2022/05/20

구조개선

이후 시나리오와 전해질 디자인에 맞춰 코드를 적용할수 있도록 준비.

기존 테스트 방식은 시간 부족과 기능 테스트에 급급하여 일부 정리되지 못하고 단순 주석처리 혹은 한가지 함수에 전부 코드를 밀어넣어 테스트하는 경우가 있었다.

```
klipTest: async function() {

  typeof window.klaytn !== 'undefined';
  const accounts = await klaytn.enable();
  const account = accounts[0];
  klaytn.selectedAddress
  klaytn.on('accountsChanged', function(accounts) {
    // Your code
  console.log(klaytn.selectedAddress);
  });
  console.log(klaytn.selectedAddress);
  console.log(account);


  var ua = require("ua_parser").userAgent();
  console.log("klipTest",ua.platform);

  // const bappName = security.name;
  // const from = security.testHajineKlip;
  // const to = security.contract_address;
  // const value = "0";
  // const abi = "{\"constant\": false,\"inputs\": [{\"name\": \"to\",\"type\": \"address\"},{\"name\": \"approved\",\"type\": \"bool\"}],\"name\": \"setApprova
  // const params = "[\"0xd6646EA5152b9d8e244a4d24027D1D4f5Abc0Ee5\",true]"
  // const res = await prepare.executeContract({ bappName, from, to, value, abi, params })
  // if (res.err) {
  //   console.log(res.err);
  // } else if (res.request_key) {
  //   request_key = res.request_key;
  // }

  // request(request_key, () => alert('모바일 환경에서 실행해주세요'));
  // // //TODO 클립앱에서 다시 웹으로 돌아오는것 확인하는 방법찾기
  // sleep(13000);
  // try {
```

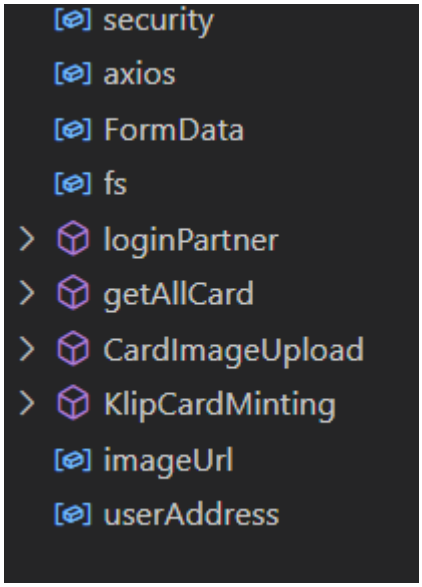
index.js의 test용 함수내부 모습

index.js와 admin.js에 존재하는 카드 민팅, 클립 함수, 카이카스 관련 함수를 세가지 파일로 분할

이후 가독성 및 유지보수의 수월함을 위해 다음과 같이 각 기능 테스트를 정리하고 함수들을 분리하였다.

```
//토큰을 setSaleOnCont로 판매등록 하기 위해선 approve가 선행되어야 함
//

const balance=await getAccountTokenBalance(account);
console.log("TestKaikas GetAccountTokenBalance",balance);


let tokenId=13;
const TokenInfo=await getTokeninfoURL(tokenId);
console.log("TestKaikas GetTokeninfoURL",TokenInfo);


await getAllTokenOfOwner(account,balance);
console.log("TestKaikas GetAllTokenOfOwner");


const approveReceipt=await approve(account);
console.log("TestKaikas approve",approveReceipt.transactionHash);


let pebPrice=KlayToPeb("0.0001");
const saleOnContReceipt=await setSaleOnCont(account,tokenId,pebPrice);
console.log("TestKaikas setSaleOnCont",saleOnContReceipt.transactionHash);


const tokenPrice=await getTokenPrice(tokenId);
console.log("TestKaikas GetTokenPrice",tokenPrice);



pebPrice=KlayToPeb("0.0001");
tokenId=21;
const buyTokenReceipt=await buyToken(account,tokenId,pebPrice);
console.log("TestKaikas buyToken",buyTokenReceipt.transactionHash);
```

Kaikas 테스트용 함수.



