

02.1ethers区块链开发 (react)

1.NFT

NFT，全称为Non-Fungible Token，指非同质化代币，是用于表示数字资产（包括jpg和视频剪辑形式）的唯一加密货币令牌，可以买卖。

2.Marketplace

我们所说的Marketplace所指是在线市场/交易平台，它的本质是**连接与撮合**，它将希望提供产品或服务的人(卖方/supplier)与希望购买产品或者服务的人(买方/buyer)联系起来，并进行匹配与撮合。

3.开发工具

[node@v16.13.1](#)

[react@17.0.2](#)

[ethers@5.5.4](#)

[ipfs-http-client@56.0.1](#)

[web3modal@1.9.5](#)

[axios@0.25.0](#)

4.部署工作

4.1MetaMask

参考web3.js初级调用智能合约：

<https://note.youdao.com/s/uvB75k7>

4.2创建react项目

- 1 1. 下载安装脚手架
- 2 yarn global add create-react-app
- 3 2. 使用脚手架创建项目
- 4 create-react-app 项目名
- 5 3. 下载相应需求
- 6 yarn add --save ethers@5.5.4
- 7 yarn add web3.storage@3.5.4
- 8 yarn add web3modal@1.9.5
- 9 yarn add axios@0.25.0

ethers:

ethers.js库旨在为以太坊区块链及其生态系统提供一个小而完整的 JavaScript API 库 它最初是与 [ethers.io](#) 一起使用，现在已经扩展为更通用的库。

优点：ethers.js 对比使用 web3.js 代码量更少，接口也更简洁，推荐优先使用 ethers.js 。

包含功能：

- 将私钥保存在客户端，安全 可信赖
- 可支持导入和导出的 JSON钱包文件（Geth，Parity和crowdsale）
- 导入和导出 BIP39 助记词（12个词备份短语）和 HD（分层确定性）钱包（英语，意大利语，日语，韩语，简体中文，繁体中文; 更多即将推出）
- 从任何合同ABI创建JavaScript 元类对象，包括 ABIv2 和 可读的 ABI
- 支持通过 [JSON-RPC](#)，[INFURA](#)，[Etherscan](#) 或 [MetaMask](#) 连接到以太坊节点。
- ENS名称 是“一等公民”；它们可以在任何可以使用以太坊地址的地方使用
- 库 非常小（压缩~88kb;未压缩284kb）
- 功能 完整，满足所有的以太坊相关需求
- 文档全面：[中文文档](#) 及 [documentation](#)
- 大量维护和添加的 测试用例
- 完全支持 TypeScript 准备好，有定义文件和完整的TypeScript源文件
- 宽松的 MIT 协议许可（包括所有依赖）；完全开源可以随意使用

参考网站：<https://learnblockchain.cn/docs/ethers.js/>

web3.storage

当您的项目、网站或应用程序需要文件存储时，Web3.Storage 可以为您服务。借助 Web3.Storage，您可以获得去中心化存储技术的所有优势，以及您在现代开发工作流程中所期望的顺畅体验。使用 Web3.Storage 所需要的只是一个 API 令牌和您的数据。在后台，Web3.Storage 由可证明的[Filecoin](#)存储支持，并使您的用户可以通过公共[IPFS](#)网络访问数据——但是当涉及到构建您的下一个应用程序、服务或网站时，您需要知道的只是Web3.Storage 使基于去中心化技术的构建变得简单。最重要的是，Web3.Storage 是免费的。

参考网站：<https://docs.web3.storage/> & <https://www.npmjs.com/package/web3.storage>

ipfs:

IPFS是星际文件系统，是一个旨在创建持久且分布式存储和共享文件的网络传输协议。它是一种内容可寻址的对等超媒体分发协议。在IPFS网络中的节点将构成一个分布式文件系统。它是一个开放源代码项目，自2014年开始由Protocol Labs在开源社区的帮助下发展。其最初由Juan Benet设计。

IPFS是一个互联网的底层协议，类似HTTP协议，上线时间是2015年的5月5号。它的目标是为了补充甚至是取代目前统治互联网的超文本传输协议（HTTP）。

IPFS是传输协议，不是区块链项目，没有使用任何区块链技术。但是具备区块链去中心化的精神。所以，IPFS没有Token、没有发币、不能挖矿；Filecoin才是Token，挖的是Filecoin。

IPFS目标是打造一个更加开放、快速、安全的互联网，利用分布式哈希表解决数据的传输和定位问题，把点对点的单点传输改变成P2P（多点对多点）的传输，其中存储数据的结构是哈希链。

参考网站: <https://www.zhihu.com/question/447408500/answer/1782830492>

web3modal:

Web3Modal 从前叫做 Web3Connect，可帮助开发人员通过简单的配置，为其应用添加钱包支持。通过 Web3Modal 库，支持 Metamask、Dapper、WalletConnect 及其他 Web3 浏览器等，还可以轻松配置库以支持 Fortmatic 和 Portis等。

参考网站: <https://github.com/Web3Modal/web3modal>

5.项目代码展示

5.1CreateItem Page:

完成功能:

- a. 用户能够上传和保存文件到IPFS;
- b. 用户能够创建一个新的独特的 NFT，并同时设置作品的元数据

```
1 import { useState, useCallback } from "react";
2 import { ethers } from "ethers";
3 import { Web3Storage } from "web3.storage/dist/bundle.esm.min.js";
4 import Web3Modal from "web3modal";
5 import "../assets/styles/CreateItem.css";
6 import NFT from "../contracts/NFT_ABI.json"; // 引入ABI
7 const nftaddress = "0x90fe47327d2e2851fD4eFEd32bc64c4b14CB1D29"; // 定义合约地址
8
9 export default function CreateItem() {
10   // 定义数据
11   const [fileUrl, setFileUrl] = useState(null);
12   const [formInput, updateFormInput] = useState({
13     name: "",
14     description: "",
15   });
16
17   // 让ref为uploaderd的input框变成上传文件夹
18   const uploader = useCallback((node) => {
19     if (node) {
20       node.setAttribute("webkitdirectory", "");
21       node.setAttribute("directory", "");
22       node.setAttribute("multiple", "");
```

```
23     }
24   }, []);
25
26   // 通过表单的change事件达成数据双向绑定
27   async function onChange(e) {
28     // 获取input上传的文件
29     let file = e.target.files;
30     const data = JSON.stringify({
31       name: formInputFree.name,
32       description: formInputFree.description,
33     });
34     //将文字转为上传的格式
35     var blob = new Blob([data]);
36     const newFile = new File([blob], "test.json");
37     console.log(newFile);
38     //将文字追加进数组，传入ipfs上传函数
39     file = [...file, newFile];
40     console.log(file);
41     try {
42       // 赋值
43       setFileUrlFree(file);
44     } catch (error) {
45       console.log("Error uploading file: ", error);
46     }
47   }
48   // 上传到IPFS并完成铸造
49   async function createMarket() {
50     // 1.发送到IPFS上
51     // 定义上传的apiToken
52     let apiToken =
53       "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkaWQ6ZXRocjoweDE5N2VEYTQ5QjQyRmVjRjI2QzBhNWMA0ThmNUYzNzVGNDU1Y2U2MWEiLCJpc3MiOiJ3ZWlzeLXN0b3JhZ2UiLCJpYXQiOiJlE2NDU1ODEwMDQ4NjAsIm5hbWUiOiJkZXZyYmJAYmIn.s9DZmDbB1VasuMmI50RzfFavxwIachm0XuELGz5RZY4";
54     // 使用apiToken构造Web3Storage
55     const client = new Web3Storage({ token: apiToken });
56     // 将文件打包CAR中并发送到web3.storage
57     const rootCid = await client.put(fileUrlFree, {
58       name: "test1",
59       maxRetries: 3,
60       wrapWithDirectory: true,
```

```

61     });
62     try {
63         alert("Upload to IPFs succeeded !!!");
64         // 2.拼接从ipfs上拿到的url
65         let url = `https://ipfs.io/ipfs/${rootCid}`;
66         console.log(url);
67         // 3.将url传入,并调用智能合约开始铸造NFT
68         createSaleFree(url);
69     } catch (error) {
70         console.log("Error uploading file: ", error);
71     }
72 }
73 // 铸造NFT
74 async function createSale(url) {
75     const web3Modal = new Web3Modal();
76     const connection = await web3Modal.connect();
77     const provider = new ethers.providers.Web3Provider(connection);
78     const signer = provider.getSigner();
79     // 4.接下来, 创建项目
80     let contract = new ethers.Contract(nftaddress, NFT.abi, signer);
81     let transaction = await contract.mintOneToken(url);
82     if (transaction) {
83         await transaction.wait();
84         alert("Project created successfully !!!");
85         await window.location.replace("/creatorDashboard");
86     }
87 }
88 // go back
89 async function goBack() {
90     await window.history.back(-1);
91 }
92 return (
93     <div className="createItem">
94         <div className="createItem-Head">
95             <h2>Create Item</h2>
96             <button onClick={goBack}>Go Back!</button>
97         </div>
98         <div className="createItem-from-box">
99             <h3>Free casting</h3>
100             <input

```

```

101         style={{ textIndent: "0.6rem" }}
102         placeholder="Asset Name"
103         onChange={(e) =>
104             updateFormInputFree({ ...formInputFree, name: e.target.value })
105         }
106     />
107     <textarea
108         style={{ textIndent: "0.6rem" }}
109         placeholder="Asset Description"
110         onChange={(e) =>
111             updateFormInputFree({
112                 ...formInputFree,
113                 description: e.target.value,
114             })
115         }
116     />
117     { /* 上传文件夹 */ }
118     <input
119         type="file"
120         ref={uploader}
121         onChange={(e) => onChange(e, "free")}
122     />
123     <button onClick={createFreeItem} className="createItem-from-btn">
124         Create Free Digital Asset
125     </button>
126 </div>
127 </div>
128 );
129 }

```

5.2 Sell Digital Asset Page:

完成功能:

- a. 作为应用程序的主要入口;
- b. 作为查询呈现待售数字资产的页面(MarketPlace上);
- c. 可以购买想要的作品

```

1 import { ethers } from "ethers";
2 import { useEffect, useState } from "react";
3 import axios from "axios";
4 import Web3Modal from "web3modal";

```

```
5 import "../assets/styles/SellDigitalAsset.css";
6 import Model from "../Model"; // 引入3D模型
7 // 引入ABI
8 import NFT from "../contracts/NFT_ABI.json";
9 import Market from "../contracts/Mkt_ABI.json";
10 // 定义合约地址
11 const nftaddress = "0x90fe47327d2e2851fD4eFEd32bc64c4b14CB1D29";
12 const nftmarketaddress = "0xFC2Ea5A1F3Bed1B545A6be182BF52C20B5e45921";
13 // 定义需要链接的网络(此时为内网测试)
14 let rpcEndpoint = "http://192.168.11.120:8545";
15 // 判断网络并作出相应操作
16 if (process.env.NEXT_PUBLIC_WORKSPACE_URL) {
17   rpcEndpoint = process.env.NEXT_PUBLIC_WORKSPACE_URL;
18 }
19 export default function Home() {
20   // 定义数据
21   const [nfts, setNfts] = useState([]);
22   const [loadingState, setLoadingState] = useState("not-loaded");
23   useEffect(() => {
24     loadNFTs();
25   }, []);
26   async function loadNFTs() {
27     // 连接网络并定义合约
28     const provider = new ethers.providers.JsonRpcProvider(rpcEndpoint);
29     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
30     const marketContract = new ethers.Contract(
31       nftmarketaddress,
32       Market.abi,
33       provider
34     );
35     // 查找marketContract合约上的所有项目
36     const data = await marketContract.fetchMarketItems();
37     const items = await Promise.all(
38       // 循环遍历所有拿到的数据
39       data.map(async (i) => {
40         const tokenUri = await tokenContract.tokenURI(i.tokenId);
41         // 处理数据
42         let mtl = `${tokenUri}/1.mtl`;
43         let obj = `${tokenUri}/1.obj`;
44         let txt = `${tokenUri}/Introduction.txt`;
```

```

45     const meta = await axios.get(`${tokenUri}/test.json`);
46     let price = ethers.utils.formatUnits(i.price.toString(), "ether");
47     let item = {
48         price,
49         itemId: i.itemId.toNumber(),
50         seller: i.seller,
51         owner: i.owner,
52         addr: tokenUri,
53         mtl,
54         obj,
55         txt,
56         name: meta.data.name,
57         description: meta.data.description,
58     };
59     return item;
60 })
61 );
62 setNfts(items);
63 setLoadingState("loaded");
64 }
65 // buyBtn
66 async function buyNft(nft) {
67     const web3Modal = new Web3Modal();
68     const connection = await web3Modal.connect();
69     const provider = new ethers.providers.Web3Provider(connection);
70     const signer = provider.getSigner();
71     const contract = new ethers.Contract(nftmarketaddress, Market.abi, signer);
72     const price = ethers.utils.parseUnits(nft.price.toString(), "ether");
73     console.log(nft.itemId, nftaddress);
74     const transaction = await contract.createMarketSale(
75         nftaddress,
76         nft.itemId,
77         {
78             value: price,
79         }
80     );
81     await transaction.wait();
82     // 重绘
83     loadNFTs();
84 }

