

# 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

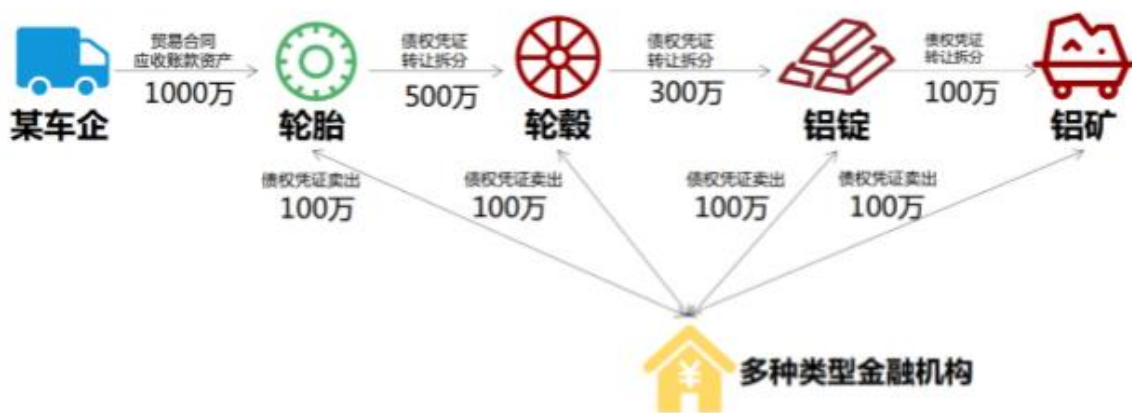
年级	2017	专业（方向）	软件工程
学号	17343026	姓名	冯瑞宁
电话	13359267726	Email	fengxiaoning0123@163.com
开始日期	2019.11.28	完成日期	2019.12.9

### 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。



场景介绍：



## 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。

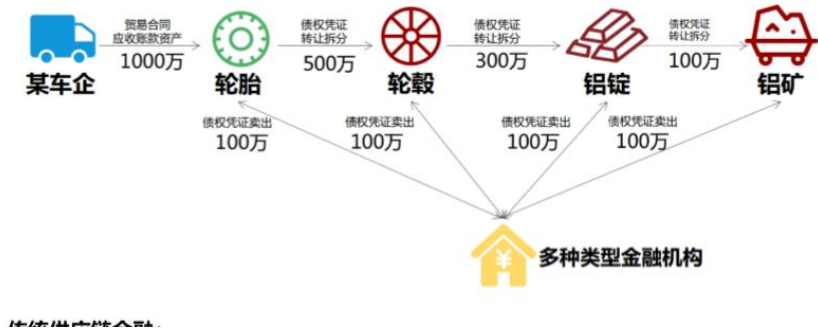
功能四：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

## 二、 方案设计

### 1.问题分析

首先，我们需要先来理解相应的供应链场景。在本金融场景中，我们又

如下的汽车产业供应链：

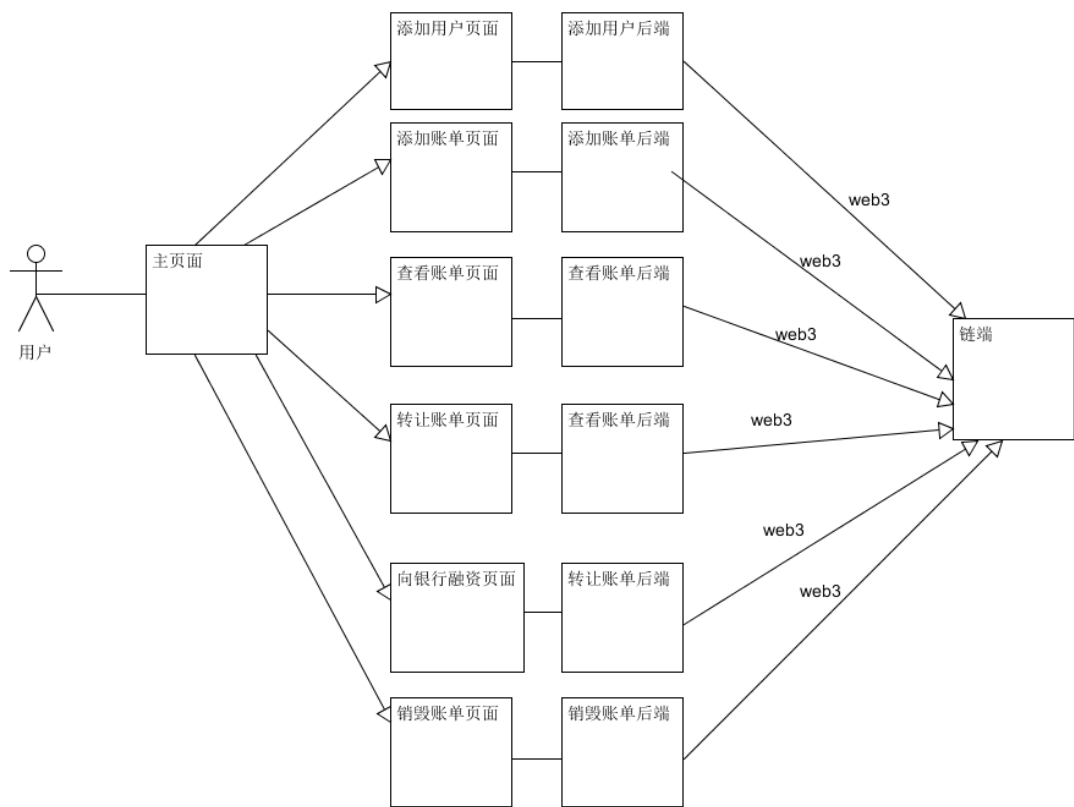


在这条供应链中，主要就是从上游到下游的各个企业，上游企业与下游企业间交易的应收账款和引入的第三方可信机构（银行）。在这个项目中，我们可以把银行作为与一般企业相同的公司来看待。银行与一般公司的区别就仅仅是银行不会作为欠款方出现。因此，在测试时，我们只需通过在合适的函数所对应的参数中提供相应银行节点的地址，就可以实现公司向银行融资的功能（功能三）。

