设计文档

### 概述

本文档确定整个软件的设计，便于开发人员进行核实，减少开发中的错误、包括项目的软件架构，使用的框架；还有原型设计，确定系统的可视化界面布局和功能；数据库的设计；前端的组成和实现；后端的设计和实现。

### 项目进度安排

|  |  |  |  |
| --- | --- | --- | --- |
| 任务名称 | 工期 | 开始时间（2022年） | 完成时间（2022年） |
| **分享交流社区** | **70天** | **4.18** | **6.26** |
| **1、启动** | **2天** | **4.18** | **4.19** |
| 小组召开启动会议 | 2天 | 4.18 | 4.19 |
| **2、计划** | **3天** | **4.20** | **4.22** |
| 制定计划 | 3天 | 4.20 | 4.22 |
| **3、执行** | **65天** | **4.23** | **6.26** |
| 3.1需求分析 | 2天 | 4.23 | 4.24 |
| 3.2 原型设计 | 3天 | 4.25 | 4.27 |
| 3.3概要设计 | 5天 | 4.28 | 5.2 |
| 3.4编码阶段 | 50天 | 5.3 | 6.21 |
| 前端 | 50天 | 5.3 | 6.21 |
| 后端 | 50天 | 5.3 | 6.21 |
| 3.5 测试 | 5天 | 6.22 | 6.26 |

### 项目架构

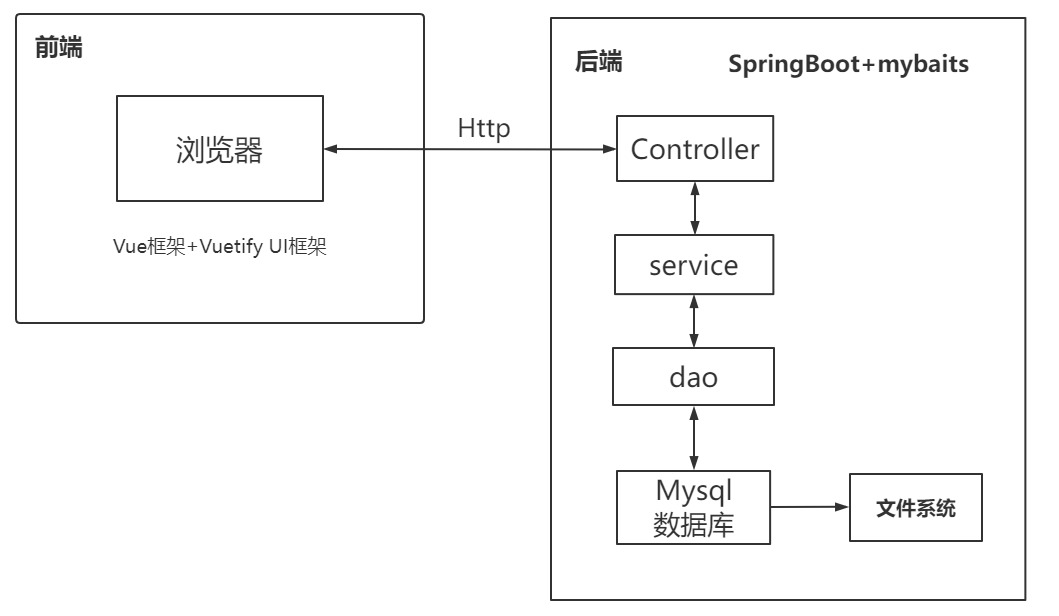


图3.1 项目架构图

上图是系统的软件架构，采用前后端分离的方式进行开发，前后端通讯通过http进行。前端使用vue框架，ui组件使用vuetify，网络请求使用axios进行http请求。后端使用SpringBoot框架进行开发，mybaits去进行数据库连接，数据库使用MySQL，图片存储在文件系统中。

