

区块链期末作业

罗逸群 17343083

一、项目背景

某车企(宝马)因为其造车技术特别牛,消费者口碑好,所以其在同行业中占据绝对优势地位。因此,在金融机构(银行)对该车企的信用评级将很高,认为他有很大的风险承担的能力。在某次交易中,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1年后归还轮胎公司1000万。这个过程可以拉上金融机构例如银行来对这笔交易作见证,确认这笔交易的真实性。在接下里的几个月里,轮胎公司因为资金短缺需要融资,这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款,金融机构认可该车企(核心企业)的还款能力,因此愿意借款给轮胎公司。但是,这样的信任关系并不会往下游传递。在某个交易中,轮胎公司从轮毂公司购买了一批轮毂,但由于租金暂时短缺向轮胎公司签订了500万的应收账款单据,承诺1年后归还轮胎公司500万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候,金融机构因为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以评估其还款能力同时验证应收账款单据的真实性,才能决定是否借款给轮毂公司。这个过程将增加很多经济成本,而这个问题主要是由于该车企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导致的。区块链+供应链金融: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。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。

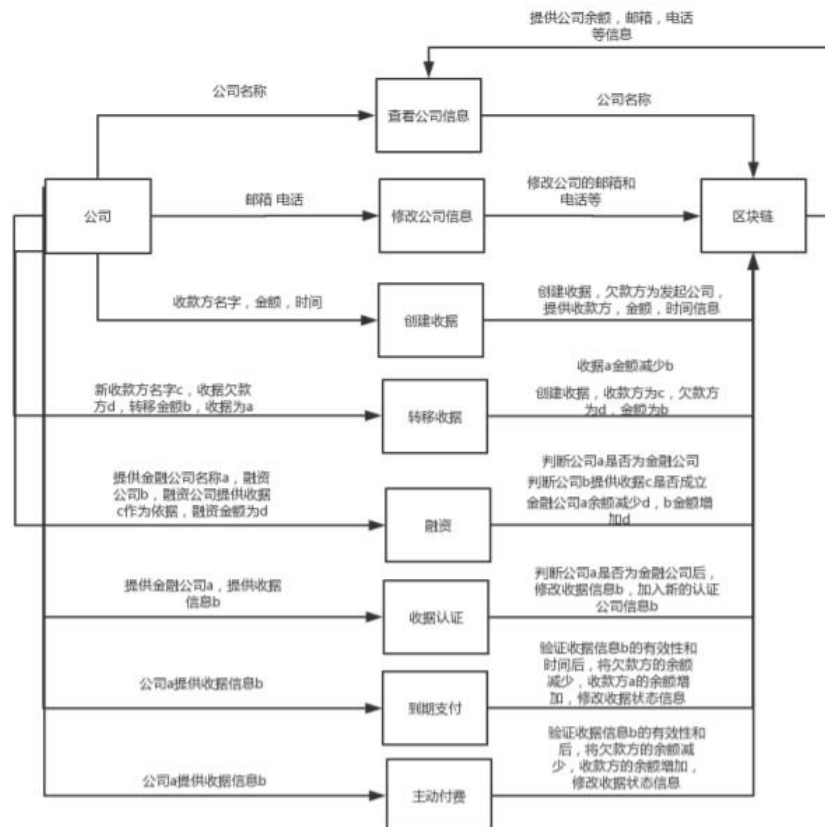
功能四:应收账款支付结算上链,应收账款单据到期时核心企业向下游企业支付相应的欠 款。

二、 方案设计

存储设计：

存储的信息主要是两类：公司信息+收据信息。存储使用的是 Table.sol + 合约映射存储的方式。利用 Table.sol 开启数据库，数据库中主要存放公司的信息如账户地址，账户名字，密码，联系方式等信息，其中让公司的账户作为主键。当公司尝试登录网页的时候，就通过查询该库判断密码是否正确。在创建账户的时候可以认为在后台也创建了对应的私钥保存起来，这样在发起交易的时候就可以通过后台的私钥判断是否为公司操作，方便认证公司等功能。而合约存储主要是存储创建收据的信息，如收据的欠款方收款方金额时间是否已经支付等信息，在需要调用这些信息的时候就通过 map 获取即可。

