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	17243086	姓名	梅诗博
电话	13246848221	Email	1617308410@qq.com
开始日期	2019/12/07	完成日期	2019/12/13

一、 项目背景

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。

二、 方案设计

前端：负责转账，注册，登陆，查询联盟链上所有的企业，查询企业的相关信任额度，查询所有转账记录的UI交互。

后端：采用`Fisco bcos`的Nodejs SDK，处理前端的请求，调用，部署智能合约的相关函数，将结果返回给前端。

链端：编写相关的智能合约。

后端功能展示：

- 后端维护的数据库功能用来记录企业的ID，信用额度，余额。虽然余额可以在智能合约上记录，但是一些企业信息较为复杂，所以依然在后端数据库上采用了备份的策略。交易上链，立即将相关的信息导入数据库，客户在查询相关信息，直接由后端调用数据库返回。

```
//代码的解释写在注释中
//这是数据库CRUD操作中的插入功能实现
if (url_info.pathname == '/insert' && req.method == "POST"){
      var str = "";//接收数据用用
      req.on('data',function(data){
          str += data;
      });
      req.on('end', () => {
          //查询数据库表
          crudService.desc("t_assets").then(tableInfo => {
              var obj = JSON.parse(str)
              //按照查询的表的数据建表，方便后续issert数据
              let table = new Table(tableInfo.tableName, obj.account,
tableInfo.valueFields, tableInfo.optional);
              //取得前端的post请求数据
              let fieldNames = tableInfo.valueFields.split(',');
              let fieldValues = obj["values"].split(',');
              if (fieldNames.length !== fieldValues.length) {
```

```
                throw new Error(`unmatch number of fields, expected ${fieldNames.length} but got ${fieldValues.length}`);
            }
            let entry = new Entry();
            for (let index in fieldNames) {
                entry.put(fieldNames[index], fieldValues[index]);
            }
            //插入数据到t_assets数据库表
            crudService.insert(table, entry).then(value=>{
                console.log(value) if(value == "1"){
                    console.log("insert successfuuly") } });
        });
    })
}
```

- 后端部署以及调用智能合约功能

```
var arr = new Array("iamhello");
    //deploy部署智能合约，在then的异步操作中调用智能合约
    api.deploy("../nodejs-sdk/packages/cli/contracts/assets.sol","../nodejs-sdk/packages/cli/contracts").then(value=>{
        console.log("success!");
        //参数类型数组作为合约函数的接收值,这里省略了前端post body的值的类型转换和赋值
        var arr = new Array("string", "uint", "uint", "uint", "uint" , "uint");
        //deploy会返回智能合约的地址，因此部署和调用实际上并不是一步操作。可以先通过部署再调用
        //这里为了介绍的简便性，采用了前期的写法，这种写法可以完成操作，但是性能较差
        //调用智能合约。智能合约地址由deploy的异步返回value.contractAddress来取得
        //不能用call调用，call只读不改
        api.sendRawTranscation(

value.contractAddress,"makedeal(string,uint,uint,uint,uint,uint)",
            arr).then(returnvalue=>{
            var returntype = new Array("bool");
            //这里需要调用utils里面的decodeParams解析二进制的智能合约调用返回，用returnType接受            //合约return值，这里省略了返回给前端的post
            console.log(utils.decodeParams(returntype, returnvalue.result.output));
        }).catch(err=>{
        });
    }).catch(err=>{
        console.log("ERROR2:");
        console.log(err);
    });
```

链端功能展示:

- 智能合约代码

```
pragma solidity ^0.4.21;

contract final{
    address public bank;
    mapping (address => uint) public balances;
    address public trust;
```

```
    struct Page{
        string name;
        uint start_time;
        uint range;
        address c1;
        address c2;
        uint money;
    }

    Page[] public page;
    PayList[] public list;

    constructor(){
        bank = msg.sender;
        balances[msg.sender] = 100000;
    }

    function addMoney(uint amount) {
        balances[msg.sender] += amount;
    }

    function makedeal(string name, uint now_time, address t, address v, uint
amount, uint range) returns(bool){
        if (v != bank) return false;
        Page memory p = Page(name, now_time, range, msg.sender, t, amount);
        page.push(p);
        return true;
    }

    function load(address t, uint amount, uint now_time) returns (bool){
        if (t != bank) return false;
        address a;
        address b;
        b = msg.sender;
        for (uint i = page.length - 1; i >= 0; -- i){
            if (page[i].c2 == b){
                a = page[i].c1;
                if (a == trust) {
                    if (page[i].start_time + page[i].range > now_time &&
page[i].money > amount){
                        balances[bank] -= amount;
                        balances[msg.sender] += amount;
                        page[i].money -= amount;
                        PayList memory pl = PayList(t, msg.sender, amount,
page[i].start_time + page[i].range);
                        list.push(pl);
                        return true;
                    }
                }
                else {
                    b = a;
                    i = page.length;
                }
            }
        }
        return false;
    }
}
```

