基于区块链的投票平台
详细设计文档

目录

一、 项目简介

我们构建一个去中心化的投票应用。利用这个投票应用，用户可以在不可信的分布环境中对特定候选人投票，每次投票都会被记录在区块链上。

之所以选择投票作为我们的第一个区块链应用，是因为集体决策，尤其是投票机制，是以太坊的一个核心的价值主张。另一个原因在于，投票是很多复杂的去中心化应用的基础构件，所以我们选择了投票应用作为学习区块链应用开发的第一个项目。

二、 系统总体设计

Ganache

Ganache 用于搭建私有网络。在开发和测试环境下，Ganache 提供了非常简便的以太坊私有网络搭建方法，通过可视化界面可以直观地设置各种参数、浏览查看账户和交易等数据。

Truffle

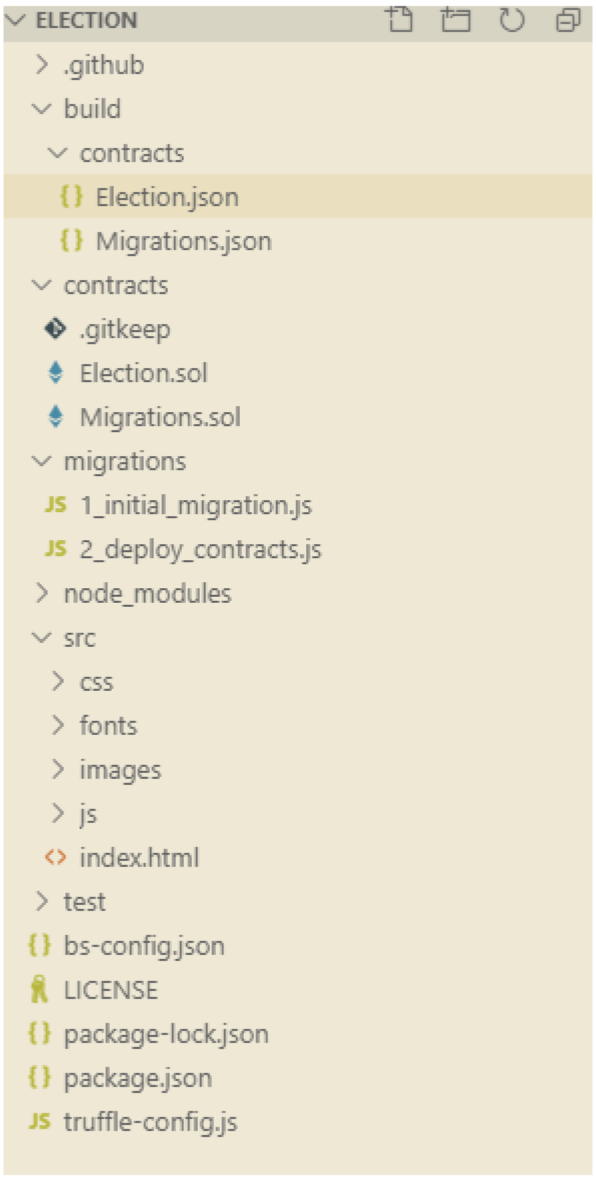
本系统使用 Truffle 开发工具。Truffle 是目前最流行的以太坊开发框架，采用 JavaScript 编写，支持智能合约的编译、部署和测试。它允许我们在网络中创建去中心化的应用。它为我们提供一套工具，使我们能够用 solidity 编程语言编写我们的智能合约。它还为我们提供了一个测试框架的智能合约，支持对合约代码的单元测试，非常适合测试驱动开发。同时内置了智能合约编译器，只要使用脚本命令就可以完成合约的编译、部署、测试等工作，大大简化了合约的开发生命周期。

