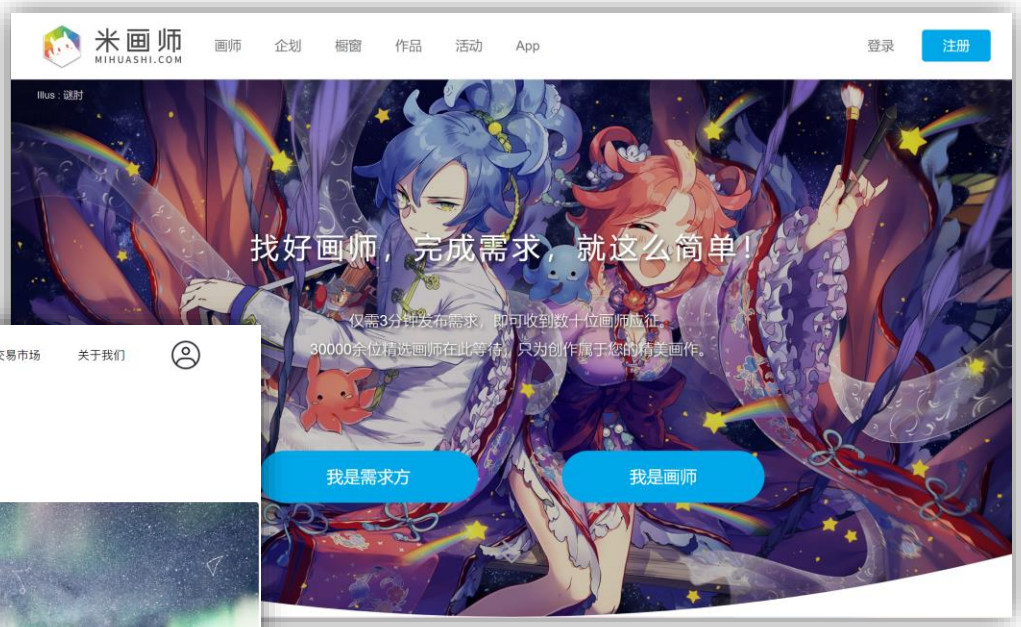
数字画作创作与拍卖平台

——一款基于以太坊的DApp

创意来源

- 米画师



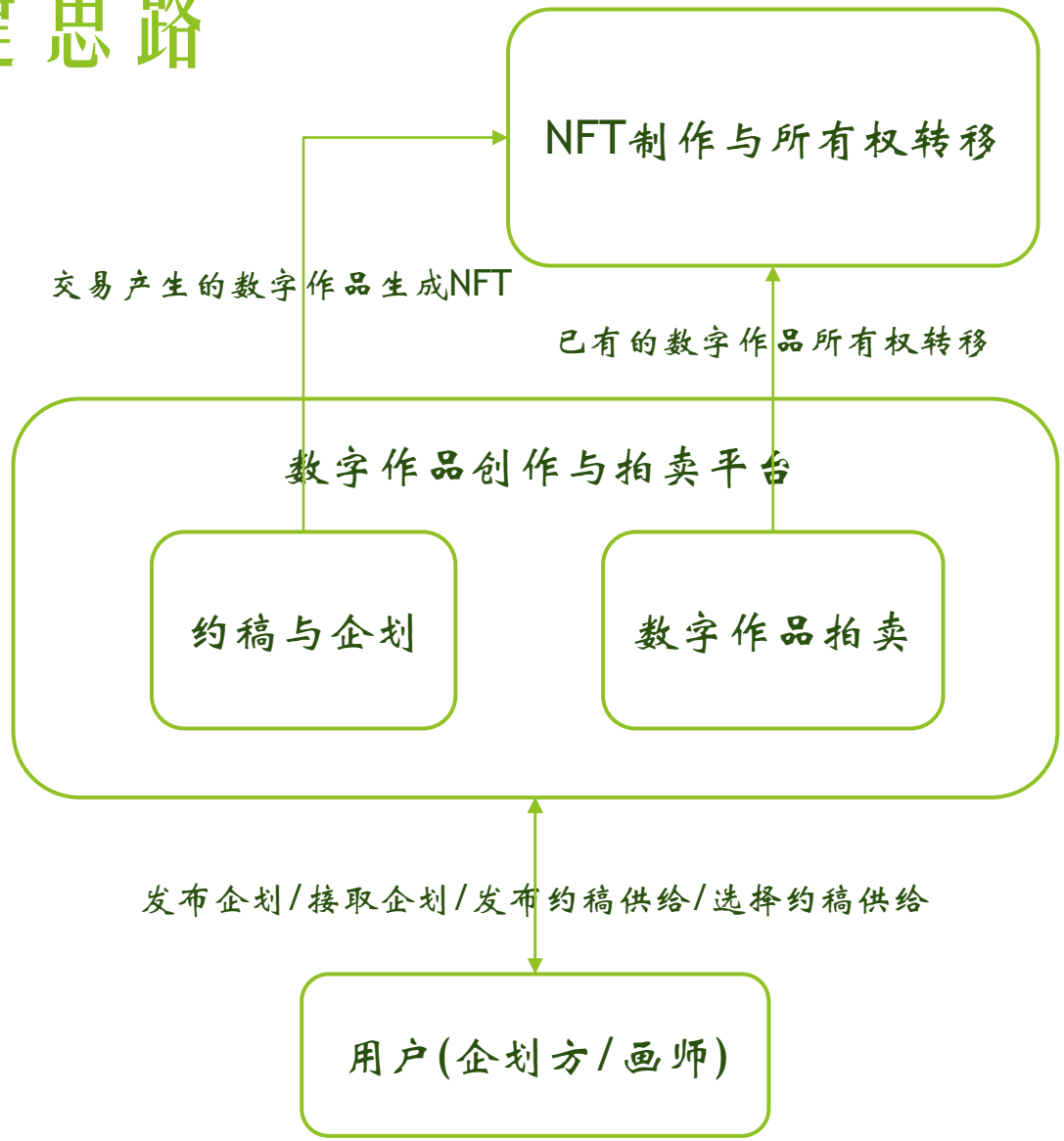
https://www.mihuashi.com/

- NFT中国

https://www.nftcn.com.cn/

