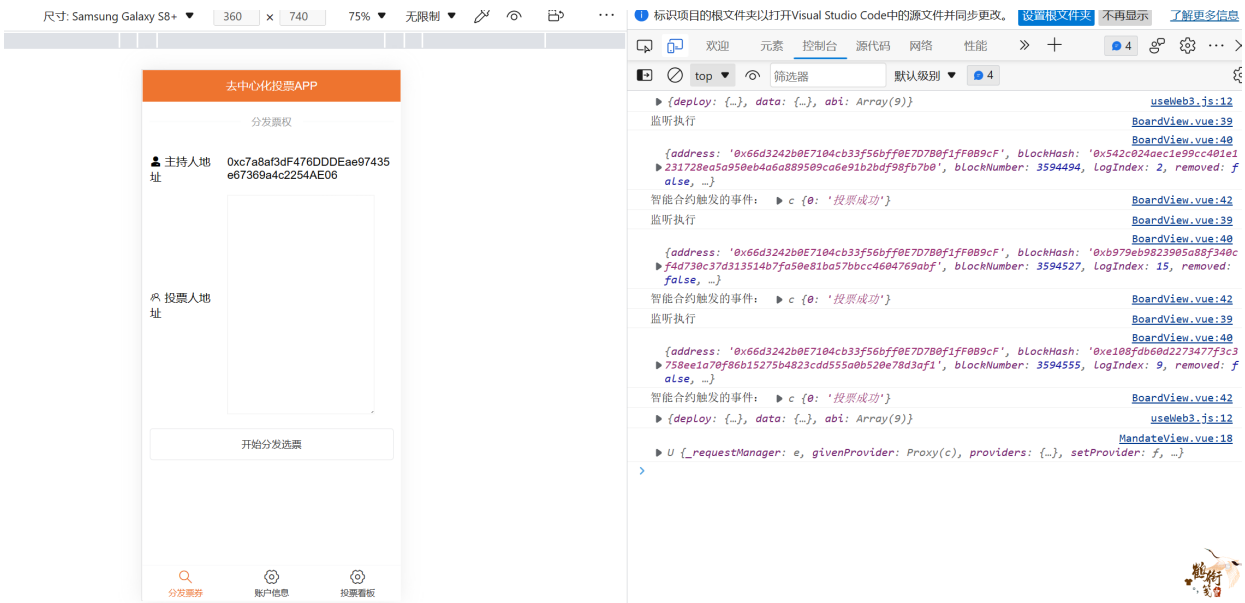
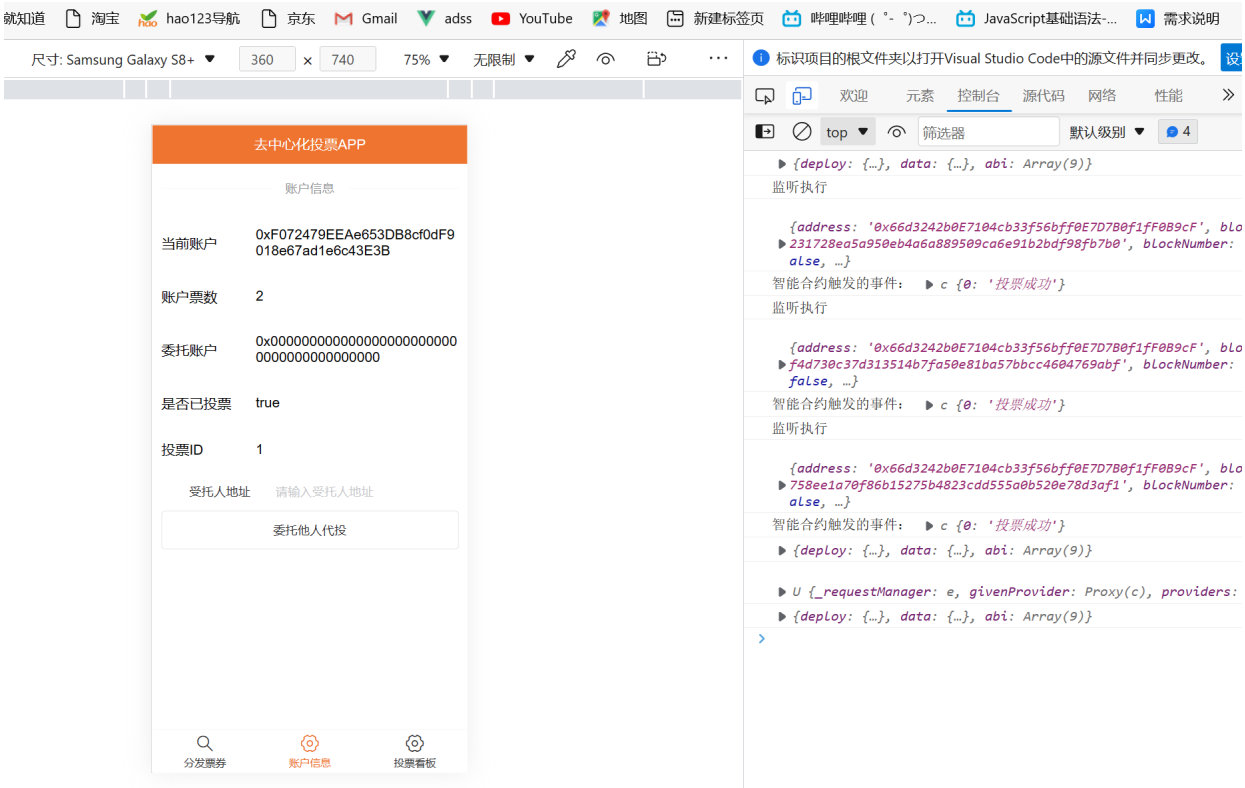
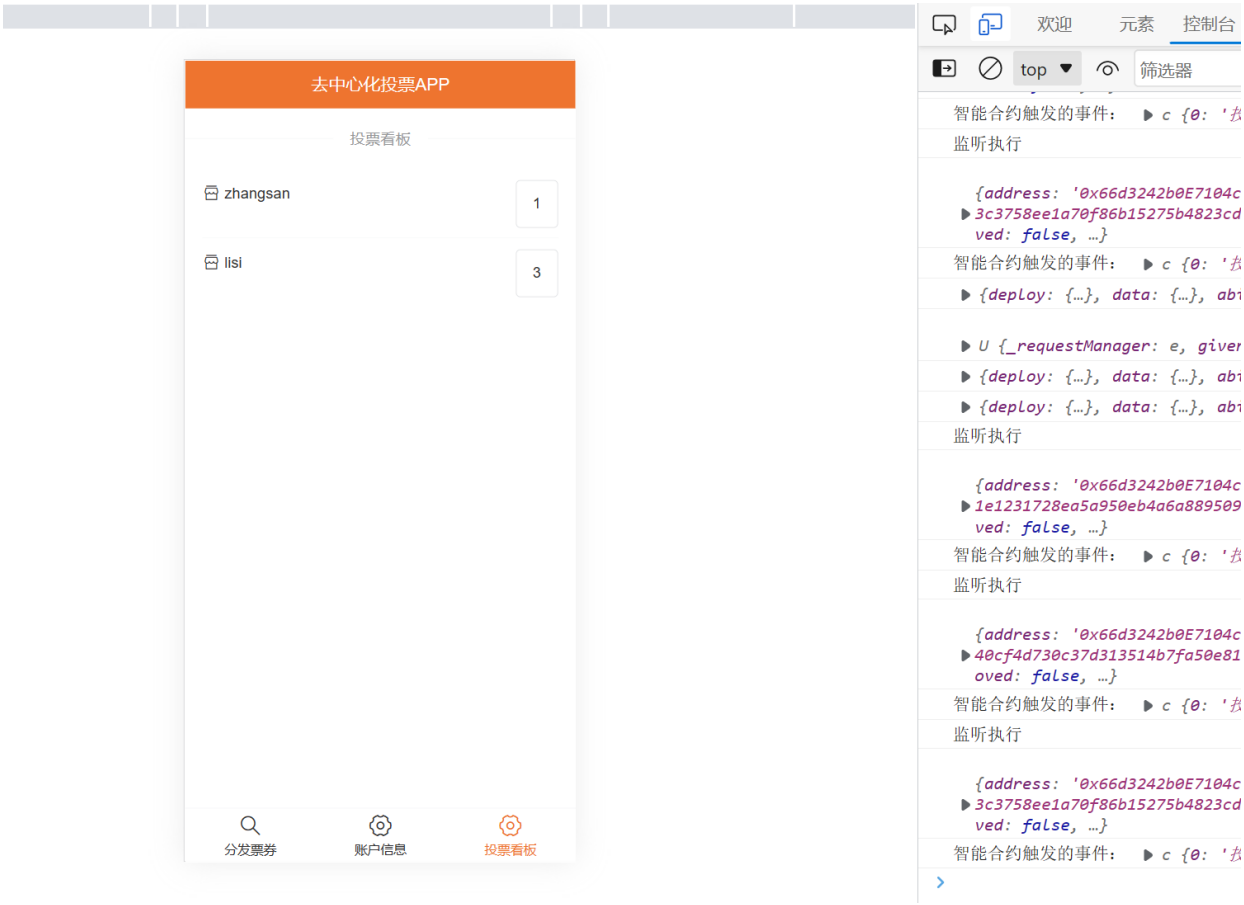
Dapp投票系统：

