OOP

* 封装
* 多态
* 继承

OOD：<https://www.jianshu.com/p/e378025920f8>

* 单一职责
* 开闭原则：抽象出相对稳定的接口，这部分应该不改动或者很少改动，封装变化
* 李氏替代：子类对象可以替换任何父类对象
* 接口隔离

Configuration：跨域，告诉浏览器安全

MVC

Controller

Impl： 从前端拿数据，传递给service；封装数据给前端

Interface：暴露接口，不提供具体实现

Service

Interface：

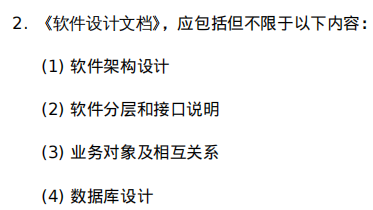
Impl：业务逻辑

AXIOS：异步请求：前端数据给后端

Dao：从数据库获取数据

Autowird: 自动装载

* MVC
* 接口提纯
* 数据库表



1. 架构：简单介绍（技术、图、MVC，前端？）

2. 软件分层：MVC，整体吹（文字具体介绍）

接口：（front/back）表格简单列举：参数、返回、功能

3. 业务对象：

3.1 entity（列表描述和数据库的对应关系）

3.2 逻辑处理（主要看service，任务的生命流程：特定功能实现）

3.3 相互关系：类图（service、dao、controller）

4. 数据库设计：数据结构、ER图

# 基本设计概念

## 技术总览

表1‑1 开发所用的主要技术总览

|  |  |
| --- | --- |
| **前端** | vue.js, elementui, animate.css, fontawesome, googlefonts, iconfont, quileditir, boxicons, axios, sessionStorage和localStorage router路由 |
| **后端** | springboot, springDataJpa, maven, junit |

## 软件功能架构总览

根据需求文档，我们将软件分为六个功能模块，分别围绕日志记录、系统消息、个人信息、团队信息、标签板块和任务招募展开。前五个模块包含的函数主要实现信息数据的增删改查，任务招募模块的函数功能则主要实现招募任务的生命流程。基于功能模块的划分，整个后端基于三层架构设计实现，每个模块拥有自己的DAO层、Service层和Controller层，结合实体层Entity和数据库完成数据的处理和传递。

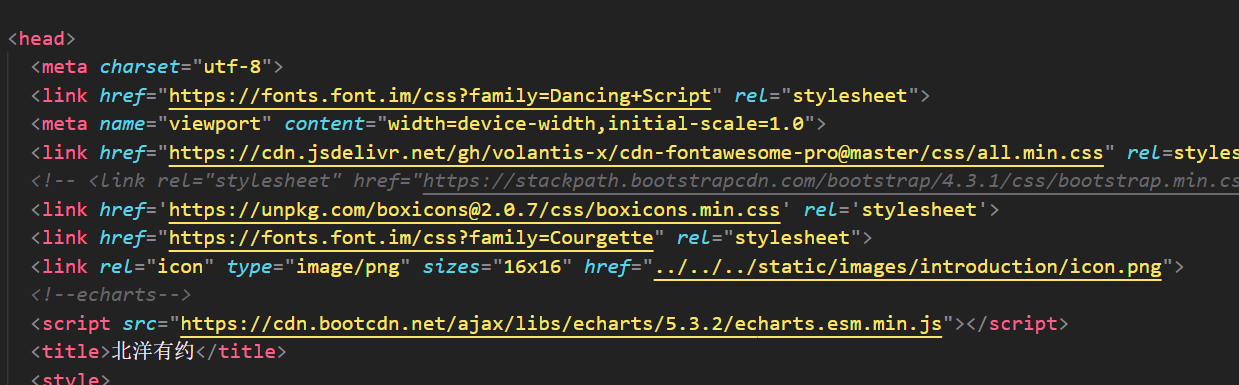
# 前端设计

## 项目依赖统一管理

公用组件库与elementui, animate.css, aioxs等技术统一在main.js中进行加载，对于googlefonts, fontawesome, echarts等在index.html中使用cdn链接加载减少最终打包的项目大小。



(a)



(b)

图 2‑1 前端组件和技术依赖导入

## 样式全局管理

为了方便修改全局的主题配色，在index.html中使用root设置了全局的主题配色。当需要修改全局主题配色时，只需要在这里进行修改即可完成全局的操作，增强了后期可维护性。

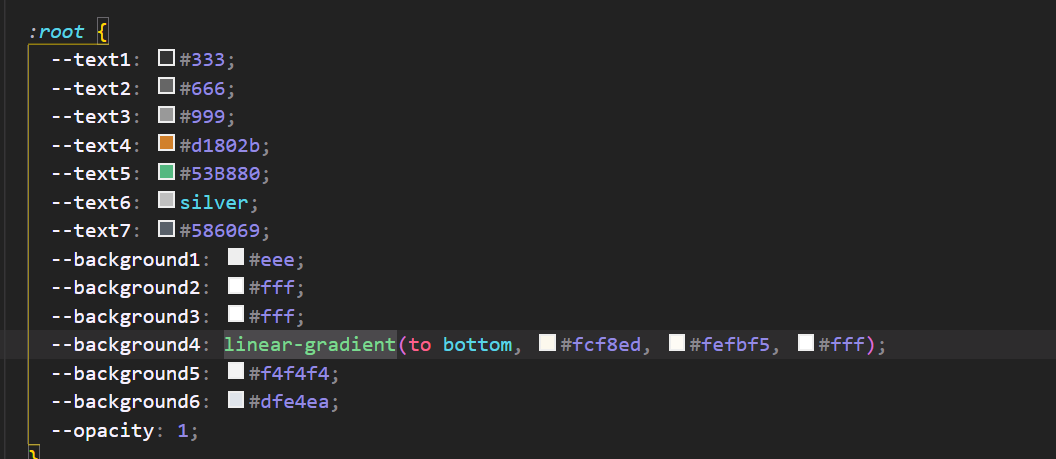


图2‑2 全局样式管理

每一个组件都有对应的局部样式文件，写在了各自vue文件中的style标签下，其中标签的scoped属性会保证其样式只在对应的组件局部生效，不会造成全局的样式污染。

全局样式的添加、组件库样式覆盖等类似需求，统一在src/styles下的index.scss中完成，scss优先级更高，可以覆盖默认样式。同时为了防止全局污染，每一个样式选择器都添加了对应页面的名称。

## 页面组件分模块管理

为了增强文件结构的可读性及功能模块的可扩展性和可维护性，每一个页面都独立地对应一个views页面下的文件夹，在每一个这样的文件夹中，index.vue是该模块页面的入口加载文件，其他的是该模块页面index.vue使用的私有组件。全局的共用组件封装在src/components下方。

