ERC1155CONTRACT

SELLCONTRACT

ANA KONTRAT DİZAYNI

SERCAN ÇELENK

14 Temmuz 2021

İÇİNDEKİLER

1 Fonksiyonlar(Functions)

1.1 Toplu Aktarım(Batch Transfer)

Toplu işlemler ve aktarımlar; gazdan tasarruf etmek yerine zamandan tasarruf etmek ve hataları kaldırmak ile ilgilidir. Toplu işlem araçları, birden çok adrese jeton gönderilmesini gerektiren herhangi bir işlem için yararlıdır.

Örnek verecek olursak; hebys.io hebys tokenlarını 1 senedir cüzdanında tutan kullanıcılara sadakat ödülü olarak tuttukları tokenın yüzde 1'i kadar ödül dağıtmak istiyor. Bunu yapmanın en kolay yolu Toplu Aktarım'lardır.

Bir başka senaryoda; hack veya ani kredi saldırılarından(flash loan) dolayı şirket zarar görmüş varsayalım.

Kısa süre önce belt.fi böyle bir saldırıya maruz kalmıştı ve yüklü miktarda dolar 4-5 saniyede çalınmıştı. Belt.fi havuzlarında likidite sağlayıcılığı yapan çoğu insan da haliyle yüzde 20'ye yakın mevduat erimesi yaşamıştı.

Buna çare olarak belt.fi r4Belt adında bir token çıkardı ve bu tokenın airdropunu kısa sürede gerçekleştirdi.

Bitfinex adlı borsa hacklendikten kısa süre sonra LEO adında bir coin çıkarıp bunun airdrobunu yaptı.

Eğer sözleşme sahibi kontratın içine Batch Transfer koymasaydı bu tarz airdropların gerçekleşmesi mümkün değildi.

Batch Transfer yani Toplu Aktarım bazı durumlarda hayati önem taşımaktadır. Verdiğim örnekler bunun en güzel kanıtıdır.

1.2 NFT Desteği(NFT Support)

Bir arz sadece 1 tane olarak verildiğinde, jeton(token) esasen değiştirilemez bir jetondur yani NFT'dir. Değiştirilemez diyorsak bunu tanımlayan ERC standartı 721'dir. Her özel(tek) token için bir NFT desteği olmalıdır ve her biri için bir metadataURL tanımlanmalıdır.

ERC1155Metadata_URI.json

```
{
    "title": "hebysToken Metadata",
    "type": "object",
    "properties": {
        "name": {
            "type": "string",
            "description": "MEYVE"
        },

        "decimals": {
            "type": "integer",
            "description": "Tek adet'tir."
        }

    }
}
```

1.3 Toplu Bakiye(Batch Balance)

Toplu Bakiye işlemi elimizdeki sorgu durumuna göre istediğimiz cüzdanları listelememize yarar. Listeleme sonrasında istediğimiz aksiyonu alabiliriz.

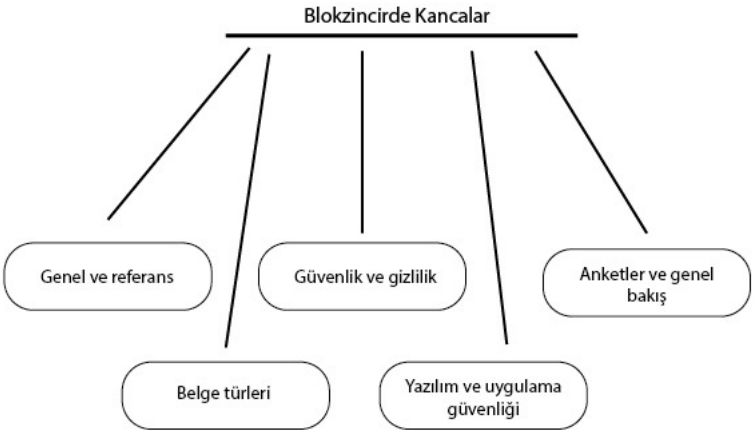
Örnek senaryo; 22.07.2021 tarihi saat 00.00'dan itibaren cüzdanında 100 adet ve daha fazla hebys tutan kişilere ödül dağıtılacaktır. 100 adet ve daha fazlasını tutan cüzdanlara Toplu Bakiye sorguları ile ulaşabiliriz.

batch_balance.sol

```
1
2
3  get_wallets=[100] and _owners=[0xctjk..., 0x1032..., 0x1756...]
4  //sercancelenk
```