## 2. 存储设计

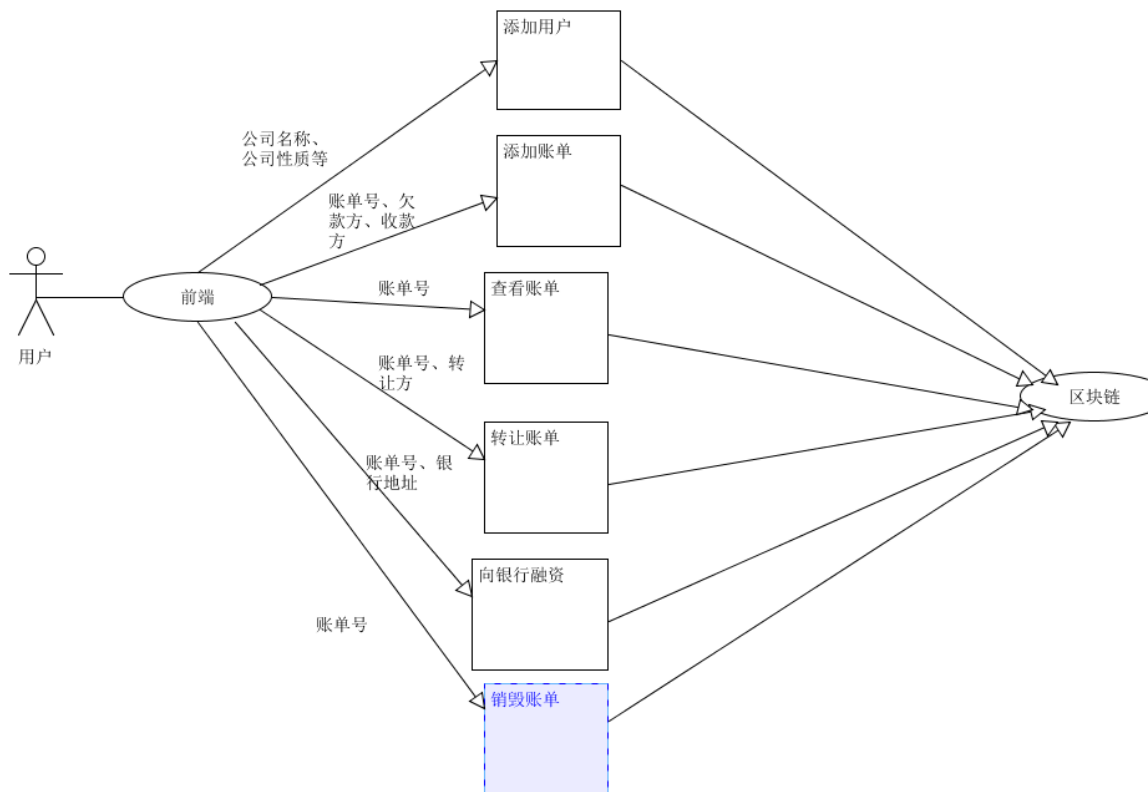
这次项目的主要思想是通过在后端 js 调用 web3 接口并使用 sendTransaction 调用链端的合约中的函数从而实现对于功能，然后在前端页面中调用后端 js 文件，从而使得用户可以在前端实现链上的相关功能操作。

数据存储设计的基本思想是将与链相关的所有数据都存储到底层的链端，通过上面描述的方法实现链端数据的相关操作，从而实现去中心化的生产链功能实现。具体的存储设计可表示如下：



### 3. 数据流图

根据上述存储设计的描述，我们可以得到数据流图如下：



## 4.链端功能介绍

我们先来看基本的数据结构。我们定义两个基本的数据结构，如下：

```
struct Company {  
    address company; // 借款公司  
    uint amount; // 借款公司借款金额  
}  
  
struct Receipt {  
    uint identifier; // 账单对应的账单号  
    uint oweAmount; // 账单对应的欠款金额  
    address sourceNode; // 账单对应的欠款方  
    Company[] companyList;  
}
```

可以看到，我们定义了两个数据结构。第一个数据结构为 Company，这个 Company 可表示为应收账款关系中的收款方，其中的变量包括定义了一个表示该收款公司的地址以及该收款公司的收款额。第二个数据结构为 Receipt，就表示应收账款账单，该数据结构包含了该账单的账单号，账单对应的欠款金额和账单对应的欠款方。然后，我们最后在该账单中加入一个类型为 Company 的数组，表示账单所对应的收款公司。

然后，我们在 Company 和 Receipt 中建立一个映射，从而可以表示一个收款公司当前有多少欠款收据以及这些欠款收据对应的状态，如下：

```
}  
mapping(address => Receipt[]) nodes; // 在Company和Receipt中建立一个映射
```

然后我们就是四个功能的实现，把相应的对上述数据结构的操作上链。我们先看功能一：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。简单来说，就是把谁因为什么东西欠了谁多少钱记录到链上。这其实就等于最基础的针对两个公司间的欠收款关系生成对应的 Receipt，具体如下：

```
// 功能一：采购商品-签发应收账款交易上链，生成一个Receipt，并能查询该Receipt中的欠款金额  
//生成一个新账单  
function addReceipt(address to, uint identifier, uint oweAmount) public {  
    Receipt[] storage receiptList = nodes[to]; //生成一个账单列表,to表示欠款方，被给了东西  
    //检查该账单号是否被使用过  
    for (uint i = 0; i < receiptList.length; ++i) {  
        require(!(msg.sender == receiptList[i].sourceNode && identifier == receiptList[i].identifier), "该账单号重复，不可用");  
    }  
    Receipt storage receipt;  
    receipt.sourceNode = msg.sender;  
    receipt.identifier = identifier;  
    receipt.oweAmount = oweAmount;  
    receiptList.push(receipt);  
}  
  
//查询账单中的欠款金额  
function getReceiptAmount(address from, address to, uint identifier) public view returns(uint) {  
    Receipt[] storage receiptList = nodes[to];  
    for (uint i = 0; i < receiptList.length; ++i) {  
        if (receiptList[i].sourceNode == from && receiptList[i].identifier == identifier) {  
            return receiptList[i].oweAmount;  
        }  
    }  
    require(false, "账单号错误，请重新输入！");  
}
```

我们分为两个函数实现这一功能，先是通过 addReceipt () 函数实现新账单的生成，然后通过 getReceiptAmount () 函数实现账单中欠款金额的查询，我们先来看 addReceipt () 函数。

这个函数的参数设置如下：我们现在用户中输入的是账单的欠款方。然后再输入账单的收款方 to，账单对应的账单号 identifier 和账单对应的欠款金额。对于函数体，我们在这里先是创建了一个账单列表，用来存储已经存在的账单。然后需要特别注意的是，我们这里需要检查我们在改函数的参数中输入的账单号和收款方与账单列表中已经存在的账单号和收款方是否相同，若相同则需要提示用户重新输入。在这之后，我们就可以创建一个新的账单结构 Receipt，并进行相应变量的参数赋值即可。这样我们就成功创建了一个新的应收款交易账单。最后，我们将新创建的账单加入到已有账单列表 receiptList 中。

然后是 getReceiptAmount () 函数。这一函数的功能是查询某账单中的欠款金额。该函数对应的参数为欠款方 from，收款方 to 和账单号 identifier。主体实现就是通过在账单列表中将输入的欠款方 from 和账单号 identifier 与账单列表中的欠款方 from 和账单号 identifier 进行匹配，若成功匹配，就返回该账单列表中账单的欠款金额 oweAmount。