CardMinting.js

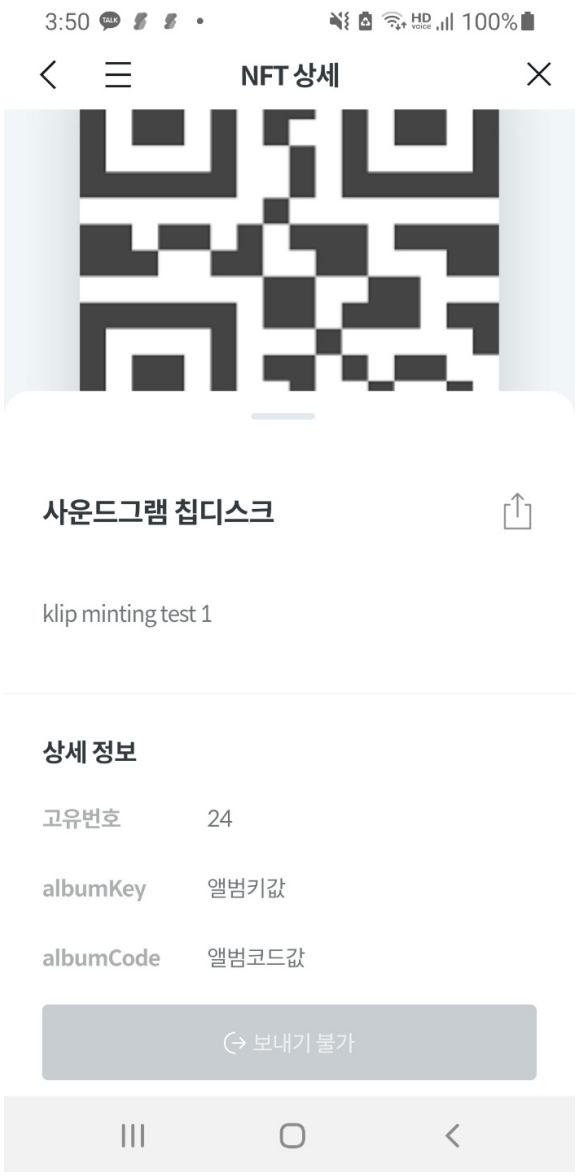
아쉬운점

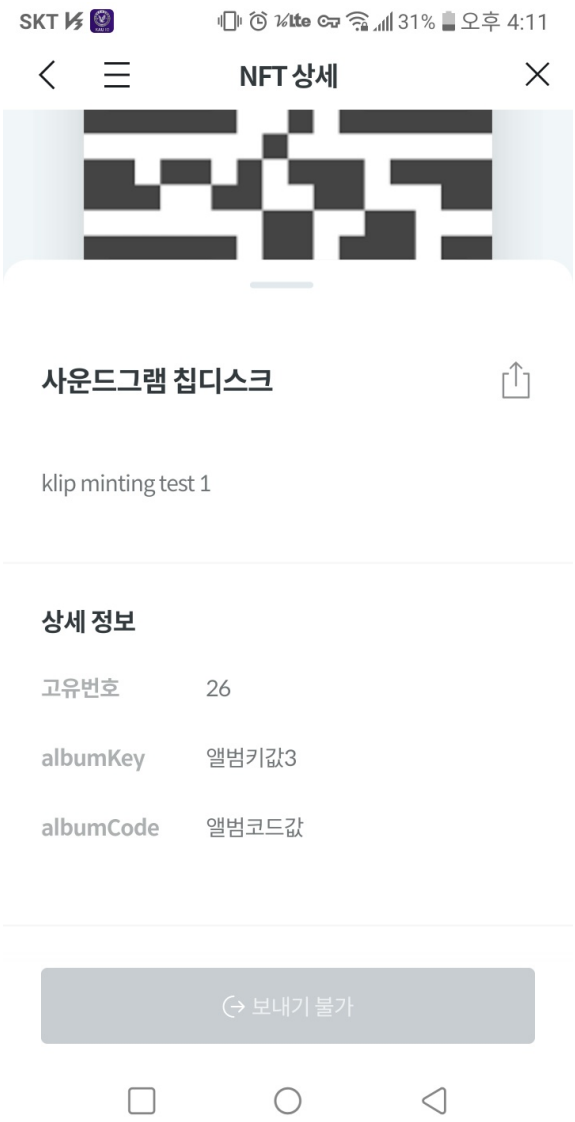
TDD등에서 사용되는 자동화 테스트 툴과 테스트 시나리오등을 사용하고 싶었으나 klip은 테스트넷을 지원하지 않고, 테스트의 주된 요인들이 Kaikas와 Klip간의 기능 호환성 여부인데 각 테스트시 핸드폰과 브라우저에서 klip과 kaikas의 인증이 필요하여 기능 분할과 console.log 위주로 진행하였다.

NFC KEY추가를 위한 attributes기능 테스트

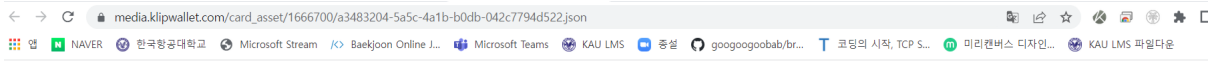
DB와 Nft와의 연동을 위해 각 토큰이 Nfc key를 보유하고 있어야하는 이슈가 있어 새로운 템플릿을 만들지 않고 기존 템플릿을 이용하여 속성을 개별로 부여 할 수 있는지 klip token의 attributes기능을 테스트하였다.

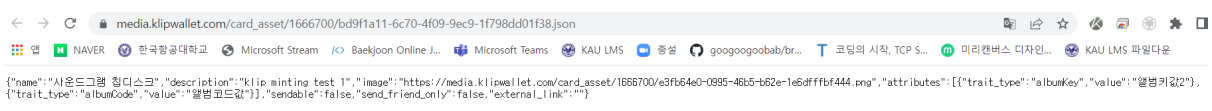
```
axios({
        url: 'https://api.klipwallet.com/v2/wallet/mint',
        method: 'post',
        headers: {
                authorization: security.authorization ,
                'Content-Type': 'application/json'
                },
        data: {
            "pin": security.pin,
            "to_address": userAddress,
            "contract_address": security.contract_address,
            "name": security.service_name,
            "description" : "klip minting test 1",
            "image": testImageUrl,
            "sendable" : false,
            "send_friendly_only" : false,
            "attributes" : [
                {
                  "trait_type": "albumKey",
                  "value": "앨범키값3"
                },
                {
                  "trait_type": "albumCode",
                  "value": "앨범코드값"
                }
            ]
            }
        })
```





같은 템플릿을 이용하여 albumKey값만 다르게 발급한 토큰의 이미지






```
(await getTokeninfoURL(24));
```

```
(await getTokeninfoURL(25));
```

다음과 같이 웹상에서도 코드로 확인이 가능함을 확인