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	17343051	姓名	黄治棋
电话	15626149800	Email	594447215@qq.com
开始日期	11 月 30 日	完成日期	12 月 13 日

一、 项目背景

在传统的供应链金融中，信任关系并不会向下游传递。比如银行信任车企的还款能力，则车企购买轮胎后，向轮胎公司签订的应收账款单据，可以作为凭证，直接让轮胎公司向银行借款。但这种信任关系不会向下游传递，也就是说，实质上银行并不信任轮胎公司的还款能力。如果轮胎公司购买轮毂，向轮毂公司签订的应收账款单据，则不可以作为凭证，直接让轮毂公司向银行借款（因为银行不认可轮胎公司的还款能力）。银行需要重新审查轮胎公司的还款能力，而这将增加很多经济成本。传统供应链金融的这个问题主要是由企业的信用无法在整个供应链中传递以及交易信息不透明化所造成的。

而本项目使用了区块链+供应链金融，可以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

存储设计：

这里用 Table.sol 来存储数据：

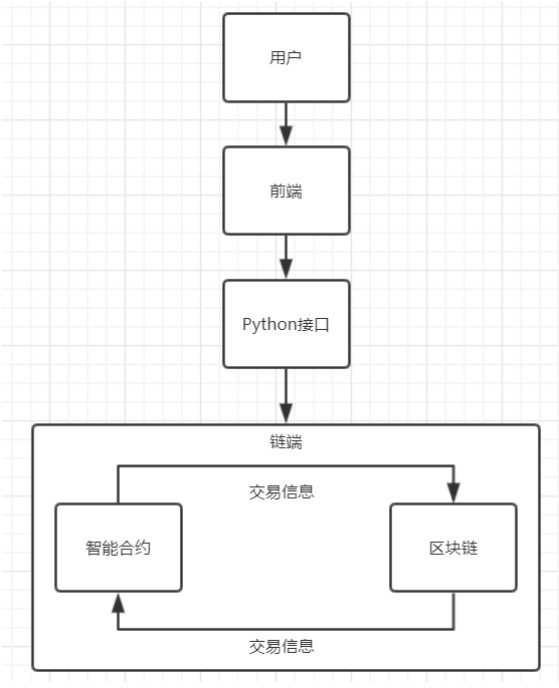
```
pragma solidity ^0.4.24;

contract TableFactory {
    function openTable(string memory) public view returns (Table);
    function createTable(string memory,string memory,string memory) public
}

contract Condition {
    function EQ(string memory, int) public;
    function EQ(string memory, string memory) public;

    function NE(string memory, int) public;
    function NE(string memory, string memory)  public;

    function GT(string memory, int) public;
    function GE(string memory, int) public;

    function LT(string memory, int) public;
    function LE(string memory, int) public;

    function limit(int) public;
    function limit(int, int) public;
}

contract Entry {
    function getInt(string memory) public view returns(int);
    function getAddress(string memory) public view returns(address);
    function getBytes64(string memory) public view returns(byte[64] memory)
    function getBytes32(string memory) public view returns(bytes32);
    function getString(string memory) public view returns(string memory);

    function set(string memory, int) public;
    function set(string memory, string memory) public;
}
```

具体而言，用 accountList 来储存用户账号，然后建立账户名和地址的两个映射 name_to_addr 和 addr_to_name，如图所示：

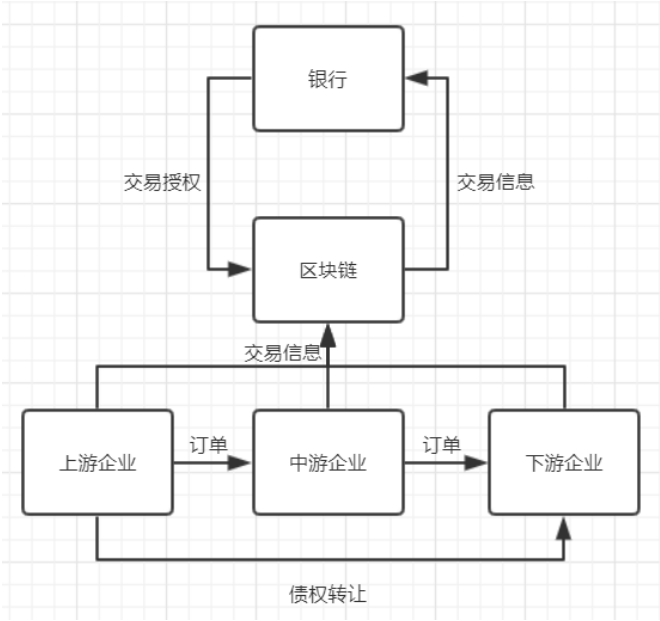
```
    string[] accountList;
    mapping(string=>address) name_to_addr;
    mapping(address=>string) addr_to_name;
```

