中山大学计算机学院本科生实验报告

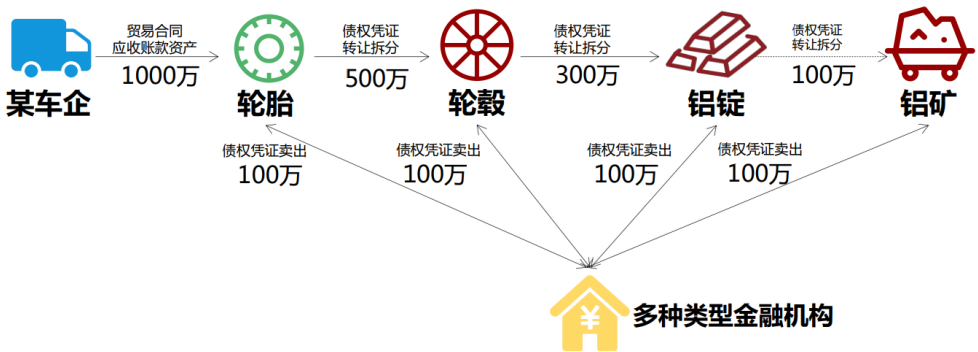
(2020 年秋季学期)

课程名称：区块链原理与技术　　任课教师：黄华威

年级	2018 级	专业（方向）	保密管理
学号	18339012	姓名	梁懿雯
	18339009		李依茜
	18339018		史佳茵
电话		Email	
开始日期	2020.12	完成日期	2021.1

一、 项目背景

本项目要基于已有的开源区块链系统 FISCO-BCOS，以联盟链为主，开发基于区块链或区块链智能合约的供应链金融平台，实现供应链应收账款资产的溯源、流转。如图是提供的供应链场景。



应该做到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。

具体来说，要实现的功能有四个：

功能一：实现采购商品，签发应收账款交易上链。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。

二、 方案设计

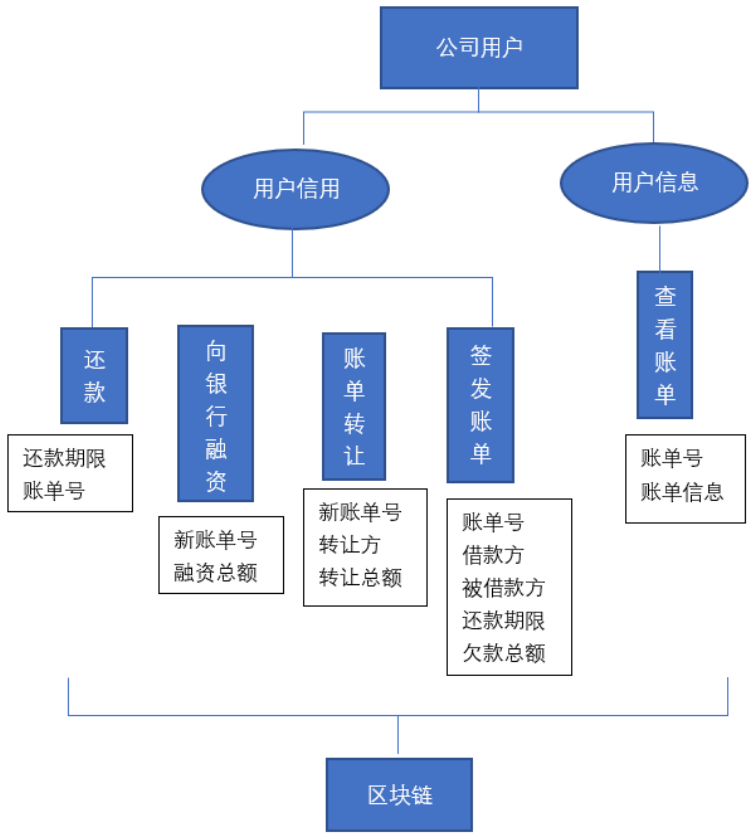
1.存储设计

存储使用了已有的智能合约 Table.sol 接口，即创建两个表，分别对应账单信息和用户信息列表。

```
function createTable() private {
    TableFactory tf = TableFactory(0x1001);

    // 资产管理表, key : bill_id, field : lender  borrower  tran_amount  paytime
    // | 账单号(主键) | 借出钱的用户 | 欠钱的用户 |     总额     | 还款时间 |
    // |    bill_id  |   lender    |  borrower  | tran_amount  | paytime  |
    // 创建表:账单
    tf.createTable("t_bill", "bill_id", "lender,borrower,tran_amount,paytime");
    // 资产管理表, key : user, field :  trust(0不受信用，1受信用)
    // |      用户名(主键)   |   是否受银行信用  |
    // |       user         |      trust       |
    // 创建表：用户
    tf.createTable("t_user", "user", "trust");
}
```

