中山大学数据科学与计算机学院本科生实验报告

(2019 年秋季学期)

课程名称： 区块链原理与技术　　任课教师： 郑子彬

年级	2017	专业（方向）	软件工程
学号	17343044	姓名	黄宝莹
电话	13437599547	Email	1723158024@qq.com
开始日期	2019.11.20	完成日期	2019.12.11

一、项目背景

如今，区块链技术已经影响了多个领域，以智能合约为载体的区块链应用也给许多行业带来了革命性的改变，以其不可篡改性和去中心化的特点深受欢迎。

在传统的供应链金融中，由于资金周转等问题，企业间签订延迟收款的账款单据的情况经常存在。但是，企业的签发和转让应收账款单据，以及融资等都需要经过繁琐的调查和认证，不仅增加了经济成本，而且还耗费许多时间，缺乏时效性。归根究底，是因为企业的信用没有一个公共的载体可以传递，交易信息的不透明化也使得认证信用的过程必不可少。

有了区块链技术的公开透明性，我们便可以省去上述过程，在线上即可对企业的信用进行认证，而且操作也更加简便。本项目则实现了这样一个区块链+供应链金融的应用。

二、方案设计

1.存储设计

以下介绍合约里数据的存储：

（1）应收账款结构体

包括了收款方、付款方、金额、到期时间、是否有第三方认证等信息。

```
struct Irrigation{
    uint id;            //id
    string reciever;    //收款方
    string payer;       //付款方
    string info;        //账单信息
    uint amount;        //金额
    string expiretime;  //到期时间
    uint state;         //0-无第三方认证  1-有第三方认证  2-抵押中  3-销毁
}
```

（2）公司结构体

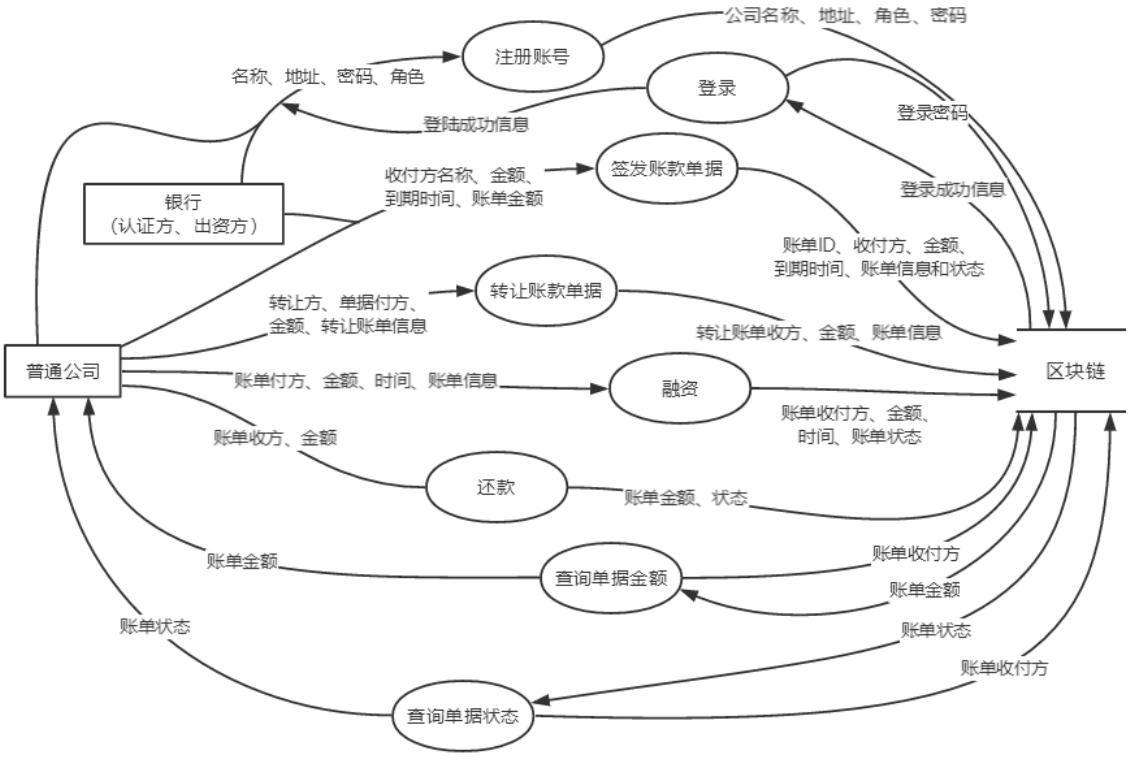
公司信息结构体包括公司名、地址、公司类型、收付账单列表等。用一个 mapping 映射关系，建立以对方公司名为键的账款数组。

```
struct Company{
    string cpName;
    address addr;
    uint cpInfo;    //0-bank认证   1-公司
    mapping(string => Irrigation) revIrr;  //作为收款方
    mapping(string => Irrigation) payIrr;  //作为付款方
}
```

