

Dweibo—一个去中心化的社交平台

16340218 王阔

github地址：<https://github.com/wkfdb/Dweibo>

选题背景

微博目前是国内无比流行的社交软件，大家可以开通自己的账号，然后发布各种内容。喜欢你的人会关注你，关注的人多了也会给人带来一定的成就感。但是微博在使用过程中一个十分严重的问题，就是它实在是太小心了。你发布的消息不能包括任何敏感词，虽然那些敏感词有的也不是很敏感，而且你发布正常内容有时候也会因为敏感词问题而被删掉，许多讲述真相的文章也经常会被下架。这就体现了中心组织的坏处。你可能没有隐私，你在微博上所有的内容对于中心组织来说都是公开，他们不仅能够获取你的信息，甚至能篡改这些信息。于是我就构想开发一个无中心组织的类微博的去中心化应用。这里你是完全匿名的，除非你在上面发布你的真实信息，否则就没人知道是你，并且由于没有中心组织，你不用担心你的信息被修改，因为只有你才能修改自己的信息。这样你发布的文章只要你不想改就没人能够动他。在这里你可以关注新的朋友，看他们发布的动态，你也可以发布动态，宣传自己，使你成为一个大V。

基本功能

- 账户管理。每个用户都应该拥有一个账户。这个账户保存了该账户的地址，用户名，密码需要自己保存，调用以太坊的账户解锁功能可以实现用户的登陆。账

户还保存了自己的动态，自己的朋友们的账户地址。

- 账户基本信息的修改。除了账户地址和账户ID(合约内部使用)之外的其他账户相关信息都设置了响应的函数来修改。
- 添加关注。你可以通过一个账户地址来添加该地址对应的账户来成为自己的关注。关注之后你就能获取他的动态。
- 取消关注。你可以取消你关注的账户。
- 添加动态。你可以在全网发布动态。关注你的人可以看到你的动态。
- 获取好友和自己的动态。你可以获取你的所有好友的所有动态。
- 删除动态。你可以删除你自己的动态。
- 各种信息的获取。全网注册的账户量，你的账户ID，你的用户名，你的关注，你的动态等。

代码讲解

合约与上次上交的合约部署报告相比基本没有区别，直接看上次作业的代码讲解即可。

使用说明

首先，本项目通过node.js来实现，总共需要安装的npm包有：

- gannache-cli
- truffle 4.1.15
- express
- body-parser
- cookie-parser
- multer
- web3 0.20.0

首先truffle init来初始化一个环境：

```
C:\Users\王阔\Desktop\Dweibo>truffle init
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:

Compile: truffle compile
Migrate: truffle migrate
Test contracts: truffle test
```

初始化之后，将编写好的sol文件放进自动生成的contracts文件夹。

migrations文件夹是拿来部署合约用的。在里面编写部署文件十分容易，新建一个js文件，命名格式为 序号 + 名字 + .js，然后按如下所示编写：

```
var Dweibo = artifacts.require("./Dweibo.sol");
module.exports = function(deployer) {
deployer.deploy(Dweibo);
};
```

然后呢因为我们要部署到私链上，所以设置truffle.js文件，告诉它应该将合约部署到哪里。truffle.js文件中的内容如下：

```
module.exports = {
networks: {
development: {
host: '127.0.0.1',
port: 8545,
network_id: '*' // Match any network id
}
}
}
```

这里的端口为什么是8545呢？因为我们的私链开启的端口就是8545.

(代码仓库里contracts文件夹就是已经部署好的truffle文件夹。直接进入到那个目录下运行 truffle.cmd migrate即可。)

然后我们打开安装好的ganache-cli，开启一个私链：

```
C:\Users\王阔>ganache-cli 视图(V) 主题(T) 帮助(H)
Ganache CLI v6.2.5 (ganache-core: 2.3.3)

Available Accounts
=====
(0) 0xd4f17ff6e5177053e793a3bd6ace6cc9224cfa07 (~ 100 ETH)
(1) 0xb5567bc206795a629846df6f086d5a6d912225f6 (~ 100 ETH)
(2) 0xcad92f4c3d6f0e4c41994073844708ff8d93b424 (~ 100 ETH)
(3) 0x6679720ea5e862160263592e9bb9fb2400afc7c (~ 100 ETH)
(4) 0x671fce48c4945b505d7df488fd69b354a2f31238 (~ 100 ETH)
(5) 0x2cf9c1047e84c0a339d7dd43ad4e9998011cba9d (~ 100 ETH)
(6) 0x8840ac65d056b8b1551195a96f79440a5b2346a1 (~ 100 ETH)
(7) 0x7fed883afbea77f87556866dcda3d362c4441999 (~ 100 ETH)
(8) 0x13e6a38d16f0a93fb0f9ed6b7a7874166e1d9d01 (~ 100 ETH)
(9) 0x29031198e3745e9746ed6dbdbf0eeee73fcfa9d8 (~ 100 ETH)

Private Keys
=====
(0) 0xbc08b50cda1da8346e8d455858d4d808add567fc2825339539db6870c40b9406
(1) 0xece7be76e3aa4d190c043d46dd0c7427a69dbc8f6d58907db516cd69acc7c3e7
(2) 0xb36a0abe3cee02bbe163b85e8ae27ec9388c30facd9cc712553fcfdac7a415ee
(3) 0xe4447edd2e425370ba43fdc352bf473027a285e4210e3e5c145a9202f1cac37b
(4) 0xde16d7bedaebac9f2a1c93210018739f6c351389d084d4f8d66726d1befef4ff7
(5) 0x8591bad67a2a930806527b17ce4fc81b642f39dce511b1e36868d54c7418a4e
(6) 0x33a2ebe55b656b312ef7583da34a3f4a1e63b9011b54e5a13aaa2388e35a4c5e
(7) 0x430a182a98e5ab2bfa49a9e2fae0c7d3e75e8d8ed0d6646ffc926504dd7481d3 A链开启的端口就是854
(8) 0x2d0780184c399d841db4210ff00946ff60969a9b12a1a320e86c912f49d12b42
(9) 0x145843e52fd1f67c5d89f291107bbe713a93b53f3dc7d8d0d092b11a2ad88cb4

HD Wallet
=====
Mnemonic: leg walnut such thunder unhappy argue brass stand venture slice swallow soon
Base HD Path: m/44'/60'/0'/0/{account_index} 装好的ganache-cli，开启一个私链：

Gas Price
=====
200000000000

Gas Limit
=====
>

Listening on 127.0.0.1:8545
```

然后我们要进行编译看看合约是否合法，并生成合约对应的abi等信息。

```
C:\Users\王阔\Desktop\Dweibo>truffle.cmd compile /o'>
C:\Users\王阔\Desktop\Dweibo>
```

因为我已经编译过了，并没有进行修改，所以没有输出什么信息。如果是第一次编译他会告诉你哪里错了，如果没错的话他会提示你生成的各种信息写入了build文件夹。

然后运行truffle.cmd migrate来进行部署。

```
C:\Users\王阔\Desktop\Desweibo>truffle.cmd migrate
Using network 'development'.
Gas Limit
Running migration: 1_initial_migration.js
  Deploying Migrations...
    ... 0x85adecb6a340b8ddb98838a82c87d00729b9e505c28573d6a45e655900358a8b
  Migrations: 0xcee1831a81d5be0182625fc164c2f2ba7aa8acd4
Saving successful migration to network...
  ... 0xda86a73bfd4acc35366c6bdf04bca7636a9e4f9ea4c97e10d47b337c7b48c0
Saving artifacts...
Running migration: 2_deploy_Dweibo.js
  Deploying Dweibo...
    ... 0x30520aab40416c61c508eb25db10304d07518a4b1a69048ce75ed38b5769ca08
  Dweibo: 0xf8c7aa78573994034afb8882bd6b7a71fa93d02d
Saving successful migration to network...
  ... 0x09368bd9d06584239244876a5755b9594d8c13f3f247f9a7a328b8e7c12c485b
Saving artifacts...因为我已经编译过了，并没有进行修改，所以没有输出什么
```

部署完成之后，我们的ganache-cli命令行上可以看到相关的部署信息：

```
eth_sendTransaction
  Transaction: 0x30520aab40416c61c508eb25db10304d07518a4b1a69048ce75ed38b5769ca08
  Contract created: 0xf8c7aa78573994034afb8882bd6b7a71fa93d02d
  Gas usage: 3121286
  Block Number: 3
  Block Time: Sat Dec 29 2018 09:05:25 GMT+0800 (GMT+08:00)

  Saving artifacts...
  Running migration: 2_deploy_Dweibo.js
  Deploying Dweibo...
  Dweibo: 0x18c7aa78573994034afb8882bd6b7a71fa93d02d
  Saving successful migration to network...
  0x09368bd9d06584239244876a5755b9594d8c13f3f247f9a7a328b8e7c12c485b
  Saving artifacts...

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt      部署完成之后，我们的ganache-cli命令行上可以看到相关的部分
eth_getCode
eth_uninstallFilter
eth_sendTransaction

  Transaction: 0x09368bd9d06584239244876a5755b9594d8c13f3f247f9a7a328b8e7c12c485b
  Gas usage: 27008
  Block Number: 4
  Block Time: Sat Dec 29 2018 09:05:26 GMT+0800 (GMT+08:00)

  eth_getTransactionReceipt
```

我们看到Contract created字段，后面就是我们的合约所在的地址啦。需要将这个地址保存下来拿来连接合约。

然后进入truffle的工作目录，找到build文件夹，找到Dweibo.json文件，里面有编译Dweibo.sol之后生成的各种信息，我们可以从这个文件中获取到生成的abi等等信息。我们需要用的就是abi。

拿到了合约的abi和地址，我们就可以连接我们的合约了！

连接合约的代码如下：

```
var Web3 = require("web3");
var web3 = new Web3();
web3.setProvider(new Web3.providers.HttpProvider("http://127.0.0.1:8545"));
var abi = [{"anonymous":false,"inputs":[{"indexed":false,"name":"owner","type":"address"}, {"indexed":false,"name":"value","type":"uint256"}], "name": "Transfer", "outputs": [{"indexed": false, "name": "from", "type": "address"}, {"indexed": false, "name": "to", "type": "address"}, {"indexed": false, "name": "value", "type": "uint256"}], "payable": false, "stateMutability": "nonpayable", "type": "event"}, {"name": "Dweibo", "methods": [{"constant": true, "inputs": [{"name": "owner", "type": "address"}], "name": "balanceOf", "outputs": [{"name": "balance", "type": "uint256"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": true, "inputs": [{"name": "from", "type": "address"}, {"name": "to", "type": "address"}, {"name": "value", "type": "uint256"}], "name": "allowance", "outputs": [{"name": "remaining", "type": "uint256"}], "payable": false, "stateMutability": "view", "type": "function"}, {"constant": false, "inputs": [{"name": "to", "type": "address"}, {"name": "value", "type": "uint256"}], "name": "transfer", "outputs": [{"name": "txHash", "type": "string"}], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs": [{"name": "from", "type": "address"}, {"name": "to", "type": "address"}, {"name": "value", "type": "uint256"}], "name": "transferFrom", "outputs": [{"name": "txHash", "type": "string"}], "payable": false, "stateMutability": "nonpayable", "type": "function"}, {"constant": false, "inputs": [{"name": "owner", "type": "address"}, {"name": "spender", "type": "address"}, {"name": "value", "type": "uint256"}], "name": "approve", "outputs": [{"name": "txHash", "type": "string"}], "payable": false, "stateMutability": "nonpayable", "type": "function"}], "network": "mainnet", "owner": "0xf8c7aa78573994034afbf8882bd6b7a71fa93d02d", "symbol": "DWEIBO", "token": true, "version": "1.0.0"}];
```

使用node.js能够连接我们的合约之后，开发就算正式开始了！然后就是漫长的前端开发之路。

我这里使用的服务器框架是express，入门级的包。

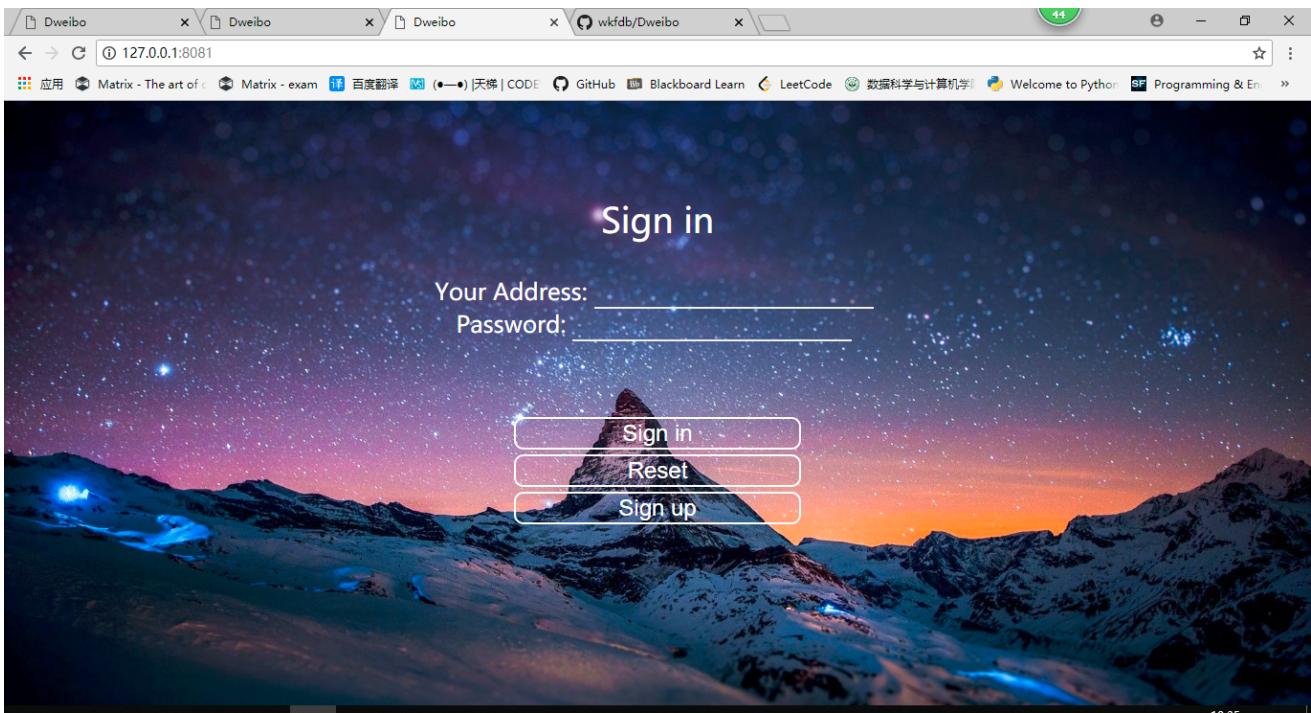
运行

这里要再次说一遍，我的私链环境是ganache-cli，部署方式为使用truffle部署，使用web3进行合约的连接，服务器框架为express.

