区块链阶段三报告

第11小组

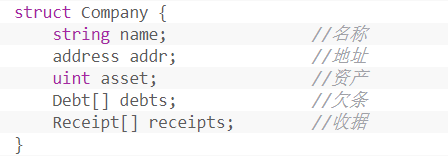
18342069 罗炜乐

18342066 鲁沛

18342071 马靖成

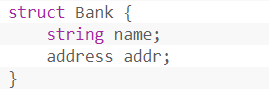
1. 链段设计说明

**公司**



公司采用结构体的形式定义，有名称，地址，资产，欠条集（应付账款）和收据集（应收账款）。所有账户都可以注册公司，并设置自己的初始资产。

**银行**



银行是唯一的，由最初部署合约时的账户自动调用构造函数形成，只有银行具有创建账单（第三方可信机构确认交易），发行货币和融资的功能。

**账单**

一张账单由收据和欠条两部份组成，分别被债权人和债务人所持有。

收据：Id用于在债权人的收据集中索引该账单，borrer\_id用于在债务人的欠条集中索引该账单。



欠条：id用于在债务人的欠条集中索引该账单，creditor\_id用于在债权人的收据集中索引该账单。

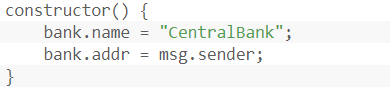


由于creditor\_id和borrer\_id的存在，使得收据和欠条可以一一对应起来并互相索引，形成一张完整的账单。

**变量**

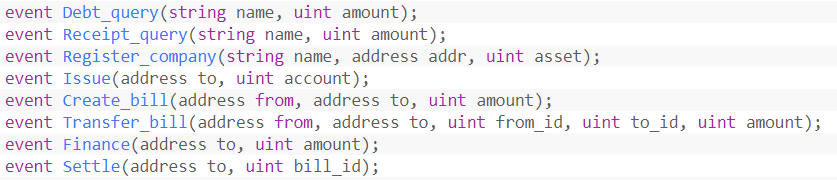


companies是由地址到公司的映射，bank则是合约中唯一的中央银行，拥有最高的权限，在构造函数中被初始化：



**事件**

8个事件作为后续函数的日志，分别是查询总负债额，查询总放债额，注册公司，发行货币，创建账单，应收账款转移，融资和结算。



**函数**

function bank\_info() public view returns(string memory, address)

说明：获取银行信息

function register\_company(string memory name, uint asset) public

说明：注册公司

function getAsset() public returns(uint)

说明：获取公司总资产

function getDebt(uint bill\_id) public returns(address, uint)

说明：根据欠条id获取欠条

function getReceipt(uint bill\_id) public returns(address, uint)

说明：根据收据id获取收据

function get\_total\_debt() public returns(uint)

说明：获取一个公司总的应还账款（总负债额）

function get\_total\_receipt() public returns(uint)

说明：获取一个公司总的应收账款（总放债额）

function issue(address to, uint amount) public

说明：向指定方发行货币

function create\_bill(address from, address to, uint amount) public

说明：签发账单

function transfer\_bill(uint from\_id, uint to\_id, address to, uint amount) public

说明：应收账款转移，涉及到三个人和两个账单

function finance(address addr, uint amount) public

说明：融资

function settle(address to, uint bill\_id)

说明：结算

**总体说明**

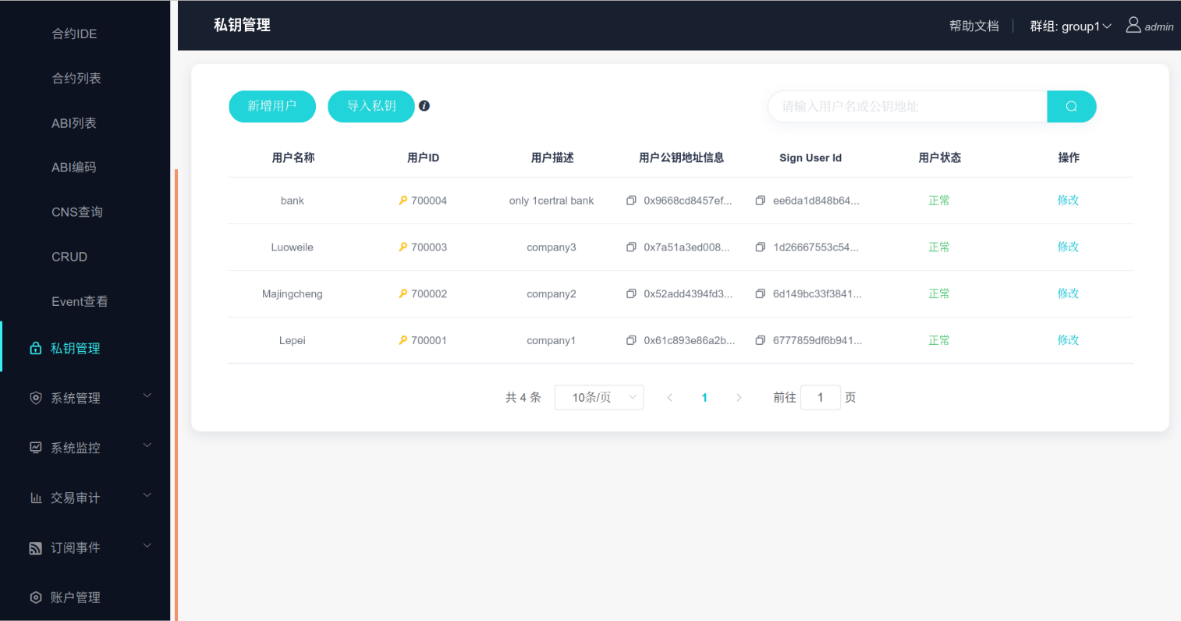
提前创建一个银行账户和几个公司账户，使用银行账户来部署合约。合约部署完成后，使用公司账户注册公司，然后即可开始创建账单等一系列操作。由于区块链的不可篡改特性和账本特性，可以解决需求中提出的问题。