1、项目效果图：







2、环境

1. 智能合约编辑器 （版本：soljson-v0.8.18）： Remix - Ethereum IDE
2. solidity 版本：^0.8.9
3. 前端：
    1. vue3 脚手架 版本：@vue/cli 5.0.8
    2. node 版本 16

3、初始环境：

注意点 要创建小狐狸钱包需要访问外网：

3.1 配置vpn

以下是一款 vpn （不用注册登录 即可使用）

科学软件有很多种，但是大多需要付费 极少数提供免费试用，当然免费试用在就没法要求网速和稳定性了在可以随意使用科学软件达成科学上网的目的，我们仅介绍一款 方便操作，并且提供一定免费时长的

1. 下载“科学”软件

   https://arr007.network/download/

   根据自己的设备情况下载对应的版本

2. 下载完成之后进行安装，如果安装过程中提示病毒请关闭防火墙 忽略

3. 下载完毕后就可以使用，不要注册

4. 因为是免费版 所以稳定性和速度 懂的都懂

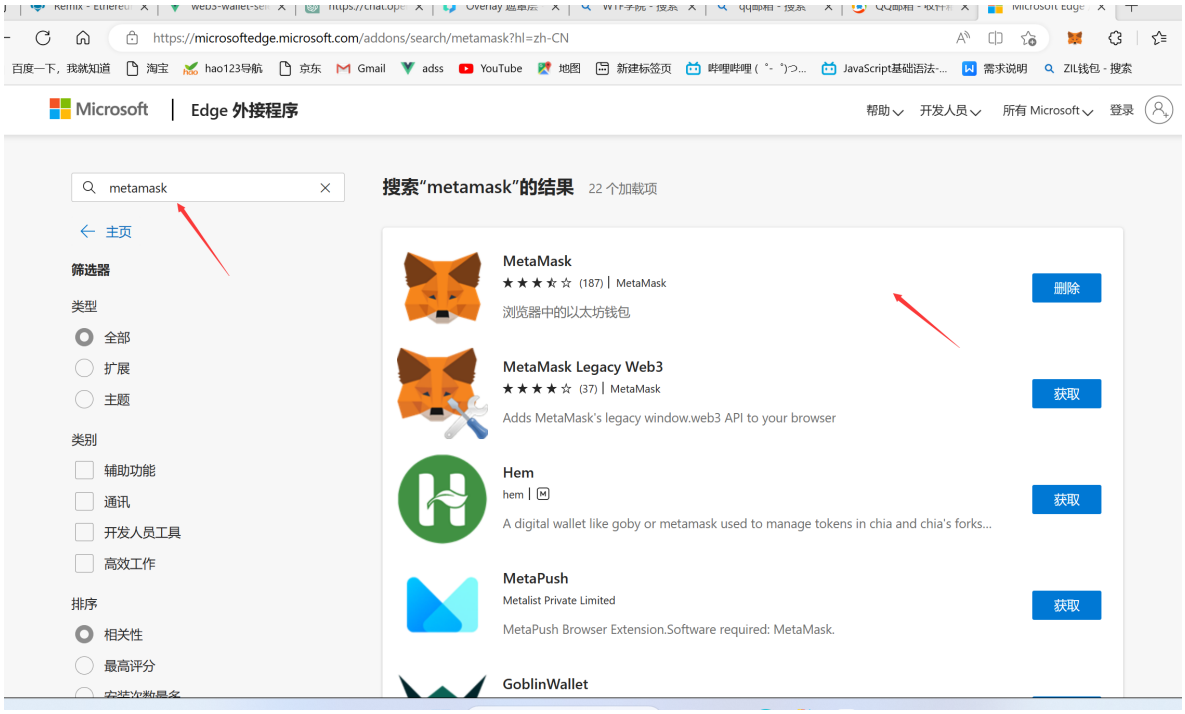
5. 通过某些网站进行验证

https://www.youtube.com/

3.2 创建小狐狸钱包：

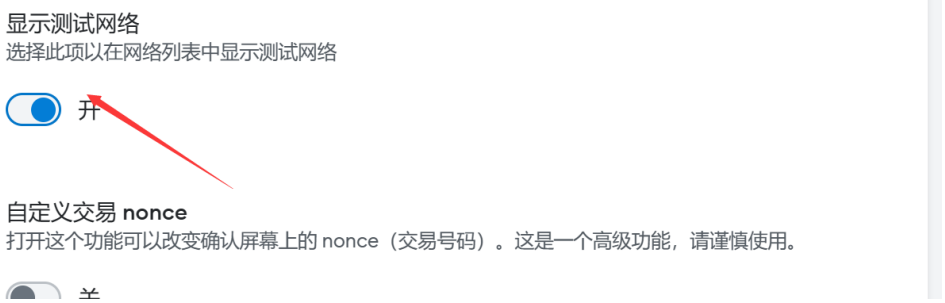
MetaMask是一个浏览器插件，可作为MetaMask Chrome扩展或Firefox附加组件使用。它的核心是它作为以太坊钱包：通过安装它，您将可以访问一个独特的以太坊钱包地址，您可以使用它开始发送和接收以太币或ERC20通证。