### 原型设计

根据需求和功能，对项目进行一个原型设计，确定页面的布局和逻辑。

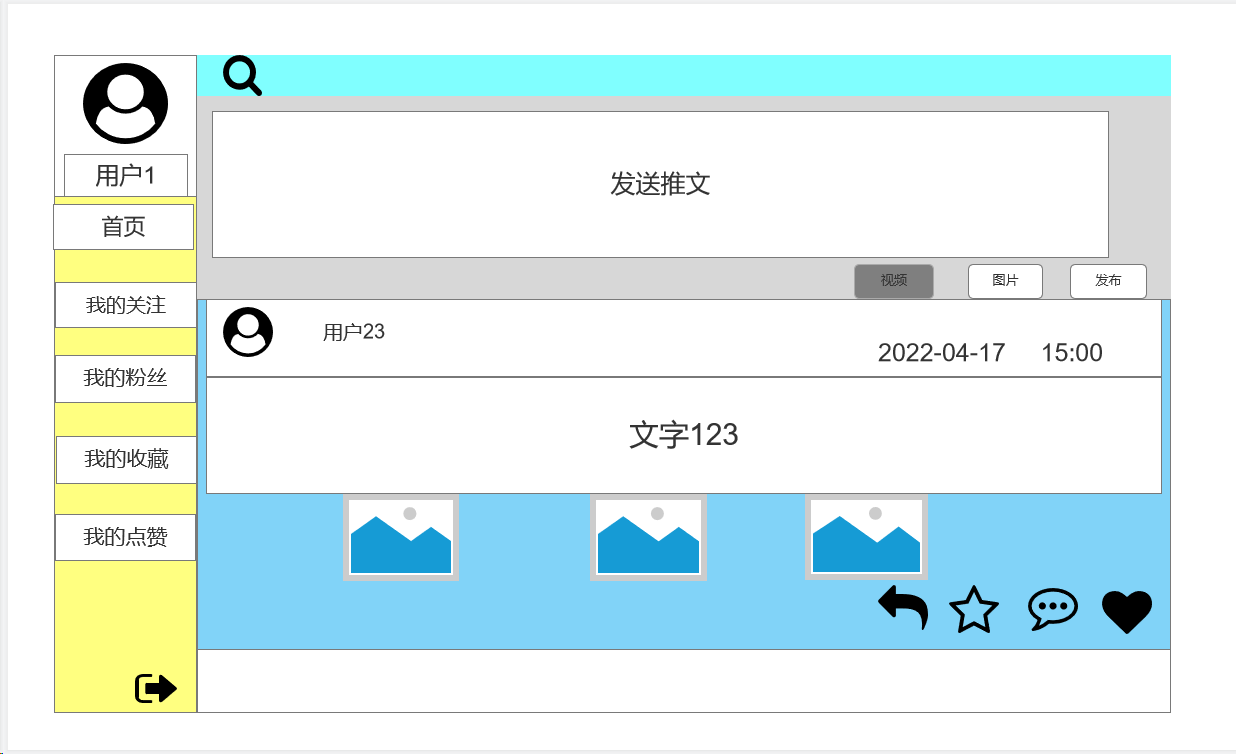


图4.1 原型设计图



图4.2 关注/粉丝界面图



图4.3 个人信息界面图

### 数据库设计

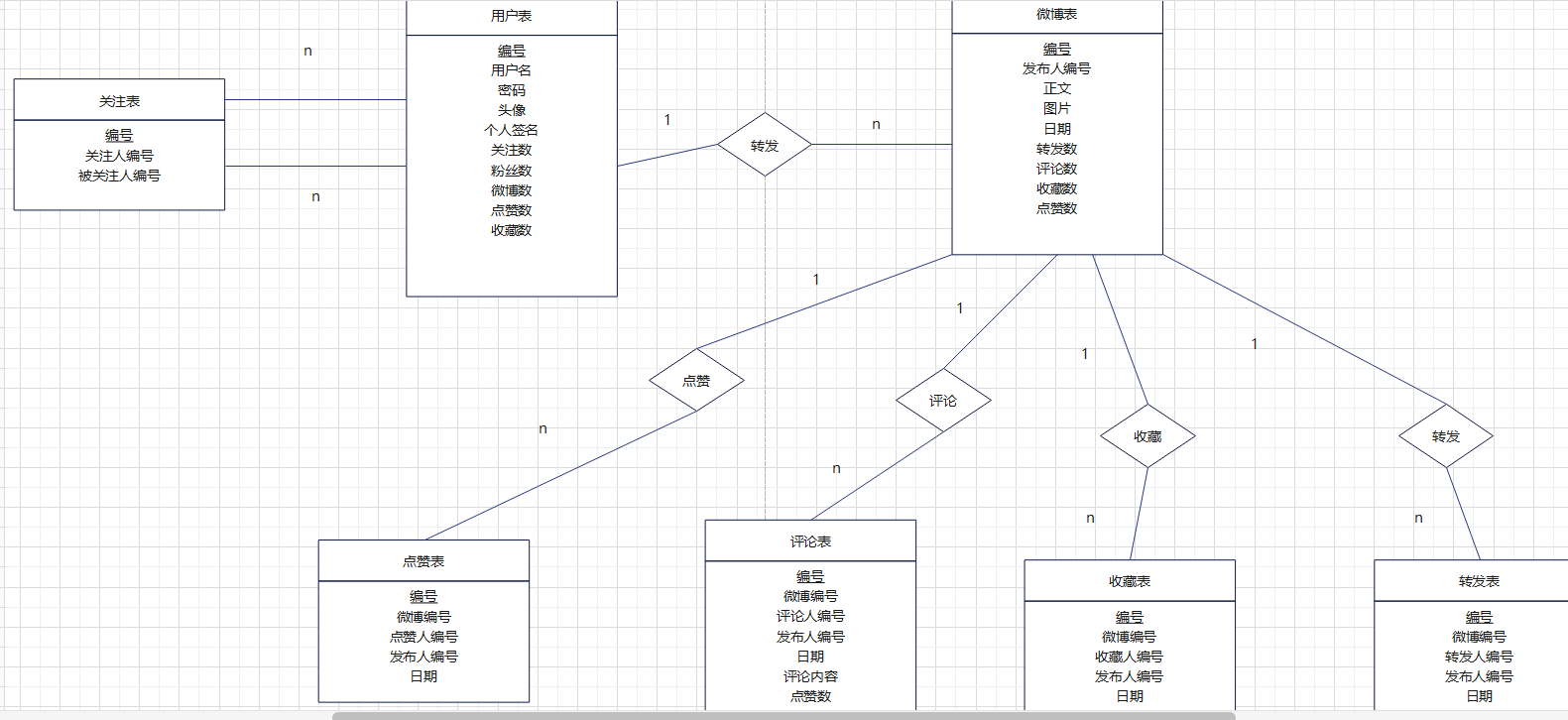


图5.1 最初的数据库设计图

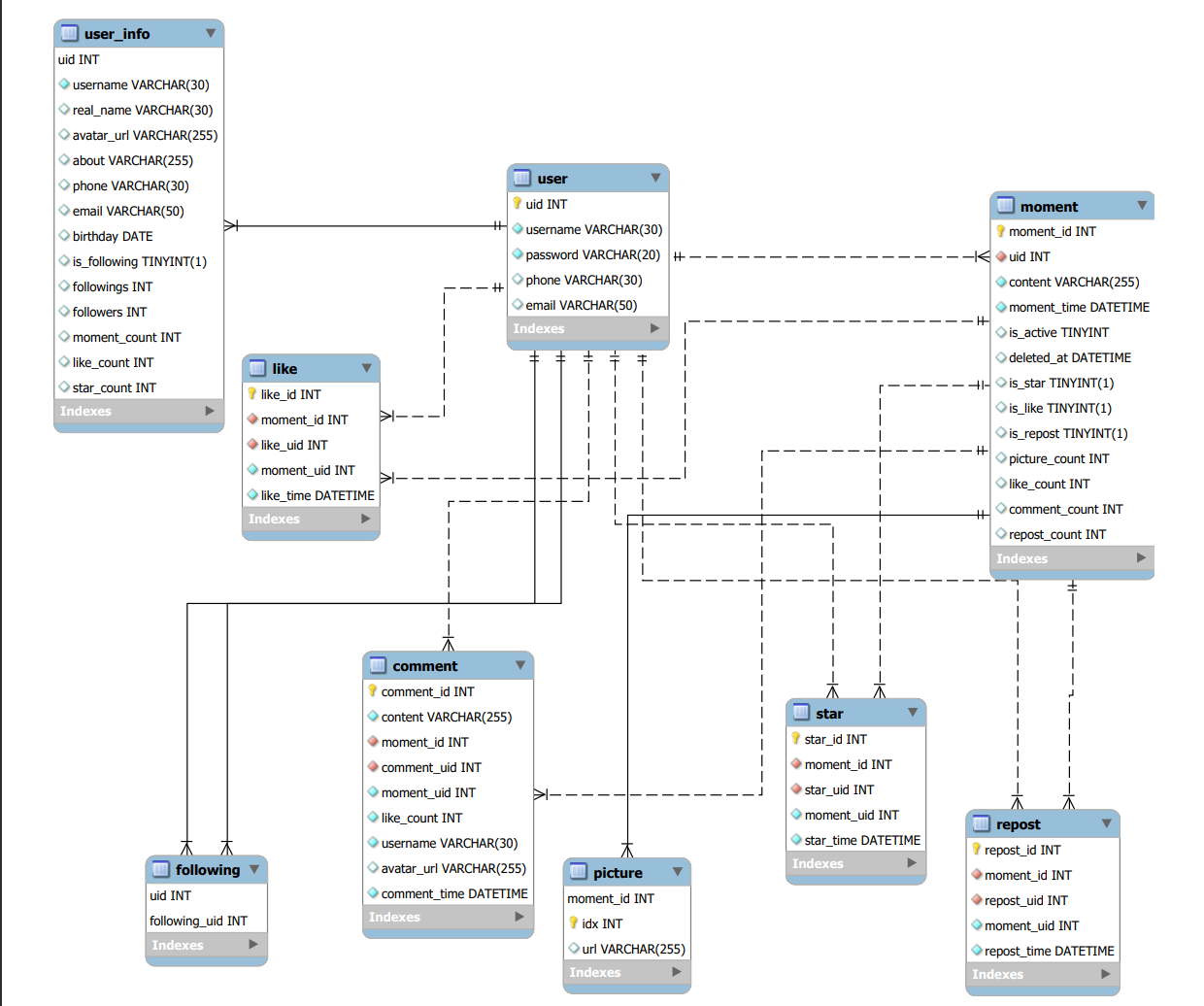


图5.2 最终数据库实体关系图

1. user表：只存放用户登录可能会用到的数据，表的所有字段如下：

uid: 用户ID，主键

username: 用户名

phone: 手机号

email: 邮箱

password: 密码

1. user\_info表：用户详细信息表，表的uid字段关联了user表的uid字段作为外键，所有字段如下：

uid: 用户ID，主键

username: 用户名

real\_name: 用户真实姓名

avatar\_url: 用户头像的URL地址

about: 用户简介

phone: 手机号码

email: 邮箱地址

birthday: 用户生日

followings: 关注的用户数量

followers: 粉丝数量

moment\_count: 动态数量

like\_count: 被点赞数量

star\_count: 收藏动态数量

1. moment表：动态的信息，表的uid字段关联了user表的uid字段作为外键，所有字段如下：

moment\_id: 动态ID，唯一标识一条动态，主键

uid: 用户ID，发布该动态的用户的ID

content: 内容，这条动态的文字内容

moment\_time: 发布这条动态的时间

is\_active: 这条动态是否可见，1表示可见，0表示不可见

deleted\_at: 如果该动态没有被删除则为空，否则为删除该动态的时间

picture\_count: 这条动态的图片数量

like\_count: 这条动态的点赞数量

comment\_count: 这条动态的评论数量

repost\_count: 这条动态的转发数量

1. picture表: 图片表， picture表以moment\_id和idx作为主键，moment\_id指向moment表的moment\_id作为外键，所有字段如下：