2.数据流图



3.核心功能设计

首先说明一下智能合约文件 mypro.sol 是如何实现基本的四个功能的。

- **事件声明**

如图是合约中五个事件的声明，这五个事件分别对应的是：用户注册，功能 1 签发账款，功能 2 转让账款，功能 3 银行融资，功能 4 支付结算。

```
pragma solidity ^0.4.24;

import "./Table.sol";

contract mypro{
    //添加用户：返回值，用户名
    event Register(int256 ret, string user, int256 trust);
    //功能一 签发应收账款：返回值，账单号，借出钱的用户，欠钱的用户，欠款总额，还款时间(多少
天后)
    event Sign(int256 ret, string bill_id, string lender, string borrower,  uint256
tran_amount, uint256 paytime);
    //功能二 转让应收账款：返回值，分解的账单号，新的账单号，新的借出钱的用户，转让总额
    event Transfer(int256 ret, string pre_bill_id, string new_bill_id, string
new_lender, uint256 tran_amount);
    //功能三 向银行融资：返回值，融资用户，融资总额，还款时间
    //event Financing(int256 ret, string user,  uint256 tran_amount, uint256 paytime);
    event Financing(int256 ret);
    //功能四 应收账款支付结算：返回值，账单号，距离借钱日过去的时间（天为单位）
    //event Pay(int256 ret, string bill_id, uint256 lendtime);
    event Pay(int256 temp_pay);
```

• 构造账单表及用户表

然后是构造函数，首先，用到了已有的 Table.sol 接口，创建两个表，分别对应账单信息和用户信息列表。账单信息包括账单号，借钱者，欠钱人，借款金额和时间。用户信息包括用户名和是否受银行信用。

下图还有两个函数：openTableUser 和 openTableBill，它们的功能是将打开表的操作封装起来。

```
constructor() public {
    //构造函数中创建表
    createTable();
}

function createTable() private {
    TableFactory tf = TableFactory(0x1001);

    // 资产管理表, key : bill_id, field : lender  borrower  tran_amount  paytime
    // | 账单号(主键) | 借出钱的用户 | 欠钱的用户 |     总额     | 还款时间 |
    // |    bill_id  |   lender    |  borrower  | tran_amount  | paytime  |
    // 创建表:账单
    tf.createTable("t_bill", "bill_id", "lender,borrower,tran_amount,paytime");
    // 资产管理表, key : user, field :  trust(0不受信用，1受信用)
    // |      用户名(主键)   |   是否受银行信用  |
    // |       user         |      trust       |
    // 创建表：用户
    tf.createTable("t_user", "user", "trust");
}

function openTableUser() private returns(Table) {
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("t_user");
    return table;
}

function openTableBill() private returns(Table) {
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("t_bill");
    return table;
}
```

• **查找用户信息**

下面这个 selectAccount 函数用于查找用户信息列表，根据用户名查看用户是否受银行信用。返回的第一个参数为 0 表示用户存在，第二个参数为 0 表示不受信用，为 1 表示受信用。

```
/*
根据用户名查询是否受银行信用
参数 ： user ： 用户名
返回值：参数一： 成功返回0，账户不存在返回-1      参数二： 第一个参数为0时有效，是否受信用
*/
function selectAccount(string user) public constant returns(int256, uint256) {
    // 打开表
    Table table = openTableUser();
    // 查询
    Entries entries = table.select(user, table.newCondition());
    uint256 trust = 0;
    if (0 == uint256(entries.size())) {
        return (-1, trust);
    } else {
        Entry entry = entries.get(0);
        return (0, uint256(entry.getInt("trust")));
    }
}
```