1. 直接在浏览器中的插件商店搜索即可下载

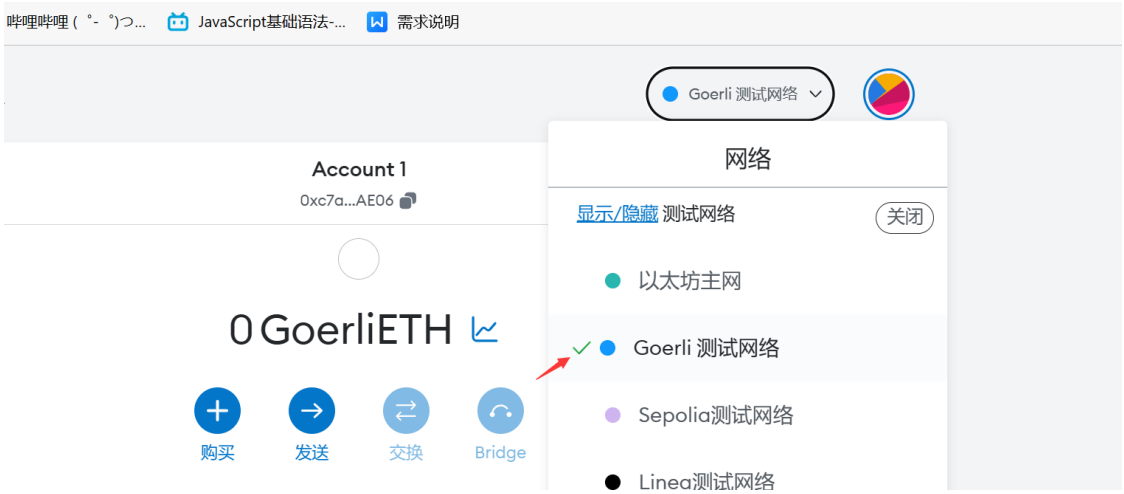


2. 之后点击进入创建一个账号：
    1. 重点： 一定要保存好自己的**密码** 和 **助记词**
3. 之后我们的项目就使用 sepolia 测试网进行
    1. 切换ceshi网络：

    2. 第二步：



3. 第三步：

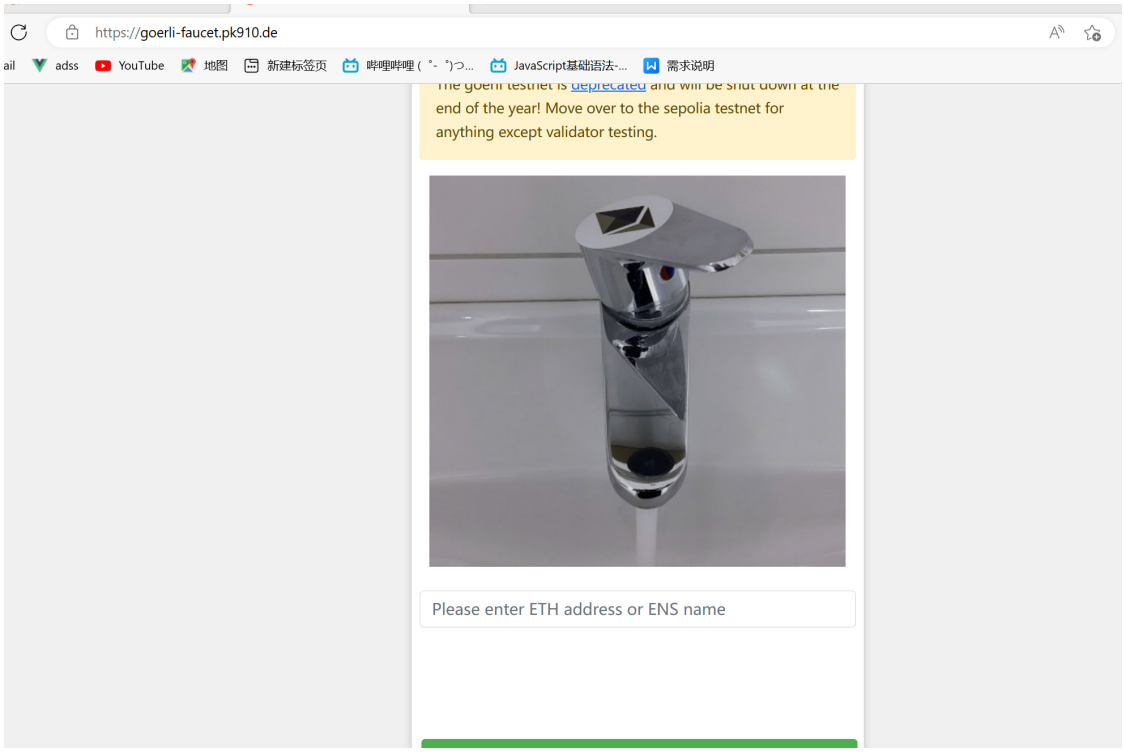


4. 由于我们的账户还没有一点的测试以太币，就需要进行挖矿了（挖的不是真的哦）

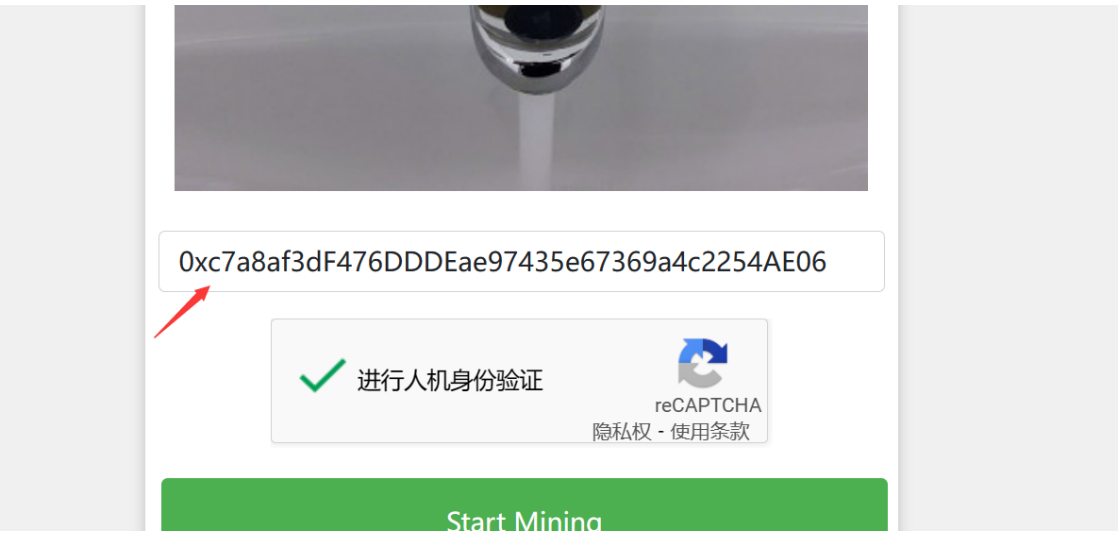
    1. https://www.tokenpocket.pro/ 以太坊 挖矿地址

       https://sepolia-faucet.pk910.de/ sepolia 挖矿地址

5. 开始挖矿：

    1. 

