**《软件详细设计文档》**

**图片兴趣社区设计与实现**

**项 目 名 称： 图片兴趣社区**

**成 员 名 单： 曾俊铭 王鹏 张衡**

**导 师： 张曙**

**工 程 领 域： 软件程序开发**

**研 究 方 向： 后端开发**

**中国科学技术大学软件学院**

1引言

1.1编写目的  
 在需求分析和概要设计文档中，已经给出该应用程序所需要的程序架构。包括系统的模块划分、各模块之间相应的api接口、api接口应该接受和返回的信息，以及数据库表的设计等。在此详细设计文档中，将对模块内部的实现进行更加详尽的说明，包括程序各个模块的详细介绍、实现算法等的设计。此文档可为开发人员进行参考。

1.2背景  
 图片是一种视觉性的内容表达，它能够直观地记录信息、传播信息。图片能弥补文字传播在感性上的不足。所以以图片分享为主题的图片兴趣社区比以文字记录为主题的社交平台更具有用户吸引力，具有很大发展空间。

国外的图片社交平台instagram具有数十亿用户。instagram是一款基于拍摄分享照片的社交软件，产品的核心就是拍摄与社交。Instagram的定位非常明确，是一个纯照片分享交流平台，吸引了一大批的用户，用户上传的照片的质量很高，分享氛围较浓。而国内的产品也层出不穷， nice、In和微博等等正处于激烈竞争中。在国内，依托着巨大的用户群体，微博、微信朋友圈占据着图片社交的大部分流量。微博具有大量繁杂功能，并不注重图片发布的品质，图片社交属性并不突出。微信朋友圈则是完全私人的图片社交，只有好友才能进行点赞评论，并且也不注重图片发布的品质。国内多数平台强调应用的功能多样性，使得应用软件越来越臃肿，逐渐使得应用的功能不突出，并且导致商业化营销元素加重。所以目前专注于开发图片分享平台，不参杂其他功能的web应用软件在国内仍然有很大的市场可发展。

1.3 参考资料

[1]许式伟. Go语言编程[M]. 人民邮电出版社, 2012.

[2]胡敏. Web系统下提高MySQL数据库安全性的研究与实现[D]. 北京邮电大学, 2015.

[3]崔岩松, 白春雨. 基于MySQL和Redis的高可用性和可扩展性的电子商务系统数据库架构[J]. 2021.

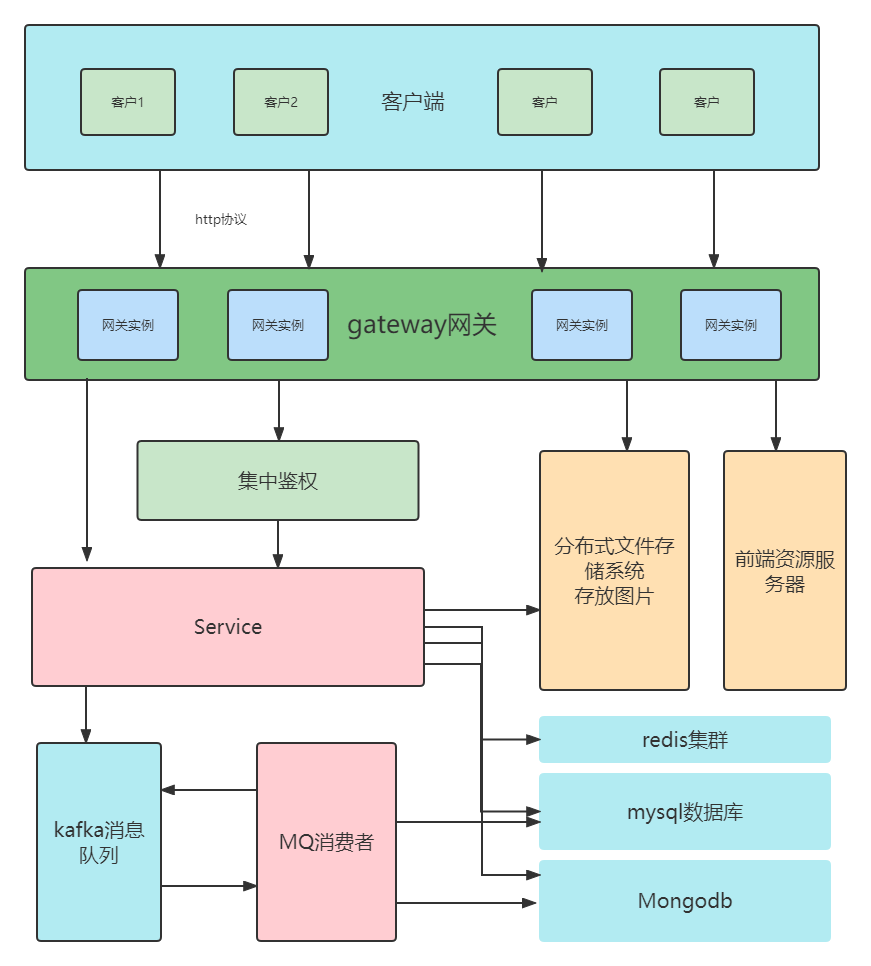
[4]曾超宇, 李金香. Redis在高速缓存系统中的应用[J]. 微型机与应用, 2013, 32(12):3.

