2022/05/06

NFT발행 두 가지 방법

1. SoundGram에서 배포한 기존의 Smart Contract에서 mintAlbum 함수 사용

   SoundGram NFT Market은 사용자가 소유한 NFT목록, 거래 기록 등 자체 데이터 베이스 보유

   사용자가 Market 내부가 아닌 외부에서 NFT이동이 가능함, 데이터 베이스의 정보가 실제 정보와 달라지는 문제 발생

2. Klip Partners에서 제공해주는 Smart Contract을 이용

   Http통신으로 Klip Partners에 설정하고 싶은 인자들을 적어서 Post

   인자에 필수적으로 Klip Partners에서 제공해준 Smart Contract 주소를 넘겨야 함

```
axios({
    url: 'https://api.klipwallet.com/v2/wallet/mint',
    method: 'post',
    headers: {
             authorization: security.authorization ,
            'Content-Type': 'application/json'
            },
    data: {
      "pin": pin,
        "to_address": [wallet_address],
        "contract_address": contract_address,
        "name": bapp_name,
        "description" : "klip minting test",
        "image": testImageUrl,
        //false로 클립에서 거래를 차단할 수 있음
        //카카오톡 UI에서 차단되는 것이고 Smart Contract거래는 정상 작동
        "sendable" : true,
        "send_friendly_only" : true,
      }
    })
    .then(function a(response) {
      console.log(response);
    })
    .catch(function (error) {
      console.log(error);
    });
```

   큰 문제는 아니지만, Smart Contract 두 가지 다루게 되므로 깔끔하지 않음

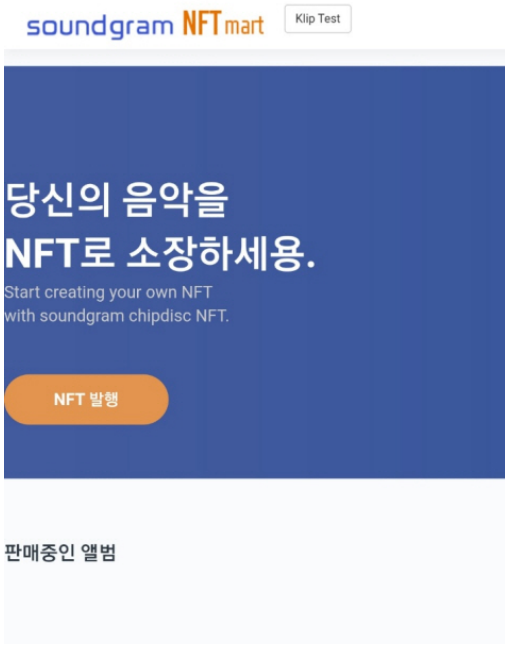
   현재는 두 번째 방법을 사용하기로 결정

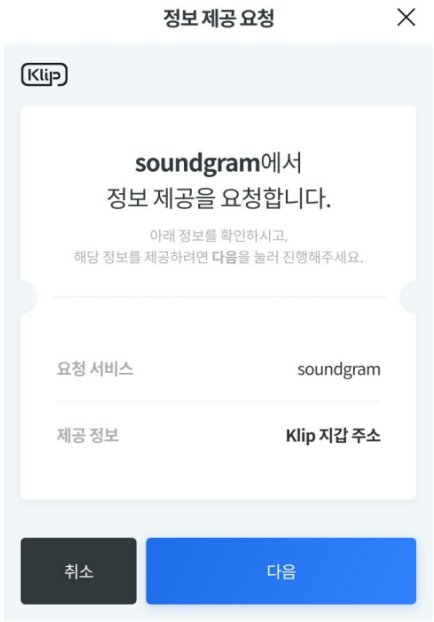
Klip App2App API

Klip Partners에서 제공한 Smart Contract주소 없이 실행 할 수 있는 기능들(모두 HTTP 통신)
세 가지 기능만 사용 (Auth요청, getCardList, excute_contract)

```
//Auth요청 예
//사용자의 클립 지갑 주소를 가져오는 과정
const res = await prepare.auth({ bappName, successLink, failLink });
request(request_key, () => alert('모바일 환경에서 실행해주세요'));
const res2 = await getResult(request_key);

klaytn_address = res2.result.klaytn_address;
//사용자가 보유한 Klay 조회
await cav.klay.getBalance(klaytn_address)
```






```
//getCardList 예
const contract = 조회할 카드의 contarct 주소
const eoa = 유저 지갑 주소
const res = getCardList({contract, eoa});
```

ABI(Application Binary Interface) :