如果上面的部分全都没有问题的话，接下来只需要运行 `node express_demo.js` 即可。运行之后我们可以看到：

```
C:\Users\王阔\Desktop\express>node express_demo.js  
应用实例，访问地址为 http://localhost:8081
```

然后打开浏览器访问localhost:8081:

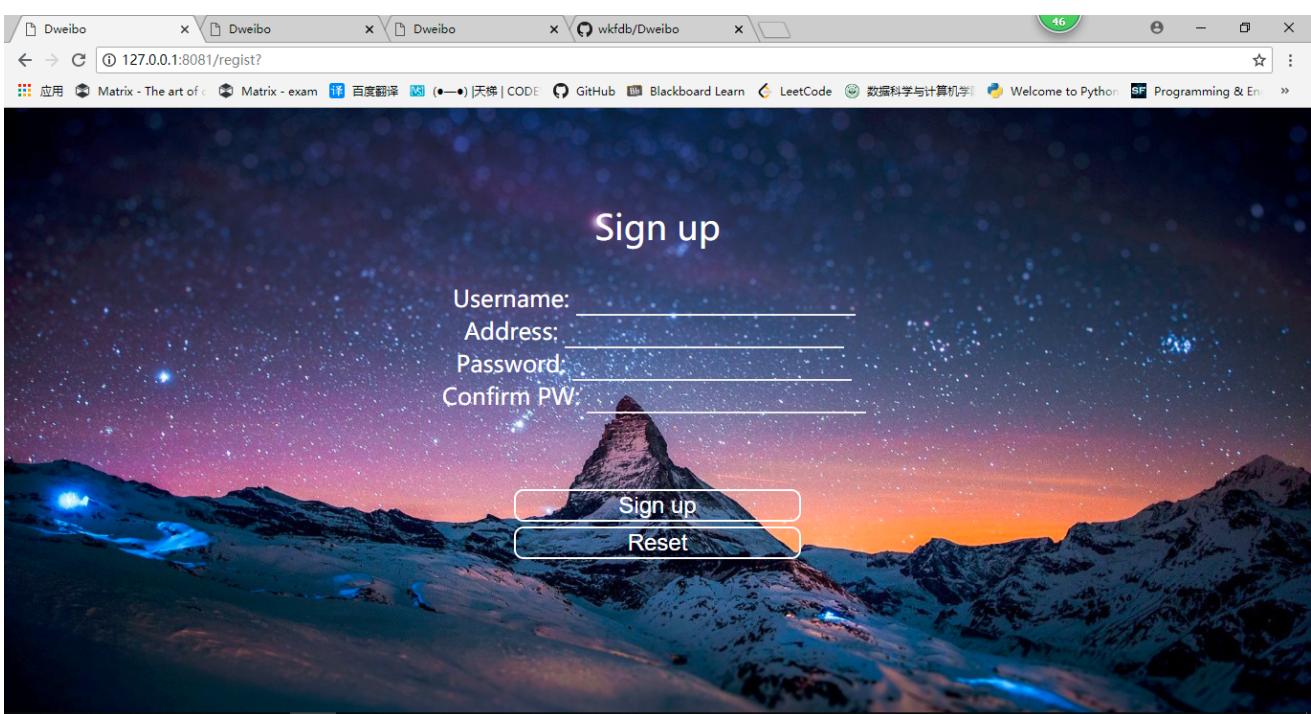


Dweibo就跑起来啦！

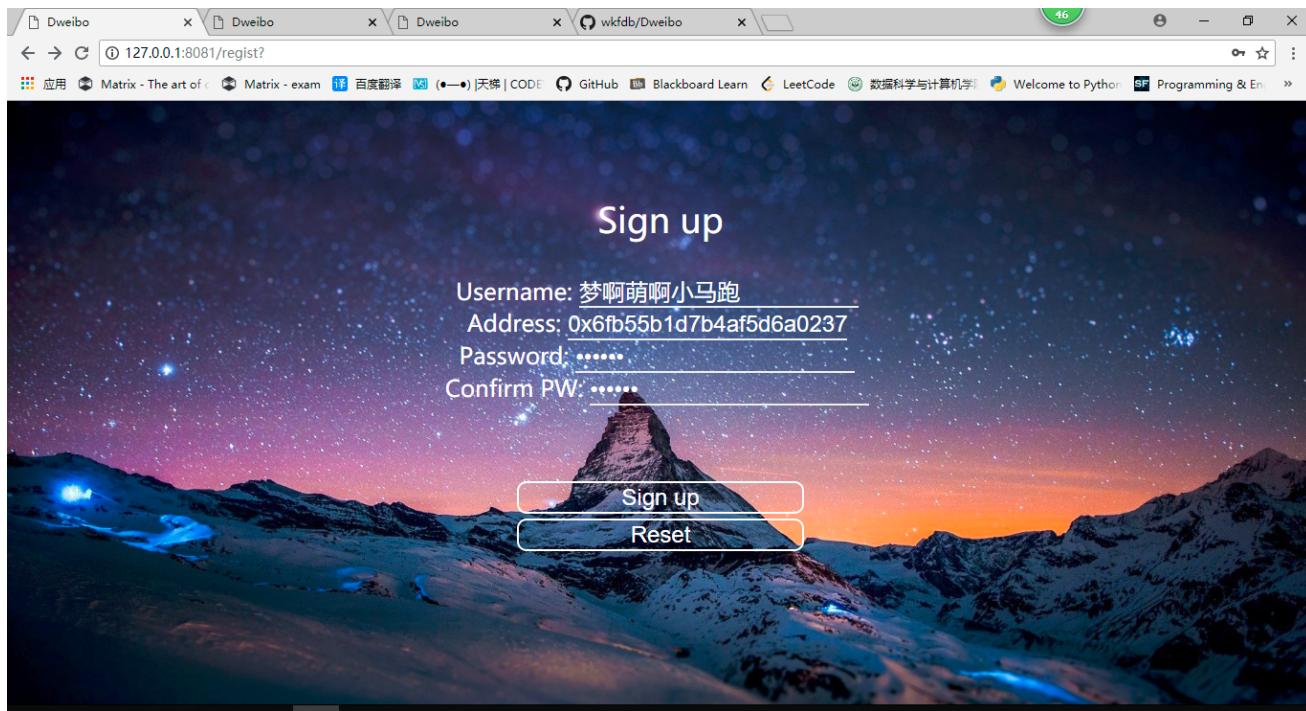
Dapp的运行效果

首先由于我们还没有账户，所以要先创建账户。这里要求用户本身就有自己的以太坊账户并且有一定的以太币，不然的话无法在合约里面保存自己的信息。注册是使用用户的以太坊地址外加一个密码来实现。