• **查找账单信息**

下面这个函数 selectBill 的功能是根据账单号查看该账单信息。返回参数见下图注释。

```
    /*
    根据账单号查询借出钱的用户、欠钱的用户、总额、还款时间
    参数 ： bill_id ： 账单号
    返回值：参数一： 成功返回0，账户不存在返回-1     参数二： 第一个参数为0时有效，借出钱的用户
            参数三： 第一个参数为0时有效，欠钱的用户     参数四： 第一个参数为0时有效，总额
            参数五： 第一个参数为0时有效，还款时间
    */
    function selectBill(string bill_id) public constant returns(int256, string, string,
uint256, uint256) {
        // 打开表
        Table table = openTableBill();
        // 查询
        Entries entries = table.select(bill_id, table.newCondition());
        //string  storage lender = "0";
        //string storage borrower = "0";
        //uint256 tran_amount = 0;
        //uint256 paytime = 0;
        if (0 == uint256(entries.size())) {
            //return (-1, lender, borrower, tran_amount, paytime);
            return (-1,"0","0",0,0);
        } else {
            Entry entry = entries.get(0);
            //lender = entry.getString("lender");
            //borrower = entry.getString("borrower");
            //tran_amount = entry.getInt("tran_amount");
            //paytime = entry.getInt("paytime");
            return (int256(0), entry.getString("lender"), entry.getString("borrower"),
uint256(entry.getInt("tran_amount")), uint256(entry.getInt("paytime")));
        }
    }
```

如下表是以上部分代码里的一些变量名的意义。

变量名	备注
bill_id	账单号
lender	借出钱的用户

borrower	欠钱的用户
paytime	还款时间
user	用户名
trust	是否受银行信用

- **用户注册**

下面这个函数是用户注册的函数，表中的变量名对应了变量的意义。

```
/*
用户注册
参数 ： user : 资产账户       trust  : 是否受银行信用
返回值：0:资产注册成功   -1:资产账户已存在   -2:其他错误
*/
function registerUser(string user, int256 trust) public returns(int256){
    int256 res = 0;
    int256 temp_check= 0;
    uint256 temp_trust = 0;
    // 查询账户是否存在
    (temp_check, temp_trust) = selectAccount(user);
    if(temp_check != 0) {
        Table table = openTableUser();

        Entry entry = table.newEntry();
        entry.set("user", user);
        entry.set("trust", trust);
        // 插入
        int temp_insert = table.insert(user, entry);
        if (temp_insert == 1) {
            // 成功
            res = 0;
        } else {
            // 失败? 无权限或者其他错误
            res = -2;
        }
    } else {
        // 账户已存在
        res = -1;
    }

    emit Register(res, user, trust);
    return res;
}
```

变量名	备注
user	传入参数 1，表示要注册的用户名
trust	传入参数 2，表示用户是否受银行信用
res	表示最后的返回结果，0/-1/-2
temp_check	表示查找账户后的结果
temp_trust	表示查找用户后返回的信用结果
temp_insert	表示插入后的返回结果

以上是基本功能的函数，下面是四个功能函数的实现说明。

• 功能一：签发应收账款

首先是功能一签发应收账款的实现函数，实现方法如下：先打开账单表，然后新建一个 entry，将账单的相关信息存储进 entry 中，然后插入账单表。可以看到该函数触发了事件 Sign，即签发应收账款。

```
    /*
    event Sign 功能一 签发应收账款：返回值，账单号，借出钱的用户，欠钱的用户，欠款总额，还款时
间(多少天后)
    参数：bill_id : 账单号    lender  : 借出钱的用户     borrower : 欠钱的用户
          tran_amount : 欠款总额     paytime : 还款时间(多少天后)
    返回值：0:签发应收账款成功    -2:其他错误
    */
    function issueBill(string bill_id, string lender, string borrower, uint256
tran_amount, uint256 paytime) public returns(int256){
        int256 res = 0;
        int256 temp= 1;

        if(temp != 0) {
            Table table = openTableBill();

            Entry entry = table.newEntry();
            entry.set("bill_id", bill_id);
            entry.set("lender", lender);
            entry.set("borrower", borrower);
            entry.set("tran_amount", int(tran_amount));
            entry.set("paytime", int(paytime));

            // 插入
            int temp_insert = table.insert(bill_id, entry);
            if (temp_insert == 1) {
                // 成功
                res = 0;
            } else {
                // 失败? 无权限或者其他错误
                res = -2;
            }
        }

        emit Sign(res, bill_id, lender, borrower, tran_amount, paytime);
        return res;
    }
```