1. 链段功能测试

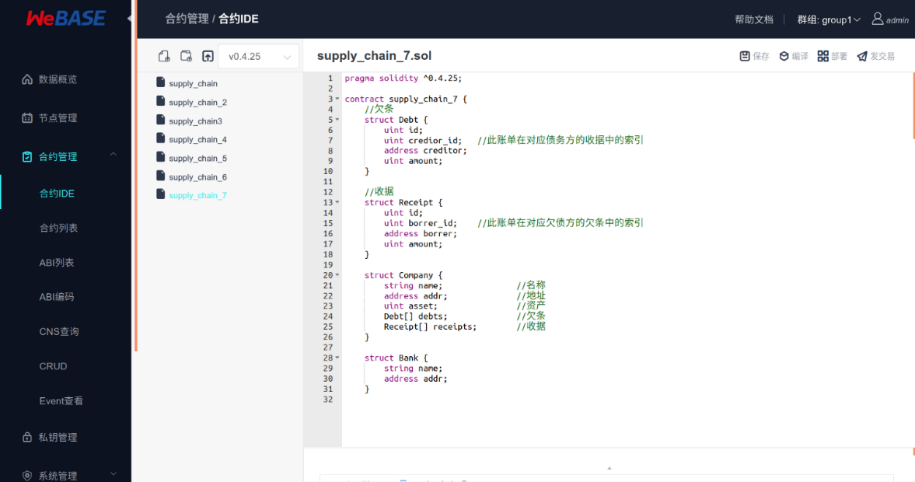
**准备工作：**首先安装WeBASE管理平台：



进入私钥管理页面，创建银行账户bank，公司账户Luoweile，Majingcheng，Lupei：



进入合约管理的合约IDE：



上传合约，保存，编译，部署（使用bank账户），即可开始操作。

使用账户Luoweile创建汽车公司Car：



使用账户Majingcheng创建轮胎公司Tyre：



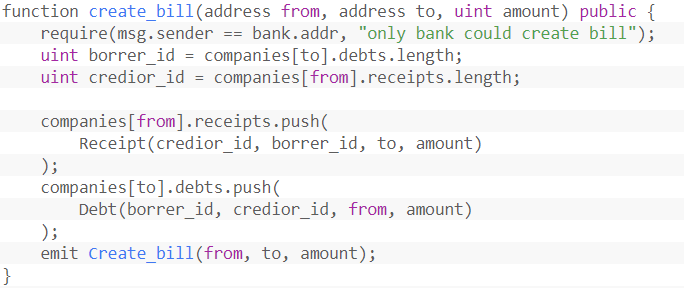
利用账户Lupei创建轮毂公司Hub：



**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。**

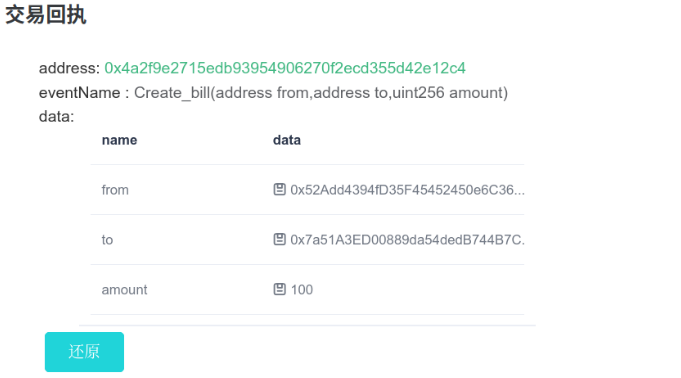
实现函数：create\_bill

实现思路：首先检查函数调用者是否是银行（只有银行才能签发账单），获取欠条在债务人的欠条集中的位置borrer\_id和收据在债权人的收据集中的位置creditor\_id，然后利用push函数在债务人的欠条集和债权人的收据集中分别插入收据和欠条组成一张完整的收据。