moment\_id: 动态ID

idx: 图片索引，指的是该图片是对应动态中的第idx张

url: 该图片对外访问的URL地址

1. comment表：评论的信息表，所有字段如下：

comment\_id: 评论ID，唯一标识一条评论，主键

content: 评论的文字内容

moment\_id: 被评论的动态ID

comment\_uid: 执行评论的用户ID

moment\_uid: 被评论动态的用户ID

like\_count: 该评论被点赞的数量

username: 执行评论的用户名

avatar\_url: 执行评论的用户头像URL地址

comment\_time: 执行评论操作的时间

1. like表：点赞的信息表，所有字段如下：

like\_id: 点赞ID，唯一标识一条点赞行为，主键

moment\_id: 点赞的动态ID

like\_uid: 执行点赞的用户ID

moment\_uid: 被点赞动态的发布用户的ID

like\_time: 执行点赞动作的时间

1. star表：收藏的信息表，所有字段如下：

star\_id: 收藏ID，唯一标识一条收藏行为，主键

moment\_id: 收藏的动态ID

star\_uid: 执行收藏的用户ID

moment\_uid: 被收藏动态的发布用户的ID

star\_time: 执行收藏动作的时间

1. repost表：转发的信息表，所有字段如下：

repost\_id: 转发ID，唯一标识一条转发行为，主键

moment\_id: 转发的动态ID

repost\_uid: 执行转发的用户ID

moment\_uid: 被转发动态的发布用户的ID

repost\_time: 执行转发动作的时间

1. following表，关注关系表：

uid: 用户ID，这条关注关系中粉丝的ID

following\_uid: 用户ID，这条关注关系中被关注者的ID

following表以uid和following\_uid共同作为主键，表中的每条记录表示了一个关注关系

### 前端设计

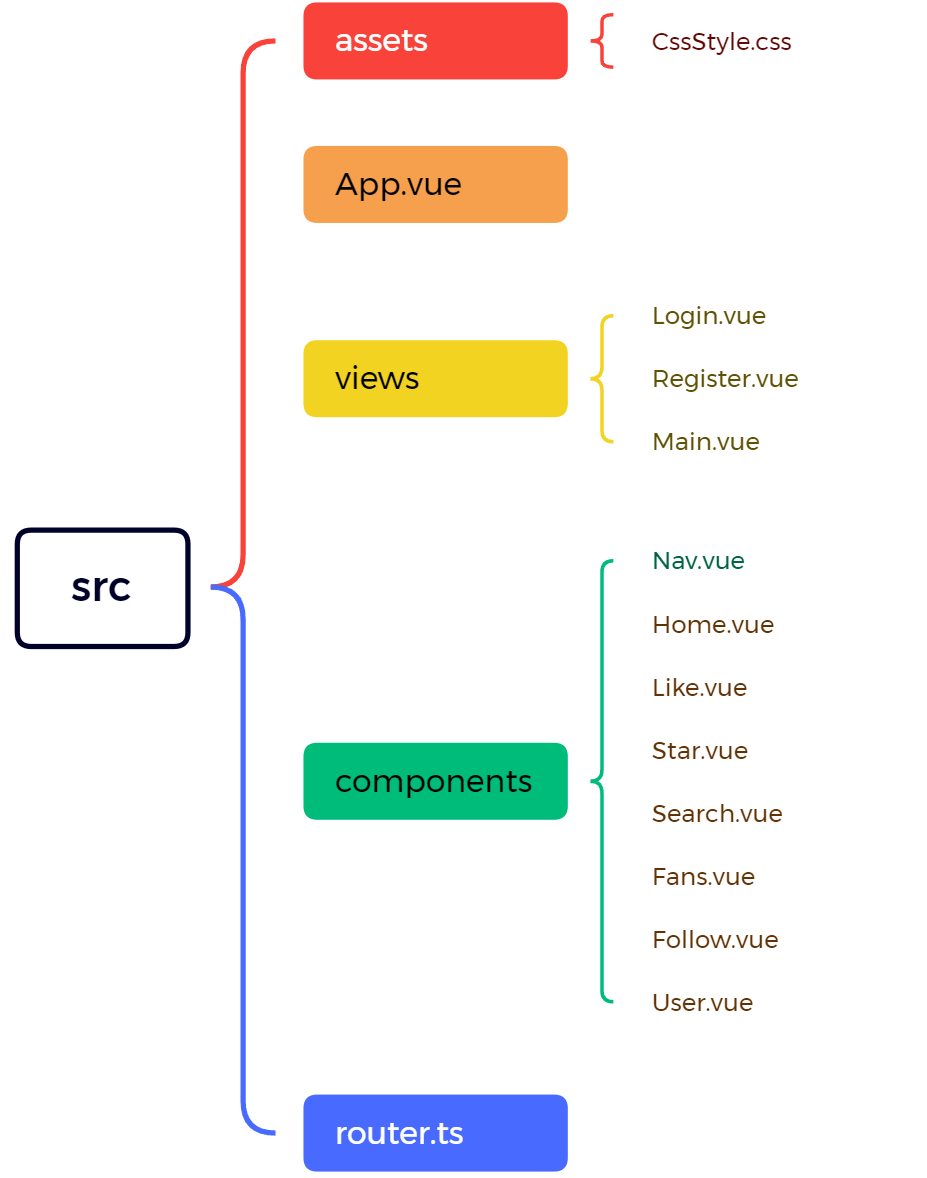


图6.1

上图显示的是前端代码核心部分，assets存放自定义的css文件和图片资源，App.vue作为网页的入口，views中定义了三个页面，分别是登录页面，注册页面，主页。Components中定义了7个组件，让main.vue去进行一个调用，7个组件分别是Nav.vue是左侧导航栏，Home.vue是主页帖子组件，Like.vue是点赞的帖子组件，Star.vue是收藏的帖子组件，Search.vue是搜索结果的组件，Fans.vue是粉丝列表组件，Follow.vue是关注用户组件，User.vue是个人主页/其他用户主页的组件。Router.ts是路由跳转声明文件，用于控制页面和组件的跳转。

