



**本科毕业设计**

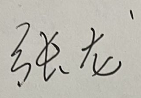
|  |  |  |  |
| --- | --- | --- | --- |
| **题　　目** | **基于区块链的校园智能合约转账以及去中心化dapp** | | |
| **学生姓名** | 张龙 | **学　　号** | 18061320 |
| **班　　级** | 180613 | **专 业** | 计算机科学与技术 |
| **学 院** | 信息工程学院 | | |
| **指导教师** | 陈健 | **职 称** | 副教授 |

**2022年 6月 5日**

学位论文原创性声明

本人所提交的学位论文《基于区块链的校园智能合约转账以及去中心化dapp》，是在导师的指导下，独立进行研究工作所取得的原创性成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中标明。

本声明的法律后果由本人承担。

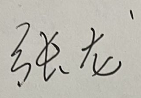


****论文作者（签名）： 指导教师确认（签名）：

2022年6月5日 2022年6月5日

学位论文版权使用授权书

本学位论文作者完全了解防灾科技学院有权保留并向国家有关部门或机构送交学位论文的复印件和磁盘，允许论文被查阅和借阅。本人授权防灾科技学院可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存、汇编学位论文。



****论文作者（签名）： 指导教师（签名）：

2022年6月5日 2022年6月5日

**基于区块链的校园智能合约转账以及去中心化dapp**

作 者 张 龙

指导教师 陈 健

**摘要**  通过区块链的加密和不可篡改性作为数据存储和逻辑处理的后端(采用geth客户端)，用solidity语言编写搭建在私有链上的智能合约，智能合约预留接口供前端使用，前端采用MetaMask浏览器钱包扩展，它会对当前浏览器页面注入一个web3.js的对象，这个对象可以直接连接区块链。

功能目前分为两个模块，投票网站采用VUE+TYPESCRIPT完成与用户交换，主要有连接钱包，获取钱包投票货币余额，获取投票合约信息，添加候选人和参与投票功能。去中心交易所采用REACT+TYPESCRIPT，主要有连接钱包，添加交易对，子货币置换功能。

**关键词: 区块链;以太坊; solidity; 投票;交易所;去中心App;**

**Design and Implementation of Enterprise Office OA Management System Based on Spring Boot**

Author Zhang Long

Instructor Chen Jian

**Abstract** Through the encryption and tamper-evident nature of the blockchain as the back-end of data storage and logic processing (using geth client), the smart contract built on the private chain is written in solidity language, the smart contract reserved interface for front-end use, the front-end uses MetaMask browser wallet extension, it will inject a web3.js object to the current browser page, this object can be directly connected to the blockchain.

The function is currently divided into two modules, the voting site uses VUE+TYPESCRIPT to complete the exchange with the user, mainly to connect the wallet, get the balance of the wallet voting currency, get voting contract information, add candidates and participate in the voting function. The decentralized exchange uses REACT+TYPESCRIPT, which mainly has the functions of connecting to the wallet, adding transaction pairs and sub-currency replacement.

**Keywords:BLOKCHAIN;ETH; SOLIDITY;VOTE;EXCHANGE; DAPP;**

**目 录**

[引言 1](#_Toc28941)

[1系统分析 2](#_Toc30118)

[1.1可行性分析 2](#_Toc12989)

[1.2需求分析 2](#_Toc2242)

[2系统设计 4](#_Toc21771)

[2.1系统架构 4](#_Toc25238)

[2.2系统功能结构设计 5](#_Toc2675)

[2.3系统工作流程设计 6](#_Toc27218)

[3关键技术分析 8](#_Toc24590)

[3.1 geth客户端 8](#_Toc28372)

[3.2 MetaMask 8](#_Toc16148)

[3.3 solidity语言 9](#_Toc23755)

[3.4 Vue/React](#_Toc2995) 10

[3.5 EVM（以太坊虚拟机） 10](#_Toc3203)

[3.6 SHA-3（Keccak-256） 11](#_Toc12396)

[3.7 非对称加密（ECDSA椭圆曲线数字签名算法) 11](#_Toc12396)

[3.8 Merkle Tree（默克尔树） 11](#_Toc12396)

[3.9 恒定乘积做市商 12](#_Toc12396)

[4 系统实现 13](#_Toc9426)

[4.1 启动区块链节点 13](#_Toc28084)

[4.2 ERC20代币设计 15](#_Toc3459)

[4.3 投票网站：投票合约设计 20](#_Toc14501)

[4.4 投票网站：连接钱包 23](#_Toc27074)

[4.5 投票网站：合约信息展示 26](#_Toc2128)

[4.6 投票网站：添加候选人和投票 30](#_Toc31398)

[4.7 交易所：合约设计 33](#_Toc20115)

[4.8 交易所：添加流动性与移除流动性 36](#_Toc21297)

[4.9 交易所：交易 39](#_Toc21297)

[5系统测试与维护 41](#_Toc29814)

[5.1 系统测试目的 41](#_Toc11116)

[5.2 测试的重要性 41](#_Toc17528)

[5.3 测试内容 41](#_Toc8508)

[5.4 测试结果 42](#_Toc11810)

[结论 44](#_Toc3339)

[致谢 45](#_Toc20978)

[参考文献 46](#_Toc18183)

引言

1998 年，戴伟（Wei Dai华裔大牛）首次提出通过解决数学难题制作去中心的b-money。

2005 年，芬尼（Hal Finney）引入了reusable proofs of work概念(可重复使用的工作量证明机制)，提供了创造密码学货币的理论。

2009中本聪启动比特币区块链网络。(第一个真正应用，并且稳定运行至今)

2013年，Vitalik Buterin发布了以太坊白皮书[1]，启动了以太坊项目。（解决了比特币转账慢的问题，并且新增了智能合约）。

区块链技术日新月异，蓬勃发展，那么利用区块链技术能做什么呢？不妨想想这样两个应用场景：

1.如果某个企业需要一个考勤系统或者工时系统，通常会设计一个中心化的系统中心，那么就会出现问题。中心化的系统中心会出现因为事故系统宕机造成数据丢失；某些管理人员可以通过修改数据库直接更改考勤数据，这种不透明不公开不稳定的方式会造成管理的徇私舞弊，员工工作积极性降低。

2.假如学校需要选出“最美（帅）女（男）神”，又或者需要学生投票选出“风采班级”，通常采用微信的投票方式，但微信投票完全可以依靠脚本进行刷票，这样造成结果不真实不透明不公开。

由于非对称加密目前不可破解性，加上以太坊提供的智能合约功能，完全可以搭建一条以太坊私有链，利用区块链的安全性和不可篡改性作为服务器后端，编写相应智能合约提供服务，通过以太坊提供的web3.js接口与前端进行交互。

# 1系统分析

1.1可行性分析

**1.1.1技术可行性**

区块链经过上十年的发展，在全世界最优秀的技术人员的努力下，已经诞生了许多优秀的现成技术，像采用以go语言为基础开发的geth客户端，可以非常容易的启动一个底层的区块链节点，建立在区块链上的solidity语言，已经经过大量的迭代跟新，完全可以胜任处理复杂的逻辑，建立在区块链上的MetaMask钱包也非常完善。通过VUE或者REACT等优秀的前端框架能非常容易的搭建用户交互页面。以上都说明技术完全可行。

**1.1.2经济可行性**

开发此项目基本是利用完全开源的项目或者技术进行搭建，成熟的技术意味着开发成本低。在维护成本上，只需要两到三台普通服务器的资源，还可以通过降低区块难度让服务器耗能相当低。以上说明，经济完全可行。

**1.1.3操作可行性**

在操作方面，采用B/S客户端与用户进行交互，用户只需访问域名便可以轻松获得服务。关于钱包方面，采用的是开源的MetaMask钱包，Google浏览器和Firefox浏览器可以方便的安装扩展。以上说明，操作完全可行。

1.2需求分析

**1.2.1用户需求**

在投票网站模块，通过分析需要以下功能：

1. 钱包交互：连接钱包，获得当前用户授权钱包的地址，参与投票的代币数量。
2. 合约交互：获取合约的存储信息，当前合约的地址，提案合约拥有者，储存的候选人id，姓名，得票数量。
3. 用户交互：用户选择候选人进行投票，合约拥有者进行添加候选人。

在交易所模块，通过分析需要以下功能：

1. 钱包交互：连接钱包，获取当前区块链网络。
2. 子货币交互：获取当前用户两种子货币余额并生成交易对合约。
3. 用户交互：子货币交换，添加流动性，移除流动性。

**1.2.2性能需求**

1. 实用性

利用区块链作为后端逻辑处理以及数据存储，可以很方便的对信息加密，与传统的采用数据库存储相比，无需使用MD5等不安全的算法，而可以直接用客户端集成的非对称加密算法处理用户账户信息，非常的实用。

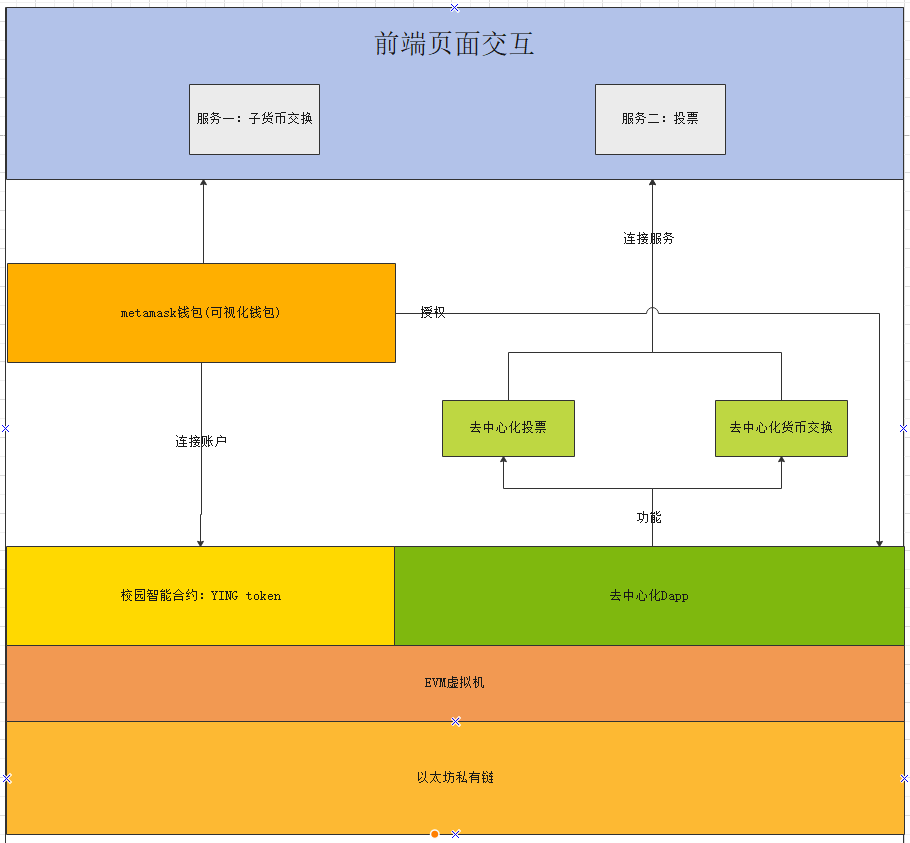
（2）可维护性

系统非常容易维护，因为是在geth客户端的基础上进行开发，钱包和区块链都不需要自己进行维护，维护人员只需关注solidity智能合约的编写以及与前端进行交互的部分即可。

