**NFT项目的技术细节**

修改日期： 2023-02-09

目录

[概述 3](#_Toc123892030)

[云端数据库 3](#_Toc123892031)

[微信小程序和云函数 4](#_Toc123892032)

[管理后台 5](#_Toc123892033)

# 概述

系统由以下部分构成：

* 云端数据库
* 微信小程序 和 云函数接口（python）
* 管理后台（PHP）
* ~~云存储 和~~ CDN

由于区块链的上链过程是异步的，区块链部分把一个或者多个交易打包为一个区块，保存成链上数据。如果交易量大，就需要等待较长时间才能完成。

区块链的操作，首先是一个数据接收的容器，然后在内部进行操作，最后提供了一个查询操作结果的办法。

以下几个步骤都需要分成 【调用】 和 【确认】 两个步骤，：

创建NFT, 创建用户，NFT转移

【调用】区块链的过程相当于创建了一个任务；

任务的执行情况，需要间隔 30-60秒进行检查，检查的结果如果是【上链成功】，就是完成了【确认】环节。

（如果任务一次检查未完成，需要安排再次检查）

正常情况下，如果前置条件没问题（比如预存款足够，NFT 属主正确），都会成功。

# 云端数据库

云端数据库是数据保存的核心，

保存有活动信息，用户信息，NFT 数据等。

# 微信小程序和云函数

根据入口参数的不同，展示每个活动页面；

活动都是有时间限制的，只有在活动日期范围内才能操作按钮；

用户不能直接看到活动清单随便点击进去，必须等到物料投放之后才能看到入口和参加。

在活动页面展示活动的内容，和用户参与方法，

用户参与过程中，需要提供个人资料，

可以参与活动抽奖，抽奖完成将展示得到的奖品，

如果得到的奖品是 NFT，需要先进行实名认证，

实名认证通过之后，奖品将放入其账号内；

超时的情况下，奖品名额将回收。

所有这些过程，都是云函数调用实现的，

涉及：

微信小程序获取用户信息，

实名认证（使用三要素认证：姓名、身份证号码和手机号）

在链上创建用户，

转移 NFT的所有权给用户

手机验证码验证

业务相关

计划每种接口使用一个云函数入口，分别是：

云函数-微信接口

云函数-实名认证接口

云函数-区块链接口

云函数-短信验证码

云函数-业务相关接口

使用微信小程序认证过程中的系统手机号码，作为所有权的验证。

暂不调用实名认证的接口； 暂不发短信给用户验证手机。

云函数连接到同一个数据库，

session 则使用共享 Memcache 进行保存。

Memcache 1G, 2元/天；

云函数使用腾讯的域名，不必购买数字证书，如果使用自己的域名就需要。

python 云函数相关代码基本无积累，需要较长的开发时间。

微信小程序请求的图片资源，是 CDN 负责分发的，

源文件来自云服务器。

# 管理后台

管理后台，负责发行 NFT，创建活动，填充活动需要的文字和图片资料，用户信息查看等统计和管理性质的操作。

图片资料同步保存到云存储上面，给分发做准备。

管理后台运行在云主机上面。

倾向于管理功能，

发行（大量的）NFT 的时候，先把数据保存到数据库，

然后可以通过 RPC 调用，让云函数激活【发行】任务，通过异步函数完成工作。

（由于不是在自己的服务器上执行，各种执行过程都不可打断，操作务必谨慎）

需要同步开发管理后台。

# 云存储 和 CDN

大量的用户访问静态资源（主要是图片）的时候，需要非常大的带宽和流量，

故需要 CDN 的支持。

目前服务器带宽可以满足 CDN 的回源要求，可以直接对它提供支持，

只要不关闭服务器，可以正常使用。

目前，不需要云存储的支持，

只要处理好 CDN 回源问题，可以简化操作。

# 区块链的相关操作

## 准备过程

分类和 NFT 的准备

创建系统账户；

在系统账户里面，创建需要的分类，完成上链，得到分类的 id；

在分类下面创建本地 NFT，没有哈希因为这个过程中不写入区块链；

接下来，创建系统里面的盲盒，

根据规划，随机完成盲盒填充过程。

## 用户的参与过程

用户进入活动页面，系统得到和保存下面的数据：

来源，openid，用户基本资料，用户手机

接下来用户获得盲盒，打开盲盒；