2. 粘贴账号点击开始挖矿



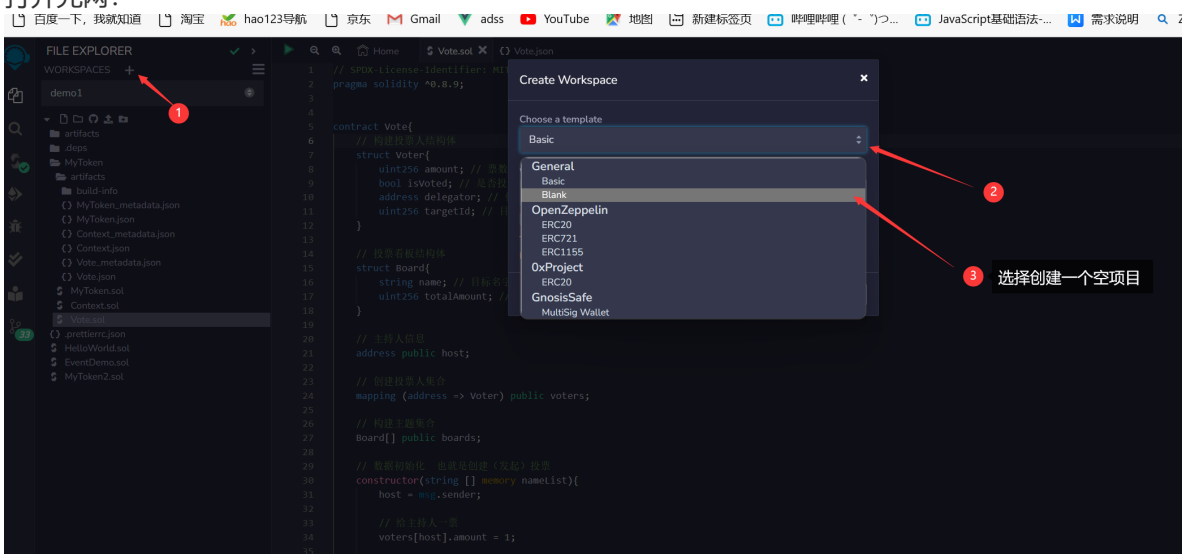
4、编写智能合约：

1. 本项目编写智能合约 都在remix上进行 应为他比较方便使用 从编写到部署　地址：[Remix - Ethereum IDE]

Remix 是以太坊智能合约编程语言Solidity IDE，其实基于浏览器的IDE，有一个很大的好处就是不用安装，打开即用。
官网 https://remix.ethereum.org/。

4.1 Remix基本功能

1. 打开光网：



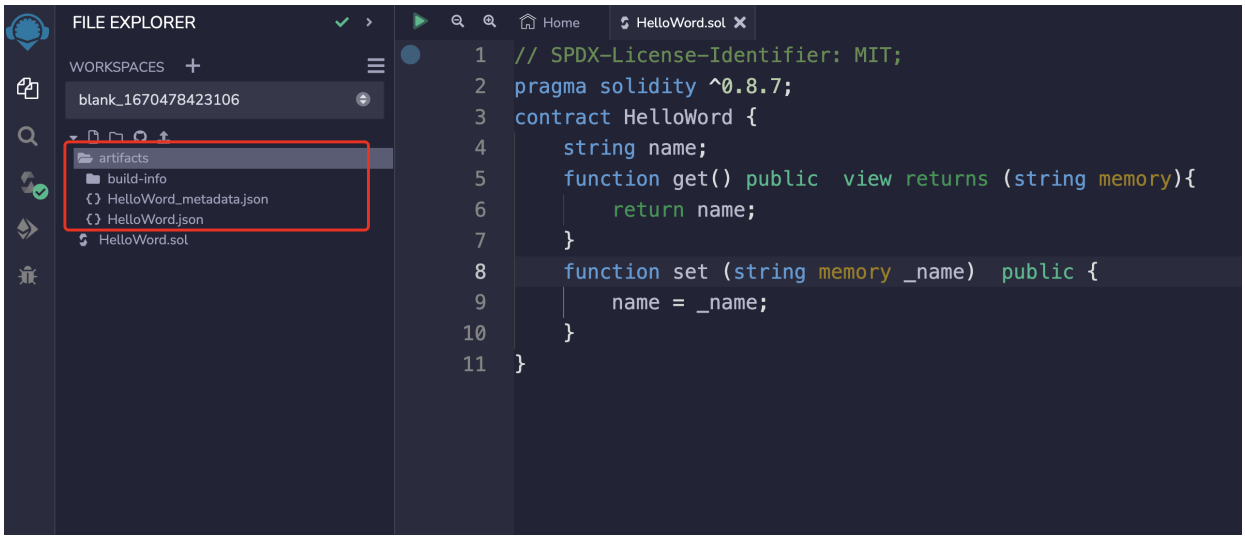
2. 在工作空间下创建一个智能合约文件,ex: `HelloWord.sol`
    - 智能合约文件以`.sol`结尾,
    - 文件名采用大驼峰命名法
    - 文件名和合约名保持一致

```
// SPDX-License-Identifier: MIT;
// 智能合约的许可协议
pragma solidity ^0.8.7;
// 智能合约的适用版本
contract HelloWord {
    string name;
    function get() public  view returns (string memory){
        return name;
    }
    function set (string memory _name)  public {
        name = _name;
    }
}
```