数据流图



核心功能介绍

- 查看公司信息：**该功能可以查看当前公司的信息如余额，联系方式等功能。其实现方式是通过查看链端数据库的公司信息来获取。在获取前通过公司的账号信息获取对应的链端账号地址，然后再由该账号地址去查询表格。

```

function createTable() private {
    TableFactory tf = TableFactory(0x1001);
    tf.createTable("t_company", "name", "addr,balance,cType,password,email,phone");
    //key: address
    //cType 0:financial company like bank, 1:ordinary compnay
}

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

function getCompany(address addr) public returns (string,uint,uint,string,string,string){
    Table table = openTable();
    Entries entries = table.select(addr, table.newCondition());
    Entry e = entries.get(0);
    return
    (e.getString("name"),e.getInt("balance"),e.getInt("cType"),e.getString("email"),e.getString("password"),
    e.getString("phone"));
}

```

- 修改公司信息：该功能可以修改当前公司的信息如联系方式等，其实现方式是通过修改数据库的公司信息来实现的。

```

function createTable() private {
    TableFactory tf = TableFactory(0x1001);
    tf.createTable("t_company", "name", "addr,balance,cType,password,email,phone");
    //key: address
    //cType 0:financial company like bank, 1:ordinary compnay
}

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

function setCompany(string _name,string _phone,string _email) public{
    Table table = openTable();
    Entries entries = table.select(msg.sender, table.newCondition());
    Entry e = entries.get(0);
    e.set("phone",_phone);
    e.set("mail",_email);
    int count = table.update(_name, en,table.newCondition());
    emit UpdateResult(count);
}

```

- 创建收据：

创建收据由欠款方公司发起，提供的信息包括收款方的名字，金额与时间，然后就会在合约内调用 `addReceipt`，通过 `map` 存储对应的收据信息。这样就可以保证不会任意公司都可以创建收据作为收据方的收款方，因此发起收据的一定是欠款方。

```
function addReceipt(string _name,address _to,uint _count, uint _expiredTime) public{
    mapR[mapSA[_name]].push(Receipts({
        from:mapSA[_name],
        to:_to,
        count:_count,
        expiredTime:now + _expiredTime,
        status:0,
        check: new string[](1)
    }));
    mapR[_to].push(Receipts({
        from:mapSA[_name],
        to:_to,
        count:_count,
        expiredTime:now + _expiredTime,
        status:0,
        check: new string[](1)
    }));
    emit AddReceipt(mapSA[_name],_to,_count,now+_expiredTime);
}
```

- 转移收据：

转移收据时可以自定义转移金额的多少。发起转移收据方一定是收据的收款方，然后提供这个收据的欠款方信息，金额和时间，这样链端就可以先通过查看收据的真实性（存在且没有过期和支付）后，在再新的收款方下对应转移金额的创建收据，同时修改旧收据的转移金额。


```

function transferRecipt(address _from,address _to,uint _count) public{
    uint index = mapR[msg.sender].length;
    for(uint i = 0 ; i < mapR[msg.sender].length ; i ++){
        if(mapR[msg.sender][i].from == _from && mapR[msg.sender][i].to == msg.sender &&
        mapR[msg.sender][i].count >= _count && mapR[msg.sender][i].status == 0){
            index = i;
            break;
        }
    }
    if(index >= mapR[msg.sender].length) return;
    for(uint j = 0 ; j < mapR[_from].length ; j ++){
        if(mapR[_from][j].from == _from && mapR[_from][j].to == msg.sender && mapR[_from][j].count
        == mapR[msg.sender][index].count && mapR[_from][j].status == 0){
            if(mapR[_from][j].count == _count){
                mapR[_from][j].status = 1;
            }
            mapR[_from][j].count -= _count;
            break;
        }
    }
    mapR[msg.sender][index].count -= _count;

    mapR[_from].push(Recipts({
        from:_from,
        to:_to,
        count:_count,
        expiredTime:mapR[msg.sender][index].expiredTime,
        status:0,
        check:new string[](1)
    }));
    mapR[_to].push(Recipts({
        from:_from,
        to:_to,
        count:_count,
        expiredTime:mapR[msg.sender][index].expiredTime,
        status:0,
        check:new string[](1)
    }));
    emit TransferRecipt(msg.sender,_from,_to,_count);
}