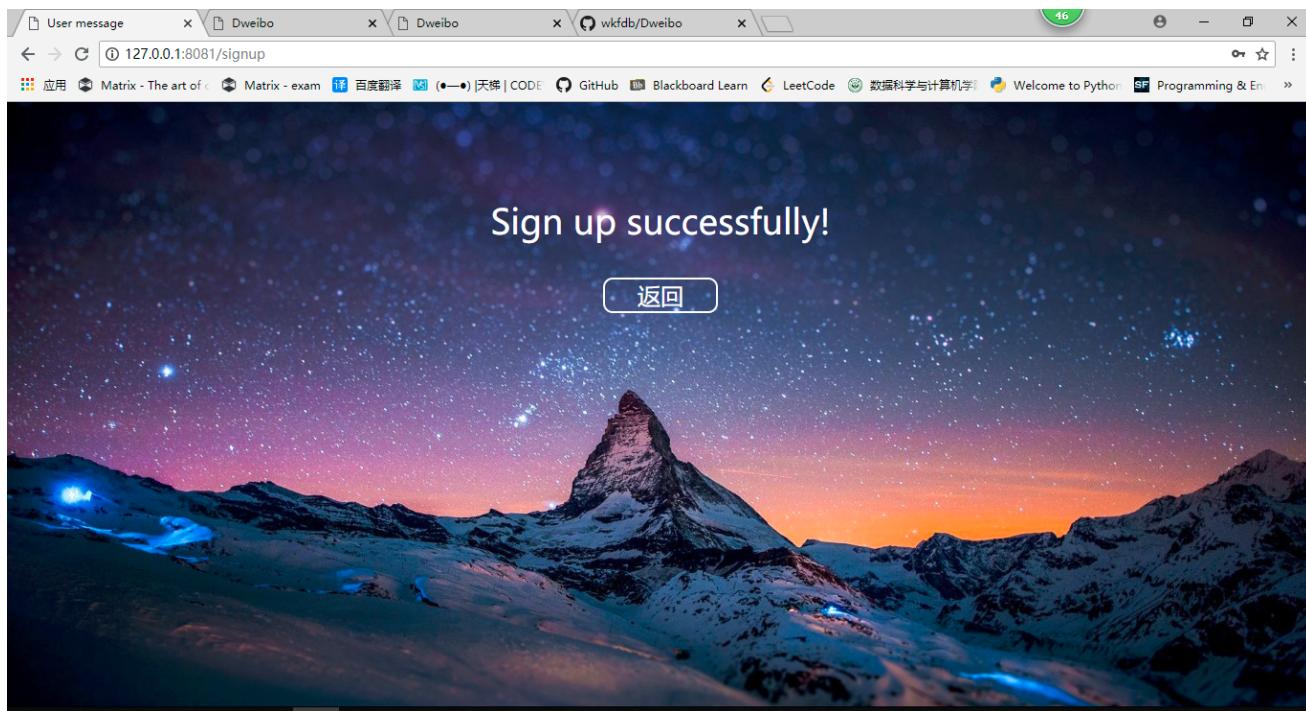
点击sign up进入注册页面：



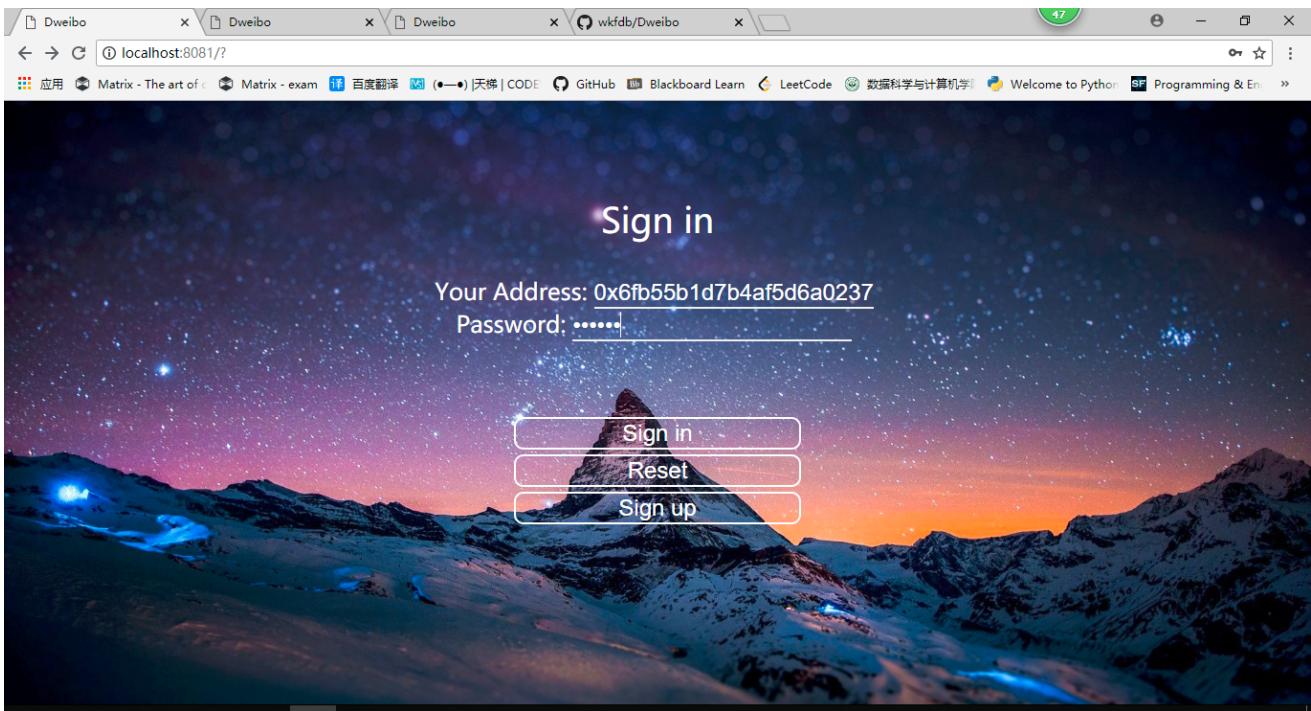
输入你的用户名，地址，密码，注意两次输入的密码要相同：



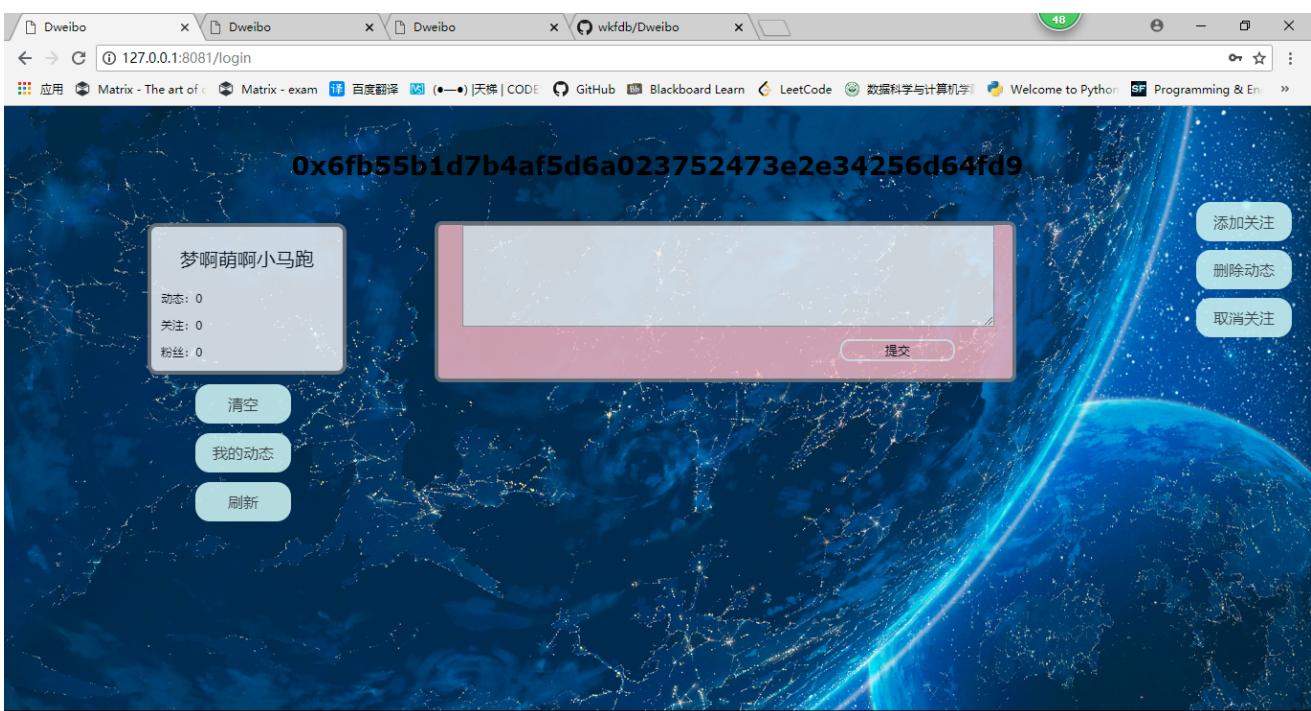
然后点击sign up，注册成功的话会告诉你注册成功：



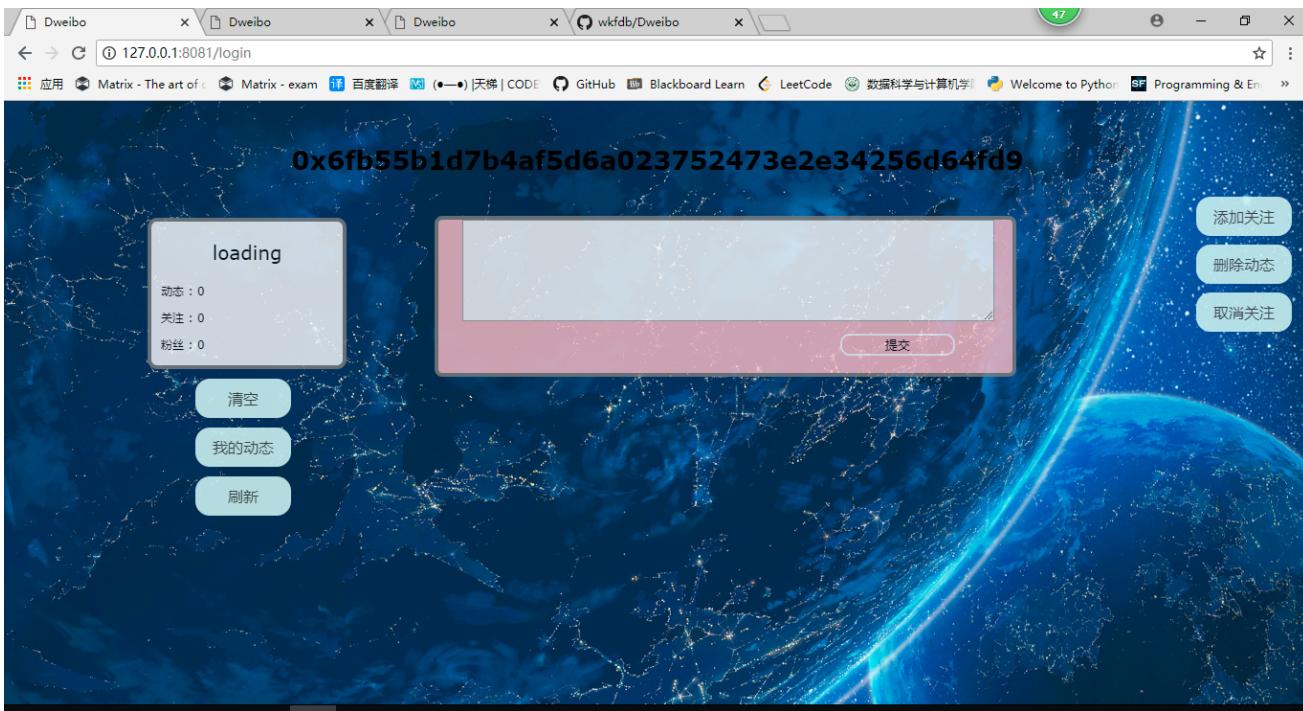
点击返回按钮进入登录页面：



点击sign in进入主页：

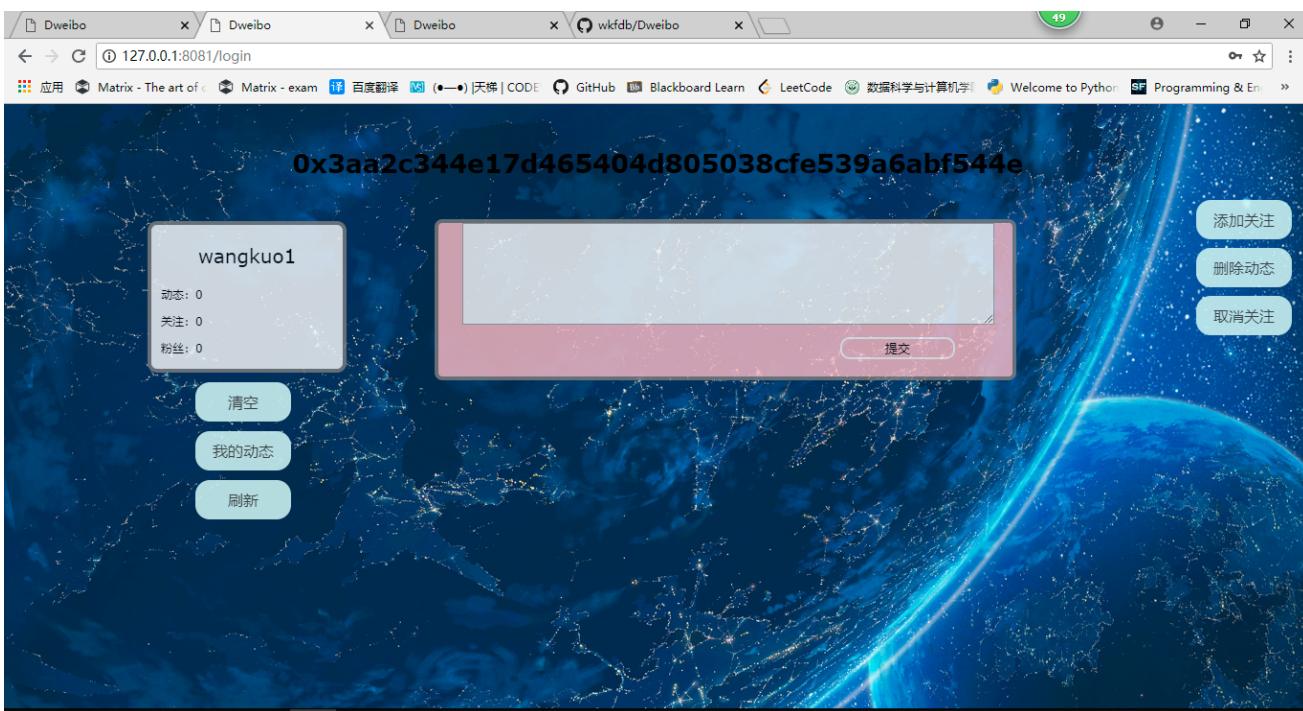


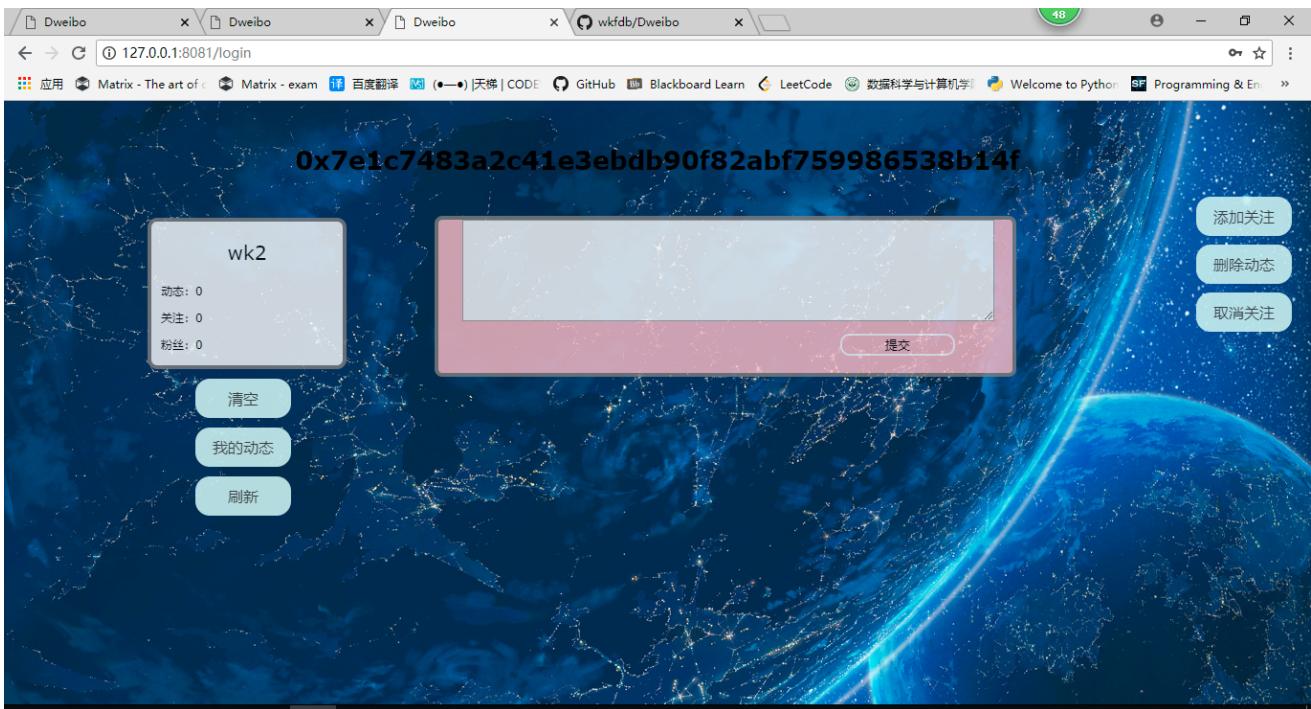
你的账户地址会显示在合约的最上方。然后用户信息是根据你的账户地址来获取得到的。刚进入页面的效果其实是这样的：