[5]兰旭辉, 熊家军, 邓刚. 基于MySQL的应用程序设计[J]. 计算机工程与设计, 2004, 25(3):3.

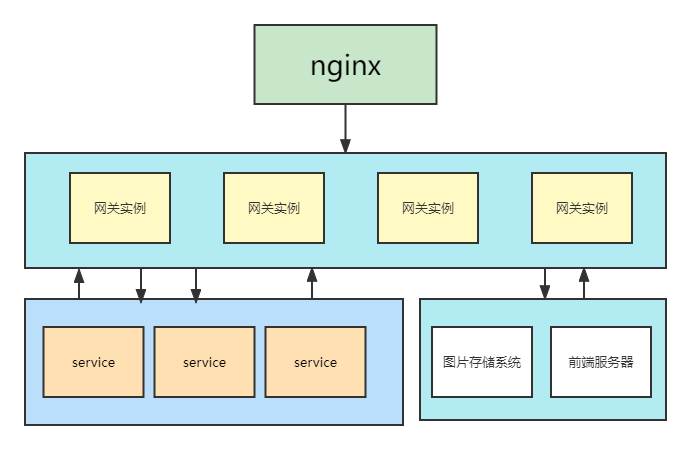
[6]焦玉乾. 基于Kafka消息队列的MongoDB日志采集和分析系统:, CN109241187A[P]. 2019.

2 项目设计

2.1项目整体架构

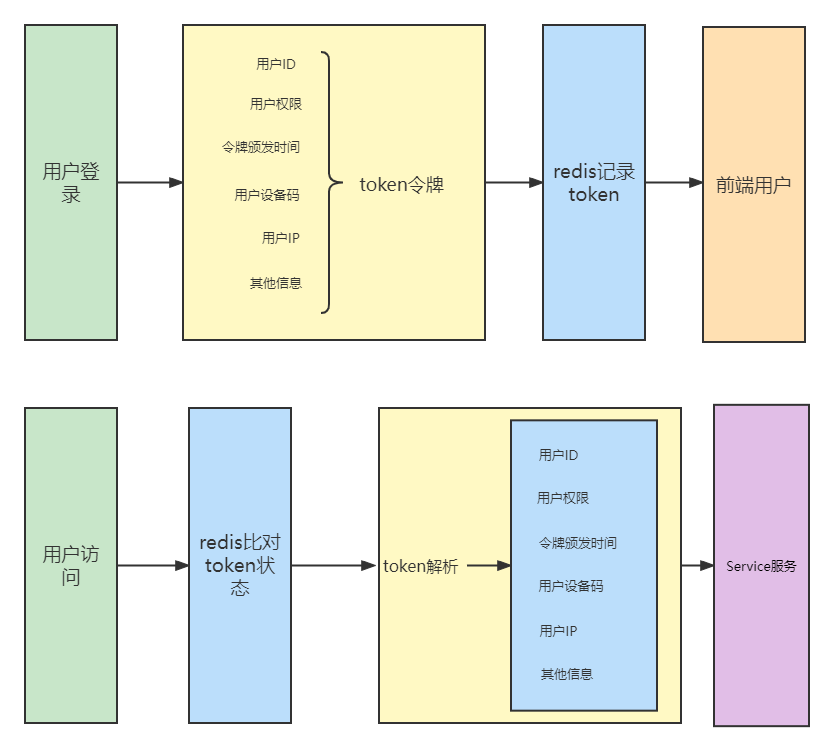
****

客户端负责展示后端提供的数据。前端通过HTML、CSS、JS和React以及TypeScript技术实现基本的页面。页面中应包含可点击的内容，每次点击后所需要的数据使用ajax技术来请求后端，避免刷新整个页面。后端收到请求后返回指定的Json格式的数据，前端收到数据后进行渲染并展示

****

网关作用：

是系统负载均衡、统计数据流量、访问控制、接口监控等。



集中式鉴权作用：

功能解耦

解决http无状态的问题

鉴权的作用：

1.接口验签

2.用户的权限管理

挤号问题

踢人下线

封禁用户

其他用户管理功能

Service层使用go语言的Gin框架设计对应的api函数以及对应URL提供给前端调用。前端使用HTTP/HTTPS协议，发出GET或POST请求到对应的URL，根据这个URL中的path进行路由选择，调用Gin框架中设计好的函数处理对应的web请求并返回相应数据。

Kafka消息队列作用：

缓冲和削峰：上游数据时有突发流量，下游可能扛不住，或者下游没有足够多的机器来保证冗余，kafka在中间可以起到一个缓冲的作用，把消息暂存在kafka中，下游服务就可以按照自己的节奏进行慢慢处理。