```



```

85 // go back
86 async function goBack() {
87     await window.history.back(-1);
88 }
89 if (loadingState === "loaded" && !nfts.length)
90     return <h1>No items in marketplace</h1>;
91 return (
92     <div className="SellDigitalAsset">
93         <div className="SellDigitalAsset-Box" style={{ maxWidth: "1600px" }}>
94             <div className="SellDigitalAsset-Box-Head">
95                 <h2>Sell Digital Asset</h2>
96                 <button onClick={goBack}>Go Back!</button>
97             </div>
98             <div className="SellDigitalAsset-Box-content">
99                 {nfts.map((nft, i) => (
100                     <div key={i} className="SellDigitalAsset-Box-content-item">
101                         <div
102                             className="model"
103                             style={{ width: "16rem", height: "14rem", borderRadius: "0.5rem 0.5rem 0
104 0", }}>
105                             <Model className="modelchild" data={nft} />
106                         </div>
107                         <div className="SellDigitalAsset-Box-content-item-info">
108                             <p className="SellDigitalAsset-Box-content-item-info-name">
109                                 {nft.name}
110                             </p>
111                             <div style={{ height: "4rem", overflow: "hidden" }}>
112                                 <p style={{ paddingLeft: "1rem", fontSize: "14px" }} >
113                                     {nft.description}
114                                 </p>
115                             </div>
116                         </div>
117                         <div className="SellDigitalAsset-Box-content-item-Btn">
118                             <p style={{
119                                 textIndent: "1rem",
120                                 fontWeight: "bold",
121                                 fontSize: "18px", }} >
122                                 {nft.price} ETH
123                             </p>
124                             <button onClick={() => buyNft(nft)}>Buy</button>

```

```
124         </div>
125     </div>
126     )})
127 </div>
128 </div>
129 </div>
130 );
131 }
132
133
```

5.3 MyDigitalAssets Page

完成功能：

- a. 返回用户拥有的作品并展示a

```
1  import { ethers } from "ethers";
2  import { useEffect, useState } from "react";
3  import axios from "axios";
4  import Web3Modal from "web3modal";
5  import "../assets/styles/MyDigitalAssets.css";
6  import Model from "../Model.js"; // 引入3D模型
7  // 引入ABI
8  import NFT from "../contracts/NFT_ABI.json";
9  import Market from "../contracts/Mkt_ABI.json";
10 // 定义合约地址
11 const nftaddress = "0x90fe47327d2e2851fD4eFEd32bc64c4b14CB1D29";
12 const nftmarketaddress = "0xFC2Ea5A1F3Bed1B545A6be182BF52C20B5e45921";
13 export default function MyAssets() {
14     // 定义数据
15     const [nfts, setNfts] = useState([]);
16     const [loadingState, setLoadingState] = useState("not-loaded");
17     useEffect(() => {
18         loadNFTs();
19     }, []);
20     async function loadNFTs() {
21         const web3Modal = new Web3Modal({
22             network: "mainnet",
23             cacheProvider: true,
24         });
```

```
25     const connection = await web3Modal.connect();
26     const provider = new ethers.providers.Web3Provider(connection);
27     const signer = provider.getSigner();
28     const marketContract = new ethers.Contract(
29         nftmarketaddress,
30         Market.abi,
31         signer
32     );
33     // 定义合约
34     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
35     // 查找marketContract合约上的自己创建的项目
36     const data = await marketContract.fetchMyNFTs();
37     const items = await Promise.all(
38         data.map(async (i) => {
39             // 调用NFT合约获取tokenURI
40             const tokenUri = await tokenContract.tokenURI(i.tokenId);
41             // 处理数据
42             let mtl = `${tokenUri}/1.mtl`;
43             let obj = `${tokenUri}/1.obj`;
44             const meta = await axios.get(`${tokenUri}/test.json`);
45             let price = ethers.utils.formatUnits(i.price.toString(), "ether");
46             let item = {
47                 price,
48                 tokenId: i.tokenId.toNumber(),
49                 seller: i.seller,
50                 owner: i.owner,
51                 addr: tokenUri,
52                 mtl,
53                 obj,
54                 name: meta.data.name,
55                 description: meta.data.description,
56             };
57             return item;
58         })
59     );
60     setNfts(items);
61     setLoadingState("loaded");
62 }
63 // 通过判断有无数据，选择显示
64 if (loadingState === "loaded" && !nfts.length)
```

```

65     return <h1 className="py-10 px-20 text-3xl">No assets owned</h1>;
66
67 // go back
68 async function goBack() {
69     await window.history.back(-1);
70 }
71
72 return (
73     <div className="MyDigitalAssets">
74         <div className="MyDigitalAssets-" style={{ padding: "0.5rem" }}>
75             <div className="MyDigitalAssets-Head">
76                 <h2>My Digital Assets</h2>
77                 <button onClick={goBack}>Go Back!</button>
78             </div>
79             <div className="MyDigitalAssets-Box">
80                 {nfts.map((nft, i) => (
81                     <div key={i} className="MyDigitalAssets-Box-item">
82                         <div
83                             className="model"
84                             style={{
85                                 width: "16rem",
86                                 height: "14rem",
87                                 borderRadius: "0.5rem 0.5rem 0 0",
88                             }}
89                         >
90                             <Model className="modelchild" data={nft} />
91                         </div>
92                         <div className="MyDigitalAssets-Box-item-info">
93                             <h3>{nft.name}</h3>
94                             <p>{nft.description}</p>
95                         </div>
96                         <div className="MyDigitalAssets-Box-item-price">
97                             <p className="text-2xl font-bold text-white">
98                                 Price - {nft.price} Eth
99                             </p>
100                         </div>
101                     </div>
102                 )))
103             </div>
104         </div>

