前后端数据交互接口规范

# 登录/注册页面

## 注册的格式

{

‘phone\_number’:12345,

‘mail’:12345@xx.com,

‘username’:12345,

‘password’:12345,

}

（注册界面是否给一个用户名比较好？）

可选项：前端约束条件，password不能过于简单。

进入后，后端首先获取手机号以及邮箱。如果这两种已经存在，则返回一个错误。

返回格式：

{

‘regist\_success’:False,

‘wrong\_code’:1,

}

‘regist\_success’显示注册是否成功。如果是True则表示成功，否则表示失败，需要继续看wrong\_code。

‘wrong\_code’：1表示电话已经注册，2表示邮箱已经注册

## 登录的格式

{

‘mail’:123@xx.com,

‘password’:1234

}

后端判断。

{

‘load\_success’:False,

‘wrong\_code’:1,

‘user\_id’:user\_id

}

‘sigh\_in\_success’表示登录是否成功。

‘wrong\_code’：1表示此邮箱没有注册过，2表示密码不正确

‘user\_id’表示如果登录成功的话，就将用户id发送给前端，以方便查找

# 首页功能

## 推荐页

JSON 分成三个部分：article\_list，user\_infor，tips

注意，在最终的article\_list当中，理想中的内容应当是根据用户的关注，针对用户感兴趣的内容进行推荐，但是在初步的实现当中还是就随机选取吧。

在登录之后，用户应当传入如下的信息：

{

‘userId’:id,

}

Article\_list由几个article元素构成，每一个article元素构成如下：

{

‘article\_name’:name,

‘article\_content’:content,

‘author’

‘related\_good\_list’:[good\_id1,good\_id2,…],

‘related\_good\_infor’:[good\_infor1,good\_infor2,…]

‘id’:id

}

在article\_list当中，仅仅放10个article。

User\_infor:{

‘collection’:[collection\_infor1,collection\_infor2,…].

‘history’:[history\_infor1,history\_infor2,…]

}

其中，collection\_infor如下collection\_infor:{

‘type’:123,

‘name’:123,

‘id’:id

}

之所以collection\_infor有一些别的信息，是因为这个地方除了链接之外还需要进行一些必要的信息呈现。

History也是一个列表，存储了history\_infor,如下：

{

‘type’:123,

‘name’:123,

‘id’:123

}

tips :{

‘likes’:True,

‘reply’:True,

‘message’:True,

}

Tips当中，如果有人进行互动的话，这些东西都会变成True，并且图标是要有所改变的。

点击点赞或者回复、私信之后，将会跳转到个人中心当中。

总体如下：

{

Article\_list:[article1,article2,…,article10],

User\_infor,

Tips,

}

如果用户滑到了底端，则自动发送一个GET或者POST信息，后端将发送一个Article\_list

到前端。

## 关注页

关注页和推荐页的JSON内容是一样的，只不过在选择article的时候，要根据用户关注的情况来选择。但是初步实现的时候就随机选择了。

## 热榜页

用户在点击“热榜”的时候，将会跳转到热榜页面。这两个页面仅仅只是主页面有所不同，其他情况是相同的。因此仅仅将good\_list来替换article\_list。如下：

Good\_list是由good构成的，属性如下：

Good{

‘name’:name,

‘img\_path’:img\_path,

‘property’:property（这个的具体内容还没有确定好）

}

# 个人中心功能

## 私信功能

当用户点击个人中心，或者是Tips当中的私信按钮时，将会跳转到私信界面。在这个界面当中，将会发送的数据包括：

Users和messages。这两个数据都是列表，列表中的内容分别是User和message

User：{

‘name’:name,

‘profile\_path’:path,

‘user\_id’:user\_id

}

Message{

‘content’:content,

‘time’:time,

}

即，总体的JSON文件是：

{

Users:[User1,User2,…],

Messages:[message1,message2,…]