现阶段数字作品创作和数字藏品售卖分离，
可以制作一个平台统一二者，
进而实现将数字作品从生产到出售全程在线上完成。

平台构建思路



也可以对外提供交易接口

上半部分交互主要是**合约之间**的调用

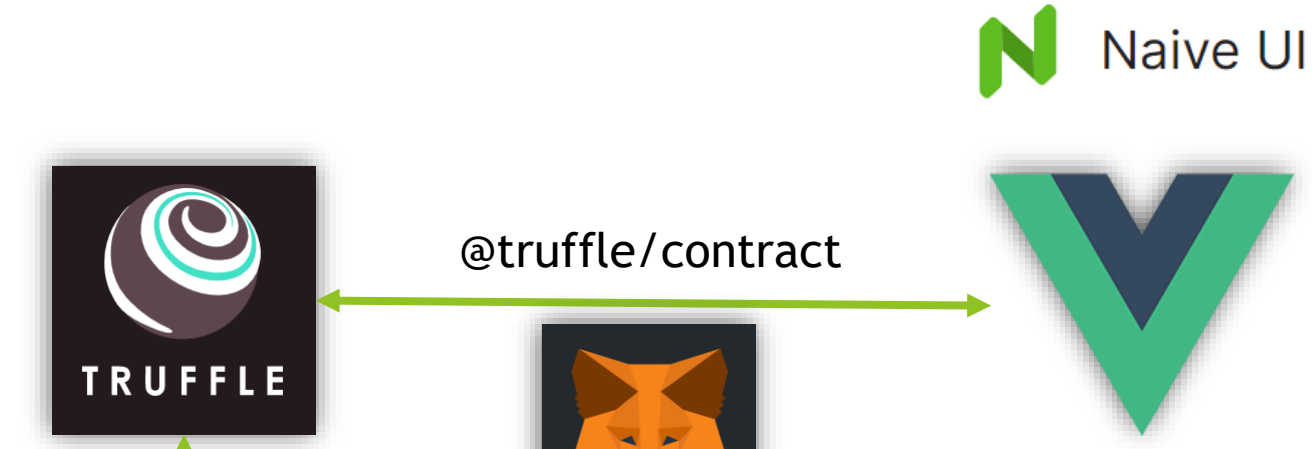
平台主要分成两个部分：**数字作品创作与拍卖平台和NFT制作与所有权转移平台**