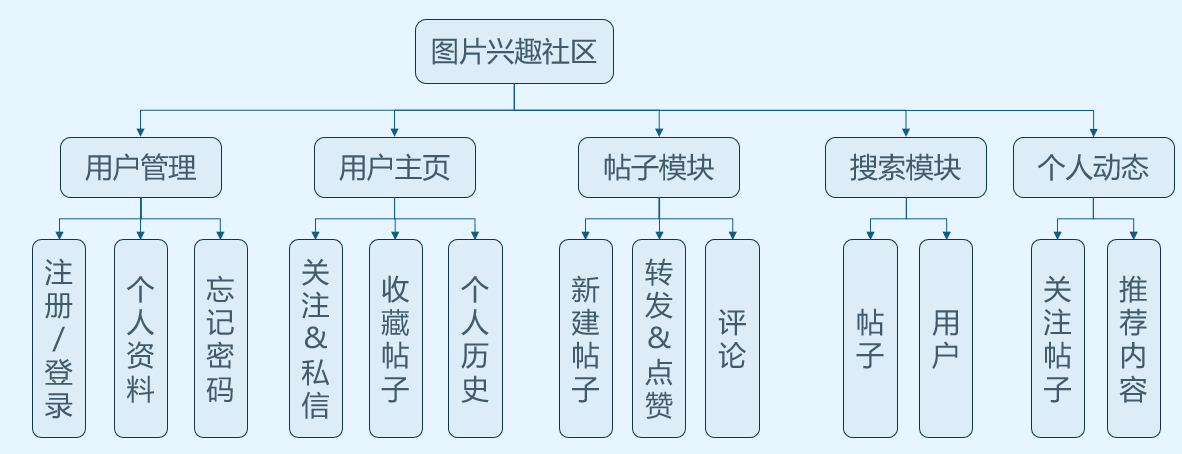
解耦和扩展性：项目开始的时候，并不能确定具体需求。消息队列可以作为一个接口层，解耦重要的业务流程。只需要遵守约定，针对数据编程即可获取扩展能力。

冗余：可以采用一对多的方式，一个生产者发布消息，可以被多个订阅topic的服务消费到，供多个毫无关联的业务使用。

健壮性：消息队列可以堆积请求，所以消费端业务即使短时间死掉，也不会影响主要业务的正常进行。

异步通信：很多时候，用户不想也不需要立即处理消息。消息队列提供了异步处理机制，允许用户把一个消息放入队列，但并不立即处理它。想向队列中放入多少消息就放多少，然后在需要的时候再去处理它们。

Redis用来存储常用数据，比如热门帖子的点赞数，评论数。避免查询mysql数据库次数过多，造成性能下降。

2.2项目模块设计

2.2.1 用户管理模块

用户注册

1. 用户进入注册页面，输入邮箱、昵称、密码、确认密码等用户信息，前端验证输入格式是否正确，正确才允许点击注册按钮，发送request请求给后端；
2. 后端根据邮箱查询是否已经存在该账户；
3. 若已存在该用户，返回前端注册失败；
4. 若未存在该用户，将该用户信息插入用户信息表中，返回前端注册成功，并跳转到登录页面。

此页面向用户提供注册功能，页面界面如下所示



* 用户名、密码、确认密码必须填写
* 用户名仅包括字母、数字，长度上限为20个字符
* 密码包含字母、数字和其他字符，长度上限为20个字符
* 输入密码或者确认密码时可以选择隐藏或可见。
* 输入用户名重复时候，无法注册，需要重新输入。
* 输入的两次密码不相同时会显示密码不匹配，此时无法注册
* 信息填写完成之后点击注册按钮进行注册

用户登录

1. 用户进入登录页面，输入账号密码，发送request请求给后端。
2. 后端根据账号和密码查询用户信息表，看是否有该账户。
3. 若无该账户，返回给前端response信息，提示登录失败。
4. 若存在该用户，将该用户存入session，同时修改用户状态表中的状态为在线状态，返回前端登录成功，跳转到首页。

