUtils, config, utils |
| entity |  |
| service | entity |
| impl | service, utils |
| mapper | entity |
| utils |  |

## 3.2 物理包图



# 4 接口视角

## 4.1 模块的职责

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 客户端用户界面 |
| 接口模块 | 负责客户端和服务器端的通信和数据传递 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 数据服务层 | 抽象出的数据操作接口 |
| 数据层 | 负责数据的持久化和访问 |

## 4.2 模块的接口规范

### 4.2.1 用户界面层分解

|  |  |
| --- | --- |
| 服务名 | 服务 |
| employee. browserUndertakingTasks | 浏览进行中的任务 |
| employee. browserFinishedTasks | 浏览已完成的任务 |
| employee. browserTaskDetail | 浏览任务细节 |
| employee.uploadTestReport | 上传测试报告 |
| employee.reportDetail | 查看报告细节 |
| employee. uploadCooperationReport | 上传协作报告 |
| employee. cooperationReportDetail | 查看协作报告细节 |
| employee. cooperateReport | 主动参与协作 |
| employee. cooperatingList | 获取协作中列表 |
| employee. cooperatedList | 获取已协作列表 |
| employer. browserChecked | 更改任务状态 |
| employer. checkReports | 审核报告 |
| employer. reportDetail | 查看报告细节 |
| employer. cooperationReportDetail | 查看协作报告细节 |
| task. publishTask | 发布任务 |
| task. acceptTask | 接受任务 |
| square. browserTasks | 浏览任务广场 |
| square. employeeTaskTetail | 浏览任务细节 |
| square. taskTetail | 浏览任务细节 |
| square. visitorBrowserTasks | 游客状态浏览任务列表 |
| report. checkReports | 获取一个报告列表 |
| report. scoreReport | 给报告评分 |
| report. cooperationReportList | 获取协作报告列表 |
| report. showReportScore | 获取报告评分 |
| report. getSimilarityGraph | 生成相似关系图 |
| report. getCooperationTree | 生成协作树 |
| user. userLogin | 用户登录 |
| user. userRegister | 用户注册 |
| user. setUserProfile | 设置用户资料 |
| user. getUserData | 获取用户资料 |
| admin. browserAllTasks | 浏览所有任务 |
| admin. browserTaskDetail | 查看任务细节 |
| admin. setRecommendRule | 设置任务推荐规则 |
| admin. getRecommendRule | 获取任务推荐规则 |
| admin. setAlgorithm | 设置相似度算法 |
| admin. getAlgorithm | 获取相似度算法 |

### 4.2.2 业务逻辑层分解

#### 4.2.2.1 EmployeeService的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| EmployeeService | | 负责接包方的职责 |
| **提供的服务（供接口）** | | |
| EmployeeService. browserUndertakingTasks | 语法 | BrowserUndertakingTasksVO browserUndertakingTasks*(*BrowserUndertakingTasksDTO browserUndertakingTasksDTO*)*; |
| 前置条件 | BrowserUndertakingTasksDTO不为空 |
| 后置条件 | 浏览进行中的任务 |
| EmployeeService. browserFinishedTasks | 语法 | BrowserFinishedTasksVO browserFinishedTasks*(*BrowserFinishedTasksDTO browserFinishedTasksDTO*)* |
| 前置条件 | BrowserFinishedTasksDTO不为空 |
| 后置条件 | 浏览已完成的任务 |
| EmployeeService. browserTaskDetail | 语法 | BrowserTaskDetailVO browserTaskDetail*(*BrowserTaskDetailDTO browserTaskDetailDTO*)*; |
| 前置条件 | BrowserTaskDetailDTO不为空 |
| 后置条件 | 浏览任务细节 |
| EmployeeService.uploadTestReport | 语法 | UploadTestReportVO uploadTestReport*(*UploadTestReportDTO uploadTestReportDTO*)*; |
| 前置条件 | UploadTestReportDTO不为空 |
| 后置条件 | 上传测试报告 |
| EmployeeService. browserTaskDetail | 语法 | BrowserTaskDetailVO browserTaskDetail*(*BrowserTaskDetailDTO browserTaskDetailDTO*)*; |
| 前置条件 | BrowserTaskDetailDTO不为空 |
| 后置条件 | 浏览任务细节 |
| EmployeeService. uploadCooperationReport | 语法 | UploadCooperationReportVO uploadCooperationReport*(*UploadCooperationReportDTO uploadCooperationReportDTO*)*; |
| 前置条件 | uploadCooperationReportDTO不为空 |
| 后置条件 | 上传协作报告 |
| EmployeeService. reportDetail | 语法 | ReportDetailVO reportDetail*(*ReportDetailDTO reportDetailDTO*)*; |
| 前置条件 | reportDetailDTO不为空 |
| 后置条件 | 浏览报告细节 |
| EmployeeService. cooperationReportDetail | 语法 | CooperationReportDetailVO cooperationReportDetail*(*CooperationReportDetailDTO cooperationReportDetailDTO*)*; |
| 前置条件 | cooperationReportDetailDTO不为空 |
| 后置条件 | 获取协作报告细节 |
| EmployeeService.cooperateReport | 语法 | CooperateReportVO cooperateReport*(*CooperateReportDTO cooperateReportDTO*)*; |
| 前置条件 | cooperateReportDTO 不为空 |
| 后置条件 | 参与到协作中 |
| EmployeeService. cooperatingList | 语法 | CooperatingListVO cooperatingList*()*; |
| 前置条件 | 无 |
| 后置条件 | 获取协作中报告列表 |
| EmployeeService. cooperatedList | 语法 | CooperatedListVO cooperatedList*()*; |
| 前置条件 | 无 |
| 后置条件 | 获取已协作报告列表 |

