中山大學
SUN YAT-SEN UNIVERSITY

《区块链原理与技术》

（最终报告）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	16 级数字媒体
学 生 姓 名 ：	颜承橹
学　　号 ：	15322244
时　　间 ：	2018　年　12　月　25　日

一. Github 地址

https://github.com/yanchlu/sellTicket

二. 选题背景

选题：使用区块链技术解决票务问题。

依据和背景：

区块链技术的特点是去中心化、公开透明，让每个人都可以参与数据库建立，而且每个建立的数据又是不可篡改的。除了实时处理交易、安全存储加密信息外，区块链技术还能够有效地减少交易双方之间并不会带来任何附加价值的中间环节，从而创造一个真正自由的市场。

在现实生活中，我们常常会面临的一个问题就是买不到票。比如看球赛时候，又比如赶春运的时候。在这些票务问题的背后往往都有黄牛党的影子。而区块链技术的出现无疑能解决这些问题，通过设置转让时间限制，区块链门票有效杜绝黄牛党倒卖门票的行为。此外，区块链的安全性既能防范他人伪造假票，又可以避免有无良商家出售重复的票。最要一点在于它注册方式独特，所即便真的出现事故，我们也能快速锁定嫌疑人的身份。

三、使用说明

1．智能合约介绍

在这个票务系统中一共有两个角色，分别是卖家seller和买家customer。

卖家角色拥有的功能是：

1） 创建并发售Ticket

```
function publishTicket(uint id,uint p,uint c,address sender) public {
    flag = 1;
    uint len = TicketSet.length;
    for(uint i = 0;i < len;i++){
        if(TicketSet[i].tid == id)
        return;
    }
    flag = 2;
    if (sender != seller) return;
    TicketSet.length = len + 1;
    Ticket memory newTicket;
    newTicket.tid = id;
    newTicket.price = p;
    newTicket.amount = c;
    TicketSet[len] = newTicket;
    flag = 3;
}
```

2) 给买家的钱包中充值（合约代码）

```
function Recharge(address c,uint money) public {
    if (msg.sender != seller) return;
    customers[c].wallet = customers[c].wallet + money;
}
```

买家角色拥有的功能是：

1) 购票。

```
function buyTicket(uint tid,uint num,address sender) public {
    flag = 1;
    uint p = 0;
    uint k;
    uint len = TicketSet.length;
    for(uint i = 0;i < len;i++){
        if(TicketSet[i].tid == tid){
            flag = 2;
            p = TicketSet[i].price;
            k = TicketSet[i].amount;
            break;
        }
    }
    if(flag == 1) return;
    Customer memory buyer = customers[sender]; // assigns reference
    p = p * num;
    flag = 2;
    if (p > buyer.wallet) return;
    flag = 3;
    if  (k < num) return;
    buyer.wallet = buyer.wallet - p;
    TicketSet[k].amount = TicketSet[k].amount - num;
    buyer.own[tid] = buyer.own[tid] + num;
    flag = 4;
}
```

2) 退票。

```
function refundTicket(uint tid,uint num,address sender) public {
    flag = 1;
    uint p;
    uint k;
    uint len = TicketSet.length;
    for(uint i = 0;i < len;i++){
        if(TicketSet[i].tid == tid){
            flag = 2;
            p = TicketSet[i].price;
            k = TicketSet[i].amount;
            break;
        }
    }
    flag = 2;
    Customer storage buyer = customers[sender];
    if(num > buyer.own[tid]) return;
    p = p * num;
    buyer.wallet = buyer.wallet + p;
    buyer.own[tid] = buyer.own[tid] - num;
    flag = 3;
}
```

3） 查看钱包余额

```
//customers' function
function showWallet(address c) public returns(uint){
    return customers[c].wallet;
}
```