# 2系统设计

## 2.1系统架构

本系统区块链底层采用以太坊客户端进行搭建，当启动geth客户端并指定好网络id=15，便初始化了自己的一条区块链。客户端底层采用go语言编写，有良好的并发性。可以在自己的区块链上采用solidity语言编写智能合约，发行数字货币；同时，可以在区块链上建立投票和交易所合约，并启动rpc（远程过程调用）和web3，这样可以通过MeatMask钱包连接到区块链，MeatMask会将一个web3对象注入到当前浏览器页面，这样就可以通过web3.js唤起钱包，以及获取链上数据。前端用户交互采用Vue+TypeScripe和React+TypeScript进行编写，达到安全快速开发。

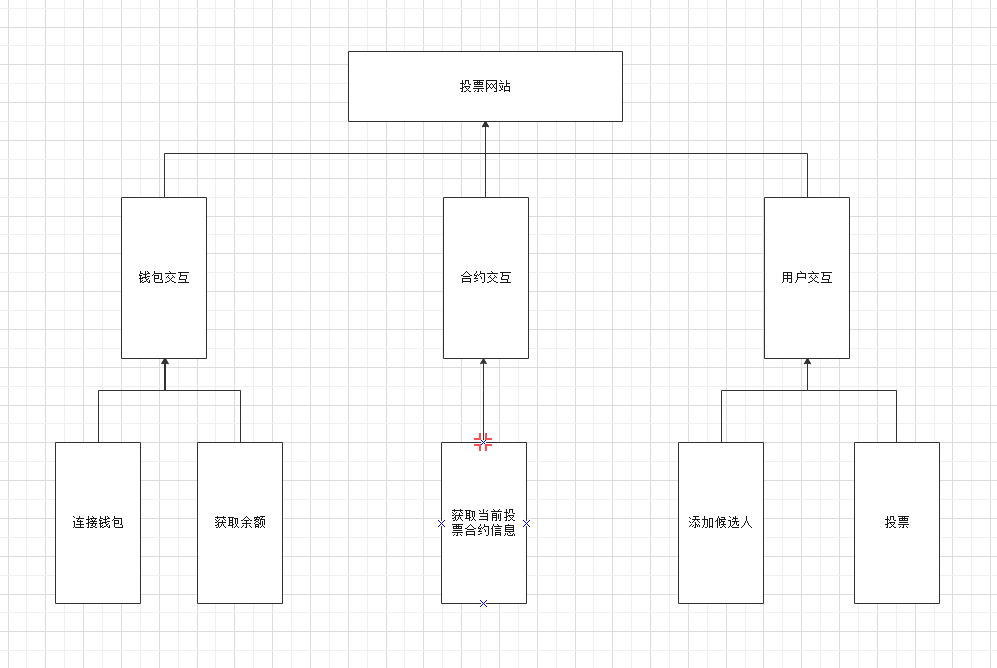


**图2-1校园区块链系统架构图**

## 2.2系统功能结构设计

在投票网站模块，通过分析需要以下功能：

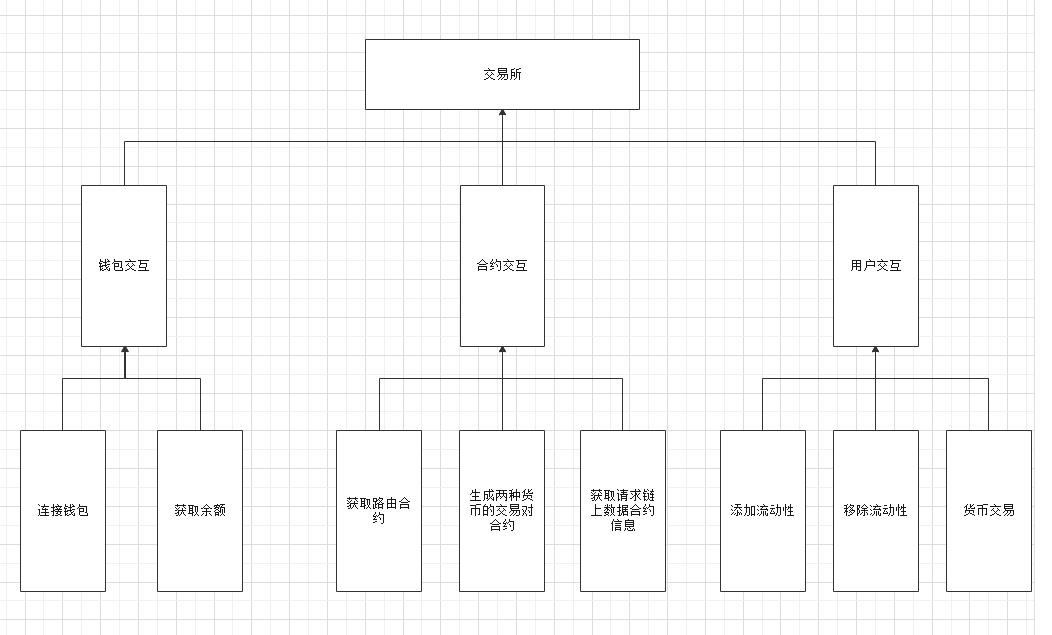
1. 钱包交互：连接钱包，获得当前用户授权钱包的地址，参与投票的代币数量。
2. 合约交互：获取合约的存储信息，当前合约的地址，提案合约拥有者，储存的候选人id，姓名，得票数量。
3. 用户交互：用户选择候选人进行投票，合约拥有者进行添加候选人。

****

**图2-2 投票网站功能结构图**

在交易所模块，通过分析需要以下功能：

1. 钱包交互：连接钱包，获取当前区块链网络。
2. 合约交互：获取当前用户两种子货币余额并生成交易对合约，获取路由合约（主要与其他合约进行交互），获取请求链上数据合约信息（主要获取当前区块数，gas价格等）
3. 用户交互：子货币置换，添加流动性（当同时拥有两种货币时候，可以将余额转给交易所，交易所获取余额生成交易对合约，并将流动性赋予用户，宣布此用户为这些流动性的拥有权），移除流动性（当移除用户拥有的流动性的时候，根据流动性返回用户质押的两种子货币）。



**图2-3 交易所网站功能结构图**

## 2.3系统工作流程设计

**2.3.1投票网站工作流程**

在网站加载完成后，系统会自动获取当前合约地址，合约owner（拥有者），以及最小参与投票代币数量，并将储存在区块链中的当前候选人名单在前端生成列表。

当用户打开网站时候会先检测是否授权连接过钱包（通过metamask注入的web3对象），若检测到没有连接会提示haven’t link wallet并用红色进行醒目提示；当选择link wallet并在唤起的钱包中对此网站进行授权，meatmask就会注入一个名为ethereum的web3Provider，如果钱包连接的是自己的私有链，这个web3的提供者就是自己私有链的web3对象，网站要通过这个web3对象与区块链进行交互。当成功连接钱包后，会获取钱包地址，以及能参与当前投票合约的代币数量，并显示绿色。接下来用户就可以使用投票功能。

在前面都已完成的情况下，用户可以选择列表中的一个人并发起投票，这时候钱包会弹出交易信息，因为以太坊区块链记录交易是需要消耗gas（gas可以理解为这次转账需要消耗的系统资源，为了防止被人恶意死循环）当确定发送交易后，钱包会将交易信息打包并广播到整个区块链（padding状态），如果区块链中有机器在记录交易，便会将当前交易执行，交易中data数据会存储合约调用函数的哈希信息以及参数，EVM虚拟机会执行这个函数，如果函数没有出错，便将函数运行结果的区块打包作为下一个区块的父区块。经过以上步骤，用户投票成功，每个用户只有一票投票权，并且要拥有大于最少参与投票数量的代币才能投票成功。

另一个功能是添加候选人，但是只有投票合约的拥有者可以，当用户是合约拥有者时，可以添加候选人，并输入候选人名字，然后发送交易（和上面一样），当交易成功确认后，前端会自动更新列表，并将此候选人票数初始化为0。

**2.3.2交易所网站工作流程**

交易所工作流程与以上大致相同，先会检测钱包和网络，如果没有连接可以授权钱包进行连接。当成功连接后可以打开搜索栏输入合约地址，点击import导入代币就能获得代币数量。但此时还不能进行交易，因为需要添加流动性。用户选择需要添加的流动性的交易对，如token A和token B，用户首先授权(approve)并提交交易，授权就是允许交易所合约调用用户代币的数量，然后点了供应(supply)，当成功提交并打包到区块链后，交易所的工厂合约会获得用户添加的token A和token B，并通过工厂合约生成一个pairA-B的交易对，合约通过一个公式计算出流动性赋予给用户，当用户不想提供流动性时，可以移除流动性并根据流动性和流动池内的两种代币数量返还两种token给用户并给予一定的手续费奖励。当区块网络上有两种代币时，可以选择将用户拥有的一种代币交易成另一种代币，交易得到的数量取决于初始添加流动性的比例以及恒定乘积做市商算法(见3.9)。

# 3关键技术分析

## 3.1 geth客户端

以太坊客户端众多，基于各种语言开发的都有，这与区块链崇尚开放的价值观有关。Geth（全称go-ethereum）是以太坊官方社区开发的客户端，是客户端里的领头羊。所以可以使用Geth命令搭建以太坊私链，也可以使用基于Geth提供的交互式命令控制台，与以太坊网络环境进行交互。使用geth客户端可以轻松同步现有的区块链网络或者开启一条自己的私有区块链。

geth客户端有以下特点：

1. 利用go语言开发，很好的利用了go的高并发性，这为庞大的区块链网络同步提供了基础。
2. 可交互，当使用geth启动一个区块链网络的时候，可以使用geth客户端的交互式命令控制台，如键入eth.blockNumber会返回当前区块链的高度。

私有链的特点：

1. 建立在本地不同步以太坊主网以及测试网络，所以无需额外的磁盘开销。
2. 网络上只有自己的交易，不会出现拥堵情况。
3. 可以自己任意分配以太，以及挖取以太。