登录到主页面的时序图：

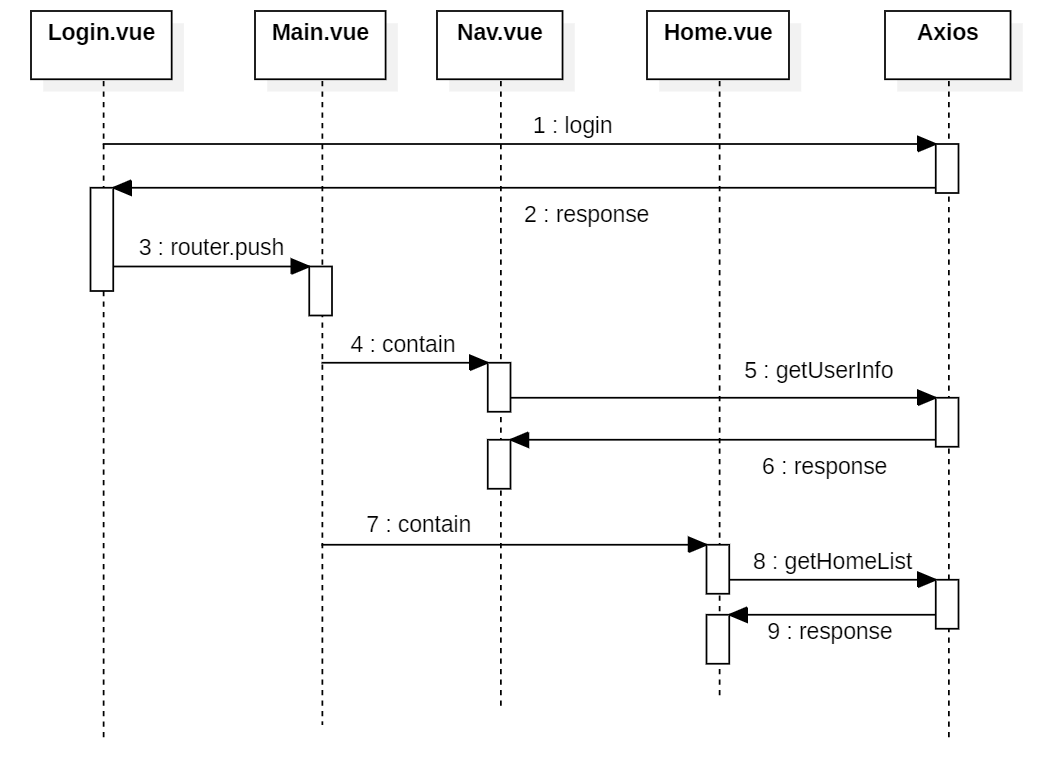


图6.2 前端时序图

在Login.vue发起登录请求后，成功时通过路由器跳转到Main.vue的页面,Main.vue包含左侧Nav.vue组件和右侧的Home.vue组件，Nav.vue组件会请求用户信息进行渲染，Home.vue组件会请求主页帖子列表信息进行渲染。

### 后端设计

1. pojo类

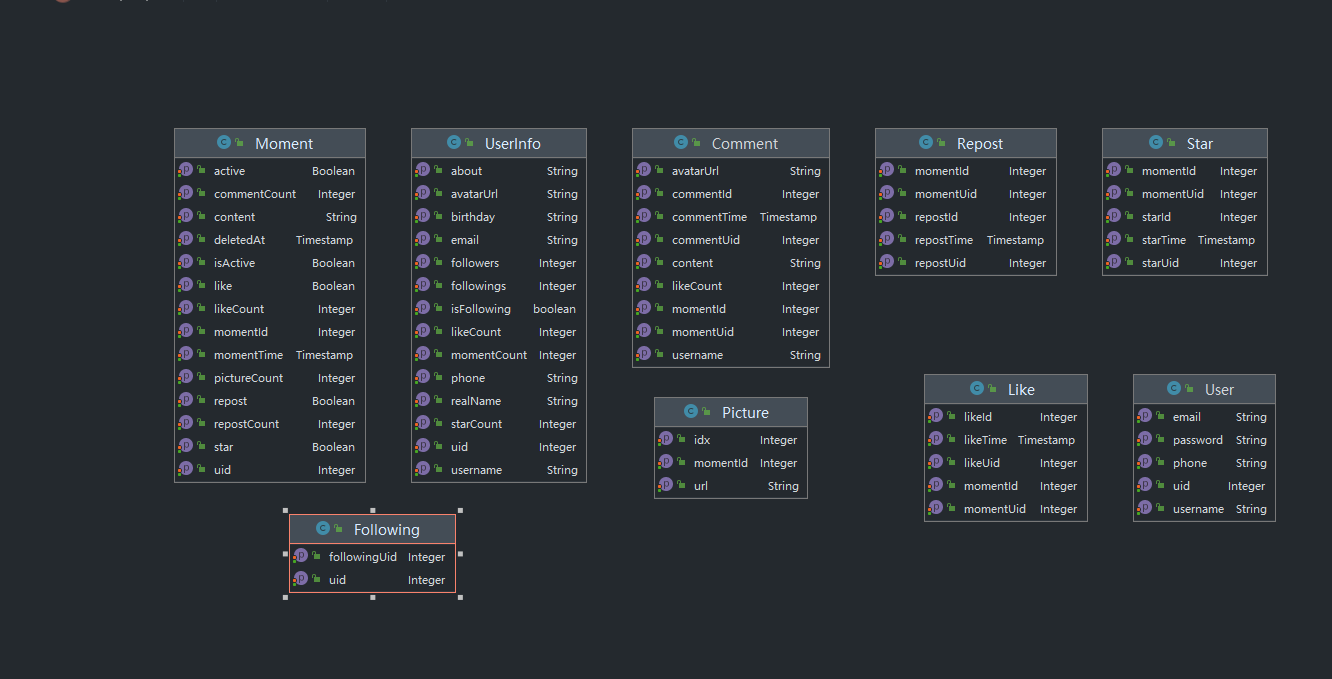


图7.1 后端pojo类

Pojo类与数据库的表一一对应，方便在后端代码中直接对应对象，将具体数据库的实现交给dao层实现。

1. 控制器类

控制器类功能主要是接收来自网页前端的请求，并将请求的参数、表单传递给对应的方法中，进而调用对应业务层中的处理类，得到业务层处理的结果后，再将结果返回给前端。一共有两个控制器类，UserController类和MomentController类。