如果开出来的是 NFT，则要求用户填写实名制信息，

其中手机号码来源于前面的数据不能修改（并不验证，也不允许修改）

用户填写之后，

创建用户链上账号，创建 NFT 给用户，

并且在完成上链操作的时候填充各个链上数据。

也就是说，有多少用户填写完成实名信息

（填写完成即可，没有后续的验证和允许修改步骤），

就创建多少链上数据。

每个用户目前需要三个链上操作：

创建用户，在系统用户下面创建NFT并转让给具体的用户。

三个单独的上链操作，都需要异步等待，同步数据到数据库。

## 用户对NFT的合成

用户可以在满足一定条件的情况下，申请合成多个 NFT 为一个系统内部权益，

合成完成后，相关的NFT将会销毁掉。

每个 NFT 的销毁（和转让）都需要额外的链上操作。

# 数据结构

**Global define**

id: record#

ts\_c: timestamp of create

cls\_id: class#

cls\_id\_hash: hash of class

user\_id: user#

创建 NFT分类和数据的，是内部绑定用户，

数据（区块链名称，一对用户hash）写在配置文件里面。

{ "BSN": ["hash\_dev", "hash\_prod"] }

## NFT分类（BSN） nft\_class

id, ts\_c, nft\_level, inner\_user\_hash, cls\_id\_bsn, cls\_name, op\_id,

内部用户，创建分类在数据库，同步到链上，记录得到的 cls\_id

分类和链上系统用户紧密关联。

type=issue\_class

## NFT清单 nft\_list

id, cls\_id, cls\_id\_bsn, nft\_name, nft\_level, nft\_link\_name, nft\_id\_hash, tx\_hash, tx\_n,

nft\_max, nft\_num,

cur\_user\_id, next\_user\_id, cur\_user\_hash, next\_user\_hash, ts\_m,

op\_type\_now?, op\_id, f\_status,

在本地数据库创建 NFT，供填充盲盒使用。创建的时候保存当前创建总数和顺序号。

用户填写实名信息后，用户上链得到 hash；NFT在系统用户的分类下面创建得到NFT的hash；从系统用户转移到当前用户，得到交易 hash 和链的高度（位置）。

NFT 必须创建在系统用户的分类下面。

用户之间转移，填写当前用户和目标用户；完成链上操作之后，修改当前用户并清空下一个用户字段。

type=mint\_nft, transfer\_nft, burn\_nft/burn\_nft\_batch

status::

数据准备 0，创建中 10，创建完毕 60，

（初次）转让中 80，转让完毕（正常） 100，

赠送进行中 131， 接收方收到 132， 完成链上操作之后回到正常状态 100，

超时退回返回正常状态 100，

销毁中 186，销毁完毕 188

第一次赋予用户的操作很复杂，为了减少链上费用支出只创建有效的NFT并转让：

使用系统用户，创建NFT，等待创建完毕；

进行初次转让，等待转让完毕。

状态码: 0 - 10 -> 60 - 80 -> 100

状态的改变需要生成记录，里面保存每次的 tx\_hash, tx\_n，

供展示和日后检查；

特别情形：

当前用户 id 是 0，且用户hash 非空的时候，是系统操作（各个）区块链（在不同环境下面）的专用用户，

并不是真实的用户账号。

next用户为空，并且状态 100 属于正常拥有状态；如果非空，则是转移进行中；

状态码大于 180 则直接对用户显示为销毁。

## 盲盒类别 box\_class

id, cls\_title, cls\_body, box\_max

## 盲盒清单 box\_list

id, type\_id, type\_title, box\_name, act\_id, cls\_id, nft\_id, user\_id, ts\_get, exp\_time, ...