（3）设立一个唯一的认证方，一个所有账款的数组，一个以公司名称为键值的公司列表，一个以公司名称为键的登录密码列表。

```
Company public authRole;   //银行作为认证方
Irrigation[] public irr;   //应收账款列表
mapping(string => Company) cplist;  //所有参与者列表
mapping(string => string) password;
```

2.数据流图



3.核心功能

（1）公司加入链

金融主体加入公司列表函数实现记录公司的名字、地址、角色、登录密码等。如果角色是 0，则代表是唯一的认证方。因此，公司不能够有重复的名称、地址，而且只有两种角色，0-权威方，具有认证和放款作用；1-普通公司。

```
function cpInit(string tname,address taddr,uint cprole, string pw) public
returns(string){
    cplist[tname] = Company({
        cpName:tname,
        addr:taddr,
        cpInfo:cprole
    });
    password[tname] = pw;
    if (cprole == 0) {
        authRole = cplist[tname];
    }
    string memory output = "Add a new company successfully!";
    return output;
}
```

（2）查询登录密码

根据公司的名称返回该公司的密码字符串。

```
function getPW(string name) public returns(string){
    return password[name];
}
```

（3）签发应收账单

公司登录后，可以采购商品，签发账单。它把账单的收方、金额、到期时间记录下来。如果是由权威方代为签发的账单，那么该账单是可信的，具有融资的资格。以下是账单结构体初始化：

```
if (msg.sender == authRole.addr) {
    tstate = 1;
}
Irrigation memory curIrr = Irrigation({
    id:len,          //id
    reciever:trecv,  //收款方
    payer:tpayer,     //付款方
    info:tinfo,       //账单信息
    amount:tamount,         //金额
    expiretime:texpiretime,       //操作时间
    state:tstate
});
```

（4）转让账单

公司无力付款给其他公司时，可以进行转让账单的操作以替代货款。上一级公司（转让者）需要输入下一级公司（接收者）、付款公司、金额等信息，同时要判断转让**账款是否用于融资或者无效状态**。如果转让的金额多于账款的金额，转让也是无效的。以下是判断转让无效的情况：

```
//Wrong Situation
if (targetbill.amount < amount) {
    return "Wrong:The amount is larger than the irrigation";
}
if (targetbill.state == 2 || targetbill.state == 3){
    return "Wrong:The irrigation has been used to financing or invalid";
}
```

转让账单后，新的收款方变成接收者，则原付款方和新收款方形成新的账单，和转让账单相比只是收款方和金额有变化：

```
string memory output = instructIrr(
downer,payer,amount,targetbill.expiretime,info);
Irrigation storage out = cppayer.payIrr[downer];
out.state = targetbill.state;
output = "Transfer the irrigation successfully!";
```

（5）融资

公司可以凭借拥有的应收账单去向银行申请融资。前提是申请的金额不能大于抵押账单的金额，而且该账单**必须是可信的，不处于抵押状态的和非无效的。**融资成功后，银行和公司之间形成新的账单：

```
instructIrr(financer,reciver,amount,texpiretime,info);
string memory output = "The financer lend money to the company
successfully!";
```

（6）还款清算

公司可以对欠下的账单进行还款清算。为了保证交易的高效性，我这里设定还款只能还足够的账单金额，一次性还清。如果该账单已经清算过了就不需要再还钱。还钱成功后，会更改对应账单的状态为无效，记录下相应的还款信息。

```
//Wrong Situation
if (billrecv.amount != amount) {
    return "Wrong:The money isn't enough to repay the irrigation";
}
if (billrecv.state == 3){
    return "Wrong:The irrigation is invalid";
}

//Success Situation
billrecv.amount = 0;
billrecv.state = 3;
billrecv.info = info;
```