会显示loading，表示正在从区块链上拉取你的个人信息。

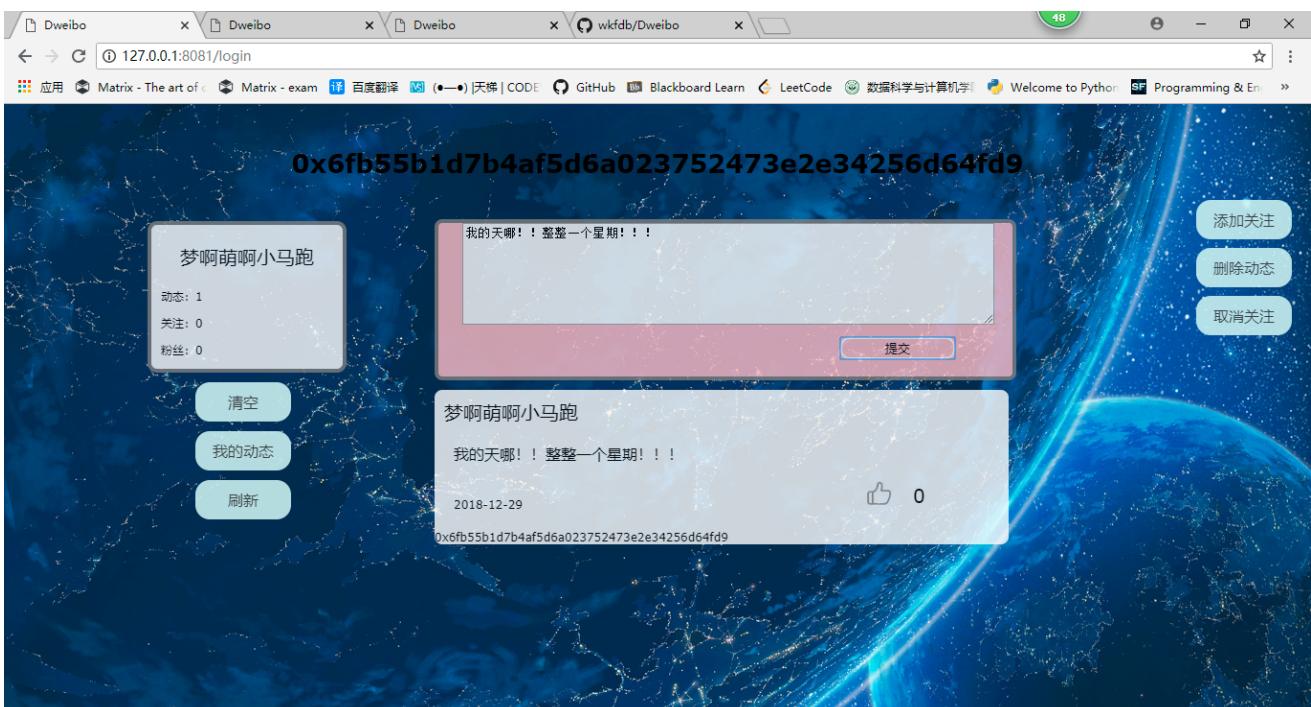
然后让我们再新建几个用户登录进来。





这里又新建了两个用户。分别叫wangkuo1和wk2。

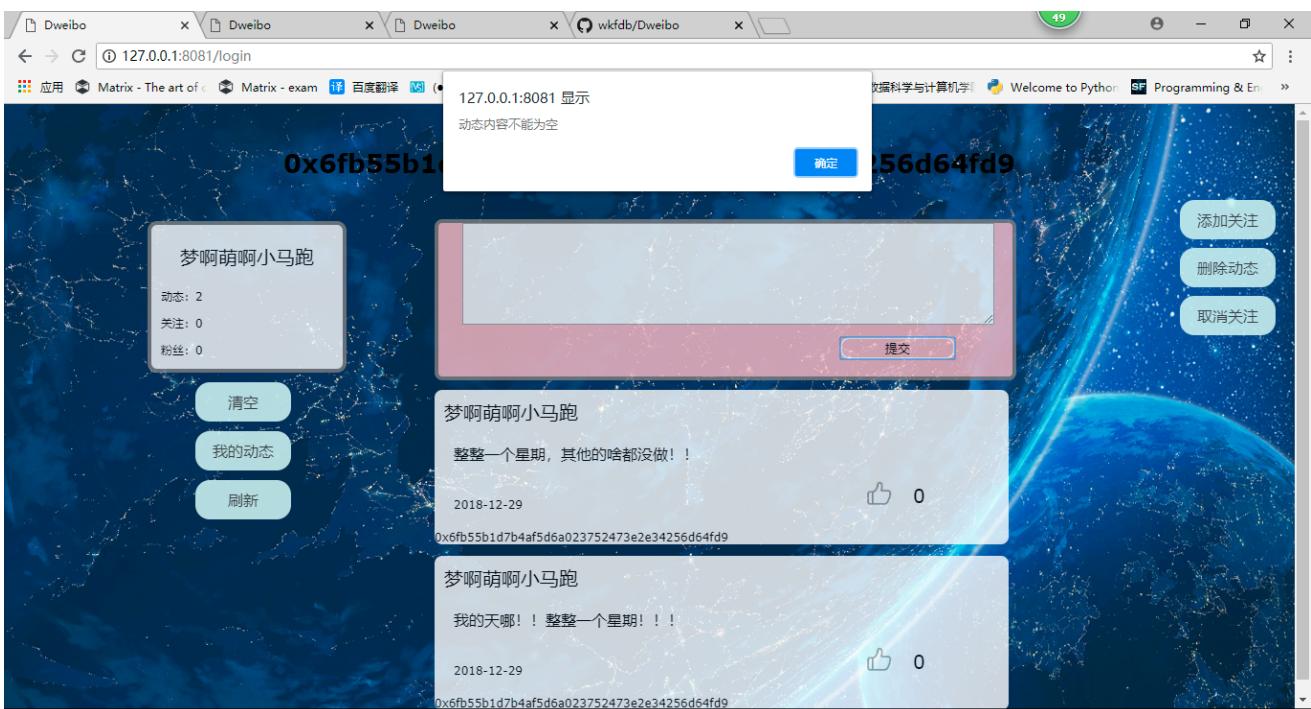
发布动态:在输入框里填入你想发布的内容，点击提交即可。然后动态就会出现在下方：



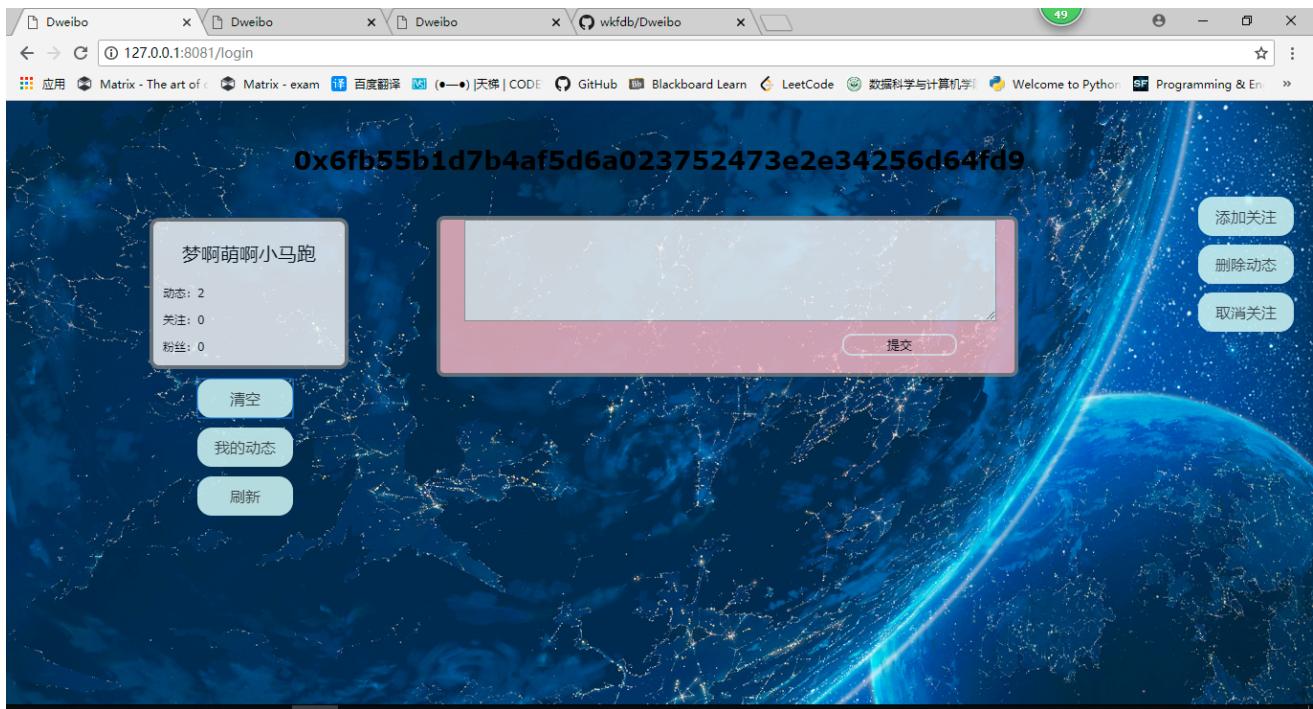
再次提交可以继续添加动态。每次添加动态之后个人信息都会相应地更新。添加了两条动态之后我们已经看到个人信息的动态那里已经变成了2.