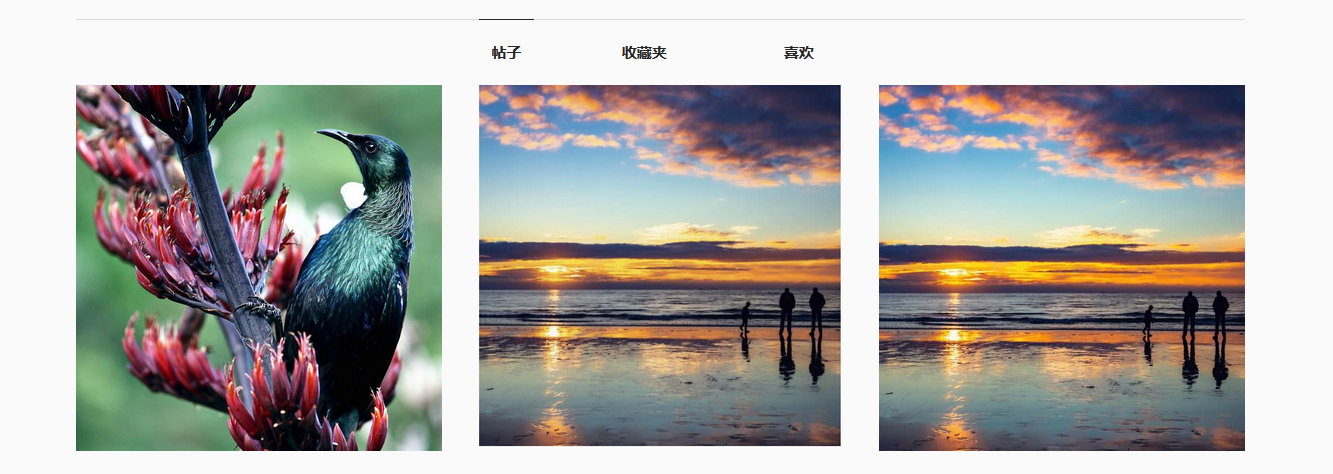
登录页面界面如下所示

* 用户名、密码必须填写
* 密码错误或者用户不存在时无法登录
* 点击忘记密码可以找回已经注册账号的密码
* 勾选记住我选项时保存该用户名和密码。
* 点击游客登陆可以使用系统默认的账号进行登录。

2.2.2 用户主页模块

**用户发布、收藏、点赞过的帖子**

1.进入用户主页模块，点击相应按钮前端直接向后端发起get请求。

2.后端去数据库中根据用户的uid查询该用户发布、收藏、点赞过了哪些帖子并进行显示。

**关注用户**

1.用户可以去关注自己喜欢的用户。

2.点击关注后，前端会发起一个post请求给后端。

3.后端主要对两个表的进行数量+1操作，一个表是fans表，一个表是follow表，分别表示粉丝以及关注。



**编辑用户信息**

1.用户可以去修改自己的资料，包括昵称、个性签名等。

2.点击编辑后，会跳转到编辑页面，填写修改后前端会发起post请求给后端。

3.后端将拿到的数据存入mysql数据库。





**私聊**

1.用户可以与其他用户进行私聊。

2.用户进入私聊页面，首先与后端进行WebSocket连接。然后发起一个get请求获取该用户的历史聊天用户列表，然后发起另一个get请求获取与当前私聊用户的聊天记录。

**发送私信的3种情况 ：**

给某个用户发私信

1.对方不在私信界面（即没有和服务器进行WebSocket连接）。此时把私信存入到mysql数据库。

2.对方打开了私信界面，但聊天的对象不是你。私信存入到mysql数据库。

3.对方打开了私信界面，且聊天对象正好是你。此时双方可以实时发消息聊天。私信存入到mysql数据库。

2.2.3 帖子模块

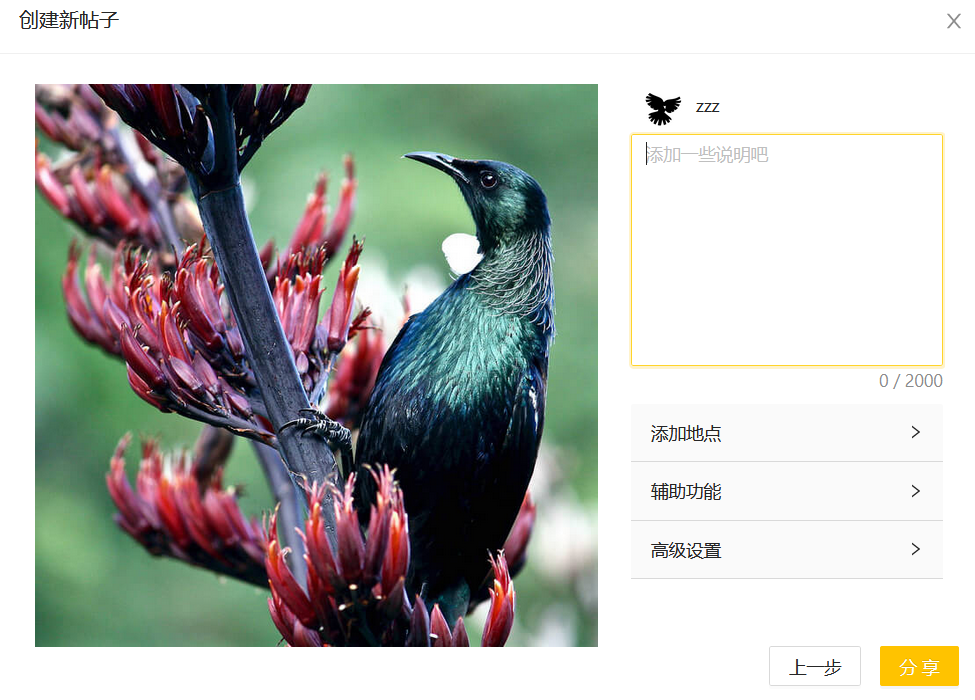
**上传帖子**

1.用户如果想发布帖子那么就需要上传图片，以及对该图片的描述，之后点击完成按钮。

2.点击之后，前端会向后端发起一个post请求。

3.后端会把上传的图片保存在服务器的磁盘中，并且在photo表中记录图片的url路径，与此同时再去post表中新建记录，用来记载帖子的一系列信息，包括发布时间，发布用户等。