（7）查询账单金额

公司可以通过输入账单的收付方来查询账单的金额，以便于转让、融资、清算等等。

```
function getAmountOfIrr(string reciver,string payer)public returns(uint)
    //Company memory recivecp = cplist[reciver];
    //Irrigation memory targetbill = recivecp.revIrr[payer];
    return cplist[reciver].revIrr[payer].amount;
    //return targetbill.amount;
}
function test() public view returns(uint){
    return 99999;
}
```

（8）查询账单状态

公司可以通过输入账单的收付方来查询账单的状态，是可信的，还是抵押中或者无效的。

```
function getStateOfIrr(string reciver,string payer)public returns(string){
    //Company memory recivecp = cplist[reciver];
    Irrigation memory targetbill = cplist[reciver].revIrr[payer];
    string memory output;
    if (targetbill.state == 0) {
        output = "The irrigation isn't guaranteed and not used to financing";
    }
    else if (targetbill.state == 1) {
        output = "The irrigation is guaranteed and not used to financing";
    }
    else if (targetbill.state == 2) {
        output = "The irrigation is used to financing";
    }
    else {
        output = "The irrigation is invalid";
    }
    return output;
}
```

三、功能测试

1.公司注册账号

（1）银行注册

输入名称 Bank，以及生成的公钥地址，密码以及确认密码，输入角色。另外，为了保证账户的安全性，本项目在输入密码时一律都会隐藏。

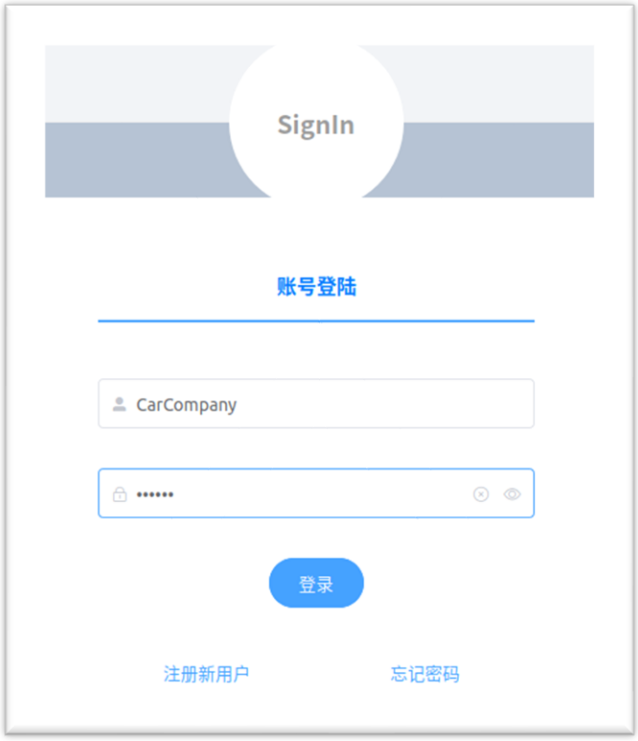
显示“注册成功”：

（2）普通公司注册

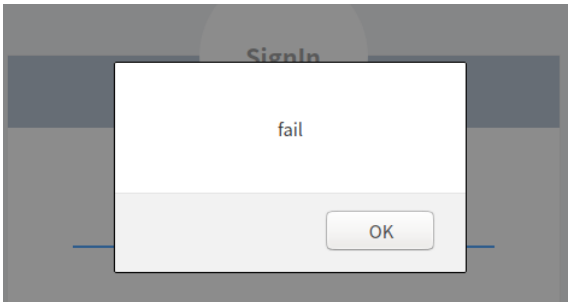
以汽车公司为例，输入名称 CarCompany，账户公钥地址等信息：

2.登录

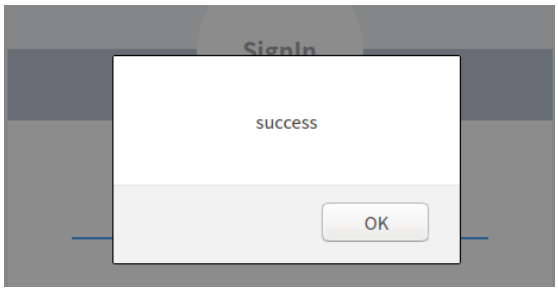
以 CarCompany 为例，输入公司名称和正确密码即可登录。



账户不存在或密码不正确时会返回“fail”信息：



登录成功时，会返回“success”信息：

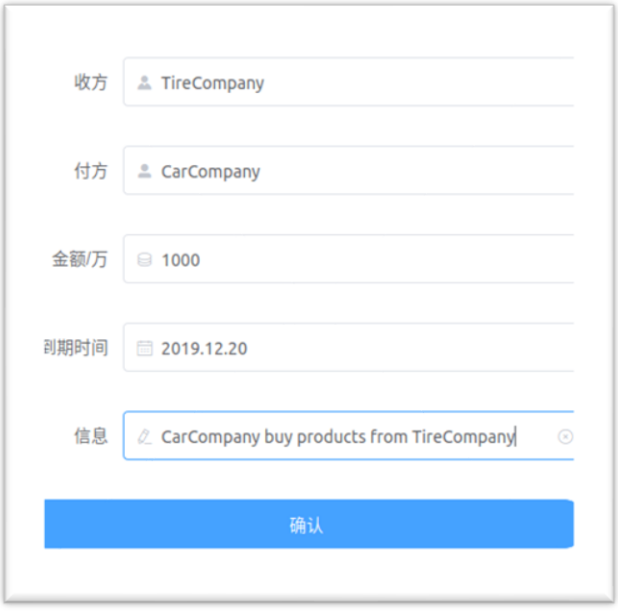


3.退出登录

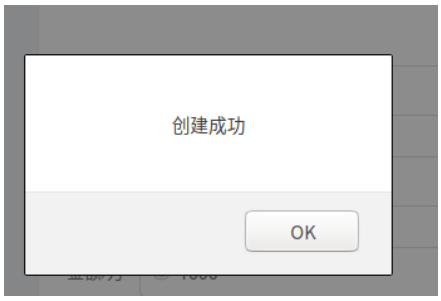
公司登录成功后，在页面右上角会出现公司的名称，点击设置按钮可以选择退出登录。

4.创建票据

CarCompany 向 TireCompany 购买商品，签订了 1000 万的账单，由 Bank 负责发起，具有可信性。填写时输入收付方名称、金额（以万为单位）、到期时间和账单信息。