```

- 融资：

该功能由金融机构如银行等发起，提供融资方公司信息，融资方提供的收据信息，融资金额，然后查看发起方是否为金融机构，提供的收据信息是否有效(存在+没有过期+没有被支付)，已经融资金额和收据金额是否匹配，如果成立就对金融公司和融资公司的余额进行修改操作。

```

function finance( address _company, address _receiptFrom, uint _count) public{
    if(mapC[msg.sender].cType == 1) return;
    uint index = mapR[_company].length;
    for(uint i = 0 ; i < mapR[_company].length ; i++){
        if(mapR[_company][i].from == _receiptFrom && mapR[_company][i].to == _company &&
        mapR[_company][i].count >= _count && mapR[_company][i].status == 0){
            index = i;
            break;
        }
    }
    if(index >= mapR[_company].length) return;
    mapC[_company].balance += _count;
    mapC[msg.sender].balance -= _count;
    emit Finance(msg.sender,_company,_receiptFrom,_count);
}

```

- 收据认证

该功能可以让金融机构对收据进行认证，发起方为金融机构，然后提供对应的收据信息，通过查看收据信息的有效性后，将收据信息处加入该公司对应的认证信息即可，该功能可以让收据的可信度提高。

```

function validate(address _from, address _to, uint _count) public {
    if(mapC[msg.sender].cType == 1) return;
    uint index = mapR[_from].length;
    for(uint i = 0 ; i < mapR[_from].length ; i++){
        if(mapR[_from][i].from == _from && mapR[_from][i].to == _to && mapR[_from][i].count ==
        _count && mapR[_from][i].status == 0){
            index = i;
            break;
        }
    }
    if(index >= mapR[_from].length) return;
    mapR[_from][index].check.push(mapC[msg.sender].name);
    for(uint j = 0 ; j < mapR[_to].length ; j++){
        if(mapR[_to][j].from == _from && mapR[_to][j].to == _to && mapR[_to][j].count == _count &&
        mapR[_to][j].status == 0){
            mapR[_to][j].check.push(mapC[msg.sender].name);
            break;
        }
    }
    emit Validate(msg.sender,_from,_to,_count);
}

```

- 到期支付功能

在收据指定的期限到了以后，将会强制要求付款方支付欠款，收款方可以获取该笔金额。核心原理是通过判断当前时间和指定的最终时间，当超过期限时自动将钱款打入收款方账户。同时修改收据信息为已支付状态

```

function payBack(address _from){
    uint index = mapR[msg.sender].length;
    for(uint i = 0 ; i < mapR[msg.sender].length ; i ++){
        if(mapR[msg.sender][i].from == _from && mapR[msg.sender][i].to == msg.sender &&
mapR[msg.sender][i].status == 0){
            index = i;
            break;
        }
    }
    if(index ≥ mapR[msg.sender].length) return;
    if(now ≥ mapR[msg.sender][index].expiredTime){
        uint ccount = mapR[msg.sender][index].count;
        mapC[_from].balance -= mapR[msg.sender][index].count;
        mapC[msg.sender].balance += mapR[msg.sender][index].count;
        mapR[msg.sender][index].status = 1;
        for(uint j = 0 ; j < mapR[_from].length ; j ++){
            if(mapR[_from][j].from == _from && mapR[_from][j].to == msg.sender && mapR[_from][j].count
= mapR[msg.sender][index].count && mapR[_from][j].status == 0){
                mapR[_from][j].status = 1;
                break;
            }
        }
        emit PayBack(_from,msg.sender,ccount);
    }
}

```

- 主动支付：

该功能允许欠款方在没有到达期限的时候主动支付欠款，支付后将钱款打入收款方账户的同时修改收据信息为已支付。

```

function payFor(address _to,uint _count){
    uint index = mapR[msg.sender].length;
    for(uint i = 0 ; i < mapR[msg.sender].length ; i ++){
        if(mapR[msg.sender][i].from == msg.sender && mapR[msg.sender][i].to == _to &&
mapR[msg.sender][i].count ≥ _count && mapR[msg.sender][i].status == 0){
            index = i;
            break;
        }
    }
    if(index ≥ mapR[msg.sender].length) return;
    mapC[msg.sender].balance -= mapR[msg.sender][index].count;
    mapC[_to].balance += mapR[msg.sender][index].count;
    mapR[msg.sender][index].status = 1;
    for(uint j = 0 ; j < mapR[_to].length ; j ++){
        if(mapR[_to][j].from == msg.sender && mapR[_to][j].to == _to && mapR[_to][j].count =
mapR[msg.sender][index].count && mapR[_to][j].status == 0){
            mapR[_to][j].status = 1;
            break;
        }
    }
    emit PayFor(msg.sender,_to,_count);
}