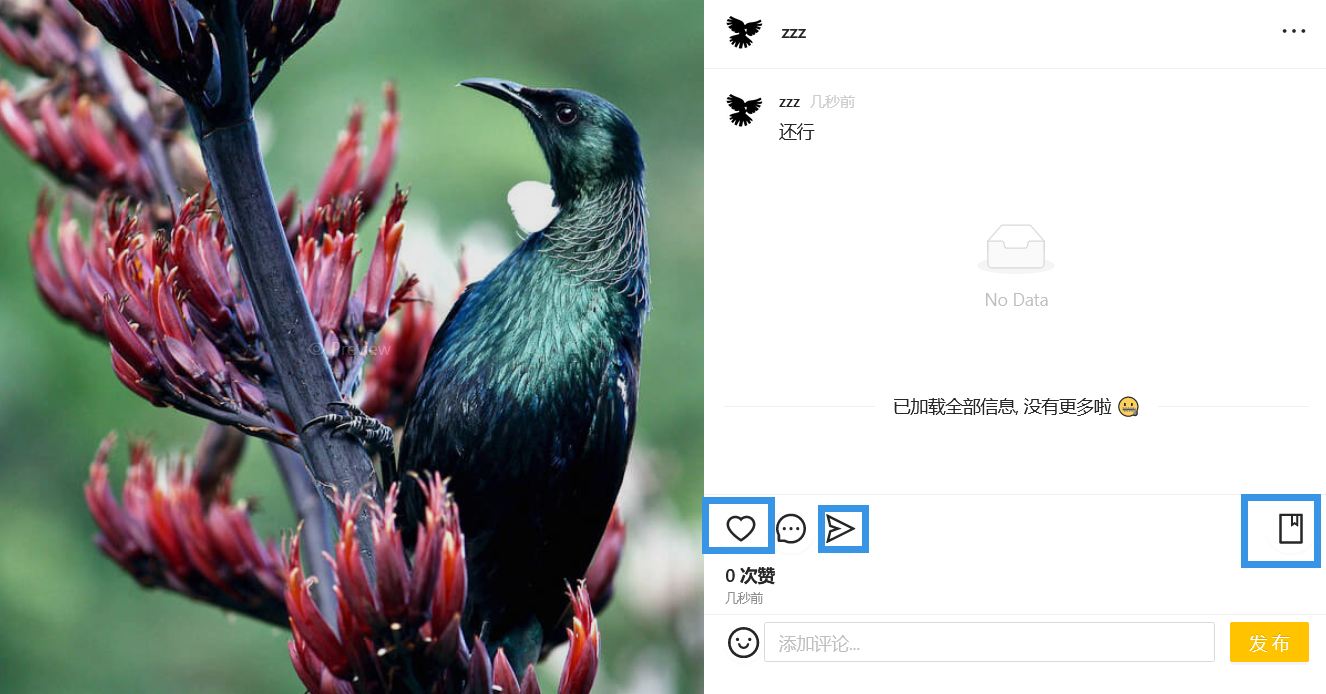


**点赞、转发、收藏帖子**

1.用户看见喜欢的帖子可以进行点赞、转发、收藏。

2.前端点击按钮后会向后端发起一个post请求。

3.后端去数据库中的表中会新建或修改一行记录，两个主要的字段是uid以及pid，表示哪个用户和相关的帖子。

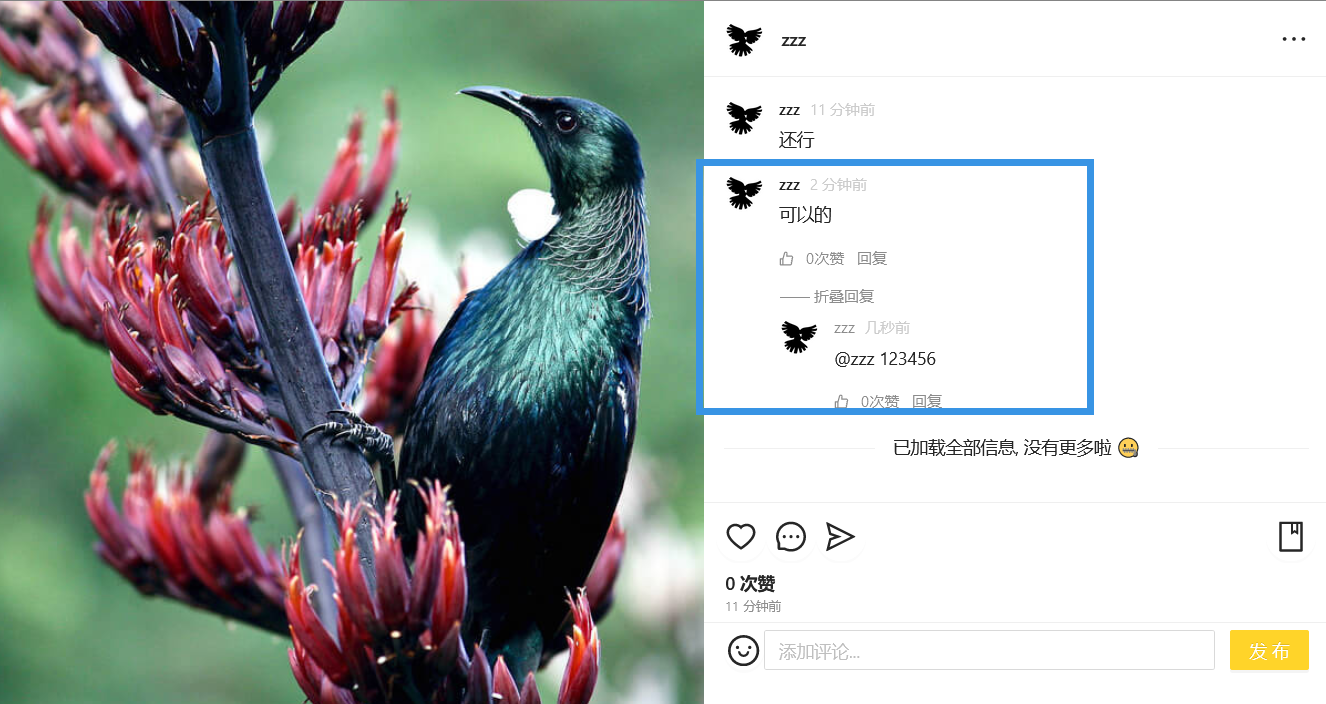


**评论和回复**

1.用户可以对帖子以及帖子下面的评论进行评论。