函数中的变量名	备注
bill_id	传入参数 1，表示账单号
lender	传入参数 2，表示借钱人
borrower	传入参数 3，表示欠钱人
tran_amount	传入参数 4，表示借钱总额
paytime	传入参数 5，表示还钱时间，x 天后才可以还钱
res	表示最后的返回结果，0 或者-2
temp_insert	表示插入后的返回结果

• 功能二：转让应收账款

下面是功能二实现应收账款的转让的实现函数，可以看到它也触发了事件 Transfer 转让账款。实现方法是：先打开账单表，然后用一个 entry_temp 选择将要分解的账单号，拿出将分解的账单的

总额，然后新建一个 entry，将各种相关信息存放进去。其中更新老账单的 pre_amount 等于将分解的账单的总额减去参数 tran_amount（转让总额）。然后更新列表。 之后再新建一个新账单的 entry，存放相关信息。其中新账单的 tran_amount 等于参数 tran_amount（转让总额）。新账单的借出钱人为参数中的 lender，新账单的欠钱用户等于老账单的欠钱用户。最后插入新账单。

```
    /*
    event Transfer 功能二 转让应收账款：返回值，分解的账单号，新的账单号，新的借出钱的用户，转
让总额
    参数 ： pre_bill_id : 分解的账单号    new_bill_id ：新的账单号
            new_lender : 新借出钱的用户    tran_amount : 转让总额
    返回值：0  签发应收账款成功      -2 其他错误
    */
    function transfer(string pre_bill_id, string new_bill_id,string new_lender, uint256
tran_amount) public returns(int256) {
        uint256 pre_amount = 0;

        Table table = openTableBill();

        Entries entry_bt = table.select(pre_bill_id, table.newCondition());
        Entry entry_lender = entry_bt.get(0);
        pre_amount = uint256(entry_lender.getInt("tran_amount"));
    //(int256(0), entry.getString("lender"), entry.getString("borrower"),
uint256(entry.getInt("tran_amount")), uint256(entry.getInt("paytime")));

        Entry entry0 = table.newEntry();
        entry0.set("bill_id", pre_bill_id);
        entry0.set("lender", entry_lender.getString("lender"));
        entry0.set("borrower", entry_lender.getString("borrower"));
        entry0.set("tran_amount", int256(pre_amount - tran_amount));
        entry0.set("paytime", int256(entry_lender.getInt("paytime")));

       // 更新被分解的账单
       int temp_insert = table.update(pre_bill_id, entry0, table.newCondition());
       if(temp_insert != 1) {
           // 失败? 无权限或者其他错误?
           emit Transfer(-2, pre_bill_id, new_bill_id, new_lender, tran_amount);
           return -2;
       }

       Entry entry1 = table.newEntry();
       entry1.set("bill_id", new_bill_id);
       entry1.set("lender", new_lender);
       entry1.set("borrower", entry_lender.getString("borrower"));
       entry1.set("tran_amount", int256(tran_amount));
       entry1.set("paytime", int256(entry_lender.getInt("paytime")));

       // 插入新账单
       int temp_insert2 = table.insert(new_bill_id, entry1);


       emit Transfer(0, pre_bill_id, new_bill_id, new_lender, tran_amount);
       return 0;
   }
```

变量名	备注
pre_bill_id	传入参数 1，要分解的账单号
new_bill_id	传入参数 2，新建立的账单号
new_lender	传入参数 3，新的借钱人
tran_amount	传入参数 4，转让金额
pre_amount	查表后得到原账单总额
temp_insert	表示插入后的返回结果
temp_insert2	表示插入后的返回结果

• **功能三：向银行融资**

然后是功能三利用应收账款向银行融资上链的实现函数，融资时要检查是否受信用，可以看到受信用时触发事件 Financing，传入的参数时 0，表示融资成功，否则传参-1，表示不成功。如果受信用，直接调用功能二 transfer 方法，向银行融资。

```
    /*
    event Financing  功能三 向银行融资：返回值，融资用户，融资总额，还款时间
        参数 ： pre_bill_id : 分解的账单号  new_bill_id ：新的账单号   tran_amount : 转让总额
        返回值： 0 银行融资成功   -1 欠款方的账单不受信用，银行不融资   -2 其他问题，融资失败
    */
    function fiancing(string pre_bill_id, string new_bill_id, uint256 tran_amount) public
returns(int256) {
        Table table0 = openTableBill();
        Table table1 = openTableUser();
        Entries entry_bt = table0.select(pre_bill_id, table0.newCondition());
        Entry entry_lender = entry_bt.get(0);

        Entries entry_ut = table1.select(entry_lender.getString("borrower"),
table1.newCondition());
        Entry entry_trust = entry_ut.get(0);

        //1为受信用
        if(entry_trust.getInt("trust")==int256(0)){
            emit Financing(-1);
            return -1;
        }

        if(entry_trust.getInt("trust")==int256(1)){
            transfer(pre_bill_id, new_bill_id,"bank",tran_amount);
            emit Financing(0);
            return 0;
        }

        emit Financing(-2);
        return -2;
    }
```

