# Delphy 系统设计草稿

|  |  |  |
| --- | --- | --- |
| 版本 | 修改人 | 修改内容 |
| V2017/7/15 | jqg | 创建文档 |
|  |  |  |

## 系统结构

### 总体结构

* 方案 1



### 1.2 流程描述

#### 1.2.1 Mobile App

用户在 Mobile App 上登录后，浏览相应的事件，进入相应的 Market，在 Market 界面进行买卖，Mobile App 生成 Ethereum TX 结构数据，发送 数据库锁定命令给 ORALCE REST SERVER, 锁定成功后，通过 Ethereum Wallet API 发送至 Ethereum 网络，获取 TX ID 后，Mobile App 将 TX（TX ID）等相关数据推送至 Oracle REST Server，Oracle REST Server 在链上上检查交易是否成功，如果成功则写入数据库，释放锁操作。基本流程：



用户在浏览相关 Event 和 Market 信息时，Mobile App 通过 REST API 将相关数据获取到 Mobile App，在相关的页面上进行显示

#### 1.2.2 Web App

用户通过浏览器连接 Oracle Web Server，登录后浏览相应的事件，进入相应的 Market，在 Market 界面进行买卖，其基本流程与 Mobile APP 保持一致。

用户在浏览相关 Event 和 Market 信息时，Web Server 获取到 Mobile App，在相关的页面上进行显示。

## Mobile APP

### 2.1 基本结构



2. 2 基本逻辑处理

2.2.1 类定义

1. 密钥对

public class KeyPair {  
 byte[] privateKey; //私钥

String uAddr; //地址  
}

1. 用户类定义

User{

String name,

String password,

KeyPair userKeyPair

}

1. 交易类

public class Transaction{

String from,//购买者地址

String to，//Market 地址

String txidOnEth，//链上的 txid

String ItemId,//所购买的选项信息

Int shares，//交易份数

BigDecimal value,//交易金额

Boolean buyOrSell,//true:buy,false:sell

}

1. 分类

public class Category{

String categoryId，

String categoryContent

}

1. Event 定义

public class Event {

String eventId，

String eventTile，

String eventContent

}

1. Market 类

public class Market{

String marketAddree，//market 地址

String eventId，//所属 EventId

Date createTime，//创建时间

Date closeTime,//关闭时间

Date endTime，//结束时间

List<MarketItemItem> itemList, //可以购买的选项

Int totalVotedPersonInMarket,//市场总计投票人数

}

1. MarketItem 类

public class MarketItem{

String itemId，//选项 ID

String itemContent,//选项内容说明

String pricePerShare，//每股价格

Int totalVotePerson, //该选项总计投票人数

}

1. 分页类

public class PageCondition {

int totalSize; //总行数（该值需要数据提供方赋值返回）  
 int pageSize; //每页的行数

int pageRangeStart; //起始页号

int pageRangeEnd; //结束页号

}

#### 2.2.1 类逻辑结构



#### 2.2.2 与 Oracle Server 通信 API

1. 用户注册

输入:  
 pusr:用户类信息

输出:   
 无

返回：

成功 true，失败 false

Public boolean register(User pusr) throws Exception;

1. 用户登录

输入:  
 pusr:用户类信息

输出:   
 无

返回：

成功 true，失败 false

Public boolean signIn(User pusr) throws Exception;

1. 获取分类信息

输入:  
 无

输出:   
 无

返回：

Category：分类数组信息

Public List<Category> getCategoryFromOracle() throws Exception;

1. 获取 Event 信息

输入:  
 PageCondition:分页信息

输出:   
 无

返回：

Event：数组信息

成功 true，失败 false

Public List<Event> getEventsFromOracle(PageCondition pageCon) throws Exception;

1. 获取指定 Event 的 Market

输入:

Event：指定的 Event

PageCondition:分页信息

isFavorite: true 查询推荐的 Market，否则查询指定 Event 下的 Market

输出:   
 无

返回：

Event：数组信息

Public List<Market> getMarketsFromOracle(Event event，PageCondition pageCon,Boolean isFavorite) throws Exception;

1. 从Market 购买或卖出份额

输入:

Transaction：Transaction 信息

输出:   
 无

返回：

Boolean 成功 true，失败 false

Public Boolean buyOrSellMarketSharesFromMarket(Transaction tx) throws Exception;

1. 查询Market 当前价格

输入:

Market：Market信息

输出:   
 无

返回：

BigDecimal 当前 Market 市场价格

Public BigDecimal getMarketSharePrice(Market market) throws Exception;

1. 按时间段查询Market 资金投入历史

输入:

Market：Market信息

输出:   
 无

返回：

BigDecimal 当前 Market

Public List< BigDecimal> getMarketInputList(Market market，Date from，Date to) throws Exception;

1. 按时间段查询Market Share 价格历史

输入:

Market：Market信息

输出:   
 无

返回：

BigDecimal 当前 Market 市场价格

Public List< BigDecimal> getMarketSharePriceList(Market market，Date from，Date to) throws Exception;

1. 查询当前用户所参与的 Market

输入:

User：指定的 User

PageCondition:分页信息

输出:   
 无

返回：

List<Market>：Market 的列表

Public List<Market> getUserMarketsFromOracle(User user，PageCondition pageCon) throws Exception;

1. 查询当前用户在某个 Market 所持有的 Share

输入:

User：指定的用户

Market：指定的 Market

输出:   
 无

返回：

Event：数组信息

Public List< MarketItem > getUserMarketItemsFromOracle(User user，Market market) throws Exception;

1. 查询当前用户的所有交易

输入:

User：指定的 User

PageCondition:分页信息

输出:   
 无

返回：

Transaction：List 信息

Public List< Transaction> getUserTxFromOracle(User user，PageCondition pageCon) throws Exception;

1. 进入指定的市场队列

输入:

Market：指定的 Market

输出:   
 无

返回：

Boolean：true 锁定成功，false 失败

Public Boolean enterMarketQueFromOracle (Market market) throws Exception;

1. 查询是否可以进行交易

输入:

Market：指定的 Market

输出:   
 无

返回：

Boolean：true 可以进行，false 不可以进行

Public Boolean gotMarketLockFromOracle(Market market) throws Exception;

1. 检查交易是否成功

输入:

Transaction：指定的交易

输出:   
 无

返回：

Boolean：true 锁定成功，false 失败

Public Boolean checkTxStatusFromOracle(Transaction tx) throws Exception;

#### Ethereum 钱包提供的 API

1. 生成密钥对

输入:

无

输出:   
 无

返回：

KeyPair：密钥对

KeyPair generateKeyPair() throws Exception;

1. 在 Ethereum 网络进行交易

输入:

Transaction 交易信息

User 用户私钥信息

输出:   
 无

返回：

Boolean：成功 true，失败 false

Public Boolean buyOrSellMarketSharesFromEthereum (Transaction tx，User user) throws Exception;

1. 获取用户钱包内的ETH 或 DLP 币数量

输入:

Use：用户信息

Int：ETH 0, DLP 1

输出:   
 无

返回：

BigDecimal：对应币种的数量

Public BigDecimal getETHFromEthereum (User user，int type) throws Exception;

## Oracle Server

### 3.1 基本组成结构



### ORACLE RestAPI

1. 响应来自 Mobile APP 的用户注册 POST 请求

app.post('/registerUser', async function (req, res) {})

1. 响应来自 Mobile APP 的用户登录 POST 请求

app.post('/ signIn’, async function (req, res) {})

1. 响应来自 Mobile APP 的获取 Eevents 的 post 请求

app.post('/ getCategoryFromOracle’, async function (req, res) {})

1. 响应来自 Mobile APP 的获取 Eevents 的 post 请求

app.post('/ getEventsFromOracle’, async function (req, res) {})