```

```
105     </div>
106   );
107 }
```

5.4 CreatorDashboard Page

完成功能：

- 展示该用户所创建的所有作品；
- 可以把创建的作品设置价格并上架到Maketplace上面；
- 展示该用户所出售的所有作品；
- 展示该用户所有的作品(创建+出售)；
- 展示该用户所上架的作品，并可以从Maketplace上面下架到自己仓库；

```
1  import { ethers } from "ethers";
2  import { useEffect, useState } from "react";
3  import axios from "axios";
4  import Web3Modal from "web3modal";
5  import Model from "../Model.js"; // 引入3D模型
6  import "../assets/styles/CreatorDashboard.css";
7
8  // 引入ABI
9  import NFT from "../contracts/NFT_ABI.json";
10 import Market from "../contracts/Mkt_ABI.json";
11 // 定义合约地址
12 const nftaddress = "0x90fe47327d2e2851fD4eFEd32bc64c4b14CB1D29";
13 const nftmarketaddress = "0xFC2Ea5A1F3Bed1B545A6be182BF52C20B5e45921";
14
15 export default function CreatorDashboard() {
16   // 定义数据
17   const [nfts, setNfts] = useState([]);
18   const [sold, setSold] = useState([]);
19   const [solds, setSolds] = useState([]);
20   const [notSold, setNotSold] = useState([]);
21   const [tokenUrii, setTokenUri] = useState(""); // 我创建的NFT(用作判断的时候)
22   const [price, setPrice] = useState({
23     price: "",
24   });
25   const [loadingState, setLoadingState] = useState("not-loaded");
26   useEffect(() => {
27     loadNFTs();
```

```
28     loadMarkets();
29 }, []);
30 async function loadNFTs() {
31     const web3Modal = new Web3Modal({
32         network: "mainnet",
33         cacheProvider: true,
34     });
35     const connection = await web3Modal.connect();
36     const provider = new ethers.providers.Web3Provider(connection);
37     const signer = provider.getSigner();
38     const nftContract = new ethers.Contract(nftaddress, NFT.abi, signer);
39     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
40     // 调用NFT合约查询自己创建的NFT
41     const data = await nftContract.fetchMyNFTs();
42     const items = await Promise.all(
43         data.map(async (i) => {
44             if (i[4] !== "") {
45                 const tokenUri = await tokenContract.tokenURI(i.tokenId);
46                 // 赋值，为了给空盒子做判断
47                 setTokenUri(tokenUri);
48                 let mtl = `${tokenUri}/1.mtl`;
49                 let obj = `${tokenUri}/1.obj`;
50                 const meta = await axios.get(`${tokenUri}/test.json`);
51                 let item = {
52                     itemId: i.tokenId.toNumber(),
53                     tokenId: i.tokenId.toNumber(),
54                     seller: i.seller,
55                     owner: i.owner,
56                     sold: i.sold,
57                     addr: tokenUri,
58                     mtl,
59                     obj,
60                 };
61                 return item;
62             } else {
63                 let item = {
64                     itemId: i.tokenId.toNumber(),
65                     tokenId: i.tokenId.toNumber(),
66                     seller: i.seller,
67                     owner: i.owner,
```

```
68         sold: i.sold,
69     };
70     return item;
71 }
72 })
73 );
74 /* create a filtered array of created items */
75 setNfts(items);
76 setLoadingState("loaded");
77 }
78 // 查询自己所出售和未出售的
79 async function loadMarkets() {
80     const web3Modal = new Web3Modal({
81         network: "mainnet",
82         cacheProvider: true,
83     });
84     const connection = await web3Modal.connect();
85     const provider = new ethers.providers.Web3Provider(connection);
86     const signer = provider.getSigner();
87     const marketContract = new ethers.Contract(
88         nftmarketaddress,
89         Market.abi,
90         signer
91     );
92     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
93     const tokendata = await marketContract.fetchItemsCreated();
94     const tokenitems = await Promise.all(
95         tokendata.map(async (i) => {
96             if (i == "") {
97                 alert("not-loaded");
98             } else {
99                 const tokenUri = await tokenContract.tokenURI(i.tokenId);
100                 let mtl = `${tokenUri}/1.mtl`;
101                 let obj = `${tokenUri}/1.obj`;
102                 const meta = await axios.get(`${tokenUri}/test.json`);
103                 let price = ethers.utils.formatUnits(i.price.toString(), "ether");
104                 let item = {
105                     price,
106                     itemId: i.itemId.toNumber(),
107                     tokenId: i.tokenId.toNumber(),
```

```
108         seller: i.seller,
109         owner: i.owner,
110         sold: i.sold,
111         addr: tokenUri,
112         mtl,
113         obj,
114     };
115     return item;
116 }
117 })
118 );
119 /* create a filtered array of sold items and all items */
120 // 筛选已售出的
121 const soldItems = tokenitems.filter((i) => i.sold);
122 console.log(soldItems);
123 // 筛选未出售的(可下架)
124 const notSoldItems = tokenitems.filter((i) => !i.sold);
125 // sold(已售出的)
126 setSold(soldItems);
127 // notSold(未出售的)
128 setNotSold(notSoldItems);
129 // all(创建的+已售出的(下架等同于出售))
130 setSolds(tokenitems);
131 setLoadingState("loaded");
132 }
133 // 通过判断有无数据, 选择显示
134 if (
135     loadingState === "loaded" &&
136     !nfts.length &&
137     loadingState === "loaded" &&
138     !sold.length &&
139     loadingState === "loaded" &&
140     !solds.length
141 )
142     return <h1 className="py-10 px-20 text-3xl">No assets created</h1>;
143
144 // go back
145 async function goBack() {
146     await window.history.back(-1);
147 }
```



```
148
149 // put on the marketplace
150 var _price = price;
151 async function putOn(nft) {
152     const web3Modal = new Web3Modal();
153     const connection = await web3Modal.connect();
154     const provider = new ethers.providers.Web3Provider(connection);
155     const signer = provider.getSigner();
156     const price = ethers.utils.parseUnits(_price, "ether");
157     let contract = new ethers.Contract(nftmarketaddress, Market.abi, signer);
158     let listingPrice = await contract.getListingPrice();
159     listingPrice = listingPrice.toString();
160     let transaction = await contract.createMarketItem(
161         nftaddress,
162         nft.tokenId,
163         price,
164         {
165             value: listingPrice,
166         }
167     );
168     await transaction.wait();
169     await window.location.replace("/sellDigitalAsset");// 跳转到sellDigitalAsset页面
170 }
171 // 获取价格的change事件
172 async function change(e) {
173     console.log(e.target.value);
174     setPrice(e.target.value);
175 }
176 // 下架
177 async function offShelf(nft) {
178     console.log(nft);
179     const web3Modal = new Web3Modal({
180         network: "mainnet",
181         cacheProvider: true,
182     });
183     const connection = await web3Modal.connect();
184     const provider = new ethers.providers.Web3Provider(connection);
185     const signer = provider.getSigner();
186     const marketContract = new ethers.Contract(
187         nftmarketaddress,
```

```

188     Market.abi,
189     signer
190   );
191   const tokendatas = await marketContract.recallSellingItem(
192     nftaddress,
193     nft.itemId
194   );
195 }
196
197 // 根据不同条件渲染不同组件
198 function renderDiv() {
199   const isTokenUrii = tokenUrii;
200   if (isTokenUrii) {
201     return (
202       <div style={{ display: "flex", flexWrap: "wrap" }}>
203         {nfts.map((nft, i) => (
204           <div key={i} className="CreatorDashboard-items-Created">
205             {/* <img
206               style={{
207                 width: "16rem",
208                 height: "17rem",
209                 borderRadius: "0.5rem 0.5rem 0 0",
210               }}
211               src={nft.image}
212             /> */}
213             <div
214               className="model"
215               style={{
216                 width: "16rem",
217                 height: "17rem",
218                 borderRadius: "0.5rem 0.5rem 0 0",
219               }}
220             >
221               <Model className="modelchild" data={nft} />
222             </div>
223             <div
224               style={{
225                 backgroundColor: "#000",
226                 color: "#fff",
227                 borderRadius: "0 0 0.5rem 0.5rem",