## 3.2 MetaMask

MetaMask是基于以太坊的以扩展形式存在的浏览器加密货币钱包，其中内置了连接以太坊区块链以及发送交易的操作界面，如任何实现了erc-20代币标准的数字货币都可以通过MetaMask进行发送交易。开发人员通过使用JavaScript插件（如 Web3js 或 Ethers）来定义 Metamask 和智能合约之间的交互，从而实现 Metamask 与其去中心化应用程序之间的连接。

MetaMask特点：

1. 以浏览器的扩展形式存在，方便安装与使用
2. 会将当前钱包连接的区块链网络的web3/Ethers注入到当前浏览页面，这样当前网站可以直接获取区块链上的数据
3. 可以自定义添加区块网络，任何兼容以太坊的区块网络都可以配置

## 3.3 solidity语言

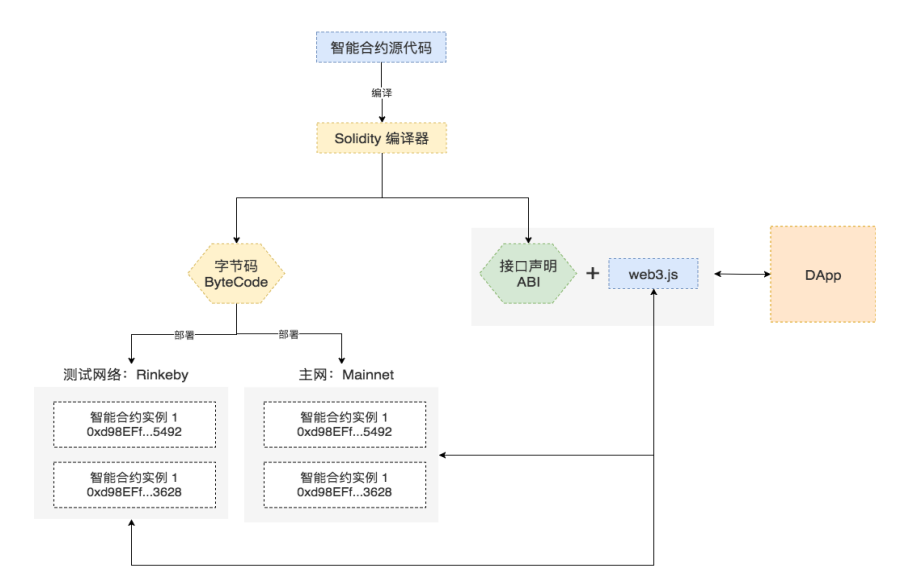
solidity是一门为实现智能合约而设计的高级程序语言，它借鉴了c++，python，以及javascript。作为高级语言，solidity支持继承、库和自定义数据类型。solidity编译字节码主要为了实现在EVM虚拟机上复杂的业务。

数据类型和关键字：由于以太坊区块链的特性，solidity内置了address数据类型专门处理账户地址，并通过关键字payable指定运行函数是否可以接收以太。

存储：有两种数据类型memory和storage。memory是内存，在运行时候暂时存储在网络机器的内存中，一旦运行结束就会消失。storage是永久化存储，声明为storage的数据会永久化存储在区块链网络上。通常合约的成员变量默认为storage类型，复杂的数据类型（如mapping(address=>uint））也声明为storage类型，而函数内的局部变量包括参数为memory类型。

安全：由于区块链常常涉及金融安全，solidity内置了重要的require函数作为安全检查，与其他语言不同的是，所有没有通过检查的操作一律回滚。

编译器：目前最好用的solidity编译器是remix（基于web），其内置了javascript虚拟机，以及可以通过网络连接web3。



**图3-1 solidity编译运行过程**

## 3.4 Vue/React

vue和react是当前世界上最流行的两大前端框架。react它由Meta（前身为 Facebook）开发，由社区进行维护。由于大公司对性能越来越高的要求，要求减少javascript对浏览器dom对象频繁的渲染并且能基于组件化模块化开发，所以用react开发的项目性能高且非常容易开发和维护。vue与react大同小异，他们的共同特点是：

1. 单页面应用。所有渲染都是基于一个页面的。
2. 组件式开发。需要的每个页面上一个模块都可以声明成一个组件，这个组件被重复引用在不同的地方，组件之间耦合性低。
3. 采用diff算法。它们都不直接操作真实DOM，而是生成虚拟DOM，只有当状态发生改变的时候才会修改真实DOM，这样极大的节省了浏览器资源。
4. 修改页面简单。无需操作真实DOM，只需修改状态，所有使用了此状态的DOM都会被更新。

## 3.5 EVM（以太坊虚拟机）

要使用以太坊虚拟机建立区块链并进行开发就需要理解EVM中的一些概念。

1. 账户：以太坊有两类账户，一个是普通账户，一个是合约账户，两类账户对EVM虚拟机来说是等同的，他们都有一个data域。在普通账户中，这个data域是空的，而在合约账户中data域存储的是被编译的合约字节码。
2. 交易：可以看作从一个账户发送到另一个账户的消息。发送的内容有两部分，一部分是以太币，一部分是数据。如果发送的地址是合约账户，那么会执行消息里的数据（通常是合约某个函数的签名以及参数）。如果是普通账户，只是接收以太币。如果是全零地址，这通常是合约部署命令，EVM虚拟机会将数据部署成一个合约，并随即生成一个地址。
3. gas：gas是瓦斯气体的意思，也就是量化发送交易需要消耗的系统资源。由于区块链网络是没有中心化机构进行维护的，如果有人恶意通过死循环代码使整个网络拥堵瘫痪，造成的后果不可想象。通过gas来量化每次执行交易需要的资源，这部分费用由用户支付，用户发送交易时有gas price表示用户预支付gas，如果gas消耗殆尽程序仍然未执行完，则会抛出out-of-gas异常，这次交易所有操作全部回滚。

## 3.6 SHA-3（Keccak-256）

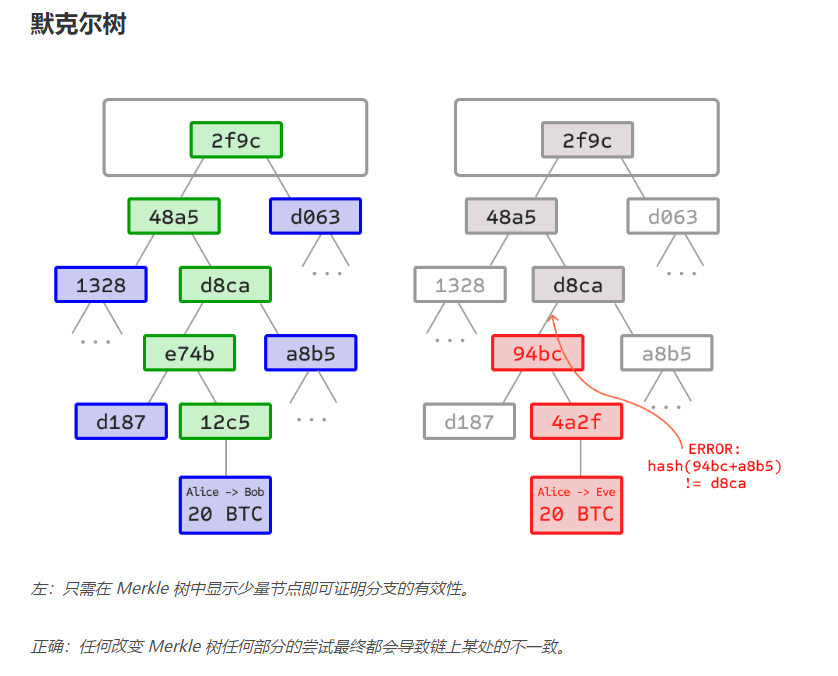
Keccak-256[3]是一种不可逆的哈希算法，哈希碰撞率极小，用于对消息的摘要。在区块链中通过对消息的hash可以判断消息是否被篡改。

## 3.7 非对称加密（ECDSA椭圆曲线数字签名算法)

ECDSA[6]是采用ECC[4]对DSA[5]算法的模拟，通过椭圆曲线生成签名对（r，s），当将交易发送给区块链网络，区块链可以通过（r，s）判断消息是否由签名者发送，从而保证消息来源的可靠性。

## 3.8 Merkle Tree（默克尔树）

Merkle Tree[1]是由消息摘要hash一层一层构建的树，父哈希为孩子hash的hash，也就是层层hash。由于区块链是公开的网络，为了预防攻击者伪造区块，采用Merkle Tree树的方式只需验证当前区块的hash便可验证之前所有区块正确。

****

**图3-2 Merkle Tree（来源:** **[1]）**

## 3.9 恒定乘积做市商

在交易所的设计中，需要思考一个问题，怎么保证一个代币兑换另一个代币的比例？

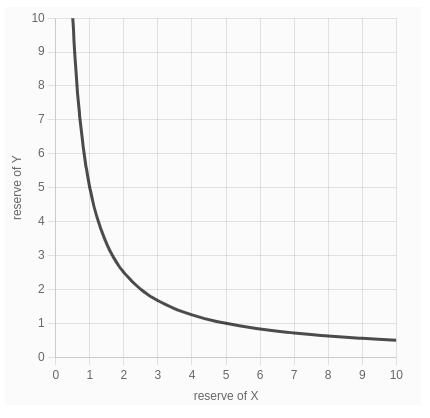
假设x与y分别为两个代币在流动池中的储备量，设Px为x的价格，Py为y的价格，则有：

假设由2000个tokenA与1000个tokenB，则Px=0.5，Py=2。当用2000个tokenA进行交换时，发现得到代币tokenB=2000\*0.5=1000个，这样池子内的tokenB全没了，将无法再进行交易。所以选择采用恒定乘积做市商算法[7]。具体为：

当用少量代币换取另一些代币公式：

变形：

通过这个公式便得交易获取的代币数量。



**图3-3 恒定乘积做市商函数图**

# 4 系统实现

**4.1 启动区块链节点**

编辑创世纪节点文件genesis.json,可以看到链id=15，难度设为1，并且初始化四个账户：

"0x4Ab565B724B664f77951F71aF6faCA0B379c9996",

"0xbf4d9b9865c1575f52babfb9ca60426ce09457e1",

"0x1dc160ee6b01962f0464e93041d655d545dba93b",

"0x2eE070cf96c5093D6c7d9dC690A813C0479d7445"