数据流图：

图一：



图二：



核心功能介绍：

1，用户登录

通过地址进行登录，并返回结果。

```
//账户登录
function login(address addr) public view returns(int){
    if (name_to_addr[addr_to_name[addr]] != address(0)){
        return 1;//代表企业已注册
    }
    return 0;//代表企业未注册
}
```

2，资产注册

注册资产并返回是否注册成功。

```
//资产注册
function register(address addr, string memory account, uint256 value) public returns(int256){
    int256 infocode = 0;
    int256 ret = 0;
    uint256 temp = 0;
    // 查询账户是否存在
    (ret, temp) = select(account);
    if(ret == 0) {
        infocode = -1;//账户已存在
    } else {//账户不存在
        name_to_addr[account] = addr;
        addr_to_name[addr] = account;
        accountList.push(account);
        Table table = openTable();
        Entry entry = table.newEntry();
        entry.set("account", account);
        entry.set("value", int256(0));
        int count = table.insert(account, entry);
        if (count == 1) {
            infocode = 0;//注册成功
        } else {
            infocode = -2;//其他错误
        }
    }

    emit RegisterEvent(infocode, account, value);

    return infocode;
}
```

3，支付账单

把应收款清零，账户资产支付相应数值。

```
//支付账单
function pay() public returns(int256){
    Table table = openTable();
    for(uint i = 0; i < accountList.length; i++){
        Entry entry = table.newEntry();
        entry.set("account", accountList[i]);
        entry.set("value", int256(0));
        table.update(accountList[i], entry, table.newCondition());
    }
    emit PayEvent(0);
    return 0;
}
```

4，资产查询

通过账户名查询资产，若账户存在则返回资产，若账户不存在则返回-1。

```
//资产查询
function select(string memory account) public view returns(int256) {
    Table table = openTable();
    Entries entries = table.select(account, table.newCondition());
    if (0 == uint256(entries.size())) {
        return -1;//账户不存在，返回-1
    } else {
        Entry entry = entries.get(0);
        return uint256(entry.getInt("value"));//账户存在，返回资产金额
    }
}
```

