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	17343106	姓名	王嘉浚
电话	15013046885	Email	853845711@qq.com
开始日期	2019.11.10	完成日期	2019.12.11

一、 项目背景

中小微企业融资难、融资贵是长久以来我国金融发展过程中需要解决的问题。世界银行、中小企业金融论坛、国际金融公司联合发布的《中小微企业融资缺口：对新兴市场微型、小型和中型企业融资不足与机遇的评估》报告中表示，中国 40%的中小微企业存在信贷困难，或是完全无法从正规金融体系获得外部融资，或是从正规金融体系获得的外部融资不能完全满足融资需求，有 1.9 万亿美元的融资缺口，接近 12 万亿元人民币。

比如以下场景。某知名车企(宝马)消费口碑好，金融机构对其信用评级很高，认为其有很大的风险承担能力。某次交易中，该车企从轮胎公司购买了一批轮胎，但由于资金暂时短缺向轮胎公司签订了 1000 万的应收账款单据，承诺一年后归还。由于轮胎公司有宝马的账款单据，金融机构认为其有额能力还款，于是愿意借款给轮胎公司，然而，这种信任关系并不会向下游传递。比如，轮胎公司因资金短缺，向轮毂公司签订了 500 万的应收账款单据，但是当轮毂公司需要贷款时，金融机构因不认可轮胎公司的还款能力，需要对轮胎公司进行详细的信用分析，而这会增加很多的经济成本。很多时候，就是这个原因导致了小微企业的融资失败。

但是，区块链金融可以有效地解决上述问题。将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

二、 方案设计

将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

【存储设计】

将企业的收款单据存储在通过 FISCO BCOS 部署的四节点联盟链上。由于我们要解决的问题是供应链上下游的信息不对等而导致的融资难问题，我们只关注企业间的欠款信息，而忽略企业的余额，这也是本项目的设计思路。

当两个企业签订了应收账款单据，将它们的公司信息、应收数目上链，在必要时，企业间的应收账款可以转移到第三方，比如上例轮毂公司可以拿到宝马的 500 万欠款证明。

【核心功能介绍】

本项目主要实现以下四个功能：

✧ 功能一：实现采购商品—签发应收账款交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

✧ 功能二：实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。

✧ 功能三：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。

✧ 功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

核心功能主要体现在部署在区块链上的智能合约。智能合约代码如下：

```
1.  contract SupplyChain{
2.
3.      struct company{
4.
5.          string name;
6.
7.          address adr;
8.
9.      }
10.
11.
12.
13.     struct receipt{
14.
15.         address come;
16.
17.         address to;
18.
19.         uint amount;
20.
21.     }
22.
23.
24.
25.     receipt[] public receipts;
26.
27.     company[] public companies;
28.
29.     mapping(address=>uint) debt;
30.
31.
32.
33.     constructor(){
34.
```