1. 响应来自 Mobile APP 获取指定 Event 的 Market 的 POST 请求

app.post('/ getMarketsFromOracle’, async function (req, res) {})

1. 响应来自 Mobile APP从Market 购买或卖出份额的 POST 请求

app.post('/ buyOrSellMarketSharesFromMarket’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ getMarketSharePrice’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ getMarketInputList’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ getMarketSharePriceList’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ *getUserMarketsFromOracle*’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ *getUserMarketItemsFromOracle*’, async function (req, res) {})

1. 响应来自 Mobile APP查询Market 当前价格POST 请求

app.post('/ *getUserMarketItemsFromOracle*’, async function (req, res) {})

1. 响应来自 Mobile APP进入 Market 队列 POST 请求

app.post('/ enterMarketQueFromOracle’, async function (req, res) {})

1. 响应来自 Mobile APP查询是否已获取市场锁 Market POST 请求

app.post('/gotMarketLockFromOracle’, async function (req, res) {})

1. 响应来自 Mobile APP查询交易在链上是否成功 POST 请求

app.post('/checkTxStatusFromOracle’, async function (req, res) {})

### Ethereum 钱包提供的 API

1. 生成密钥对

输入:

无

输出:   
 无

返回：

KeyPair：密钥对

KeyPair generateKeyPair() throws Exception;

1. 在 Ethereum 链上进行交易

输入:

Transaction 交易信息

User 用户私钥信息

输出:   
 无

返回：

Boolean：成功 true，失败 false

Public Boolean buyOrSellMarketSharesFromEthereum (Transaction tx，User user) throws Exception;

1. 获取用户钱包内的ETH 或 DLP 币数量

*输*入:

User用户信息

Int：0 为 ETH , 1 为 DLP

输出:   
 无

返回：

BigDecimal：对应币种的数量

Public BigDecimal getETHFromEthereum (User user，int type) throws Exception;

1. 检查交易是否上链

*输*入:

User用户信息

Int：0 为 ETH , 1 为 DLP

输出:   
 无

返回：

Boolean：true 成功，false 失败

Public Boolean checkTxOnEthereum (String txId) throws Exception;

## 通信数据约定

对于 Mobile APP 到 ORACLE REST API 的数据传输格式采用 json 字符串的格式，采用 key：value 的形式组织数据。

### 4.1 register 函数 json 数据

post 请求：

{“cmd”:” register”,

”User”:{

“name”:” ValueOfName”,

”password”:”ValueOf password”,

“address”:”valueOfAdress”

}

}

Rest API 回应:

成功

{“cmd”:” register”,

“token”:”valueOftoken”

”result”:”ok”

}

失败

{“cmd”:” register”,

”result”:”fail”,

”reason”:”description of reason”

}

### signIn 函数 json 数据

post 请求

{“cmd”:” signIn”,”User”:{“name”:” ValueOfName”,”password”:”ValueOf password”}}

Rest API 回应:

成功

{“cmd”:” signIn”,”result”:”ok”}

失败

{“cmd”:” signIn”,”result”:”fail”,”reason”:”description of reason”}

### getCategoryFromOracle 函数 json 数据

POST 请求

{“cmd”:”getCategoryFromOracle” }

REST API 回应

成功

{“cmd”:” getCategoryFromOracle”,

”result”:”ok”,

Categories:[

{”categoryId”:”value of categoryId”,”

categoryContent”:”value of categoryContent”

},

{”categoryId”:”value of categoryId”,”

categoryContent”:”value of categoryContent”}

…]

}

失败

{“cmd”:” getCategoryFromOracle”,”result”:”fail”,”reason”:”description of reason”}

### getEventsFromOracle函数 json 数据

POST 请求

{“cmd”:”getEventsFromOracle” ,

PageCondition :

{

“pageSize”:”valueof pageSize”,”

pageRangeStart”:”valueof pageRangeStart”,”

pageRangeEnd”:”valueofpageRangeEnd”