```

```

228         flex: "1",
229     }}
230 >
231     <p
232         style={{ textAlign: "center", margin: "0.8rem 0 0.8rem 0" }}
233         className="text-2xl font-bold text-white"
234     >
235         Price -
236         <input style={{ width: "3rem" }} onChange={change}></input>
237         &nbsp;Eth
238     </p>
239     <p style={{ display: "flex", justifyContent: "center" }}>
240         <button
241             style={{
242                 cursor: "pointer",
243                 border: "0",
244                 backgroundColor: "#80c342",
245                 color: "#fff",
246                 height: "1.5rem",
247                 borderRadius: "1.5rem",
248                 width: "8rem",
249             }}
250             onClick={putOn.bind(this, nft)}
251         >
252             Put on the shelf
253         </button>
254     </p>
255 </div>
256 </div>
257     )}}
258 </div>
259 );
260 } else {
261     return <div></div>;
262 }
263 }
264
265 return (
266     <div>
267         { /* 展示自己所创建的项目 */ }

```

```

268     <div className="p-4">
269         <div className="CreatorDashboard-Head">
270             <h2 className="text-2xl py-2">Items Created</h2>
271             <button onClick={goBack}>Go Back!</button>
272         </div>
273         <div>{renderDiv()}</div>
274     </div>
275     { /* 展示已售出的项目 */ }
276     <div className="px-4">
277         {Boolean(sold.length) && (
278             <div className="CreatorDashboard">
279                 <h2 style={{ margin: "1rem" }}>Items sold</h2>
280                 <div style={{ display: "flex", flexWrap: "wrap" }}>
281                     {sold.map((nft, i) => (
282                         <div key={i} className="CreatorDashboard-items-sold">
283                             <div
284                                 className="model"
285                                 style={{
286                                     width: "16rem",
287                                     height: "17rem",
288                                     borderRadius: "0.5rem 0.5rem 0 0",
289                                 }}
290                             >
291                                 <Model className="modelchild" data={nft} />
292                             </div>
293                             <div style={{ flex: "1" }}>
294                                 <p style={{ textAlign: "center", paddingTop: "1rem" }}>
295                                     Price - {nft.price} Eth
296                                 </p>
297                             </div>
298                         </div>
299                     )))
300                 </div>
301             </div>
302         )}
303     </div>
304     { /* 展示未出售的项目 */ }
305     <div className="px-4">
306         {Boolean(notSold.length) && (
307             <div className="CreatorDashboard">