```


三、 功能测试

结果是否已经写入链端数据库是基于后台来测试的，下面给出测试过程中使用合约功能时的交易信息等作为实现功能的佐证。

1. 公司账户：

这里创建了四个户，分别对应题目中的车企，轮胎公司，银行和，轮毂公司。其中银行负责提供 见证功能以及融资功能

用户名称	用户id	用户描述	用户公钥地址信息	用户状态	操作
Bank	7000006		0x58027524d931e7eef0eaa38...	正常	修改
WheelandRun	7000007		0x3e046d00551b4e0f2914d...	正常	修改
Wheel	7000008		0x52630603e5e605c7e163eb...	正常	修改
Car	7000005		0x2509c0ff5ac0e0e0a003e41b...	正常	修改

将公司加入入智能合约当中：

该步骤将公司的信息以及账户地址的映射关系加入入到智能合约账户当中保存下来，调用合约的(setCompany)。 如银行:

```

transactionHash: "0x1ee8d320010cdb2463
6bfb"
transactionIndex: 0
blockHash: "0xd6f48afb277dc1d5d8f310d4l
blockNumber: 53
gasUsed: 89315
contractAddress: "0x00000000000000000000
root: null
status: "0x0"
from: "0x58027524dff31e7ed0aca38a8534f
to: "0x67564de8d2a3c447e5de67731fb02b
input: "0xe1d5975e000000000000000000000000(
00060000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
output: "0x"
logs: []
logsBloom: "0x0000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000000000000000000000000000000000000(
00000"
statusOK: true
gasUsedRaw: "0x15ce3"
blockNumberRaw: "0x35"
transactionIndexRaw: "0x0"

```




这里statusOK显示已经正确加入。

其余的其余结果类似。

2. 建立收据:

这里建立车企和车轮公司之间的收据，限定数额为50，期限为200:

```
removed: false
logIndex: null
transactionIndex: null
transactionHash: null
blockHash: null
blockNumber: null
address: 0x67564de8d2a3c447e5de67731fb02bc215da3279
eventName : AddReceipt(address from,address to,uint256 count,uir
data:
```


name	data
from	 0x25B9cD5aC8e6E8A0B3e41b4a...
to	 0x52630665D5D6F0D7E163ebcd...
count	 50

这里显示的是合约中事件记录的结果，可以看到这里的from是 car 公司的账户地址，而 to 则是 wheel 公司的账户地址，以及金金额为设定好的50，因此说明收据建立成功。

3. 银行加入收据证明：

log的 data 54中输出部分，分别为bank，car和wheel的账户地址

```
address: 0x67564de8d2a3c447e5de67731fb02bc215da3279
eventName : Validate(address bank,address from,address to,uint2
data:
```

name	data
bank	 0x58027524Dff31E7Ed0aca38A85...
from	 0x25B9cD5aC8e6E8A0B3e41b4a...
to	 0x52630665D5D6F0D7E163ebcd...

log的 data 中输出部分，分别为bank，car和wheel的账户地址

查看收据信息：

data:		
name	type	data
	uint256	50
	uint256	1573817891...
	uint256	0
	string	BankName

从收据的信息最后一行来看，有银行的名字，说明已经被银行机构证明了。

4. 部分转移收据：

将轮胎的50收据中，拆出其中的25转移给轮毂公司：

from	0x52630665D5D6F0D7E163ebcd...
originFrom	0x25B9cD5aC8e6E8A0B3e41b4a...
to	0x2ed4f0d6F0551b4DDF3914d97...

这里的收据信息分别是轮胎，车企和轮毂




查看对应的收据信息：

name	type	data
	uint256	 25
	uint256	 1573817891...
	uint256	 0
	string	 BankName

可以看到刚被银行证明的收据，余额只剩下25了了，这说明转移收据成功了。

5. 融资：

拿车轮公司的收据向银行融资：




name	data
bank	 0x58027524D#31E7Ed0aca38A85.
from	 0x52630665D5D6F0D7E163ebcd...
receiptFrom	 0x25B9cD5aC8e6E8A0B3e41b4a...