考虑到接口可能在后期需要进行更改，并且伴随着业务的增加，后期的接口数量也会显著增加，我们按模块分装了每一个页面使用的接口调用方法，apis下的每一个文件夹都对应一个页面的所有接口请求函数。每一个页面使用的第三方静态资源如图标、图片素材等，也分模块存储到了static下，保证各文件的存储内容到开发后期也是易于辨识的，便于在不会影响其他模块正常工作的前提下对某个模块进行需改。

## Axios封装请求

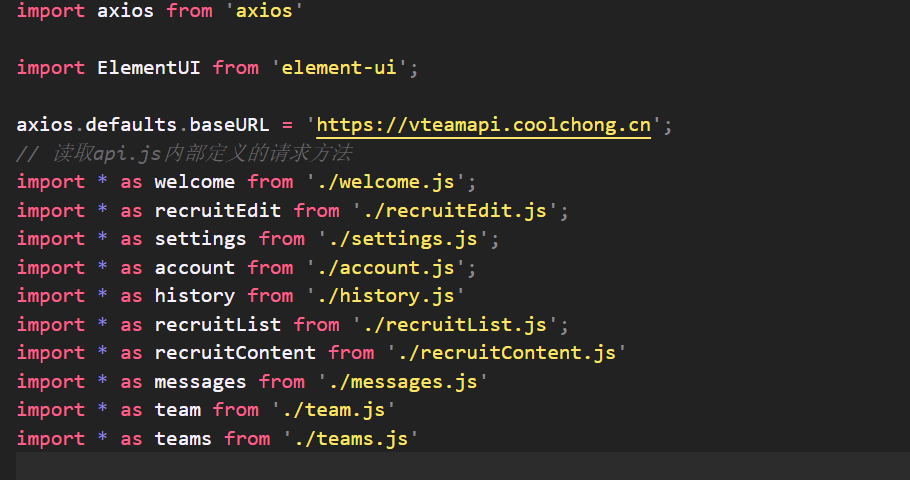


图2‑3

一般情况下，如果将请求函数的具体实现写入vue组件，后期维护接口时就不得不在众多组件的script js中查找修改。再加上每一个vue组件中的请求函数都会写出具体的url路径，如果有1000个请求接口函数，就会有1000个url和具体的请求函数实现方法随机分布在不同的vue组件中，造成后期维护成本指数型增长。因此为了提高后期的代码可扩展性，我们对axios请求函数进行统一的封装管理。

我们建立了index.js用于提取各模块所有的请求函数，封装成一个对外开放的对象apis。这个apis对象包含了整个项目所有请求函数的具体实现，并通过main.js加载到全局，当页面调用请求函数时，就无需再写具体的实现逻辑了，只需要调用apis已经提前封装好并对外开放的请求函数。

还需要在index.js中使用axios.baseUTL设置统一的接口路径，所有封装的请求函数都会默认加载这个路径。如果后台ip发生更改，前台开发人员也只需要对index.js中的baseURL进行一次修改，即可完成整个项目上百个请求函数路径的更改。

## 请求拦截器的使用

这里我们使用了请求拦截器以及响应拦截器，其中请求拦截器主要用于实现token的自动加载。由于我们对除了登录注册以外的所有接口都使用了token认证，因此前台需要在这些请求发送前，在请求头中加入token值，而这个token值是存储到了本地的sessionStorage中进行持久化存储。使用请求拦截器会在每一个请求发送之前首先检测本地是否存储有token值，如果检测到了会自动在请求头中加入token值，从而实现token的自动检验装载。



图2‑4 请求拦截器

响应拦截器主要用于报错。由于后台可能会出现请求处理的错误并返还500错误码，所以每一个请求函数都需要添加catch来进行错误码的前台报错。但是这使代码变得冗余，我们可以使用响应拦截器对请求响应进行拦截，然后检验返还的状态码。如果状态码为200，则正常请求并返还响应结果；如果返还的请求响应状态码不正常，响应拦截器就会自动catch这些响应报错，从而实现了代码的简洁性。



图2‑5 响应拦截器

# 后端架构设计

## 三层架构及接口设计原则

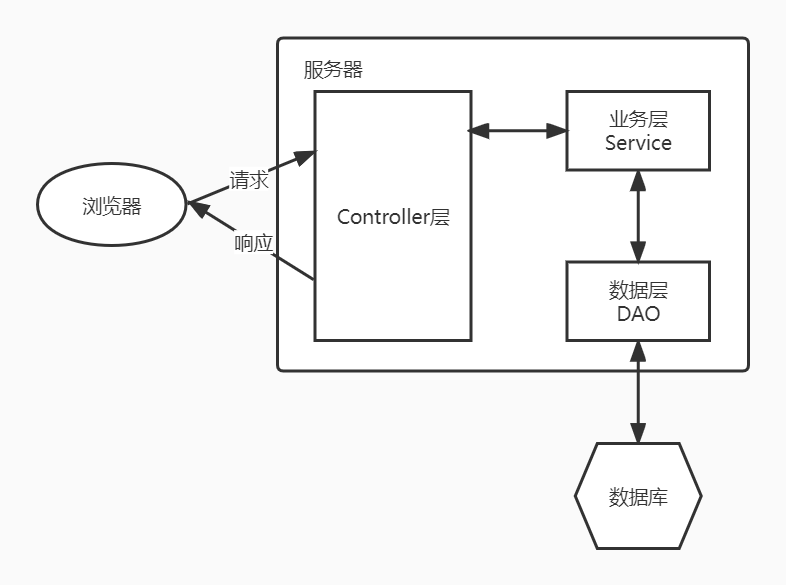


图3‑1 三层架构示意图

从逻辑上划分，基于Spring框架，我们将软件后端分为Repository层、DAO层、Entity层、Service层和Controller层。Controller层调用Service层的方法，Service层调用DAO层中的方法，其中调用的参数使用Entity层进行传递。