并赋予初始分配以太币数量，以太链中是以wei为最基本单位1eth=10^18 wei。然后输入下面命令可以启动私有链。以下是对参数的解释。

--datadir：区块链数据储存目录

--networkid：区块链网络，定义为15

--http：启动用http方式连接的远程过程调用

--http.api：运行远程过程调用的模块其中有重要的web3和eth模块。

--http.corsdomain：运行使用http进行远程调用的ip或者域名，使用‘\*’意味着任何站点都可连接使用。

--unlock：意味着这次启动需要解锁的账户

console：启动交互控制台。

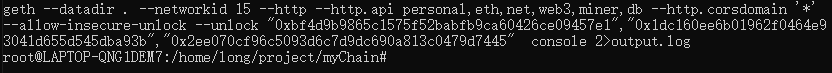
第一次启动时候使用下面命令通过genesis.json进行初始化：

geth --datadir "yourdatadir" init genesis.json

最后打开钱包选择localhost：8545可成功连接区块链



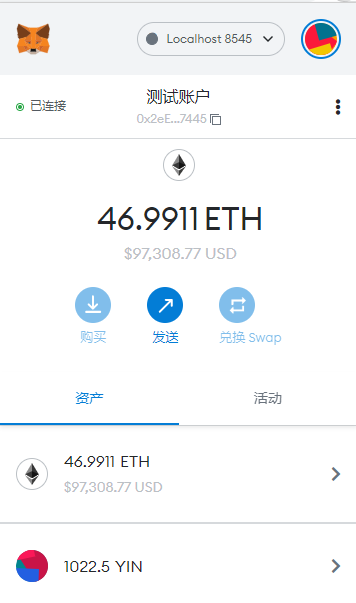
**图4-1 创世纪文件配置**



**图4-2 启动节点**

****

**图4-3 成功启动节点**

****

**图4-4 MetaMask钱包连接区块链**

**4.2 ERC20代币设计：**

利用solidity设计数字货币需要遵循一些标准，由Fabian Vogelsteller于2015年11月提出，是一种在智能合约中实现代币API的代币标准。

ERC-20代币需要实现以下函数和两个事件（**图4-5**）（**图4-6)**

接下来我将对这些函数的功能以及我的设计进行讲解：

1. 安全数学（图4-2-3）：设计了加减乘除四个函数，由于以太坊事实上并不存在浮点数，而是采用uint256无符号整型做运算（为什么这样采用在于浮点数有精度损失），由于无符号数会有溢出可能性，所以需要对运行结果进行检查，如果结果与事实不符，直接抛出异常，EVM虚拟机就会将这次结果进行回滚，而不至于影响资金安全。
2. 可读函数（图4-2-4）：

对其中name（名称），symbol（符号），decimals（小数精度），totalSupply（总供应量），owner（铸币人）设置为public，view关键字意味着是公开且只读的，不会修改内容。且在编译过程中变量会自动生成满足erc20规范的只读函数，而钱包或者网站读取这些数据是不需要消耗gas的，也就是不需要进行转账操作的。对balanceOf，freezeOf，allowance分别采用mapping (address => uint256)，mapping (address => uint256)，mapping (address => mapping (address => uint256))。mapping是一种映射存储结构，类似于key-value存储。其中balanceOf就是存储账户余额的映射，而freezeOf是用户冻结代币的存储结构，allowance映射里面存储了一个映射，这是用来存储每个地址授权另一个地址允许转账此代币余额数量，这在去中心交易所和授权投票中非常重要。

1. 构造函数：

构造函数负责在部署合约的时候进行初始化，一旦合约部署，数据都将不可修改，只能在网络上独立的运行。

1. transfer函数：

这个主要是对子货币进行转账操作，前面说过，balanceOf数据结构就是存储的用户地址拥有的子货币数量，所以实际上只是修改了balanceOf数据结构里面的值。前面的if判断是安全检查，保证转账正确，最后将事件发射出去，可以被区块链监测到。

1. transferFrom函数：

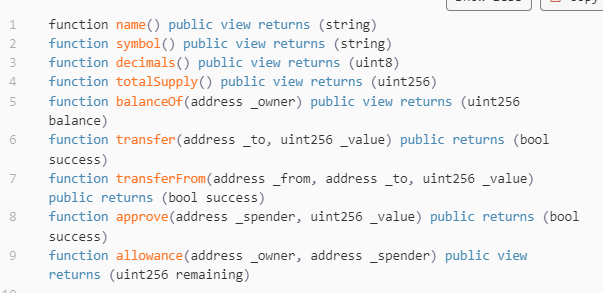
这个函数与transfer函数作用相同，只不过transfer是用户自己将自己的币转账给其他人，transferFrom函数当用户授权给了他人后，被授权人可以将用户的币转给其他人，前面if是是安全检查。

1. burn，freeze，unfreeze函数：

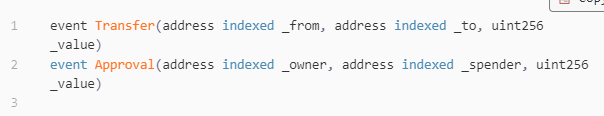
还增加了燃烧，冻结，和解冻方法，与上面都是对合约内数据结构数据进行修改。

1. withdrawEther和回退函数：

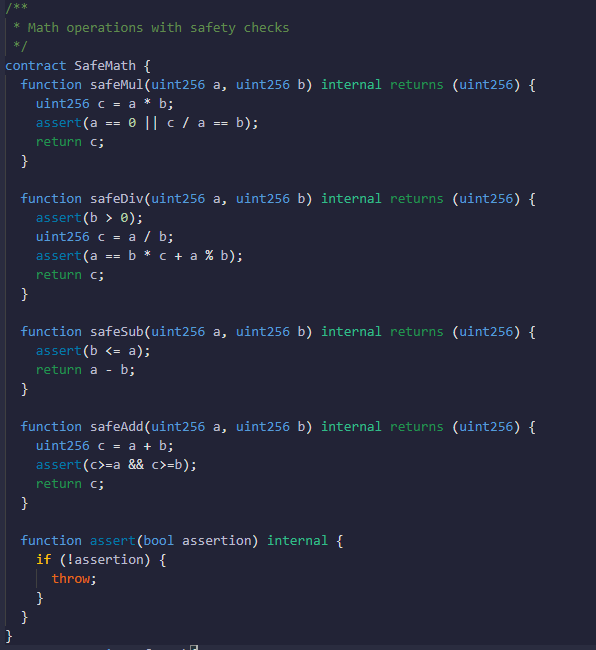
前面说过，普通地址和合约地址对虚拟机来说是一样的存在，只是合约地址多了一个data数据，所以合约也能接收以太。用一个匿名函数做回退函数并声明关键字payable，这样当其他地址向合约地址转以太的时候就会找不到要执行的函数，然后默认执行回退函数，回退函数就可以接收以太。withdrawEther就是将合约地址里面的以太转移出来。