1. UserController类

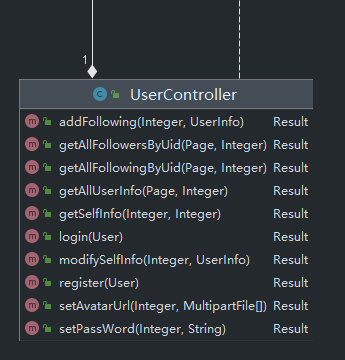


图7.2.1 用户控制器类

该控制器类用来接收和用户信息相关的请求，如注册、登录、重新设置密码、查看用户信息等。调用控制器中对应的方法，并将各种参数传给业务层对应方法，获得结果后再返回给前端。

1. MomentController类

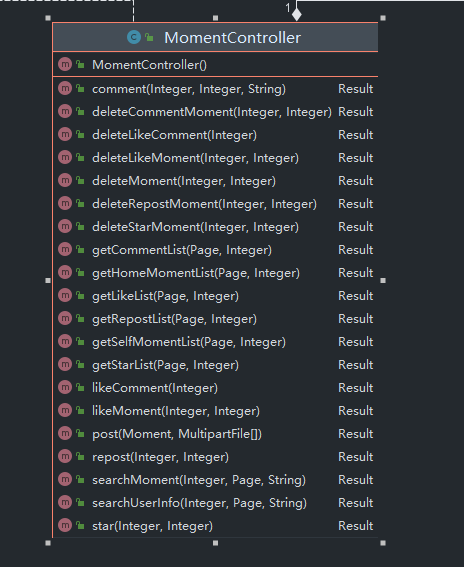


图7.2.2 动态控制器类

动态控制器类，该控制器类主要负责接收和动态有关的请求，如发布动态、点赞动态、评论动态、转发动态、收藏动态、搜索动态、获取某个用户的所有动态列表、获取主页动态列表、获取点赞列表、获取收藏列表等。调用控制器中对应的方法，并将各种参数传给业务层对应方法，获得结果后再返回给前端。

1. 数据库交互类

在业务层之前，先对数据库交互层进行介绍，因为业务层实现需要依赖数据库的操作。

数据库层面采用了MyBatis持久化框架，使得后端开发人员不需要重复关注JDBC的操作，更多是通过声明接口，再通过注解或者XML配置文件的方式写接口对应的MySQL语句来和数据库交互。

对于数据库的每个表，都有一个对应的mapper来提供接口对该表进行操作，因此一共有9个mapper，具体如下：

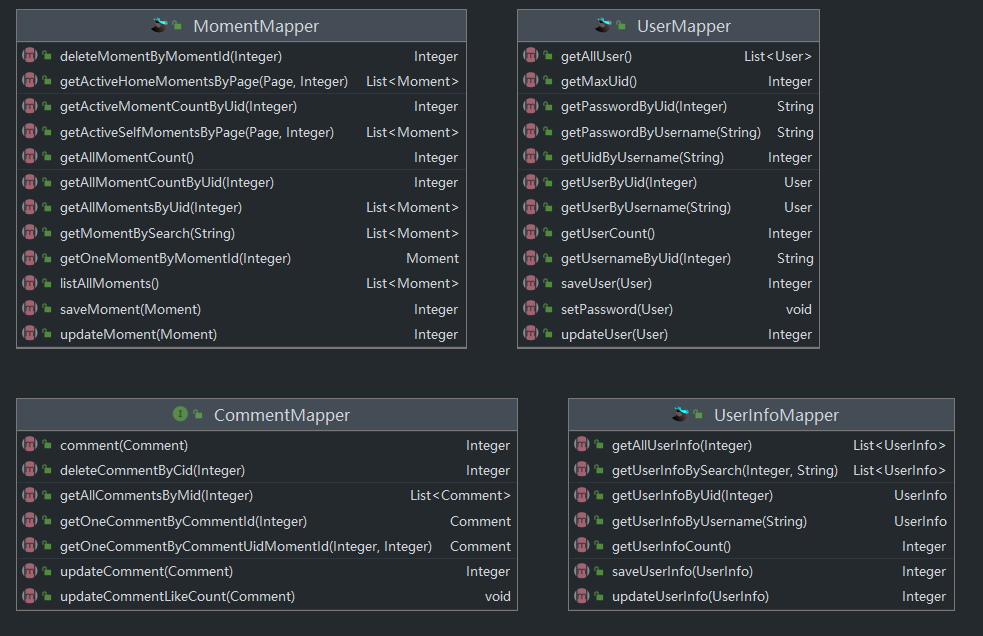


图7.3.1 mapper (一)

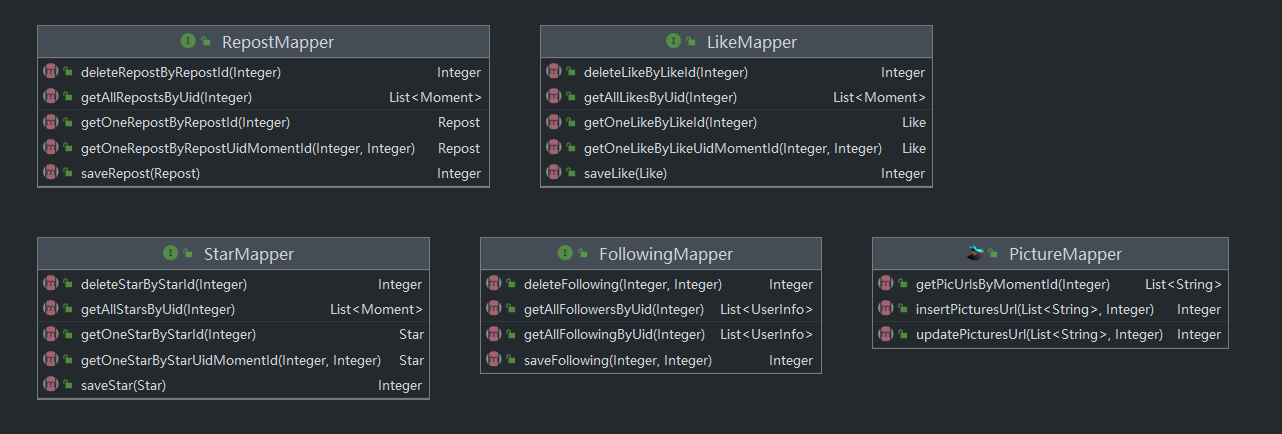


图7.3.2 mapper(二)