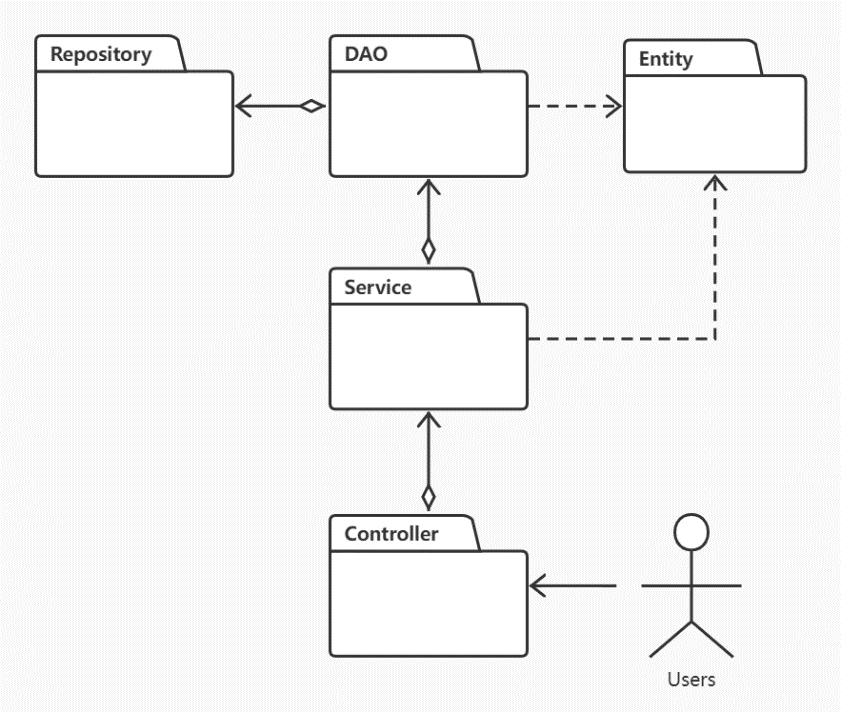


图3‑2 包图

Repository层直接与数据库对接，继承数据库的操作函数。

Entity层是实体层，也就是所谓的model，是数据库在项目中的类，该文件包含实体类的属性和对应属性的set、get方法。在本软件项目中，Entity层主要包括的数据结构有：日志记录、系统消息、招募中的任务信息、标签板块名称、用户个人展示信息、组队成功的团队信息和用户的系统信息。

DAO层会导入Repository类，对数据库实际使用到的方法进行封装以供Service层调用。比如在日志记录的功能模块中，DAO层在insertHistory()方法中调用Repository层的save()函数，对应数据库中的“增”。

Service层主要负责业务模块的逻辑应用设计。Service层会调用DAO层的接口，接收DAO层返回的数据，完成项目的基本功能设计。封装Service层的业务逻辑有利于业务逻辑的独立性和重复利用性。

Controller层负责具体的业务模块流程的控制，包括前后端交互，接受前端请求，调用Service层，接收service层返回的数据，最后返回具体的页面和数据到客户端。即，将前端的请求和数据类型进行汇总整理交给Service层进行业务处理，并将Service层处理的结果返回给前端。

例如在系统消息功能模块的工作流程中，用户进入主页，页面需要显示任务招募信息列表，调用Controller层的函数getScreenRecruitInfo()并发送http请求，Controller层调用Service层的getScreenRecruitInfo()函数并等待返回结果。Service层调用DAO层函数getValidRecruitInfo()，并注入Entity层的RecruitInfo类用于传递DAO层获得数据。DAO层将查询得到的原始数据存储在Entity层的RecruitInfo类中后，Service层根据需求调用RecruitInfo类的get方法，将获得的属性数据进行处理整合并统一格式，再返还给Controller层。Controller将Service层处理好的招募列表信息呈现给用户。

在实现三层架构的过程中，我们使用了“先设计放接口的类，再创建实现的类”的思维方式，通过对接口而不是对类编程达到代码重用的目的，尽可能地保证了接口的功能单一和高可复用性。三层架构的代码结构清晰，每层之间耦合度低，可维护性高，可扩展性高，利于开发任务同步进行，容易适应需求变化。

