一、交易所分析

交易所现状

中国政府在2017年对数字货币作出了很多监管政策，因为严重破坏了市场的稳定性，支持区块链技术，但是目前明确禁止交易所非法集资，明确禁止数字货币的发行与交易，但是国外对数字货币很谨慎，没有明确禁止和支持，但允许发展。因此很多空气币都下架了，国内发展好的交易所也搬移到了国外发展，如火币网、OKCoin等。

交易所分类

交易所从交易流程可分为以下三种类型:

- 中心化交易所：在交易所本身的资料库中增减使用者资产栏位。
- 去中心化交易所：在区块链上直接交换，数字货币会直接发回使用者的钱包，或是保存在区块链上的智能合约。
- 半去中心化交易所：数字货币统一发至交易所指定钱包进行撮合交易，交易用的钱包实际控制人属于交易所。

交易所只是交易的工具，一般投资者并不十分关心交易所是否中心化、去中心化，中心化交易所的特点就是效率高处理速度快，所以大量中心化交易所产生了，也有部分交易量低的去中心化交易所。

交易类型

C2C交易

Customer/Consumer to Customer/Consumer，简单理解就是消费者对消费者的交易。

USDT交易

USDT是Tether公司推出的美元的代币，即1USDT=1美元，价格浮动可由交易所自身控制。

币币交易

数字资产与数字资产之间的交易。

思考

1. 在火币、gate.io等中心化交易所交易，挂单买卖时是否连接到了区块链上进行了转账操作?
2. 在个人中心，进行提现、充值业务时是否连接到了区块链上进行了转账操作?
3. 交易所上的账户是否拥有钱包账户？是否拥有自己可以操作的钱包账户?
4. 若拥有钱包账户，为何挂单买卖交易时无需输入钱包秘钥?
5. 若拥有自己可以操作的钱包账户，为何提现、充值时无需输入钱包秘钥?
6. 提现操作时的转出账户是哪个?
7. 提现操作时的转入账户是哪个?
8. 往钱包地址充值时，运行在区块链上进行的交易，那么交易所中心是如何知道的？并且将相应金额添加到了交易所的你的账户余额上?

如果你不能回答或是不能确定上面的问题，请继续后面课程，我们会逐一解答各个问题，并给出解决方案。

二. 业务功能

接下来我们做一个中心化交易所，它会拥有这些功能：用户系统、挂单、币转币、提现、充值、交易查询等。

用户系统

由于是中心化交易所，因此用户数据得存在自己的服务器中，这里使用mysql数据库存储数据，用户注册登录后，我们使用token验证用户的身份。

币转币

因为现在越来越多的代币在运作，并且上线了各大交易所，所以在交易的时候不止可以转以太币，还能转代币，并且支持币转币，在币转币的过程中就需要锁单、挂单、拆单、交易生成订单等过程。

可见该交易过程非常繁琐、各大交易所为了用户体验即交易及时、也是为了获取更多利益，将该交易演变成了不在链上交易，那么是如何交易的呢？通常都是在中央服务器拥有用户的账户系统，自己的token数据都是存在了中央服务器的数据库中，因此币转币的交易与其它交易如电商购物交易等，都是修改的数据库数据。由此可见大家在交易时所花费的交易费都由交易所获取。

提现

由于在交易所中的买卖交易没有在链上进行交易，也就是你所买的token，并不实际属于你，因此可以通过提现，将自己在交易所账户上拥有的token提到自己的钱包地址，这样就算交易所倒闭，自家的资金也是在钱包里的，可以通过其他交易所或者钱包进行提现或转账。

充值

有了提现功能、自然就会有充值功能，自己钱包地址的余额不能直接在交易所平台转账、买卖操作，因此通过充值功能将钱包地址的余额充到交易所平台的账户余额中，即可就行买卖交易了。