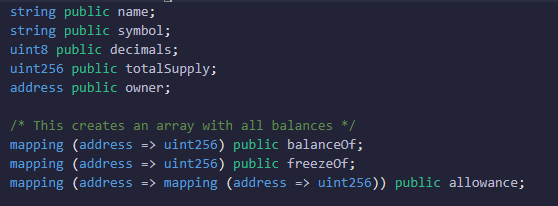
**图4-5 ERC-20函数接口标准**



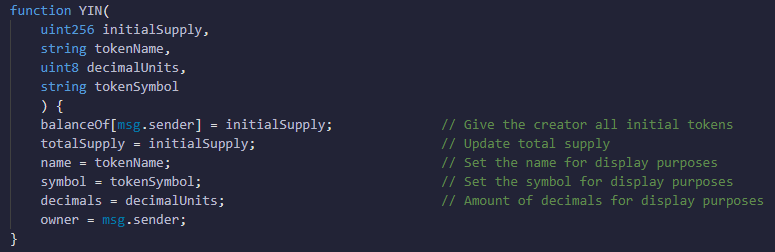
**图4-6 ERC-20事件接口标准**



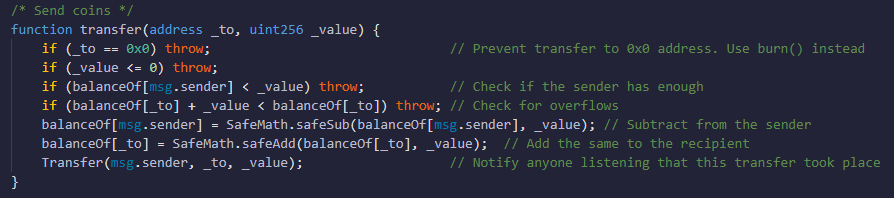
**图4-7 安全数学**



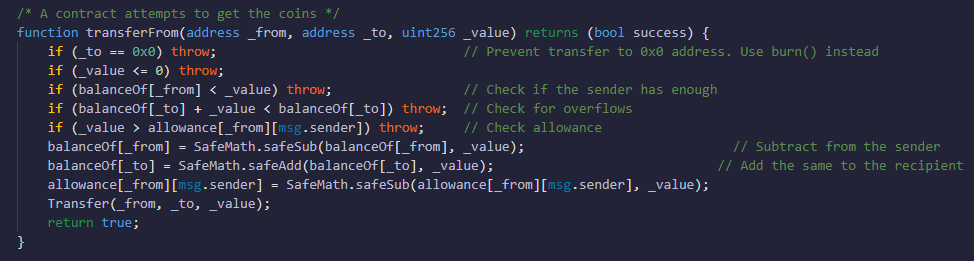
**图4-8 可读函数**



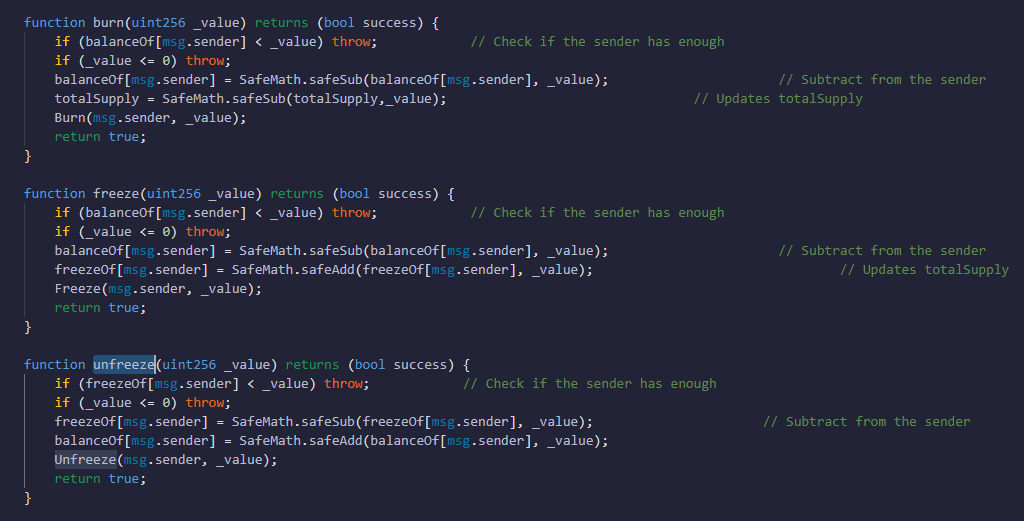
**图4-9 构造函数**



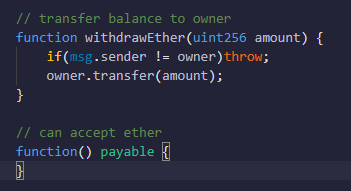
**图4-10 tansfer函数**



**图4-11 tansferFrom函数**



**图4-12 burn，freeze，unfreeze函数**



**图4-13 withdrawEther和回退函数**

**4.3 投票网站：投票合约设计**

1. 成员变量和构造函数：

owner：投票发起人，会在构造函数中对其进行初始化

voteCoin：是能参与投票需要的代币地址，也就是设计好的代币并成功部署到区块链上的地址,后面需要这个地址来获取投票人代币余额

minBalance：最少需要代币，由于以太坊最小单位是wei，需要18个0表示一个代币

Candidate：一个存储候选人的结构体

voters：一个映射存储结构，记录地址是否完成投票

candidates：一个映射存储结构，记录候选人数据。

1. 两个改变函数：

顾名思义，改变合约发起人和最小参与投票代币数量，在前端没有提供接口，可由运维人员用remix编译器进行修改。

1. addCandidate函数：

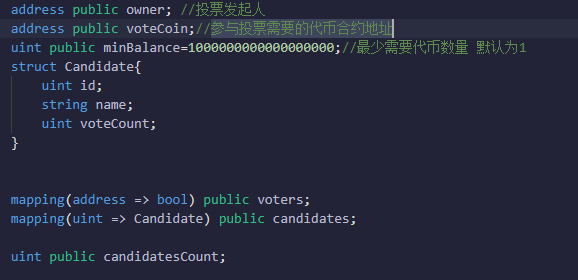
添加候选人，入口的require函数是对消息发送者进行检查，只有合约拥有人可以添加候选人。

1. vote函数：

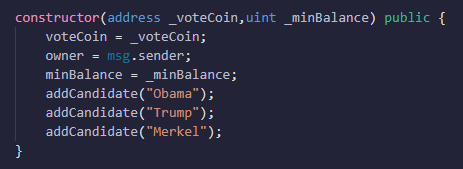
提供投票接口，每个人只要一票并且需要拥有代币数量大于minBalance，执行完成发布事件。

1. callBalance函数：

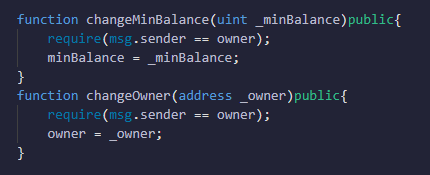
这是一个很重要的函数，由于投票合约需要调用代币合约来获取当前投票者的代币余额，所以需要在合约间进行交互。可以看到address类型有个call函数，其中参数分别是合约函数名签，函数名用abi.encodeWithSignature()函数对其编码，参数是需要调用合约的函数名：balanceOf(address)以及函数参数值，之后对其返回值进行解码，这样就能获得当前消息发送者地址拥有的代币数量。



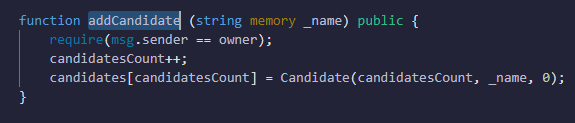
**图4-14 成员变量**



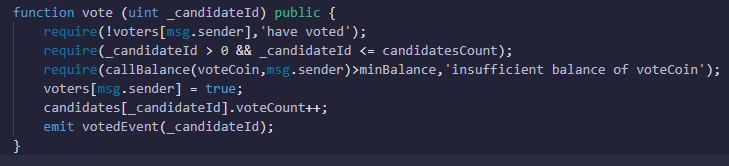
**图4-15 构造函数**



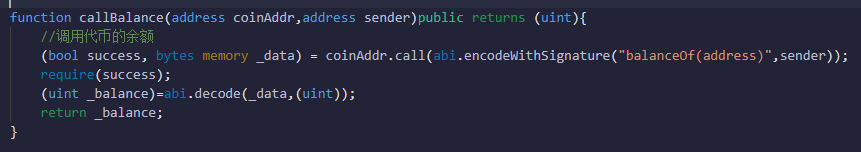
**图4-16 改变函数**



**图4-17 addCandidate函数**



**图4-18 vote函数**

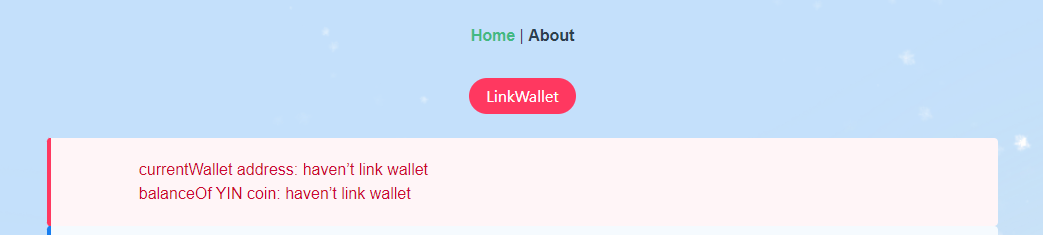


**图4-19 callBalance函数**

**4.4 投票网站：连接钱包**

当点击linkwallet，安装的钱包扩展会弹出授权(图4-2-2)当选择好账户成功授权后，前端样式会更改，并获取到账户地址以及投票合约。接下来我将对设计实现进行解释：

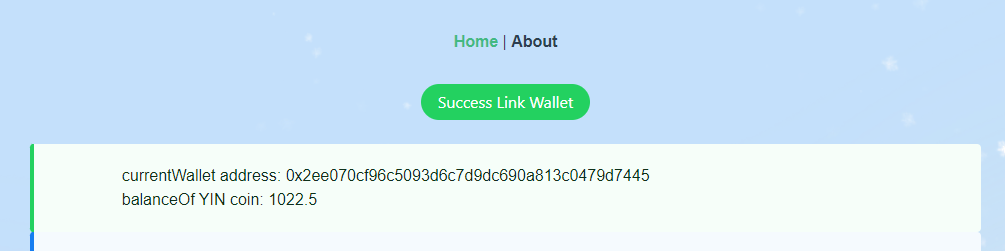
首先将状态定义一个默认值，在初始化时候前端会默认显示这些值，并根据值修改相应的样式。在linkWallet函数中，可以看到提供了一个web3Provider = ethereum，这个etherum对象就是钱包注入的，当使用这个对象调用enable方法钱包就会弹出授权并启动awaitlink函数，这个函数中是一个定时器，用来实时监控授权结果，当授权成功后会修改状态变量，页面就会更新。



**图4-20 钱包未连接**

****

**图4-21 钱包连接授权**

****

**图4-22 钱包成功连接**

****

**图4-23 连接钱包前端**

****

**图4-24 linkWallet函数**

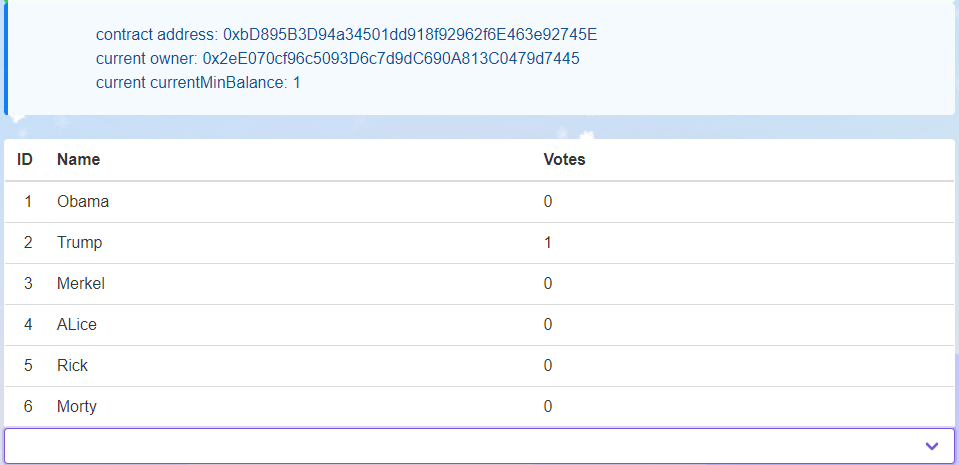
****

**图4-25 awitLink函数**

**4.5 投票网站：合约信息展示**

合约信息展示分成两部分，一部分是合约成员变量的展示：合约地址，拥有者，最小代币数量。对于列表部分使用使用状态变量tableDate进行绑定。

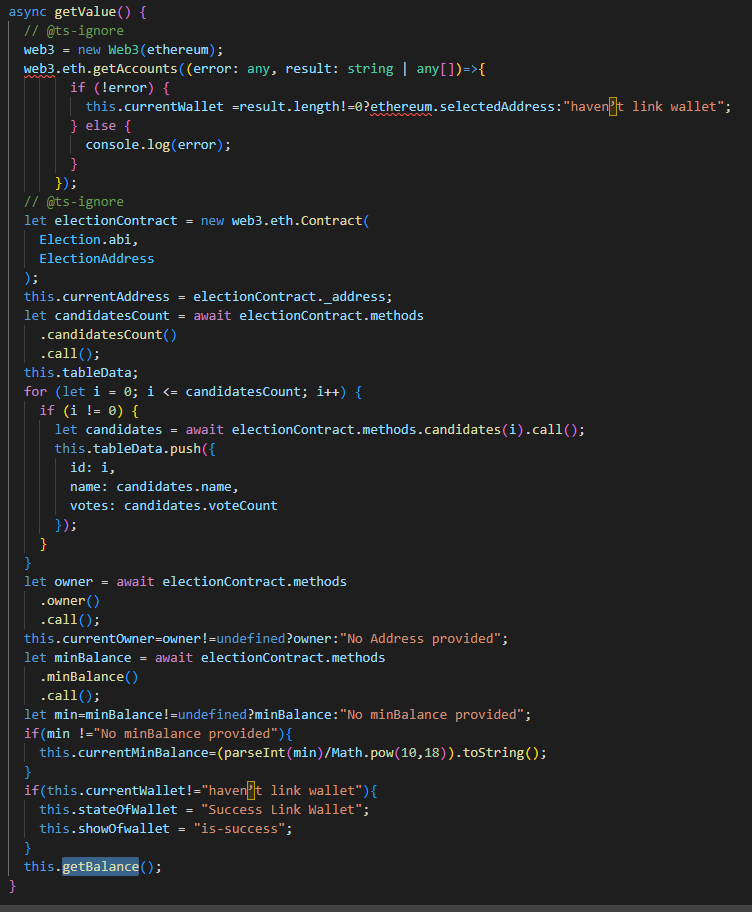
getValue函数是在vue渲染完成界面后运行的，主要是获取合约信息，但是与合约进行交互，前端并不知道合约有哪些函数，可以建立一个json文件，用来存储对投票合约编译后的结果，这个编译文件可以通过soljs完成，soljs可以在npm中进行安装。其中abi（全称Application Binary Interface）是函数的接口说明，这样可以通过new web3.eth.Contract(Election.abi,ElectionAddress);获取合约实例，这样就可以轻松实现和合约的交互，同理对代币的交互使用getBalance函数内部逻辑也是一样的。当然调用合约获取信息是不需要消耗gas的。