综合这两个函数，我们就实现了功能一，基本应收账款的交易上链。

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。举个例子，比如车企欠轮胎 1000，轮胎欠轮毂 500，那么可以等价于车企欠轮胎和轮毂各自 500，要把这些应收账款的转让交易上链。具体实现如下：

```

// 功能二：应收账款的转让上链。B公司从C公司要了一批东西，B公司可将相应的与A公司欠款账单（A公司为欠款方）给C公司，那么C公司可凭该账单向A公司要B公司买东西的
function transferReceipt(address from, uint Identifier1, address to, uint16 Identifier2, uint64 splitAmount) public {
    require(splitValue > 0, "转让金额必须大于0");
    uint i = 0;
    uint index = 0;
    bool found = false;
    Receipt[] storage receiptList = nodes[msg.sender];
    for (i = 0; i < receiptList.length; ++i) {
        if (receiptList[i].sourceNode == sender && receiptList[i].identifier == Identifier1) {
            found = true;
            index = i;
            break;
        }
    }
    require(found, "账单号错误，请重新输入！");
    // 计算B公司有多少钱可以转让
    Receipt storage receipt = receiptList[index];
    uint sum = 0;
    Company[] storage companyList = receipt.companyList;
    for (i = 0; i < companyList.length; ++i) {
        sum += companyList[i].amount;
    }
    uint left = receipt.oweAmount - sum;
    Receipt[] storage toList = nodes[to];
    for (i = 0; i < toList.length; ++i) {
        require(!(receipt.sourceNode == toList[i].sourceNode && Identifier2 == toList[i].identifier), "该账单号重复，不可用");
    }
    // 生成一个新账单，表示此时A公司同时欠B公司和C公司钱

    // 生成一个新账单，表示此时A公司同时欠B公司和C公司钱
    Receipt storage newReceipt;
    newReceipt.sourceNode = receipt.sourceNode; //相同的欠款方
    newReceipt.identifier = Identifier2;
    newReceipt.oweAmount = splitAmount;
    toList.push(newReceipt);
    receipt.oweAmount -= splitAmount;
}

```

这个函数的参数设置为：账单转让方 from，旧账单 identifier1，账单被转让方 to，新账单 identifier2,和转让金额 spiltAmount。

首先，我们要找到转让方拥有的（作为收款方）的账单，并根据找到的这个转让方作为收款方的账单在计算该转让方当前有多少钱可以转让，比如 A 公司欠 B 公司 100，B 公司欠 C 公司 50，那么此时 B 公司就还有 50 可以进行账单转让。我们用如下函数来计算一个公司的总欠款：

```

uint sum = 0;
Company[] storage companyList = receipt.companyList;
for (i = 0; i < companyList.length; ++i) {
    sum += companyList[i].amount;
}

```

可以看到，通过账单中的 Company[]可以得到该转让方公司的欠款公司的列表 CompanyList[]。从而得到该公司的总欠款额 sum。然后，用该公司的收款额减去该公司的欠款额就可以计算出该公司当前有多少钱可以转让，如下：

```

}
uint left = receipt.oweAmount - sum;

```

计算该公司有多少钱可以转让是因为若该公司的 left 小于 0 的话，那么表示该公司的收款额小于欠款额，则此时该公司是不能转让账单的。

最后，我们在转让方的欠款方和转让方的借款方之间生成一个新的账单，从而成功完成账单的转让上链，具体代码与功能一中的参数赋值类似，并需要在最后将转让方的收款额减去转让的钱，表示此时被转让方已经不再拥有这笔欠款。

```
// 生成一个新账单，表示此时A公司同时欠B公司和C公司钱
Receipt storage newReceipt;
newReceipt.sourceNode = receipt.sourceNode; //相同的欠款方
newReceipt.identifier = Identifier2;
newReceipt.oweAmount = splitAmount;
toList.push(newReceipt);
receipt.oweAmount -= splitAmount;
```

功能三：利用应收账款向银行融资上链，账单中的收款方可根据账单向银行借款。比如，车企欠你 500，你可以用这个凭证向银行借 500，也需要把这种交易上链。具体如下：

```
// 功能三：利用应收账款向银行融资上链，账单中的收款方可根据账单向银行借款
function bankFinancing(address from, uint16 Identifier1, address to, uint64 amount) public {
    require(value > 0, "amount必须大于0");
    uint index = 0;
    uint i = 0;
    bool found = false;
    Receipt[] storage list = nodes[to];
    for (i = 0; i < list.length; ++i) {
        if (list[i].sourceNode == from && list[i].identifier == Identifier1) {
            index = i;
            found = true;
            break;
        }
    }
    require(found, "账单号错误，请重新输入！");

    // 比较该公司的所需借款额和拥有的欠款额
    Receipt storage receipt = list[index];
    uint sum = 0;
    Company[] storage companyList = receipt.companyList;
    for (i = 0; i < companyList.length; ++i) {
        sum += companyList[i].amount;
    }
    uint left = receipt.oweAmount - sum;
    require(left >= amount, "该公司的所需借款额大于其拥有的欠款额");

    // 若该公司符合融资条件
    Company storage thiscompany;
    thiscompany.company = msg.sender;
    thiscompany.amount = amount;
    receipt.companyList.push(thiscompany);
}
```

首先看参数设置：欠款方 from（这里指需要向银行融资的公司），可凭借的应收账款 Identifier1，收款方 to（这里指银行）和向银行的借款额 amount。

首先，我们找到欠款方 from 对于的可作为凭据的应收账款，然后通过该公司所拥有的的其作为收款方的凭据账单来计算该公司所拥有的净收款额，计算与功能二中类似，我们先是计算出该公司的欠款额，然后用收款额减去该公司的欠款额就得到了该公司的净收款额，如下：

```
// 比较该公司的所需借款额和拥有的净收款额
Receipt storage receipt = list[index];
uint sum = 0;
Company[] storage companyList = receipt.companyList;
for (i = 0; i < companyList.length; ++i) {
    sum += companyList[i].amount;
}
uint left = receipt.oweAmount - sum;
require(left >= amount, "该公司的所需借款额>其拥有的净收款额");
```



若该公司的净收款额大于了其所需借款额，那么该公司就符合了融资条件，我们就将银行加入到该公司的账单的欠款的公司列表中，表示其与银行目前的债务关系，如下：

```
//若该公司符合融资条件
Company storage thiscompany;
thiscompany.company = msg.sender;
thiscompany.amount = amount;
receipt.companyList.push(thiscompany);
}
```

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。即到期还钱，销毁在链上对应的应收账款，表明这笔账此时已经了结了。

```
// 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
function payBack(address from, uint Identifier1, address to, uint amount) public {
    uint index = 0;
    uint i = 0;
    bool found = false;
    Receipt[] storage list = nodes[to];
    for (i = 0; i < list.length; ++i) {
        if (list[i].sourceNode == from && list[i].identifier == Identifier1) {
            index = i;
            found = true;
            break;
        }
    }
    require(found, "账单号错误，请重新输入!");
    Receipt storage receipt = list[index];
    Company[] storage companyList = receipt.companyList;
    found = false;
    for (int i = 0; i < companyList.length; ++i) {
        if (companyList[i].provider == msg.sender) {
            index = i;
            found = true;
            break;
        }
    }
    require(found, "公司错误，请重新输入!");
    Company storage company = companyList[index];
    list.pop(receipt);
    companyList.pop(company);
}
```