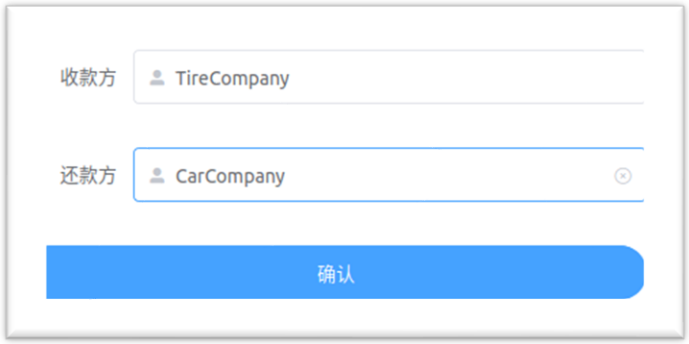


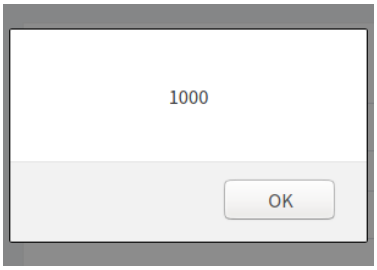
创建账单成功后，会显示如下：



5.查询票据金额

此时查询 CarCompany 和 TireCompany 之间的票据，前者为收方，后者为付方，返回信息是该票据的金额。

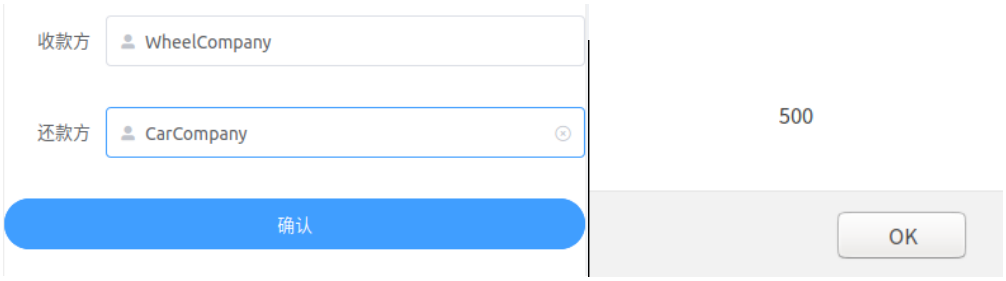




6.转让票据

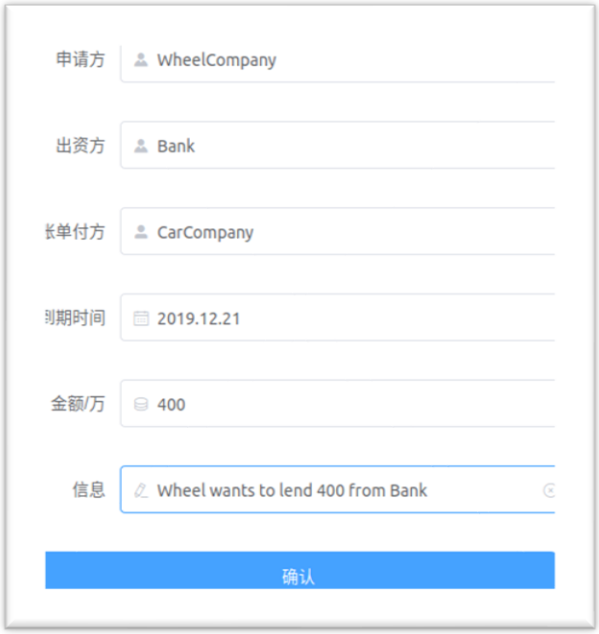
由于轮胎公司 TireCompany 从轮毂公司处买了 500 万的商品，但是没有现款支付，所以才去转让汽车公司的应收票据给 WheelCompany ，从 1000 万的账单中拆出 500 万。