4.2 合约编译

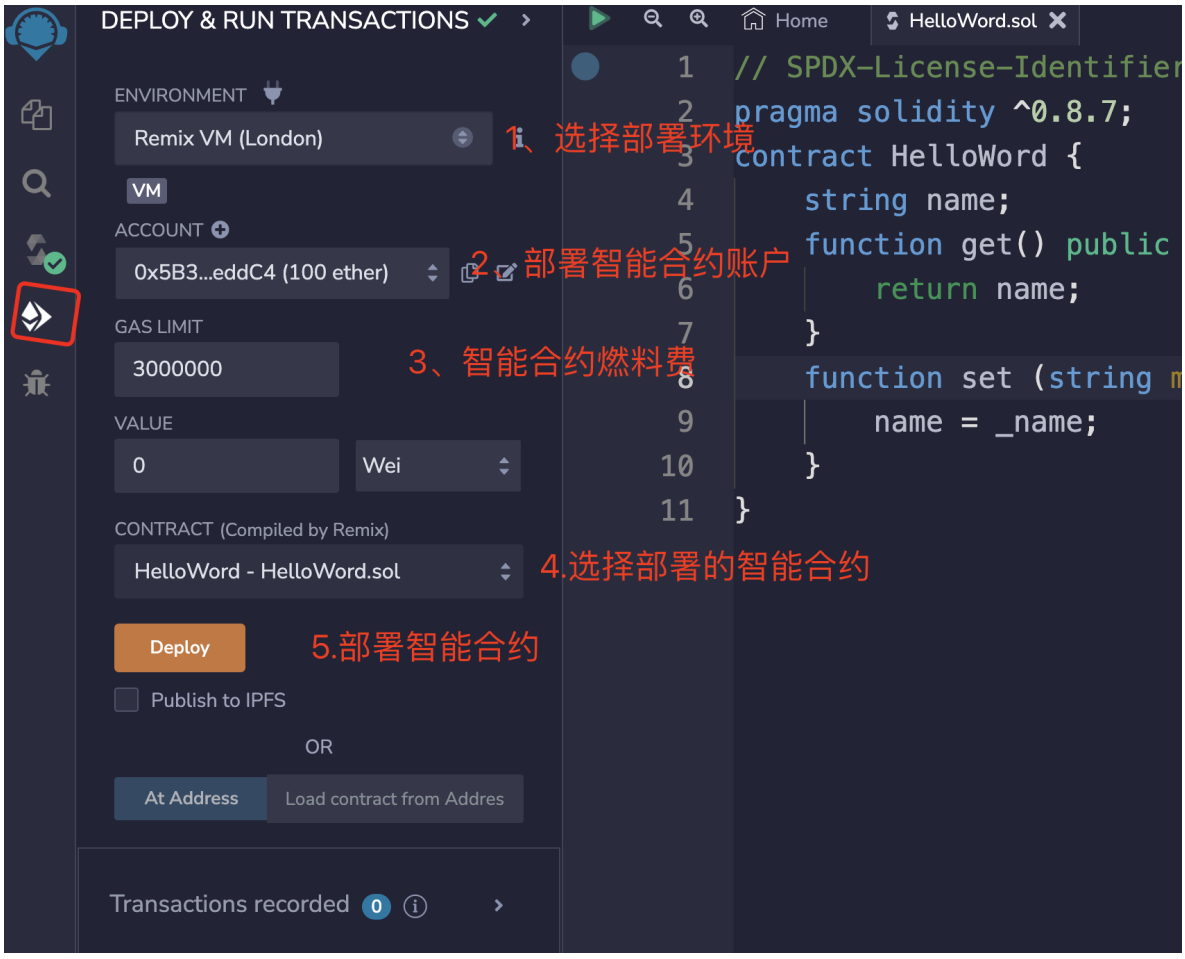
编译结果：

目录产生一个`artifacts`文件夹

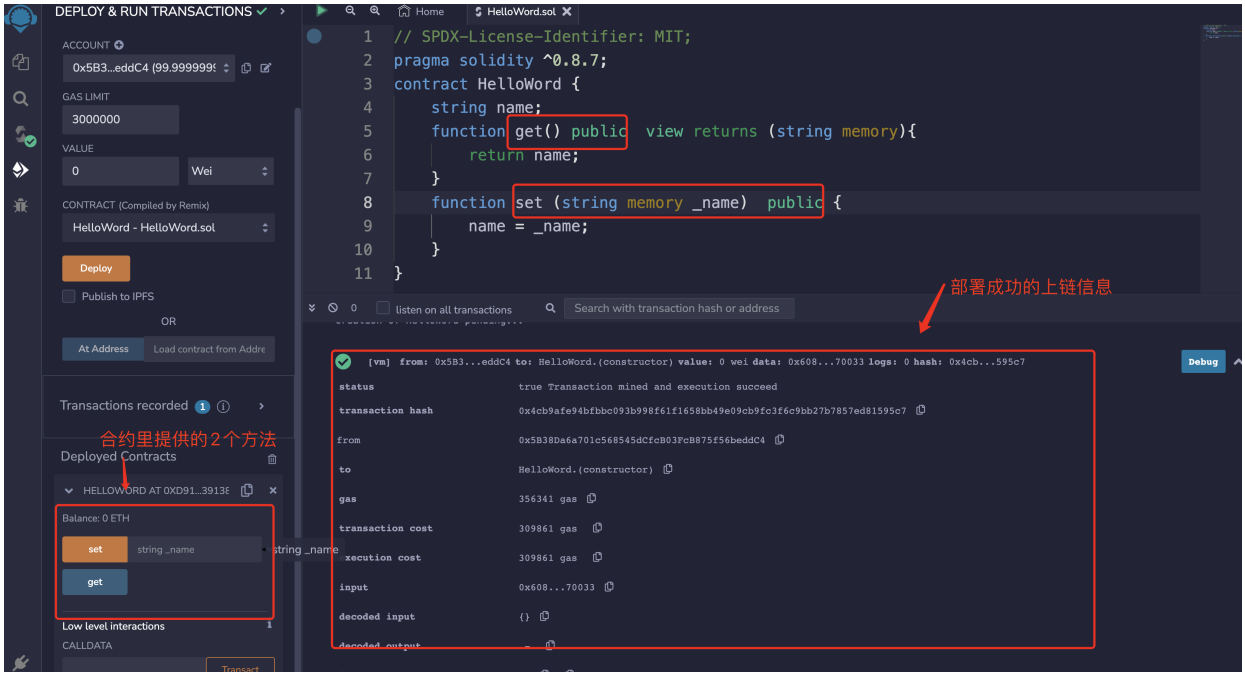


4.3 合约部署

1. 通过第四个菜单进入部署界面
2. 选择部署环境
3. 选择部署合约的账户地址
4. 设置gas限制
5. 选择要部署的合约
6. `deploy` 按钮进行部署