MetaMask

MetaMask 是一款浏览器插件的钱包，不需要下载客户端，只需要添加其至浏览器扩展程序即可使用，非常方便，而且可以很方便得调试和测试以太坊的智能合约。

三、 智能合约设计

1. 目录结构


```
ELECTION
  .github
  build
    contracts
      Election.json
      Migrations.json
  contracts
    .gitkeep
    Election.sol
    Migrations.sol
  migrations
    1_initial_migration.js
    2_deploy_contracts.js
  node_modules
  src
    css
    fonts
    images
    js
    index.html
  test
  bs-config.json
  LICENSE
  package-lock.json
  package.json
  truffle-config.js
```

2. 详细实现

候选人建模

```
struct Candidate {
    uint id;
    string name;
    uint voteCount;
}
```

保存候选人

```
mapping(uint => Candidate) public candidates;
```

保存人数

```
uint public candidatesCount;
```

增加候选人

```
function addCandidate(string memory _name ) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount,_name,0);
}
```

增加两位候选人

```
constructor() public {
        addCandidate("Stephen Curry");
        addCandidate("Lebron James");
}
```

保存投票人

```
mapping(address => bool) public voters;
```

投票实现

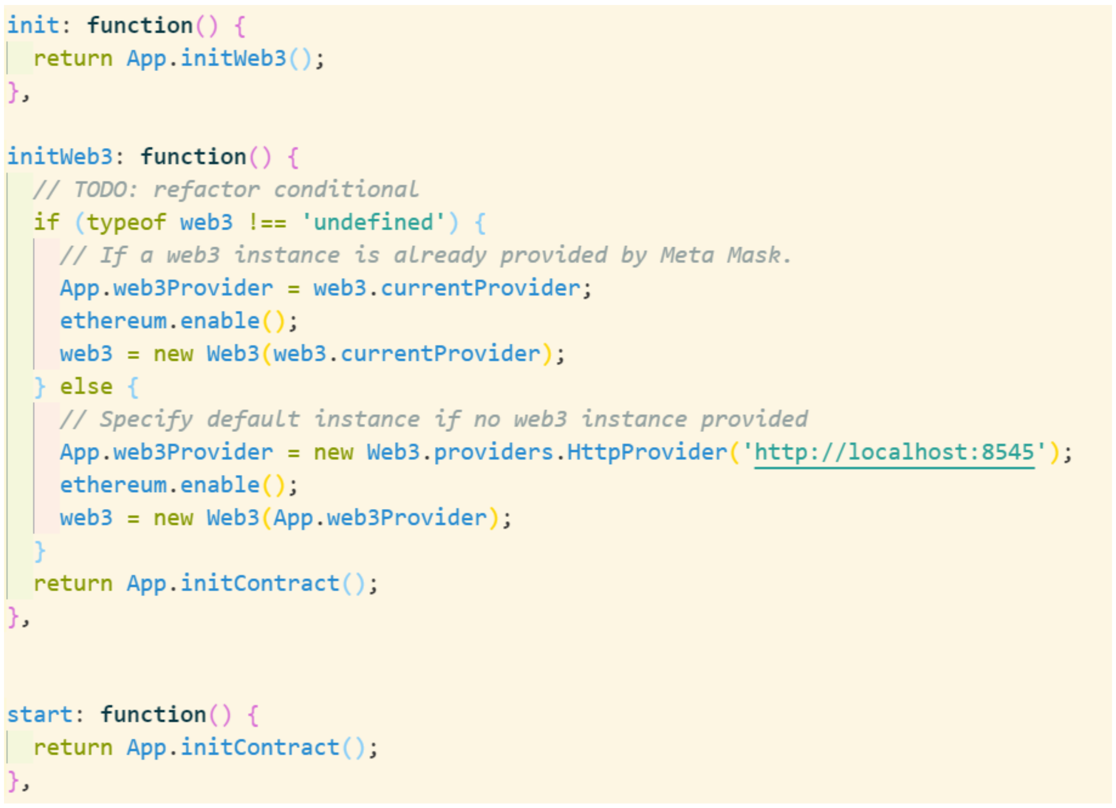
```
function vote(uint _candidateId) public {
        //要求投票者从没投过票
        require(!voters[msg.sender]);  //msg.sender是调用这个函数的账户
        //要求候选的Id合法
        require(_candidateId > 0 &&_candidateId <= candidatesCount);
        //确定投票
        voters[msg.sender] = true;
        //更新候选者票数
        candidates[_candidateId].voteCount ++;

        emit votedEvent(_candidateId);
}
```