下半部分交互主要是**合约与前端**之间调用

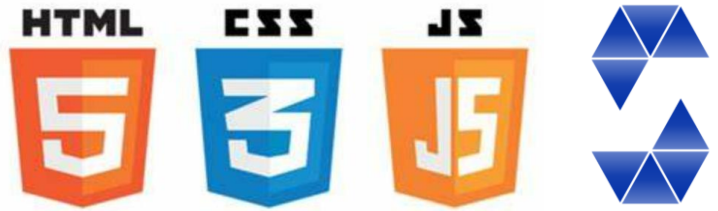
使用区块链作为平台底层架构的合理性

- 数字作品的源头考究和所有权一直是一个不好解决的问题，数字作品又有很大一部分是通过交易产生的，那么通过在数字作品认证NFT时加入交易Hash及某种交易协议（如商用协议[作品所有权归买家，作者不能以个人名义发布和使用当前作品]/平等协议[作品所有权归双方所有，但是双方都不得将其商用]等）这个特点就能很好的追溯作品的产生，也可以更好的确认作品所有权，基于此也可以研发一套带有所有权的NFT模式。
- 齐白石的画作价值很高，那么一些有名气的数字作品作者的画作价值应当也会较高，而数字画作的拥有权现今用NFT来刻画，那么齐白石的作品能被拍卖，数字作品NFT也应当能拿来被拍卖；有时候买家并不希望自己出名，一些收藏家也是这种心态，因此匿名的区块链拍卖交易平台就是非常好的拍卖手段。
- 区块链作为一种特殊的分布式数据库，很好的满足了当今数字作品所缺的可溯源性、不可篡改性、匿名性。