使用create\_bill函数创建Car公司欠Tyre公司100的账单：





使用getDebt函数可以在Car公司查到欠Tyre公司100的欠条0：



而使用getReceipt函数可以在Tyre公司查到Car公司欠其100的收据0：

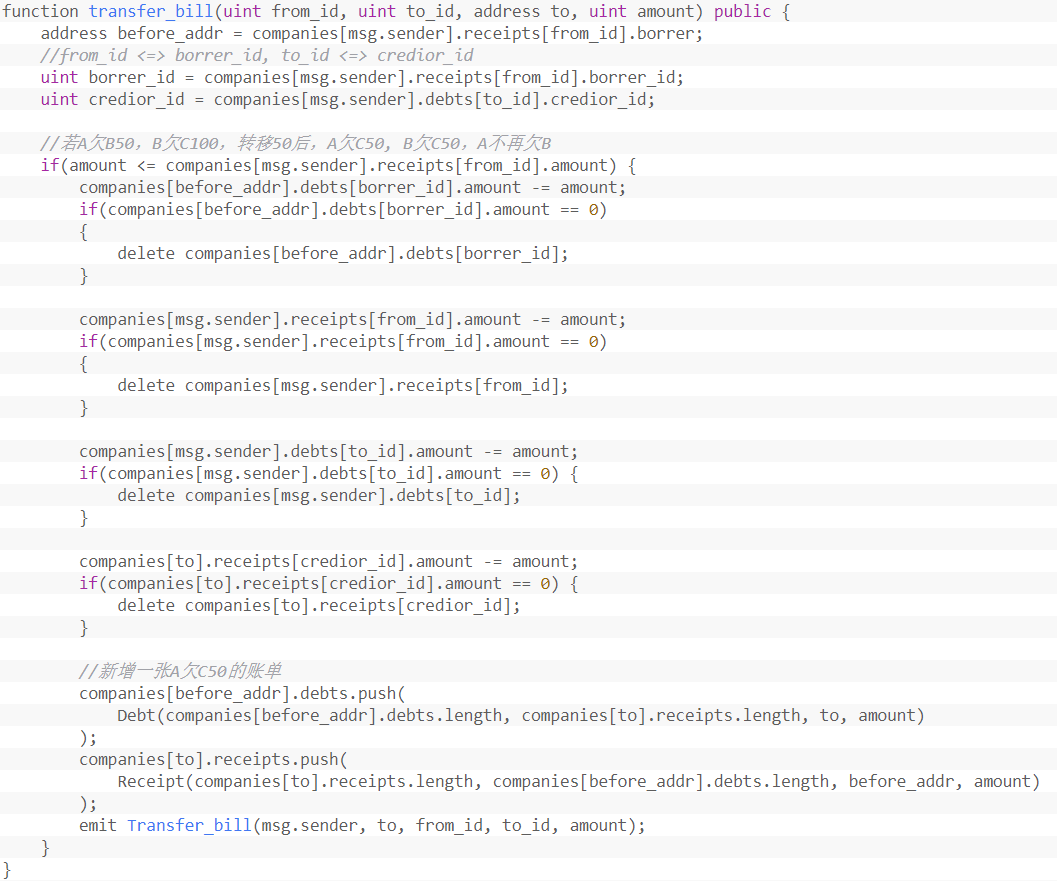


使用create\_bill函数创建Tyre公司欠Hub公司100的账单，则此时Car公司欠Tyre公司100，而Tyre公司欠Hub公司100，此处不再贴图。

**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。**

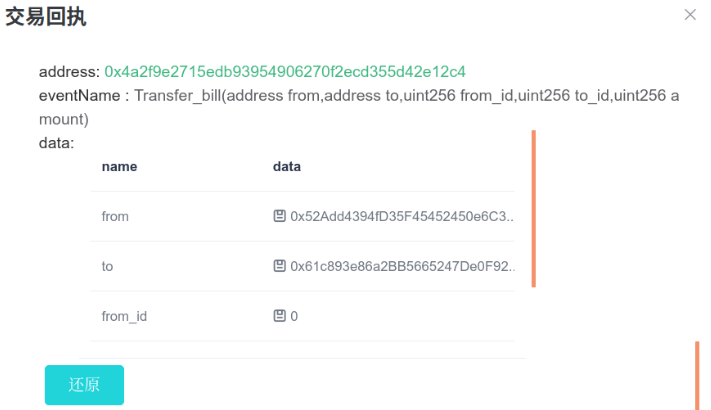
实现函数：transfer\_bill

实现思路：输入参数有账款来源from\_id和账款去向to\_id，即转移主体将某张收据里的应收账款转移给某张欠条里的债权人。首先根据转移主体收据（from\_id）中的borrer找到债务人，根据收据里的borrer\_id找到债务人欠条集中对应的欠条，在该欠条中将欠款额减少指定的金额，相应的，转移主体的收据中的还款额也减少指定的金额；然后根据转移主体欠条（to\_id）中的creditor\_id找到债权人收据集中对应的收据，在该收据中将还款额减少指定的金额，相应的，转移主体的欠条中的欠款额也减少指定的金额；最后新增一张债务人和债权人之间的账单，金额为指定金额。需要注意的是，当账单中的金额减少为0时，要将收据和欠条同时删除。



