中山大学计算机学院（软件学院）本科生实验报告

年级	2018	专业	软件工程
任课教师	黄华威	Email	suyy26@mail2.sysu.edu.cn
开始日期	2020.12.30	结束日期	2021.1.29
学号姓名	18342107 宿永烨 18342133 张泽健 17343114 王烁旌		

一、 项目背景

现有三家公司 A、B 和 C，情况如下：

公司 A 规模大，信用良好，实力丰厚，有很大的风险承担的能力，拥有直接从银行等金融机构获得大笔融资的资格。公司 B、C 规模相对较小，不能直接从银行金融机构获得大笔资金。

现有以下交易场景：

公司 A 向公司 B 订购一批货物；公司 B 生产这批货物，需要向公司 C 订购一批材料。公司 A 现在资金短缺，向公司 B 签订一份一定数额的单据，承诺会给付公司 B 货款。这时，公司 B 可以用这份跟公司 A 签订的单据向银行贷款，凭借的是公司 A 的信用。并且，公司 B 有了公司 A 授权的信用，公司 B 能以同样的方式向公司 C 签订单据，公司 C 也可以凭借单据向银行贷款。

传统方法下，这样的流程需要非常复杂的授权、验证过程。本项目基于腾讯微众银行开源的 FISCO BCOS 区块链平台，使用区块链技术高效地实现这样的信用传递过程，实现了基于区块链的供应链金融平台。基于区块链、智能合约等，实现基于区块链的供应链金融平台。实现供应链应收账款资产的溯源、流转。

实现功能包括：

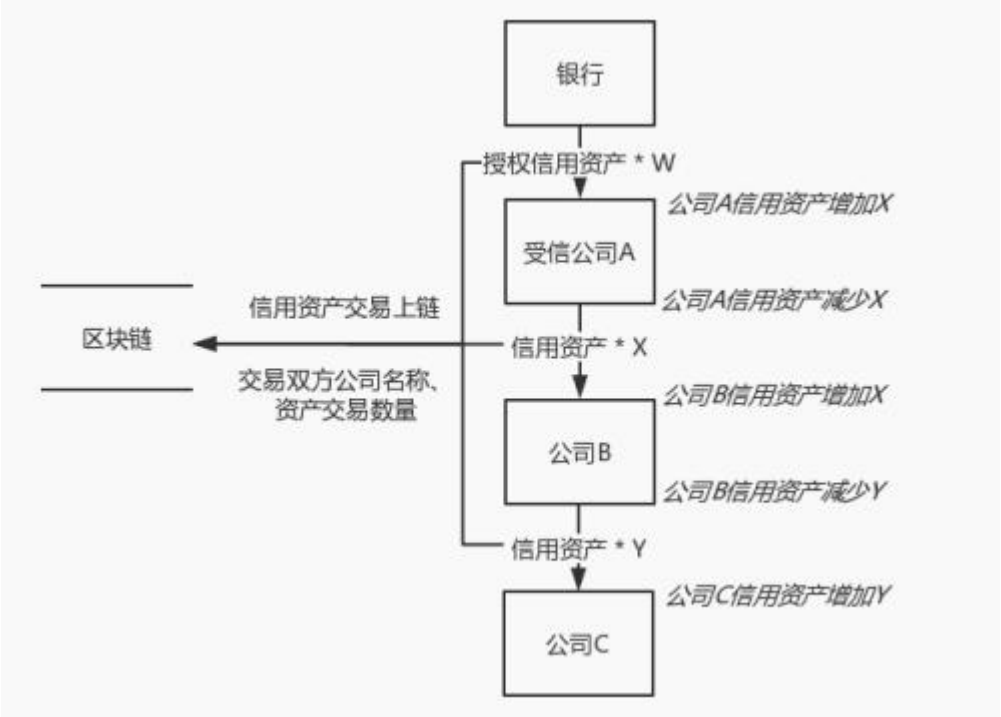
功能一：实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