특정 언어나 플랫폼에 종속되지 않은 방식으로 기술된 Application Interface 정의
Smart Contract의 함수와 Parameter에 대한 Metadata가 정의된 것
JS기반의 작업을 할 때 객체를 만들게 할 수 있고,
객체의 Method를 호출하는 것으로 Contract의 함수가 호출되도록 할 수 있음

```
//Smart Contract 내부에 선언된 함수
function mintAlbum (string memory _key, uint256 _albumCode) public {
  .....
}
```

```
//ABI와 Smart Contract 주소를 인자로 받는 객체 생성
const tokenContract = new cav.klay.Contract(ABI, ADDRESS);
//Smart Contract의 함수를 쉽게 호출
await tokenContract.methods.mintAlbum(key, Code).send({from: sender, gas: 0});
```

핵심 기능 excute contract - 현재 진행 중
특정 smart contract의 함수를 실행
SoundGram의 smart contract에 있는 함수를 실행하도록 요청 할 수 있음
smart contract를 메인 넷에 배포해야 함

```
axios({
        url: 'https://a2a-api.klipwallet.com/v2/a2a/prepare',
        method: 'post',
        headers: {
                'Content-Type': 'application/json'
                },
        data: {
          "bapp": { "name" : security.name },
            "type": "execute_contract",
            "transaction": {
                "from" : 보내는 지갑 주소,
                "to": 실행할 smart contract 주소,
                "value": "0",
                "abi": 사용할 ABI 선택
                "Params" : 사용할 ABI에 들어갈 인자
            }
        }
        })
        .then(function a(response) {
          console.log(response);
        })
        .catch(function (error) {
          console.log(error);
        });
```

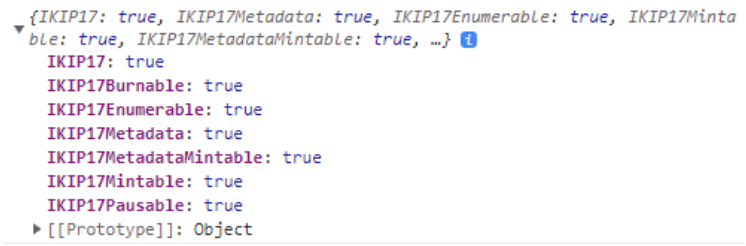
카이카스

카이카스는 아래와 같은 코드로 계정로그인을 하고 caver라이브러리를 준비시킬 수 있음

```
const accounts = await klaytn.enable();
const account = accounts[0];
caver = new Caver(klaytn);
    if (typeof window.klaytn !== 'undefined') {
      // Kaikas user detected. You can now use the provider.
      const provider = window['klaytn']
      caver.setProvider(provider);
```

클립 nft가 어떠한 인터페이스를 가지는지 알아보기 위하여 확인

```
caver.kct.kip17.detectInterface(Klip_contract)
```




```
function testGetNftBalance(addressOwner){
  const myContract = new caver.klay.Contract(KIP17Fullabi,klipContract )
  return myContract.methods.balanceOf(addressOwner).call();
};

function getTokenOfOwnerByIndex(addressOwner,index){
  const myContract = new caver.klay.Contract(KIP17Fullabi, klipContract)
  return myContract.methods.tokenOfOwnerByIndex(addressOwner,index).call();
```

```
}
function testtokenByIndex(addressOwner,balance){
  for (var i = 0; i < balance; i++) {
    (async () => {
      var tokenId = await getTokenOfOwnerByIndex(
        addressOwner,
        i
      );
      console.log(tokenId);
    })();
  }
```