变量名	备注
pre_bill_id	传入参数 1，要融资的账单号
new_bill_id	传入参数 2，新建立的账单号
tran_amount	传入参数 3,银行融资的总额
entry_bt	找到账单表中的融资用户的账单，bt=bill table
entry_lender	找到账单表中的融资用户的账单，来获取该账单的欠款人
entry_ut	找到用户表中的欠款人的条目，ut=user table
entry_trust	找到用户表中的欠款人的条目，来获取是否受信用
pre_amount	查表后得到原账单总额

• **功能四：应收账款支付**

最后是功能四应收账款支付结算上链的实现函数，先打开账单表，然后新建一个 condition，调用 EQ 等于参数 bill_id 来选择账单号，调用 LE 小于等于参数 paytime 来比较表中的还款时间。如果表中有该账单 且借钱时间小于等于还款的账单，则还钱（删除）。这里面用到了 table 里的 remove 函数，若还款成功则删除账单。

```
/*
event Pay  功能四 应收账款支付结算：返回值，账单号，距离借钱日过去的时间（单位:天）
参数 ： bill_id ： 账单号     lendtime ： 距离借钱日过去的时间
*/

function pay(string bill_id, uint256 lendtime) public returns(int){
    Table table = openTableBill();

    Condition condition = table.newCondition();
    condition.EQ("bill_id", bill_id);
    condition.LE("paytime", int256(lendtime));

    int temp_pay = table.remove(bill_id, condition);

    emit Pay(temp_pay);
    return temp_pay;
}
```

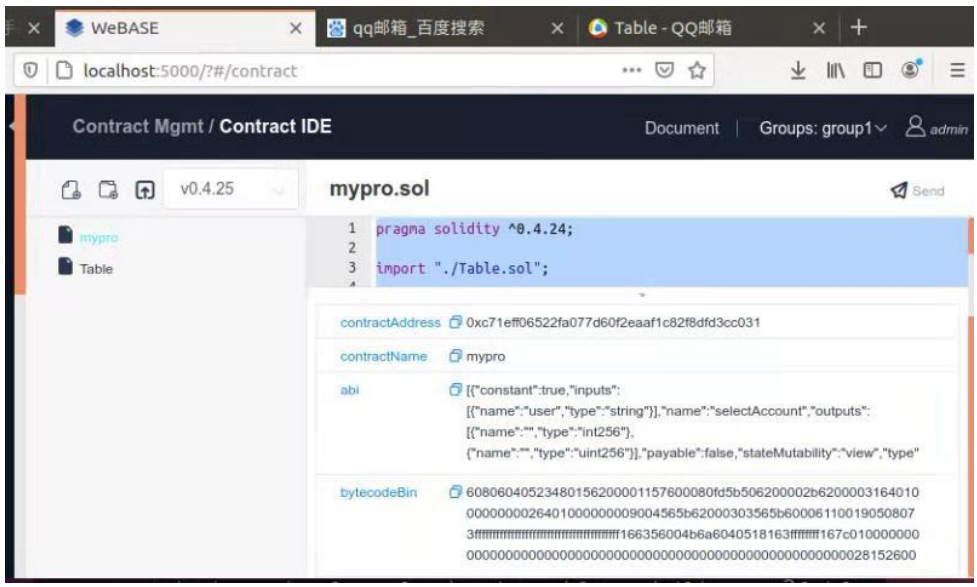
变量名	备注
bill_id	传入参数 1，要还款的账单号
lendtime	传入参数 2，距离借钱日过去的时间
paytime	表中设的还款时间，lendtime 需大于 paytime
temp_pay	表示还款后的返回结果