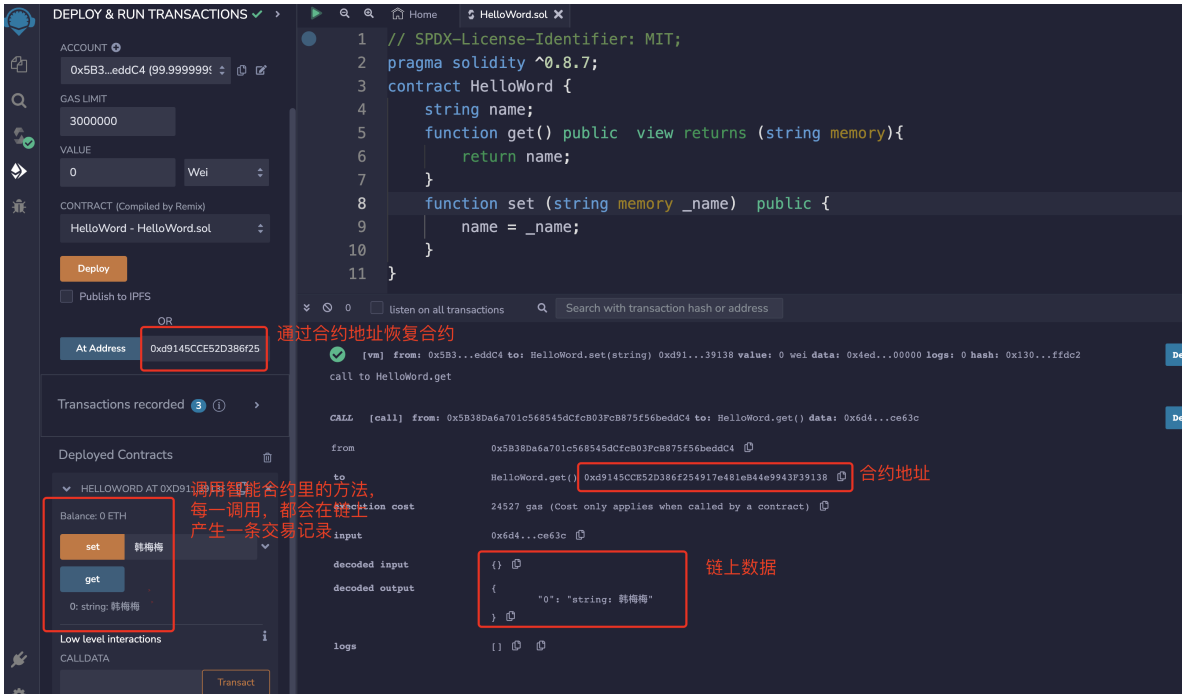


4.4 部署成功效果



4.5 合约调试

1. 通过函数的返回值查看变量



2. event Log

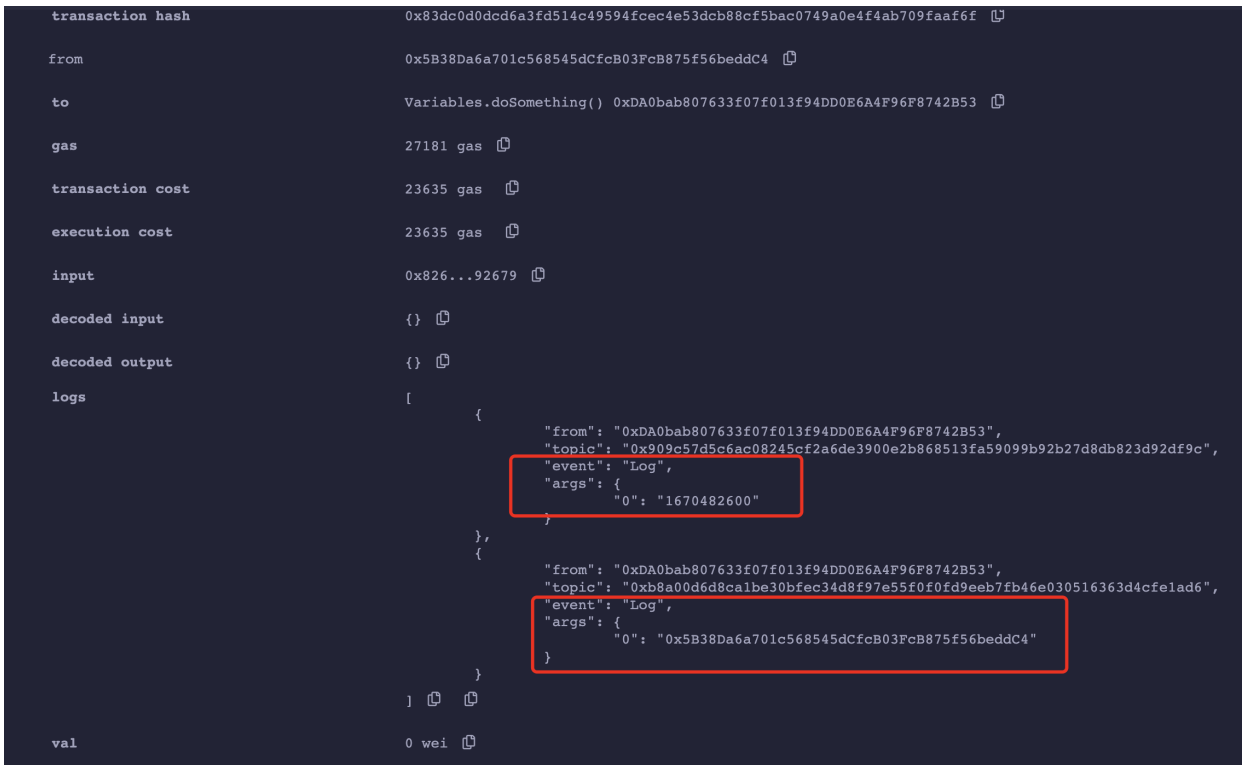
solidity默认没有consol.log 或者 print 类似的事件系统 但是我们可以通过，注册事件查看对应的log日志

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract Variables {
```

```
    event Log(address);
    event Log(uint);

    function doSomething() public {
        uint timestamp = block.timestamp; // Current block timestamp
        address sender = msg.sender; // address of the caller
        emit Log(timestamp);
        emit Log(sender);
    }
}
```



Dapp投票系统智能合约

好了以上就介绍了 Remix 的基本操作流程 下面是本项目所编写的智能合约：

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;


contract Vote{
    // 构建投票人结构体
    struct Voter{
        uint256 amount; // 票数
        bool isVoted; // 是否投过票了
        address delegator; // 代理人地址
        uint256 targetId; // 目标ID
```

```
    }

    // 投票看板结构体
    struct Board{
        string name; // 目标名字
        uint256 totalAmount; // 票数
    }

    // 主持人信息
    address public host;

    // 创建投票人集合
    mapping (address => Voter) public voters;

    // 构建主题集合
    Board[] public boards;

    // 数据初始化  也就是创建（发起）投票
    constructor(string [] memory nameList){
        host = msg.sender;

        // 给主持人一票
        voters[host].amount = 1;

        //初始化board 备选人的信息计入看板
        for(uint256 i = 0; i <nameList.length;i++){
            Board memory boardItem = Board(nameList[i],0);
            boards.push(boardItem);
        }
    }

    // 返回看板集合
    function getBoardInfo()public view returns(Board[] memory) {

        return boards;
    }

    // 给某写地址赋予投票
    function mandate(address[] calldata addressList) public {
        // 只有所有者 也就是主持人可以调用该方法
        require(msg.sender == host, "Only the owner has permissions"); // 进行判断 只有主持人可以分发票数
        for (uint256 i=0 ; i < addressList.length;i++){
            // 如果该地址已经投过票， 不做处理了
            if(!voters[addressList[i]].isVoted){
                voters[addressList[i]].amount = 1;
            }
        }

    }
```

```
    // 投票
    function vote(uint256 targetId) public {
       Voter storage sender = voters[msg.sender];
       require(sender.amount !=0, unicode"Has no right to vote 你已经没有票了"); // 如果等于0 就表示没有票 否则继续执行以下操作
       require(!sender.isVoted, "Already voted"); // 如果等于true 就表示投过票了
       sender.isVoted = true;
       sender.targetId = targetId;
       boards[targetId].totalAmount += sender.amount;
       // 触发事件
       emit voteSuccess(unicode"投票成功");
    }

    // 投票成功的事件
    event voteSuccess(string);

    // 投票委托 （将投票权委托给别人）
    function delegate(address to) public {
        // 获取委托人信息
        Voter storage sender = voters[msg.sender];

        // 进行判断 如果委托人投过票了就不能再委托别人进行投票了
        require(!sender.isVoted,"you already");

        // 不能委托给自己
        require(msg.sender != to,unicode"不能委托给自己");

        // 避免循环委托
        while (voters[to].delegator != address(0)){  // voters[to].delegator != address(0) 表示如果代理的地址不为空 就进入
            to = voters[to].delegator;  // 获取代理人的代理地址
            require(to == msg.sender, unicode"不能循环委托");  // 比对代理人代理地址是否和本人一样 一样就是循环
        }

        // 开始授权
        sender.isVoted = true;
        sender.delegator = to;

        // 代理人数据修改
        Voter storage delegator_ = voters[to];
        if(delegator_.isVoted){ //判断是否投过票了  如果投过票了 就直接追票
            boards[delegator_.targetId].totalAmount += sender.amount;
        } else{ //否则就直接加上一票
            delegator_.amount += sender.amount;
        }

    }


}
```

```
/*
[zhangsan,lisi]
[0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2,0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02
db]
*/
```