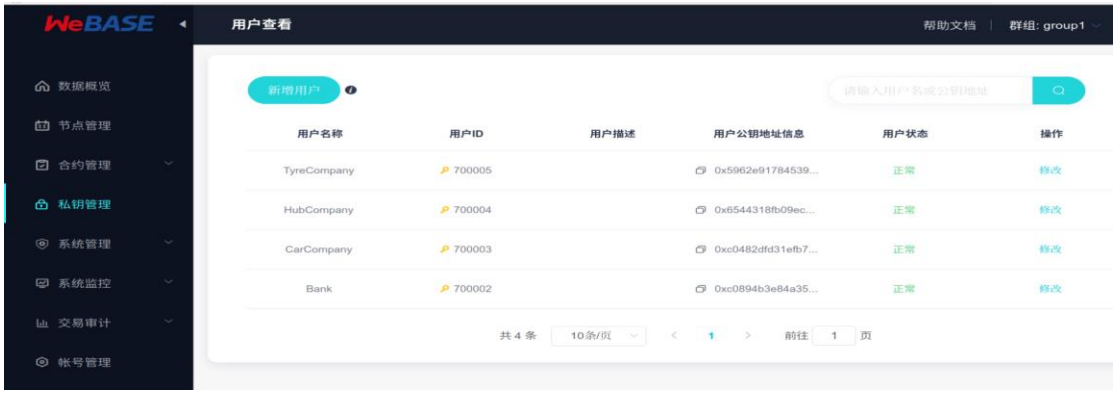
```
35.
36.
37.     }
38.
39.
40.
41.     function Transaction(address receive,uint amount,bool is_bank){
42.
43.         if(is_bank){
44.
45.             GetDebt();
46.
47.             if(debt[msg.sender]<0){
48.
49.                 return;
50.
51.             }
52.
53.         }
54.
55.         receipt memory r;
56.
57.         r.come = msg.sender;
58.
59.         r.to = receive;
60.
61.         r.amount = amount;
62.
63.         receipts.push(r);
64.
65.     }
66.
67.
68.
69.     function Transfer(address adr){
70.
71.         address adr1 = msg.sender;
72.
73.         address adr2 = adr;
74.
75.         for(uint i=0;i<receipts.length;i++){
76.
77.             if(receipts[i].come==adr1){
78.
79.                 address adr3 = receipts[i].to;
80.
81.                 for(uint j=0;j<receipts.length;j++){
82.
83.                     if(receipts[j].come==adr3 && receipts[j].to==adr2){
84.
85.                         receipts[i].amount -= receipts[j].amount;
86.
87.                         receipt memory r;
88.
89.                         r.come = adr1;
90.
91.                         r.to = adr2;
92.
93.                         r.amount = receipts[j].amount;
94.
95.                         receipts.push(r);
96.
97.                     }
98.
99.                 }
100.
101.             }
102.
```

```
        }

    }


    function GetDebt(){

        uint sum = 0;

        for(uint i=0;i<receipts.length;i++){

            if(receipts[i].come==msg.sender){

                sum -= receipts[i].amount;

            }

            else if(receipts[i].to==msg.sender){

                sum += receipts[i].amount;

            }

        }

        debt[msg.sender] = sum;

    }


    function PayDebt(address receive, uint amount){

        for(uint i=0;i<receipts.length;i++){

            if(receipts[i].come==msg.sender && receipts[i].to==receive){

                if(receipts[i].amount-amount>0){

                    receipts[i].amount-=amount;

                    break;

                }

                else if(receipts[i].amount-amount==0){

                    for(uint j=i;j<receipts.length-1;j++){

                        receipts[j].come = receipts[j+1].come;

                        receipts[j].to = receipts[j+1].to;

                        receipts[j].amount = receipts[j+1].amount;

                    }

                }

                else return;

            }

        }

    }
}
```