}

}

REST API 回应

成功

{“cmd”:”getEventsFromOracle”,

”result”:”ok”,

Events:[

{

” eventId”:”value of eventId”,

” eventTile”:”value of eventTile”,

“eventContent”:”value of eventContent”

},

{

” eventId”:”value of eventId”,

” eventTile”:”value of eventTile”,

“eventContent”:”value of eventContent”

},

…]

}

失败

{“cmd”:” getEventsFromOracle”,

”result”:”fail”,

”reason”:”description of reason”

}

### getMarketsFromOracle函数 json 数据

POST 请求

{“cmd”:” getMarketsFromOracle” ,

”eventId”:”value of eventId”,

PageCondition :{

“pageSize”:”valueof pageSize”,”

pageRangeStart”:”valueof pageRangeStart”,”

pageRangeEnd”:”valueofpageRangeEnd”

}

}

REST API 回应

成功

{“cmd”:” getMarketsFromOracle”,

”result”:”ok”,

Markets:[

{

marketAddree:”valueof marketAddree”

eventide:”valueof eventide”

createTime:”valueof createTime”

closeTime:”valueof closeTime”

endTime:”valueof endTime”

MarketItemItems:[

{

itemId:”valueof itemId”，

itemContent:”valueof itemContent”,

pricePerShare:”valueof pricePerShare”，

totalVotePerson:”valueof totalVotePerson”,

}

…

]

totalVotedPersonInMarket:”valueof totalVotedPersonInMarket”

},

…]

}

失败

{“cmd”:” getMarketsFromOracle”,

”result”:”fail”,

”reason”:”description of reason”

}

### buyOrSellMarketSharesFromMarket函数 json 数据

POST 请求

{“cmd”:” buyOrSellMarketSharesFromMarket”

TX:{

From:”valueofFrom”,

To:”valueofTo”,

ItemId:”valueof ItemId”,

Shares:”valueofShares”,

Value:”value”,

buyOrSell:”valueof buyOrSell”

}

}

REST API 回应

成功

{“cmd”:” buyOrSellMarketSharesFromMarket”,

”result”:”ok”,

}

失败

{

“cmd”:” buyOrSellMarketSharesFromMarket”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.7 getMarketSharePrice函数 json 数据

post 请求：

{“cmd”:” getMarketSharePrice”,

”MarketId”:”valueOfMarketId”

}

Rest API 回应:

成功

{“cmd”:” getMarketSharePrice”,

”result”:”ok”

“price”:”valueOfShare”

}

失败

{“cmd”:” getMarketSharePrice”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.8 getMarketInputList函数 json 数据

post 请求：

{“cmd”:” getMarketInputList”,

”MarketId”:”valueOfMarketId”

“From”:”valueOfFrom”,

“To”:”valueOfTo”

}

Rest API 回应:

成功

{“cmd”:” getMarketInputList”,

”result”:”ok”

input:[

{input:”valueOfInput”},

{input:”valueOfInput”},

…

]

}

失败

{“cmd”:” getMarketInputList”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.9 getMarketSharePriceList函数 json 数据

post 请求：

{“cmd”:” getMarketSharePriceList”,

”MarketId”:”valueOfMarketId”

“From”:”valueOfFrom”,

“To”:”valueOfTo”

}

Rest API 回应:

成功

{“cmd”:” getMarketSharePriceList”,

”result”:”ok”

prices:[

{price:” valueOfprice”},

{ price:”valueOf price”},

…

]

}

失败

{“cmd”:” getMarketInputList”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.10 *getUserMarketsFromOracle*函数 json 数据

post 请求：

{“cmd”:” *getUserMarketsFromOracle*”,

”userAress”:”valueOfUserAdress”

}

Rest API 回应:

成功

{“cmd”:” *getUserMarketsFromOracle*”,

”result”:”ok”