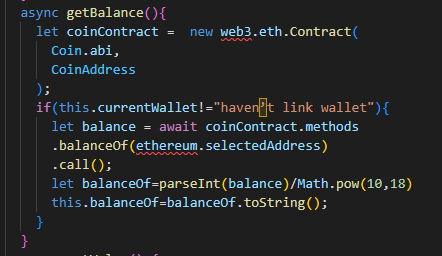
**图4-26 合约信息展示**

****

**图4-27 合约信息展示前端代码**

****

**图4-28 getValue函数**

****

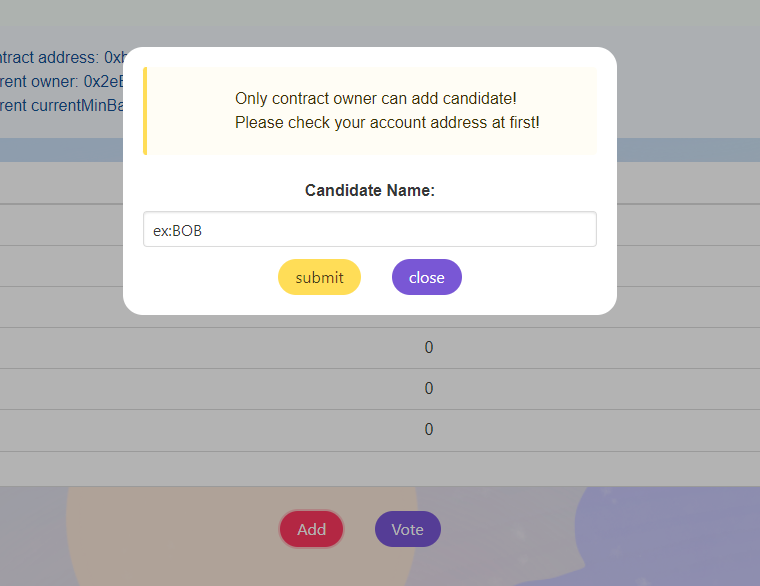
**图4-29 getBalance函数**

****

**图4-30 投票json文件abi（部分）**

**4.6 投票网站：添加候选人和投票**

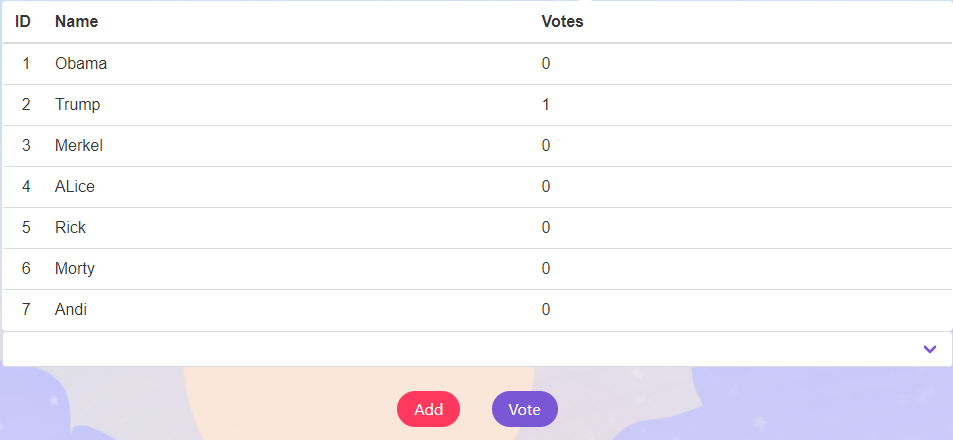
当点击Add按钮会弹出添加候选人按钮，当输入候选人名字点击submit后，钱包会弹出，当发送交易完成并成功打包进区块后，前端会更新候选人列表，投票过程大同小异，不再赘述。



**图4-31 添加候选人页面**

****

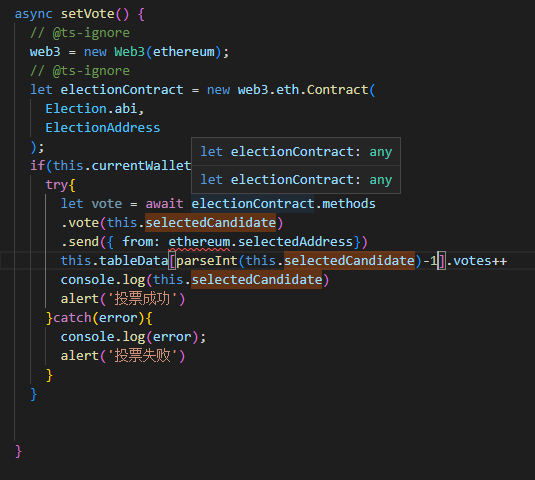
**图4-32包确认发送交易界面**

****

**图4-33成功添加候选人Andi**

****

**图4-34 addCandidate函数**

****

**图4-35 setVote**

**4.7 交易所：合约设计**

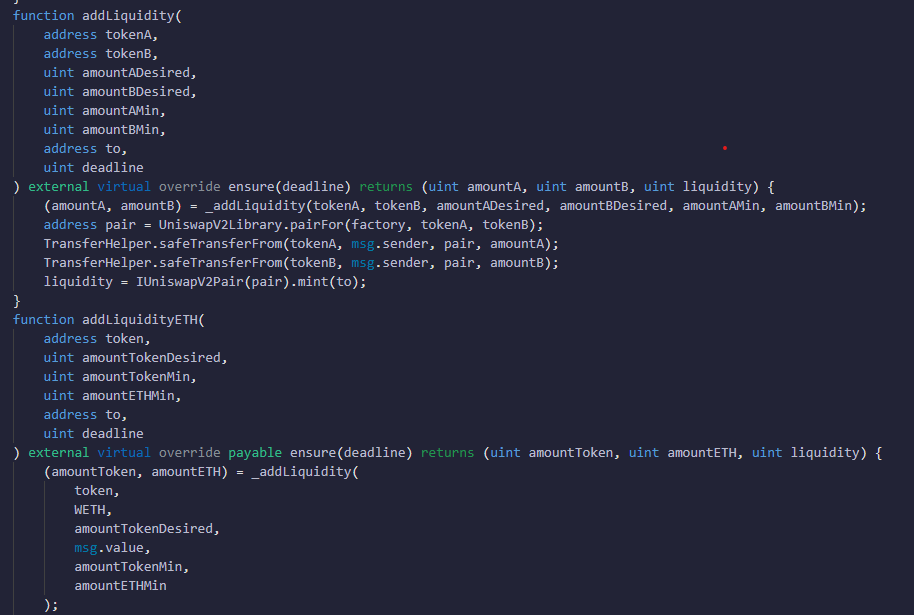
交易所合约分为四大块，分别是route（路由合约），factory（工厂），Multicall（多重请求），WETH（包装以太坊）四部分。

route合约：主要负责与前端进行交互，前端需要的增加流动性和移除流动性以及代币交易，创建交易对需要调用工厂合约

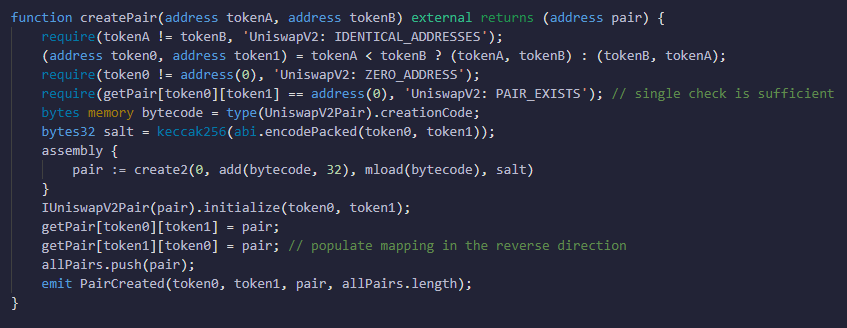
factory合约：主要是生产交易对Pair。当第一次添加两种代币时候，需要创建一个交易对合约（在工厂合约同一文件内），这个合约用来接收两种代币，计算流动性，然后赋予给流动性提供者。当移除流动性时会根据流动性返还两种代币余额给流动性提供者。

Multicall 合约：主要是请求当前区块信息，由于会特别频繁，可以通过这个合约打包成一次请求。

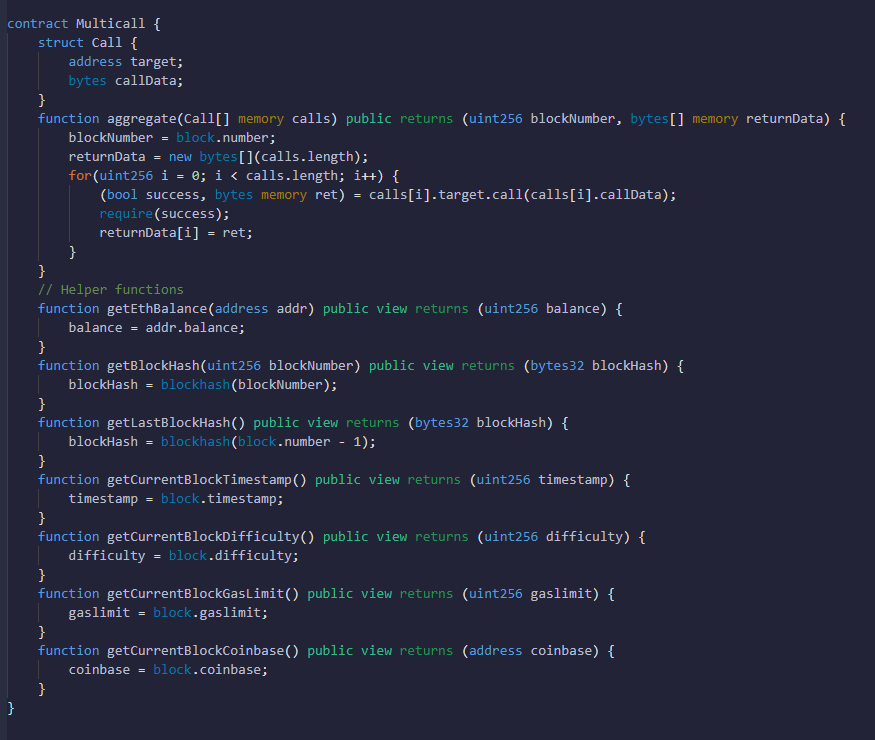
WETH：当需要以太与子货币交换时，通常是一个地址接收以太同时发送子货币，这就需要这个地址可信，可以设计一个合约当成它是以太，这就是包装以太，当向这个合约转一定数量以太时，就会铸造等量的包装以太，包装以太和以太是等价值的。下面是部分代码截图