对于这一功能，我们参数设置为：欠款方 from，账单号 Identifier1,收款方 to，欠款金额 amount。我们根据对应的输入参数匹配相对应的账单以及账单对应的公司，并在账单列表和公司列表中 pop 掉该账单和账单对应的公司即可，具体如下：

```
Receipt storage receipt = list[index];
Company storage company = companyList[index];
list.pop(receipt);
companyList.pop(company);
}
```

最后，由于我们这次的作业是要基于 fisco 的底层链实现，所以这里我们不能直接像上次一样在 webase 在线平台上直接添加用户私钥。因此，这次的作业还需要我们多实现一个添加用户的功能，这个功能实现起来比较简单，这里就不赘述，如下：

```

User storage user = blankMap[msg.sender];

user.ID = id;
user.name = name;
user.balance = balance;
user.role = role;

if (role == Role.MainFactory) mainFactoryMap[msg.sender] = user;
else if (role == Role.Supplier) supplierMap[msg.sender] = user;
else if (role == Role.FinancingInstitution) financingInstitutionMap[msg.sender] = user;

```

## 5.后端交互介绍

首先，我们需要部署 node.js SDK,在 LINUX 中需要先部署 nodejs 环境，如下：

```

fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.2/install.sh | bash
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 11699 100 11699 0 0 19662 0 --:--:-- --:--:-- --:--:-- 19662
=> Downloading nvm from git to '/home/fisco-bcos/.nvm'
=> Cloning into '/home/fisco-bcos/.nvm'...
remote: Enumerating objects: 7573, done.
remote: Total 7573 (delta 0), reused 0 (delta 0), pack-reused 7573
Receiving objects: 100% (7573/7573), 2.48 MiB | 55.00 KiB/s, done.
Resolving deltas: 100% (4788/4788), done.
* (HEAD detached at v0.33.2)
  master
=> Compressing and cleaning up git repository
Counting objects: 7573, done.
Compressing objects: 100% (7517/7517), done.
Writing objects: 100% (7573/7573), done.
Total 7573 (delta 5063), reused 2282 (delta 0)

=> Appending nvm source string to /home/fisco-bcos/.bashrc
=> Appending bash_completion source string to /home/fisco-bcos/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ source ~/.$(basename $SHELL)rc
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ nvm install 8
Downloading and installing node v8.16.2...
Downloading https://nodejs.org/dist/v8.16.2/node-v8.16.2-linux-x64.tar.xz...
##### 100.0%
Computing checksum with sha256sum
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v8.16.2 (npm v6.4.1)
Creating default alias: default -> 8 (-> v8.16.2)
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ nvm use 8
Now using node v8.16.2 (npm v6.4.1)

```

然后，我们部署 Node.js SDK，如下：

```
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ git clone https://github.com/FISCO-BCOS/nodejs-sdk.git
Cloning into 'nodejs-sdk'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 365 (delta 28), reused 39 (delta 16), pack-reused 297
Receiving objects: 100% (365/365), 117.71 KiB | 59.00 KiB/s, done.
Resolving deltas: 100% (192/192), done.
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ npm config set registry https://registry.npm.taobao.org
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy$ cd nodejs-sdk
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy/nodejs-sdk$ npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 698 packages from 379 contributors in 67.249s
fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy/nodejs-sdk$ npm run repoclean

> nodejs-sdk@0.9.0 repoclean /home/fisco-bcos/webase-deploy/nodejs-sdk
> lerna clean && npm run clean-solc-0.4 && npm run clean-solc-0.5

lerna notice cli v3.19.0
lerna info Removing the following directories:
lerna info clean packages/api/node_modules
lerna info clean packages/cli/node_modules
? Proceed? Yes
lerna info clean removing /home/fisco-bcos/webase-deploy/nodejs-sdk/packages/api/node_modules
lerna info clean removing /home/fisco-bcos/webase-deploy/nodejs-sdk/packages/cli/node_modules
lerna success clean finished

> nodejs-sdk@0.9.0 clean-solc-0.4 /home/fisco-bcos/webase-deploy/nodejs-sdk
> rm -rf ./packages/api/common/solc-0.4/node_modules
```

```
> nodejs-sdk@0.9.0 clean-solc-0.5 /home/fisco-bcos/webase-deploy/nodejs-sdk
> rm -rf ./packages/api/common/solc-0.5/node_modules

fisco-bcos@fiscobcos-VirtualBox:~/webase-deploy/nodejs-sdk$ npm run bootstrap

> nodejs-sdk@0.9.0 bootstrap /home/fisco-bcos/webase-deploy/nodejs-sdk
> lerna bootstrap && npm run install-solc-0.4 && npm run install-solc-0.5

lerna notice cli v3.19.0
lerna info Bootstrapping 2 packages
lerna info Installing external dependencies
lerna info hoist Installing hoisted dependencies into root
lerna info hoist Finished bootstrapping root
lerna info hoist Pruning hoisted dependencies
lerna info hoist Finished pruning hoisted dependencies
lerna info Symlinking packages and binaries
lerna success Bootstrapped 2 packages

> nodejs-sdk@0.9.0 install-solc-0.4 /home/fisco-bcos/webase-deploy/nodejs-sdk
> cd ./packages/api/common/solc-0.4/ && npm install

npm notice created a lockfile as package-lock.json. You should commit this file.
added 66 packages from 35 contributors in 7.326s

> nodejs-sdk@0.9.0 install-solc-0.5 /home/fisco-bcos/webase-deploy/nodejs-sdk
> cd ./packages/api/common/solc-0.5/ && npm install

npm notice created a lockfile as package-lock.json. You should commit this file.
added 25 packages from 15 contributors in 6.506s
```

用 node -v 和 npm -v 查看版本号，可以看到安装成功，如下：

```
fisco-bcos@fiscobcos-VirtualBox:~$ node -v
v8.16.2
fisco-bcos@fiscobcos-VirtualBox:~$ npm -v
6.4.1
```

然后在 fisco 中启动节点，确认节点已启动后，打开 fisco 控制台，部署合约，获得合约地址，如下：

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ bash nodes/127.0.0.1/start_all.sh
try to start node0
try to start node1
try to start node2
try to start node3
try to start node4
node2 start successfully
node1 start successfully
node3 start successfully
node4 start successfully
node0 start successfully
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ tail -f nodes/127.0.0.1/node0/log/log* |grep ++
info|2019-10-24 23:59:58.994008|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=61d5b320...
info|2019-10-25 00:59:57.586077|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=398fid7b...
info|2019-10-25 01:59:57.985851|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=5b1b9a69...
info|2019-10-25 03:59:59.003463|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=33dcf59a...
info|2019-10-25 04:59:56.980370|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=aa907313...
info|2019-10-25 05:59:58.760389|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=d2000691...
info|2019-10-25 07:59:59.165481|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=3282c1f8...
info|2019-10-25 09:59:58.393729|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=5c4117ac...
info|2019-10-25 11:59:57.452641|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=452f0f5a...
info|2019-10-25 12:59:57.183963|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=ea363d14...
info|2019-10-25 13:59:58.962233|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=1,tx=0,nodeIdx=3,hash=a65b2225...
info|2019-10-25 14:59:59.537086|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=3,tx=0,nodeIdx=3,hash=b49279c3...
info|2019-10-25 15:40:03.906380|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=5,tx=0,nodeIdx=3,hash=7dd0a54d...
info|2019-12-13 00:32:05.469783|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=5,tx=0,nodeIdx=3,hash=3e3cf9b3...
info|2019-12-13 00:32:09.557791|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=5,tx=0,nodeIdx=3,hash=0a48b961...
info|2019-12-13 00:32:13.592370|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=5,tx=0,nodeIdx=3,hash=5098aaa4...
info|2019-12-13 00:32:17.628560|[g:1][CONSENSUS][SEALER]+++++++ Generating seal on,blkNum=5,tx=0,nodeIdx=3,hash=026ae66b...
AC

fisco-bcos@fiscobcos-VirtualBox:~/fisco$ cd ~/fisco/console && bash start.sh
=====
Welcome to FISCO BCOS console(1.0.5)!
Type 'help' or 'h' for help. Type 'quit' or 'q' to quit console.