下面展示功能模块的部分结构类图：

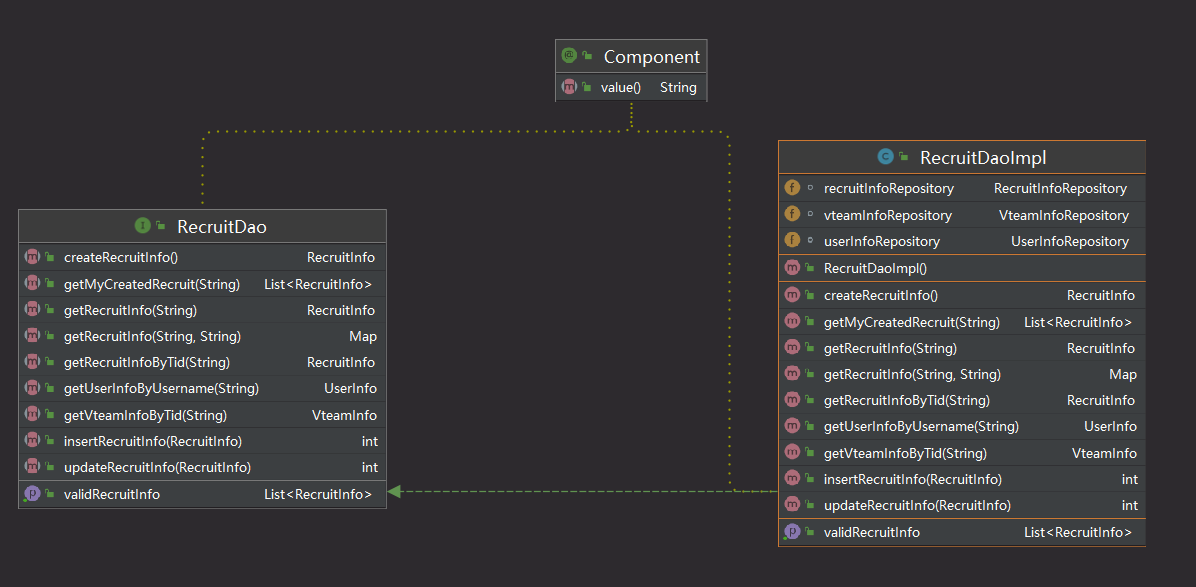


图3‑3 DAO层任务招募模块函数结构

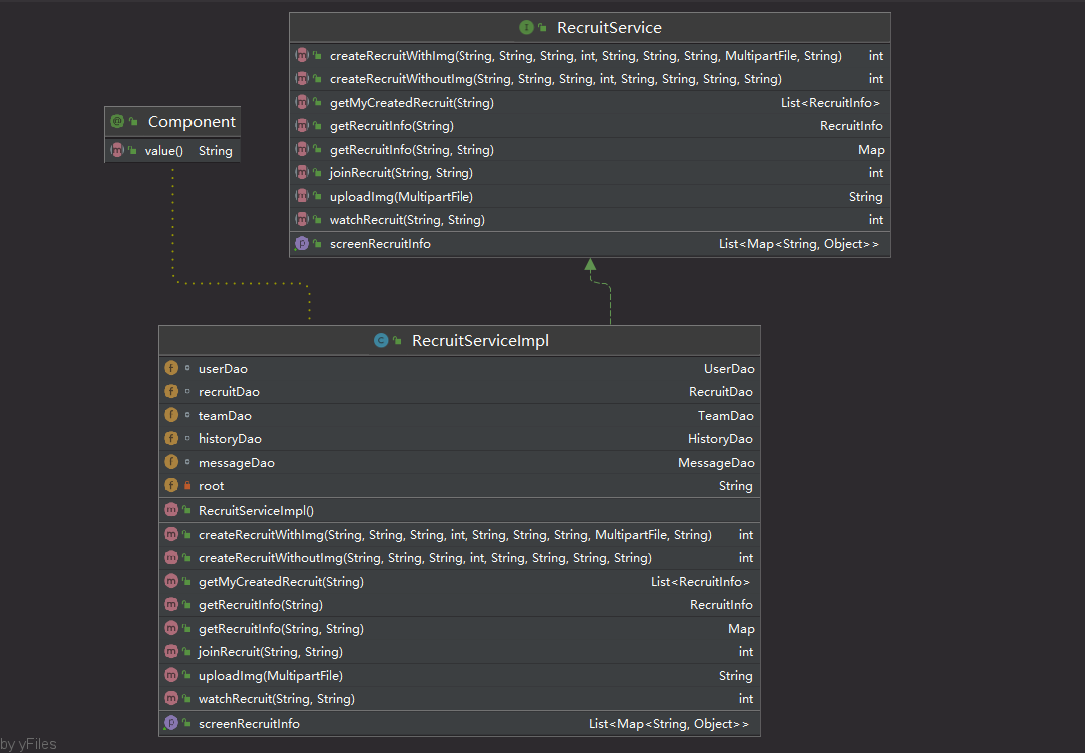


图3‑4 Service层任务招募模块函数结构

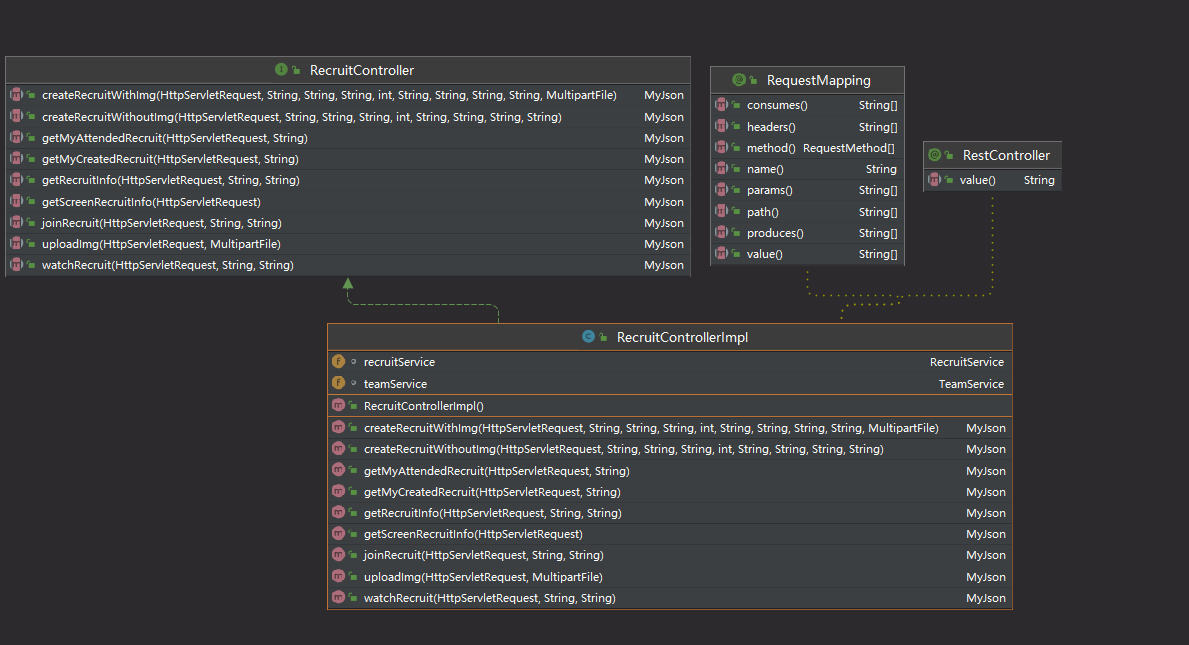


图3‑5 Controller层任务招募模块函数结构

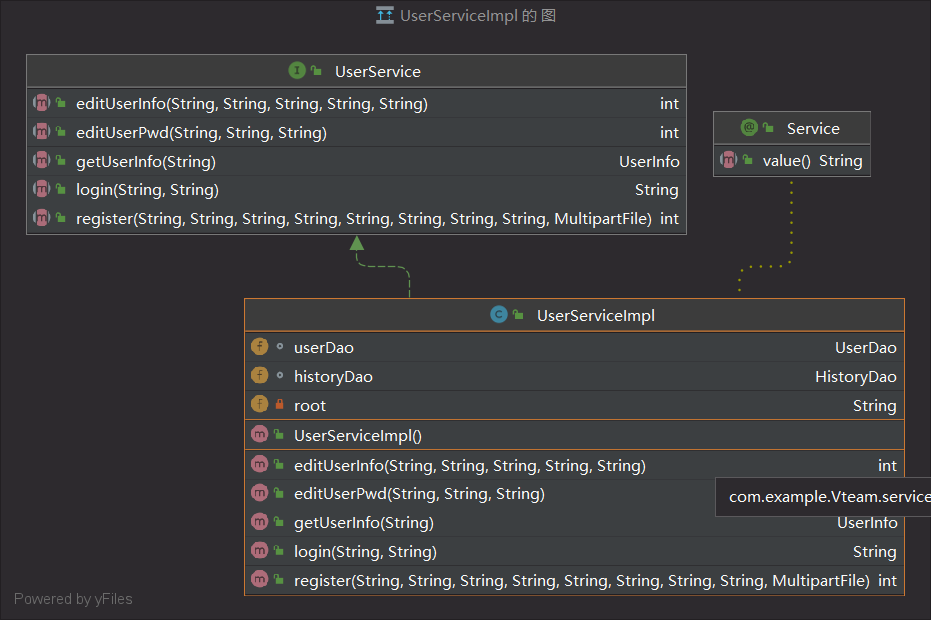


图3‑6 Service层用户信息模块函数结构

## 接口设计

### 日志记录功能模块

表3‑1 日志记录功能模块接口

|  |  |  |
| --- | --- | --- |
| **HistoryDAO** | **HistoryService** | **HistoryController** |
| List searchHistory (String username);  int insertHistory (History history); | List getAllHistory (String username); | MyJson getAllHistory  (HttpServletRequest request,  @RequestParam("username") String username); |

### 系统消息功能模块

表3‑2 系统消息功能模块接口

|  |  |  |
| --- | --- | --- |
| **MessageDAO** | **MessageService** | **MessageController** |
| List searchMessage (String username);  int setSingleMessageStatus (String mid);  int setAllMessageStatus (String username);  void insertMessage (Message message); | List getAllMessage (String username);  int setSingleMessageStatus (String mid);  int setAllMessageStatus (String username); | MyJson getAllMessage (HttpServletRequest request,  @RequestParam("username") String username);  MyJson setSingleMessageStatus (HttpServletRequest request,  @RequestParam("mid") String mid);  MyJson setAllMessageStatus (HttpServletRequest request,  @RequestParam("username") String username); |