Tyre调用transfer\_bill，将Car欠Tyre的100转给Hub：





此时的账单状态变更为：Car不再欠Tyre钱100，Tyre不再欠Hub 100，而Car欠Hub 100。

Car调用getDebt函数查看欠条0：



Tyre调用getDebt函数查看欠条0：



Car调用getDebt函数查看欠条1：



Tyre调用getReceipt函数查看收据1：



**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。**

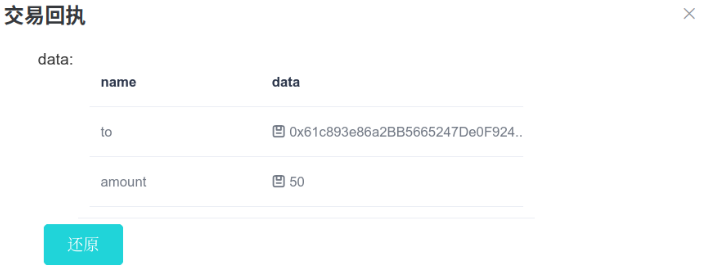
实现函数：finance

实现思路：只有银行有资格融资（给公司发行货币），将公司总的应收账款作为其信用额度，申请的融资金额要小于信用额度。



Hub向银行申请50的融资：





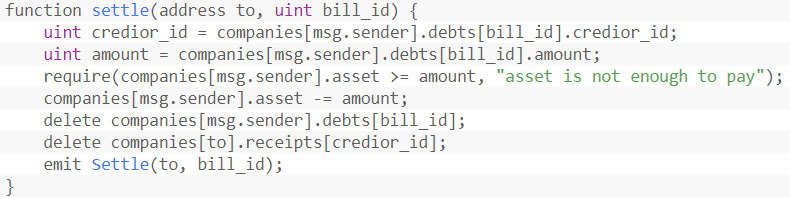
Hub调用getAsset查询自己的总资产，从1000变成了1050：



**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。**

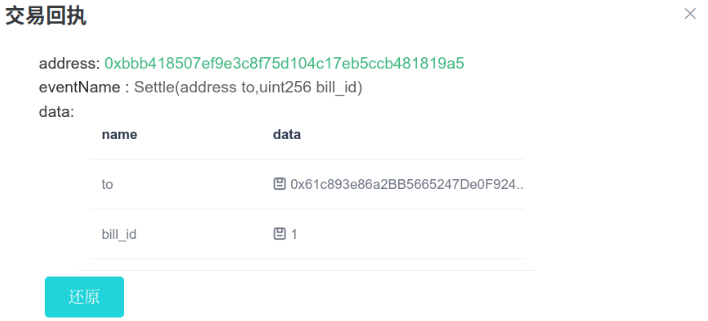
实现函数：settle

实现思路：根据债务人的欠条id找到欠条中记录的债权人的收据id（creditor\_id），然后检查债务人的资产是否足够结算，若足够，则将欠条和收据都删除掉，即删除此账单，并且从债务人的总资产中扣除相应金额。