三、 功能测试

基于 WeBase 管理平台对以上介绍合约进行合约部署及功能调用。

1. 部署合约

合约是 mypro.sol 文件中。如图可以看到已经部署成功，需要先编译 table 文件。



2.注册用户

注册 5 个用户，分别为银行（bank）、汽车公司（BMWCompany）、轮胎公司（WheelCompany）、轮毂公司（HubCompany）、铝锭公司（AlpigCompany）、铝矿公司（AlCompany）。其中有信用的公司为银行、汽车公司，表现为 trust=1。不受信用的公司 trust=0。

①用户：bank

Trading content

data:

name	data
ret	0
user	bank
trust	1

②用户：BMWCompany

Trading content

data:

name	data
ret	0
user	BMWCompany
trust	1

③用户：WheelCompany

Trading content

ret	0
user	WheelCompany
trust	0

④用户：HubCompany

Trading content

data:

name	data
ret	0
user	HubCompany
trust	0

⑤用户：AlpigCompany

Trading content

data:

name	data
ret	0
user	AlpigCompany
trust	0

⑥用户：AlCompany

Trading content

name	data
ret	0
user	AlCompany
trust	0

⑦查询用户功能：选择 selectAccount 函数查询 BMWCompany，可以看到返回值为 0，信用值为 1。

Trading content

```
[
  0,
  1
]
```

3. 四个功能测试

• 功能一：签发账单

签发账单 bill1，车企从轮胎公司购买一批轮胎并签订应收账款单据。如图是交易成功后显示的界面，账单表项显示：账单号为 bill1，借钱者是轮胎公司，欠钱者是汽车公司，交易金额是一千万，要求还款时间是 365 天之后。

Trading content

data:

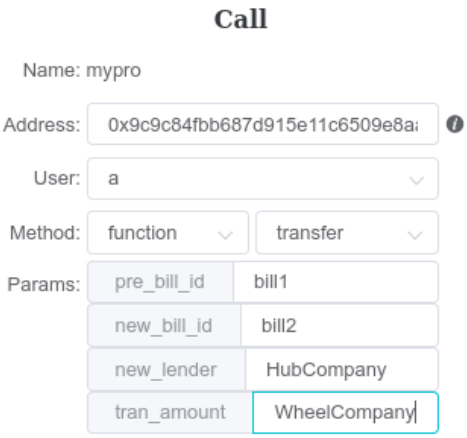
bill_id	bill1
lender	WheelCompany
borrower	BMWCompany
tran_amount	10000000
paytime	356

下面用账单查询功能验证，选择 selectBill 函数查询，输入账单号 bill1，结果如图，说明交易提交成功。

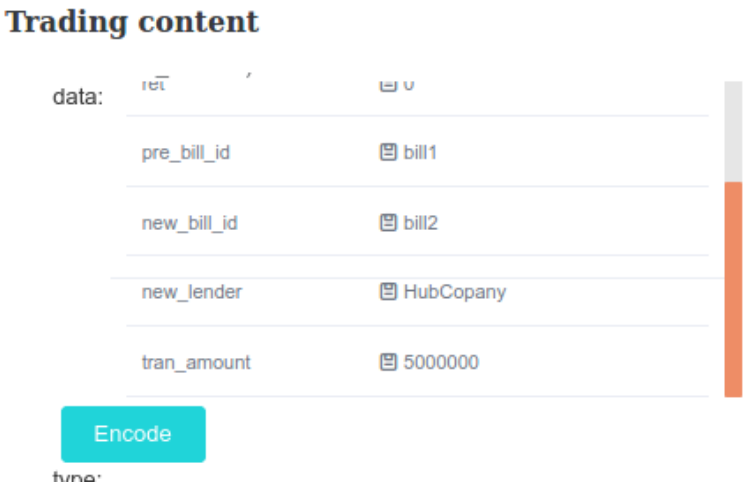
• 功能二：实现账单转让

轮胎公司从轮毂公司购买一笔轮毂，将于车企的应收账款单据部分转让给轮毂公司。轮胎公司凭借与受信用的车企的账单 bill1，转让 500 万给轮毂公司，轮毂公司成为 bill2 的债权持有人。