#### 4.2.2.2 EmployerService的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| EmployerService | | 负责发包方的职责 |
| **提供的服务（供接口）** | | |
| EmployerService. checkReports | 语法 | ReportDetailVO reportDetail*(*ReportDetailDTO reportDetailDTO*)* |
| 前置条件 | ReportDetailDTO不为空 |
| 后置条件 | 审核测试报告 |
| EmployerService. reportDetail | 语法 | CheckReportsVO checkReports*(*CheckReportsDTO checkReportsDTO*)* |
| 前置条件 | CheckReportsDTO不为空 |
| 后置条件 | 查看报告细节 |
| EmployerService. browserChecked | 语法 | BrowserCheckedVO browserChecked*(*BrowserCheckedDTO browserCheckedDTO*)*; |
| 前置条件 | BrowserCheckedDTO不为空 |
| 后置条件 | 更改任务状态 |
| EmployerService. browserTaskDetail | 语法 | BrowserTaskDetailVO browserTaskDetail*(*BrowserTaskDetailDTO browserTaskDetailDTO*)*; |
| 前置条件 | browserTaskDetailDTO不为空 |
| 后置条件 | 浏览报告细节 |
| EmployerService. cooperationReportDetail | 语法 | CooperationReportDetailVO cooperationReportDetail*(*CooperationReportDetailDTO cooperationReportDetailDTO*)*; |
| 前置条件 | cooperationReportDetailDTO不为空 |
| 后置条件 | 查看协作报告详情 |

#### 4.2.2.3 SquareService的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| SquareService | | 负责对任务广场的职责 |
| **提供的服务（供接口）** | | |
| SquareService. browserTasks | 语法 | BrowserTasksVO browserTasks*()* |
| 前置条件 | 存在已发布任务 |
| 后置条件 | 查看任务广场列表 |
| SquareService. employeeTaskTetail | 语法 | EmployeeTaskDetailVO employeeTaskTetail*(*EmployeeTaskTetailDTO employeeTaskTetailDTO*)* |
| 前置条件 | employeeTaskTetailDTO不为空 |
| 后置条件 | 接包方查看任务细节 |
| SquareService.taskDetail | 语法 | TaskDetailVO taskDetail*(*TaskDetailDTO employerTaskTetailDTO*)*; |
| 前置条件 | employerTaskTetailDTO不为空 |
| 后置条件 | 发包方浏览任务细节 |

#### 4.2.2.4 TaskService的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| TaskService | | 负责对文件流控制 |
| **提供的服务（供接口）** | | |
| TaskService. publishTask | 语法 | PublishTaskVO publishTask*(*TaskPublishDTO taskPublishDTO*)* |
| 前置条件 | TaskPublishDTO不为空 |
| 后置条件 | 发布任务 |
| TaskService. acceptTask | 语法 | AcceptTaskVO acceptTask*(*AcceptTaskDTO acceptTaskDTO*)* |
| 前置条件 | acceptTaskDTO不为空 |
| 后置条件 | 接受任务 |

#### 4.2.2.5 TaskService的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| AdminService | | 负责管理员相关的操作 |
| **提供的服务（供接口）** | | |
| AdminService. browserAllTasks | 语法 | BrowserAllTasksVO browserAllTasks*()*; |
| 前置条件 | 管理员身份正确 |
| 后置条件 | 获取所有任务 |
| AdminService. browserTaskDetail | 语法 | BrowserTaskDetailVO browserTaskDetail*(*BrowserTaskDetailDTO browserTaskDetailDTO*)*; |
| 前置条件 | browserTaskDetailDTO 不为空 |
| 后置条件 | 查看任务细节 |
| AdminService. setRecommendRule | 语法 | SetRecommendRuleVO setRecommendRule*(*SetRecommendRuleDTO setRecommendRuleDTO*)*; |
| 前置条件 | setRecommendRuleDTO不为空 |
| 后置条件 | 添加推荐规则 |
| AdminService. getRecommendRule | 语法 | GetRecommendRuleVO getRecommendRule*()*; |
| 前置条件 | 管理员身份正确 |
| 后置条件 | 获得推荐规则 |
| AdminService. setAlgorithm | 语法 | SetAlgorithmVO setAlgorithm*(*SetAlgorithmDTO setAlgorithmDTO*)*; |
| 前置条件 | setAlgorithmDTO不为空 |
| 后置条件 | 设置相似度算法 |
| AdminService. getAlgorithm | 语法 | GetAlgorithmVO getAlgorithm*()*; |
| 前置条件 | 管理员身份正确 |
| 后置条件 | 获取相似度计算算法 |