5、vue前端

5.1 环境搭建

5.1.1 项目创建:

版本信息:

Node 16.14

1. 安装vue-cli

```
npm install -g @vue/cli
```

2. 验证vue-cli 安装

```
$ vue -V
// 结果 出现vue版本号 安装成功
@vue/cli 5.0.8
```

3. 通过vue-cli创建项目

```
$ vue create <项目名称>
// 选择配置
Vue CLI v5.0.8
? Please pick a preset:
  Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
❯ Manually select features  `选择该项`
```

4. 选择自定义配置

```
? Check the features needed for your project: (Press <space> to select, <a> to
toggle all, <i> to invert selection, and <enter> to proceed)
 ◉ Babel `选择该项`
 ◯ TypeScript
 ◯ Progressive Web App (PWA) Support
 ◉ Router `选择该项`
 ◯ Vuex
❯◉ CSS Pre-processors `选择该项`
 ◯ Linter / Formatter
 ◯ Unit Testing
 ◯ E2E Testing
```

5. 选择版本

```
? Choose a version of Vue.js that you want to start the project with (Use arrow
keys)
❯ 3.x
  2.x
```

6. 选择路由设置

```
? Use history mode for router? (Requires proper server setup for index fallback
in production) (Y/n) y  `输入y或者n`
```

7. 选择预处理语言

```
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported
by default):
  Sass/SCSS (with dart-sass)
❯ Less
  Stylus
```

8. 选择配置文件存放目录

```
? Where do you prefer placing config for Babel, ESLint, etc.?
❯ In dedicated config files
  In package.json
```

9. 安装成功后 运行测试

```
 $ cd 项目名称
 $ npm run serve
```

5.1.2 第三方包安装

1. web3相关第三方包

```
npm install web3 bip39 ethereumjs-tx@1.3.7 ethereumjs-util ethereumjs-wallet
 node-polyfill-webpack-plugin
```

2. node-polyfill 兼容文件配置

    1. 下载polyfill 插件

    ```
    npm install node-polyfill-webpack-plugin -D
    ```

    2. 在vue.config.js 文件中配置插件

    ```
    const { defineConfig } = require('@vue/cli-service')
    // 引入插件
     const NodePolyfillWebpackPlugin = require("node-polyfill-webpack-plugin");
    module.exports = defineConfig({
      transpileDependencies: true,
     configureWebpack: {
     plugins: [
     new NodePolyfillWebpackPlugin()
        ],
      },
    })
    ```

3. 配置vant-ui ui组件库 https://vant-contrib.gitee.io/vant/#/zh-CN

    1. 安装

    ```
    $ npm i vant
    $ npm i unplugin-vue-components -D
    ```

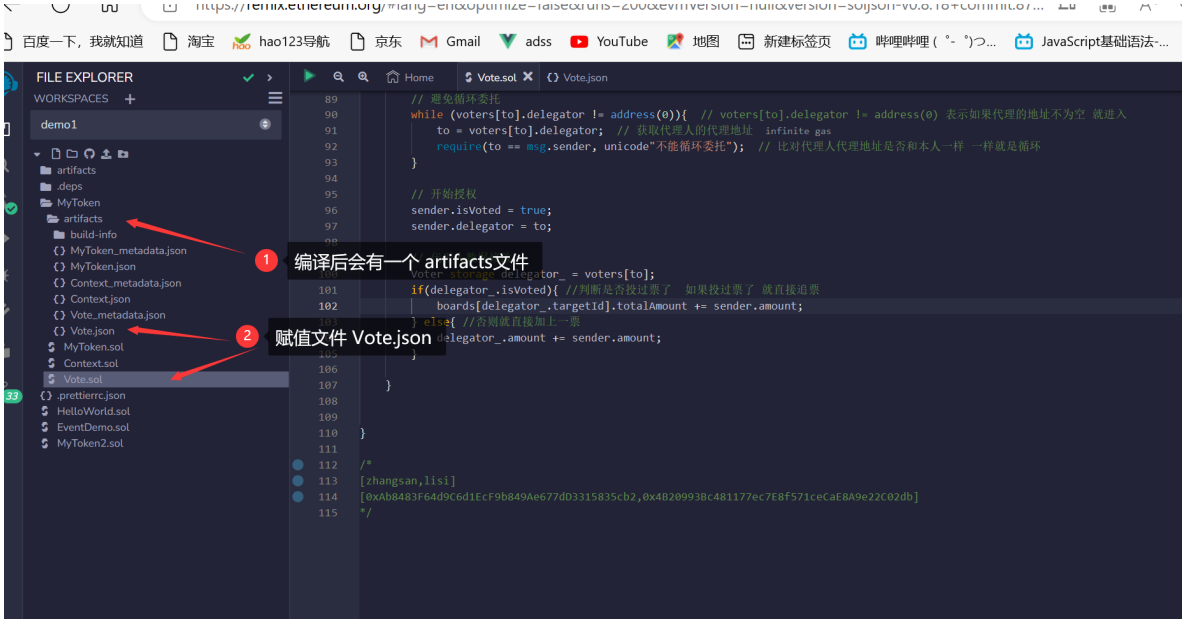
    2. 在vue.config.js 文件中配置插件

    ```
    const { defineConfig } = require("@vue/cli-service");
    // 引入插件
    const NodePolyfillWebpackPlugin = require("node-polyfill-webpack-plugin");
    // vant
    const { VantResolver } = require('unplugin-vue-components/resolvers');
    const ComponentsPlugin = require('unplugin-vue-components/webpack');
    module.exports = defineConfig({
      transpileDependencies: true,
      configureWebpack: {
        plugins: [
          new NodePolyfillWebpackPlugin(),
     ComponentsPlugin({ resolvers: [VantResolver()]}),
        ],
    ```

```
  },
});
```