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。

概括来讲，就是四个字“信债可转”，其信用资产在交易中的流通步骤可以由下图解释：



二、 实现方案设计

链上的智能合约(Programming Language: Solidity)源代码位于 contracts 文件夹中，有四个 sol 文件。

main.sol 文件是一个 aggregate 文件，合成了上述所有功能，并实现了三个辅助函数打印当前厂商银行以及账单的信息，这样有助于使用者与测试开发人员调试我们的链代码。listCompany 打印厂商信息，listBank 打印银行的信息，listReceipt 打印账单信息，主要用于测试以及展示。

Console.Sol 是辅助类，提供 log 函数用于打印。

companyAndBank.sol 文件定义了结构体 company 表示厂商和银行对象，两者通过字段 status 进行区分，isCompany 和 isBank 实现厂商以及银行的查询 功能，addCompany 和 addBank 实现厂商和银行的注册功能，能调用查询以避免重复注册，companys 和 banks 保存所有已注册厂商和银行的信息。

```
pragma solidity ^0.4.22;
contract companyAndBank{
        struct company{
            string cname;//公司名
           string status;//标记，区分银行和普通公司
        }
        company[] companys;
        company[] banks;
        function isCompany(string a)returns(bool){
            uint i;
            for(i=0;i<companys.length;i++){
    if(keccak256(abi.encodePacked(companys[i].cname))==keccak256(abi.encodePacked(a)))
```

```
            return true;
            }
            return false;
        }
        function isBank(string a)returns(bool){
            uint i;
for(i=0;i<banks.length;i++){                if(keccak256(abi.encodePacked(banks
[i].cname))==keccak256(abi.encodePacked(a)))
                 return true;
            }
            return false;
        }
        function addCompany(string a)returns(bool){
            if(isCompany(a))
                return false;
            companys.push(company({
                cname:a,
                status:"COMPANY"
            }));
            return true;
        }
        function addBank(string a)returns(bool){
            if(isBank(a))
                return false;
            banks.push(company({
                cname:a,
                status:"BANK"
            }));
            return true;
        }
    }
```

receiptAndFunc.sol 文件定义了结构体 receipt 表示账单，receipts 记录所有有效的欠账信息，fun1-5 五个函数实现基于 receipt 的所有基本操作，fun1 实现功能一：实现采购商品—签发应收账款 交易上链，会自动合并相同债券和债务方的欠账，避免 A 欠 B,B 欠 A 的循环，fun2 实现功能二：实现应收账款的转让上链，会计算支付一方的所有应收款避免超额支付，fun3 实现功能三：利用应收账款向银行融资上链，会计算所有应收款避免超额融资，