### 4.2.3 数据层分解

#### 4.2.3.1 TaskMapper的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| TaskMapper | | 负责对任务的访问控制 |
| **提供的服务（供接口）** | | |
| TaskMapper. get\_all\_by\_user\_id\_without\_paging | 语法 | public List*<*Task*>* get\_all\_by\_user\_id\_without\_paging*(* @Param*(*"user\_id"*)*long user\_id *)*; |
| 前置条件 | 用户存在 |
| 后置条件 | 获取用户的所有任务 |
| TaskMapper. get\_all\_without\_paging | 语法 | public List*<*Task*>* get\_all\_without\_paging*()*; |
| 前置条件 | 无 |
| 后置条件 | 获得任务大厅的所有任务 |
| TaskMapper. update\_task\_state | 语法 | public void update\_task\_state*(* @Param*(*"task\_id"*)* long task\_id, @Param*(*"task\_state"*)* int task\_state*)*; |
| 前置条件 | 任务已存在 |
| 后置条件 | 更新任务状态 |
| TaskMapper. get\_by\_task\_id | 语法 | public Task get\_by\_task\_id*(*@Param*(*"id"*)* long task\_id *)*; |
| 前置条件 | 任务已存在 |
| 后置条件 | 获取任务信息 |

#### 4.2.3.2 UserMapper的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | **职责** | |
| UserMapper | 负责对用户访问控制 | |
| **提供的服务（供接口）** | | |
| UserMapper. get\_by\_id | 语法 | User get\_by\_id*(*@Param*(*"user\_id"*)* long id *)*; |
| 前置条件 | 用户已存在 |
| 后置条件 | 根据其他输入，获得用户信息 |
| UserMapper. get\_by\_phone\_number | 语法 | User get\_by\_phone\_number *(*@Param*(*"phone\_number"*)* String phone\_number *)*; |
| 前置条件 | 用户已注册且存在手机号 |
| 后置条件 | 根据输入，返回用户信息 |
| UserMapper. get\_role\_by\_id | 语法 | Integer get\_role\_by\_id*(*@Param*(*"user\_id"*)* Long id *)*; |
| 前置条件 | 用户已存在 |
| 后置条件 | 根据其他userId，获得用户角色 |
| UserMapper. insert | 语法 | void insert*(* @Param*(*"user"*)* User user *)*; |
| 前置条件 | 用户不存在 |
| 后置条件 | 插入用户数据 |
| UserMapper. update\_nick\_name | 语法 | void insert*(* @Param*(*"user"*)* User user *)*; |
| 前置条件 | 用户不存在 |
| 后置条件 | 插入用户数据 |

#### 4.2.3.3 ReportMapper的接口规范

|  |  |  |
| --- | --- | --- |
| **模块** | | **职责** |
| ReportMapper | | 负责对域的访问控制 |
| **提供的服务（供接口）** | | |
| ReportMapper. get\_by\_report\_id | 语法 | public Report get\_by\_report\_id*(*@Param*(*"id"*)* long report\_id *)*; |
| 前置条件 | 测试报告已存在 |
| 后置条件 | 根据输入返回报告信息 |
| ReportMapper. get\_all\_by\_user\_id\_without\_paging | 语法 | public List*<*Report*>* get\_all\_by\_user\_id\_without\_paging*(* @Param*(*"user\_id"*)*long user\_id *)*; |
| 前置条件 | 数据库存在该域 |
| 后置条件 | 根据输入，获得用户所有报告信息 |
| ReportMapper. get\_all\_by\_user\_id\_and\_task\_id | 语法 | public List*<*Report*>* get\_all\_by\_user\_id\_and\_task\_id*(* @Param*(*"user\_id"*)*long user\_id ,@Param*(*"task\_id"*)*long task\_id*)*; |
| 前置条件 | 数据库存在该域 |
| 后置条件 | 根据输入，获得报告信息 |
| ReportMapper. get\_all\_by\_task\_id\_without\_paging | 语法 | public List*<*Report*>* get\_all\_by\_task\_id\_without\_paging*(* @Param*(*"task\_id"*)*long task\_id *)*; |
| 前置条件 | 域名task\_id不为空且存在 |
| 后置条件 | 根据输入，报告列表 |