再看下账户余额信息：

可以看到车轮的余额增加了。

6. 还债

车企公司将会偿还轮毂公司(从车轮公司转让的)收据:

from	 0x25B9cD5aC8e6E8A0B3e41b4a...
to	 0x2ed4f0d6F0551b4DDF3914d97...
count	 25

可以看到日志中已经输出了付款25了

可以继续查看轮毂公司的余额:

name	type	data
	string	 WheelRimN...
	uint256	 1
	uint256	 125

可以看到日志中输出了轮毂的名字和账户信息, 从一开始设定100增加至125。

在看车企公司的余额:

name	type	data
	string	 CarName
	uint256	 1
	uint256	 75

在还了轮毂公司的余额后减少了25。说明还债成功。

四、 界面展示

注册页面：

基于区块链的供应链金融平台

Block-Chain-Based Supply Chain Finance Platform

注册

使用已有账户登录

登录页面：

基于区块链的供应链金融平台

Block-Chain-Based Supply Chain Finance Platform

☐ 记住账号

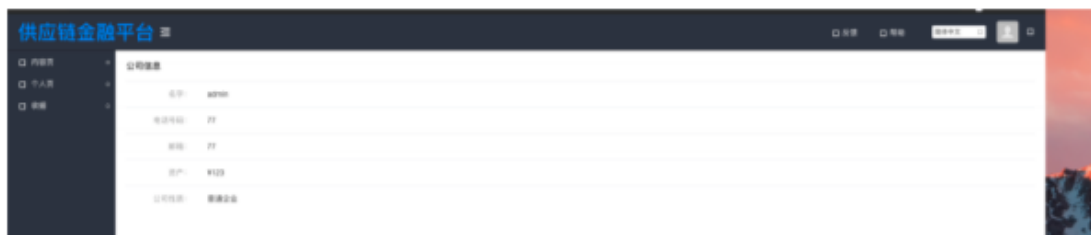
登录

立即注册

进入后页面：



个人信息页面：



信息修改页面：

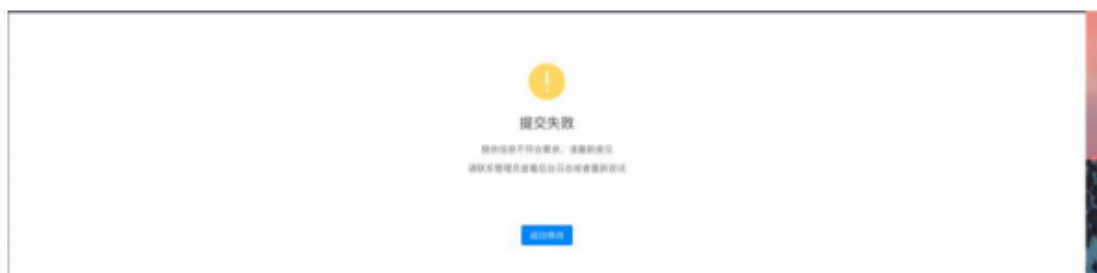


创建收据页面(其余功能页面类似):





失败显示页面



五、 心得体会

本次实验主要是学会创建基于区块链应用，持续了将近一个月，在这个过程中主要学到了链端，后端，前端，区块链，合约创建部署等知识，这个过程中经历了很多痛苦与挫折但是最终也做出了这项产品。在这个过程中学到了很多，在链端主要学会了使用 fisco-bcos 的节点搭建以及对应的控制台使用，在链端中这些节点是区块链的基础，不同于以往我所认为的区块链这样，这些节点更像是为应用和合约服务。另外学会了 webase 的搭建及使用，能够在上面进行合约的创建，编写，调试，编译，这个过程了解了这些应用的执行原理，区分账户地址，私钥，节点，区块，交易这些专业名词和具体使用，以及学会了 sol 编写合约。此外学会了利用 spring-boot 搭建后台，主要是利用 RESTful 可以快速创建基于 API 的 web 服务器，开发起来很快，很适合这些小型的应用。另外就是链端和服务端的连接，主要是通过把部署合约转化成 java 合约后，通过调用这个合约的 API 来实现执行合约功

能，便于服务器能够调用链端的合约功能。在前端方面主要是学会使用了模版+框架创建页面，框架提供了简化实现功能与编码的程序，同时模版可以快速选择想要的界面布局和艺术等。这个过程虽然消耗了很多精力但是也很值得。