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	大三(17 级)	专业（方向）	软件工程
学号	17343115	姓名	温卓沛
电话	13680426099	Email	1421166369@qq.com
开始日期	2019.11.11	完成日期	2019.12.13

一、 项目背景

区块链+供应链金融应用：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

1. 实现功能

- 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。

2. 具体设计思想与实现

1) 存储设计

合约的实现主要基于两大结构体，分别对应公司个体和账单：

```
struct company {
    address company_name;
    string company_address;
    string company_nature;
}

struct receipt {
    address from;
    address to;
    uint256 amount;
    string status;
    string time;
}
```

- 上述 company 结构体中内含有其名称(公钥)、公司地址和待用于扩展的 company_nature 公司的类型。
- 上述 company 结构体中内含有其名称(公钥)、公司地址和待用于扩展的 company_nature 公司的类型。
- 并使用全局定义数组进行数据存储：

```
receipt[] public receipts;
company[] public companies;
```

2) 功能实现

- 为方便从链上合约获取返回的数据，此阶段将阶段二中的合约中的各个合约方法的返回值(原更大用处用作人工检验功能效果)更改为单 bool 返回类型以便给予调用者信号该操作是否已实现。
- 此外，考虑到若是一个公司没有注册便在加入链后能操作交易事项是较为不合理的，因此多加了一层在本链中需要进行注册记录，此做法可作为安全作考虑：

```
/*
描述 : 公司(银行)注册
参数 :
        company_address  : 公司地址
        company_nature   : 公司性质(暂自定义)
返回值:
        int256  :   0  公司注册成功  -1 公司账户已存在
        address :   公司名称(公钥)
        string  :   公司地址
        string  :   公司性质(暂自定义)
```

```
11.
12. */
13. function register(address addr, string company_address, string company_nature) returns(b
    ool) {
14.   bool ret = false;
15.   if (isCompanyRegistered(addr)) {
16.     emit RegisterEvent(-1, addr, company_address, company_nature);
17.     ret = false;
18.   }
19.   company memory tmp;
20.   tmp.company_address = company_address;
21.   tmp.company_name = addr;
22.   tmp.company_nature = company_nature;
23.
24.   companies.push(tmp);
25.
26.   ret = true;
27.   emit RegisterEvent(0, addr, company_address, company_nature);
28.   return ret;
29. }
```

- 求中判断是否已注册操作：

```
1.  function isCompanyRegistered(address company_name) returns(bool) {
2.    for (uint i = 0; i < companies.length; i++) {
3.      if (companies[i].company_name == company_name) {
4.        return true;
5.      }
6.    }
7.    return false;
8.  }
```

- 对于功能一

  判断双方是否已注册：

```
1.  if (!isCompanyRegistered(to_company)) {
2.        emit PurchaseEvent(-1, from_company, to_company, amount, time);
3.        return false;
4.      }
5.      if (!isCompanyRegistered(from_company)) {
6.        emit PurchaseEvent(-2, from_company, to_company, amount, time);
7.        return false;
8.      }
```

若满足条件则直接添加数据即可：

```
1.  receipt memory tmp;
2.      tmp.from = from_company;
3.      tmp.to = to_company;
4.      tmp.amount = amount;
5.      tmp.time = time;
6.      tmp.status = "F";
7.
8.      receipts.push(tmp);
9.
10.     emit PurchaseEvent(0, from_company, to_company, amount, time);
11.     return true;
```

- 对于功能二

  实现应收账款的转让上链：

  遍历 receipts，找寻出上链单位(公司)和己方公司的账单：

```
for(uint i = 0; i < receipts.length; i++) {
 if(receipts[i].to == msg.sender && receipts[i].from == from_company) {
    // ...
  }
}
```

  再次遍历 receipts，招寻处下链单位(公司)和己方公司的账单：

```
for(uint j = 0; j < receipts.length; j++) {
   if(receipts[j].from == msg.sender && receipts[j].to == to_company) {
      // ...
   }
}
```

  进行转账金额合理性判断并作入链或打回操作：

```
if (receipts[i].amount < amount || receipts[j].amount < amount) {
  emit TransferEvent(-1, pass_company, pass_company, amount, time);
  return false;
} else {
  receipts[i].amount -= amount;
  receipts[i].time = time;
  receipts[j].amount -= amount;
  receipts[j].time = time;

  receipt memory tmp;
  tmp.from = from_company;
  tmp.to = to_company;
  tmp.amount = amount;
  tmp.time = time;
  tmp.status = "F";

  receipts.push(tmp);

  emit TransferEvent(0, from_company, to_company, amount, time);
  return true;
}
```