两个角色都有的功能是：

1） 查看票的价格

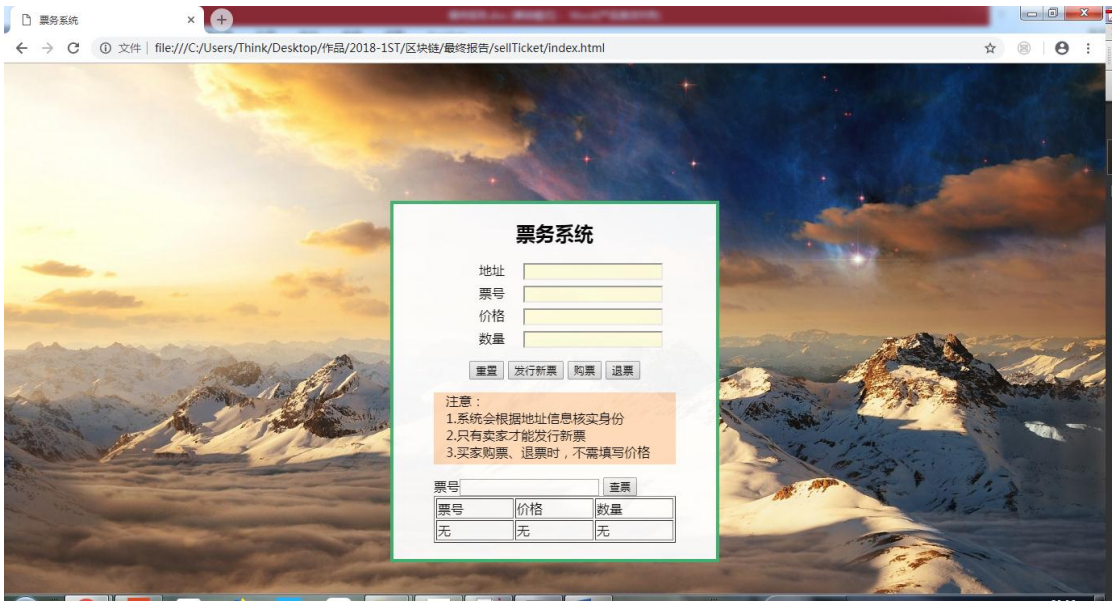
```
function showTicketPrice(uint id)constant public returns (uint) {
    return TicketSet[id].price;
}
```

2） 查看剩余票数

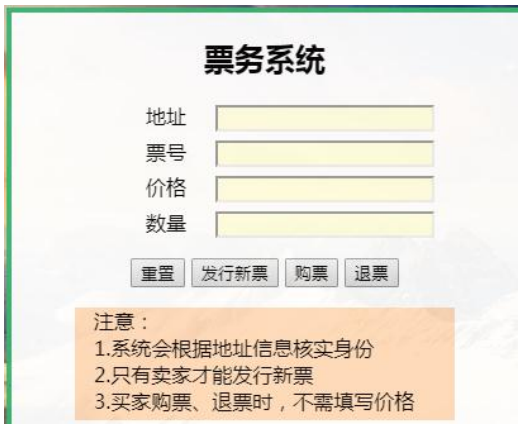
```
function showTicketAmount(uint id)constant public returns (uint) {
    return TicketSet[id].amount;
}
```

2．成品介绍

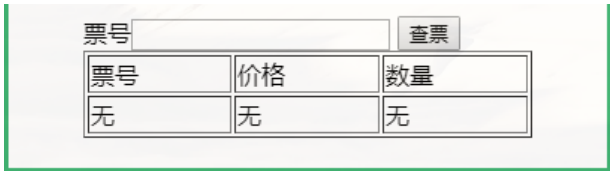
下图是通过JS实现的实际成品。



票务系统分为上下两个部分，上面的区域有三个功能，分别对应卖家发行新票，买家购票和买家退票。发行新票时，只需填入地址，将票的信息填入三个输入栏，然后点击“发行新票”按钮即可，此时的“数量”对应发行票的张数。购票和退票则不需要管“价格”一栏，此时的“数量”对应购买的票的张数。