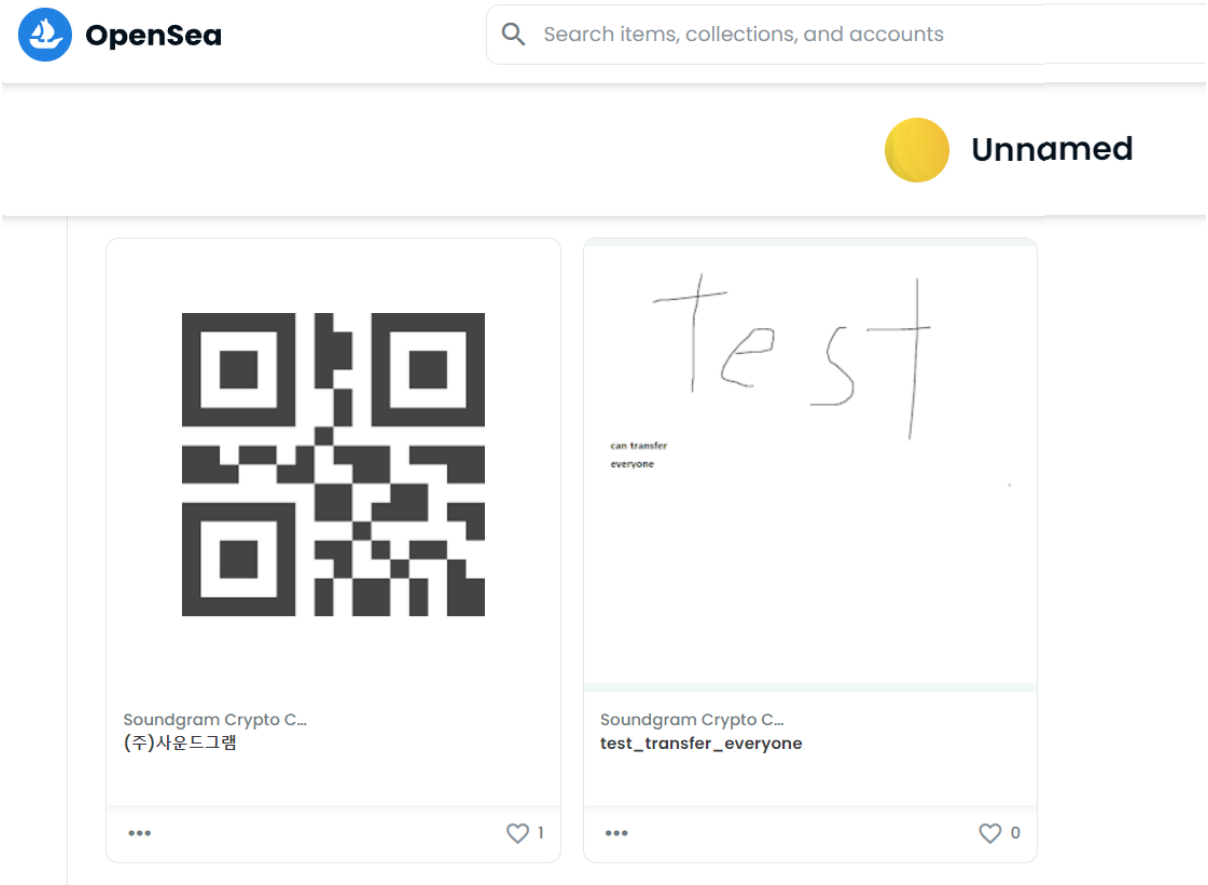
1

13

2

13

19




```
function testNftUri(addressOwner){
  const myContract = new caver.klay.Contract(KIP17Fullabi, klipContract)
  return myContract.methods.tokenURI(13).call();
}
```

https://media.klipwallet.com/card_asset/1666700/fcb313ad-dbd2-42d9-81da-31266007f433.json

Solidity

```
contract SoundgramAlbumSales {
  KIP17Full public nftAddress;

  mapping(uint256 => uint256) public tokenPrice;

  constructor(address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
  }

  function setForSale(uint256 _tokenId, uint256 _price,address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
      address tokenOwner = nftAddress.ownerOf(_tokenId);    //public, external 함수만 이렇게 호출 가능
      require(tokenOwner == msg.sender, "caller is not token owner");
      require(_price > 0, "price is zero or lower");
      require(nftAddress.isApprovedForAll(tokenOwner,address(this)), "token owner did not approve TokenSales contract");

      tokenPrice[_tokenId] = _price; //특정 토큰의 가격 저장.
  }

  function purchaseToken(uint256 _tokenId,address _tokenAddress) public payable {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
      uint256 price = tokenPrice[_tokenId];
      address tokenSeller = nftAddress.ownerOf(_tokenId);

      require(msg.value >= price, "caller sent klay lower than price"); //가격 이상의 비용을 지불해야 함.
      require(msg.sender != tokenSeller, "caller is token seller");    //본인은 구매 불가
      address payable payableTokenSeller = address(uint160(tokenSeller));

      payableTokenSeller.transfer(msg.value);

      nftAddress.safeTransferFrom(tokenSeller, msg.sender, _tokenId); //tokenSeller->구매자 토큰 전송
      tokenPrice[_tokenId] = 0;
  }

  function removeTokenOnSale(uint256 _tokenId,address _tokenAddress) public {
      nftAddress = KIP17Full(_tokenAddress); //YTT 주소를 넘김. 컨트랙에 있는 함수 사용 가능.
      address tokenSeller = nftAddress.ownerOf(_tokenId);
      require(msg.sender == tokenSeller, "caller is not tokenSeller");
      tokenPrice[_tokenId] = 0;
  }
}
```

```
const SAT = artifacts.require('SoundgramAlbumToken');<-변경

module.exports = function (deployer) {
  deployer.deploy(AlbumSales, SAT.address) //constructor의 인자
    .then(() => {
      if (AlbumSales._json) {
        fs.writeFile(
          'deployedABI_AlbumSales',
          JSON.stringify(AlbumSales._json.abi),
          (err) => {
            if (err) throw err
            console.log("파일에 ABI 입력 성공");
          })
      }

      fs.writeFile(
        'deployedAddress_AlbumSales',
        AlbumSales.address,
        (err) => {
          if (err) throw err
          console.log("파일에 주소 입력 성공");
        })
    })
}
```

그 외 이슈 : 테스트용으로 1200원일 때 구매한 Klay코인이 900원으로 떡락함