fun4 对应实现的是功能四：应收账款支付结算上链，fun5 实现一个自定义功能，应付账款支付结算上链。

```
pragma solidity ^0.4.22;
import "./companyAndBank.sol";
contract receiptAndFunc is companyAndBank{
    struct receipt{

        string rfrom;//债务人

        string rto;//债权人

        uint256 mounts;//债务数额

    }
    receipt[] receipts;
    function fun1(string a,string b,uint256 c)returns(bool){

        //a 欠 b c 元

        if(!isCompany(a))
            return false;
        if(!isCompany(b)&&!isBank(b))
            return false;

if(keccak256(abi.encodePacked(a))==keccak256(abi.encodePacked(b)))
            return false;
        receipts.push(receipt({
            rfrom:a,
            rto:b,
            mounts:c
        }));
        return true;
    }
    function fun2(string a,string b,string c,uint256 d)returns(bool){

        //a 把 b 欠的 d 元 转让给 c

        if(!isCompany(a))
            return false;
        if(!isCompany(b))
            return false;
        if(!isCompany(c)&&!isBank(c))
            return false;

if(keccak256(abi.encodePacked(a))==keccak256(abi.encodePacked(b)))
```

```
                return false;

if(keccak256(abi.encodePacked(a))==keccak256(abi.encodePacked(c)))
                return false;

if(keccak256(abi.encodePacked(b))==keccak256(abi.encodePacked(c)))
                return false;
            uint256 sum=0;
            for(uint j=0;j<receipts.length;j++){

if(keccak256(abi.encodePacked(receipts[j].rfrom))==keccak256(abi.encodePacked(b)) &&
keccak256(abi.encodePacked(receipts[j].rto))==keccak256(abi.encodePacked(a))){
                    sum+=receipts[j].mounts;
                }

            }//计算所有 b 欠 a 的钱

            if(sum>=d){
                uint256 dd=d;
                for(uint i=0;i<receipts.length;i++){

if(keccak256(abi.encodePacked(receipts[i].rfrom))==keccak256(abi.encodePacked(a)) &&
keccak256(abi.encodePacked(receipts[i].rto))==keccak256(abi.encodePacked(c))){
                        if(dd>=receipts[i].mounts){
                            dd-=receipts[i].mounts;
                            if(i<receipts.length-1){
                                receipts[i]=receipts[receipts.length-1];
                                i--;
                            }
                            receipts.length--;
                        }
                        else{
                            receipts[i].mounts-=dd;
                            break;
                        }
                    }
                }
                for(i=0;i<receipts.length;i++){

if(keccak256(abi.encodePacked(receipts[i].rfrom))==keccak256(abi.encodePacked(b)) &&
```

```
keccak256(abi.encodePacked(receipts[i].rto))==keccak256(abi.encodePacked(a))
){
                                if(d>=receipts[i].mounts){
                                    d-=receipts[i].mounts;
                                    receipts[i].rto=c;
                                }
                                else{
                                    receipts[i].mounts-=d;
                                    receipts.push(receipt({
                                        rfrom:b,
                                        rto:c,
                                        mounts:d
                                    }));
                                    break;
                                }
                            }
                        }
                        return true;
                    }
                    return false;
                }
                function fun3(string a,string b,uint256 c)returns(bool){

                    //a 向银行 b 借 c 元

                    if(!isCompany(a))
                        //return false;
                    if(!isBank(b))
                        return false;
                    uint256 sum=0;
                    uint j;
                    for(j=0;j<receipts.length;j++){

if(keccak256(abi.encodePacked(receipts[j].rto))==keccak256(abi.encodePacked
(a)))
                            sum+=receipts[j].mounts;
                    }
                    if(sum>=c){
                        for(uint i=0;i<receipts.length;i++){

if(keccak256(abi.encodePacked(receipts[i].rto))==keccak256(abi.encodePacked
(a))){
                                if(c>=receipts[i].mounts){
                                    c-=receipts[i].mounts;
                                    receipts[i].rto=b;
```

```
                    }
                    else{
                        receipts[i].mounts-=c;
                        receipts.push(receipt({
                            rfrom:receipts[i].rfrom,
                            rto:b,
                            mounts:c
                        }));
                        break;
                    }
                }
            }
            return true;
        }
        return false;
    }
    function fun4(string a)returns(bool){
        //所有 a 欠的钱还清
        if(!isCompany(a))
            return false;
        for(uint i=0;i<receipts.length;i++){

if(keccak256(abi.encodePacked(receipts[i].rfrom))==keccak256(abi.encodePack
ed(a))){
                if(i<receipts.length-1){
                    receipts[i]=receipts[receipts.length-1];
                    i--;
                }
                receipts.length--;
            }
        }
    }
    function fun5(string a)returns(bool){
        //所有欠 a 的钱还清
        if(!isCompany(a))
            return false;
        for(uint i=0;i<receipts.length;i++){

if(keccak256(abi.encodePacked(receipts[i].rto))==keccak256(abi.encodePacked
(a))){
                if(i<receipts.length-1){
                    receipts[i]=receipts[receipts.length-1];
```

```
                    i--;
                }
                receipts.length--;
            }
        }
    }
}
```

以上几乎是链端的所有说明，此外，链端采用 fisco bcos 的 nodeJS SDK，主要是部署合约以及提供调用接口，通过在 Ubuntu 的 command line 打开运行即可。

前端部分：在前端我们使用的是 HTML+CSS 这种较为纯粹的网页描述语言，只要激活本地的 host 就可以在浏览器上面看到我们的项目了。

在后端这里使用的是经典的 Node.js，Node.js 具有的 express 框架负责页面的跳转以及链端数据的调用，通过命令行打开运行，多亏了 Nodejs SDK，使得项目可以正常运行，开始想使用第二阶段使用的 python 及其相关的 SDK 作为后端的数据处理，但是由于 Ubuntu 系统相关依赖包不兼容，所以改用了 Nodejs，这也更符合 Web-Based 的相关要求。