动态发布的内容不能为空：



点击清空按钮会在页面上清空你的动态消息。

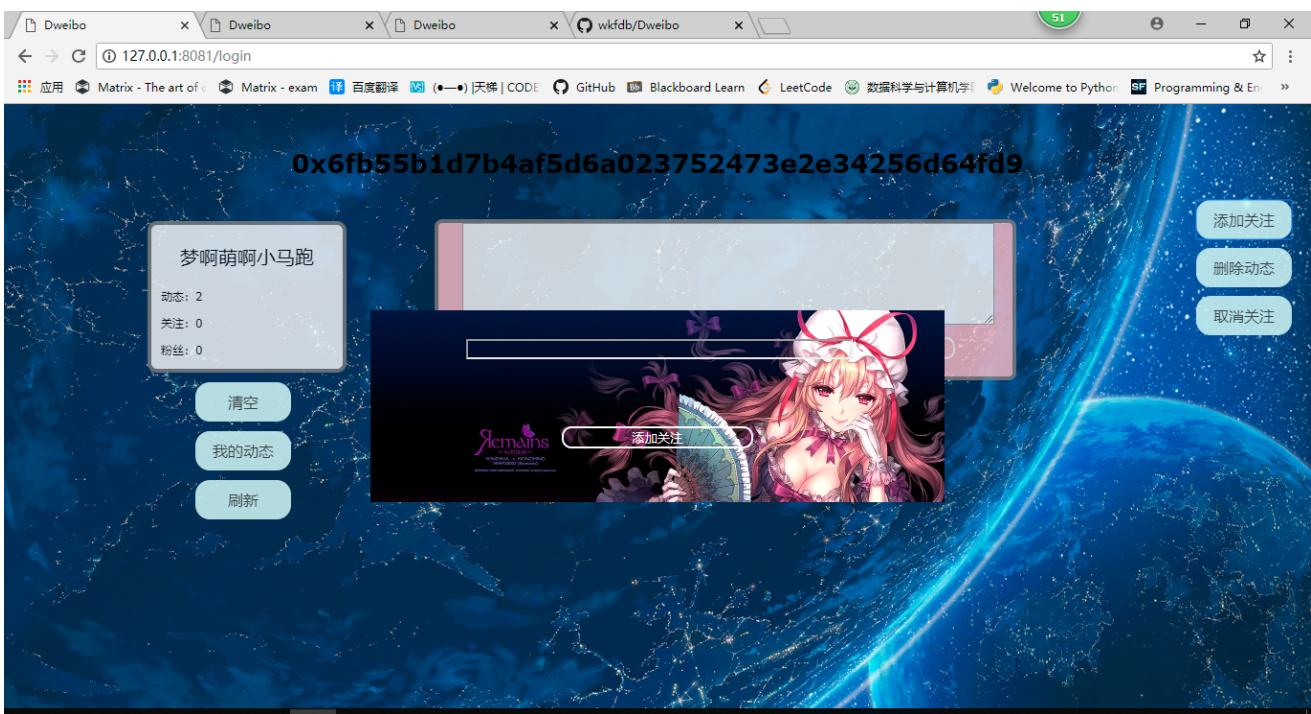


然后让其他两个用户也发布一下动态。

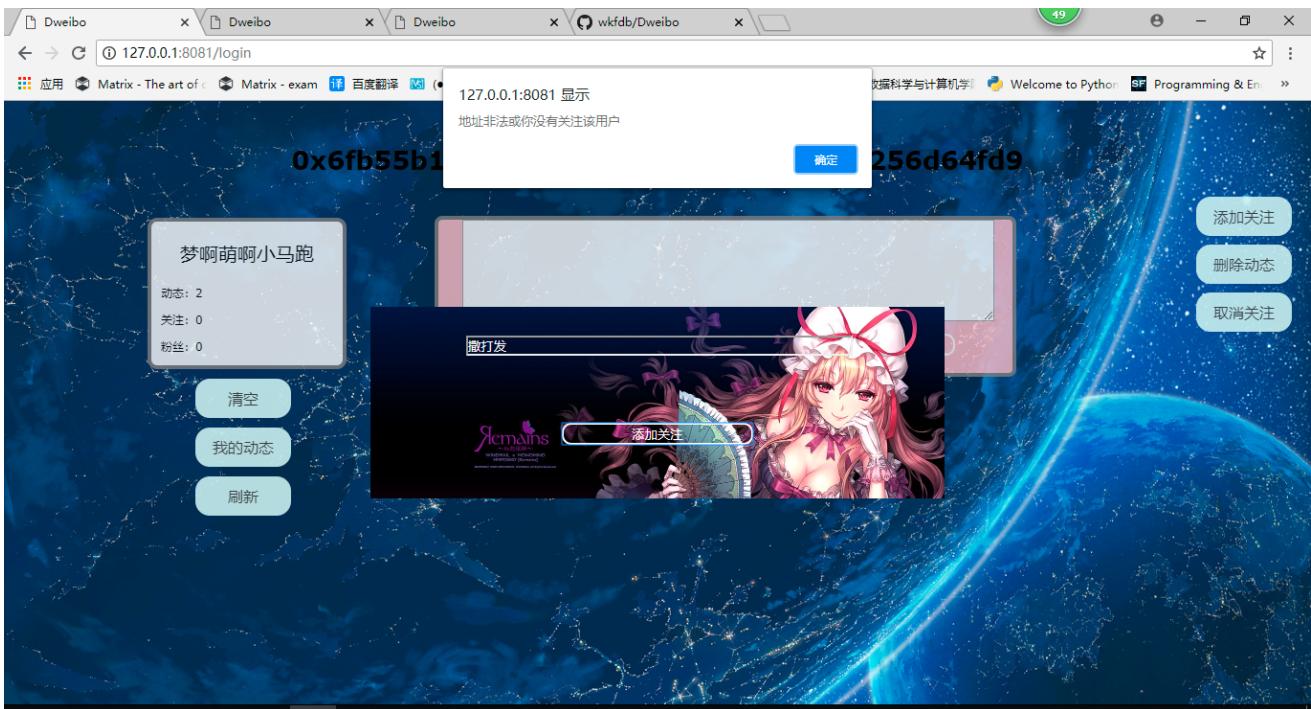




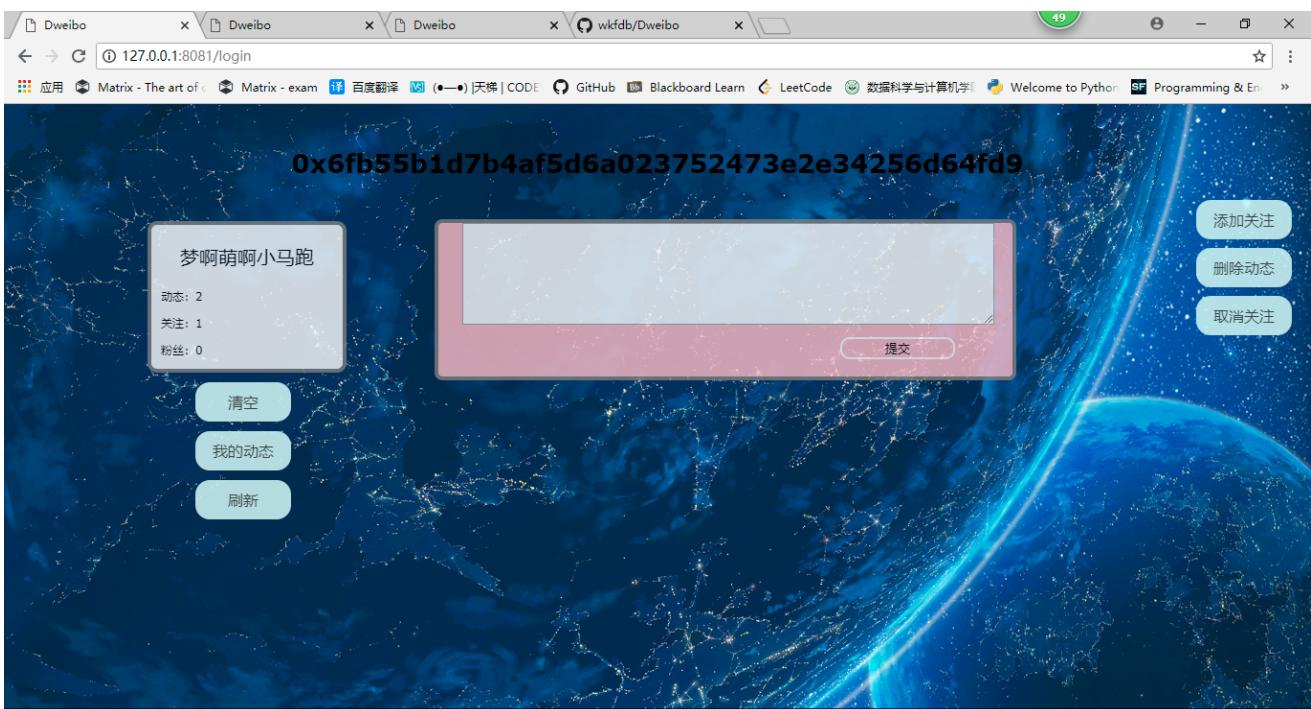
接下来使用一下我们的关注功能，点击添加关注按钮，弹出一个框框：



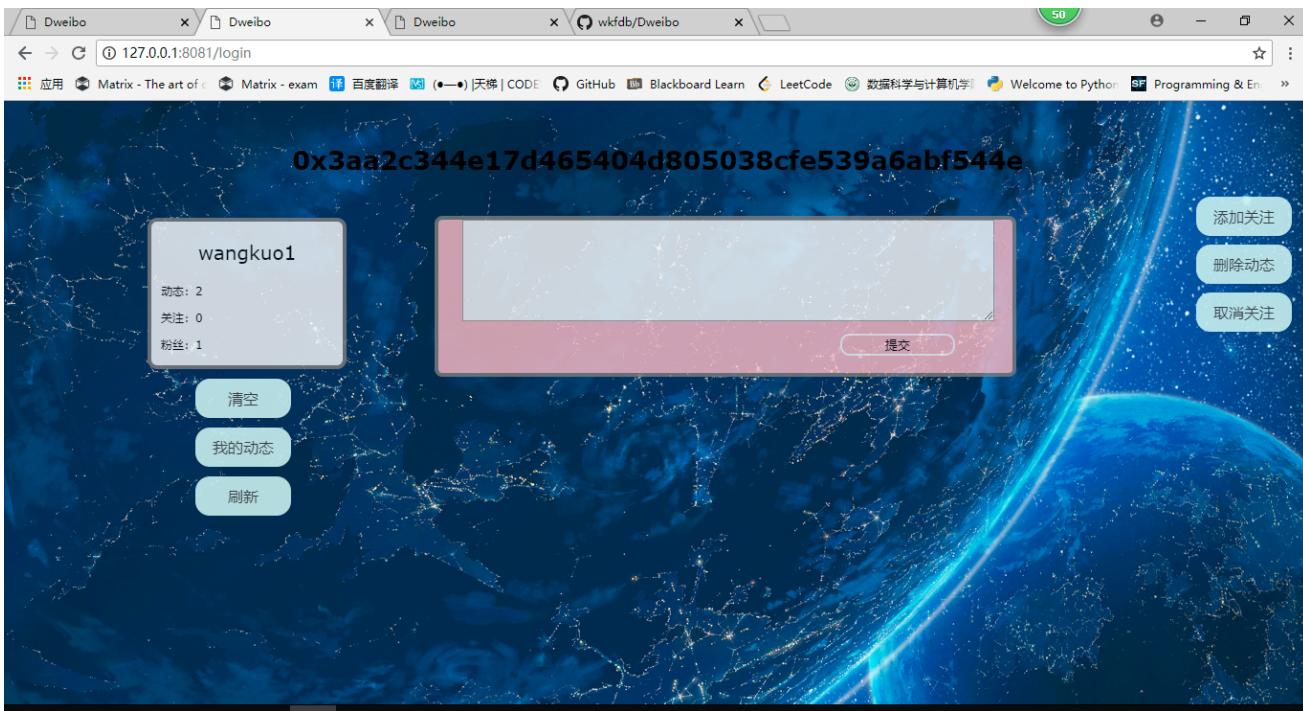
在框框里输入一个地址，如果该地址非法获取该地址没有注册Dweibo的话会输出报错信息：