```

```

308     <h2 style={{ margin: "1rem" }}>Items notSold</h2>
309     <div style={{ display: "flex", flexWrap: "wrap" }}>
310         {notSold.map((nft, i) => (
311             <div key={i} className="CreatorDashboard-items-notSold">
312                 <div
313                     className="model"
314                     style={{
315                         width: "16rem",
316                         height: "17rem",
317                         borderRadius: "0.5rem 0.5rem 0 0",
318                     }}
319                 >
320                     <Model className="modelchild" data={nft} />
321                 </div>
322                 <div style={{ flex: "1", textAlign: "center" }}>
323                     <p style={{ textAlign: "center", paddingTop: "1rem" }}>
324                         Price - {nft.price} Eth
325                     </p>
326                     <button
327                         style={{
328                             backgroundColor: "#f60",
329                             color: "#fff",
330                             width: "5rem",
331                             height: "1.5rem",
332                             borderRadius: "1rem",
333                             marginTop: "0.2rem",
334                         }}
335                         onClick={offShelf.bind(this, nft)}
336                     >
337                         下架
338                     </button>
339                 </div>
340             </div>
341         )))
342     </div>
343 </div>
344 )}
345 </div>
346 { /* 展示所有的项目(创建的+已出售的(下架等同于出售)) */}
347 <div className="px-4" style={{ marginBottom: "2rem" }}>

```

```

348     {Boolean(solds.length) && (
349         <div className="CreatorDashboard">
350             <h2 style={{ margin: "1rem" }}>All Items</h2>
351             <div style={{ display: "flex", flexWrap: "wrap" }}>
352                 {solds.map((nft, i) => (
353                     <div key={i} className="CreatorDashboard-items-all">
354                         <div
355                             className="model"
356                             style={{
357                                 width: "16rem",
358                                 height: "17rem",
359                                 borderRadius: "0.5rem 0.5rem 0 0",
360                             }}
361                         >
362                             <Model className="modelchild" data={nft} />
363                         </div>
364                         <div style={{ flex: "1" }}>
365                             <p style={{ textAlign: "center", paddingTop: "1rem" }}>
366                                 Price - {nft.price} Eth
367                             </p>
368                         </div>
369                     </div>
370                 )))
371             </div>
372         </div>
373     )}
374 </div>
375 </div>
376 );
377 }

```

5.5 Pledge Page

完成功能：

- a. 将指定NFT质押到HoldAssets合约中

```

1  import React from "react";
2  import { ethers } from "ethers";
3  import { useEffect, useState } from "react";
4  import axios from "axios";
5  import Web3Modal from "web3modal";

```

```
6 import Model from "./Model.js";
7 import "../assets/styles/Pledge.css";
8 // 引入ABI
9 import NFT from "../contracts/NFT_ABI.json";
10 import HoldAssets from "../contracts/HoldAssets_ABI.json";
11 // 定义合约地址
12 const nftaddress = "0x90fe47327d2e2851fD4eFEed32bc64c4b14CB1D29";
13 const holdAssetsaddress = "0xbD818FA996C19F64841390De77Ee2b43E0d78F0d";
14 export default function Pledge() {
15   const [nfts, setNfts] = useState([]); // 查询到自己所创建的NFT
16   const [pNfts, setPnfts] = useState([]); // 查询到自己所质押的NFT
17   const [loadingState, setLoadingState] = useState("not-loaded");
18   useEffect(() => {
19     loadNFTs();
20     pledgedNFTs();
21   }, []);
22
23   // 查询到自己所创建的NFT
24   async function loadNFTs() {
25     const web3Modal = new Web3Modal({
26       network: "mainnet",
27       cacheProvider: true,
28     });
29     const connection = await web3Modal.connect();
30     const provider = new ethers.providers.Web3Provider(connection);
31     const signer = provider.getSigner();
32
33
34     const nftContract = new ethers.Contract(nftaddress, NFT.abi, signer);
35     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
36     const data = await nftContract.fetchMyNFTs();
37     console.log(data);
38     const items = await Promise.all(
39       data.map(async (i) => {
40         const tokenUri = await tokenContract.tokenURI(i.tokenId);
41         let mtl = `${tokenUri}/1.mtl`;
42         let obj = `${tokenUri}/1.obj`;
43         const meta = await axios.get(`${tokenUri}/test.json`);
44         let item = {
45           tokenId: i.tokenId.toNumber(),
```

```

46         seller: i.seller,
47         owner: i.owner,
48         addr: tokenUri,
49         mtl,
50         obj,
51         name: meta.data.name,
52         description: meta.data.description,
53     });
54     console.log(item);
55     return item;
56 })
57 );
58 setNfts(items);
59 setLoadingState("loaded");
60 }
61
62 if (loadingState === "loaded" && !nfts.length)
63     return <h1 className="py-10 px-20 text-3xl">No assets owned</h1>;
64
65 // go back
66 async function goBack() {
67     await window.history.back(-1);
68 }
69
70 // 质押
71 async function pledge(nft) {
72     console.log(nft);
73     const web3Modal = new Web3Modal({
74         network: "mainnet",
75         cacheProvider: true,
76     });
77     const connection = await web3Modal.connect();
78     const provider = new ethers.providers.Web3Provider(connection);
79     const signer = provider.getSigner();
80     const tokenContract = new ethers.Contract(nftaddress, NFT.abi, provider);
81     const nftContract = new ethers.Contract(nftaddress, NFT.abi, signer);
82     // 调起合约执行质押对应的NFT
83     const data = await nftContract.pledgeToNFT(holdAssetsaddress, nft.tokenId);
84     if (data) {

```



```

125         <p>{nft.description}</p>
126     </div>
127     <div className="Pledge-Box-item-price">
128         <p className="text-2xl font-bold text-white">
129             Price - {nft.price} Eth
130         </p>
131         <p>
132             <button onClick={pledge.bind(this, nft)}>质押</button>
133         </p>
134     </div>
135 </div>
136     )})
137 </div>
138 </div>
139 </div>
140 );
141 }
142
143

```

5.6 ReleasePledge Page

完成功能：

- a. 将指定NFT解除质押到原合约中

```

1  import React from "react";
2  import { useEffect, useState } from "react";
3  import { ethers } from "ethers";
4  import Web3Modal from "web3modal";
5  import "../assets/styles/ReleasePledge.css";
6  // 引入合约ABI
7  import HoldAssets from "../contracts/HoldAssets_ABI.json";
8  import NFT from "../contracts/NFT_ABI.json";
9  // 定义合约地址
10 const holdAssetsaddress = "0xbD818FA996C19F64841390De77Ee2b43E0d78F0d";
11 const nftaddress = "0x19FBDCcb3a1BC662638D8Aa9C5a139B3FB17eAE1";
12
13 export default function Releasepledge() {
14     const [pNfts, setPnfts] = useState([]); // 查询到自己所质押的NFT
15     const [loadingState, setLoadingState] = useState("not-loaded");
16     useEffect(() => {