使用服务之前，首先需要初始化全局的 Configuration 对象，用这个方法为各个服务提供必要的配置信息。 Configuration 对象位于 nodejs-sdk/packages/api/common/configuration.js ，配备文件位于 nodejs-sdk/packages/cli/conf/config.json。

```
const Configuration = require('./nodejs-sdk/packages/api/common/configuration').Configuration;
Configuration.setConfig('./nodejs-sdk/packages/cli/conf/config.json');
```

Web3jService 类位于 nodejs-sdk/packages/api/web3j 提供访问 FISCO BCOS 2.0+节点 JSON-RPC；部署合约；调用合约等功能

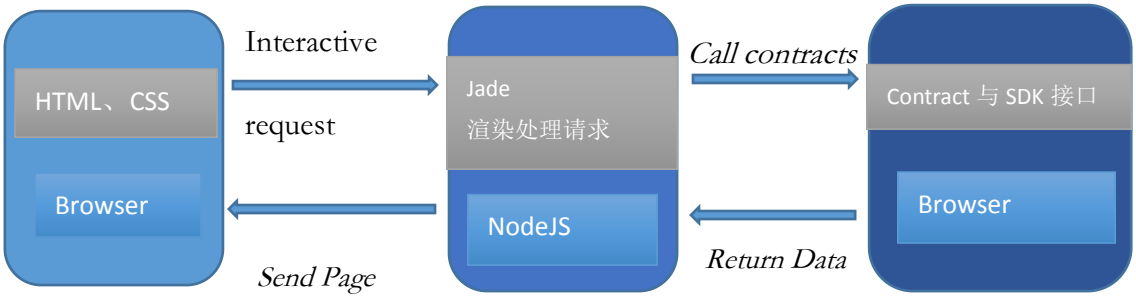
```
res.cookie("contractAddress",ad,{maxAge:"10000000",httpOnly:false});
```

ad (address)是合约地址，在部署时返回 contractAddress 字段，由于每次部署需要的时间很长，采用 cookies 技术，在初次运行时自动进行部署，保存在 cookies 中，下次打开时通过检测 cookies 可以避免重复部署。

```
var api=require('../nodejs-sdk/packages/api/index.js');
var web3j=new api.Web3jService();
var ad="0xf18b20ef02bbac594755ee8e778415bfb967b5c8";
```

后端代码主要在 routes/index.js 中，利用 express 框架自带的 Router 进行页面跳转和处理请求，接受前端的请求，根据请求从链端拉取数据或调用函数，再渲染页面加上数据返回给前端。

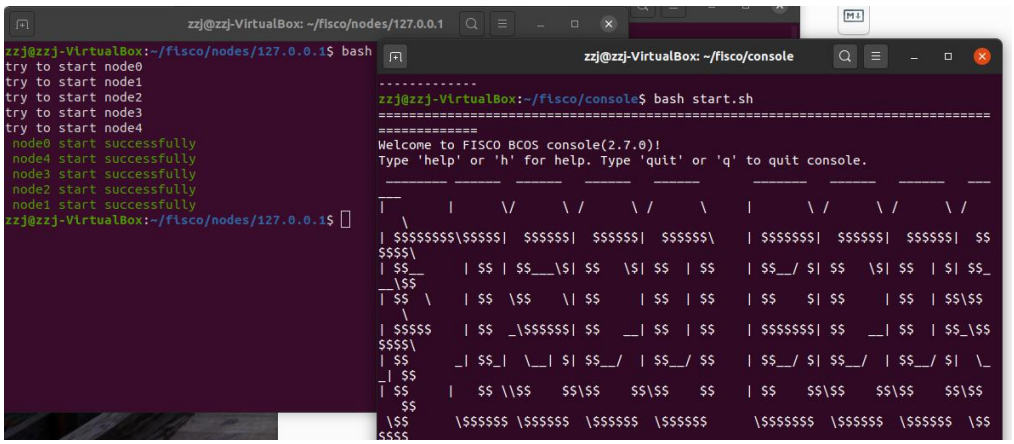
而前端采用 jade 生成 html，位于 views/index.jade，三个变量 title, isDeploy, choices，分别表示标题，是否显示部署界面，以及附加表格。public/javascripts/index.js 文件包含前端的 JS 代码，主要用于界面跳转。