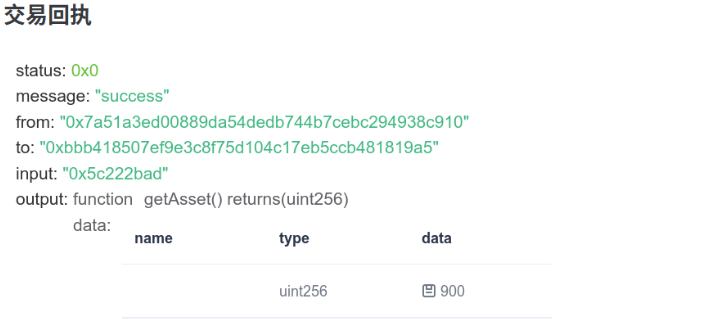


Car调用settle函数，结算欠Hub的100：





Car调用getAsset函数，查询自己的总资产，由于结算了100，总资产从1000变为900，而Hub的总资产从1050变为1150：

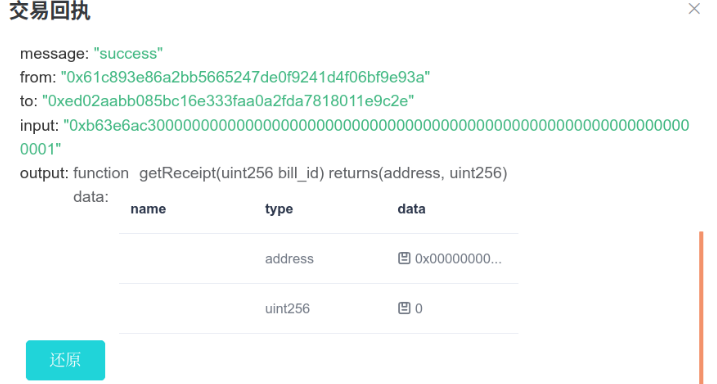




Car查询自己的欠条1，已经被清空：



Hub查询自己的收据1，同样被清空：



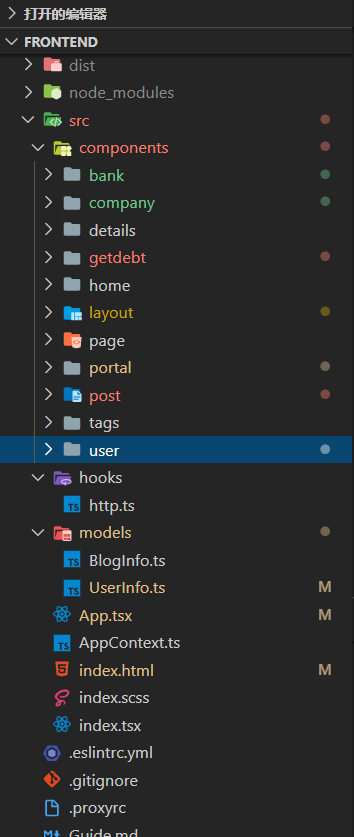
则结算成功。

三、前端部分

**（1）、项目框架：本次前端采用的是JavaScript的react框架。UI 库使用 Fluent UI。**

**对于UI库的使用，参考**https://developer.microsoft.com/en-us/fluentui#/controls/web

**项目结构组织如下：**



我将链上的功能分解为了六个模块，利用侧边栏（layout）来管理：

1、“我的”（portal）部分，这个部分用于管理用户的登录、注册、登出。