1. 业务类

后端设计中，controller层调用具体service层执行业务操作，包括UserService用来处理用户相关业务，如注册、登录、修改密码、修改个人信息、获取粉丝列表、获取关注列表等，如图7.4.1所示；MomentService用来处理动态相关业务，例如发布动态(评论)、删除动态(评论)、点赞、收藏、转发、获取主页动态列表、获取自己(他人)动态列表、搜索动态等，如图7.4.2所示。

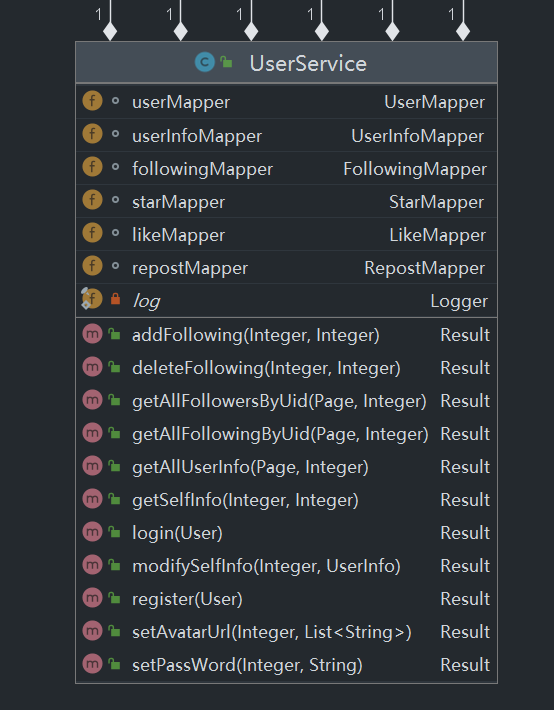


图7.4.1 UserService类

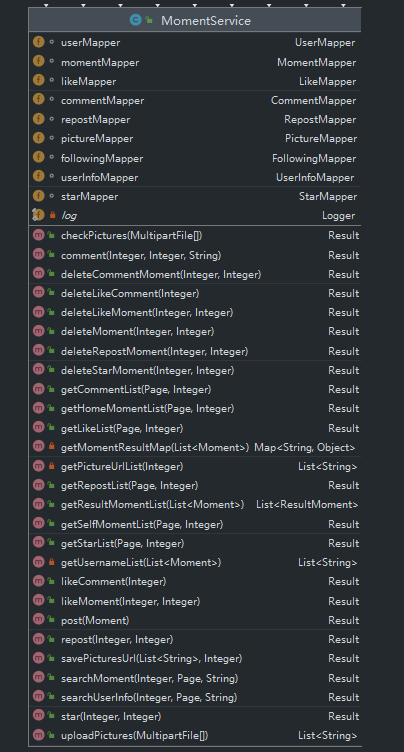


图7.4.2 MomentService类

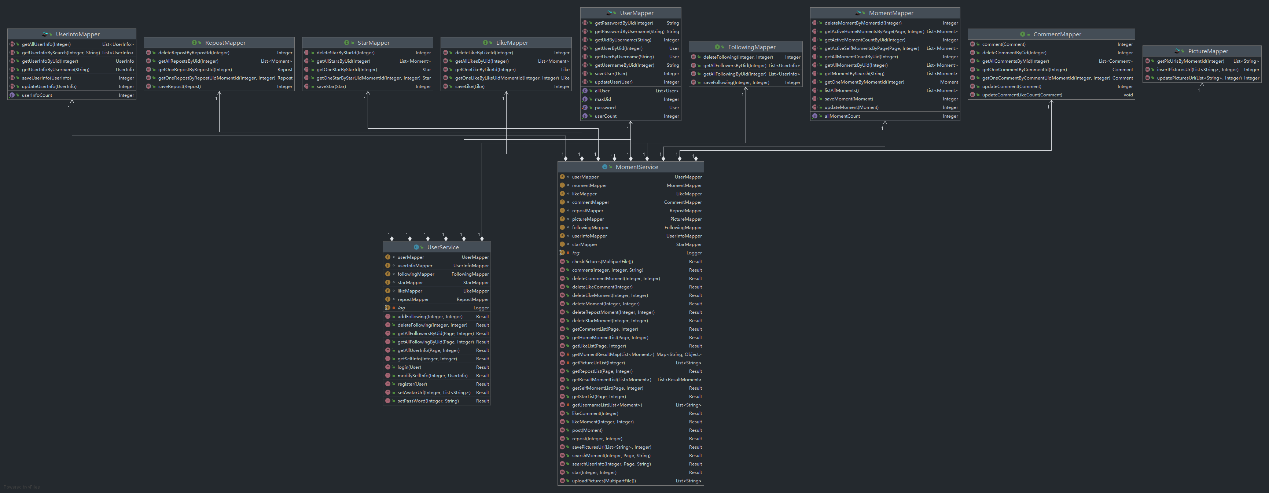


图7.4.3 Service和Mapper关系图

（五）其他类

(1) 为了更方便返回前端结果，设计了Result类和ResultCode类。Result类主要有code用来表示这次响应代码，如200表示成功；data用来存储一些需要返回的数据对象，如动态列表；msg则是这次响应需要附带的信息，方便前端提示。ResultCode则是在开发过程中自定义了一些可能出现的错误，进行封装。通过Result类和ResultCode类这两个类的封装，使得返回给前端更加便捷，几乎只需在Service层返回Result对象给Controller层，而Controller层通过添加注解@ResponseBody也可以直接返回Result对象，转为JSON返回给前端。