其交互结构如上图所示。

三、 功能测试

首先需要运行 fisco,进入 fisco/nodes/127.0.0.1/目录，运行 start_all.sh，成功会返回四个节点都被激活，会显示如下界面。



返回 my_project 根目录,运行 npm start 启动 express 项目。

在浏览器地址输入 localhost:3000,回车进入前端页面，返回结果如下所示

具体使用展示部分，我们录制了视频，可以通过视频了解我们的项目。

四、 界面展示

启动页面：

功能选择页面

以及功能调用页面：

五、 心得体会与总结

1. Solidity 不支持返回变长变量，所以查询结构体时，无法看到其中的变长数组和 mapping 映射。

2. 经过三个阶段实现了该项目，对 fisco 的架构设计以及原理应用有了逐步加深的理解，同时对其背后包含的区块链技术的原理和应用也理解更加深刻，但是 fisco 是个很大的项目，区块链技术也是包罗万象，在实践中有很大的用武之地。

3. 此外还学习了 solidity 以及智能合约的一些内容，和以往写过的编程语言有很大不同，这也使我们了解到，各种编程语言其实都是完

成任务的工具，如果善于使用它们，并能结合生产生活实际，那么就能做出很棒的项目来。

4. 代码存在某个未找到的 bug,导致可以成功利用 sdk 调用合约，但运行结果却没有写入链端，因为无法写入，打印功能也就无法体现。

本次大作业由于时间关系其实还有很多想法没有实现，也是很久之后对 Nodejs 作为后端框架进行开发网页处理的一次温习，这其中从环境配置、链端合约改进、后端代码、前端网页设计都又遇到一些困难，一个个克服。通过开发这一个平台，也体会到了区块链应用的优势和难处，优势在于数据都是在链上的因此安全性比较可靠，而难处一是合约一旦部署就无法改变，所以尽量要从合约接口等等就前期设计好，否则调试改进比较困难。

5. 由于将数据都只存在链上，每次访问都要调用，加载时间会比较长，可能需要研究更好的接口模式或者预加载等，也可能需要本地化存储以一定数据结构格式保存的数据文件，这次设计令我对区块链应用的设计更加有了思路，也对区块链应用的前景有了新的了解与期待。

6. 由于对 Python 较为熟悉，所以一开始使用的是基于 PythonSDK 的 fisco bcos 平台底层架构的一键安装，其一键安装的设计、合约 IDE 以及 SDK 在很大程度上提升了区块链应用开发的效率，这是我在完成这次作业的过程中切身体会到的。我们在第二次作业中用 Python 搭建后端，所以使用 Python SDK 与链进行交互，但是遇到了种种交叉编译有问题或受到环境约束，不得已放弃 Python 转而

使用较为成熟的另一套后端。fisco bcos 的 Python SDK 相比 Java SDK、NodeJS SDK 等还是比较局限，可以清晰地感觉到 Java SDK 从功能到文档都成熟得多。当然，我也希望自己能有机会为 fisco bcos Python SDK 的完善出一份力，毕竟是陪伴我们小组进行第一次区块链应用开发的工具包，对其使用的便捷性还是有所体会。

六、 后记

为便于评测，我们将实现了的加分项部分写出，如下：

功能：为每个基础功能提供了类型检测以及避免超额支付融资等优化，基本不用考虑用户输入的问题，另外还附带实现了一个自定义功能，应付账款支付结算上链。

前端：采用 HTML 界面，比起命令行应该算得上用户友好，交互逻辑也简洁明了，同时合约中带有一定纠错功能，用户输入也无需过分关心。

不足之处：

后端代码基本集中在路由跳转部分，而且有些代码可以转移到前端，例如用 JS 添加表格而不是重新渲染发送页面。链端代码仅仅实现了比较基础的函数，还可以添加许多基于基础操作的高级操作，例如实现两个银行的兼并（涉及到账本的合并，需要一些经济学算法来支持这样的实现），通过三家公司的共识机制或公司合并消除欠账循环（甲欠乙,乙欠丙,丙欠甲）等。