2、“查看欠条与收据”（getdebt）部分，这个部分有四个功能，用户根据id查找自己的欠条、收据以及查看自己公司的总负债、总放债数额。

3、“个人公司”（company）部分，功能包括用户创建、修改自身公司的名字、资产。以及查看公司总资产。

4、“银行”（bank）部分，包括仅允许银行用户操作的两个功能：签发账单和融资。

5、“个人金融操作”（user）部分，包括应收账款转移和结算（还钱）两个功能

6、还额外提供了一个直接操作合约的平台以及对合约的使用的解释说明。

**（2）、前端效果展示：**

**1、注册页面**：



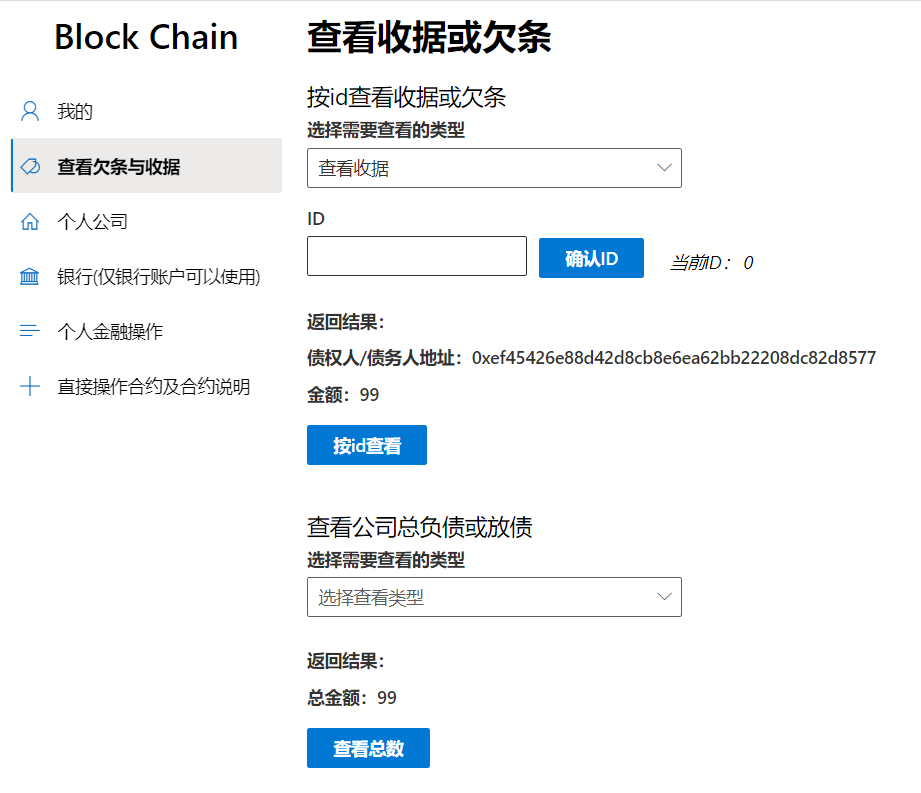
**2、登录页面**



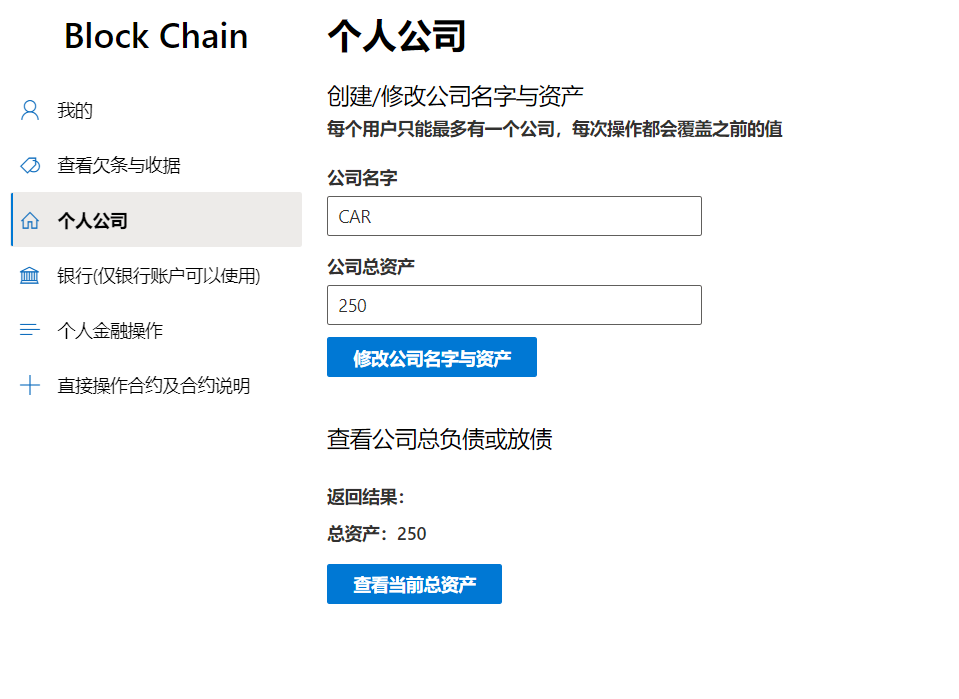
**3、登录后**



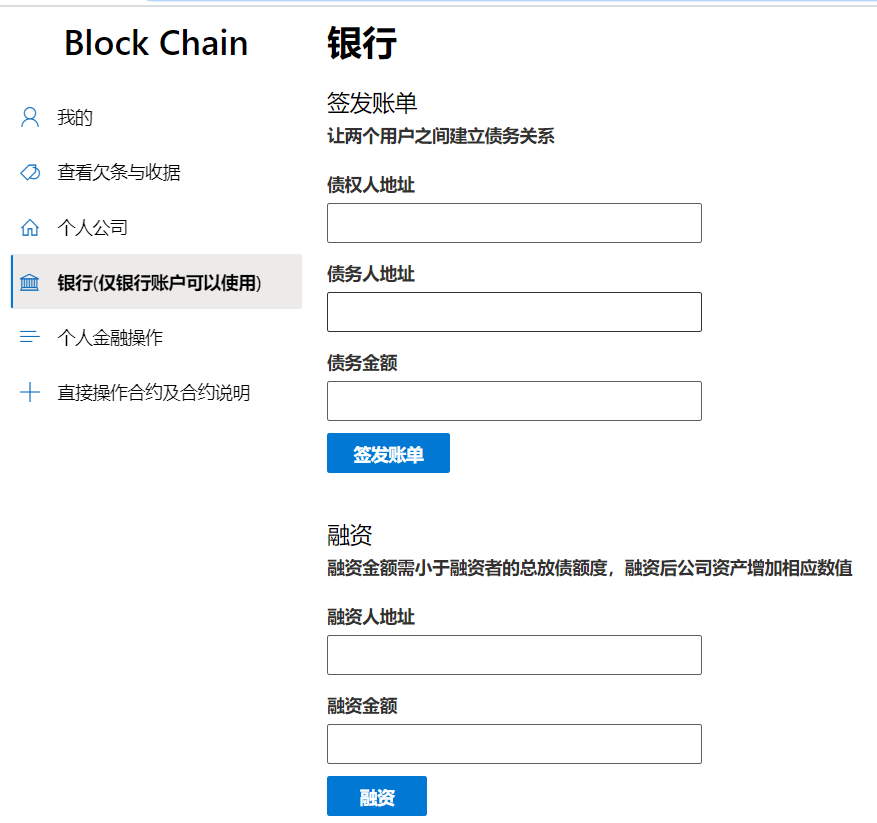
**4、查看欠条与收据页面**



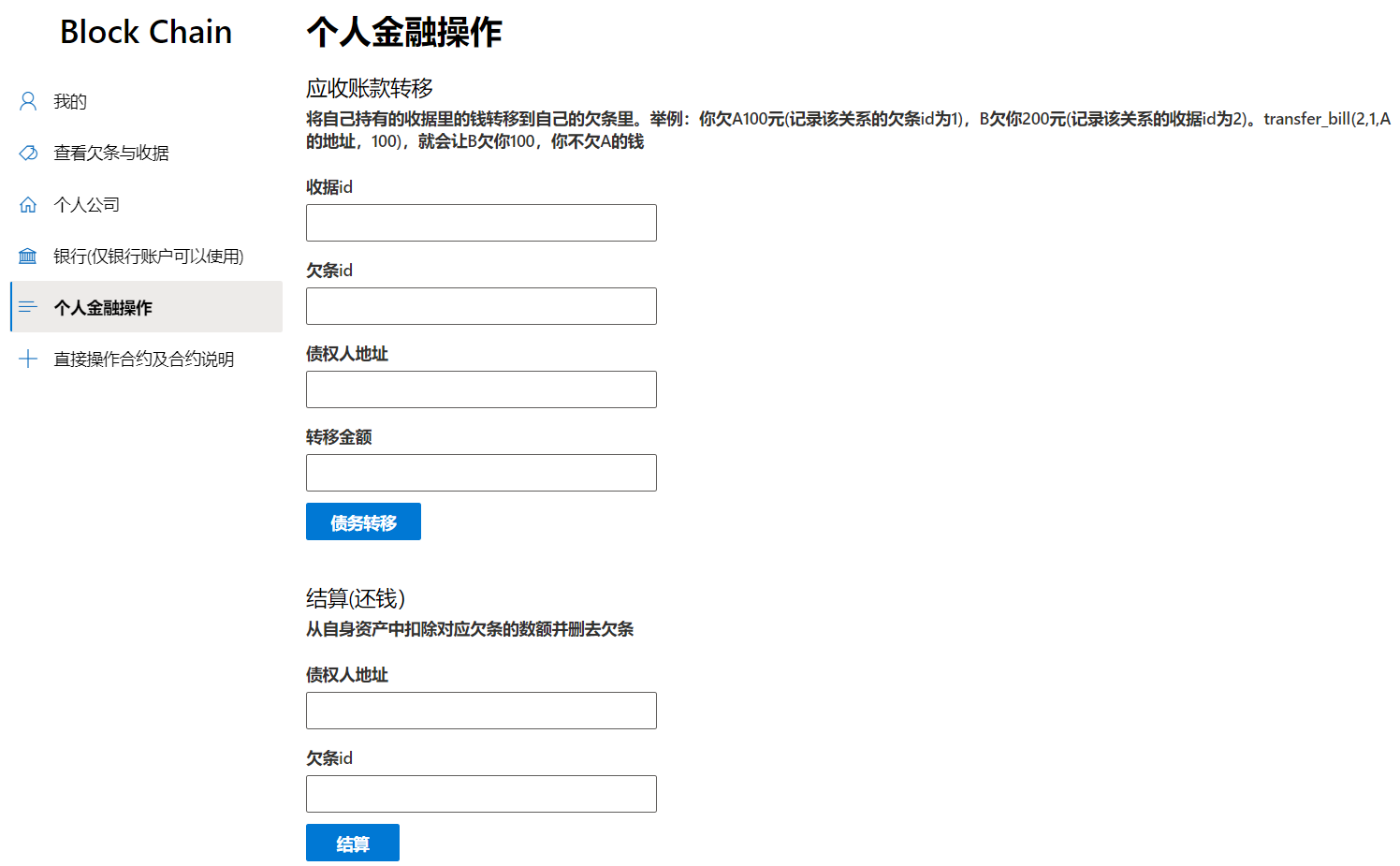
**5、个人公司页面**



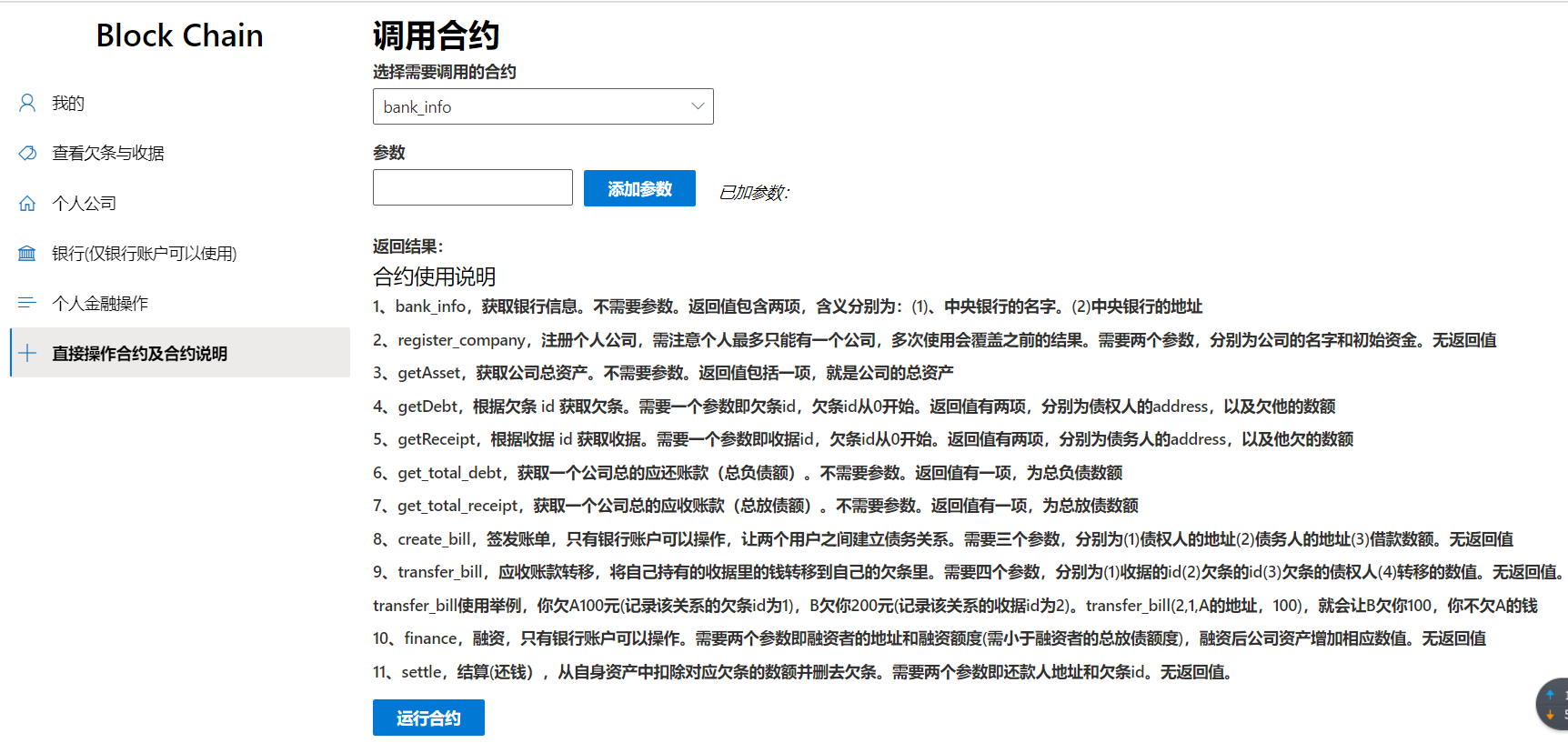