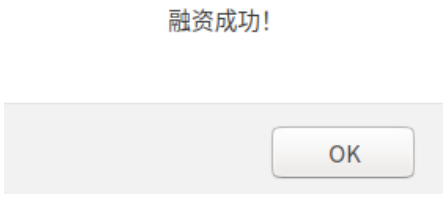
此时查询 WheelCompany 和 CarCompany 之间，可证实票据的确转让成功，它们之间存在 500 万的应收账款。



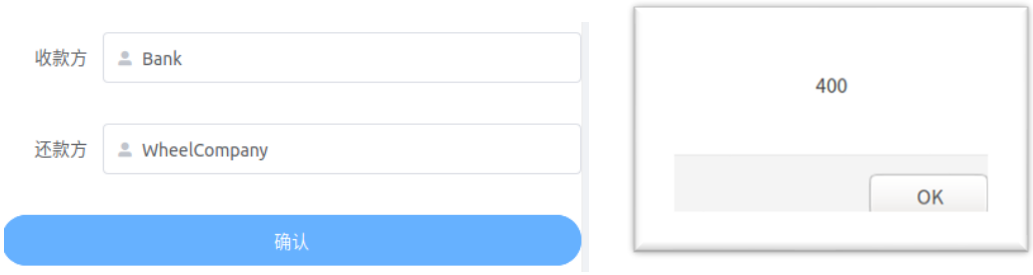
7.融资

WheelCompany 向 Bank 融资，以 CarCompany 账单为抵押，融资金额不能大于抵押账单的金额，所以金额为 400 万时，融资成功:



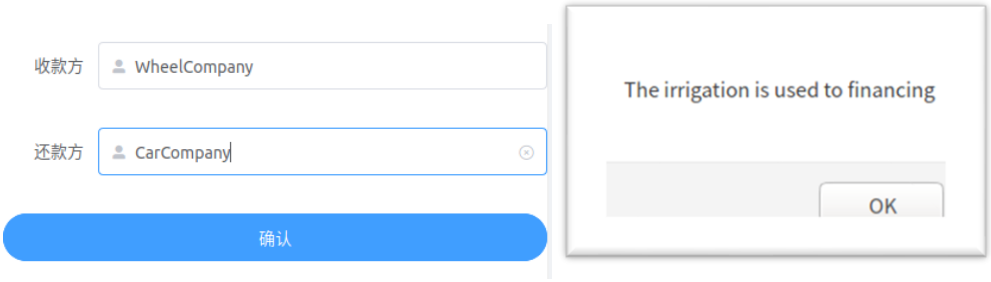


由于融资其实是在 Bank 和公司之间产生新的票据，和转让的性质不同，所以此时查询 WheelCompany 和 Bank 之间，的确存在 400 万的账单:

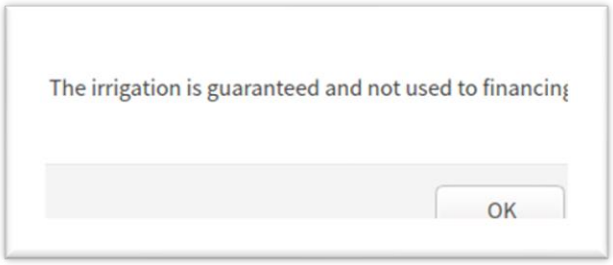


8.查询账单状态

经过步骤 7 后，可以看到 WheelCompany 和 CarCompany 之间的账单处于抵押状态，因为该账单已经被用于融资了，即处于抵押状态。

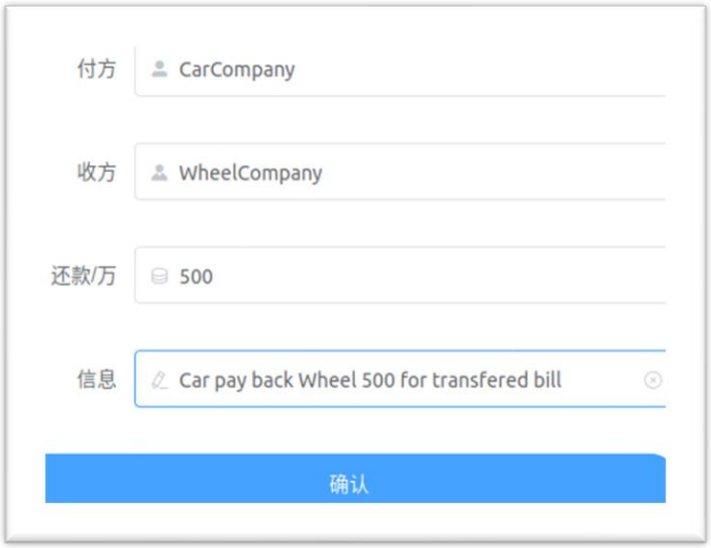


由于 TireCompany 和 CarCompany 之间的账单并没有被用于融资，所以处于可信未抵押状态。

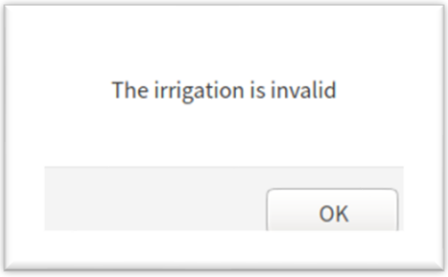


9.还款

在到期日期之前，CarCompany 还给 WheelCompany 500 万（账单上的金额），填入还款信息：

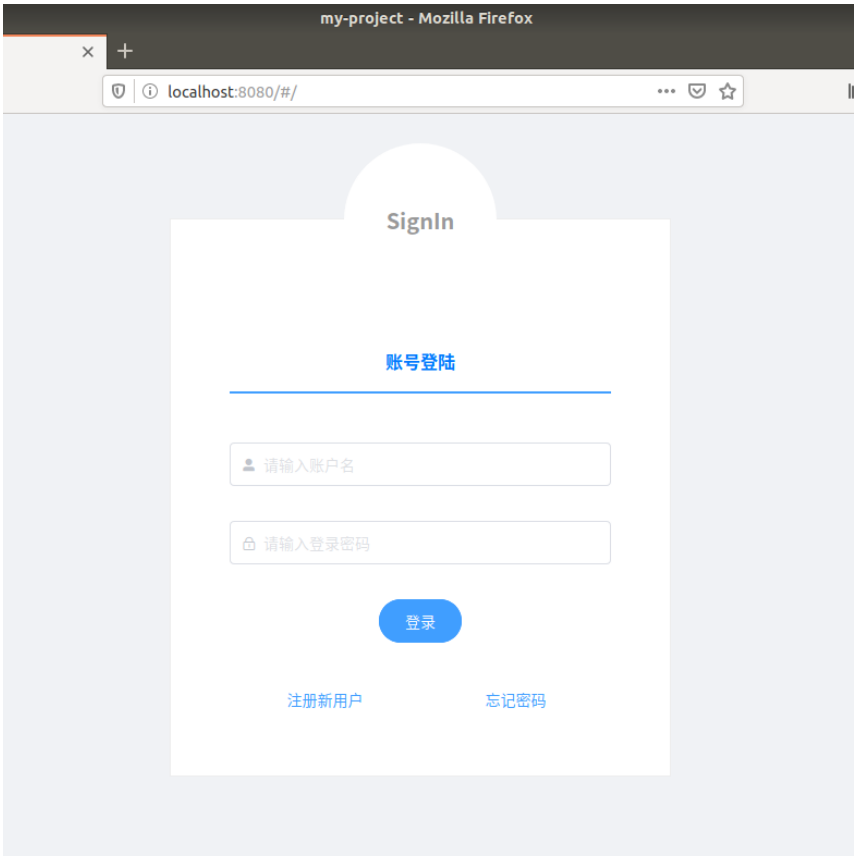


此时查询二者之间账单已经清算，所以状态为无效：



四、界面展示

1.登录页面



2.注册页面

3.菜单栏

公司和银行登录后可以看到左侧的可选操作菜单栏，分为“查询信息”和“账款管理”两部分：

4.创建票据页面

5.查询票据金额和状态

6.转让应收账款页面

7.融资页面

8.还款页面

五、加分项

1.数据隐私保护：用户注册和登录需要密码，而且输入时会隐藏字符。

2.权限和认证：只有银行和公司自己才能够签发以该公司为参与者的账单，以及转让、融资、还款等功能。只有银行发起的账款，代表已经被认证，具有可信度，才可以用于融资。

3.用户界面友好：登录、注册、功能页面简洁大方，布局清晰，用户体验良好。

六、项目启动

1.配置证书及 Channel 端口

按照官网搭建好 FISCO BCOS 链之后:

（1）修改配置文件，证书配置位于 nodejs/packages/api/conf/config.json 文件的 authentication 配置项中。根据实际使用的证书文件的路径修改该配置项的 key、cert 及 ca 配置，其中 key 为 SDK 私钥文件的路径，cert 为 SDK 证书文件的路径，ca 为链根证书文件的路径。

（2）修改配置文件，节点 IP 及端口配置 packages/api/conf/config.json 文件的 nodes 配置项中。根据要连接 FISCO BCOS 节点的实际配置修改该配置项的 ip 及 port 配置，其中 ip 为所连节点的 IP 地址，port 为节点目录下的 config.ini 文件中的 channel_listen_port 配置项的值。