本次项目所需技术



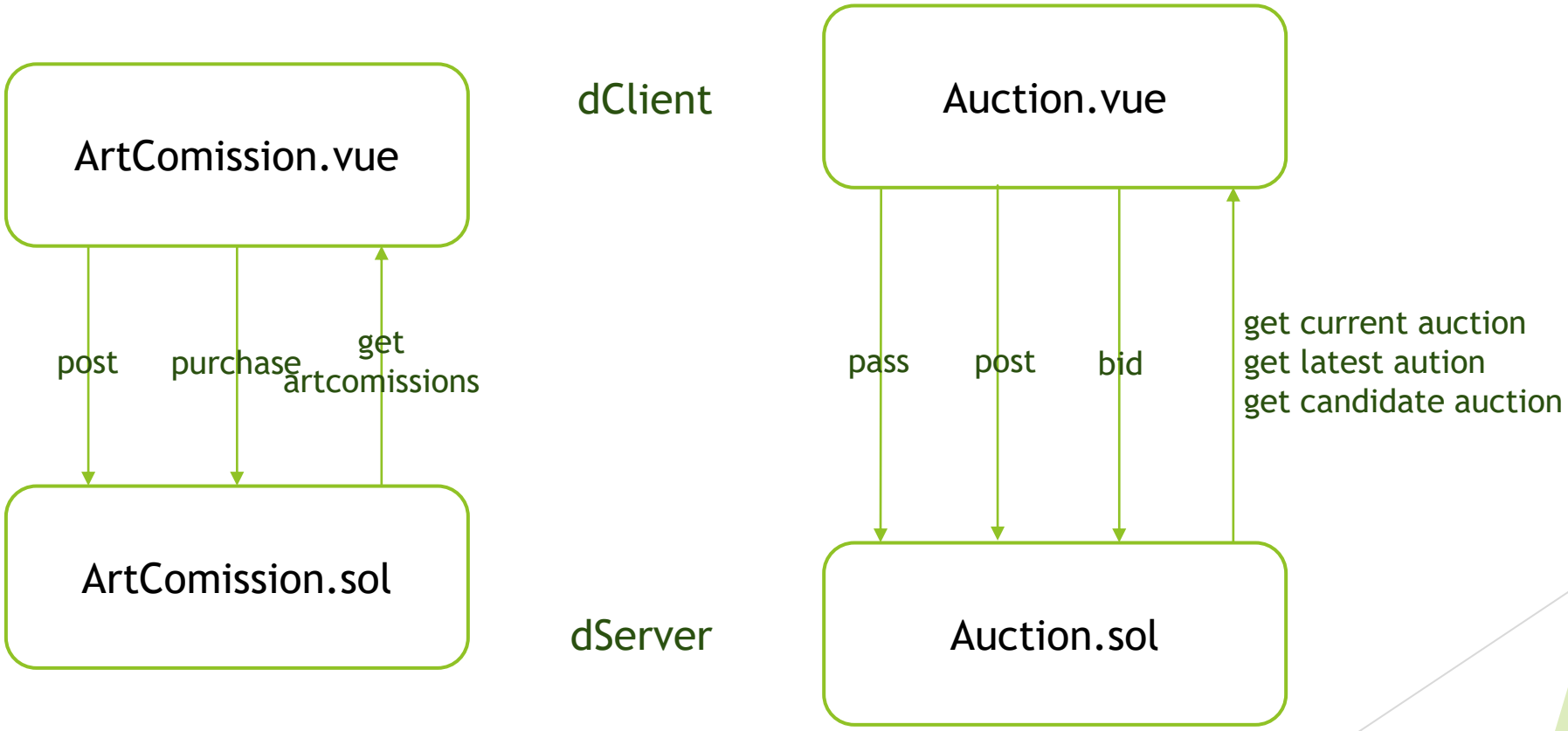
Ganache



HTML/CSS/JS/Solidity

平台具体构建思路

由于时间问题，本次作品仅完成了下半部分：平台与用户之间的交互



项目中遇到的问题

- 1、Vue框架与truffle框架如何交互问题：Vue属于前后端分离式框架，而truffle官方和实验指导中都是一种不分离的结构，导致使用Vue的割裂感很强，在尝试了一些网上所谓的Vue+truffle框架后，总是存在一些问题。因此本项目基本是从源头构建了dServer和dClient结构，dServer即truffle框架及其内部所有文件，dClient即Vue框架及前端所有交互文件，之间的交互通过@truffle/contract完成（此库与实验指导中的truffle-contract并不相同，属于是较新的库，能更好的适配Vue3和一些TS语法），由于我们已经确定使用了MetaMask，所以不需要使用Web3.js了，具体的使用方法如下：（这里将每个vue视图与每个合约对应部署）

```
// 初始化合约以及连接MetaMask
const contract = require('@truffle/contract')
const artifact = require('../assets/contracts/ArtComission.json')
const ArtComissionContract = contract(artifact);
ArtComissionContract.setProvider(window.ethereum)
```

项目中遇到的问题

- 2、使用了@truffle/contract后调用问题：使用@truffle/contract后，由于自身对于js一些语法不是很熟悉，导致调用失败，后面参考@truffle/contract后，在实验指导的调用方法上做了一定的改变，使用了await和async来获得account。（实验指导中的获取方法被浏览器警告会在不久的将来被移除）

具体方法如下：（调用bid函数实例）

```
const purchase = () => {
    AuctionContract.deployed().then(async (instance) => {
        const accounts = await ethereum.request({ method: 'eth_requestAccounts' })
        const account = await accounts[0]
        return instance.bid(bid_price.value, { from: account, value: (bid_price.value * 1e12) })
    }).then((response) => {
        loadPostedData()
    }).catch((err) => {
        alert('error', err.message)
        console.log(err.message);
    });
}
```

项目中遇到的问题

- 3、address payable问题：在之前的实验中msg.sender转address payable使用的方法在我的solidity中报错（或许是版本问题），采用了payable(msg.sender)方法：

```
// address payable _authorAddress = address(uint160(msg.sender));
address payable _authorAddress = payable(msg.sender);
```

- 4、在重复truffle编译出的json文件问题：每次migrate后每个合约会生成json文件，这里注意需要等truffle migrate结束后，将json文件放入vue框架内后，再启动vue框架，否则会因为上一个json文件与当前合约部署的并不是同一个版本问题而前端找不到调用函数的接口（有时候sol文件不做修改重新部署也会存在问题，因此每次需要部署完后再移动json文件后再启动vue项目。或者更改编译后结果输出路径，修改成前端文件对应目录即可）