**6、银行页面**



**7、个人金融操作部分：**



**8、直接调用合约部分**



**（3）、前端实现关键部分代码：（实现过程的较为详细教学放在前端的guide.md，这里只说最关键的部分）**

**1、将输入框中的内容绑定为状态，便于改动页面上的元素且便于作为参数传递。**

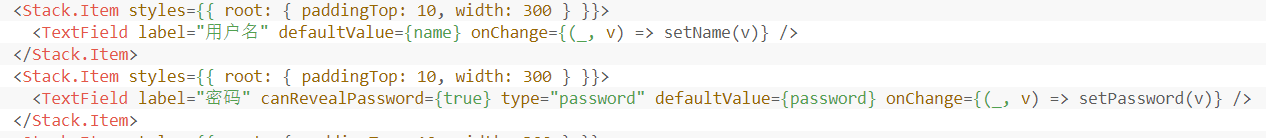
**以Login页面中的用户名和密码为例，演示这一过程的重要性。**

**一、这部分是定义状态和改变状态的函数，以及如何利用状态来作为参数发送http请求**



**二、在这个部分中，将输入框中的内容绑定在状态中。**

**注意到我们调用了上面的setname和setpassword**



**三、以“直接调用合约”部分的选择合约的多选框为例，演示多选框内容以及多个参数的绑定**



**多选框的内容定义如下（就是我们的所有合约）：**

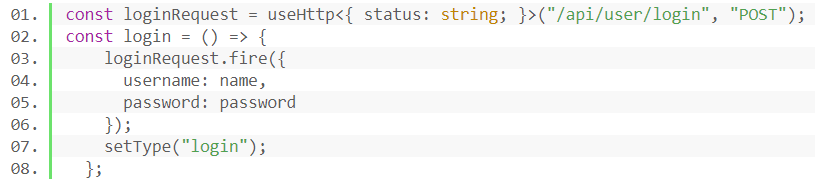


**然后我们在多选框中调用onchange函数记录当前选中的title(合约名)，在”添加参数”按钮中记录当前的参数列表。**



**2、演示发送http请求和解析的过程**

**这个部分应该是前后端交互的重中之重，react中有两种发送请求的方法。一种是提前预设好请求类型和返回体，然后利用fire函数去发送**



**另一种则是在发送时定义变量、类型等，我们以“个人公司”页面中的两个功能为例，一个是改公司名字与资产的请求post1(需要两个参数，在代码中命名为title，title2.).另一个是查看公司总资产的请求post2（无参数）。**



**可以看到我们在发出时才定义了http请求的类型，和content-type以及参数的key-value值。**

**然后是http返回参数的解析，react将返回参数解析为字典对象。所以一种解析方法就像上图中的，直接取data.result[0]，就取到返回值中key为result的字段。**

**另外一种情况下，我们也许需要将data中的参数都完整的展示出来，这个时候我们就需要用到JSON.parse，将返回的字典对象再解析为json字符串，然后输出。在“直接调用合约”部分我们就是这样展示结果的。如下：**



**3、演示发送react如何将http元素和JavaScript元素结合，到达渲染和绑定同时进行。**

**在本次项目中，我使用了fluentUI的stack来简单的组织元素。以个人公司的元素为例**



**代码组织如下：**