# 5 信息视角

## 5.1 VO定义

### 5.1.1 Result

|  |  |  |
| --- | --- | --- |
| 含义 | 属性 | 字段 |
| 返回码 | Integer | code |
| 返回消息 | String | message |

### 5.1.2 File

|  |  |  |
| --- | --- | --- |
| 含义 | 属性 | 字段 |
| 文件id | long | fileId |
| 文件url | String | fileURL |
| 文件名称 | String | fileName |

### 5.1.3 Report

|  |  |  |
| --- | --- | --- |
| 含义 | 属性 | 字段 |
| id标识码 | Long | reportId |
| 所属任务id | Long | taskId |
| 报告名 | String | reportName |
| 报告状态 | int | reportState |
| 缺陷解释 | String | defectExplain |
| 缺陷复现步骤 | String | defectReproductionStep |
| 测试设备信息 | String | testEquipmentInformation |
| 缺陷截图 | Array<object> | defectPictureList |

### 5.1.1 Task

|  |  |  |
| --- | --- | --- |
| 含义 | 属性 | 字段 |
| 任务id | long | taskId |
| 任务开始时间 | long | taskStartTime |
| 任务结束时间 | long | taskEndTime |
| 需要的工人总数 | int | workerNumTotal |
| 任务状态 | int | taskState |
| 任务名称 | String | taskName |
| 还需要的工人数 | int | workerNumLeft |
| 任务类型 | int | taskType |
| 报告id列表 | Array<Integer> | reportIdList |
| 需求说明文件列表 | Array<object> | requirementDescriptionFileList |
| 可执行文件列表 | Array<object> | executableFileList |

## 5.2 数据库表

数据库表包含user表、task表和report表

# 6.pipeline脚本

## 6.1 后端

|  |
| --- |
| node("slave1") {  def workspace = pwd()  def git\_branch = 'master'  def git\_repository = 'git@git.nju.edu.cn:191820133/backend-volatile.git'  def vm\_ip = '124.222.135.47'  def vm\_port = '22'  def vm\_user = 'lyk'  def vm\_project\_place = "/usr/local/src"  def vm\_target\_place = "/usr/local/src/target/"  def IMAGE\_NAME = 'volatile\_backend'  def IMAGE\_NAME\_WITH\_TAG = 'volatile\_backend:latest'  def IMAGE\_TO\_RUN = 'lyklove/volatile\_backend:latest'  def CONTAINER\_NAME = 'volatile\_backend'  stage('clone from gitlab into slave\'s workspace') {  echo "workspace: ${workspace}"  git branch: "${git\_branch}", url: "${git\_repository}"  }  stage('cd to build context') {  echo "the context now is:"  sh "ls -al"  sh "cd ${workspace}"  echo "cd to build context, now the context is:"  sh "ls -al"  }  stage('build jar on slave machine, jacoco file generated') {  sh 'mvn --version'  sh "mvn clean package jacoco:report -Dmaven.test.failure.ignore=true"  echo "build finish on ${vm\_ip}"  }  stage( 'testing, using jacoco' ) {  jacoco (  execPattern: '\*\*/target/jacoco.exec',  classPattern: '\*\*/classes',  sourcePattern: '\*\*/src/main/java',  exclusionPattern: '\*\*/src/test\*',  // inclusionPattern: '\*\*/com/hel/auto/service/\*.class,\*\*/com/hel/auto/controller/\*.class',  )  }  stage("build docker image"){  sh "docker build -t ${IMAGE\_NAME} ."  }  stage("login to dockerhub"){  withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {  sh 'docker login -u $username -p $password'  }  }  stage("push to dockerhub"){  echo "begin push to dockerhub"  sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"  sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"  }  stage("clean previous image and container"){  sh "docker container rm -f ${CONTAINER\_NAME}"  sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"  sh "docker image rm ${IMAGE\_TO\_RUN}"  }  stage( "pull image" ){  sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"  }  stage("run container") {  sh "docker image ls"  sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"  }  stage("signal gitlab: deployed"){  updateGitlabCommitStatus name: 'deployed', state: 'success'  }  } |