四、 系统实现

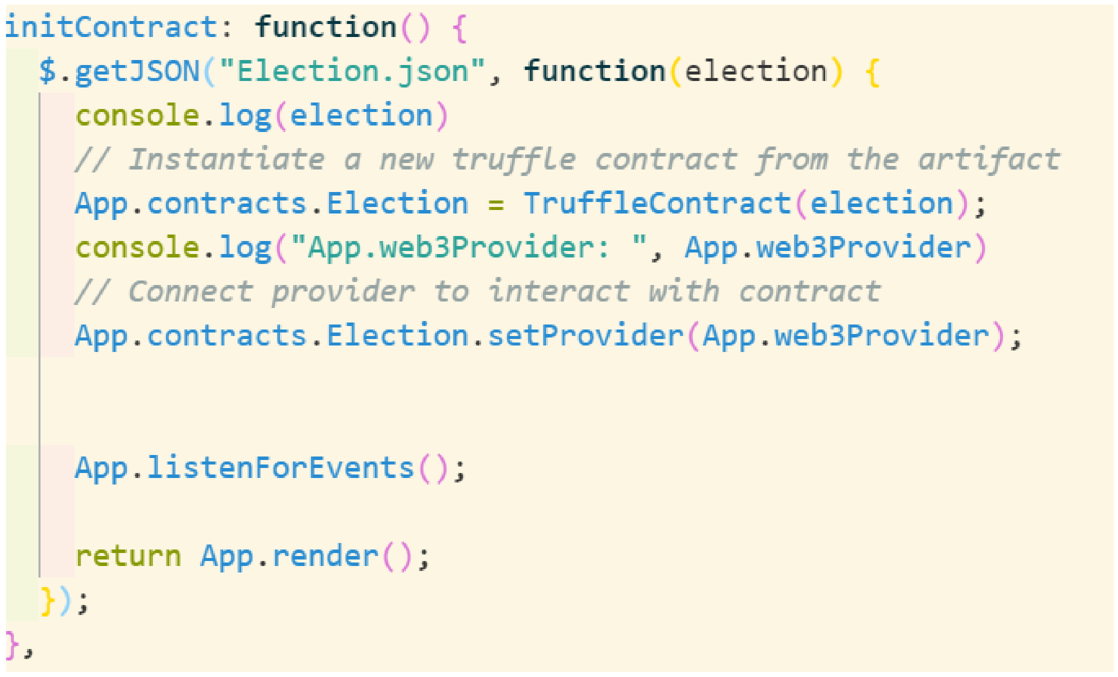
html 页面

```html
<h1 class="text-center">Election Results</h1>
<hr/>
<br/>
<div id="loader">
  <p class="text-center">Loading...</p>
</div>
<div id="content" style="display: none;">
  <table class="table">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Name</th>
        <th scope="col">Votes</th>
      </tr>
    </thead>
    <tbody id="candidatesResults">
    </tbody>
  </table>
  <hr/>
  <form onSubmit="App.castVote(); return false;">
    <div class="form-group">
      <label for="candidatesSelect">Select Candidate</label>
      <select class="form-control" id="candidatesSelect">
      </select>
    </div>
    <button type="submit" class="btn btn-primary">Vote</button>
    <hr />
  </form>
  <p id="accountAddress" class="text-center"></p>
</div>
```