- 对于功能三

  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(**即对所有与本公司有关的账单进行遍历，然后统计出盈亏，进行融资申报——正则得，负则打回：**)：

  判断本公司或银行是否注册：

```
// 判断我方公司是否已注册
if (!isCompanyRegistered(company_address)) {
  emit FinanceEvent(-2, company_address, bank_address, 0, time);
```

```
    return false;
  }

  if (!isCompanyRegistered(bank_address)) {
    emit FinanceEvent(-1, company_address, bank_address, 0, time);
    return false;
  }
```

判断本公司是否“负债”：

```
  // 判断是否欠债
      int256 receiptSum = 0;
      for (uint i = 0; i < receipts.length; i++) {
        if (receipts[i].from == company_address) {
          receiptSum = receiptSum - (int256)(receipts[i].amount);
        } else if (receipts[i].to == company_address) {
          receiptSum = receiptSum + (int256)(receipts[i].amount);
        }
      }

      if (receiptSum < 0) {
        emit FinanceEvent(-
  3, company_address, bank_address, (uint256)(receiptSum), time);
        return false;
      }
```

最后入链：

```
  receipt memory tmp;
  tmp.from = company_address;
  tmp.to = bank_address;
  tmp.amount = (uint256)(receiptSum);
  tmp.time = time;
  tmp.status = "T";

  receipts.push(tmp);

  emit FinanceEvent(0, company_address, bank_address, (uint256)(receiptSum), time);
  return true;
```

- 对于功能四

  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：

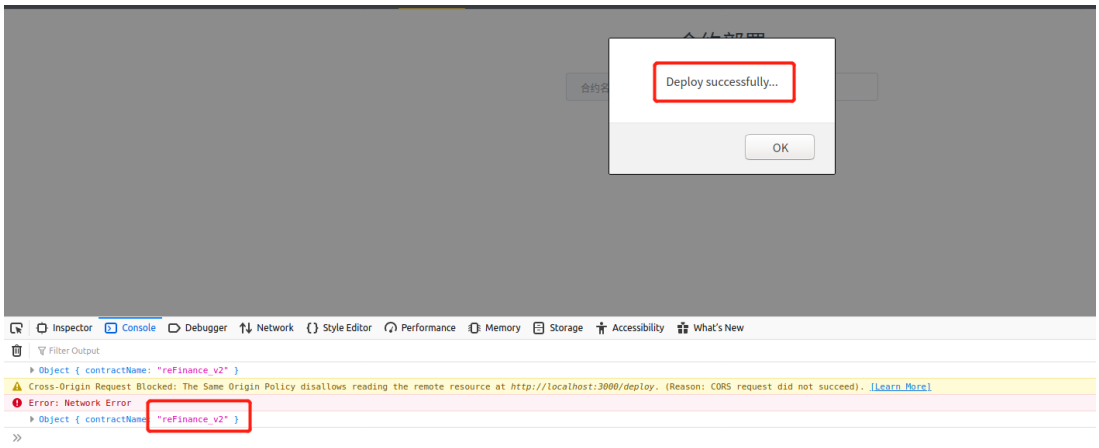
  找寻本公司作为欠款方的欠款账单，然后进行款额合理性判断，最后根据判断决定入链或打回：

```
  function payOff(address from_company, address to_company, uint256 amount, string time) r
  eturns(bool) {
      for (uint i = 0; i < receipts.length; i++) {
        if (receipts[i].from == from_company && receipts[i].to == to_company) {
          if (receipts[i].amount >= amount) {
            receipts[i].amount = receipts[i].amount - amount;
            emit PayOffEvent(0, from_company, to_company, amount, time);
            return true;
```