```

```
17     pledgedNFTs();
18 }, []);
19
20 // 查询到自己所质押的NFT
21 async function pledgedNFTs() {
22     const web3Modal = new Web3Modal({
23         network: "mainnet",
24         cacheProvider: true,
25     });
26     const connection = await web3Modal.connect();
27     const provider = new ethers.providers.Web3Provider(connection);
28     const signer = provider.getSigner();
29     const holdAssets = new ethers.Contract(holdAssetsaddress, HoldAssets.abi, signer),
30     // 调起合约查询自己所质押的NFT
31     const data = await holdAssets.fetchMyPledgeNFTs();
32     console.log(data);
33     setPnfts(data);
34     setLoadingState("loaded");
35 }
36
37 if (loadingState === "loaded" && !pnfts.length)
38     return <h1 className="py-10 px-20 text-3xl">No assets owned</h1>;
39
40 // 解除质押
41 async function turnbackpledge(nft) {
42     const web3Modal = new Web3Modal({
43         network: "mainnet",
44         cacheProvider: true,
45     });
46     const connection = await web3Modal.connect();
47     const provider = new ethers.providers.Web3Provider(connection);
48     const signer = provider.getSigner();
49     const holdAssets = new ethers.Contract(holdAssetsaddress, HoldAssets.abi, signer),
50     const holdAssetsProvider = new ethers.Contract(holdAssetsaddress, HoldAssets.abi,
51     provider),
52     // 调起合约执行解除质押对应的NFT
53     const data = await holdAssets.turnbackNFT(
54         nftaddress,
55         nft
56     );
```

```
56     console.log(data);
57     // await data.wait();
58     if (data) {
59         // 解除质押的event事件
60         await holdAssetsProvider.on(
61             "TurnbackNFT",
62             (author, acc, event) => {
63                 console.log(author, author.toNumber());
64                 console.log("Event-tokenID:", author.toString()); // 解除质押的BigNumber
65                 console.log("Event-addr:", acc); // 当前账户地址
66                 console.log(event); // 当前事件的交易信息
67                 if (author) {
68                     alert("Successful release of pledge NFT!!!");
69                     window.location.replace("/pledge");
70                 }
71             }
72         );
73     } else {
74         alert("Pledge release NFT failed!!!");
75     }
76 }
77
78 // go back
79 async function goBack() {
80     await window.history.back(-1);
81 }
82
83 return (
84     <div className="ReleasePledge">
85         <div className="ReleasePledge-Head">
86             <h2>My NFT</h2>
87             <button onClick={goBack}>Go Back!</button>
88         </div>
89         <div className="ReleasePledge-Box">
90             {pNfts.map((nft, i) => (
91                 <div key={i} className="ReleasePledge-Box-items">
92                     <div className="ReleasePledge-Box-item-info">
93                         <h3>TokenId: {nft.toNumber()}</h3>
94                         <p>{nft.description}</p>
95                     </div>
```

```

96         <div className="ReleasePledge-Box-item-price">
97             <p className="text-2xl font-bold text-white">
98                 Price - {nft.price} Eth
99             </p>
100         <p>
101             <button onClick={turnbackpledge.bind(this, nft)}>
102                 取回质押
103             </button>
104         </p>
105     </div>
106 </div>
107     )}}
108 </div>
109 </div>
110 );
111 }
112
113

```

5.7 Model Page

完成功能：

a. 3D模型

```

1  import React from "react";
2  import { useEffect, useState } from "react";
3  import * as THREE from "../utils/three.min.js";
4  import { OrbitControls } from "../utils/OrbitControls.js";
5  import { OBJLoader } from "../utils/OBJLoader.js";
6  import { MTLLoader } from "../utils/MTLLoader.js";
7
8
9  export default function Model(props) {
10     var container; // 容器
11     var renderer; // 渲染器
12     var scene; // 场景
13     var camera; // 相机
14     var controller; // 视角控制器
15     var light; // 光源
16     var stats; // 性能检测器