创建一堆盲盒；

往盲盒里面（多次）填充 NFT，随机填充；

例外情形：可以是空盲盒（0），也可能是积分（-score）；

总体填充量，不允许超过盲盒数量，也就是一个盒子不能放多个东西。

## 盲盒历史 box\_history

id, ts\_c, box\_id, user\_id, evt

记录每个具体盲盒的过程，谁得到了，为啥消失了，又被谁得到了；

是谁得到里面的 NFT 的。

一旦成为 NFT，就没有后续状态了。

## 活动清单 act\_list

id, dt\_from, dt\_to, act\_title, act\_body, box\_type\_id, max\_box, f\_status, ...

记录每个活动的情况，用于对外展示。

状态可以是未开始，进行中，结束。

status::

0=creating, 60=ready, 80=ing, 100=sold out, 109=expired

状态的改变需要生成记录，供日后检查

## NFT的历史记录 nft\_history

id, nft\_id, ts\_c, ...

记录每个具体 NFT 的情况，什么时候创建，什么时候上链，什么时候被谁得到了，什么时候销毁的。

## 链上交易事件记录（BSN） tx\_event

id, ts\_c, tx\_type, tx\_id, tx\_code, tx\_hash, tx\_n, op\_id,

收到事件，对相关的数据表进行变更

\* 创建分类：填充分类的 hash

\* 创建NFT：填充得到 NFT 的 hash 和高度

\* NFT转让：填充得到 NFT 的 hash 和高度；修改NFT 的属性

来自事件订阅，具体需要细化，

记录之后应该触发后续的数据库更新操作。

## 用户清单 user\_list

id, ts\_c, wx\_openid, nick, mobile, avt\_url, nft\_link\_name, acc\_hash, ref\_uid, score,

realname\_id, realname\_name, f\_realname,

用户状态，包括了小程序获取到信息，手机号码；

实名认证信息的填充；

具体使用的哪个链（比如 BSN），上链的账号 hash；

realname::

0=created, 10=filled, 100=checked

## 转赠NFT nft\_giveout

id, user\_id, nft\_link\_name, user\_hash, nft\_id, cls\_id, cls\_id\_bsn, f\_status,

uid\_to, nft\_link\_name2, user\_hash2, tx\_hash, tx\_n,

ts\_c, ts\_fin,

status::

0=created, 10=link visited, 100=got, 120=expired?

## 用户积分历史 score\_history

id, ts\_c, uid, score\_diff, score\_after,

用户积分的获取，积分扣除等历史信息；

## 用户的标签 user\_tag

id, ts\_c, user\_id, act\_id,

用户的标签，可能是多个

## 用户扫码地址 user\_url

id, user\_id, url, ts\_c

每次用户扫码，都需要记录下来，以免多次进入。

## 系统信息 sys\_msg

id, ts\_c, user\_id, msg\_title, msg\_body, f\_read

## 藏品-系统可合成物品 compose\_list

id, comp\_title, comp\_body

## 藏品-依靠哪些东西才能合成 compose\_item

id, comp\_id, cls\_id, cls\_name

合成品依赖于物品分类，与具体的NFT无关

## 藏品-用户合成后的NFR user\_nfr\_list

id, ts\_c, ts\_expire, user\_id, comp\_id, comp\_title, f\_status, verify\_code, ...

合成品具备权利和状态，有效期

## 藏品-合成物品/NFR涉及的NFT user\_nfr\_item

id, ts\_c, user\_id, nft\_id, nft\_hash, nft\_title,

## 系统文档 sys\_doc

id, doc\_short, doc\_title, doc\_body

例如：隐私政策，许可协议等

## 二维码（物料）清单 qr\_list

id, act\_id, qr\_title, qr\_uri

保存生成的二维码，供随时导出和使用。

## 内部用户部分

第一版管理后台简化，不设置权限，

所有后台用户都可以操作，操作透明化。

有操作日志，哪个账号干了啥都可以看到，日志不能修改和删除。

内部用户表

id, iu\_name, pwd\_hash, f\_status, ...

操作日志 op\_log

id, ts\_c, ip\_v4, iu\_id, iu\_name, fun\_name, op\_detail, ...

Append only