主体：

游戏公司G、游戏玩家A、B、交易平台

展示场景：

1、游戏公司G生成唯一道具S，向系统发起请求，系统返回道具ID，A购买之后，G向系统发起请求，通知道具的所有权为A。

2、A在交易平台出售道具S，交易平台通过ID查询确定S属于A，在平台上发布S的交易信息。

3、B决定购买S，向平台交付购买金。

4、A收到平台提示，在游戏中将道具S转让给B

5、游戏公司将S向系统发起转让申请，此时S的所有人为B

6、交易平台查询到交易已成功，将扣除手续费的钱款发放给A，交易完成。

页面：

1、游戏公司G

1.1 生成道具页面、所有权更改申请

1.2 查看当前游戏中的道具

2、游戏G的账号登录界面

3、A的界面

3.1商场表单，显示当前游戏公司发售的产品，可以购买

3.2显示自己已有的道具、转让道具

3.3 交易平台、平台信息、信息栏

4、B的界面

4.1商场表单，显示当前游戏公司发售的产品，可以购买

4.2显示自己已有的道具、转让道具

4.3交易平台、平台信息、信息栏

操作说明

1.1 选择道具生成的数量、种类，以及相关说明信息，点击生成

点击刷新，游戏玩家购买道具，或者玩家之间交易道具，显示在该页面，

点击确认，更改道具的所有权；

1.2 点击刷新，显示当前游戏G中所有的道具的ID、种类、发行数量、说明信息、当前所有者

2、输入用户名和密码，登录A和B的界面

3.1 点击刷新，将发售的道具的ID、类型、发行数量、相关信息发布在该页面，A点击购买之后，显示购买成功

3.2 点击刷新，显示当前属于A的道具的ID、种类、发行数量、说明信息、所有权变更历史

输入交易ID，交易对象 ，点击确认，完成所有权转让

3.3输入自己的道具ID，设置价格，点击发布，点击刷新，可以在平台上当前所有交易信息。

当B购买之后，刷新信息栏，可以看到B已支付的信息。前往3.2界面转让道具，转让成功之后点击我已转让。

4.3 B在平台上购买道具，扣除金额。当A将道具转让之后，点击刷新，会看到提示交易成功。

数据说明

**1.1**

输入道具数量、道具种类、说明信息，点击生成，提交

{

Number：

Type：

Info：

} 发送给服务器；服务器向区块链系统发送请求，生成道具

点击购买刷新，服务器将当前的购买交易：

{

User：

Id：

}返回，放在交易申请表单中。

点击交易刷新，服务器将当前的用户之间的交易：

{

Seller：

Buyer：

Id：

}返回，放在交易申请表中。

点击交易申请的确认按钮，前端将ID、新的所有者 信息送给服务器

{

Id:

NewOwner：

}

**1.2**

前端向服务器查询所有属于游戏G的道具，服务器返回所有道具的ID、种类、发行数量、说明信息、所有者（未被购买，所有者为G）（数量不定）

{

Id：

Type：

Number：

Info：

Owner：

}

**2、**

输入账号的密码，进入A和B的界面

{

User：

Password：

}

**3.1**

点击商场表单下方的刷新，返回当前所有者为G（即未售出）的道具的信息

{

Id：

Type：

Number：

Info：

}

点击购买，前端将当前用户、购买道具ID提交服务器

{

User：

Id：

}服务器将购买申请存储，以供游戏公司G刷新读取购买申请

**3.2**

点击刷新，向服务器请求当前属于A的道具信息，将信息显示在该页面

{

Id：

Type：

Number：

Info：

History:

}

输入道具ID、交易方，点击确认，将当前用户、买方、道具ID发送给服务器

{

User：

Buyer：

Id：

}服务器检查该ID是否为User所有，是则返回正确，将该交易申请存储，等待游戏公司G刷新读取交易申请，否则返回错误

**3.3**

A输入道具ID，设置价格，提交服务器

{

User：

Id：

Price：

}服务器查询Id是否属于User，是则将存储在交易平台数据库，否则返回错误

点击交易平台的刷新信息，服务器返回交易平台数据库中所有的待售信息

{

Id：

Type：

Number：

Info：

History：

Price：

}

点击账户余额的刷新，提交用户名

{

User：

}返回当前用户的账户余额

{

Balance：

}

点击信息栏的刷新，提交User，服务器查询交易平台数据库中的待售信息中A的道具是否被买下（即判断Buyer 和IsBought字段），是则返回被购买的道具ID和买家。