## 6.2 前端

|  |
| --- |
| node("slave1") {  def workspace = pwd()  def git\_branch = 'master'  def git\_repository = 'git@git.nju.edu.cn:191820133/frontend-volatile.git'  def vm\_ip = '124.222.135.47'  def vm\_port = '22'  def vm\_user = 'lyk'  def vm\_project\_place = "/usr/local/src"  def vm\_target\_place = "/usr/local/src/target/"  def IMAGE\_NAME = 'volatile\_frontend'  def IMAGE\_NAME\_WITH\_TAG = 'volatile\_frontend:latest'  def IMAGE\_TO\_RUN = 'lyklove/volatile\_frontend:latest'  def CONTAINER\_NAME = 'volatile\_frontend'  stage('clone from gitlab into slave\'s workspace') {  echo "workspace: ${workspace}"  git branch: "${git\_branch}", url: "${git\_repository}"  }  stage('cd to build context') {  echo "the context now is:"  sh "ls -al"  sh "cd ${workspace}"  echo "cd to build context, now the context is:"  sh "ls -al"  }  stage('get version info'){  sh 'node -v'  sh 'npm -v'  sh 'vue -V'  sh 'npm list vue'  }  stage('build with npm') {  // sh 'npm config set registry http://registry.cnpmjs.org'  sh 'npm install'  sh 'npm run build'  echo "build finish on ${vm\_ip}"  }  // stage('npm run serve'){  //  // echo 'not using docker yet!!'  // sh 'npm run serve'  // }  stage("build docker image"){  sh "docker build -t ${IMAGE\_NAME} --no-cache ."  // sh "imageId=`docker images | grep #{IMAGE\_NAME} | awk '{print $3}'`"  }  stage("login to dockerhub"){  withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {  sh 'docker login -u $username -p $password'  }  }  stage("push to dockerhub"){  echo "begin push to dockerhub"  sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"  sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"  }  stage("clean previous image and container"){  sh "docker container rm -f ${CONTAINER\_NAME}"  sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"  sh "docker image rm ${IMAGE\_TO\_RUN}"  }  stage( "pull image" ){  sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"  }  stage("run container") {  sh "docker image ls"  sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"  }  stage("signal gitlab: deployed"){  updateGitlabCommitStatus name: 'deployed', state: 'success'  }  } |

## 6.3 python

|  |
| --- |
| node("slave1") {  def workspace = pwd()  def git\_branch = 'master'  def git\_repository = 'git@git.nju.edu.cn:191820133/ai-volatile.git'  def vm\_ip = '124.222.135.47'  def vm\_port = '22'  def vm\_user = 'lyk'  def vm\_project\_place = "/usr/local/src"  def vm\_target\_place = "/usr/local/src/target/"  def IMAGE\_NAME = 'volatile\_ai'  def IMAGE\_NAME\_WITH\_TAG = 'volatile\_ai:latest'  def IMAGE\_TO\_RUN = 'lyklove/volatile\_ai:latest'  def CONTAINER\_NAME = 'volatile\_ai'  stage('clone from gitlab into slave\'s workspace') {  echo "workspace: ${workspace}"  git branch: "${git\_branch}", url: "${git\_repository}"  }  stage('cd to build context') {  echo "the context now is:"  sh "ls -al"  sh "cd ${workspace}"  echo "cd to build context, now the context is:"  sh "ls -al"  }  stage("build docker image"){  sh "docker build -t ${IMAGE\_NAME} ."  }  // stage("login to dockerhub"){  // withCredentials([usernamePassword(credentialsId: 'DOCKERHUB\_KEY', passwordVariable: 'password', usernameVariable: 'username')]) {  // sh 'docker login -u $username -p $password'  // }  // }  //  stage("push to dockerhub"){  // echo "begin push to dockerhub"  sh "docker image tag ${IMAGE\_NAME\_WITH\_TAG} lyklove/${IMAGE\_NAME\_WITH\_TAG}"  // sh "docker image push lyklove/${IMAGE\_NAME\_WITH\_TAG}"  }  stage("clean previous image and container"){  sh "docker container rm -f ${CONTAINER\_NAME}"  // sh "docker image rm ${IMAGE\_NAME\_WITH\_TAG}"  // sh "docker image rm ${IMAGE\_TO\_RUN}"  }  // stage( "pull image" ){  // sh "docker pull lyklove/${IMAGE\_NAME\_WITH\_TAG}"  // }  stage("run container") {  sh "docker image ls"  sh "docker container run --name ${CONTAINER\_NAME} --net=host -d ${IMAGE\_TO\_RUN}"  }  stage("signal gitlab: deployed"){  updateGitlabCommitStatus name: 'deployed', state: 'success'  }  } |

# 7. Python算法

## 7.1 python算法设计

Author: 陆昱宽

### 7.1.1 报告相似度计算

分为两个步骤， 首先是文本相似度计算， 如果原报告和候选报告的文本相似度较高，则将该候选报告认作抄袭，直接返回，不将它纳入相似度计算的输入集合； 之后是语义相似度计算， 计算原报告和（没有抄袭的）候选报告的语义相似度，然后将相似度较高的报告返回

* 文本相似度： 目前采用TF\_IDF算法
* 语义相似度：目前有多个算法：CosineSimilarity、SemanticSearch、BM25

以上算法均采用策略模式，可以进行替换。 算法接口接受“algorithm”参数，可以由后端指定使用某算法

代码示例：策略模式

def get\_top\_n\_similar\_report\_ids(self, queried\_report, compared\_reports, N, algorithm):

queried\_report = [queried\_report]