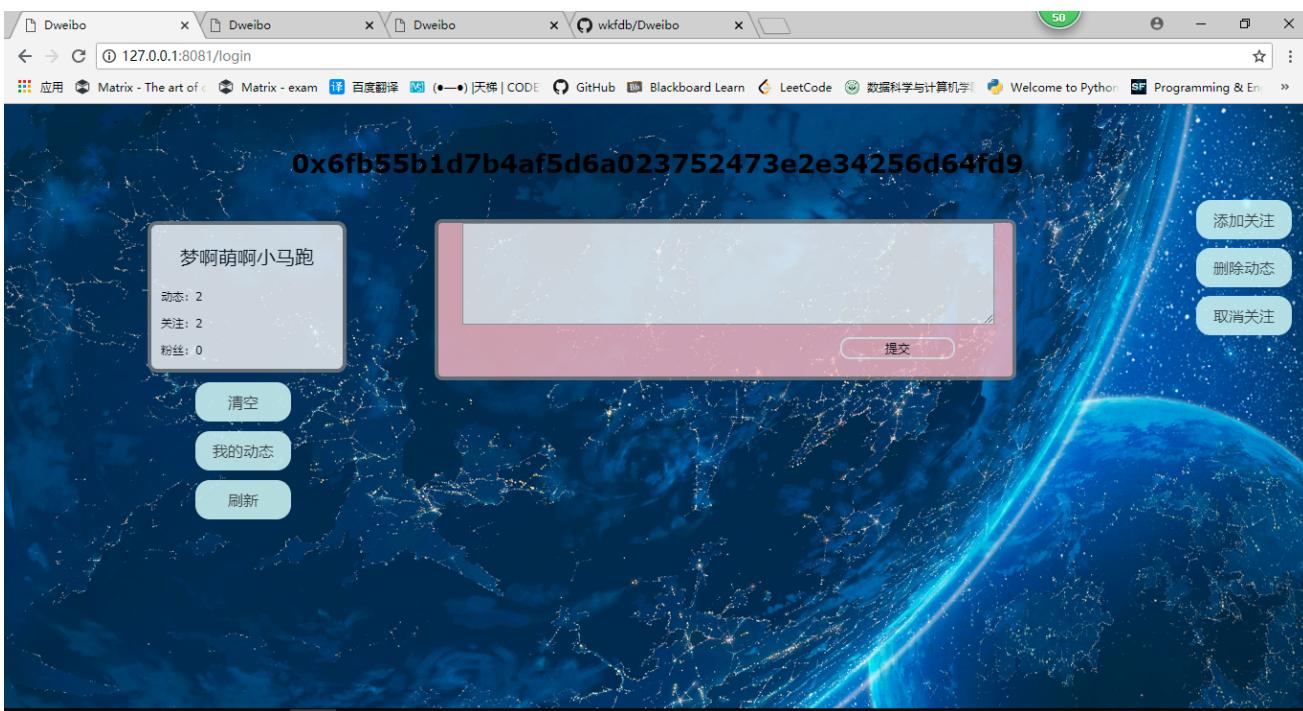
然后填入一个正常的地址后，你的关注数会相应的更新：



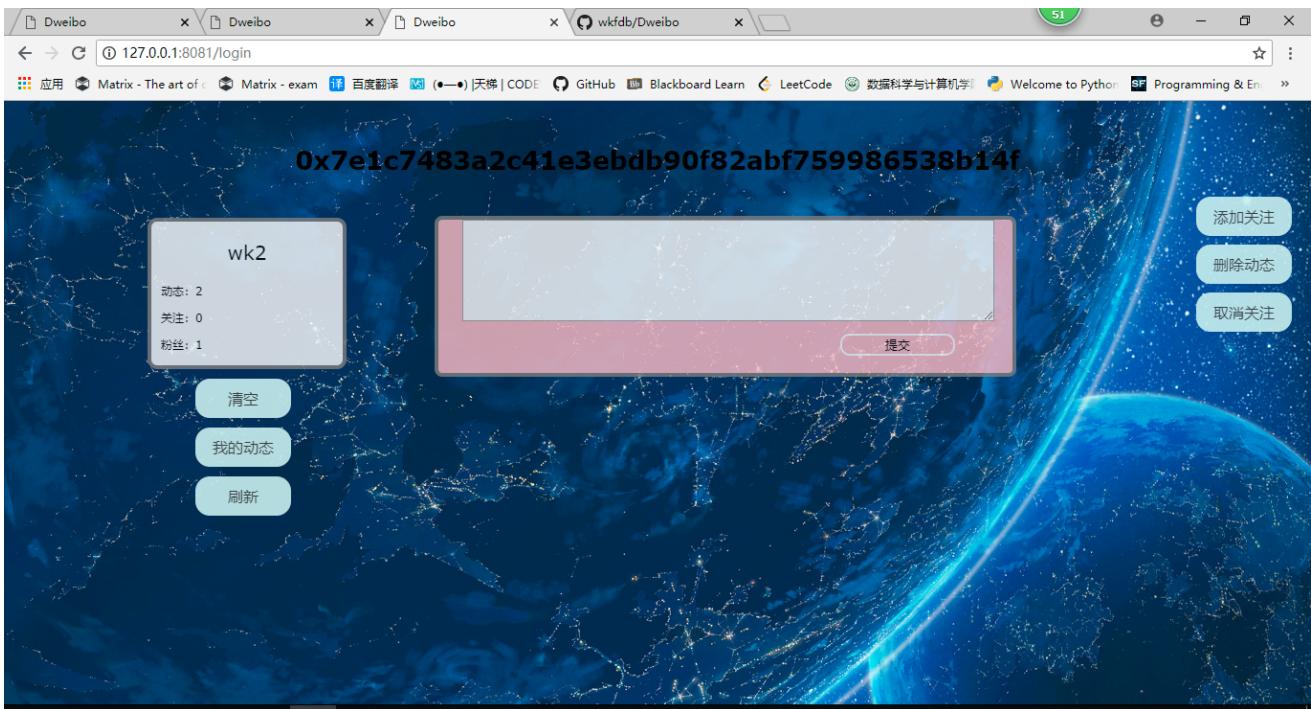
然后被关注的人粉丝数也会相应地改变：



wangkuo1用户的粉丝数变成了1.然后我们可以接续添加关注，不妨让“梦啊萌啊小马跑”来关注更多的人：



wk2用户也喜提粉丝一名：



关注了别人之后，点击刷新按钮，就可以看到你关注的人的动态了！



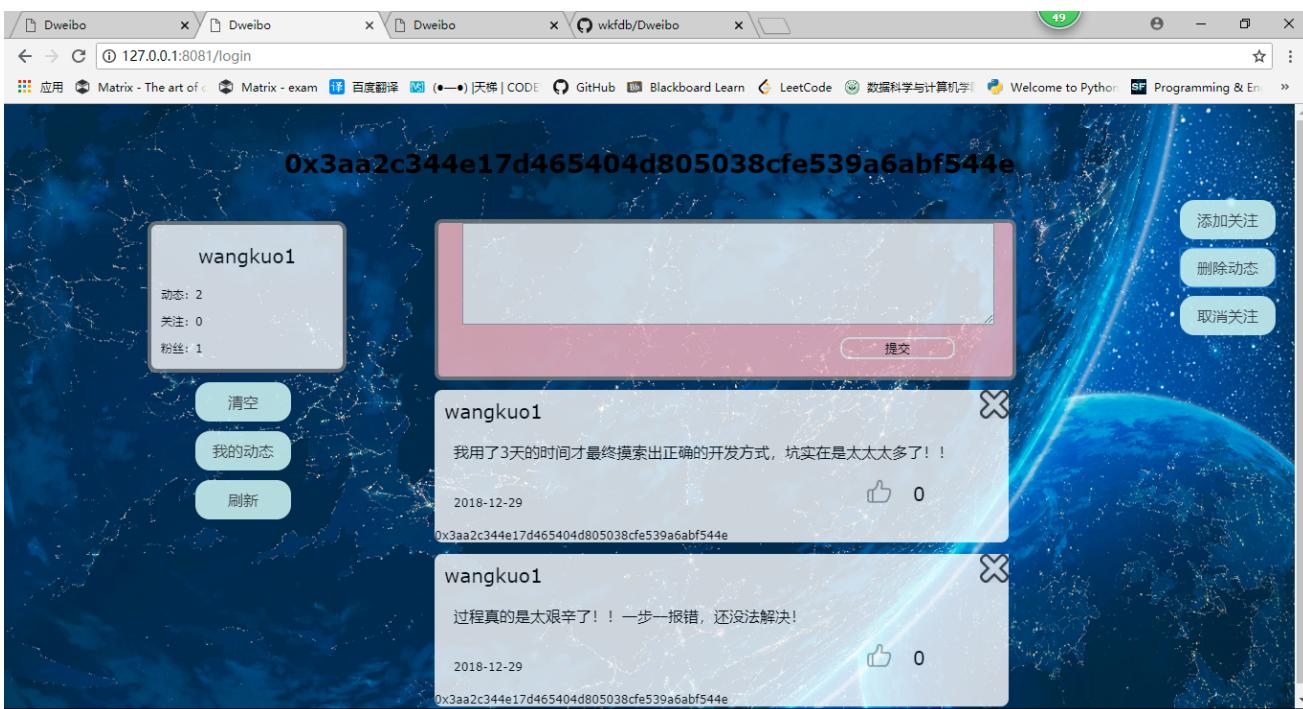
欸，别人发的动态有点多，一个屏幕显示不过来。

除了看别人的动态，自己当然还能看自己的动态：点击我的动态按钮，显示自己的动态：

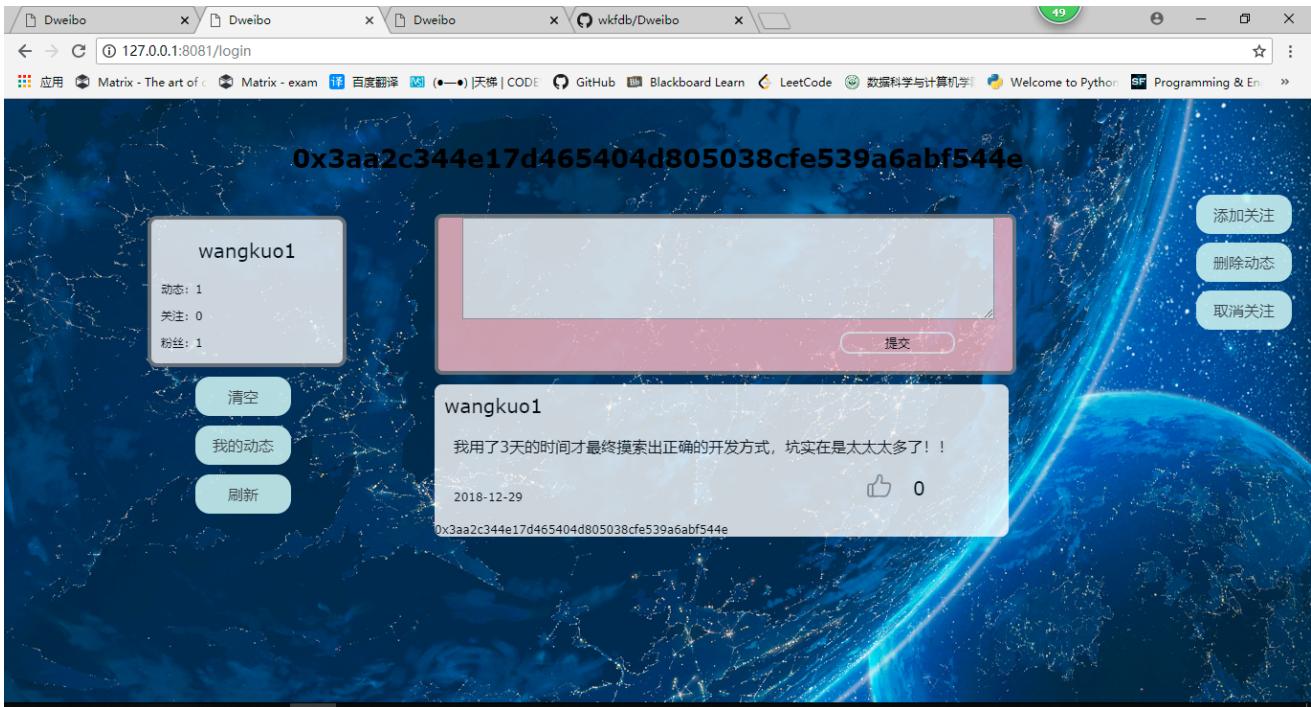


然后就是我们最最核心的部分了！删除动态！在Dweibo上，能够删除自己动态的人，只有你自己！

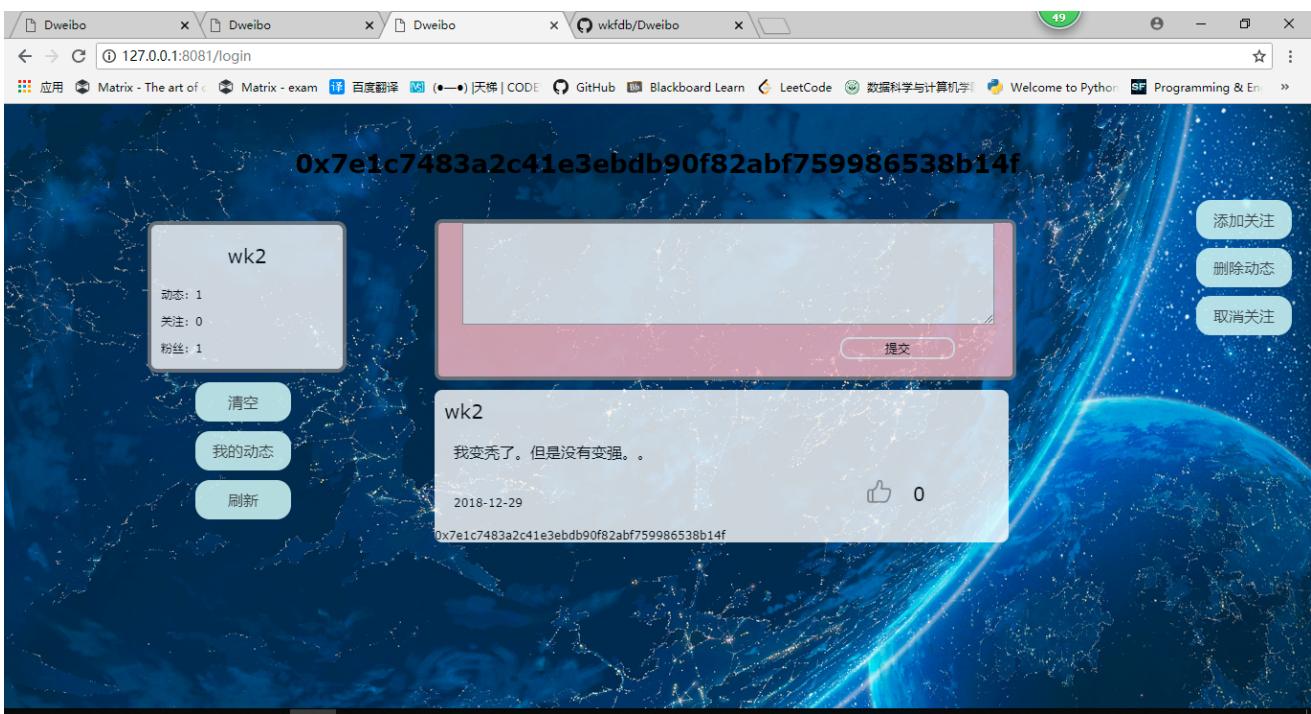
我们让wangkuo1用户来删除一下自己的动态。点击我的动态，先显示出自己的动态，然后点击右边的删除动态按钮，每条动态右上角都出现了一个叉：