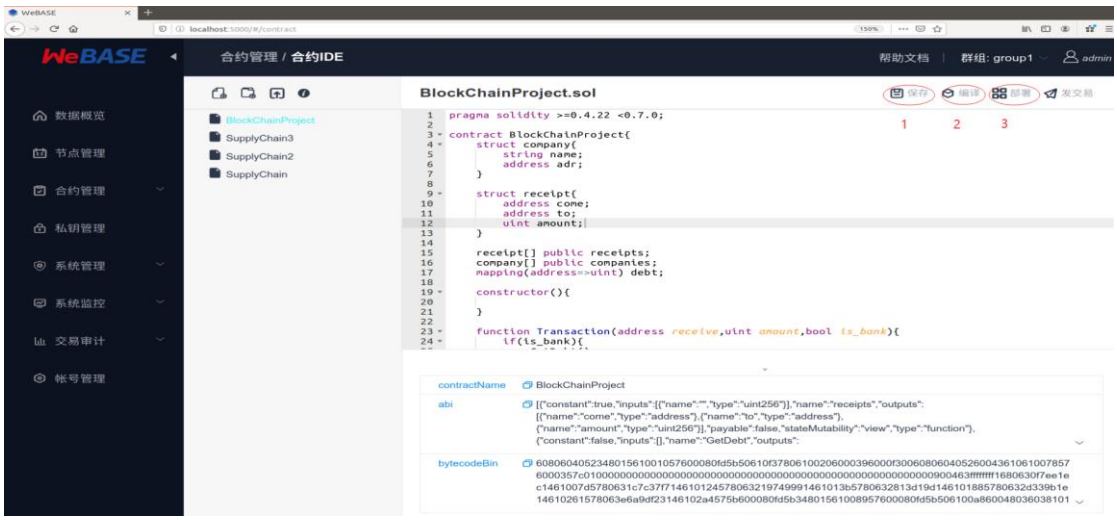
三、 功能测试

首先在 WeBase 平台上测试智能合约。

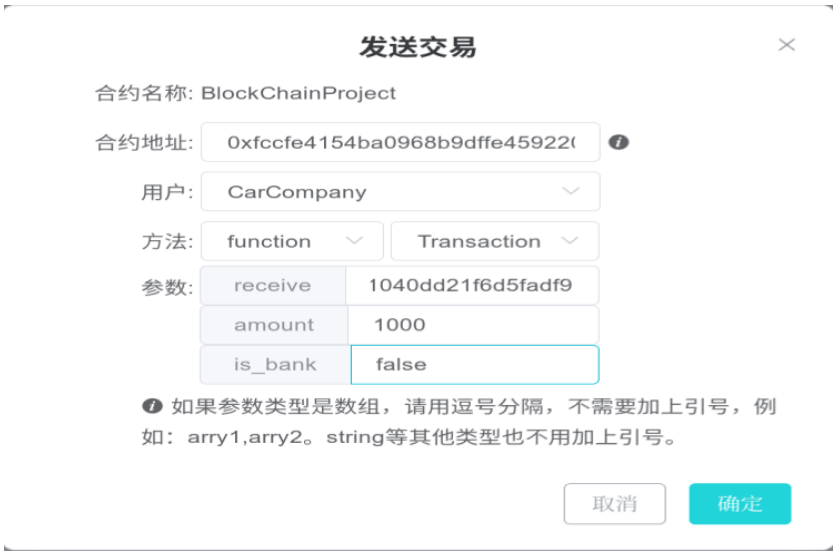
首先，根据不同公司创建用户，一共有：银行(Bank)、汽车厂(CarCompany)、轮胎厂(TyreCompany)、轮毂厂(HubCompany)：



对编写好的合约进行编译、部署：



功能一：实现采购商品—签发应收账款交易上链。



(说明：车企欠轮胎厂 1000 万元)

部署成功会显示相关信息。

(说明：轮胎厂欠轮毂厂 500 万元)

查看链上的交易信息：

说明交易信息成功上链。功能一完成。

功能二：实现应收账款的转让上链。

(说明：车企欠轮胎厂的部分金额，转嫁到轮毂厂)

再次查看链上的交易信息：

现在车企欠轮胎厂和轮毂厂各 500 万元，说明功能二实现。

功能三：利用应收账款向银行融资上链。

(获取轮毂厂的总债务)

轮毂厂向银行借贷：

说明功能三实现。

功能四：应收账款支付结算上链。

(车企还轮胎厂 200 万元)

(车企还欠轮胎厂 300 万元)

车企再还 300 万元给轮胎厂，则抹去链上该条交易信息。

以上是智能合约的测试结果。

四、 心得体会

本次项目是一个基于区块链的供应链金融平台，主要解决供应链上下游债权信息不对等而导致的微小企业融资难的问题，是区块链的一个简单应用。通过实验，我们不仅学习了如何使用 FISCO BCOS 构建联盟链并将我们需要的数据上链，更深刻地体会到了区块链在解决信任问题上的巨大优势与广阔的应用前景。

虽然学习区块链课程还不到 一个学期，学习的内容有限，但是通过这样有意义的项目，我们将自身所学付诸实践，在此过程中也巩固了课堂所学的理论内容。最后，非常感谢郑老师、助教、微众银行的老师、技术支持们对我们的悉心讲解！