面主要介绍一些变量和一些函数的功能：

- bank

  银行对象

- balances

  映射关系

- trust

  银行信用高的企业，实际上每个企业都会有一定的信任额度（视频有展示）

- page

  交易结构体

- function makedeal(string name, uint now_time, address t, address v, uint amount, uint range)

  交易函数

- function load(address t, uint amount, uint now_time) returns (bool)

  贷款函数，t是要贷款的银行，这里只有一个bank，amount是贷款金额，判断是否能接足够金额给其企业时使用，now_time是现在的时间，存在必要性是每次进行贷款都要判断其信任企业的协议已经过期，如果过期不能贷款。

三、功能测试&界面展示


深证通信
WeBank
微众银行
Tencent
亦笔科技
YIBI Technology
HUAWEI
FISCO BCOS
超图金控
四方精创
FORMS SYNTRON
神州信息
DCITS
博彦科技
BEYONDSOFT

App启动界面

注册登录企业


11:09
LTE
Fisco bcos
SYSU
确认注册
您的公钥是：
1A1zP1eP5QGefNafi2DMPTfTL5SLmv7Div
请妥善保管您的私钥：
8F723C7C35D83E5E5F6B29E6E225A36B68DF

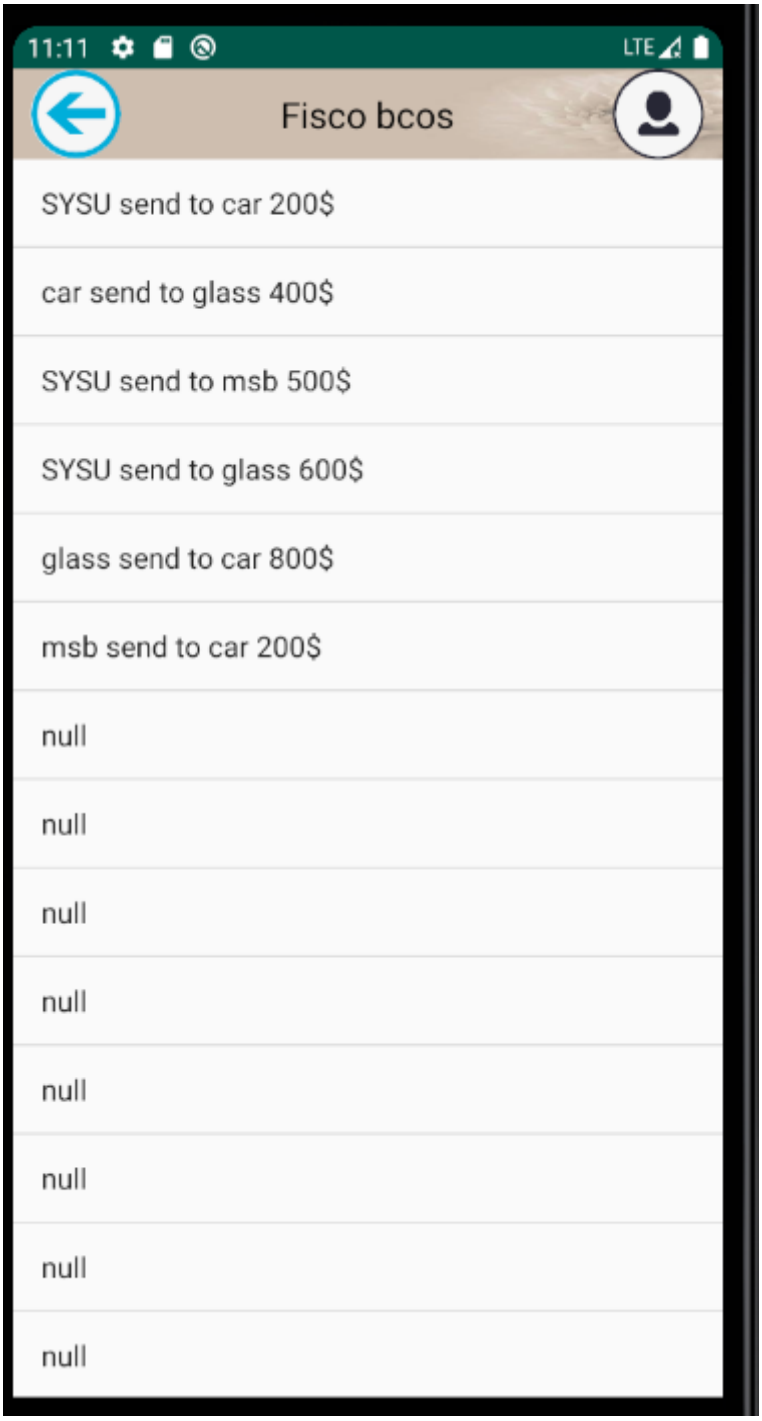
ID注册，返回公钥私钥

登陆

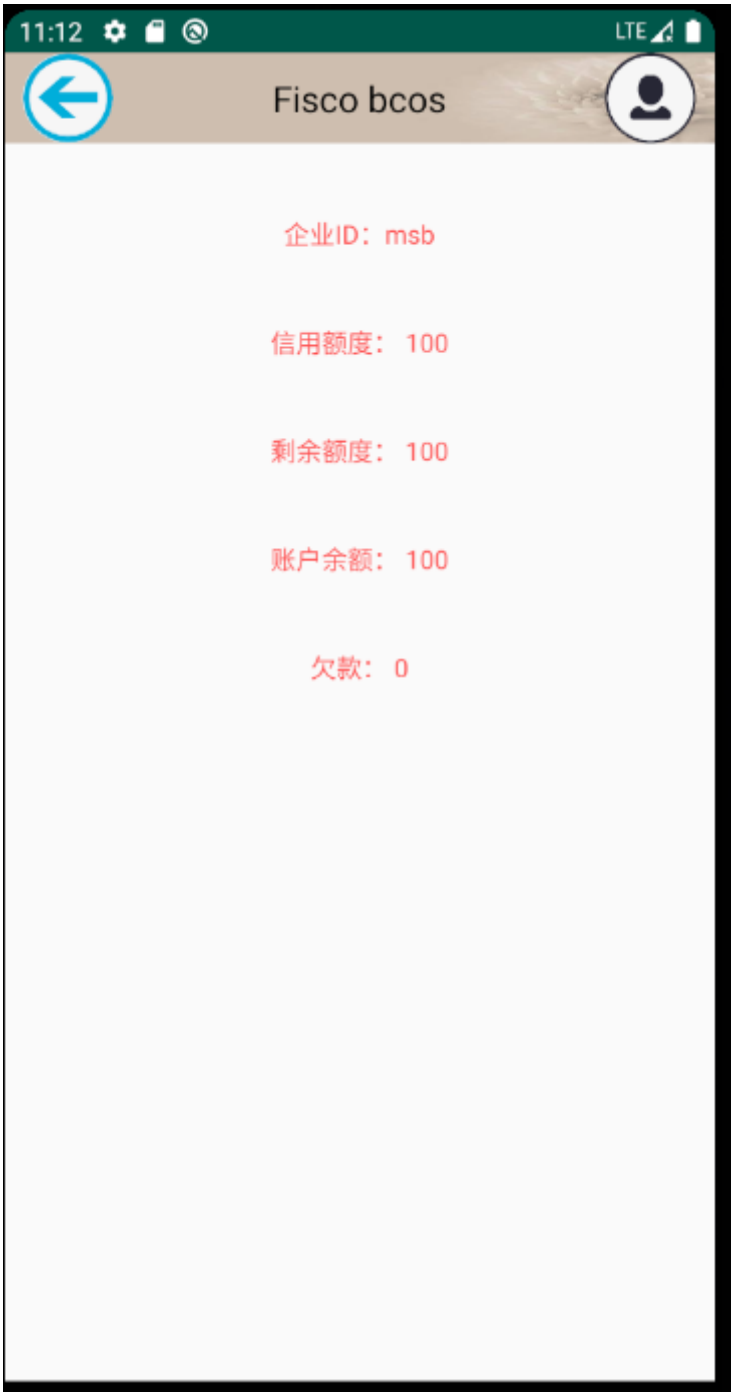
前端功能


11:11
LTE
Fisco bcos
企业ID： msb 企业信用剩余额度: 100
企业ID： glass 企业信用剩余额度: 300
企业ID： car 企业信用剩余额度: 500
null
null
null
null
null
null
null
null
null
null
null