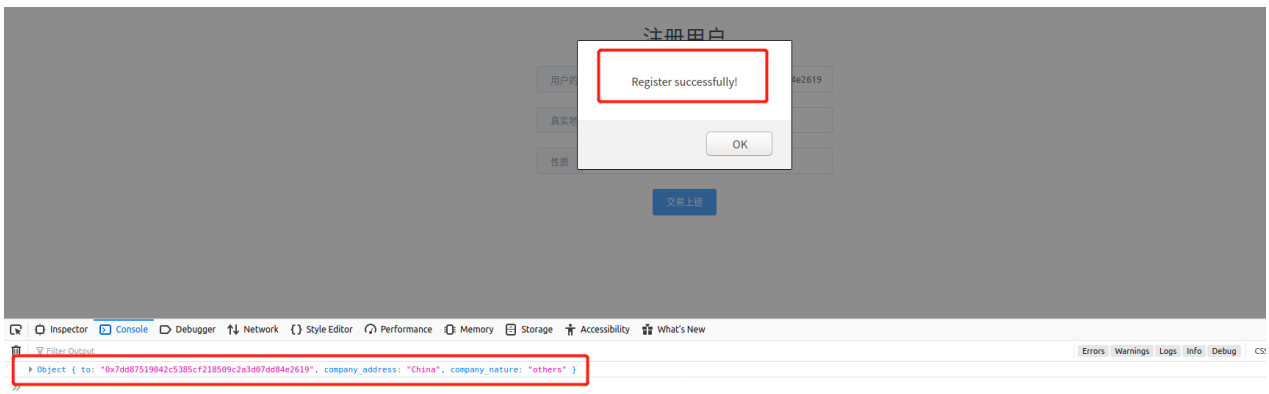
```
        } else {
          emit PayOffEvent(-1, from_company, to_company, amount, time);
          return false;
        }
      }
    }
    emit PayOffEvent(-2, from_company, to_company, amount, time);
    return false;
  }
```