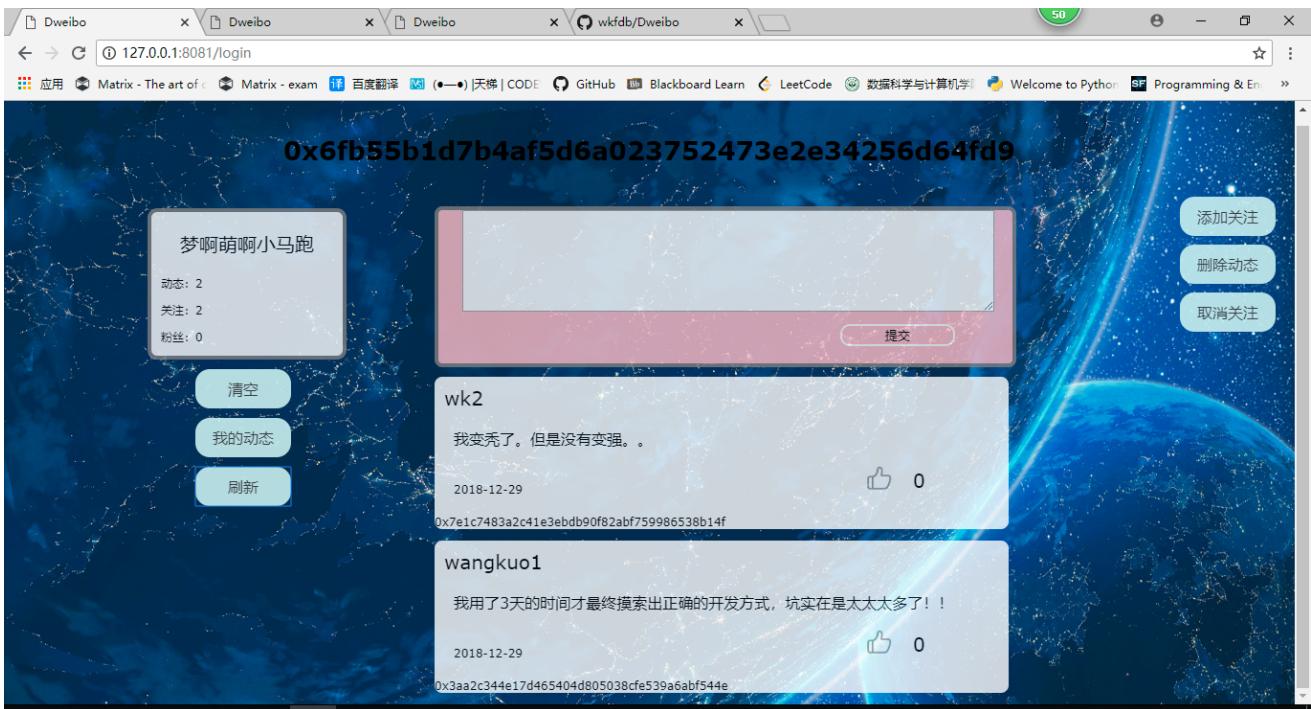
点击要删除的动态即可删除：删除之后动态数减1，动态显示页面重新显示自己的动态。



我们让wk2用户也删除一下自己的动态：

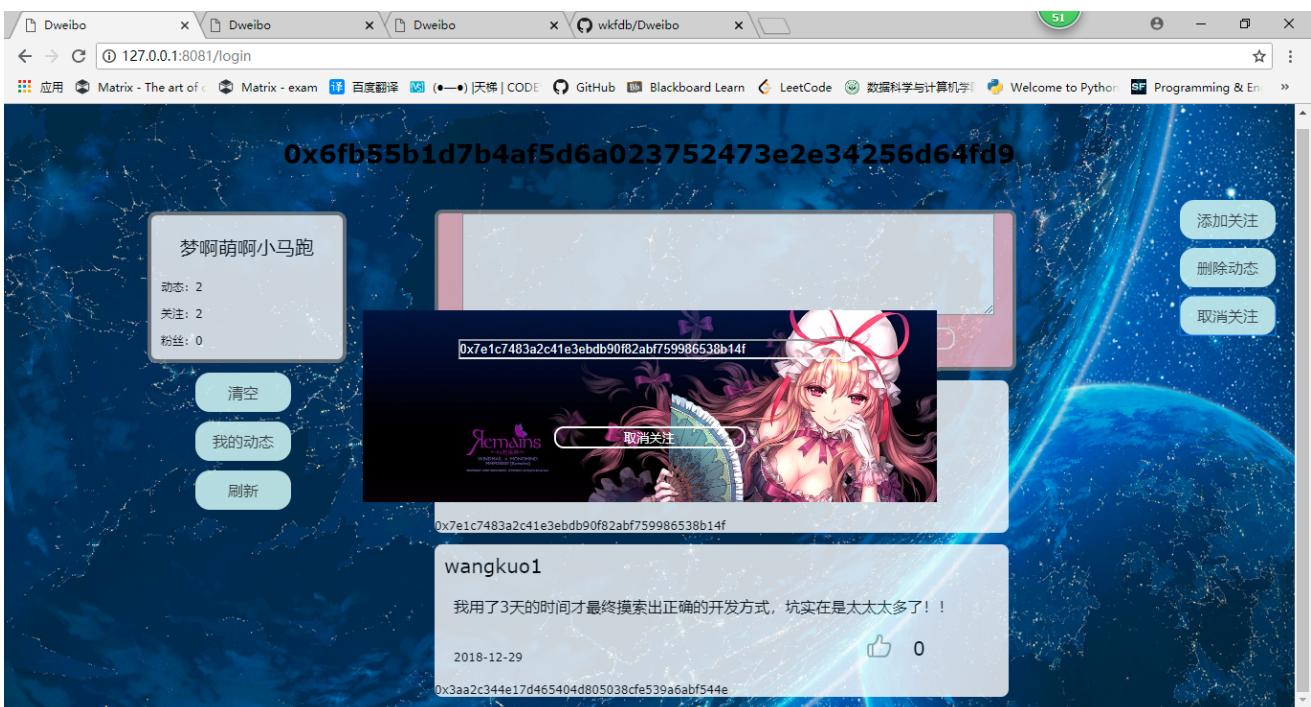


删除了自己的动态之后，其他人也就无法获取被删掉的动态了。回到“梦啊萌啊小马跑”页面，点击刷新：

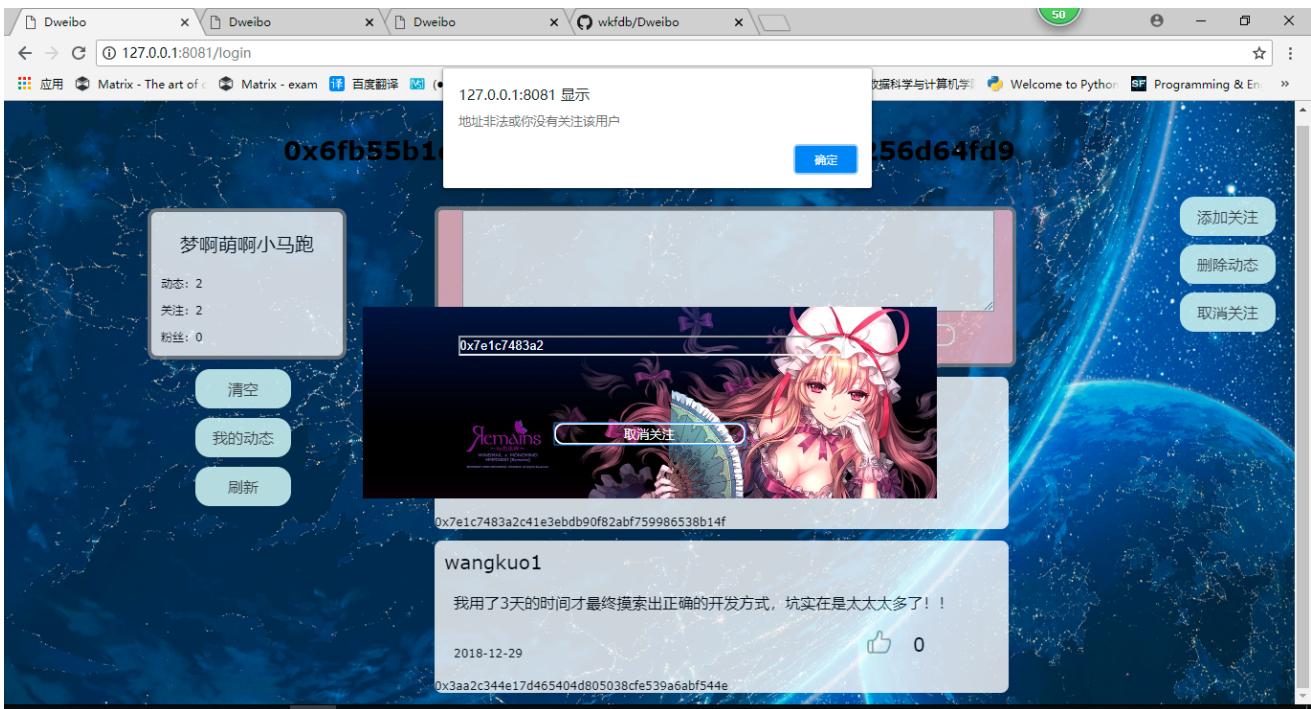


可以发现被删掉的动态已经彻底被删掉了。

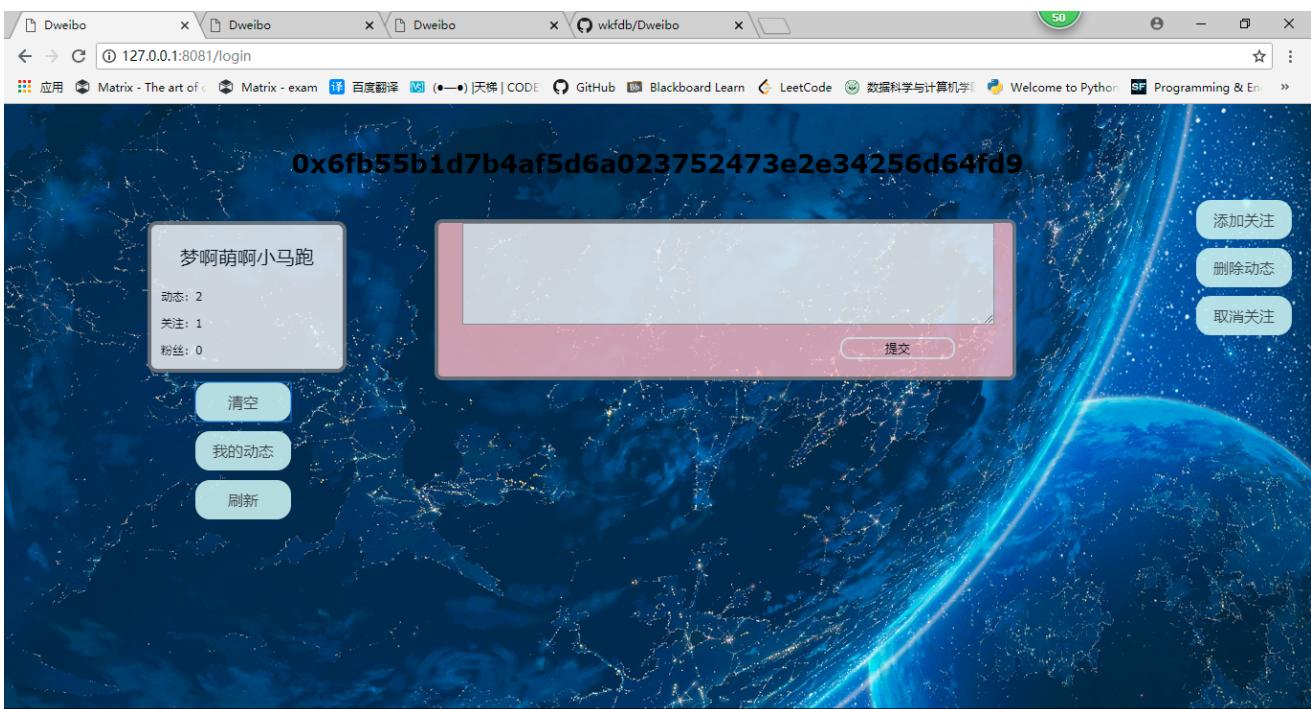
那么有时候我们不喜欢这个人了，我们就可以取消关注。比如我不喜欢wk2了，我就可以不再关注他，点击取消关注按钮：