如图调用 transfer 函数，账单 bill1 转让成账单 bill2，新的借钱者是轮毂公司，转账金额 500 万。



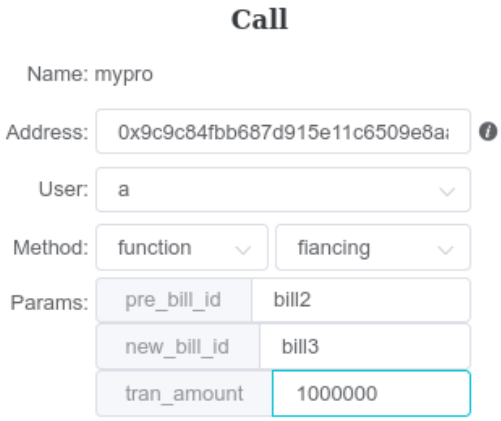
如下图是调用 transfer 后的结果，形成了一张新的账单，信息如下。



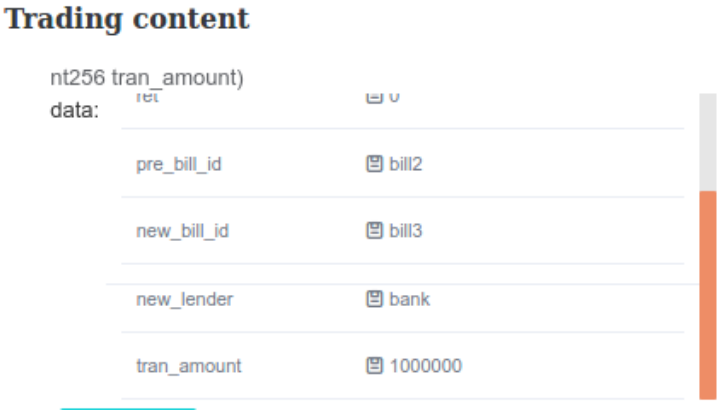
• 功能三：供应链公司凭应收账款单向银行融资

成为债权持有人后，轮毂公司凭借 bill2 向银行融资 100 万，融资成功。

如图是调用融资 fiancing 函数，轮毂公司本身不受信用，但是他有借钱给汽车公司的账单 bill2，而汽车公司受信用，故可以融资。如图，融资 100 万，新产生的账单号是 bill3。