Markets:[

{

marketAddree:”valueof marketAddree”

eventide:”valueof eventide”

createTime:”valueof createTime”

closeTime:”valueof closeTime”

endTime:”valueof endTime”

MarketItemItems:[

{

itemId:”valueof itemId”，

itemContent:”valueof itemContent”,

pricePerShare:”valueof pricePerShare”，

totalVotePerson:”valueof totalVotePerson”,

}

]

totalVotedPersonInMarket:”valueof totalVotedPersonInMarket”

},

…

]

}

失败

{“cmd”:” *getUserMarketsFromOracle*”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.11 *getUserMarketItemsFromOracle*函数 json 数据

post 请求：

{“cmd”:” *getUserMarketItemsFromOracle*”,

”MarketId”:”valueOfMarketId”

“UserAdress”:”valueOfUserAdrress”

}

Rest API 回应:

成功

{“cmd”:” *getUserMarketItemsFromOracle*”,

”result”:”ok”

VotedItems:

[

{

itemId:”valueof itemId”，

itemContent:”valueof itemContent”,

votedShares:”valueOf votedShares”

},

….

]

}

失败

{“cmd”:” *getUserMarketItemsFromOracle*”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.12 *getUserTxFromOracle*函数 json 数据

post 请求：

{“cmd”:” *getUserTxFromOracle*”,

”userAddress”:”valueOf userAddress”,

PageCondition :{

“pageSize”:”valueof pageSize”,”

pageRangeStart”:”valueof pageRangeStart”,”

pageRangeEnd”:”valueofpageRangeEnd”

}

}

Rest API 回应:

成功

{“cmd”:” *getUserTxFromOracle*”,

”result”:”ok”

txs:[

{

From:”valueofFrom”,

To:”valueofTo”,

ItemId:”valueof ItemId”,

Shares:”valueofShares”,

Value:”value”,

buyOrSell:”valueof buyOrSell”

},

…

]

}

失败

{“cmd”:” *getUserTxFromOracle*”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.13 enterMarketQueFromOracle函数 json 数据

post 请求：

{“cmd”:” enterMarketQueFromOracle”,

”MarketId”:”valueof MarketId”

}

Rest API 回应:

成功

{

“cmd”:” enterMarketQueFromOracle”,

”result”:”ok”

}

失败

{“cmd”:” enterMarketQueFromOracle”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.14 gotMarketLockFromOracle函数 json 数据

post 请求：

{“cmd”:” gotMarketLockFromOracle”,

”MarketId”:”valueof MarketId”

}

Rest API 回应:

成功

{“cmd”:” gotMarketLockFromOracle”,

”result”:”ok”

}

失败

{“cmd”:” gotMarketLockFromOracle”,

”result”:”fail”,

”reason”:”description of reason”

}

### 4.15 checkTxStatusFromOracle函数 json 数据

post 请求：

{“cmd”:” checkTxStatusFromOracle”,

”TxId”:”valueof TxId”

}

Rest API 回应:

成功

{“cmd”:” checkTxStatusFromOracle”,

”result”:”ok”

“txAddress”:”valueOftxAddress”

}

失败

{“cmd”:” checkTxStatusFromOracle”,

”result”:”fail”,

”reason”:”description of reason”

}

## 5. 技术问题

### 5.1 记账问题

Delphy 的市场价格是通过 ORACLE Server 来计算的，这就需要 Mobile APP 的交易数据需要首先发送至 ORACLE Server 将交易数据写入数据库，然后再将数据发送到 Ethereum 去记账。

* 用户交易有可能在 Ethereum 上链不成功，该如何处理？

上链成功后再写入数据库.

* 如果将交易撤销，Delphy 的市场价格该如何处理？

上链不成功，不写入数据库，将 TXID 和失败状态记录到数据库

* 计算价格和上链操作合在一起作为一个原子操作，一个用户进行操作的时候，其他用户进行等待？Demo 可以暂时使用，但未来产品化如何来提高性能？