| $$$$$$| $$$$$$| $$$$$$| $$$$$$| $$$$$$| | $$$$$$| $$$$$$| $$$$$$| $$$$$$| | |
| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ |
| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ |
| $$$$$$| $ $ | $$$$$$| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $$$$$$|
| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ |
| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ |
| $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ | $ $ |

=====
[group:1]> deploy SupplyChain3.sol
contract address: 0x47dc184882bf3df21046c028448a18bc80aba82b
```

然后，我们获取合约的 abi，可以在 console/sdk 中找到，并安装 web3SDK，使得我们的程序可以调用 web3 接口，如下：

```
fisco-bcos@fiscobcos-VirtualBox:~/fisco/console$ ./sol2java.sh org.fisco.bcos.SupplyChain3.contract
Compile solidity contract files to java contract files successfully!
fisco-bcos@fiscobcos-VirtualBox:~/fisco/console$ cd ..
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ curl -LO https://github.com/FISCO-BCOS/LargeFiles/raw/master/tools/asset-app.tar.gz
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 151 100 151 0 0 226 0 --:--:-- --:--:-- --:--:-- 225
100 212k 100 212k 0 0 136k 0 0:00:01 0:00:01 --:--:-- 743k
fisco-bcos@fiscobcos-VirtualBox:~/fisco$ tar -zxvf asset-app.tar.gz

fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ cd ..
fisco-bcos@fiscobcos-VirtualBox:~$ cp fisco/nodes/127.0.0.1/sdk/* asset-app/src/test/resources
fisco-bcos@fiscobcos-VirtualBox:~$ cd asset-app
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ ./gradlew build
Downloading https://services.gradle.org/distributions/gradle-5.6.2-bin.zip
.....
Welcome to Gradle 5.6.2!

Here are the highlights of this release:
- Incremental Groovy compilation
- Groovy compile avoidance
- Test fixtures for Java projects
- Manage plugin versions via settings script

For more details see https://docs.gradle.org/5.6.2/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 3m 57s
4 actionable tasks: 4 executed
```

最后，打开 dist 目录，成功实现 web3 接口和合约的交互，如下：

```
+ actionable tasks: 4 executed
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ cd dist
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash asset_run.sh deploy
deploy Asset success, contract address is 0x6723e149a8a32e8af40a0c453287b8d2ceb40535
```

然后我们看如何在后端 nodejs 服务器通过 web3 接口调用合约中的函数。我们以任意一个 js 文件为例，首先，需要创建一个 web3 对象，若连接端口与默认端口不同，则自行设置一个连接端口，如下：

```
if (typeof web3 !== 'undefined') {
    web3 = new Web3(web3.currentProvider);
} else {
    web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));
}
```

接下来，我们需要定义一个 API 接口，用之前获取到的合约的 abi 来定义这个合约接口。

我们这里用到的就是 web3.eth.contract 函数，如下：

```
var supplyChain3 = web3.eth.contract([{"constant":false,"inputs":[{"name":"from","type":"address"},{"name":"to","type":"address"},{"name":"receiptID","type":"bytes32"},{"name":"amount","type":"uint256"},{"name":"signTime","type":"uint256"},{"name":"endTime","type":"uint256"}],"name":"transferReceipt","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"name":"totalFinancingAmount","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"payDebt","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"name":"role","type":"uint8"},{"name":"userAddr","type":"address"}],"name":"getBalance","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"receiptID","type":"bytes32"},{"name":"signTime","type":"uint256"},{"name":"amount","type":"uint256"},{"name":"endTime","type":"uint256"},{"name":"supplierAddr","type":"address"}],"name":"signReceipt","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"amount","type":"uint256"}],"name":"financing","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"id","type":"bytes32"},{"name":"name","type":"string"},{"name":"balance","type":"uint256"},{"name":"role","type":"uint8"}],"name":"addUser","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"function"}]);
```

然后，用我们之前得到的合约地址来生成合约对象，这里用到的是.at 函数，如下：

```
var con = supplyChain3.at('0x8619c257e8a73763a5d19bc26213a6c8d75beb3c');
```

接下来就是最核心的部分，用 sendTransaction () 调用合约中的函数，sendTransaction () 的定义如下：



## web3.eth.sendTransaction

web3.eth.sendTransaction(transactionObject [, callback])

发送一个交易到网络。

参数:

- `Object` - 要发送的交易对象。
  - `from`: `String` - 指定的发送者的地址。如果不指定, 使用 `web3.eth.defaultAccount`。
  - `to`: `String` - (可选) 交易消息的目标地址, 如果是合约创建, 则不填。
  - `value`: `Number|String|BigNumber` - (可选) 交易携带的货币量, 以 `wei` 为单位。如果合约创建交易, 则为初始的基金。
  - `gas`: `Number|String|BigNumber` - (可选) 默认是自动, 交易可使用的 `gas`, 未使用的 `gas` 会退回。
  - `gasPrice`: `Number|String|BigNumber` - (可选) 默认是自动确定, 交易的 `gas` 价格, 默认是网络 `gas` 价格的平均值。
  - `data`: `String` - (可选) 或者包含相关数据的字节字符串, 如果是合约创建, 则是初始化要用到的代码。
  - `nonce`: `Number` - (可选) 整数, 使用此值, 可以让你覆盖你自己的相同 `nonce` 的, 正在 `pending` 中的交易<sup>11</sup>。
- `Function` - 回调函数, 用于支持异步的方式执行<sup>12</sup>。

返回值:

`String` - 32字节的交易哈希串。用16进制表示。

如果交易是一个合约创建, 请使用 `web3.eth.getTransactionReceipt()` 在交易完成后获取合约的地址。

我们以之前实现的最基本的 `AddReceipt()` 增添账单为例, `sendTransaction` 函数如下:

```
var currAccount = $("select").find("option").attr("value");
con.addReceipt.sendTransaction($("#id").val(), $("#name").val(), $("#value").val(), {from:currAccount},{to:toAccount},
unction(error, addr) {
```

如图, 表示增加了一个从当前用户到目标用户的数额为 `value` 的欠款账单。在该函数执行后, 我们就会将在 `html` 页面中自行设置的参数通过 `web3` 传送给链端, 从而使该交易上链, 成功实现 `html` 页面与底层链端的交互。

根据这一实现, 我们就可以仿照 `addReceipt.js` 实现其他合约中函数对于的 `js` 文件, 从而

使得当我们在后端中实现一个交易时，该交易能够在链端上链，并通过该交易实现链端对于的功能。

## 6.前端功能介绍

前端的主要功能实现的就是直接与用户进行交互的页面以及与实现好的后端建立交互。

首先是在前端 html 页面中调用我们之前在进行 web3 配置的时候得到的目录 node\_modules 中的 web3/dist/web3.min.js，使得前端可以实现对 web3 接口的使用。然后，调用之前写好的 nodejs 后端，使得可以实现前端、后端与链端之间的交互，还是以 addReceipt 函数对应的 addReceipt.html 页面调用 addReceipt.js 为例，具体实现如下：

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script src="../node_modules/web3/dist/web3.min.js"></script>
<script src="addReceipt.js"></script>
```

然后是用户交互页面。我们根据之前对问题的分析和开始的存储设计图和数据流图，可以明确我们在前端的实现主要是对应合约中的五大功能：添加账单、查看账单、转让账单、向银行融资以及销毁账单。然后，由于是基于 fisco-bcos 底层链端的实现，我们不能像在大作业二中那样直接在 webase 平台中添加用户私钥，从而表示生产链上的每个公司，所以这里我们就还需要基于之前在链端增添的添加用户的功能实现一个对应的 html 页面。

总结起来就是要实现一个主页面和六个分页面，从主页面可以进入到六个分页面中，从而在主页面中实现六大功能的选择。从主页面进入任一分页面则可以通过调用 js 后端实现与链端的交互，从而实现六大功能任一功能的实现。

我们先来看主页的实现。主页就是设置 6 个按钮，分别代表不同的功能，通过 button 实现，具体如下：

```
<div id="content" class="div1" ; border-radius:100px;">

  <div style="position:relative;top:80px;left:160px;">
    <button id="addUser"><h3>添加用户</h3></button>

    <button id="addReceipt"><h3>添加账单</h3></button>

    <button id="getReceiptAmount"><h3>查看账单</h3></button>

    <button id="transferReceipt"><h3>转让账单</h3></button>

    <button id="bankFinancing"><h3>向银行融资</h3></button>

    <button id="payBack"><h3>销毁账单</h3></button>

  </div>
```

然后用点击事件触发器 click () 和 window.location.href 实现主页面和子页面之间的跳转，即就是使得用户点击每个按钮就能跳转到实现对应功能的页面，如下：

```
<script>
  $("#addUser").click(function() {
    window.location.href="addUser.html";
  });

  $("#addReceipt").click(function() {
    window.location.href ="addReceipt.html";
  });

  $("#getReceiptAmount").click(function() {
    window.location.href ="getReceiptAmount.html";
  });

  $("#transferReceipt").click(function() {
    window.location.href ="transferReceipt.html";
  });

  $("#bankFinancing").click(function() {
    window.location.href ="bankFinancing.html";
  });

  $("#payBack").click(function() {
    window.location.href ="payBack.html";
  });
});
```



接下来是子页面的实现，还是以其中一个为例，我们以 addUser.html 为例，我们用 label 来提示用户需要输入的信息，用 input 来进行用户输入，最后，若用户点击“返回主页”，则还是用点击事件触发器 click () 和 window.location.href 实现跳转到之前的主页，如下：

```
<h1>添加用户</h1>

<label for="id" class="col-1g-2 control-label">公司ID</label>
<input id="id" type="text">

<label for="name" class="col-1g-2 control-label">公司名称</label>
<input id="name" type="text">

<label for="balance" class="col-1g-2 control-label">公司资产</label>
<input id="balance" type="text">

<button id="add">添加 </button>
<button id="back">返回主页 </button>
```

```
$("#back").click(function() {
    window.location.href="index.html";
});
```

transferReceipt.html:

```
<h1>转让账单</h1>

<label for="from" class="col-1g-2 control-label">欠款方</label>
<input id="from" type="text">

<label for="to" class="col-1g-2 control-label">收款方</label>
<input id="to" type="text">

<label for="receiptid" class="col-1g-2 control-label">账单号</label>
<input id="receiptid" type="text">

<button id="add">转让 </button>

<button id="back">返回主页 </button>
```

Payback.html:

```

<h1>销毁账单</h1>

<label for="receiptid" class="col-1g-2 control-label">账单号</label>
<input id="receiptid" type="text">

<button id="add">销毁</button>

<button id="back">返回主页</button>

```

bankFinancing.html:

```

<div class="container">

  <h1>向银行融资</h1>

  <label for="amount" class="col-1g-2 control-label">融资金额</label>
  <input id="amount" type="text">

  <button id="add">融资 </button>

  <button id="back">返回主页 </button>

```

getReceiptAmount.html:

```

<h1>查看账单</h1>

<label for="receiptid" class="col-1g-2 control-label">账单号</label>
<input id="receiptid" type="text">

<button id="add">确认 </button>

<button id="back">返回主页 </button>

</div>

```

addReceipt.html:

```

<h1>添加账单</h1>

<label for="receiptid" class="col-lg-2 control-label">账单号</label>
<input id="receiptid" type="text">

<label for="amount" class="col-lg-2 control-label">欠款金额</label>
<input id="amount" type="text">

<label for="fromaddr" class="col-lg-2 control-label">欠款方地址</label>
<input id="fromaddr" type="text">

<button id="add">添加</button>

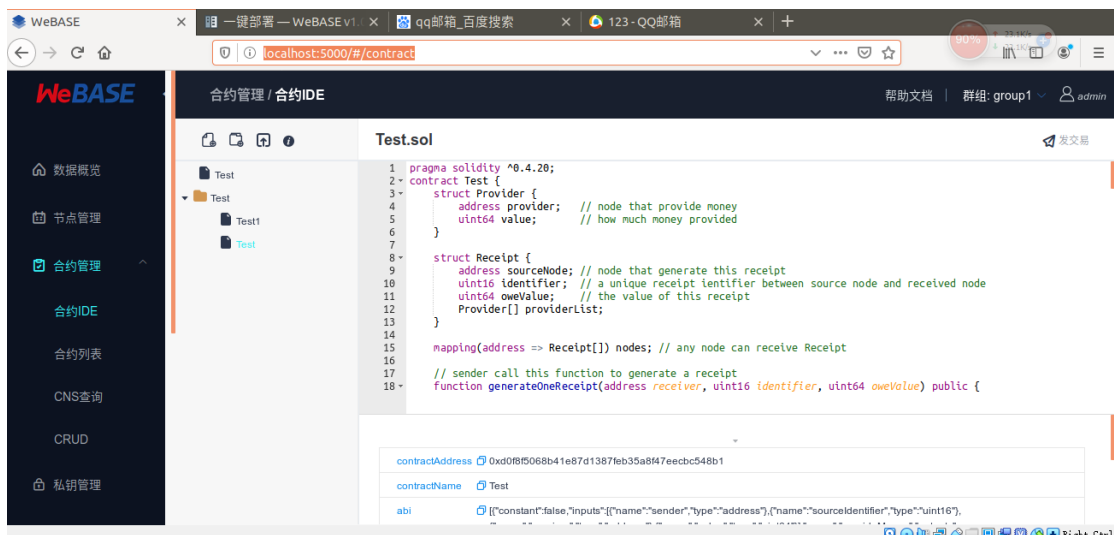
<button id="back">返回主页 </button>