****

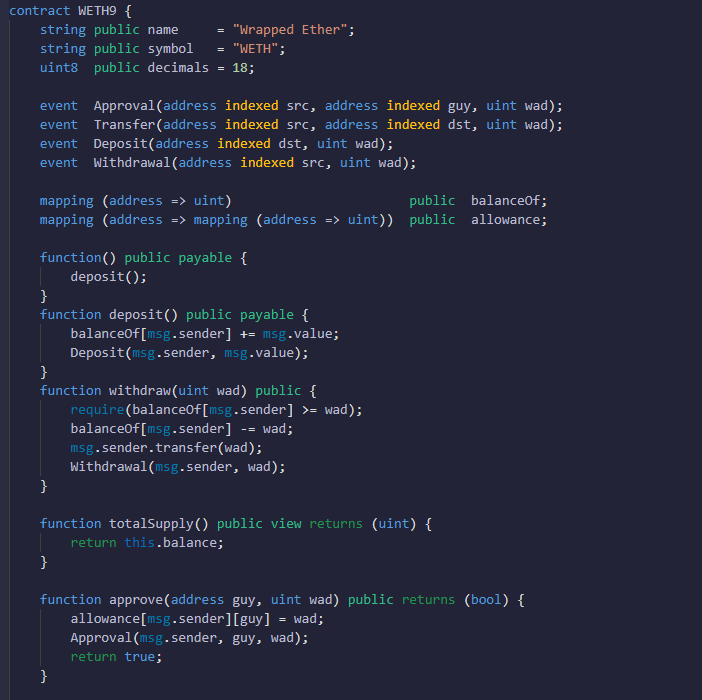
**图4-36 route合约（部分）**

****

**图4-37 factory合约（部分）**

****

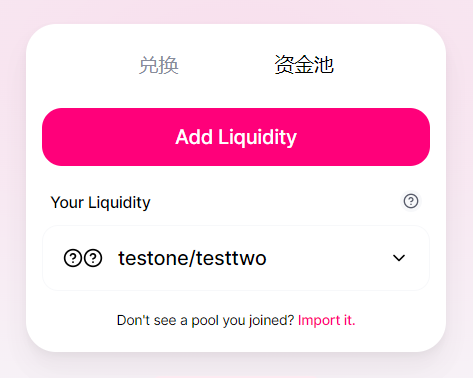
**图4-38 MultiCall合约**

****

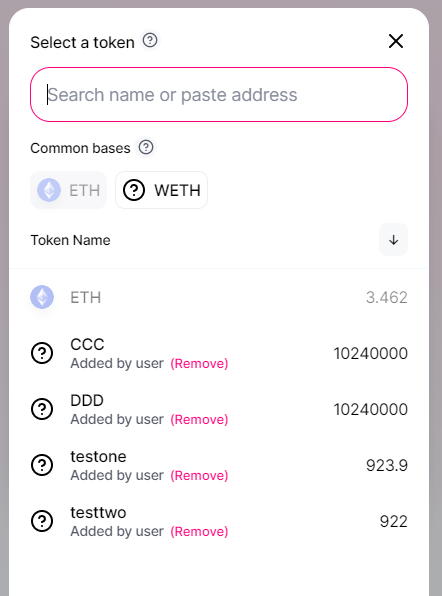
**图4-39 WETH合约（部分）**

**4.8 交易所：添加流动性与移除流动性**

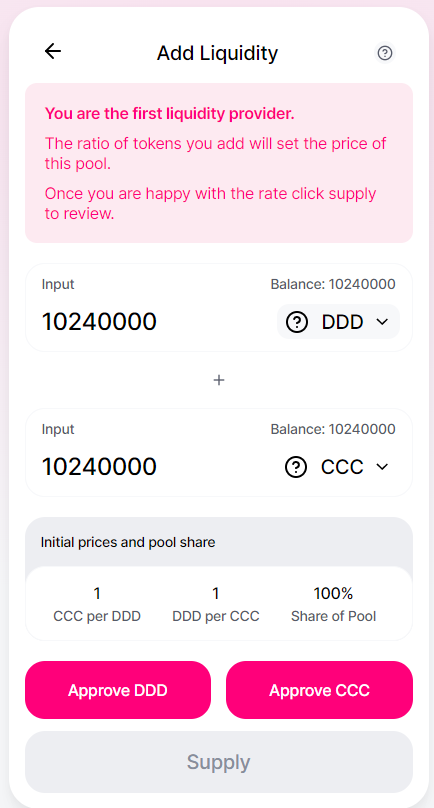
假设以及发布了两个ERC-20代币，那么怎样才能让他们相互交易呢？首先点击资金池，然后选择Add Liquidity。在搜索框中输入两个代币地址，选择拥有的代币数量，然后点击approve，这是一个授权操作，允许交易所调用你的代币余额。然后点击supply，这是供应操作，代币会转给交易所的Pari合约，然后获得Pair合约的流动性。

****

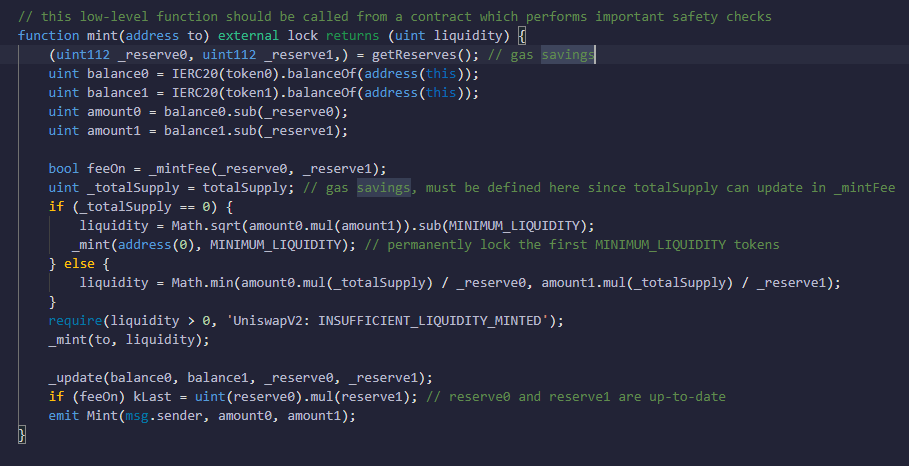
**图4-40 资金池**

****

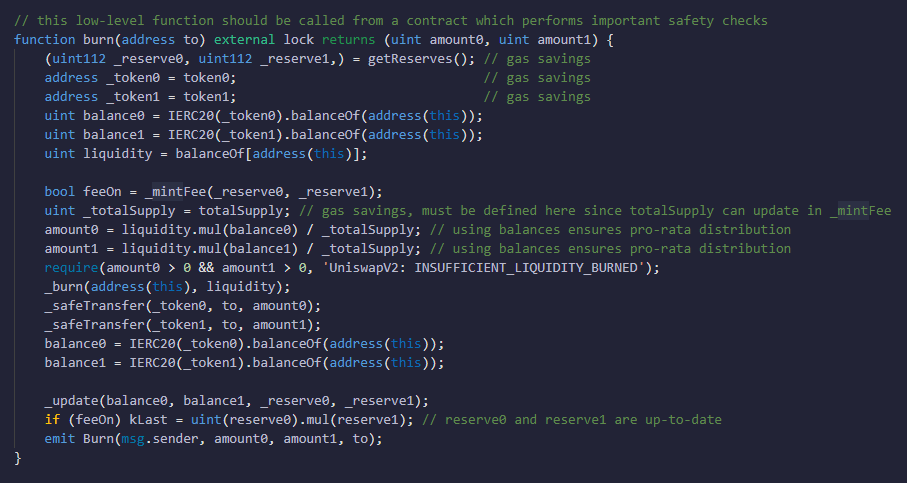
**图4-41 搜索代币**

****

**图4-42 授权和供应**

****

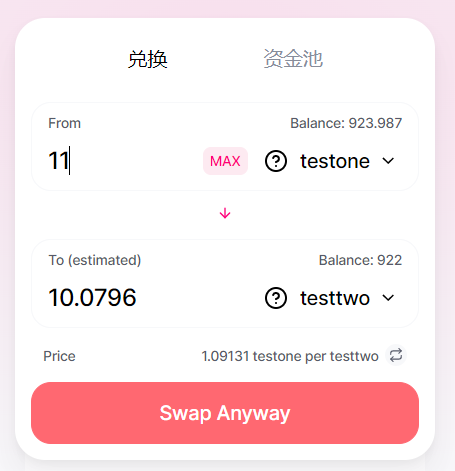
**图4-43 铸造流动性代币核心函数**

****

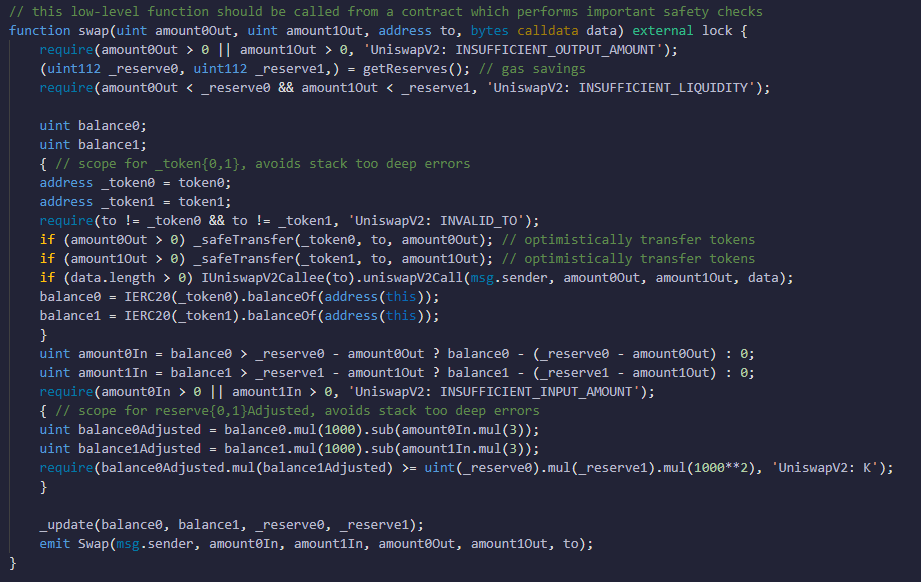
**图4-44 移除流动性核心函数**

**4.9 交易所：交易**

当以及添加过流动性池子后，池子同时拥有储备的两种代币，这样能参与交易，交易算法采用恒定乘积做市商。

****

**图4-45 可以交易**

****

**图4-46 swap算法实现**

# 5系统测试与维护

**5.1** **系统测试目的**

为了检测区块链网络稳定以及保证合约安全和网站功能完整没有缺陷，需要对整个系统进行测试。首先要保证区块链的稳定，因为所有合约都是运行在区块链上，如果区块链数据丢失，将造成非常严重的后果。其次要保证智能合约没有漏洞（历史上以太坊就因为智能合约漏洞导致分叉），所以需要保证合约功能完整并且注意可见性，对各种情况处理回退。再者需要保证Dapp网站所有功能均已实现。