2.发表评论时会发起一个post请求，如果是一级评论，则parent字段为0，否则，parent字段为1，表示该评论是评论他人已经发表的评论。

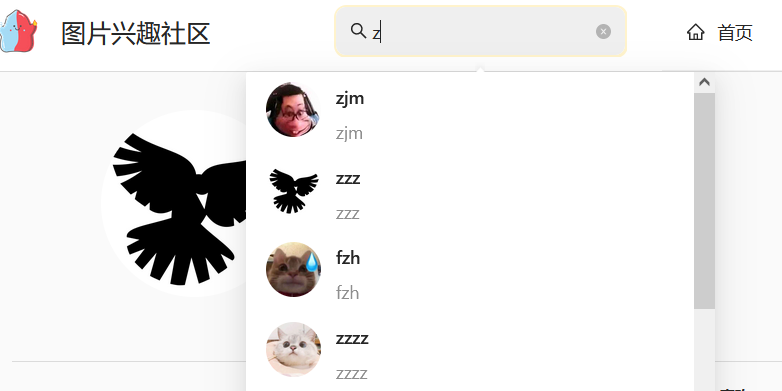
3. 后端将评论存储在数据库中，打开帖子时从数据库获取评论和回复。



2.2.4 搜索模块

1.用户可以在搜索框输入用户ID搜索用户，支持模糊搜索。

2.输入用户ID时发起一个get请求，用户输入作为参数传递给后端。后端在Mysql数据库根据该参数作为模糊查询条件，查询对应的用户信息并返回给前端。模糊查询根据Mysql的like语句实现。