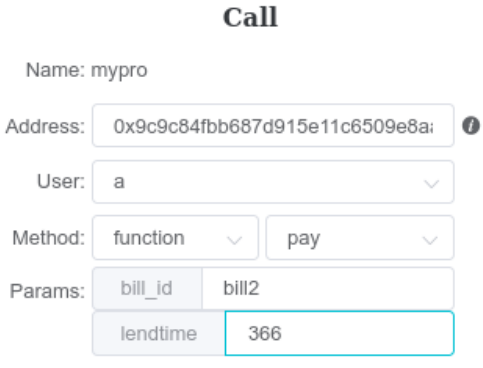


下图是融资成功后的显示结果，产生了新账单 bill3，新的借钱者是 bank，也就是说后面汽车公司要还这一百万给 bank。

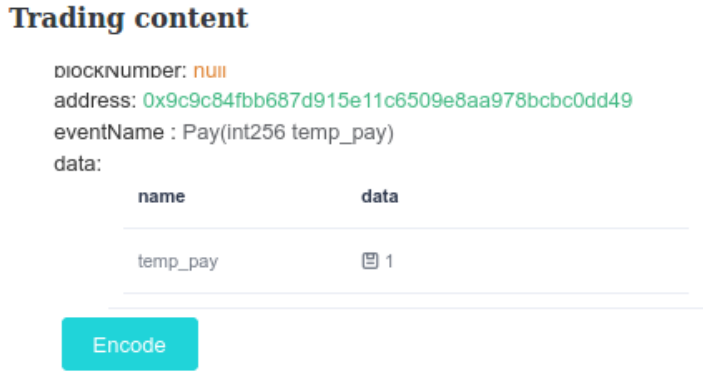


• 功能 4：核心企业还款

车企向下游企业还款。如图调用还款 pay 函数，要还钱的账单号是 bill2，也就是还给轮毂公司，借钱时间是 366 天之前。

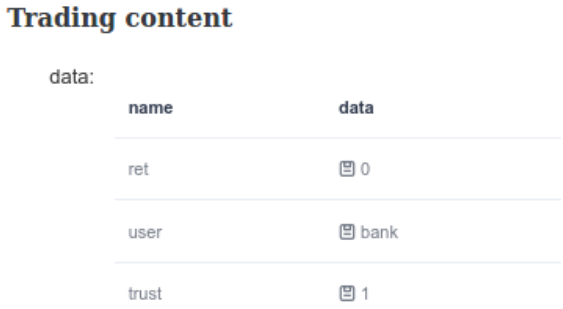


如图显示返回值为 1，表示还款成功。

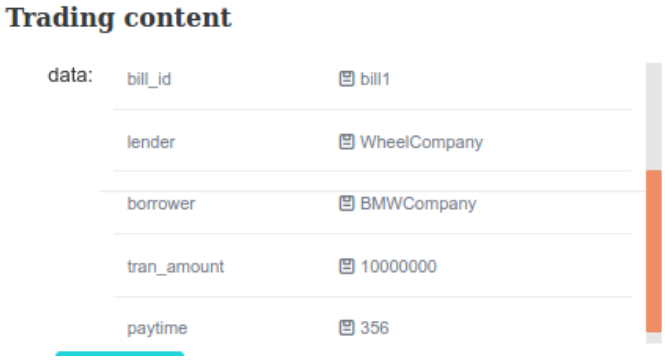


四、 界面展示

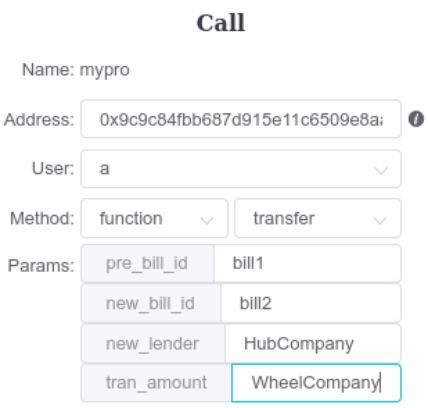
1. 用户注册



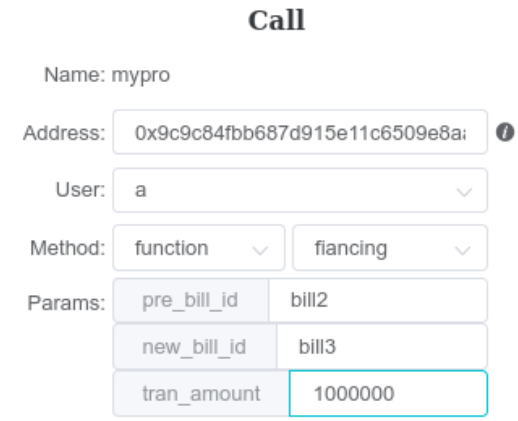
2. 签发账单



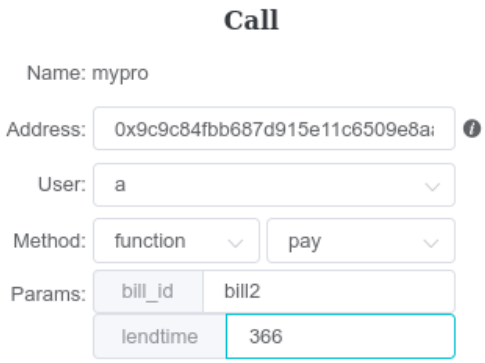
3. 账单转让



4. 融资



5. 还款



五、 心得体会

在这次实验过程中，我们小组不但学习到怎么配置 Fisco-bcos 链，还掌握了很多基于 Fisco-bcos 的应用开发的技巧。尽管大作业的实现难度从个人独立完成降到了小组合作完成，但由于大家都时间有限，因此最后的实现成果还是有一些缺陷。在大作业的三个阶段中，我们遇到了许多困难，譬如配置 Fisco-bcos 链、编写代码、搭建 Webase 等等，尽管花费了很多时间研究，但由于我们小组都是第一次接触 Fisco-bcos 方面的知识，因此实验刚开始时也没有什么思路，只能借助别人的帮助。我们通过在网上查阅了资料以及参考了一些同学的完成方式，最后基本成功解决困难。

总而言之，通过这次大作业，我们小组学到了很多区块链应用的知识。通过完成大作业设计，我们将课堂上学习到的或者课外查阅到的区块链知识转化为实践。曾经看起来很高深莫测的区块链，可以转换为我们生活中的一些应用场景，这大大地加深了我们对区块链的理解。经过一个学期的区块链课程学习，我们也对区块链的应用前景和发展潜力有了更多了解，今后也会持续关注这方面的发展动向。