三. 架构设计

后台使用Koa框架搭建，和前面学习的钱包一样，采用MVC架构。

四. 数据库设计

在这个交易所项目中我们建立如下这些表储存系统的数据，如：

- user：用户表
- wallet：钱包表
- token：代币表
- usertoken：用户持有的token表
- transaction：挂单表
- order：挂单买卖交易订单表
- walletorder：钱包token交易订单表
- income：交易所收入表

user：用户表

字段名	含义
id	用户id，主键自增
phone	电话，这里作用户名
password	密码，md5处理后的数据
status	用户状态，默认为0正常

wallet：钱包表

储存所有用户的多个钱包，因为每个用户会有多个钱包，如以太坊钱包、比特币钱包等。

字段名	含义
id	钱包id，主键自增
userid	关联的用户id
address	钱包地址
privatekey	私钥
password	钱包的密码
keystore	备份文件
type	钱包类型，0是以太坊钱包，1是比特币钱包

token：代币表

记录交易所支持交易的所有token。

字段名	含义
id	代币id，主键自增
symbol	token符号
abi	合约二进制接口
address	合约地址
name	token名字
decimals	token小数位数
totalnum	token发布的总量
type	代币类型，用于标记所使用的钱包类型，与钱包类型的type对应

usertoken：用户持有的token表

与userid、tokenid关联，记录所有用户的所有token余额。

字段名	含义
id	usertoken id，主键自增
userid	用户id
tokenid	token id
balance	用户账户余额，包含锁定余额与可用余额
lockbalance	用户账户锁定余额
tokenbalance	用户钱包地址余额

transaction：挂单表

记录所有用户买卖token委托挂出的表。

字段名	含义
id	挂单 id，主键自增
userid	用户id
tokenid	交易的token id
replacetokenid	被交易的token id
price	委托价格
totalcount	委托总数量
count	可交易数量（因为挂单后可能一次交易未交易完委托的总数量）
status	交易状态，0代表可交易、1代表交易结束、2代表用户撤单
type	交易类型，0代表买，1代表卖
time	挂单时间

order：订单表

当挂单交易成功后，记录数据到订单表。为何不与transaction表公用？因为一个挂单可能会生成多个订单，并且生成的订单时间与挂单时间不一样。因此进行了拆分，但是order表需要关联transaction表。

字段名	含义
id	订单id，主键自增
userid	用户id
transactionid	挂单id
count	成功交易数量
time	订单生成时间

walletorder：钱包token交易订单表

记录用户使用钱包提现与充值的表。

字段名	含义
id	钱包token交易订单 id，主键自增
userid	用户id
tokenid	token id
hash	交易哈希
count	转账数额
type	交易类型，0代表提现，1代表充值
time	交易时间

income：交易所收入表

记录用户每笔买卖交易时赚的差价，另外还可包含手续费等。

字段名	含义
id	交易所收入id，主键自增
orderid	订单id
balance	所赠金额

五、项目依赖库

1. jsonwebtoken

实现web token令牌，默认签名算法是HMAC SHA256，也支持RSA SHA256签名等，expiresIn定义了到期声明

```
jwt.sign({
  data: 'foobar'
}, 'secret', { expiresIn: '1h' });
```

验证：

```
// verify a token symmetric - synchronous
var decoded = jwt.verify(token, 'secret');
console.log(decoded.data) // foobar

// verify a token symmetric
jwt.verify(token, 'secret', function(err, decoded) {
  console.log(decoded.data) // foobar
});
```

2. mysql

mysql的node.js驱动程序。连接上后调用的每个方法都按顺序排队并执行，完成后关闭调用`end()`，确保在将退出数据包发送到mysql服务器之前执行所有剩余查询。

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'me',
  password : 'secret',
  database : 'my_db'
});

connection.connect();

connection.query('SELECT 1 + 1 AS solution', function (error, results, fields) {
  if (error) throw error;
  console.log('The solution is: ', results[0].solution);
});

connection.end();
```

使用连接池可以不用逐个创建和管理连接，而是使用提供的内置连接池`mysql.createPool(config)`使用pool.getConnection()对于共享后续查询的连接状态很有用。这是因为两个调用 pool.query()可以使用两个不同的连接并且并行运行。如果要关闭连接并将其从池中删除，请调用`connection.destroy()`。池将在下次需要时创建新连接。

```
var mysql = require('mysql');
var pool  = mysql.createPool(...);

pool.getConnection(function(err, connection) {
  if (err) throw err; // not connected!

  // Use the connection
  connection.query('SELECT something FROM sometable', function (error, results,
fields) {
    // When done with the connection, release it.
```

```
    connection.release();

    // Handle error after the release.
    if (error) throw error;

    // Don't use the connection here, it has been returned to the pool.
  });
});
```

3. 其它

其它依赖库，在钱包项目中讲解过。