### 任务招募功能模块

表3‑3任务招募功能模块接口

|  |  |  |
| --- | --- | --- |
| **RecruitDAO** | **RecruitService** | **RecruitController** |
| RecruitInfo createRecruitInfo ();  int insertRecruitInfo (RecruitInfo recruitInfo);  RecruitInfo getRecruitInfo (String rid);  int updateRecruitInfo (RecruitInfo recruitInfo);  List<RecruitInfo> getMyCreatedRecruit (String username);  RecruitInfo getRecruitInfoByTid (String tid);  Map getRecruitInfo (String username, String rid);  List<RecruitInfo> getValidRecruitInfo ();  UserInfo getUserInfoByUsername (String username);  VteamInfo getVteamInfoByTid (String tid); | int createRecruitWithImg (String username,  String rtitle,  String description,  int maxMates,  String endTime,  String rclass,  String content, MultipartFile img, String rtags);  int createRecruitWithoutImg (String username,  String rtitle,  String description,  int maxMates,  String endTime,  String rclass,  String content,  String rtags);  int joinRecruit  (String rid,  String username);  List<RecruitInfo> getMyCreatedRecruit (String username);  RecruitInfo getRecruitInfo  (String tid);  Map getRecruitInfo (String username,  String rid);  List<Map<String,Object>> getScreenRecruitInfo ();  String uploadImg (MultipartFile img);  int watchRecruit  (String username,  String rid); | MyJson createRecruitWithImg (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("rtitle")  String rtitle,  @RequestParam("description") String description,  @RequestParam("maxMates")  int maxMates,  @RequestParam("endTime")  String endTime,  @RequestParam("rclass")  String rclass,  @RequestParam("content")  String content,  @RequestParam("rtags")  String rtags,  @RequestParam("img") MultipartFile img  );  MyJson createRecruitWithoutImg (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("rtitle")  String rtitle,  @RequestParam("description") String description,  @RequestParam("maxMates")  int maxMates,  @RequestParam("endTime") String endTime,  @RequestParam("rclass")  String rclass,  @RequestParam("content") String content,  @RequestParam("rtags") String rtags  );  MyJson joinRecruit (HttpServletRequest request,  @RequestParam("rid")  String rid,  @RequestParam("username") String username);  MyJson getMyCreatedRecruit (HttpServletRequest request, @RequestParam("username") String username);  MyJson getMyAttendedRecruit (HttpServletRequest request,  @RequestParam("username") String username);  MyJson uploadImg (HttpServletRequest request, @RequestParam("img") MultipartFile img);  MyJson getRecruitInfo (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("rid")  String rid);  MyJson getScreenRecruitInfo (HttpServletRequest request);  MyJson watchRecruit (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("rid") String rid); |

### 队伍信息功能模块

表3‑4队伍信息功能模块接口

|  |  |  |
| --- | --- | --- |
| **TeamDAO** | **TeamService** | **Controller** |
| VteamInfo createTeamInfo();  int insertTeamInfo (VteamInfo vteamInfo);  VteamInfo getVteamInfo (String tid);  int updateVteamInfo (VteamInfo vteaminfo);  List<VteamInfo> getMyAttendedTeam (String username);  List<UserInfo> getTeamInfo  (String tid);  UserInfo getUserInfoByUsername (String username);  List<VteamInfo> getUserVteamInfo (String username);  VteamUser getVteamUserByUsername (String username);  List<RecruitInfo> getRecruitInfoByTid (String tid); | List<VteamInfo> getMyAttendedTeam (String username);  List<UserInfo> getTeamInfo  (String tid);  List<Map<String,Object>> getUserTeamInfo  (String username); | MyJson getTeamInfo (HttpServletRequest request,  String tid);  MyJson getUserTeamInfo (HttpServletRequest request,  @RequestParam("username") String username); |

### 用户信息功能模块

表3‑5 用户信息功能模块接口

|  |  |  |
| --- | --- | --- |
| **UserDAO** | **UserService** | **UserController** |
| int insertVteamUser (VteamUser vteamUser);  int insertUserInfo (UserInfo userInfo);  VteamUser getVteamUser (String username);  UserInfo getUserInfo (String username);  int editUserInfo (String username, String name,  String sex,  String email,  String school);  int editUserPwd (String username, String oldPwd,  String newPwd); | int register  (String username, String password, String name,  String email,  String phoneNumber, String school,  String grade,  String sex, MultipartFile avatar);  String login  (String username, String password);  UserInfo getUserInfo (String username);  int editUserInfo (String username, String name,  String sex,  String email,  String school);  int editUserPwd (String username, String oldPwd,  String newPwd); | MyJson register (@RequestParam("username") String username,  @RequestParam("password") String password,  @RequestParam("name")  String name,  @RequestParam("email")  String email,  @RequestParam("phoneNumber") String phoneNumber,  @RequestParam("school")  String school,  @RequestParam("grade")  String grade,  @RequestParam("sex")  String sex,  @RequestParam("avatar") MultipartFile avatar);  MyJson editUserInfo (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("name")  String name,  @RequestParam("sex")  String sex,  @RequestParam("email")  String email,  @RequestParam("school")  String school);  MyJson editUserPwd (HttpServletRequest request,  @RequestParam("username") String username,  @RequestParam("oldPwd")  String oldPwd,  @RequestParam("newPwd")  String newPwd);  MyJson login (@RequestParam("username") String username,  @RequestParam("password") String password);  MyJson getUserInfo (HttpServletRequest request,  @RequestParam("username") String username); |

### 标签板块功能模块

表3‑6 标签板块功能模块接口

|  |  |  |
| --- | --- | --- |
| **SettingsDAO** | **SettingsService** | **SettingsController** |
| List<Settings> getAllSettings (); | List<Settings> getSettings (); | MyJson getSettings (HttpServletRequest request); |

## Controller层接口说明

### 日志记录功能模块

表3‑7 HistoryController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| getAllHistory | HttpServletRequest request  (定义的请求体)  String username  (用户id) | MyJson  规定的返回格式，其中的result带有用户的所有组队记录 | 获取指定id的组队记录 |

### 系统消息功能模块

表3‑8 MessageController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| getAllHistory | HttpServletRequest request  (定义的请求体)  String username  (用户id) | MyJson  规定的返回格式，其中的result带有用户的所有组队记录 | 获取指定id的组队记录 |
| setSingleMessageStatus | HttpServletRequest request  (定义的请求体)  String mid  (消息id) | MyJson  规定的返回格式，其中的result带有该消息的状态信息 | 设定指定消息id对应的消息状态(未读/已读) |
| setAllMessageStatus | HttpServletRequest request  (定义的请求体)  String username  (用户id) | MyJson  规定的返回格式，其中的result带有用户所有消息的状态信息 | 设定所有指定id用户的所有消息状态(未读/已读) |