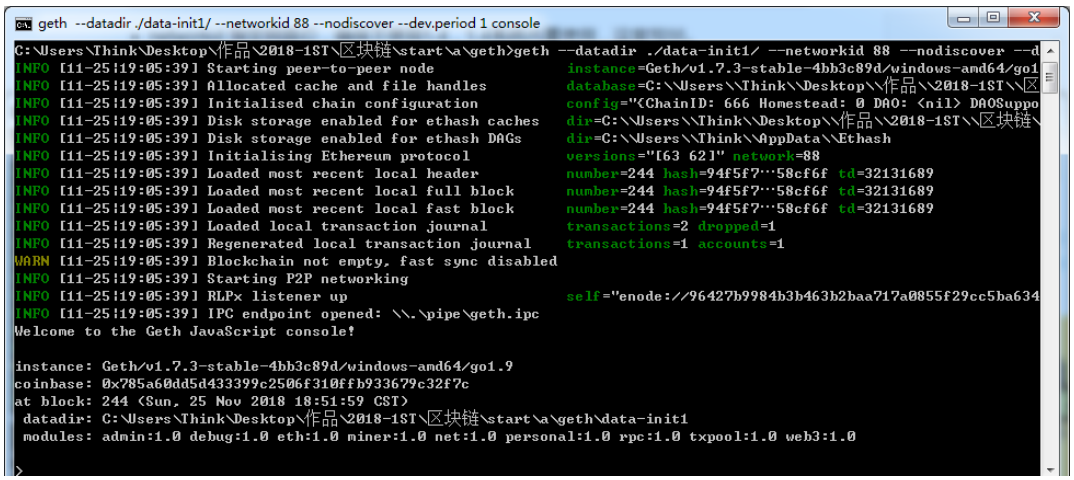


下半部分则是用于查看票务信息，输入票的id后，点击“查票”按钮将会显示出对应的票的信息。



3.部署说明

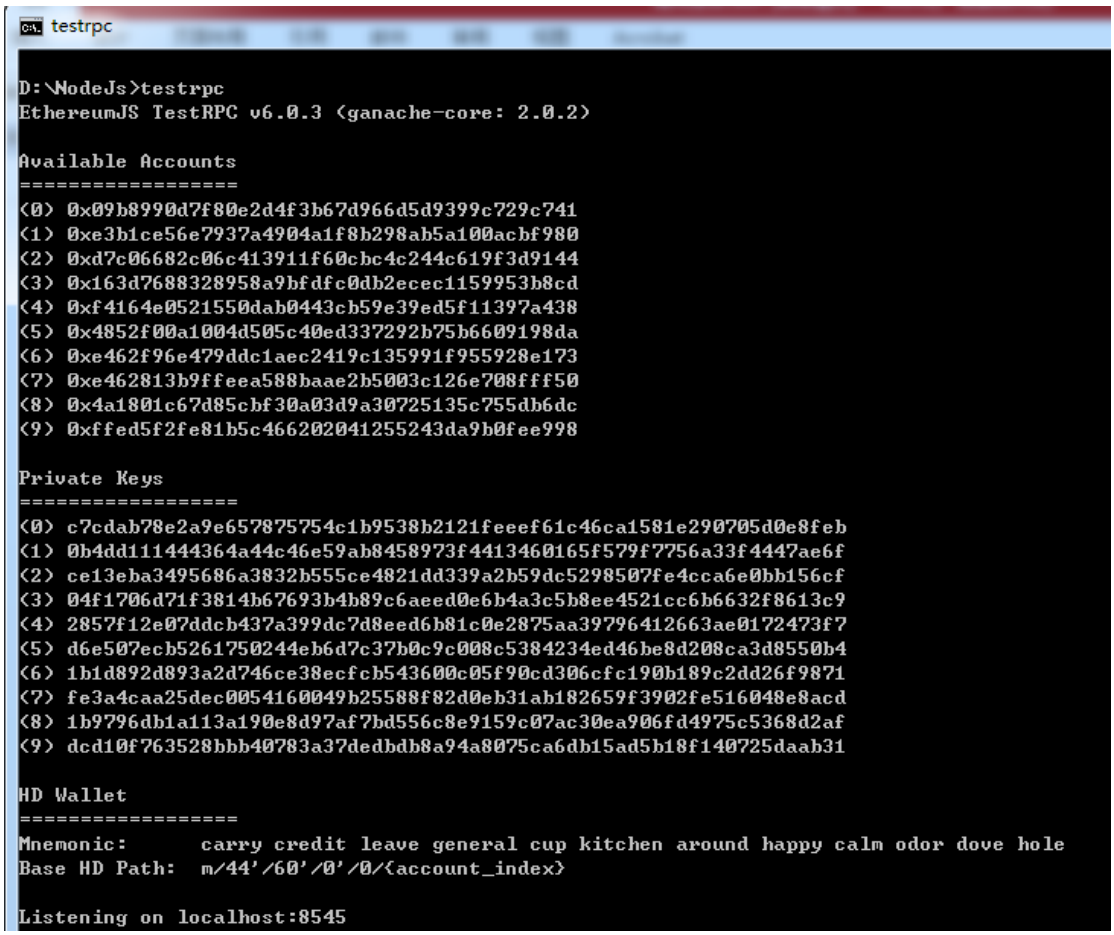
开启节点A，部署智能合约，作为卖家seller



随后将合约部署在 A 上，挖矿完成合约部署。

Contract mined! 说明合约部署成功。此时的合约地址为 0xf55cee308a729d297942e243ae2676ce0f699f20

在电脑上安装web3.js，它提供了和geth通信的JavaScript API。内部使用JSON RPC协议与geth通信。 随后启动testrpc。



如此一来便获得了客户账户。

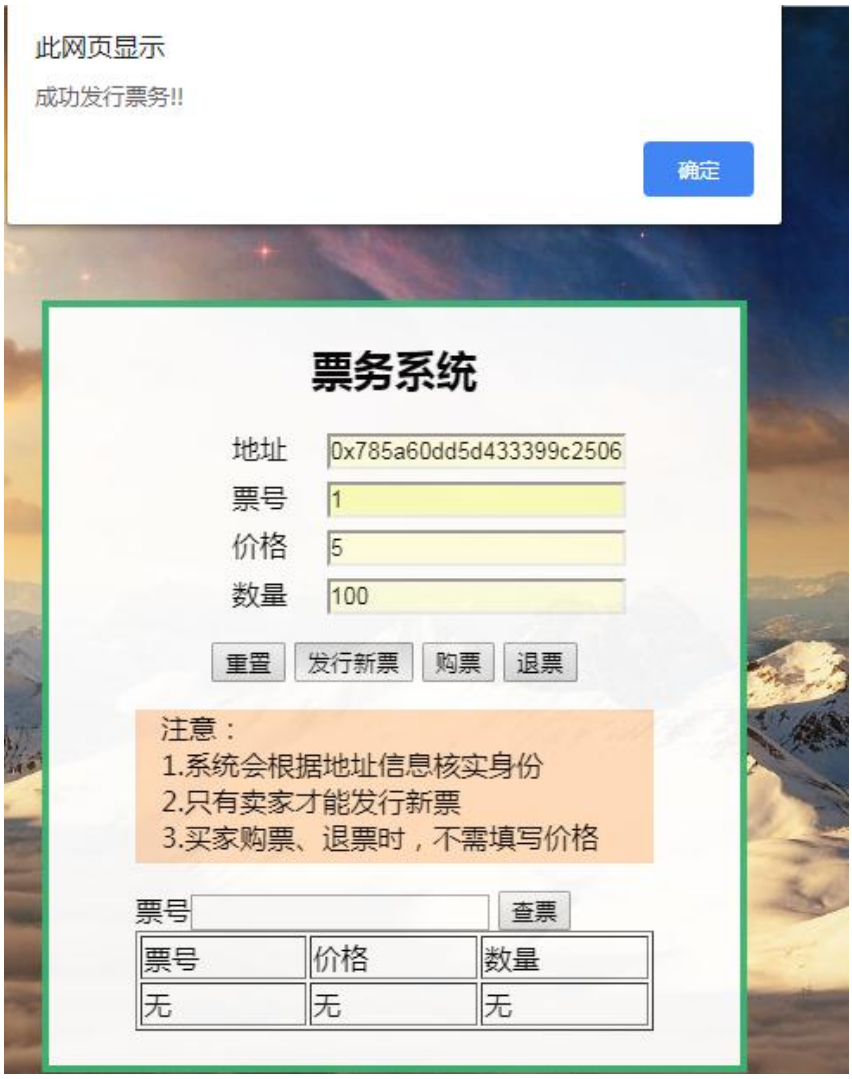
将之前获得的以太坊的相关信息填入JS代码中。

```
<script language="javascript">
        isStarted = false;
        if (typeof web3 !== 'undefined') {
            web3 = new Web3(web3.currentProvider);
        } else {
            web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545/"));
        }
        web3.eth.defaultAccount = web3.eth.coinbase;
        var api = [{"constant":false,"inputs":[{"name":"tid","type":"uint256"},{"name":"num",
        var sellTicketContract = web3.eth.contract(api);
        var st = sellTicketContract.at("0xf55cee308a729d297942e243ae2676ce0f699f20");
```