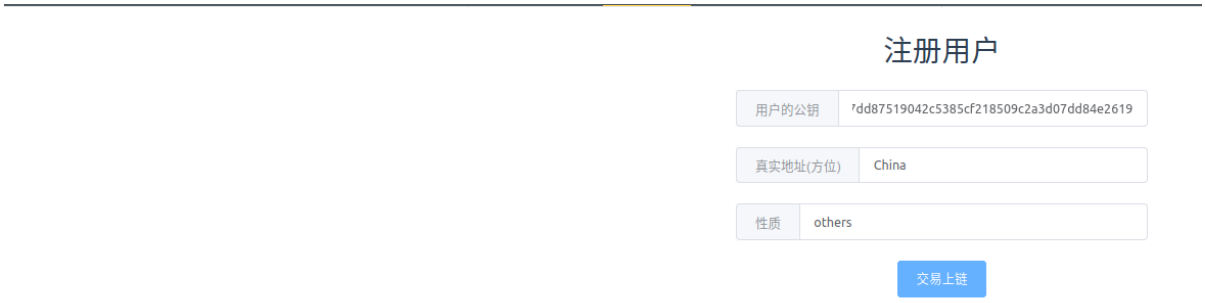
三、 功能测试

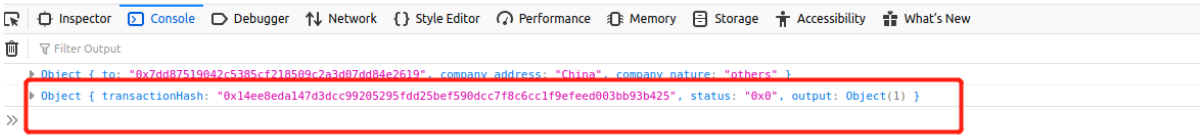
合约部署：



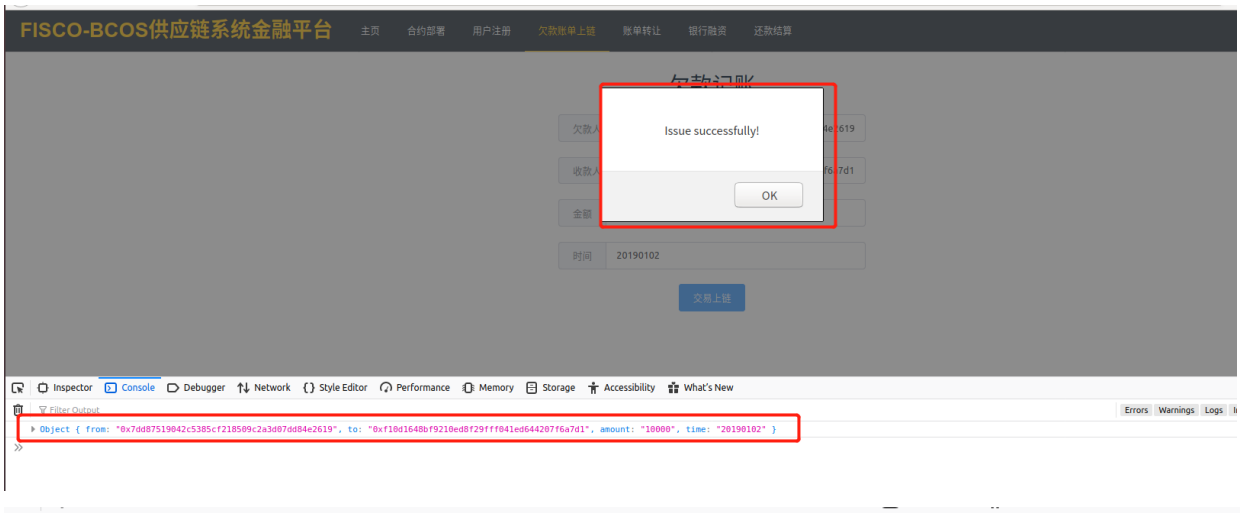
用户注册：



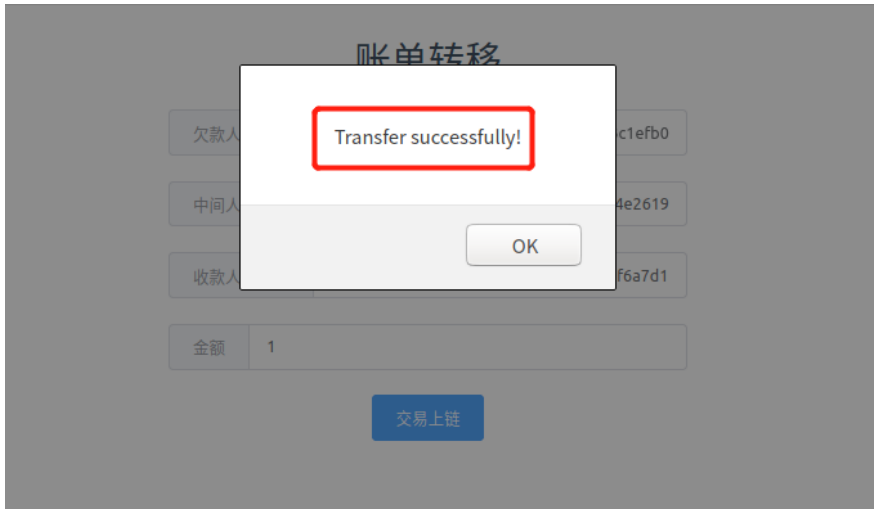




功能一，记账上链：

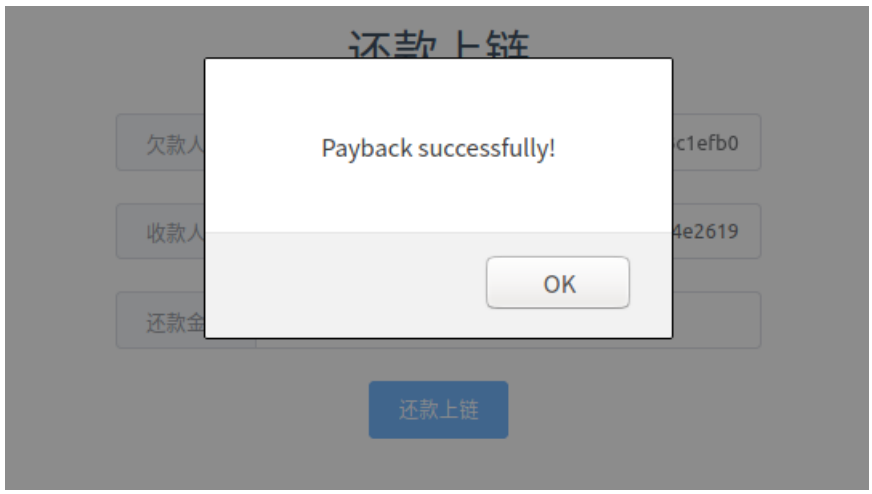


功能二：账单转让



功能三：融资

功能四：还款



四、 界面展示

主界面如下：

其余页面由于时间问题没有截图然后在此处展示，可通过运行代码进行相关程序代码实际观看。

五、 心得体会

本次区块链大作业任务是基于链端+后端+前端的实验，本人认为在前端的实现相对较为独立，并以此为开端，但是各式各样的框架也是初步接触，因此在实现的时候较为困难，而后在查阅众多资料后使用了 Vue+ElementUI 对相关前端功能加以实现。

此外，由于自己开发能力有限，且整体时间相对紧张，因此我和舍友王梓鉴同学（学号 17343112）共同完成的本次前后端的开发，我主要负责前端功能设计开发，王梓鉴同学主要负责后端功能设计开发，因此可以说是我们共同完成了本次实验前后端的开发任务(链端功能基于上次的实现而进一步实现)，并在设计实现之后分别对我们各自的合约进行部署和相关功能操作，希望可以理解。