5.2 构建项目：

1. 在src目录下导入智能合约 编译后的json文件：



1. 放在 项目中了 自行复制粘贴

2. css 文件

```
.nav {
  background-color: #ee742f;
  div {
    color: #fff !important;
  }
}
.box1 {
  padding: 0 10px;
  .host {
  }
  .btn {
    margin-top: 20px;
  }
}
.box2 {
  padding: 0 10px;
}

.label {
```

```
  width: 100px;
}
.address {
  word-wrap: break-word;
  word-break: break-all;
  user-select: all;
}
.votors {
  width: 200px;
  height: 300px;
  border-color: #eee;
}
```

3. App.vue

```
<script setup>
import {ref} from "vue";

const active = ref('')

</script>

<template>

  <div>
    <nav class="header">去中心化投票APP</nav>
    <main>
      <RouterView/>
    </main>
    <van-tabbar route v-model="active" active-color="#ee742f">
      <van-tabbar-item to="/" icon="search">分发票券</van-tabbar-item>
      <van-tabbar-item to="/account" icon="setting-o">账户信息</van-tabbar-item>
      <van-tabbar-item to="/board" icon="setting-o">投票看板</van-tabbar-item>
    </van-tabbar>
  </div>

</template>


<style lang="less" scoped>
.header {
  height: 44px;
  background-color: #ee742f;
  color: #fff;
  text-align: center;
  line-height: 44px;
}

</style>
```

```
<style  lang="less">
html,
body{
  margin: 0;
  padding: 0;
  overflow: hidden;
}
</style>
```

MandateView.vue

```
<script setup>
// 定义主持人信息
import {h, onMounted, ref} from "vue";
import useWeb3 from "@/hooks/useWeb3";
import {showNotify} from "vant";

const {web3, contractAddress, voteContract, getAccount} = useWeb3()

console.log(voteContract)
const host = ref("");
// 选民地址
const voterAddress = ref('');


// 获取主持人信息
const getHost = async () => {
  host.value = await voteContract.methods.host().call()
}


// 分发票权
const mandateSend = async () => {
  debugger
  const arr = eval(voterAddress.value); //  eval 将字符串转换为数组
  const account = await getAccount()
  voteContract.methods.mandate(arr).send({
    from: account
  }).on("receipt", () => {
    showNotify({type: 'success', message: '票券分发成功'})
  })
};

onMounted(async () => {
  await getHost()
})
```

```
/**
 *  账号地址:
 *  0xc7a8af3dF476DDDEae97435e67369a4c2254AE06
 *  0xDb753211383DbEE35f1eb8e18D7b4D6e6AA6E999
 *  0xad7bC79ab65395159AfC94fc4689F847C1D542f8
 *  0xF072479EEAe653DB8cf0dF9018e67ad1e6c43E3B
 *
['0xc7a8af3dF476DDDEae97435e67369a4c2254AE06','0xDb753211383DbEE35f1eb8e18D7b4D6e6AA6E999','0xad7bC79ab65395159AfC94fc4689F847C1D542f8','0xF072479EEAe653DB8cf0dF9018e67ad1e6c43E3B']
 */
</script>

<template>
  <div class="box1">
    <van-divider>分发票权</van-divider>
    <div class="host">
      <van-space>
        <p class="label">
          <van-icon name="manager"/>
          主持人地址
        </p>
        <p class="address">{{ host }}</p>
      </van-space>
    </div>
    <div>
      <van-space>
        <p class="label">
          <van-icon name="friends-o"/>
          投票人地址
        </p>
        <textarea class="votors" v-model="voterAddress"></textarea>
      </van-space>
    </div>
    <div class="btn">
      <van-button block @click="mandateSend">开始分发选票</van-button>
    </div>
  </div>
</template>

<style scoped lang="less">

</style>
```

AccountView.vue

```
<script setup>
import {onMounted, ref} from "vue";
```

```
import useWeb3 from "@/hooks/useWeb3";
import {showNotify} from "vant";
const {web3, contractAddress, voteContract, getAccount} = useWeb3()

const account = ref("");

// 选民信息
const voterInfo = ref({});

// 受托人地址
const delegatorAddress = ref("");


// 基本使用
const getVoteInfo = async () => {
  account.value = await getAccount();
  voterInfo.value = await voteContract.methods.voters(account.value).call();
};

// 进行委托
const delegate = () => {
  voteContract.methods
      .delegate(delegatorAddress.value)
      .send({ from: account.value })
      .on("receipt", (event) => {
        showNotify({type: 'success', message: '委托成功'})
      });
};

onMounted(async () => {
  await getVoteInfo();
});
</script>

<template>
  <div class="box2">
    <van-divider>账户信息</van-divider>
    <van-space>
      <p class="label">当前账户</p>
      <p class="address">{{ account }}</p>
    </van-space>
    <br />
    <van-space>
      <p class="label">账户票数</p>
      <p class="address">{{ voterInfo.amount }}</p>
    </van-space>
    <br />
    <van-space>
      <p class="label">委托账户</p>
      <p class="address">{{ voterInfo.delegator }}</p>
    </van-space>
    <br />
```

```
    <van-space>
      <p class="label">是否已投票</p>
      <p class="address">{{ voterInfo.isVoted }}</p>
    </van-space>
    <br />
    <van-space>
      <p class="label">投票ID</p>
      <p class="address">{{ voterInfo.targetId }}</p>
    </van-space>
    <br />

    <div class="btn">
      <van-cell-group inset>
        <van-field
            v-model="delegatorAddress"
            label="受托人地址"
            placeholder="请输入受托人地址"
        />
      </van-cell-group>
      <van-button block @click="delegate">委托他人代投</van-button>
    </div>
  </div>
</template>

<style scoped lang="less">

</style>
```

BoardView.vue

```
<script setup>
import useWeb3 from "../hooks/useWeb3";
import { ref, onMounted } from "vue";
const { web3, voteContract, getAccount } = useWeb3();

// 看板信息
const board = ref([]);

const account = ref("");

// 获取看板信息
const getBoardInfo = async () => {
  const result = await voteContract.methods.getBoardInfo().call();
  board.value = result;
};

// 进行投票
const vote = async (index) => {
  console.log(index)
```

```
  // 获取当前账户信息
  const account = await getAccount()
  //调用 智能合约中的方法 参数为 投给谁的ID
  const result = await voteContract.methods.vote(index).send({ from: account })
};

const initEventListen = () => {
  voteContract.events
      .voteSuccess({ fromBlock: 0 }, (err, event) => {
        console.log("监听执行");
        console.log(event);
      })
      .on("data", (event) => {
        console.log("智能合约触发的事件：", event.returnValues);
      });
};

onMounted(() => {
  initEventListen();
  getBoardInfo();
});
</script>

<template>
  <div class="box3">
    <van-divider>投票看板</van-divider>
    <van-cell :title="item.name" icon="shop-o" v-for="(item, index) in board">
      <template #right-icon>
        <van-button @click="vote(index)">{{ item.totalAmount }}</van-button>
      </template>
    </van-cell>
  </div>
</template>

<style scoped lang="less">

</style>
```