```

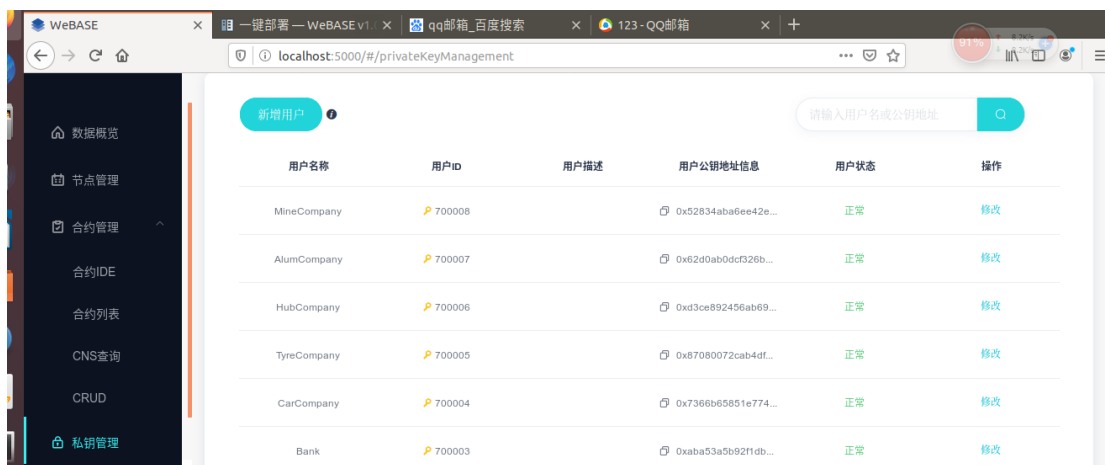
### 三、 功能测试

#### 1.在 webase 上对合约进行功能测试:

配置好 webase 的数据库环境并一键部署后，我们登录 webase 的在线编译平台，点击合约管理，合约 IDE，将合约载入到平台上，并编译部署到链上，如下：



然后，我们在私钥管理中创建六个用户，分别是供应链上的五个公司和银行，如下：





[illegible]

### 功能二：

如图，表示铝矿公司，铝锭公司和轮毂公司之间的转发交易，源合约号为12345，要新建的合约号为1234，转发金额为10，如下：

合约名称: Test

合约地址: 0xd0f8f5068b41e87d1387feb35e

用户: MineCompany

方法: function splitReceipt

参数: sender 3e6e38421b26c85c  
sourceIdentifier 12345  
nextReceiver 25cd4000522  
destIdentifier 1234  
splitValue 10

❗ 如果参数类型是数组，请用逗号分隔，不需要加上引号，例如：array1,array2。string等其他类型也不用加上引号。

取消 确定

可以看到，交易成功，如下：









960  
000  
675

## 2.最终效果测试：

应用

作业提交系统

PowerPoint Prese...