具体如下显示：

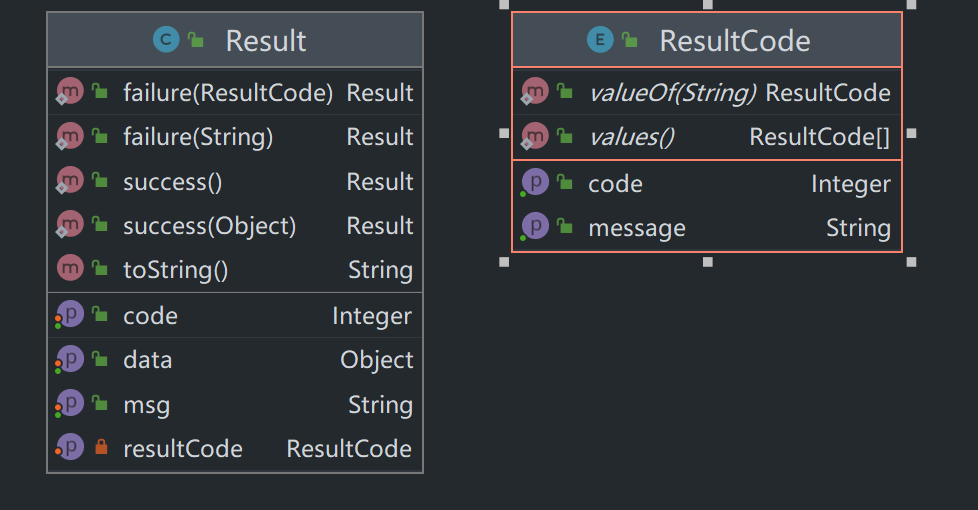


图7.5.1 Result和ResultCode类

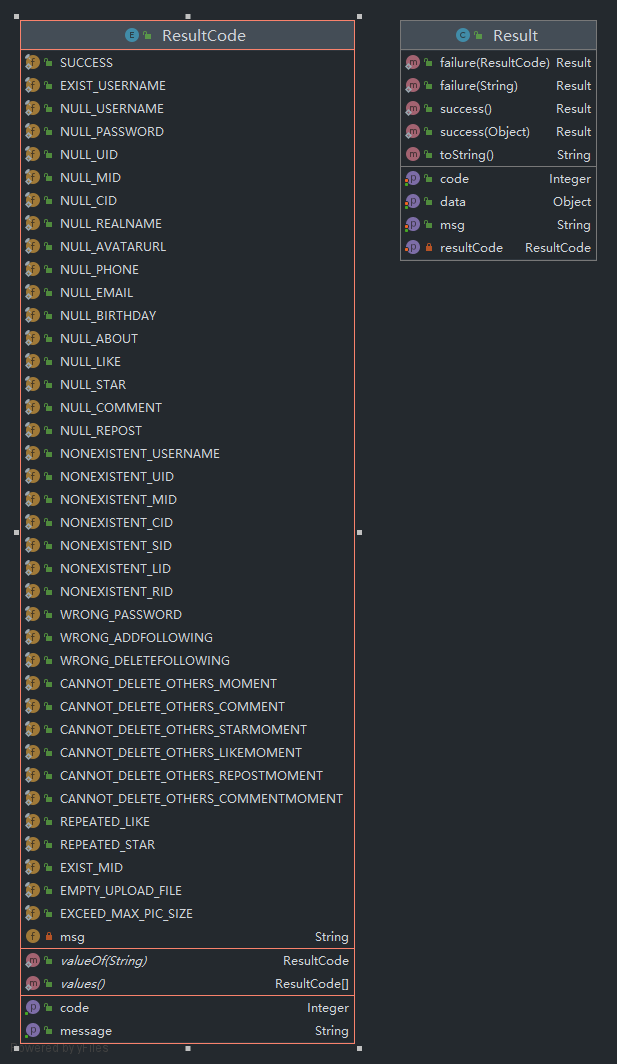


图7.5.2 ResultCode类字段

(2) 为了更方便开发，将一些共同的操作提取了出来，作为工具类，主要有以下三个类：

MyEnvBeanUtil类用来读取配置文件，UploadUtil类用来处理文件上传、图片上传相关功能，JwtUtil类用来处理token签发、验证相关内容。

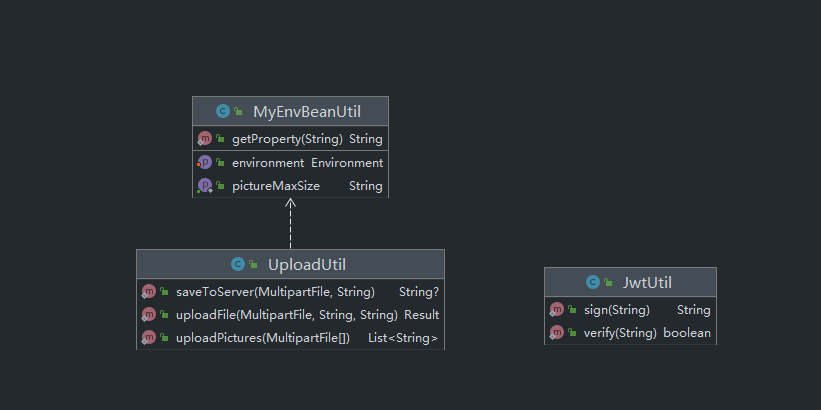


图7.5.4 工具类