**5.2** **测试的重要性**

测试有着等同开发一样重要的地位，良好的系统从来都是数次开发-测试-开发，在一次次版本迭代以及漏洞修复之后趋于成熟，而良好完整的测试，就能在开发过程中尽可能的减少漏洞。

**5.3 测试内容**

对于私有区块链，主要系统压力测试，需要保证区块网络有一定的算力，能抗住频繁的交易。

对于智能合约采用黑盒测试，给予测试测试用例条件和结果，判断运行结果是否一致。这一点可以采用truffle框架进行。truffle框架可以对编写的测试脚本进行用例测试并给出结果。对于其他合约例如可见性，以及数值运算需要更多的测试。

对于Dapp网站采用黑盒测试。对投票网站测试：连接钱包，合约信息，投票，添加候选人等功能进行测试；对交易所网站测试：获取代币数量，添加流动性，移除流动性，交易等功能进行测试

## 5.4 测试结果

**图5.1 投票网站功能检测结果表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | **功能检测结果表** | | |
| 功能模块 | 测试内容 | 测试结果 |
| 1 | 连接钱包 | 使用十个区块链账户连接 | 该功能正常 |
| 2 | 合约信息展示 | 部署三个不同的投票合约 | 该功能正常 |
| 3 | 添加候选人 | 添加十个不同姓名符号的候选人 | 该功能正常 |
| 4 | 投票 | 对十个账户进行不同候选人 | 该功能正常 |

**图5.2 交易所网站功能检测结果表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | **功能检测结果表** | | |
| 功能模块 | 测试内容 | 测试结果 |
| 1 | 连接钱包 | 使用十个区块链账户连接 | 该功能正常 |
| 2 | 获取货币 | 部署五个货币并获取 | 该功能正常 |
| 3 | 添加流动性 | 将五个货币组成六个交易对进行添加 | 该功能正常 |
| 4 | 移除流动性 | 六个交易对进行移除 | 该功能正常 |
| 5 | 交易 | 对六个交易对用一个货币交易另一个 | 该功能正常 |

**图5.3 合约漏洞测试**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | **功能检测结果表** | | |
| 合约 | 测试内容 | 测试结果 |
| 1 | ERC-20代币合约 | 各个函数输出结果是否一致，用非owner地址修改是否报错回滚 | 该功能正常 |
| 2 | 投票合约 | 投票合约各函数执行结果是否与预期一致，对错误是否进行报错回滚。 | 该功能正常 |
| 3 | Factory合约 | 工厂合约是否能正常生成交易对，各函数执行结果是否与预期一致 | 该功能正常 |
| 4 | Route合约 | 工厂合约是否能正常生与前端交互，各函数执行结果是否与预期一致 | 该功能正常 |
| 5 | WETH | 能否正常接收以太并铸造包装以太 | 该功能正常 |

**图5.4 私有链压力测试结果表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | **功能检测结果表** | | |
| 功能模块 | 测试内容 | 测试结果 |
| 1 | 私有链 | 利用脚本同时发送50个转账请求 | 该功能正常 |

# 结论

从一位神秘人中本聪在2008年开启区块链纪元以来，区块链蓬勃且快速的发展，在这个无人监管的网络竟然能平稳的运行上十年而不趋于崩溃。其带来的应用和发展也在进一步的联系实体社会，但是由于其常常游离于法律的边缘，各种虚假的ICO和不牢固的金融骗局使许多投资者血本无归而被人诟病，以及遭到抵制。但不因视其为毒蛇猛兽，它以非对称加密为基石的底层实现在不涉及金融方面的各个生活方面在未来将会有所应用，这种新的技术完全可以让人类的数字社会更加的蓬勃发展，其数据隐私性与公开性完全胜过当今世界中心化机构。用户数据不应当，也不应该只由世界上少数几个大公司垄断，并互相利用这些数据牟利。未来，人类反垄断意识将会在区块链上更清晰的体现。所以应该思考的是怎样将区块链技术在法律允许的范围内应用于正轨，并使得世界人民生活更加的美好。

**致谢**

在经过长达几个月的努力，终于完成了毕设和论文，在这个过程中，我非常感谢在大学期间对我淳淳教导的老师们。像最早将我加入实验室的潘志安老师，可以说他是我编程的启蒙老师，在实验室我学会了网站的整体开发；还非常感谢沈焱萍老师，在她的《网络安全》课程上，我第一次接触了密码学，这对我的毕业设计非常重要；还非常感谢袁静老师，在她那里第一次接触了人工智能，开拓了我的视野并且燃起了学习数学的火焰；还非常感谢计算机科学教研主任黄猛老师，他对我们教学学习的安排让我能体系的学习到计算机编程知识；还有很多很多······

然后我要感谢我的指导老师陈健老师，在毕设期间，总是淳淳教导督促我抓紧进度，且对我的问题不厌其烦的指导。在完成论文期间，提供了很多很多指导性意见。他还不忘提醒如果我在就业选择区块链要注意法律风险，可见不仅陈老师不仅是技术上的导师更是人生道德的指导老师，非常感谢。

最后我要感谢全世界无私的开发者，在他们的努力下，搭建起来一座一座的知识高塔，并无私的奉献给世界，因为他们巨人的肩膀我才得以完成我的毕业设计和论文。

**参考文献**

[1] Vitalik Buterin - Ethereum Whitepaper 2013 https://ethereum.org/en/whitepaper/

[2] Gavin Wood - Ethereum Yellow Paper 2016 <https://ethereum.github.io/yellowpaper/paper.pdf>

[3]刘花,包小敏.SHA-3候选算法Keccak的Matlab设计与实现[J].计算机科学,2012,39(S1):425-428.

[4]侯整风,李岚.椭圆曲线密码系统(ECC)整体算法设计及优化研究[J].电子学报,2004(11):1904-1906.

[5]张海蓉,黄玉兰,迟学芬.DSA数字签名的安全性分析[J].吉林大学学报(信息科学版),2009,27(06):563-568.

[6]张伟.ECDSA算法实现及其安全性分析[J].信息与电子工程,2003(02):7-12.

[7] Formal Specification of Constant Product (x × y = k) Market Maker Model and Implementation Yi Zhang, Xiaohong Chen, and Daejun Park Runtime Verification, Inc. October 24, 2018 https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf

[8]张亮,刘百祥,张如意,江斌鑫,刘一江.区块链技术综述[J].计算机工程,2019,45(05):1-12.DOI:10.19678/j.issn.1000-3428.0053554.

[9]黄洁华,高灵超,许玉壮,白晓敏,胡凯.众筹区块链上的智能合约设计[J].信息安全研究,2017,3(03):211-219.

[10]穆程刚,丁涛,董江彬,宁可儿,董晓博,贺元康,王永庆,陈天恩,刘健,Mohammad SHAHIDEHPOUR.基于私有区块链的去中心化点对点多能源交易系统研制[J].中国电机工程学报,2021,41(03):878-890.DOI:10.13334/j.0258-8013.pcsee.200392.

[11]姚前.密码学、比特币和区块链[J].清华金融评论,2018(12):99-104.DOI:10.19409/j.cnki.thf-review.2018.12.031.

[12]欧阳丽炜,王帅,袁勇,倪晓春,王飞跃.智能合约:架构及进展[J].自动化学报,2019,45(03):445-457.DOI:10.16383/j.aas.c180586.

[13]马春光,安婧,毕伟,袁琪.区块链中的智能合约[J].信息网络安全,2018(11):8-17.

[14]贺海武,延安,陈泽华.基于区块链的智能合约技术与应用综述[J].计算机研究与发展,2018,55(11):2452-2466.

[15]李赫,孙继飞,杨泳,汪松.基于区块链2.0的以太坊初探[J].中国金融电脑,2017(06):57-60.

[16]黄秋波,安庆文,苏厚勤.一种改进PBFT算法作为以太坊共识机制的研究与实现[J].计算机应用与软件,2017,34(10):288-293+297.

[17]胡键伟,尹丰.去中心化应用(DApp)技术原理和质量评测分析[J].中国新通信,2018,20(17):100.

[18]周奕军,李悦.基于区块链的时间胶囊Dapp设计[J].信息技术与信息化,2019(09):24-26.

[19]邱欣欣,马兆丰,徐明昆.以太坊智能合约安全漏洞分析及对策[J].信息安全与通信保密,2019(02):44-53.

[20]方言. 基于以太坊智能合约的形式化验证技术研究与实现[D].电子科技大学,2019.

[21]高一琛,赵斌,张召.面向以太坊的智能合约自动生成方法研究与实现[J].华东师范大学学报(自然科学版),2020(05):21-32.

[22]葛瑶. 基于以太坊的电子投票系统设计与实现[D].山东大学,2019.

[23]骆宾逸. 以太坊智能合约漏洞及安全函数设计方法研究[D].浙江师范大学,2019.DOI:10.27464/d.cnki.gzsfu.2019.000794.

[24]胡立.以太坊又爆漏洞 黑客大战一触即发[J].计算机与网络,2018,44(14):58.

[25]顾燕. 基于区块链的身份认证系统的设计与实现[D].北京邮电大学,2018.

[26]姚忠将,葛敬国.关于区块链原理及应用的综述[J].科研信息化技术与应用,2017,8(02):3-17.

[27]韩冬,张程正浩,孙伟卿,张巍,杨文威,肖敏.基于区块链技术的智能配售电交易平台架构设计[J].电力系统自动化,2019,43(07):89-96.

[28]夏沅.区块链智能合约技术应用[J].中国金融,2018(06):81-82.

[29]赵哲. 基于区块链的档案管理系统的研究与设计[D].中国科学技术大学,2018.

[30]安庆文. 基于区块链的去中心化交易关键技术研究及应用[D].东华大学,2017.