### 任务招募功能模块

表3‑9 RecruitController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| createRecruit | HttpServletRequest request  (定义的请求体)  @RequestParam("username") String username,  (用户id)  @RequestParam("rtitle") String rtitle,  (招募标题)  @RequestParam("description") String description,  (招募简介)  @RequestParam("maxMates") int maxMates,  (最大招募人数)  @RequestParam("endTime") String endTime,  (截止日期)  @RequestParam("rclass") String rclass,  (招募类别)  @RequestParam("content") String content,  (招募详细内容)  @RequestParam("img") MultipartFile img,  (招募介绍图片)  @RequestBody String[] rtags  (招募标签) | MyJson  规定的返回格式，其中的result带有是否创建成功的信息 | 创建招募 |
| joinRecruit | HttpServletRequest request,  (定义的请求体)  @RequestParam("rid") String rid,  (招募id)  @RequestParam("username") String username  (用户id) | MyJson  规定的返回格式，其中的result带有是否加入成功的信息 | 某id用户加入到某id招募中 |
| getMyCreatedRecruit | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username  (用户id) | MyJson  规定的返回格式，其中的result带有指定用户创建的所有招募 | 获取指定用户创建的招募信息 |
| getMyAttendedRecruit | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username  (用户id) | MyJson  规定的返回格式，其中的result带有指定用户参与的所有招募 | 获取指定用户参与的招募信息 |
| uploadImg | HttpServletRequest request,  (定义的请求体)  @RequestParam("img") MultipartFile img  (招募简介的图片) | MyJson  规定的返回格式，其中的result带有指定路径的招募图片 | 获取指定路径的招募图片 |
| getRecruitInfo | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username,  (用户id)  @RequestParam("rid")  String rid  (任务id) | MyJson  规定的返回格式，其中的result带用户参与的某招募的信息 | 获取某招募的信息 |
| getScreenRecruitInfo | HttpServletRequest request,  (定义的请求体) | MyJson  规定的返回格式，其中的result带首页所有招募的信息 | 获取首页招募信息列表 |
| watchRecruit | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username,  (用户id)  @RequestParam("rid")  String rid  (任务id) | MyJson  规定的返回格式，其中的result带某一条招募的信息 | 查看某一条招募信息 |

### 队伍信息功能模块

表3‑10 TeamController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| getTeamInfo | HttpServletRequest request,  (定义的请求体)  String tid  (某队伍的id) | MyJson  规定的返回格式，其中的result带某队伍的信息 | 获取某队伍的信息 |
| getUserTeamInfo | HttpServletRequest request,  (定义的请求体)  String username  (某用户id) | MyJson  规定的返回格式，其中的result带某用户的队伍信息 | 获取用户组队信息 |

### 用户信息功能模块

表3‑11 UserController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| register | @RequestParam("username") String username,  (用户id)  @RequestParam("password") String password,  (用户密码)  @RequestParam("name") String name,  (用户名)  @RequestParam("email") String email,  (用户邮箱)  @RequestParam("phoneNumber") String phoneNumber,  (用户电话)  @RequestParam("school") String school,  (用户学校)  @RequestParam("grade") String grade,  (用户年级)  @RequestParam("sex") String sex,  (用户性别)  @RequestParam("avatar") MultipartFile avatar  (用户头像的图片路径) | MyJson  规定的返回格式，其中的result带是否注册成功的信息 | 注册用户 |
| editUserInfo | HttpServletRequest request,  @RequestParam("username") String username,  (用户id)  @RequestParam("name") String name,  (用户名)  @RequestParam("sex") String sex,  用户性别)  @RequestParam("email") String email,  (用户邮箱)  @RequestParam("school") String school)  (用户学校) | MyJson  规定的返回格式，其中的result带是否编辑成功的信息 | 编辑用户信息 |
| editUserPwd | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username,  (用户id)  @RequestParam("oldPwd") String oldPwd,  (旧密码)  @RequestParam("newPwd") String newPwd  (新密码) | MyJson  规定的返回格式，其中的result带是否修改密码成功的信息 | 修改密码 |
| login | @RequestParam("username") String username,  (用户id)  @RequestParam("password") String password  (用户密码) | MyJson  规定的返回格式，其中的result带是否登录成功的信息 | 用户登录 |
| getUserInfo | HttpServletRequest request,  (定义的请求体)  @RequestParam("username") String username  (用户id) | MyJson  规定的返回格式，其中的result带用户基本信息 | 获取用户信息 |

### 标签板块功能模块

表3‑12 SettingsController接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | **参数** | **返回值** | **功能** |
| getSettings | HttpServletRequest request,  (定义的请求体) | MyJson  规定的返回格式，其中的result带主页标签板块信息 | 获取首页供筛选的标签与板块信息 |

# 业务对象和数据库

## Entity实体层和数据库的对应

本软件的业务对象根据每个功能模块进行设计，每个Entity类的数据对应了数据库表中的属性，并设立set和get方法进行数据传递和缓存。

### 日志记录Entity

表4‑1日志记录数据结构

|  |  |
| --- | --- |
| **History** | **history** |
| @Id  @Column(name = "hid")  private String hid;  @Column(name = "username")  private String username;  @Column(name = "htitle")  private String htitle;  @Column(name = "history")  private String history;  @Column(name = "htime")  private String htime; | create table history  (  hid varchar(70) primary key,  username varchar(20),  htitle varchar(70),  history varchar(200),  htime datetime  ) |

### 系统消息Entity

表4‑2 系统消息数据结构

|  |  |
| --- | --- |
| **Message** | **message** |
| @Id  @Column(name = "mid")  private String mid;  @Column(name = "status")  private int status;  @Column(name = "username")  private String username;  @Column(name = "mtitle")  private String mtitle;  @Column(name = "message")  private String message;  @Column(name = "mtime")  private String mtime; | create table message  (  mid varchar(70) primary key,  status integer,  username varchar(20),  mtitle varchar(70),  message varchar(200),  mtime datetime  ); |

### 任务招募Entity

表4‑3 任务招募数据结构

|  |  |
| --- | --- |
| **RecruitInfo** | **recruit\_info** |
| @Id  @Column(name = "rid")  private String rid;  @Column(name = "rtitle")  private String rtitle;  @Column(name = "description")  private String description;  @Column(name = "subscribe")  private int subscribe;  @Column(name = "rclass")  private String rclass;  @Column(name = "rtags")  private String rtags;  @Column(name = "content")  private String content;  @Column(name = "creator")  private String creator;  @Column(name = "img")  private String img;  @Column(name = "startTime")  private String startTime;  @Column(name = "endTime")  private String endTime;  @Column(name = "tid")  private String tid;  @Column(name = "byTeacher")  private int byTeacher;  @Column(name = "isDestroy")  private int isDestroy; | create table recruit\_info  (  rid varchar(70) primary key,  rtitle varchar(70),  description varchar(500),  subscribe integer,  rclass varchar(20),  rtags varchar(200),  content varchar(700),  creator varchar(20),  img varchar(70),  startTime datetime,  endTime datetime,  tid varchar(70),  byTeacher integer,  isDestroy integer  ); |