2.生成账户

通过账户生成脚本 get_accounts.sh 生成了 PEM 格式的账户私钥文件和公钥，把公钥作为账户的地址，生成的私钥文件置于 api/conf/accounts 目录下，并修改 api/conf/config.json 文件的 privateKey 项:

```
root@fiscobcos-VirtualBox:~/Desktop# bash get_account.sh
[INFO] Account Address   : 0xd7ee249e0d998f2dcc83fc305f3ecf38600d5eed
[INFO] Private Key (pem) : accounts/0xd7ee249e0d998f2dcc83fc305f3ecf38600d5eed.p
em
[INFO] Public  Key (pem) : accounts/0xd7ee249e0d998f2dcc83fc305f3ecf38600d5eed.p
ublic.pem
```

3.启动前后端

（1）后端：进入 backend 文件夹，运行以下命令:

```
npm install
node backend.js
```

（2）前端：进入 vue 文件夹，运行以下命令:

```
npm run dev
```

（3）打开浏览器 localhost:8080 即可。

七、心得体会

这次区块链大作业对我来说真的是一次不小的挑战。不仅要写前后端，而且还要用微众 FISCO-BCOS 的联盟链对应的 SDK 来写后端和链端的交互。我选择的框架是 Vue+Express，用 Node.js 来写后端，套用了 Element UI 的前端格式。

由于官网文档在 Node.js SDK 方面的资料比较少，不是特别完整和详细，所以一开始我对如何把智能合约和后端连接起来比较困惑。而身边的同学大多都是用 Java SDK，我就在微信群求助。幸运的是，很多大佬同学给了我提示和帮助，而且非常耐心，微众的老师也很热心地在群里解答我们的问题，在此感谢他们。

由于上一次写前后端已经是一年前的事了，所以复习了很久才回想起来整个过程。经过这次大作业的锻炼，我对区块链应用的理解也更深了，诚然，它在供应链方面可以做到更加公开透明，操作流程也简化很多。但是，我的代码水平还是不足，所以并没有设计太复杂的函数，以后会多多练习，希望可以真正地把区块链用到实处上。