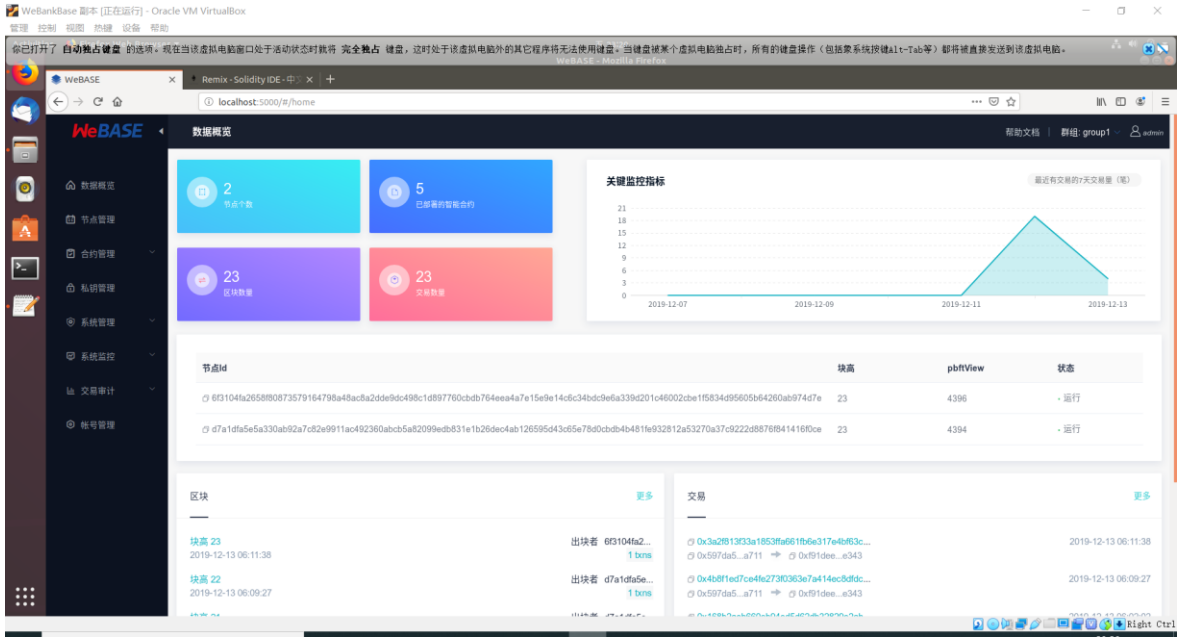
5，债权转让

在用户间转移资产，有详细考虑异常情况，避免转移/接收账户不存在、账户资产不足等问题。

```
//账户间转移资产
function transfer(address addr, string memory to_account, uint256 amount) public returns(int256) {
    string memory from_account = addr_to_name[addr];
    int infocode = 0;
    int256 ret = 0;
    uint256 from_value = 0;
    uint256 to_value = 0;

    (ret, from_value) = select(from_account);
    if(ret != 0) {// 转移账户不存在
        infocode = -1;
        emit TransferEvent(infocode, from_account, to_account, amount);
        return infocode;

    }

    (ret, to_value) = select(to_account);
    if(ret != 0) {//接收账户不存在
        infocode = -2;
        emit TransferEvent(infocode, from_account, to_account, amount);
        return infocode;
    }

    if(from_value < amount) {// 转移资产的账户金额不足
        infocode = -3;
        emit TransferEvent(infocode, from_account, to_account, amount);
        return infocode;
    }

    Table table = openTable();
    Entry entry0 = table.newEntry();
    entry0.set("account", from_account);
    entry0.set("value", int256(from_value - amount));
    int count = table.update(from_account, entry0, table.newCondition());
```