Sicily Online Juc

区块链

区块链

区块链

区块链

区块链

区块链

添加用户

创建成功！用户地址为：  
0xC6489E0aA4E60B0515fc43C9B17DAC1fcf2082c2

确定

公司ID

111

公司名称

车企

公司资产

200

添加

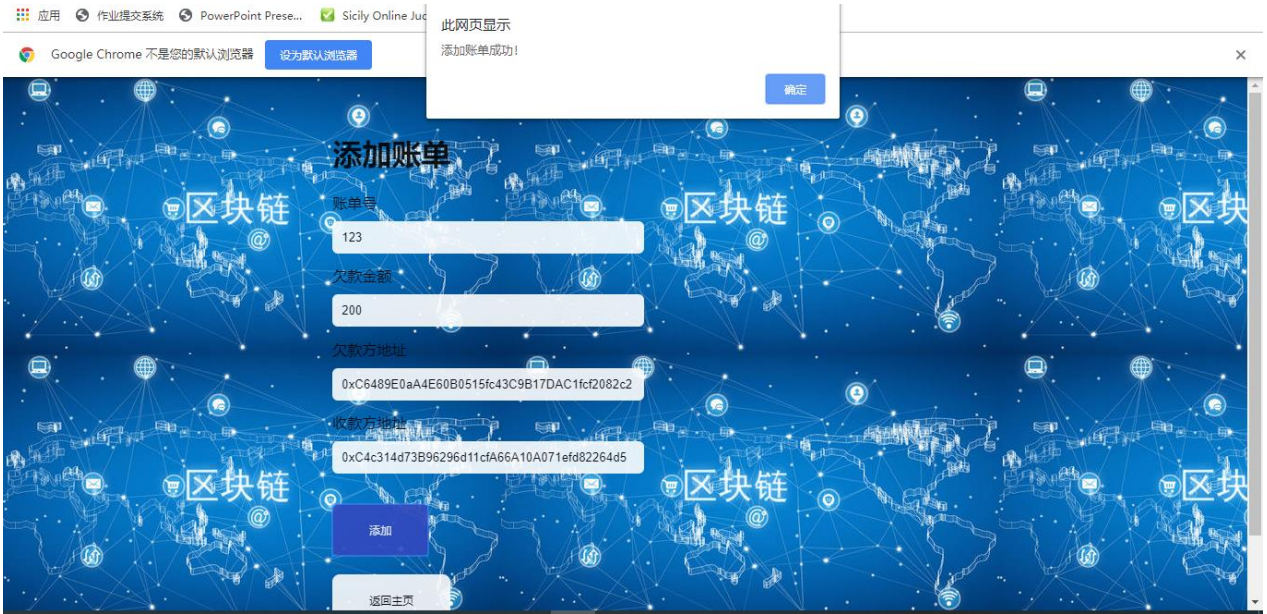
返回主页



添加公司轮胎公司，得到轮胎公司的地址：



添加车企和轮胎公司之间的账单：



查看该账单：



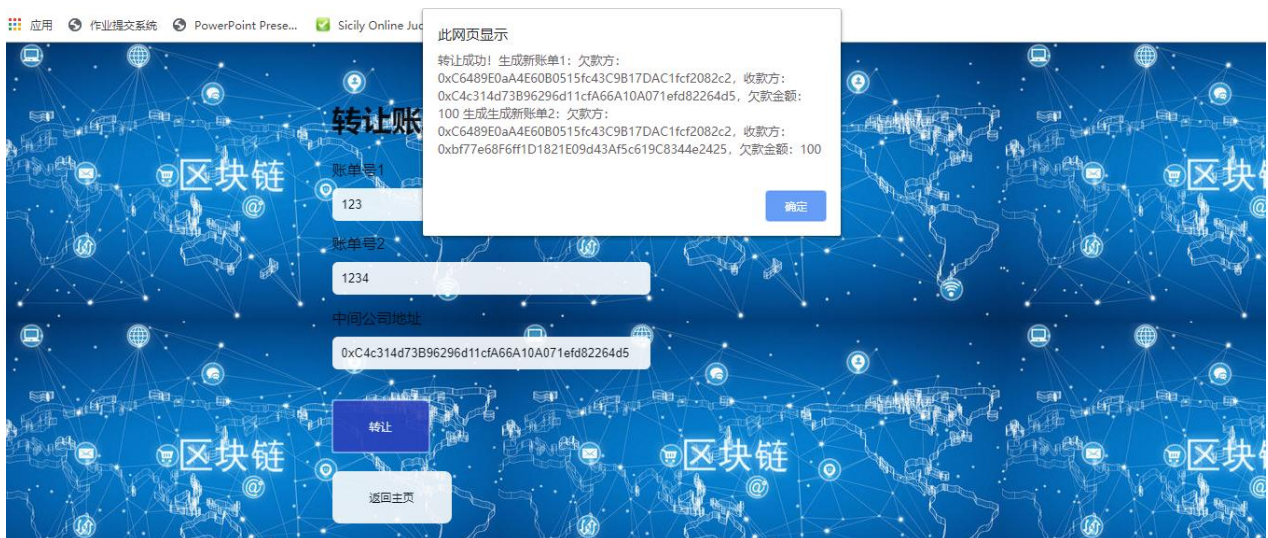
然后，我们再新建一个用户：轮毂公司，得到轮毂公司的地址为：

0xbf77e68F6ff1D1821E09d43Af5c619C8344e2425，然后再新建一个轮胎公司与轮

毂公司间的账单，账单号为 1234，欠款方为轮胎公司，收款方为轮毂公司，欠款金额为

100。然后使用转让账单功能，使得此时车企既欠轮胎公司 100，也欠轮毂公司 100。如

下：



可以看到，我们需要先提供两个账单，表示车企此时欠轮胎公司 200，轮胎公司欠轮毂公

司 100.然后，将中间公司地址设为轮胎公司的地址，这样会生成两个新账单，车企欠轮

胎公司 100 的账单和车企欠轮毂公司 100 的账单。

接下来是向银行融资，我们需要提供一个凭据账单号，这个凭据账单号中的借款方就是融

资方，他可以通过这个账单向银行进行融资，从而生成一个该公司为欠款方，银行为收款



方的账单，如下：



其中 0x250d4119b9a15e1d668737c0bcfcf4bff0764dc1 为银行地址（银行也是可以创建的用户公司）。

最后，是销毁账单，输入账单号即可销毁账单，如下：



查看该销毁的账单，提示查找不到，说明销毁成功：



四、 界面展示

主页：



这里由于是在 windows 上测试发生显示错误，若在 linux 的谷歌浏览器中即可正常显示。

添加用户：





### 添加用户

公司ID

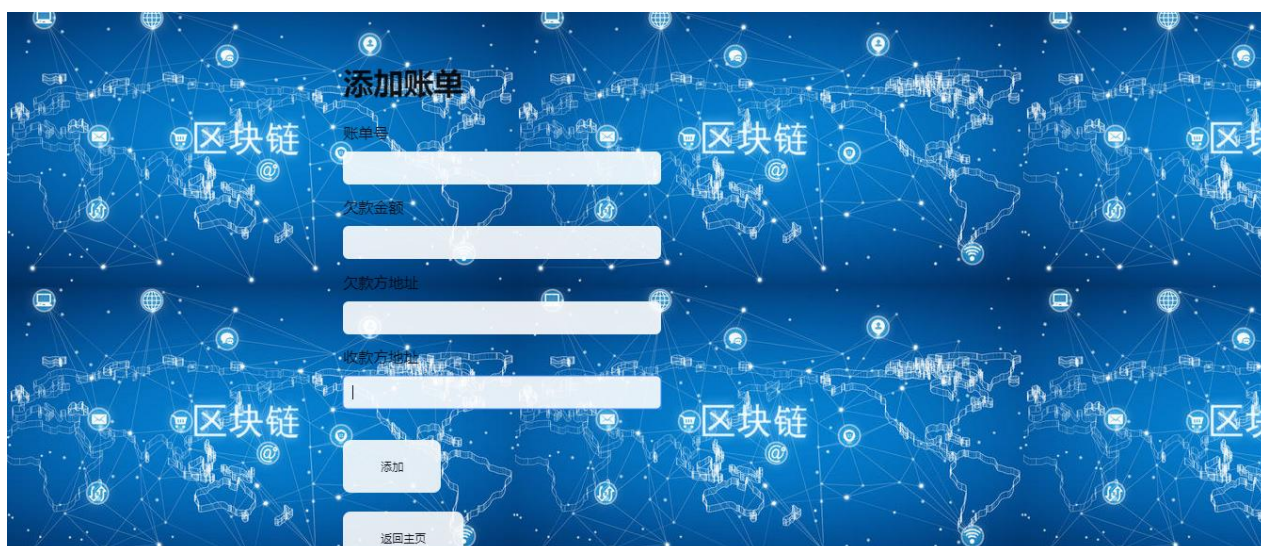
公司名称

公司资产

添加

返回主页

添加账单：



### 添加账单

账单号

欠款金额

欠款方地址

收款方地址

添加

返回主页

查看账单：





## 查看账单

账单号

确认

返回主页

向银行融资：



## 向银行融资

凭据单号

融资金额

融资

返回主页

转让账单：



## 转让账单

账单号1

账单号2

中间公司地址

转让

返回主页

销毁账单：



## 五、 心得体会

终于完成了这次大作业，总的来说，这次大作业对我来说难度还是比较大的。一开始实现的时候我也是完全无从下手，后来在仔细查看了网上的教程并咨询了同学后，我才有了头绪，确定了代码的整体模块和具体的实现方式，并耗费了很长的时间进行相关配置，这才完成了作业。

通过这次作业，我对区块链的相关知识进行了进一步的巩固和学习，深刻体会到了区块链的“去中心化”的这一特点，了解到了基于 fisco 底层的节点的生成方法，并能基于给出的场景写出实现相应功能的合约，并将合约部署到链上。并且，我也通过这次作业学习了 web3 的基本架构及使用方法，并学会了通过 web3 实现后端 nodejs 与链端的交互，这无疑为我以后的区块链方面的相关编程打下了基础。最后，通过这次作业我还复习了之前学习过的前端编程方面的知识，包括 html, css, js 的编写，通过编写前端代码也进一步提高了我的编程能力。

然而，在进行作业的过程我也意识到了我的不足，包括区块链相关知识不够熟悉，前端编程不够熟练，自己独立解决问题的能力较差等。对此，我还要在以后的学习中进一步弥补自己的这些不足，进一步提升自己对区块链相关知识的理解，并多动手实践，提高自己的编程能力。