2.2.5 动态模块

个人动态是一个类似于朋友圈、微博动态的功能。

用户可以在个人动态里看到所关注的人最近发的帖子

拉方式pull

查询帖子表，将作者id=自己关注的人的帖子拉出

* 对于小数据规模可以采用连表查询
* 对于大数据量采用多次单查询

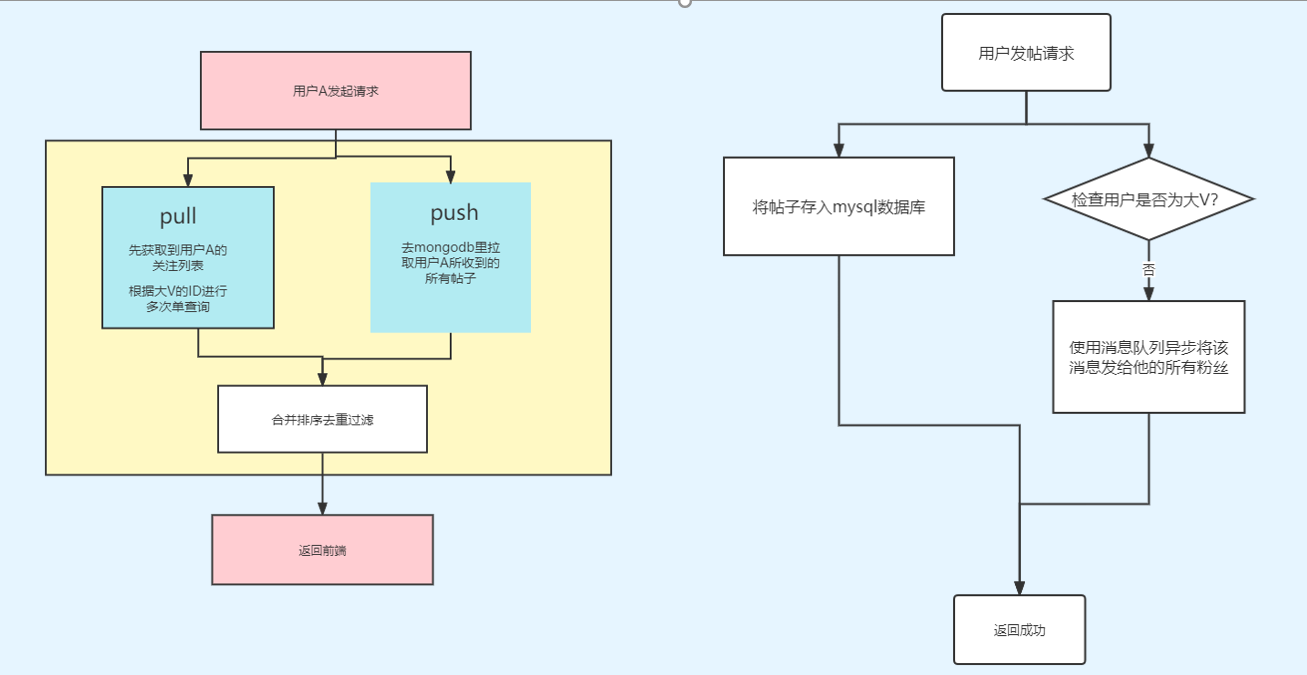
缺点：数据规模大了整体偏慢

推方式push

发帖用户每次发布新帖子时，将其帖子id放到自己每一个粉丝的帖子列表里

看帖用户每次只需把自己帖子列表读出即可

缺点：对于“大V”数据冗余严重 瞬时负载过大

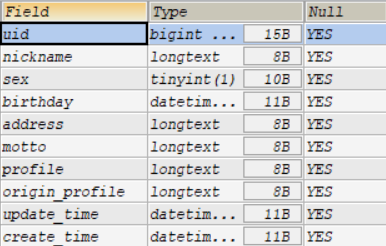


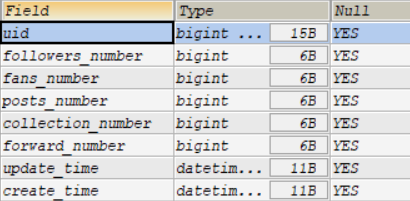
2.3 数据库设计

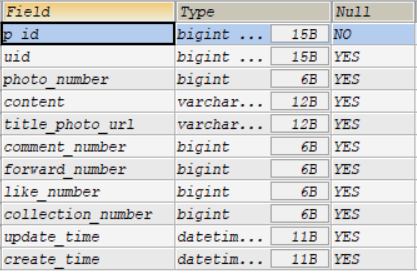
建立数据库所涉及的表如下所示：

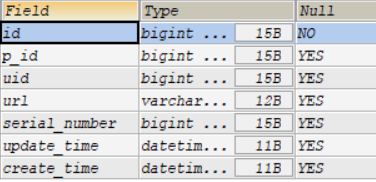
用户表：user（存储用户账号、密码等信息）

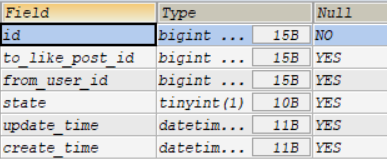


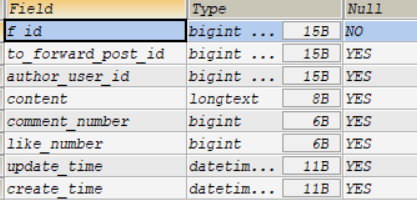
用户信息表：user\_detail（存储用户昵称、个性签名、头像等信息）