连接以太坊



```
init: function() {
  return App.initWeb3();
},

initWeb3: function() {
  // TODO: refactor conditional
  if (typeof web3 !== 'undefined') {
    // If a web3 instance is already provided by Meta Mask.
    App.web3Provider = web3.currentProvider;
    ethereum.enable();
    web3 = new Web3(web3.currentProvider);
  } else {
    // Specify default instance if no web3 instance provided
    App.web3Provider = new Web3.providers.HttpProvider('http://localhost:8545');
    ethereum.enable();
    web3 = new Web3(App.web3Provider);
  }
  return App.initContract();
},


start: function() {
  return App.initContract();
},
```



```
initContract: function() {
  $.getJSON("Election.json", function(election) {
    console.log(election)
    // Instantiate a new truffle contract from the artifact
    App.contracts.Election = TruffleContract(election);
    console.log("App.web3Provider: ", App.web3Provider)
    // Connect provider to interact with contract
    App.contracts.Election.setProvider(App.web3Provider);


    App.listenForEvents();

    return App.render();
  });
},
```

监听事件

```
listenForEvents: function() {
  console.log("listenForEvents")
  App.contracts.Election.deployed().then(function(instance) {
    // Restart Chrome if you are unable to receive this event
    // This is a known issue with Metamask
    // https://github.com/MetaMask/metamask-extension/issues/2393
    instance.votedEvent({}, {
      fromBlock: 0,
      toBlock: 'latest'
    }).watch(function(error, event) {
      console.log("event triggered", event)
      // Reload when a new vote is recorded
      App.render();
    });
  });
},
```

投票人登录账户

```
web3.eth.getCoinbase(function(err, account) {
  console.log(err)
  console.log("account: ", account)
  if (err === null) {
    App.account = account;
    $("#accountAddress").html("Your Account: " + account);
  }
});
```

加载合约信息

```
App.contracts.Election.deployed().then(function(instance) {
  electionInstance = instance;
  return electionInstance.candidatesCount();
}).then(function(candidatesCount) {
  var candidatesResults = $("#candidatesResults");
  candidatesResults.empty();
  var candidatesSelect = $('#candidatesSelect');
  candidatesSelect.empty();
  for (var i = 1; i <= candidatesCount; i++) {
    electionInstance.candidates(i).then(function(candidate) {
      var id = candidate[0];
      var name = candidate[1];
      var voteCount = candidate[2];
      // Render candidate Result
      var candidateTemplate = "<tr><th>" + id + "</th><td>" + name + "</td><td>" + voteCount + "</td></tr>"
      candidatesResults.append(candidateTemplate);
      // Render candidate ballot option
      var candidateOption = "<option value='" + id + "' >" + name + "</ option>"
      candidatesSelect.append(candidateOption);
    });
  }
  return electionInstance.voters(App.account);
}).then(function(hasVoted) {
  // Do not allow a user to vote
  if (hasVoted) {
    $('form').hide();
  }
  loader.hide();
  content.show();
}).catch(function(error) {
  console.warn(error);
});
```

投票

```
castVote: function() {
  var candidateId = $('#candidatesSelect').val();
  App.contracts.Election.deployed().then(function(instance) {
    return instance.vote(candidateId, {
      from: App.accounts
    });
  }).then(function(result) {
    // Wait for votes to update
    $("#content").hide();
    $("#loader").show();
  }).catch(function(err) {
    console.error(err);
  });
}
```