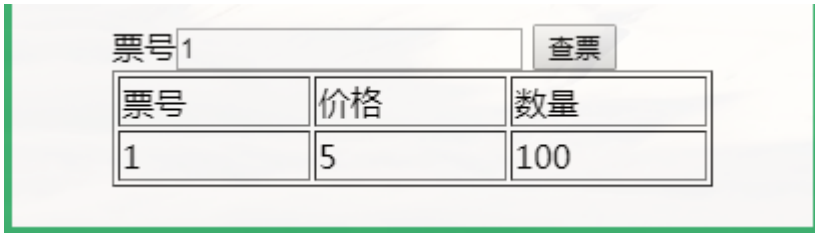
三、测试

1. 票务发行测试

（1）先尝试发行新票，填入卖家地址、票号、价格和数量信息，点击发行新票按钮，能看到新票发行成功。



然后在下面一栏查看票的信息



此时以太坊中也出现提示信息

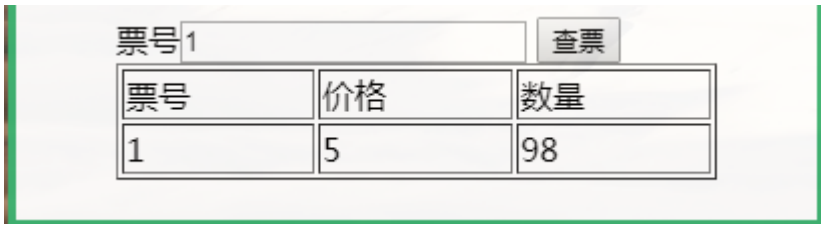
(2)再次发行新票，使用相同的票号，此时会反馈错误

（3）使用买家地址发行新票，同样也是失败的

2．购票测试

（1）使用买家地址购买2张1号票，不需填入价格，显示购票成功。

再次查询票的信息，显示此时只有98张票了



（2）使用买家账号再次买票，这次购买99张

显示购票失败，因为剩余票数只有98张。

（3）尝试让买家去购买不存在的票，票务系统能够发现并反馈该错误

3. 退票测试

（1）让买家先退一张票

然后查看票数

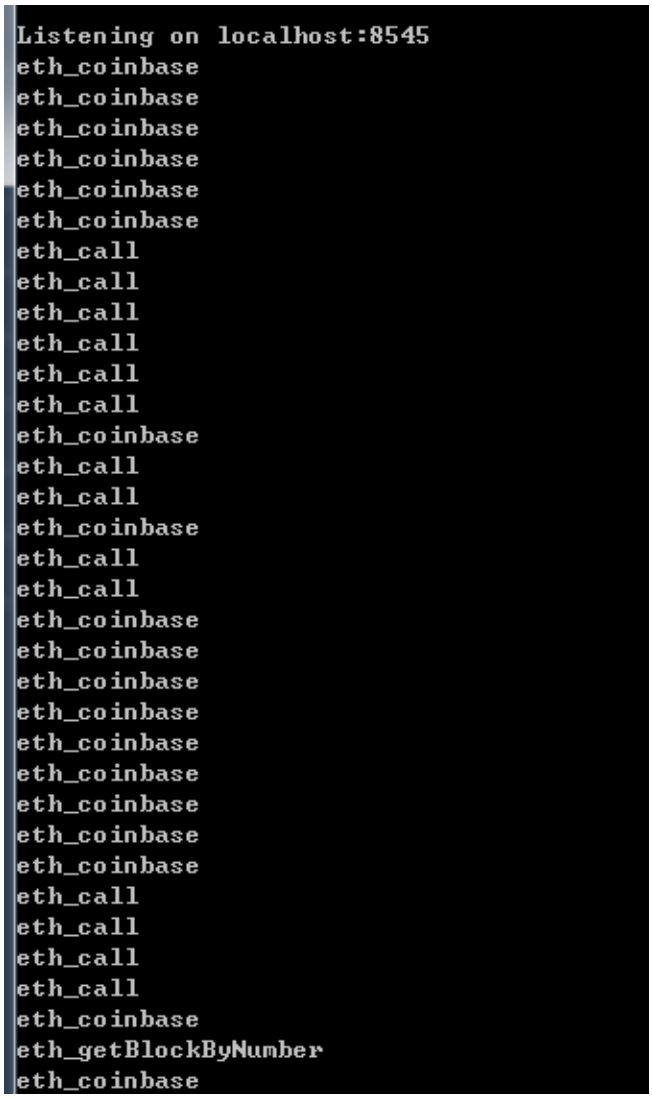
能看到确实将1张票退了回去。

（2）尝试让买家退换更多的票

提示退票失败，退票数超出了持票数。

（3）还有一种情况会出现退票失败，那就是根本就没有要退的那类票。

最后是testprc的截图



四、总结

这个票务系统可以说是“麻雀虽小五脏俱全”，买家卖家的基本功能都几乎实现了。不过在编写的时候我还是遇到了些难题。虽然在部署报告中已经将函数都测试过了，确保函数能运行，不过当我真正将后端和前端结合起来时，才发现这是一个不小的挑战。考虑到这是区块链课程，不是WEB课程，所以前端我写的有些简陋，这也导致了一些地方看起来比较奇怪，比如说在发行新票、购票和退票时需要输入地址。同时有些功能也很难体现出来，比如买家查看钱包的余额、卖家为买家的钱包中充虚拟货币等。原先我还设想弄个登录界面，但后来还是放弃了。

总体来说，这个基于JS开发的项目终于在今年下半年正式完成了，以后我也将更加努力学习区块链的知识，钻研代码，提升技术，争取做到前端后端两开花。