### 队伍信息Entity

表4‑4 队伍信息数据结构

|  |  |
| --- | --- |
| **VteamInfo** | **vteam\_info** |
| @Id  @Column(name = "tid")  private String tid;  @Column(name = "maxMates")  private int maxMates;  @Column(name = "currentMates")  private int currentMates;  @Column(name = "teamMates")  private String teamMates;  @Column(name = "successTime")  private String successTime;  @Column(name = "isSuccess")  private int isSuccess; | create table vteam\_info  (  tid varchar(70) primary key,  maxMates integer,  currentMates integer,  teamMates varchar(500),  successTime datetime,  isSuccess integer  ); |

### 成员信息Entity

表4‑5 成员信息数据结构

|  |  |
| --- | --- |
| **VteamUser** | **vteam\_user** |
| @Id  @Column(name="username")  private String username;  @Column(name="password")  private String password;  @Column(name="status")  private String status; | create table vteam\_user  (  # 账号  username varchar(20) primary key,  # 密码  password varchar(20),  # 权限  status varchar(10)  ); |

### 用户信息Entity

表 4‑6用户信息数据结构

|  |  |
| --- | --- |
| **UserInfo** | **user\_info** |
| @Id  @Column(name = "username")  private String username;  @Column(name = "email")  private String email;  @Column(name = "name")  private String name;  @Column(name = "phoneNumber")  private String phoneNumber;  @Column(name = "school")  private String school;  @Column(name = "grade")  private String grade;  @Column(name = "sex")  private String sex;  @Column(name = "avatarPath")  private String avatarPath;  @Column(name = "registerTime")  private String registerTime; | create table user\_info  (  # 账号  username varchar(20) primary key,  email varchar(70),  name varchar(20),  phoneNumber varchar(11),  school varchar(20),  grade varchar(20),  sex varchar(10),  avatarPath varchar(270),  registerTime datetime  ); |

### 标签板块Entity

表4‑7 标签板块数据结构

|  |  |
| --- | --- |
| **Settings** | **settings** |
| @EmbeddedId  SettingsKey id; | create table settings  (  primary key (class, tags),  class varchar(20),  tags varchar(20)  ); |