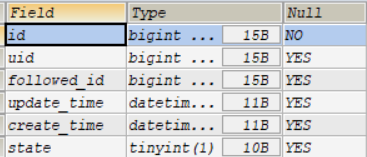
用户数据表：user\_data（存储用户关注、粉丝、收藏等数据）

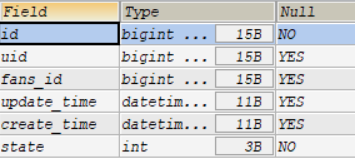
帖子表：post（存储帖子点赞、收藏等数据）

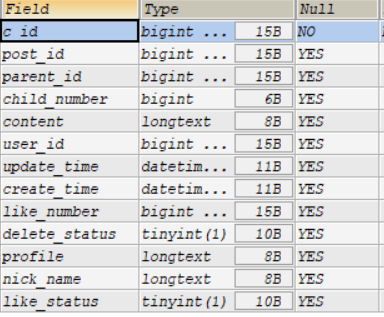
帖子图片表：post\_photo（存储帖子图片url）

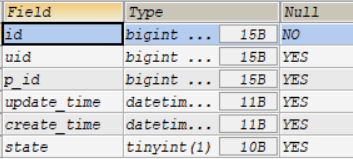
点赞表：liked（存储用户点赞信息）

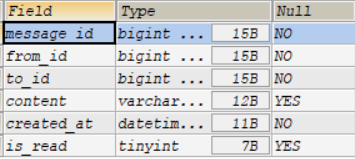
转发表：forward（存储用户转发信息）

关注表：follow（存储用户关注uid信息）

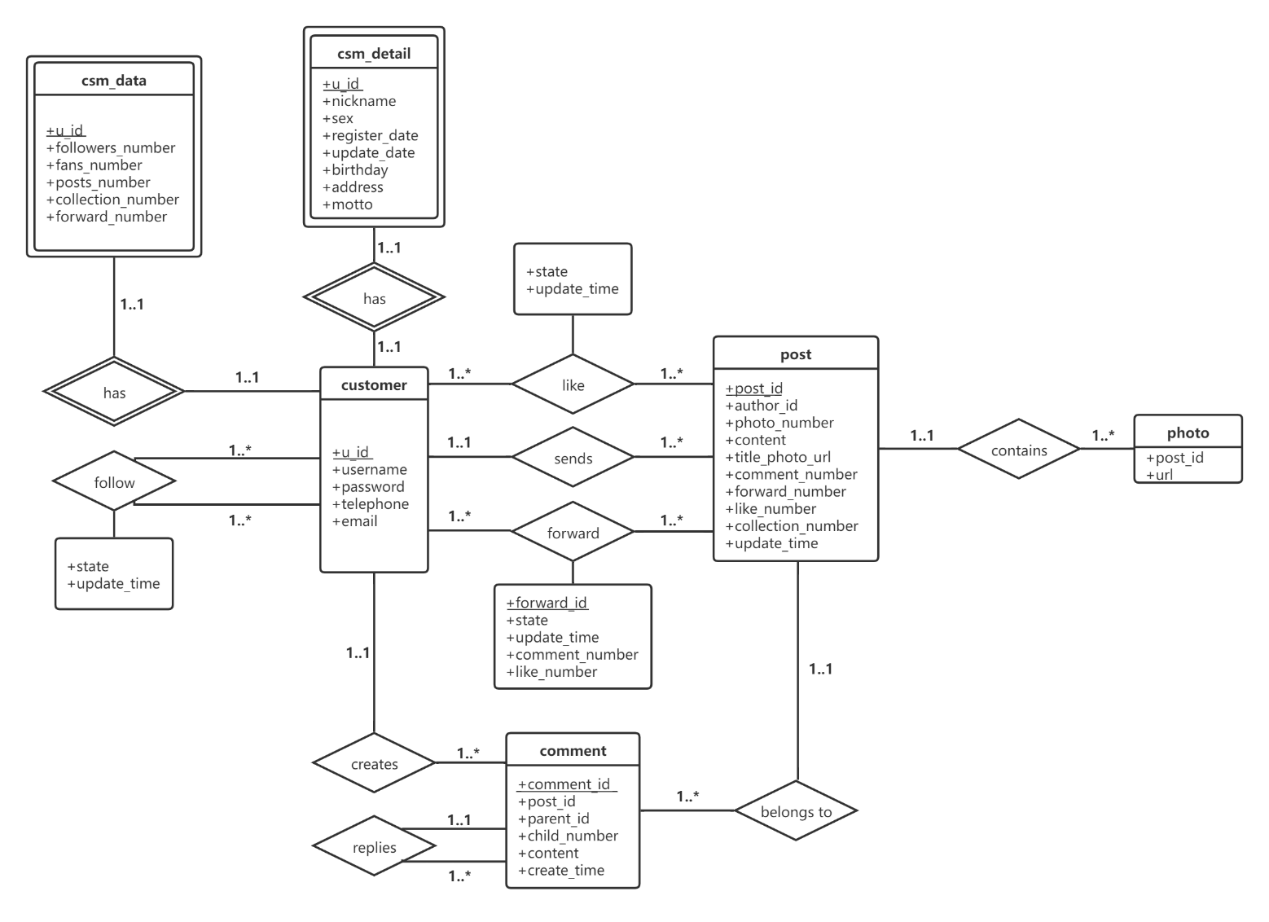
粉丝表：Fans（存储用户的粉丝uid信息）

评论表：Comment（存储用户在一个帖子的评论信息）

收藏表：Collection（存储用户收藏的帖子pid等信息）

私聊表：Chat\_message（存储用户私聊信息）

E-R实体联系图



2.4 图片存储

使用fhds来存储图片

* 在上传图片时，将其存入fhds后，会生成一个url标识定位该图片
* 在mysql里只存放该url

访问时：

* 先访问后端service去获取该图片的url
* 再根据url去图床里调取图片