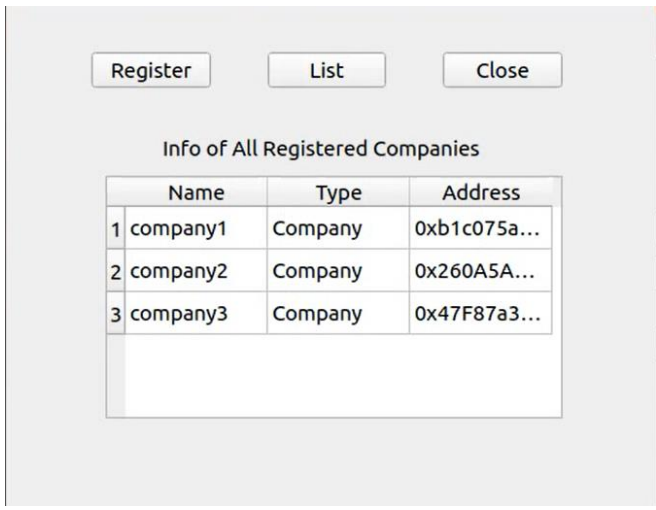
三、 功能测试

使用的是 FISCO-BCOS 链：

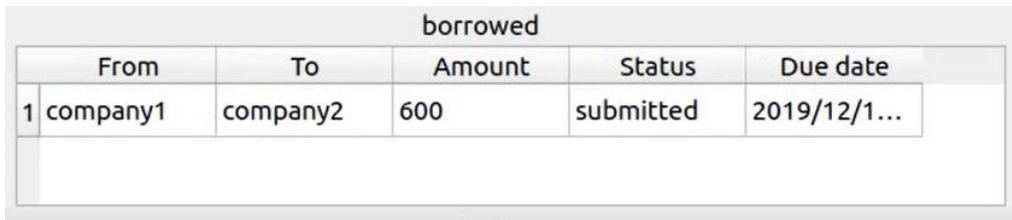


测试流程（演示视频流程）

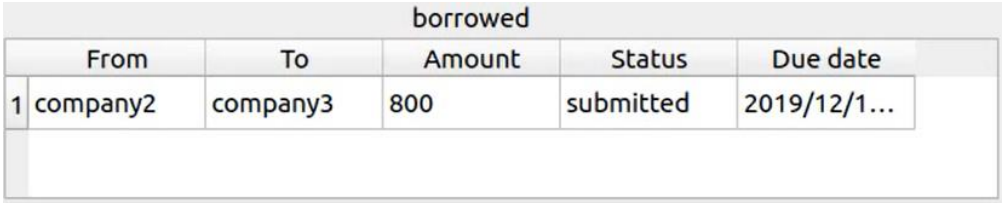
1，建立三家公司：company1，company2，company3



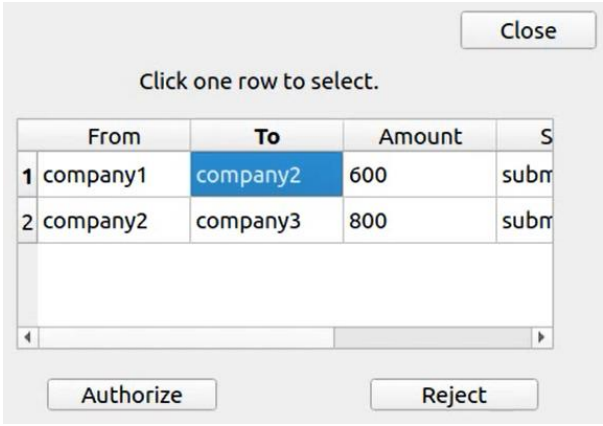
2，company1 向 company2 下订单，订单金额为 600



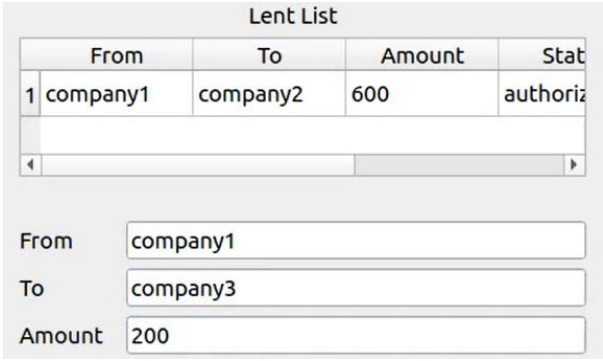
3，company2 向 company3 下订单，订单金额为 800



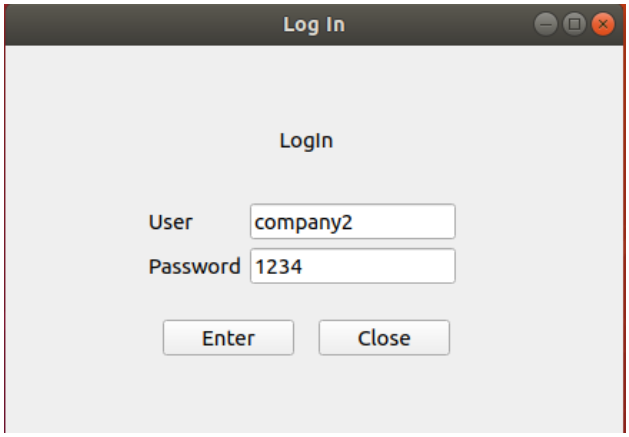
4，金融机构确认这两笔交易

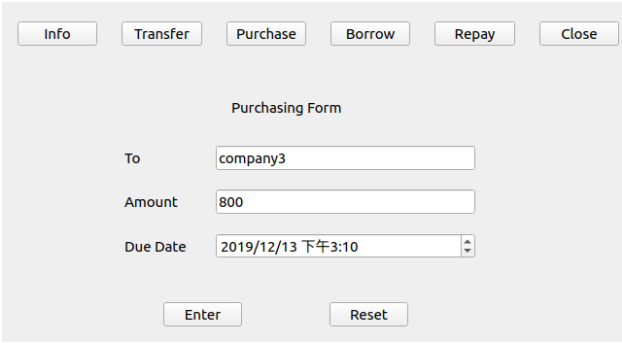


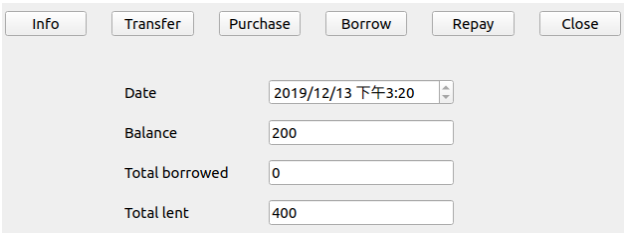
5，company2 将对 company1 的债权转移 200 到 company3



四、 界面展示







水平有限，做的比较朴素。

五、 心得体会

这次作业过程中我学习了许多知识，了解了 FISCO-BCOS 区块链平台，学习了 WeBase 的配置方法，学习了智能合约的编写过程，实践了使用 SDK 开发区块链应用的过程，还搭建了比较朴素的前后端。

这次遇到的困难主要有两个：
一是使用 Python-SDK 时也一直有各种环境上的问题，感谢群里各位大佬的热心解答，解决了很多我遇到的困难。但最后仍有一些问题会出现，文档也有一点不完整，所以只能退而求其次，最后实现的功能较为朴素简单。

二是在于实现前后端，因为之前从来没有学习过相关知识，所以对前后端的概念十分陌生。尽管使用了 PyQt5 框架，但对框架的熟悉程度不高，无法解决一些前端与后端、链端数据交互的问题，前后端稳定性不佳，经常会在使用中崩溃。所以最后演示中也只能演示部分简单的功能。

感觉自己代码能力有限，在许多困难无法解决的情况下，用了最大努力来尽量完成作业要求，希望 TA 给分手下留情 =_=

总的来说，这次学习让我接触了很多新鲜的知识和领域，包括区块链、前后端、SDK 等，十分感谢老师和 TA 的辛苦付出！

Github 网址：https://github.com/BETARUN/BlockChain