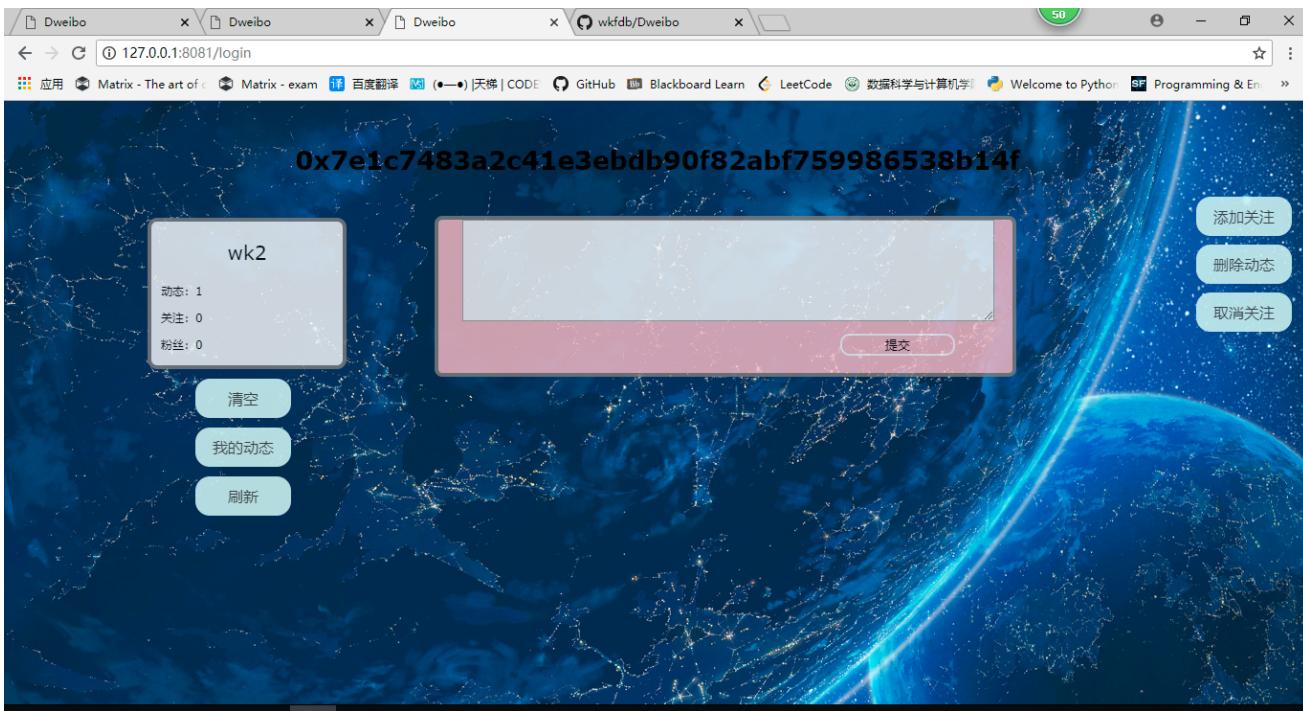


点击取消关注，当然如果有错的话还是会报错：



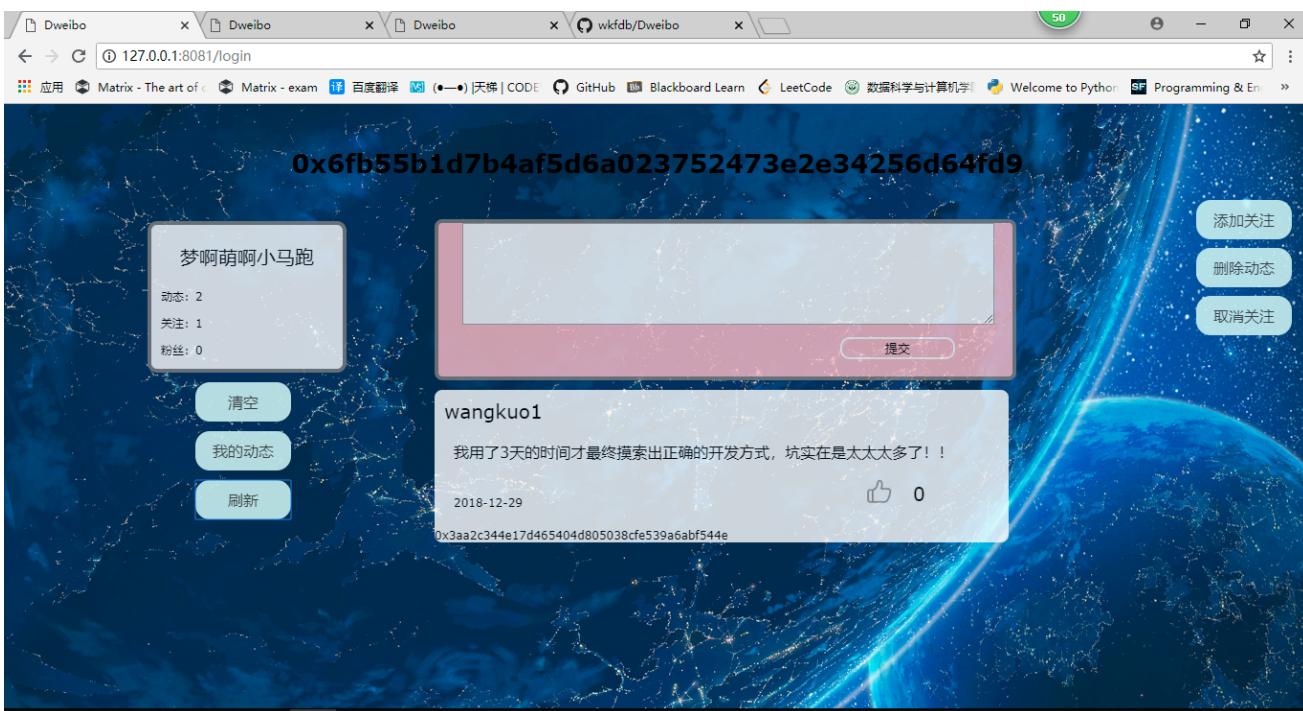
然后取消之后，关注数减少，对应的人那里粉丝数也会降低：



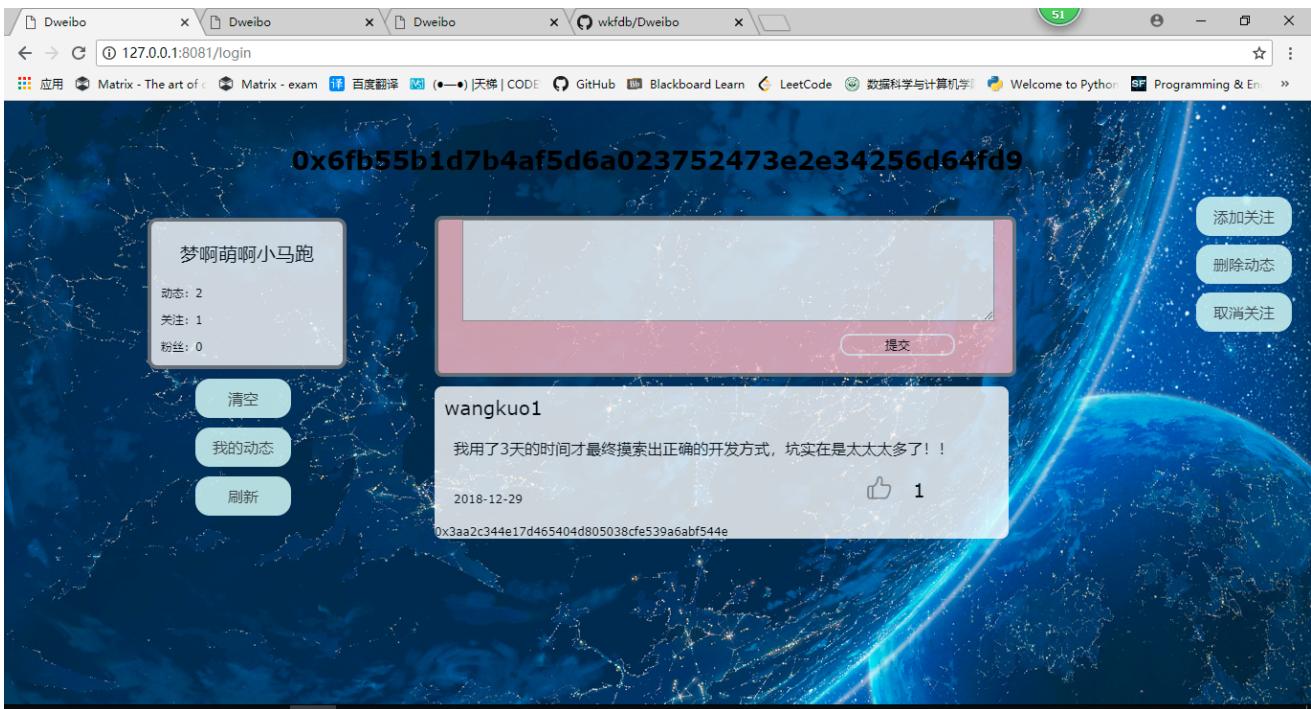


wk2的粉丝没有了！！

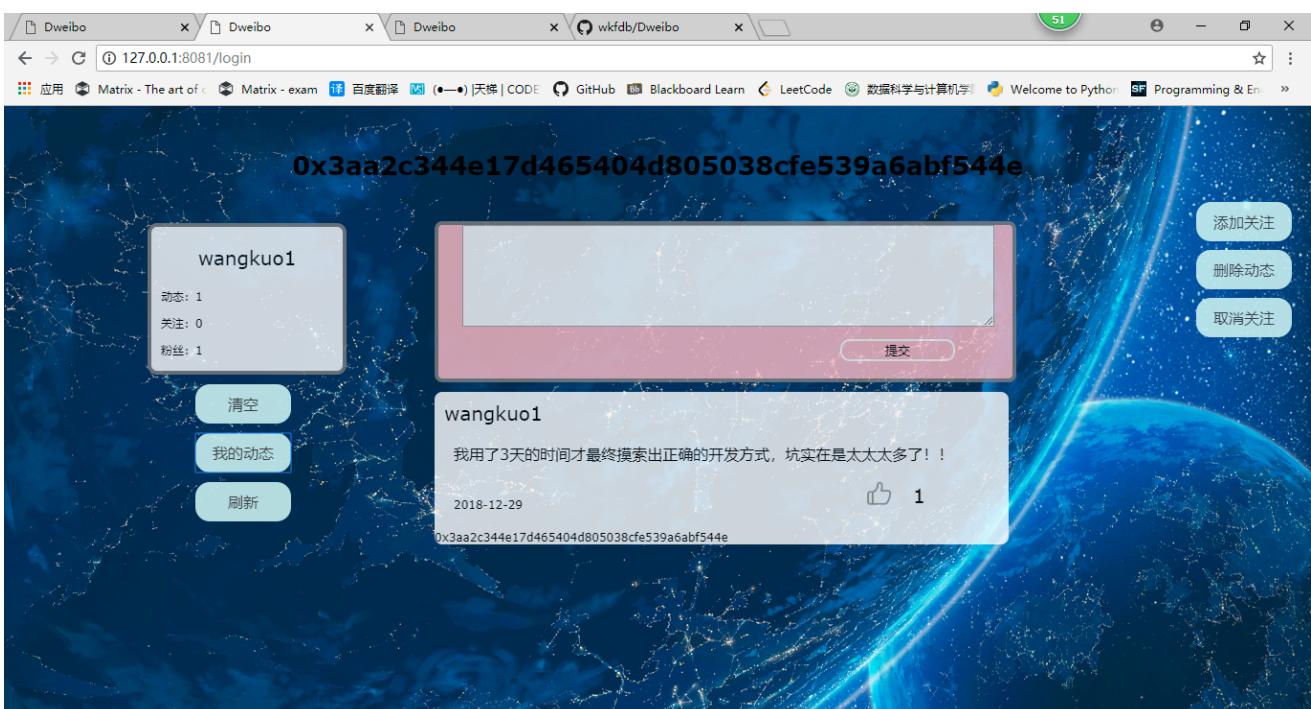
我们重新点击刷新，会发现再也无法获取wk2的动态：



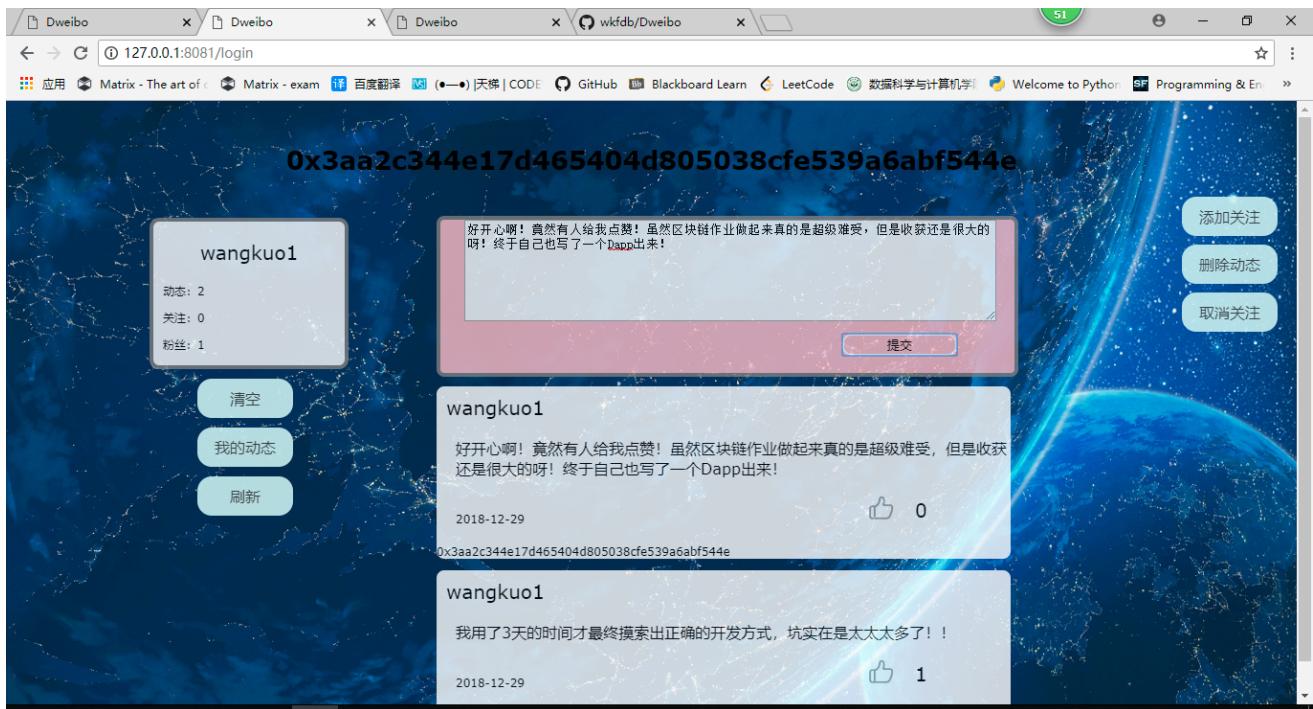
我们还可以点赞！！但是自己不能赞自己。给自己点赞点了也没用，但是你可以给别人点赞：



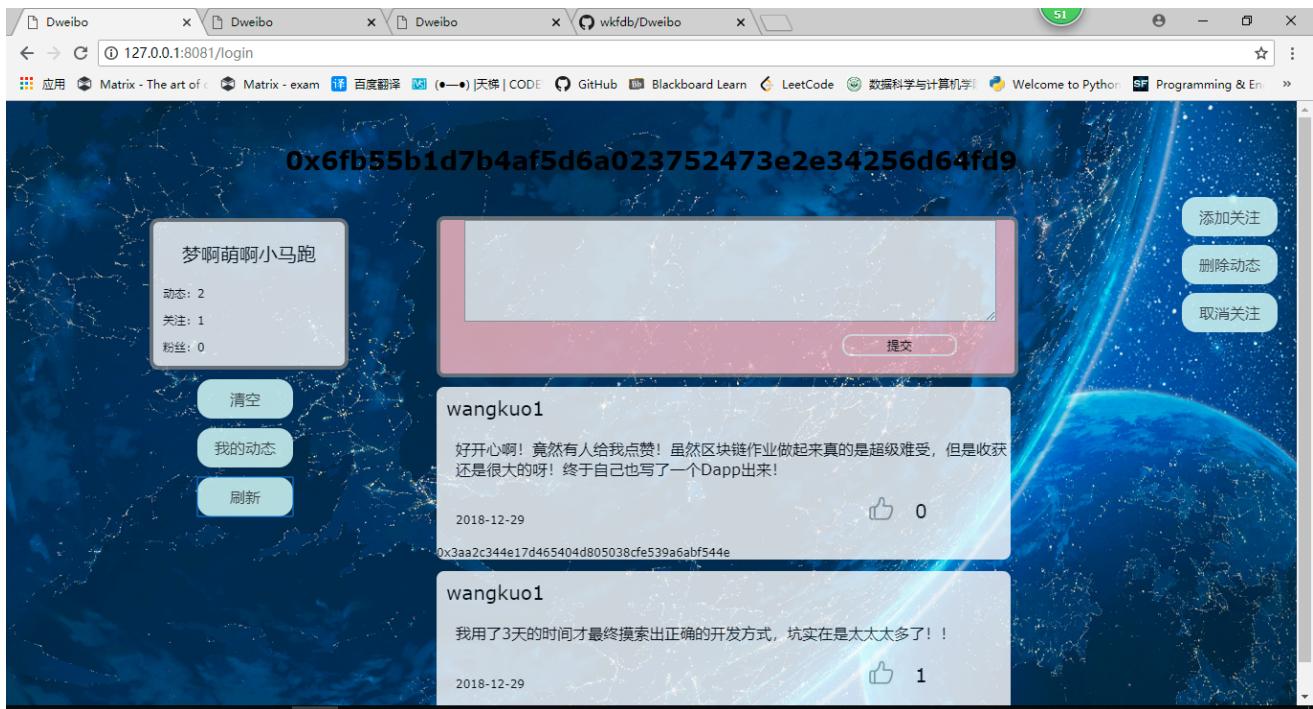
看wangkuo1那么辛苦，我们给他点个赞！然后wangkuo1重新查看自己的动态，惊奇的发现：有人给自己点赞了！



wk1表示十分开心，并重新发布了一条动态：



梦啊萌啊小马跑刷刷刷，看到了wangkuo1发布的新动态，表示十分欣慰。



看一下私链环境那里：

```
命令提示符 - ganache-cli> se_demo.js
eth_call144a17d465404d805038cf539a6abf544e
eth_sendTransaction b90f82abf759986538b14f
BigNumber [ s: 1, e: 0, c: [ 2 ] ]
  Transaction: 0xcb5dd90c667ef6a440731df3ef8abc0346c47200f0e9b29ad3e43d10ea943e5c
  Gas usage: 46761
  Block Number: 19
  BlockTime: Sat Dec 29 2018 11:18:00 GMT+0800 (GMT+08:00)
BigNumber [ s: 1, e: 0, c: [ 2 ] ]
eth_call1
eth_call1c344e17d465404d805038cf539a6abf544e
eth_call1
eth_call17483a2c41e3ebdb90f82abf759986538b14f
eth_call183a2c41e3ebdb90f82abf759986538b14f
eth_call1r [ s: 1, e: 0, c: [ 2 ] ]
eth_call1
eth_call1c344e17d465404d805038cf539a6abf544e
eth_call1
eth_sendTransaction 00000000000000000000000000000000
BigNumber [ s: 1, e: 0, c: [ 2 ] ]
  Transaction: 0x5a1d65a6c5926f3c0236584f6422fb485dc0f82645fe44dcd16cc4a36c71d6cd
  Gas usage: 276780404d805038cf539a6abf544e
  Block Number: 20
  BlockTime: Sat Dec 29 2018 11:20:43 GMT+0800 (GMT+08:00)
0x3aa3c344e17d465404d805038cf539a6abf544e
eth_call1r [ s: 1, e: 0, c: [ 2 ] ]
eth_call1
eth_call1c344e17d465404d805038cf539a6abf544e
eth_call1
eth_call1000000000000000000000000000000000000000000000000000000000000000
eth_call1
```

有call命令还有sendTransaction命令。需要更新数据的函数需要用sendTransaction的方式调用，然后区块链会写入新的区块来完成信息的更新。

总结

那么我的Dapp项目就是这样的了！一个去中心化的社交平台。社交平台这东西写起来真的是麻烦！点赞这东西实现起来好难！还有删除动态什么的。感觉前后端的开发自己实在是太菜了，这一个星期以来写这个东西真的是费了很大的劲。区块链给我的感觉是一个无中心组织的数据库。你可以拉取数据也可以保存数据。但是这些数据没有任何的管理员可以有特权去获取他们，或者说，每一个用户，都是这个数据库的管理员。整个开发过程更像是自己搭建了一个社交网站。

好累。整整一个星期。啥都不做只怼数据库的一个星期。我觉得这门课已经在我的大学生活里留下了浓墨重彩的一笔。我从来没有见过如此，困难的作业。可能是我的项目选的太麻烦了，社交应用。我的天！不过论收获还是挺大的。毕竟这是一个从头到尾都是自己写出来的东西，感触太多了。主要是坑太多了，安装包，卸载包什么的，truffle的版本还不兼容！然后许多包安装起来十分困难！各种报错，就连uninstall都要报错！摸爬滚打鼻青脸肿。怕了怕了。