```

```
17  useEffect(() => {
18    threeStart();
19  }, []);
20  // 初始化
21  async function threeStart() {
22    // canvas容器
23    container = document.getElementById("canvas");
24    container.id = "canvas" + props.data.tokenId;
25    // document.body.appendChild(container);
26    // 场景
27    initScene();
28    // 渲染器
29    initThree();
30    // 相机
31    initCamera();
32    // 插件
33    plugIn();
34    // 光
35    initLight();
36    // 画的内容
37    initObject();
38    // 游戏循环
39    render();
40  }
41
42  // 渲染器
43  function initThree() {
44    // 创建渲染器
45    renderer = new THREE.WebGLRenderer({
46      // 在 css 中设置背景色透明显示渐变色
47      alpha: true,
48      // 开启抗锯齿
49      antialias: true,
50    });
51    // 渲染背景颜色同雾化的颜色
52    renderer.setClearColor(scene.fog.color);
53    // 定义渲染器的尺寸；在这里它会填满整个屏幕
54    // console.log(container.clientWidth, container.clientHeight);
55    renderer.setSize(256, 272);
56    // renderer.setSize(window.innerWidth, window.innerHeight);
```

```
57 // 打开渲染器的阴影地图
58 renderer.shadowMap.enabled = true;
59 // renderer.shadowMapSoft = true;
60 renderer.shadowMap.type = THREE.PCFSoftShadowMap;
61 // 在 HTML 创建的容器中添加渲染器的 DOM 元素
62 container.appendChild(renderer.domElement);
63 // 监听屏幕，缩放屏幕更新相机和渲染器的尺寸
64 window.addEventListener("resize", handleWindowResize.bind(this), false);
65 }
66
67
68 // 窗口大小变动时调用
69 function handleWindowResize() {
70     // 更新渲染器的高度和宽度以及相机的纵横比
71     // renderer.setSize(window.innerWidth, window.innerHeight);
72     // camera.aspect = window.innerWidth / window.innerHeight;
73     // camera.updateProjectionMatrix();
74 }
75 // 插件
76 function plugIn() {
77     // 性能检测器
78     // stats = new Stats();
79     // container.appendChild(stats.dom);
80     // 视角控制
81     controller = new OrbitControls(camera, renderer.domElement);
82     controller.target = new THREE.Vector3(0, 0, 0); // 设置控制点
83     controller.autoRotate = true;
84     // 点击事件
85     container.addEventListener("mousedown", (event) => {
86         let mouse = new THREE.Vector2();
87         let raycaster = new THREE.Raycaster();
88         // 计算鼠标点击位置转换到3D场景后的位置
89         mouse.x = (event.clientX / renderer.domElement.clientWidth) * 2 - 1;
90         mouse.y = -(event.clientY / renderer.domElement.clientHeight) * 2 + 1;
91         // 由当前相机（视线位置）像点击位置发射线
92         raycaster.setFromCamera(mouse, camera);
93         let intersects = raycaster.intersectObjects(scene.children, true);
94         if (intersects.length > 0) {
95             // 拿到射线第一个照射到的物体
96             console.log(intersects[0].object);
```

```
97     }
98   });
99   // 辅助线
100   // this.scene.add(new THREE.GridHelper(50, 50, 0xffffffff, 0x555555));
101 }
102 // 场景
103 function initScene() {
104   scene = new THREE.Scene();
105   // 在场景中添加雾的效果，参数分别代表‘雾的颜色’、‘开始雾化的视线距离’、刚好雾化至看不见的视
   线距离’
106   scene.fog = new THREE.Fog(0xeeeeee, 1, 600);
107 }
108 // 相机
109 function initCamera() {
110   camera = new THREE.PerspectiveCamera(
111     45,
112     container.clientWidth / container.clientHeight,
113     1,
114     10000
115   );
116   camera.position.z = 80;
117 }
118 // 初始化光
119 function initLight() {
120   // 户外光源
121   // 第一个参数是天空的颜色，第二个参数是地上的颜色，第三个参数是光源的强度
122   let hemisphereLight = new THREE.HemisphereLight(0xaaaaaa, 0x000000, 2);
123
124
125   // 环境光源
126   let ambientLight = new THREE.AmbientLight(0xffffff, 0.2);
127
128
129   // 方向光是从一个特定的方向的照射
130   // 类似太阳，即所有光源是平行的
131   // 第一个参数是关系颜色，第二个参数是光源强度
132   let shadowLight = new THREE.DirectionalLight(0xffffff, 0.9);
133
134
135   // 设置光源的位置方向
```



```
136     shadowLight.position.set(50, 50, 50);
137
138
139     // 开启光源投影
140     shadowLight.castShadow = true;
141
142
143     // 定义可见域的投射阴影
144     shadowLight.shadow.camera.left = -400;
145     shadowLight.shadow.camera.right = 400;
146     shadowLight.shadow.camera.top = 400;
147     shadowLight.shadow.camera.bottom = -400;
148     shadowLight.shadow.camera.near = 1;
149     shadowLight.shadow.camera.far = 1000;
150
151
152     // 定义阴影的分辨率；虽然分辨率越高越好，但是需要付出更加昂贵的代价维持高性能的表现。
153     shadowLight.shadow.mapSize.width = 2048;
154     shadowLight.shadow.mapSize.height = 2048;
155
156
157     // 为了使这些光源呈现效果，需要将它们添加到场景中
158     scene.add(hemisphereLight);
159     scene.add(shadowLight);
160     scene.add(ambientLight);
161 }
162 // 画内容
163 function initObject() {
164     const mtlLoader = new MTLLoader();
165     function mtlFun(mtlurl, objurl, x, y, z) {
166         mtlLoader.load(mtlurl, (mtl) => {
167             for (const key in mtl.materialsInfo) {
168                 let val = mtl.materialsInfo[key];
169                 let urls = val.map_kd.split("\\");
170                 val.map_kd = `${props.data.addr}/${urls[urls.length - 1]}`;
171                 // console.log(val.map_kd);
172             }
173             mtl.preload();
174             const objLoader = new OBJLoader();
```

```

175     objLoader.setMaterials(mtl);
176     objLoader.load(objurl, (obj) => {
177         obj.traverse(function (child) {
178             if (child.material) {
179                 child.material.transparent = true; // 默认为true,可省略
180                 child.material.alphaTest = 0.7;
181                 child.material.depthWrite = true; // 默认为true,可省略
182             }
183         });
184         obj.position.set(x, y, z);
185         scene.add(obj);
186     });
187 });
188 }
189 mtlFun(props.data.mtl, props.data.obj, -15, -30, 0); // 中间
190 // mtlFun(
191 //
192 "https://ipfs.io/ipfs/bafybeihcwnwnycaopdhfel7sc3yvt4aq6czxe4a2d45fhqv4wj55neo4pq/3.mtl"
193 ,
194 //
195 "https://ipfs.io/ipfs/bafybeihcwnwnycaopdhfel7sc3yvt4aq6czxe4a2d45fhqv4wj55neo4pq/3.obj"
196 ,
197 // -15,
198 // -30,
199 // 0
200 // ); // 中间
201 }
202 // 游戏循环
203 function render() {
204     renderer.clear();
205     renderer.render(scene, camera);
206     requestAnimationFrame(render);
207     // stats.update();
208     // TWEEN.update();
209 }
210
211 return (
212     <div
213         id="canvas"
214         className="webgl"

```

```
212     refs="webgl"
213     style={{
214         // display: "inline-block",
215         width: "256px",
216         height: "272px",
217         backgroundColor: "#000",
218         border: "0",
219     }}
220 ></div>
221 );
222 }
223
```

技术支持:

十二生肖网络科技有限公司

技术部: 总监: 吕磊

前端: 陆俊佑 杨加骏

3D模型: 曾源