五、 系统测试

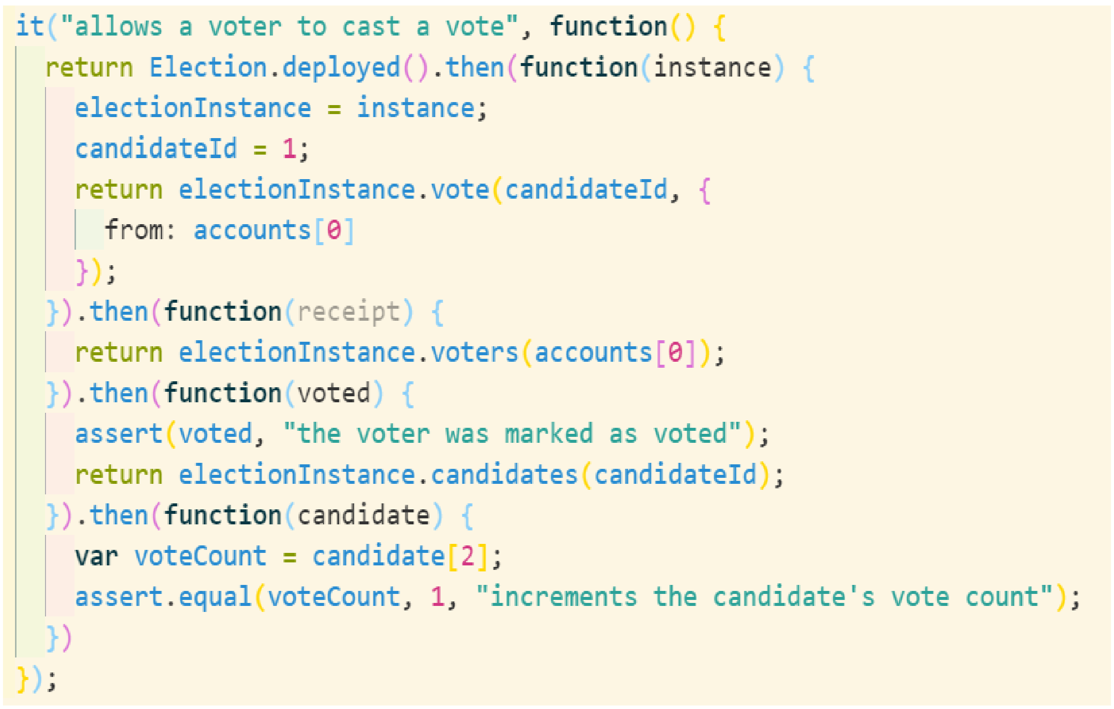
检查候选者人数

```
var electionInstance;
//初始化有两个候选者
it("initializes with two candidates", function() {
  return Election.deployed().then(function(instance) {
    return instance.candidatesCount();
  }).then(function(count) {
    assert.equal(count, 2);
  });
});
```

检查候选者信息

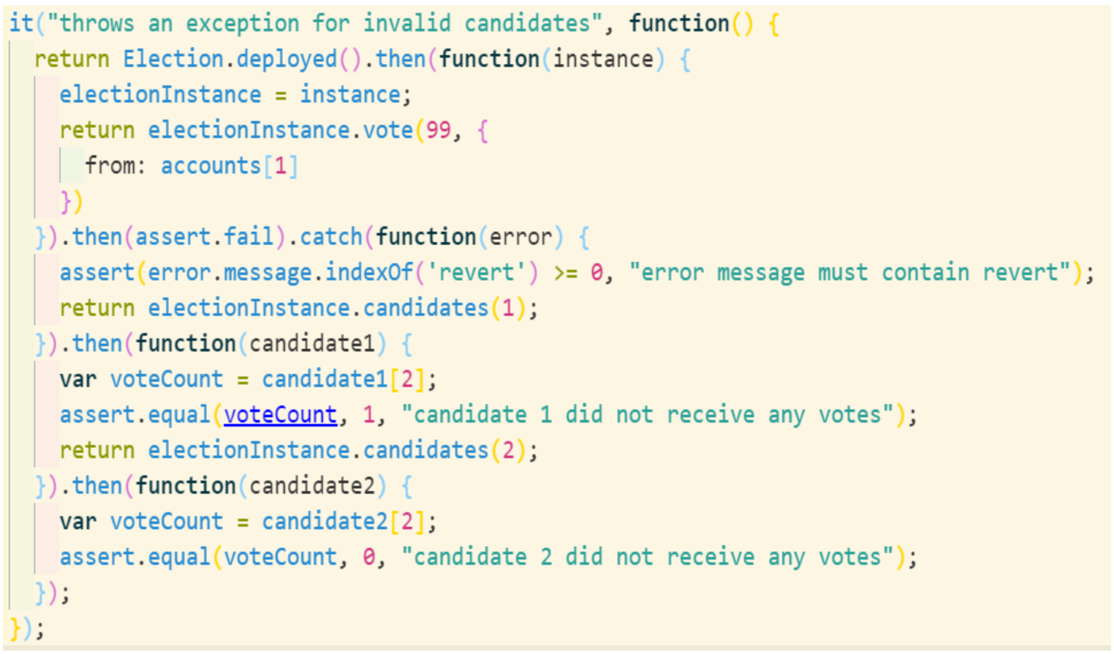
```
it("it initializes the candidates with the correct values", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    return electionInstance.candidates(1);
  }).then(function(candidate) {
    assert.equal(candidate[0], 1, "contains the correct id");
    assert.equal(candidate[1], "Candidate 1", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
    return electionInstance.candidates(2);
  }).then(function(candidate) {
    assert.equal(candidate[0], 2, "contains the correct id");
    assert.equal(candidate[1], "Candidate 2", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
  })
});
```