}

这意味着在登入到个人中心的时候，前端页面仅仅拿到了和某一个特定用户的对话信息并且按照时间前后进行渲染。

当用户想要和另外一个用户进行聊天的时候，他必须点击此人的头像，并且向后端传送一个JSON：

{

‘user\_id’:user\_id,

‘target\_user\_id’:target\_uesr\_id

}

即当前用户的id（从cookie当中获取），和目标用户的id。传送到后端之后，后端发送一个messages，格式与上述是一样的。

当用户发送了一条消息的时候，向后端发送一个信息：

{

‘user\_id’:user\_id,

‘target\_user\_id’:user\_id,

‘content’:content,

‘time’:time

}

## 点赞

进入到点赞页面之后，将会传入一个like的列表，每一个like的结构如下：

{

‘total’:number,

‘last\_like\_user’:user\_name,

‘content’:content,

‘like\_id’:like\_id

}

Total 是指总共有多少人进行点赞，last\_like\_user对于最后点赞的人的姓名进了记录，‘content’记录了这条点赞的comment或者reply的内容，like\_id记录了这个点赞的id。  
从而用户在进入该页面的时候，页面可以为用户反馈一个列表“XX等XX人点赞了我：XX（此处为评论的摘要）”

当用户点击这个内容的时候，网页将会跳转到到用户被点赞的这个文章/商品处，自动打开评论区并且将用户被点赞的这个回复置顶（如果这个置顶做的太麻烦的话就算了）。这个动作需要传送的JSON将会在下文进行揭示。

## 评论

进入到评论页面，将会传输一个列表replies，每一个reply的结构如下：

reply{

‘user\_name’:user\_name,

‘reply\_id’：reply\_id,

‘type’：‘comment’,

‘content’:content

}

注意，这里的reply是一种广义的回复，它可能是指评论comment，也可以是指对于评论的回复reply。但是此处不宜将这两种回复用不同的list区分开，因为最好是按照时间的顺序进行排序，而不是分类别进行排序。

这也是为什么要设置一个type参数的原因。如果type为comment，那么渲染的时候应当是XX（使用user\_name填充）评论了你。如果type为reply，那么渲染的时候应当是XX回复了你。无论是哪种情况，都应当将content的内容渲染到页面上去，以帮助用户回忆这个评论的状况。

## 收藏和历史记录

无论是哪一种，其传递的都是一个good\_list，good\_list当中的项是good\_infor，如下：

good\_infor:{

‘good\_id’:123,

‘good\_name’:123,

‘good\_type’:123,

}

# 评论以及互动

评论区的交互是比较复杂的。

## 评论区的渲染

无论是针对文章还是商品，评论区在一开始都不是直接展示的。当用户点击“显示评论”或者从其他地点跳转到评论区页面的时候，评论区进行渲染。

当用户点击“显示评论”的时候，将会向后端发送一个请求，并且带有JSON信息：

{

‘target\_id’:target\_id

}

这个target\_id是文章或者商品的id，以便于后端到数据库当中去取数据。

后端返回给前端的JSON格式如下：

Comment\_list[comment1,comment2,…]

Comment{

‘content’:content,

‘username’:username,

‘id’：id，

‘user\_id’：id,

‘profile\_path’:profile\_path,

‘time’:time,

‘like\_number’:like\_number,

‘reply\_list’:[reply1,reply2,…]

}

Reply{

‘content’:content,

‘username’:username,

‘target\_user’:target\_user,

‘reply\_type’：type，

‘time’:time，

‘target\_id’:id,

‘id’：id，

‘user\_id’:id,

‘like\_number’:like\_number

}

由于reply既可以回复comment，也可以回复reply，所以使用了reply\_type进行了区分。如果回复的类型是comment，则不需要展示target\_user（因为这种情况是默认的），否则是需要展示target\_user的。然后之所以要将id作为一个组件参数，是因为在添加评论/回复的时候会用到。

在渲染评论区的时候，可以考虑仅仅渲染出comment而不渲染出reply。只有当用户点击“查看回复”的时候才将reply的内容呈现出来。

## 评论/回复的添加

在添加评论的时候，需要传递一个json：

{

‘user\_id’:user\_id,

‘target\_id’,target\_id,

‘content’:content

}

target\_id是被评论的对象的id。然后，后端将会补齐像时间之类的参数。

在添加回复的时候，需要传递一个json

{

‘user\_id’:user\_id,

‘target\_user\_id’:user\_id,

‘comment\_id’:id,

‘content’:content

}