{

Att：Seller （表明这条信息是是关于用户出售的道具的）

Id：

Buyer：

}

转让道具之后，点击我已转让，提交用户、买家、道具Id

{

User:

Id：

Buyer:

}服务器查询Id是否属于Buyer，是则修改待售信息中的Istraded字段，Istransfered为false，则增加A的账户余额，置Istransfered为真，返回OK信息。

**4.3页面**

点击购买按钮，提交用户名、Id

{

User：

Id：

}服务器查询当前账户余额是否大于Id的价格，是则扣除金额，返回成功，并将交易平台数据库中的待售信息中的 Buyer 和IsBought字段修改。

点击信息栏的刷新按钮,提交User，服务器从查询交易平台数据库中的待售信息中查询被B买下的道具的记录中的Istraded是否为真，是则返回

{

Att：Buyer（表明这条信息是关于用户买下的道具的）

Id：

}并将该销售记录删除

前后端交互

需要两个数据库：

**1、游戏公司的数据库**

存储购买申请和交易申请

**2、交易平台的数据库**

存储A、B的账户余额

存储用户发布的出售信息

除了基础信息之外，还有Buyer、Isbought、Istraded，Istransfered字段

所有的post都需要返回OK信息。

**游戏公司G：**

1、游戏公司将道具的数量、种类，以及相关说明信息 转成json格式发送给服务器；

post：

{

Number:

Type:

Info:

}

2、前端得到用户名、购买ID（数目不定）（购买行为指的是购买G发行的道具）

get

{

User：

Id：

}

3、前端得到当前发起的交易的卖方，买方、道具ID（数目不定）

get

{

Seller：

Buyer：

Id：

}

4、前端将ID、新的所有者 信息送给服务器，服务器返回“OK”.

post：

{

Id:

NewOwner:

}

5、前端向服务器查询所有游戏道具，服务器返回所有道具的ID、种类、发行数量、说明信息、所有者（数目不定）

get

{

Id：

Type：

Number：

Info：

Owner：

}

**玩家A/B：**

1、服务器将所有人为G的道具（即未售出）的ID、种类、发行数目、相关信息返回前端（数目不定）

get：

{

Id：

Type：

Number：

Info：

}

2、前端提交购买用户、购买的道具ID

post

{

User：

Id：

}服务器将购买申请存储，以供游戏公司G刷新读取购买申请

3、前端向服务器查询所有属于A的道具，服务器返回道具的ID、种类、数量、说明信息、所有权变更历史。（数量不定）

get

{

Id：

Type：

Number：

Info：

History:

}

4、前端将当前用户、买方、道具ID发送给服务器

{

User：

Buyer：

Id：

}服务器检查该ID是否为User所有，是则返回正确，将该交易申请存储，等待游戏公司G刷新读取交易申请，否则返回错误

5、前端提交User，服务器返回User的账户余额

{

Balance：

}

6、输入道具ID，设置价格，提交服务器

{

User：

Id：

Price：

}服务器查询Id是否属于User，是则将待售信息存储在交易平台数据库，否则返回错误

7、点击交易平台的刷新信息，服务器返回交易平台数据库中所有的待售信息

{

Id：

Type：

Number：

Info：

History：

Price：

}

8、点击购买按钮，提交用户名、Id

{

User：

Id：

}服务器查询当前账户余额是否大于交易平台数据库中该Id的价格，是则扣除金额，返回成功，并将交易平台数据库中的待售信息中的 Buyer 和IsBought字段修改。

9、点击信息栏的刷新，提交User，服务器查询交易平台数据库中的待售信息中A的道具是否被买下（即判断Buyer 和IsBought字段），是则返回被购买的道具ID和买家。

{

Att：Seller （表明这条信息是是关于用户出售的道具的）

Id：

Buyer：

}

10、转让道具之后，点击我已转让，提交用户、买家、道具Id

{

User:

Id：

Buyer:

}服务器查询Id是否属于Buyer，是则修改待售信息中的Istraded字段，Istransfered为false，则增加A的账户余额，置Istransfered为真，返回OK信息。

11、点击信息栏的刷新按钮,提交User，服务器从查询交易平台数据库中的待售信息中查询被B买下的道具的记录中的Istraded是否为真，是则返回

{

Att：Buyer（表明这条信息是关于用户买下的道具的）

Id：

}并将该销售记录删除