查看联盟链企业


11:11
LTE
Fisco bcos
SYSU send to car 200$
car send to glass 400$
SYSU send to msb 500$
SYSU send to glass 600$
glass send to car 800$
msb send to car 200$
null
null
null
null
null
null
null
null

查看所有交易

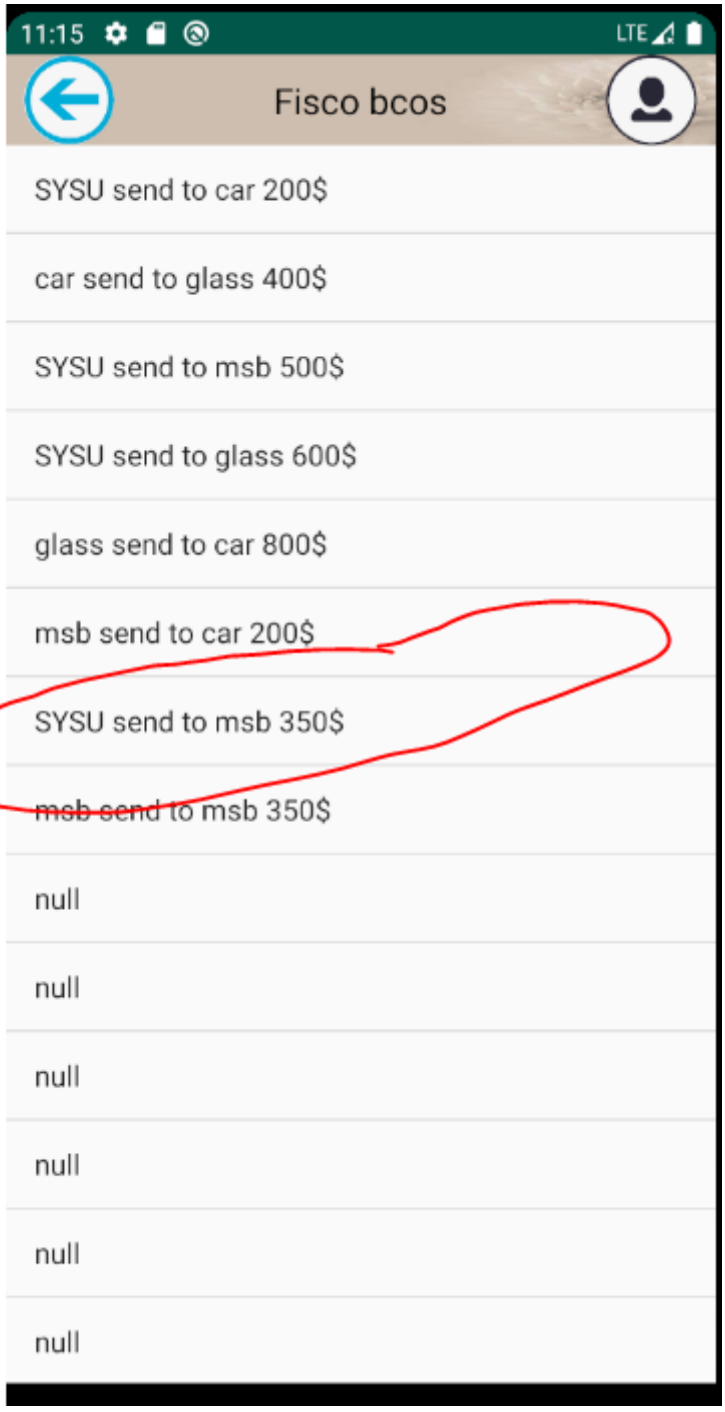


个人企业信息

转账交易

贷款额度+余额不够支付，交易失败

转账成功，余额减少



转账成功，记录上链

其他更多详细测试请看视频

四、实验心得

通过这次实验，我收获了很多区块链实践的知识，这也是这门课这学期最大的收获了。从第一节课老师说起来区块链大作业，内心慌得不行，感觉这东西离自己很遥远。上了大半学期的课，区块链也慢慢的清晰了起来，从智能合约到Fisco框架，开始认识到区块链的重要性和厉害之处，相信未来肯定会有区块链的大舞台。