# return get\_top\_n\_similar\_result\_ids( queried\_report, compared\_reports, N, algorithm)

if algorithm == 'CosineSimilarity':

textSimilarityStratefy = CosineTextSemanticSimilarityStrategy(queried\_report, compared\_reports)

elif algorithm == 'SemanticSearch':

textSimilarityStratefy = SemanticSearchTextSemanticSimilarityStrategy(queried\_report, compared\_reports)

else:

textSimilarityStratefy = BM25TextSemanticSimilarityStrategy(queried\_report, compared\_reports)

return textSimilarityStratefy.get\_top\_n\_semantically\_similar\_reports(N)

### 7.2.2任务推荐

任务推荐涉及到系统中的所有任务，因此Python端需要数据库中的所有相应数据，因此算法分为两部分：

1. Spring服务传给Python服务，系统中所有的用户及其参与过的任务的数据，为算法进行数据准备
2. Spring服务传给Python服务，要被推荐的用户及其参与过的任务的数据，算法针对已有的全部用户全部用户的数据， 和新用户的数据，进行相似度计算，相似度即推荐程度

任务推荐采用的是**基于物品的协同过滤**，也就是说， 对于要被推荐的用户， 已知他已经接受了若干任务， 那么对其中每一个任务， 已知选择该任务的以往用户还**喜欢**别的任务（这是算法已经算出来的）， 那么就认为这两个任务相似

* 我们通过计算用户特征和任务特征的相似度，来判断用户“喜欢”某个任务，这是个特征工程。 当然，任务必须是用户已参加的。

文章可参见：[https://www.jianshu.com/p/f306a37a7374](https://www.jianshu.com/p/f306a37a7374" \t "_blank)

### 7.2.3推荐规则

管理员用户可以设定推荐规则， 我们对规则的定义是：要增加或减少其影响因子的特征的集合。

由于我们的算法分两步，第一步需要得到用户喜欢的任务集合，这是个特征工程。 我们的判断依据是， 算出用户和任务的所有特征，每个特征都有其权重， 被重视的特征的权重增加， 被轻视的特征的权重减少。 管理员设定的“推荐规则”，就是设定哪些特征要被重视，哪些要被轻视

## 7.2 Python算法接口

Author： 陆昱宽

本项目Spring服务和相似报告，任务推荐等算法解耦， 后者由python服务实现， 并暴露rest接口，与Spring服务通信。 Spring只需作为客户，请求python接口，python服务即可进行相应计算并返回结果。

### 7.2.1 得到相似报告

python服务依据众包工人所提交的众测报告，检测已有报告与其的相似度，并将相似报告展示给众包工人

算法会返回相似度大于等于阈值的前N篇报告及其相似度（按降序排列）

相似度阈值和相似报告数可以为空，此时算法会采用默认的参数值

@app.route('/getRecommendedTasks', methods=['POST'])

def getRecommendedTasks():

pass

#### 7.2.1.1参数

计算报告相似度， 需要得到报告对象（其具有id， 报告文本等各种属性 ）， 该报告所属的任务的所有其他报告对象， 以及推荐规则

示例：

{

"report": {

"report\_id": 1,

"report\_name": "打解体军使习",

"defect\_reproduction\_step": "Duis sunt mollit",

"user\_id": 71,

"defect\_explain": "mollit ipsum eiusmod nulla",

"task\_id": 56,

"test\_equipment\_information": "Duis est consectetur labore",

"defect\_picture\_list": [

{

"file\_name": "金花用对",

"file\_url": "http://vdgcsax.my/ssvkyrnh",

"file\_id": "22"

},

{

"file\_name": "题进精一观",

"file\_url": "http://griwd.iq/cypu",

"file\_id": "92"

},

{

"file\_name": "人人才华团难",

"file\_url": "http://lbbhgv.wf/htcpccct",

"file\_id": "28"

},

{

"file\_name": "具价理主治她者",

"file\_url": "http://gzybui.mr/yrffm",

"file\_id": "81"

},

{

"file\_name": "表压老指积会被",

"file\_url": "http://dhhhh.mc/mpjkgs",

"file\_id": "37"

}

]

},

"related\_reports": [

{

"report\_id": 10,

"report\_name": "据把况别住",

"defect\_reproduction\_step": "Duis in ut ipsum voluptate",

"test\_equipment\_information": "anim",

"task\_id": "72",

"user\_id": "12",

"defect\_explain": "magna in elit laborum",

"defect\_picture\_list": "http://dummyimage.com/400x400"

},

{

"report\_id": 18,

"report\_name": "光约亲",

"defect\_reproduction\_step": "nostrud",

"test\_equipment\_information": "ea",

"task\_id": "65",

"user\_id": "13",

"defect\_explain": "Lorem irure proident",

"defect\_picture\_list": "http://dummyimage.com/400x400"