测试是否允许投票者进行投票



```
it("allows a voter to cast a vote", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    candidateId = 1;
    return electionInstance.vote(candidateId, {
      from: accounts[0]
    });
  }).then(function(receipt) {
    return electionInstance.voters(accounts[0]);
  }).then(function(voted) {
    assert(voted, "the voter was marked as voted");
    return electionInstance.candidates(candidateId);
  }).then(function(candidate) {
    var voteCount = candidate[2];
    assert.equal(voteCount, 1, "increments the candidate's vote count");
  })
});
```

测试对于非合法候选者进行投票



```
it("throws an exception for invalid candidates", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    return electionInstance.vote(99, {
      from: accounts[1]
    })
  }).then(assert.fail).catch(function(error) {
    assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
    return electionInstance.candidates(1);
  }).then(function(candidate1) {
    var voteCount = candidate1[2];
    assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
    return electionInstance.candidates(2);
  }).then(function(candidate2) {
    var voteCount = candidate2[2];
    assert.equal(voteCount, 0, "candidate 2 did not receive any votes");
  });
});
```

测试能否重复投票

```
it("throws an exception for double voting", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    candidateId = 2;
    electionInstance.vote(candidateId, {
      from: accounts[1]
    });
    return electionInstance.candidates(candidateId);
  }).then(function(candidate) {
    var voteCount = candidate[2];
    assert.equal(voteCount, 1, "accepts first vote");
    // Try to vote again
    return electionInstance.vote(candidateId, {
      from: accounts[1]
    });
  }).then(assert.fail).catch(function(error) {
    assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
    return electionInstance.candidates(1);
  }).then(function(candidate1) {
    var voteCount = candidate1[2];
    assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
    return electionInstance.candidates(2);
  }).then(function(candidate2) {
    var voteCount = candidate2[2];
    assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
  });
});
```

六、 系统部署

我们使用的 ganache 客户端默认使用 7545 端口，根据不同的需求和以太坊客户端的配置也可以修改端口号。truffle-config.js 实现如下：

```
module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // for more about customizing your Truffle configuration!
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*" // Match any network id
    },
    develop: {
      port: 8545
    }
  }
};
```

Truffle 项目只需要使用简单的命令行就可以完成合约的编译，把合约部署到以太坊，并把整个项目部署到本地服务器，然后使用前端页面就可以实现和区块

链的交互。在部署项目之前，先打开 ganache 客户端，然后在终端命令行进入项目根目录，分别执行以下命令：

truffle compile 编译智能合约，如果合约存在语法错误，编译将失败，并提示错误信息。

truffle migrate 部署智能合约，把编译完成的合约部署到以太坊客户端 ganache 中。

npm run dev 默认使用 3000 端口，把项目自动化部署到服务器中。

上述步骤执行完后，会自动打开 localhost:3000 界面，使用 MetaMask 导入 ganache 生成的账号进行测试使用。