## 相互关系

### ER图

本项目的数据库ER图如下：

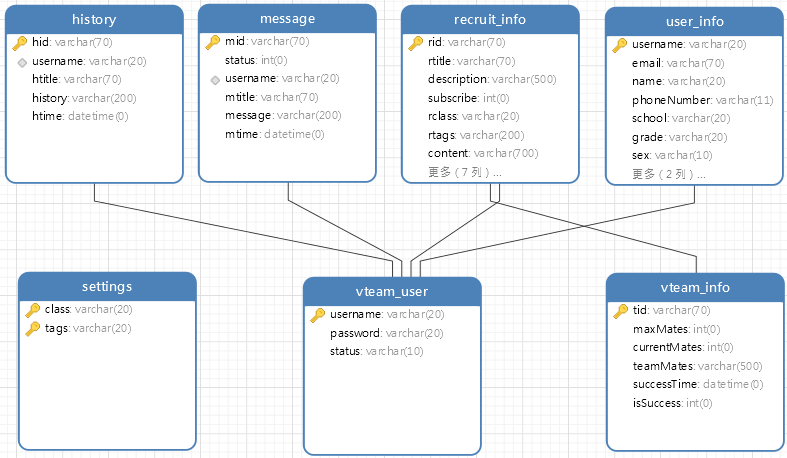


图4‑1 数据库表之间的联系

### DAO层

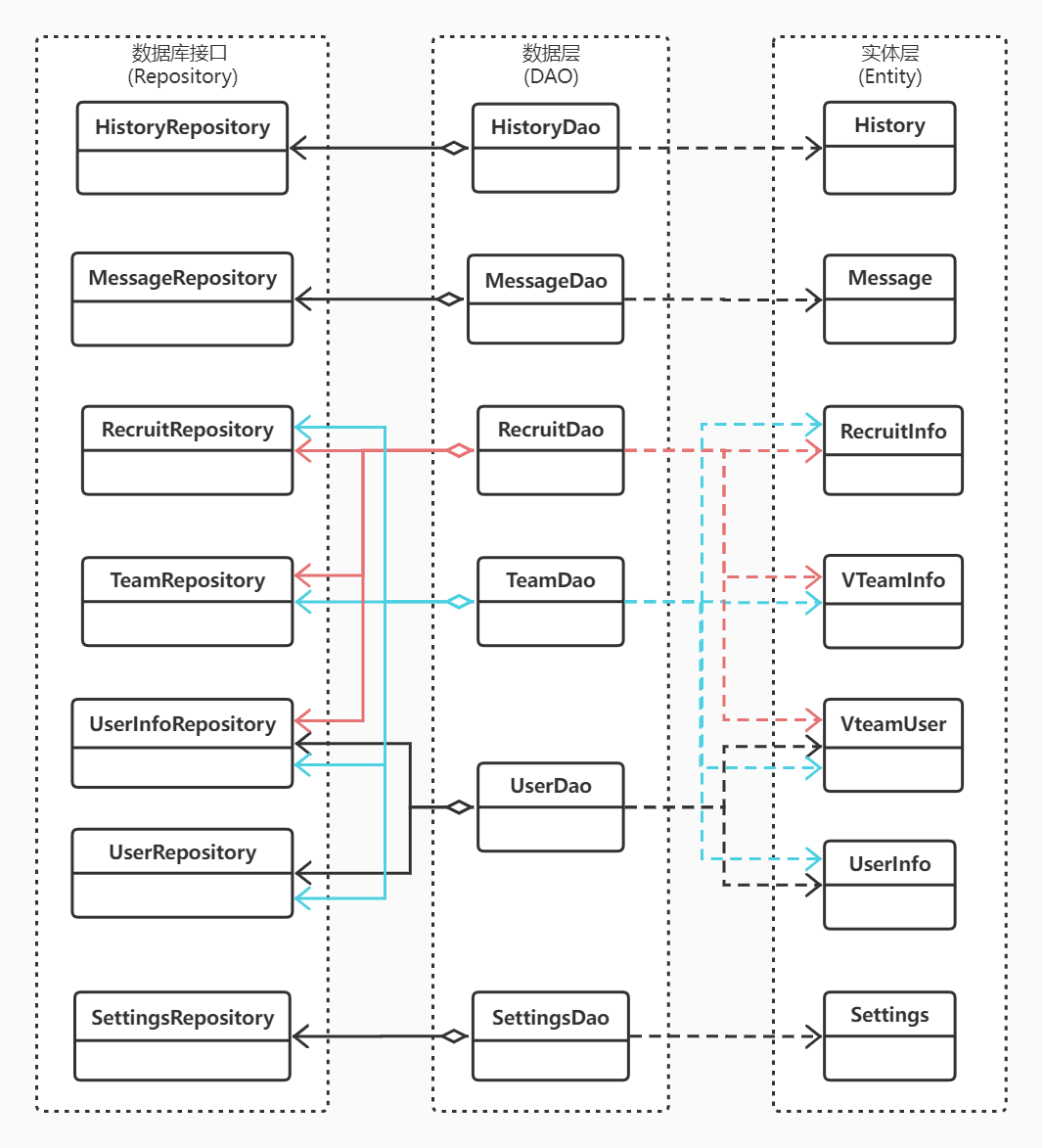


图4‑2 类图1: DAO Repository Entity

### Service层

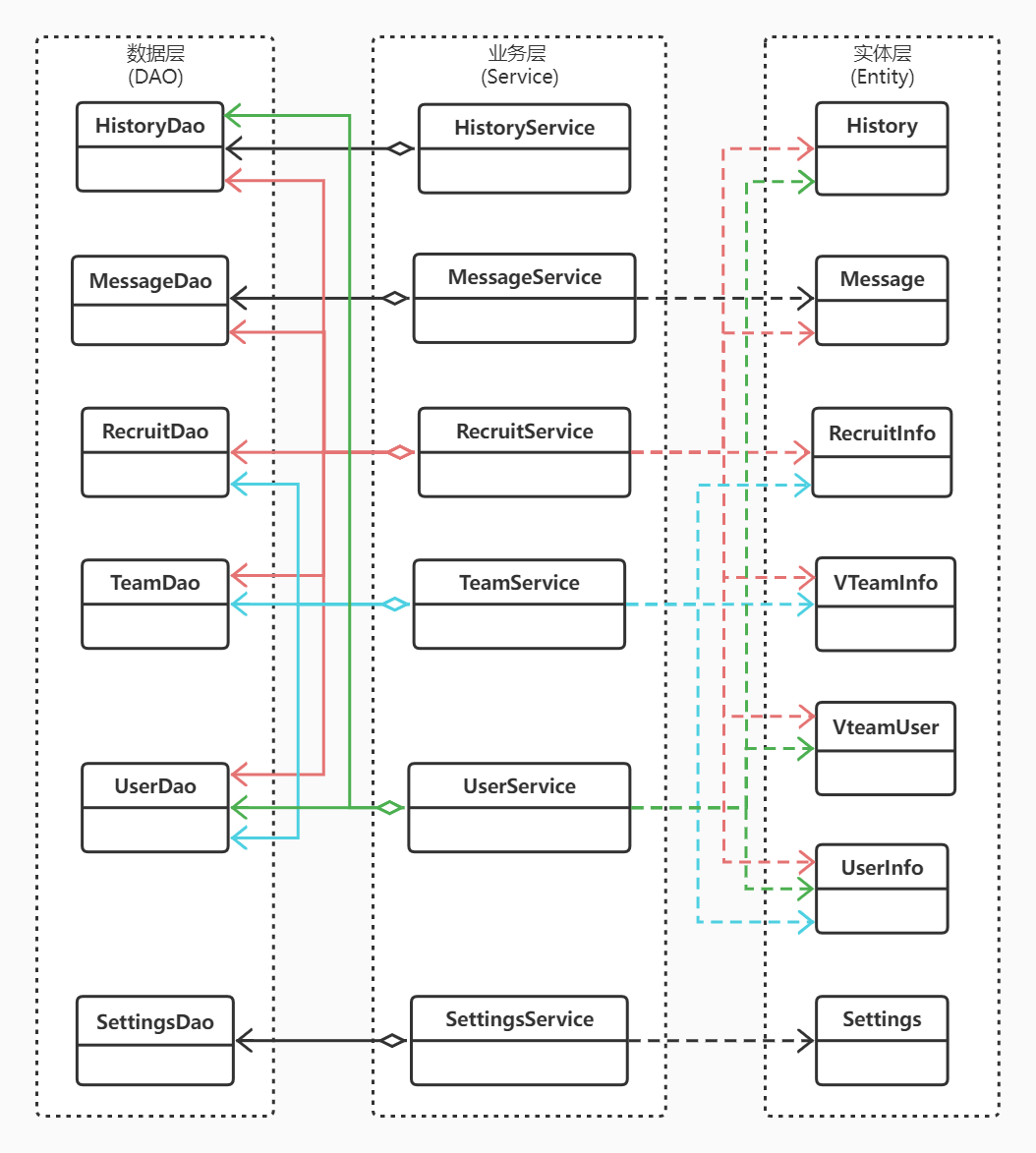


图 4‑3 类图2: DAO Service Entity

### Controller层

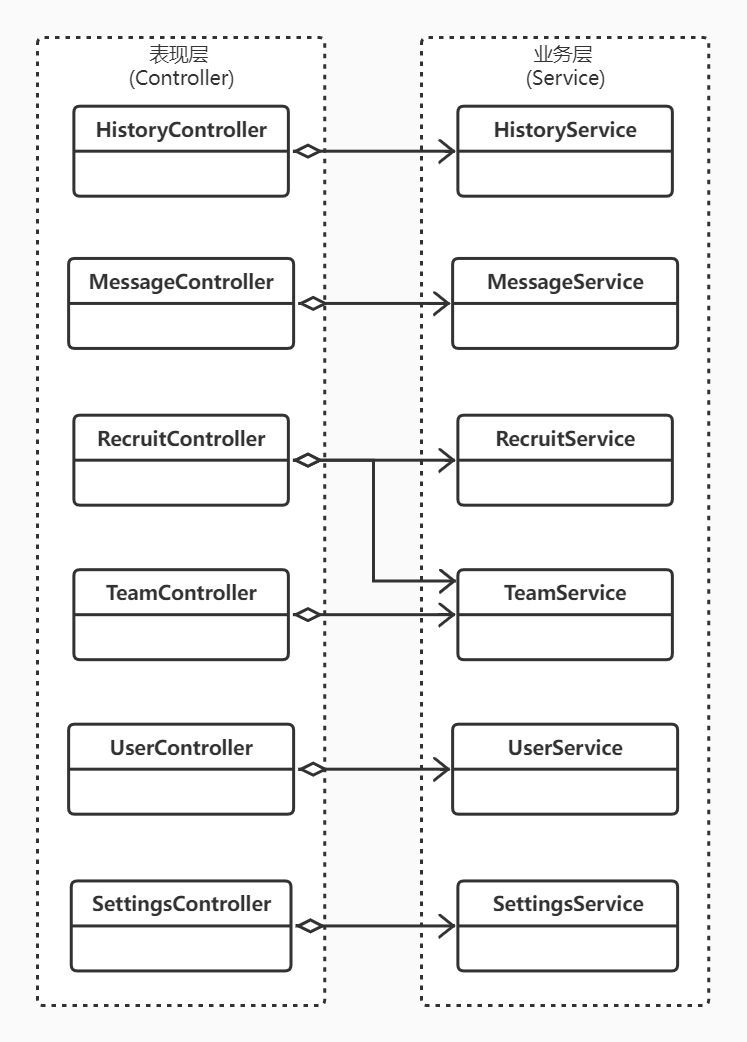


图4‑4 类图3: Controller Service