},

{

"report\_id": 32,

"report\_name": "织月县素量",

"defect\_reproduction\_step": "adipisicing",

"test\_equipment\_information": "nulla nisi aute non",

"task\_id": "96",

"user\_id": "57",

"defect\_explain": "elit dolor",

"defect\_picture\_list": "http://dummyimage.com/400x400"

},

{

"report\_id": 42,

"report\_name": "道三日其包",

"defect\_reproduction\_step": "sed Ut anim cillum proident",

"test\_equipment\_information": "aliqua",

"task\_id": "56",

"user\_id": "27",

"defect\_explain": "amet velit nisi",

"defect\_picture\_list": "http://dummyimage.com/400x400"

}

],

"similar\_report\_num": 20,

"algorithm": "CosineSimilarity"

}

#### 7.2.1.2返回值

返回与原报告相似的报告，按相似度降序排列

示例值：

[

{

"report\_id": 7,

"similarity": 2.922

},

{

"report\_id": 2,

"similarity": 1.855

},

{

"report\_id": 9,

"similarity": 0.543

}

]

### 7.2.2得到推荐任务

#### 7.2.2.1参数

{

"user": {

"user\_id": 77,

"user\_features": {

"proficiency": 0.65,

"preference": [

"performance test",

"function test"

],

"activity": 0.11,

"test\_devices": [

"Android",

"Windows",

"iOS"

]

},

"related\_tasks": [

{

"task\_id": 44,

"task\_features": {

"task\_type": "performance test",

"task\_introduction": "irurefadadadadassda non in amet",

"worker\_num\_left": 26,

"worker\_num\_total": 57,

"task\_difficulty": 0.432377,

"required\_test\_devices": [

"Android",

"Linux",

"iOS"

]

}

},

{

"task\_id": 82,

"task\_features": {

"task\_type": "performance test",

"task\_introduction": "ex eiuaddasasdasadssmod exercitation cupidatat",

"worker\_num\_left": 0,

"worker\_num\_total": 25,

"task\_difficulty": 0.61,

"required\_test\_devices": [

"Android"

]

}

},

{

"task\_id": 11,

"task\_features": {

"task\_type": "function test",

"task\_introduction": "suntfs似懂非懂上的",

"worker\_num\_left": 53,

"worker\_num\_total": 60,

"task\_difficulty": 0.3237,

"required\_test\_devices": [

"Android",

"Linux",

"iOS"

]

}

},

{

"task\_id": 99,

"task\_features": {

"task\_type": "function test",

"task\_introduction": "adipisicing veniam",

"worker\_num\_left": 2,

"worker\_num\_total": 11,

"task\_difficulty": 0.72,

"required\_test\_devices": [

"Android",

"iOS"

]

}

}

]

},

"recommendation\_threshold": 0.54,

"recommended\_task\_num": 34,

"algorithm": "ItemCF",

"recommendation\_rules": {

"emphasized\_user\_features": [

"proficiency",

"activity",

"test\_devices"

],

"desalted\_user\_features": null,

"emphasized\_task\_features": [

"task\_type",

"required\_test\_devices"

],

"desalted\_task\_features": [

"difficulty"

]

}

}

#### 7.2.2.2返回值

返回推荐的任务，按相似度降序排列

[

{

"task\_id": 44,

"recommendation\_degree": 1.8725890517453543

},

{

"task\_id": 32,

"recommendation\_degree": 1.7997126963057966

},

{

"task\_id": 11,

"recommendation\_degree": 1.1843293050020447

},

{

"task\_id": 82,

"recommendation\_degree": 1.1718505872700213

},

{

"task\_id": 34,

"recommendation\_degree": 1

},

{

"task\_id": 75,

"recommendation\_degree": 1

},

{

"task\_id": 70,

"recommendation\_degree": 1

}

]

### 7.2.3为推荐任务准备数据

java后端向算法传输数据库中所有用户和其参与的所有任务， 用于给算法提供数据， 后端在进行任务推荐之前必须调用该接口

#### 7.2.3.1参数

数据库中所有用户，及其参与的所有任务

{

"user\_list": [

{

"user\_id": 77,

"user\_features": {

"proficiency": 0.65,

"preference": [

"performance test",

"function test"

],

"activity": 0.11,

"test\_devices": [

"Android",

"Windows",

"iOS"

]

},

"related\_tasks": [

{

"task\_id": 44,

"task\_features": {

"task\_type": "performance test",

"task\_introduction": "irurefadadadadassda non in amet",

"worker\_num\_left": 26,

"worker\_num\_total": 57,

"task\_difficulty": 0.432377,

"required\_test\_devices": [

"Android",

"Linux",

"iOS"

]

}

},

{

"task\_id": 82,

"task\_features": {

"task\_type": "performance test",

"task\_introduction": "ex eiuaddasasdasadssmod exercitation cupidatat",

"worker\_num\_left": 0,

"worker\_num\_total": 25,

"task\_difficulty": 0.61,

"required\_test\_devices": [

"Android"

]

}

},

{

"task\_id": 11,

"task\_features": {

"task\_type": "function test",

"task\_introduction": "suntfs似懂非懂上的",

"worker\_num\_left": 53,

"worker\_num\_total": 60,

"task\_difficulty": 0.3237,

"required\_test\_devices": [

"Android",

"Linux",

"iOS"

]

}

},

{

"task\_id": 99,

"task\_features": {

"task\_type": "function test",

"task\_introduction": "adipisicing veniam",

"worker\_num\_left": 2,

"worker\_num\_total": 11,

"task\_difficulty": 0.72,

"required\_test\_devices": [

"Android",

"iOS"

]

}

}

]

},

{

"user\_id": 17,

"related\_tasks": [

{

"task\_id": 18,

"task\_features": {

"task\_type": "sunt",

"worker\_num\_left": 87,

"worker\_num\_total": 59,

"task\_introduction": "culpa id et eu",

"task\_difficulty": 93,

"required\_test\_devices": [

"eu velit ad ipsum non",

"fugiat aliquip reprehenderit",

"sed exercitation magna dolore commodo"

]

}

}

],

"user\_features": {

"proficiency": 18,

"test\_devices": [

"magna",

"laboris voluptate consequat aliqua est",

"occaecat velit anim cillum"

],

"activity": 21,

"preference": [

"eu",

"occaecat"

]

}

},

{

"user\_id": 61,

"related\_tasks": [

{

"task\_id": 49,

"task\_features": {

"task\_type": "mollit do qui",

"worker\_num\_left": 56,

"worker\_num\_total": 64,

"task\_introduction": "quis officia adipisicing",

"task\_difficulty": 10,

"required\_test\_devices": [

"ut"

]

}

},

{

"task\_id": 28,

"task\_features": {

"task\_type": "occaecat anim ut nulla",

"worker\_num\_left": 73,

"worker\_num\_total": 1,

"task\_introduction": "ad enim Excepteur",

"task\_difficulty": 73,

"required\_test\_devices": [

"nostrud in laboris proident dolore",

"cillum dolore Duis et laboris",

"voluptate in id",

"minim elit",

"id"

]

}

}

],

"user\_features": {

"proficiency": 53,

"test\_devices": [

"aliqua exercitation Ut magna est",

"Ut voluptate",

"ad consequat",

"est Lorem ex Ut nisi"

],

"activity": 55,

"preference": [

"irure consequat in non velit",

"ut ullamco nulla cillum reprehenderit",

"mollit sit ex sunt qui",

"officia nostrud cillum nulla eiusmod"

]

}

},

{

"user\_id": 89,

"related\_tasks": [

{

"task\_id": 70,

"task\_features": {

"task\_type": "veniam pariatur enim",

"worker\_num\_left": 63,

"worker\_num\_total": 56,

"task\_introduction": "sint eiusmod aliquip",

"task\_difficulty": 36,

"required\_test\_devices": [

"ea dolor id",

"ullamco occaecat"

]

}

},

{

"task\_id": 52,

"task\_features": {

"task\_type": "velit sed cillum commodo nostrud",

"worker\_num\_left": 41,

"worker\_num\_total": 72,

"task\_introduction": "enim veniam pariatur in",

"task\_difficulty": 5,

"required\_test\_devices": [

"ut",

"dolore incididunt in minim",

"sed",

"elit adipisicing laboris Excepteur qui",

"fugiat voluptate mollit adipisicing cupidatat"

]

}

},

{

"task\_id": 75,

"task\_features": {

"task\_type": "amet",

"worker\_num\_left": 39,

"worker\_num\_total": 28,

"task\_introduction": "laboris magna anim do ipsum",

"task\_difficulty": 93,

"required\_test\_devices": [

"nisi",

"dolore adipisicing pariatur et aliquip"

]

}

},

{

"task\_id": 34,

"task\_features": {

"task\_type": "consequat quis",

"worker\_num\_left": 19,

"worker\_num\_total": 91,

"task\_introduction": "reprehenderit ex dolore",

"task\_difficulty": 92,

"required\_test\_devices": [

"in culpa ipsum dolore nisi",

"aute",

"sed esse fugiat dolore",

"Ut commodo occaecat"

]

}

}

],

"user\_features": {

"proficiency": 50,

"test\_devices": [

"in ut Lorem cillum mollit",

"incididunt ad elit",

"anim incididunt occaecat dolor",

"mollit pariatur sit"

],

"activity": 30,

"preference": [

"voluptate ut non"

]

}

}

]

}

#### 7.2.3.2返回值

http状态码：200