1.4 Güvenli Transfer Kuralı(Safe Transfer Rule)

Güvenli transferin en önemli kuralı from fonksiyonunda; çağrıcı yani arayan elindeki jetonu harcamak(göndermek) için muhakkak sözleşmeyi onaylamış olması gerekmektedir.
Çağrıcı çağrı yaptığında geri dönüş şartları; Adres bakiyesi 0 ise,
Çağrı yapılan değer ile cüzdandaki değer uyuşmuyorsa,
Deftere yazılmayı bekleyen pending işlemler dahil, cüzdan bakiyesi yetersizse,
Ve diğer çeşitli sebepler,



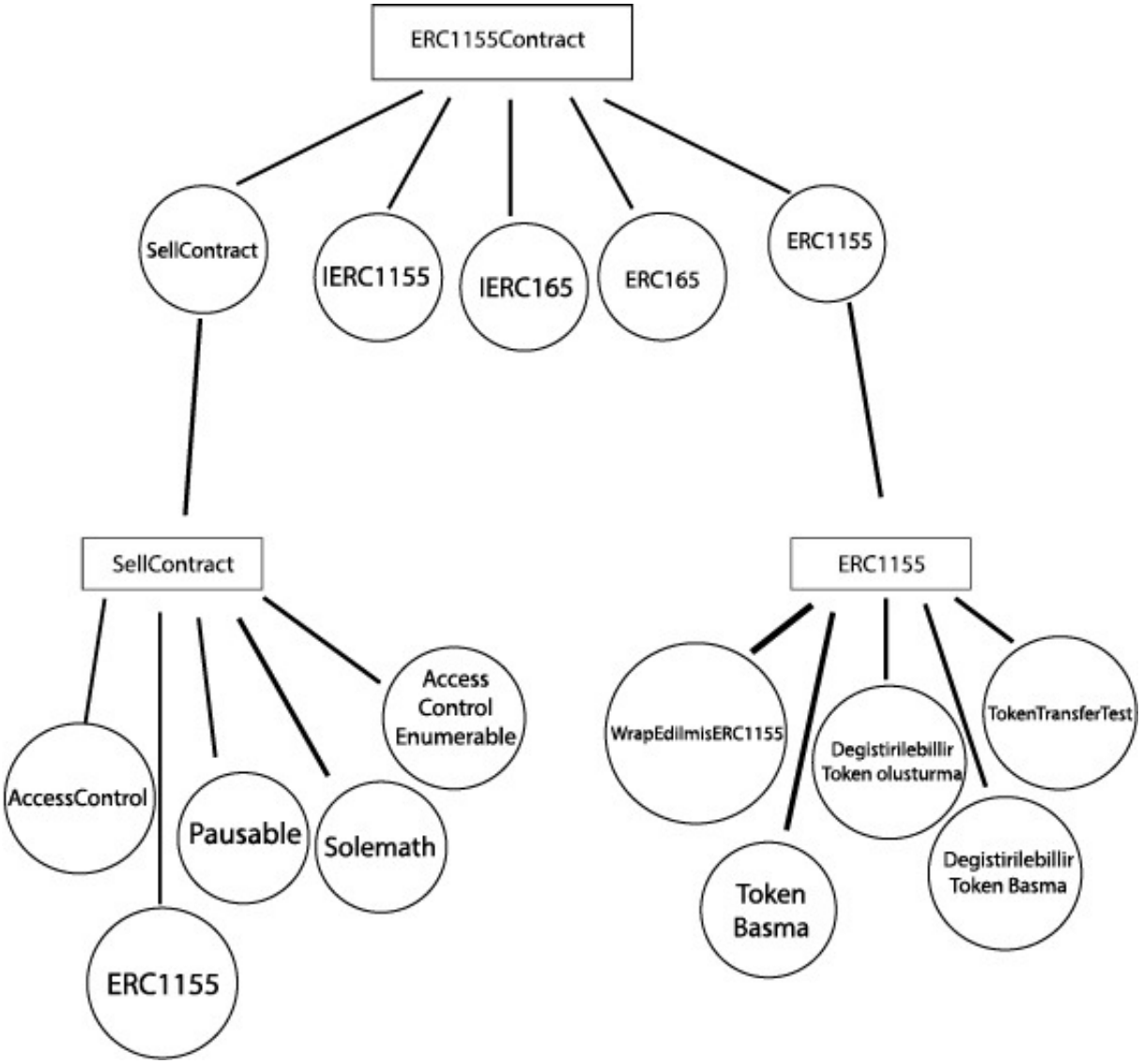
1.5 Kancalar(Hooks)

Kötü niyetli işlemlerin deftere girmesini önlemek; devam eden işlemleri(pending) senkronize etmek ve analiz etmek için çalışma zamanı kancası şarttır.(Runtime Hook)

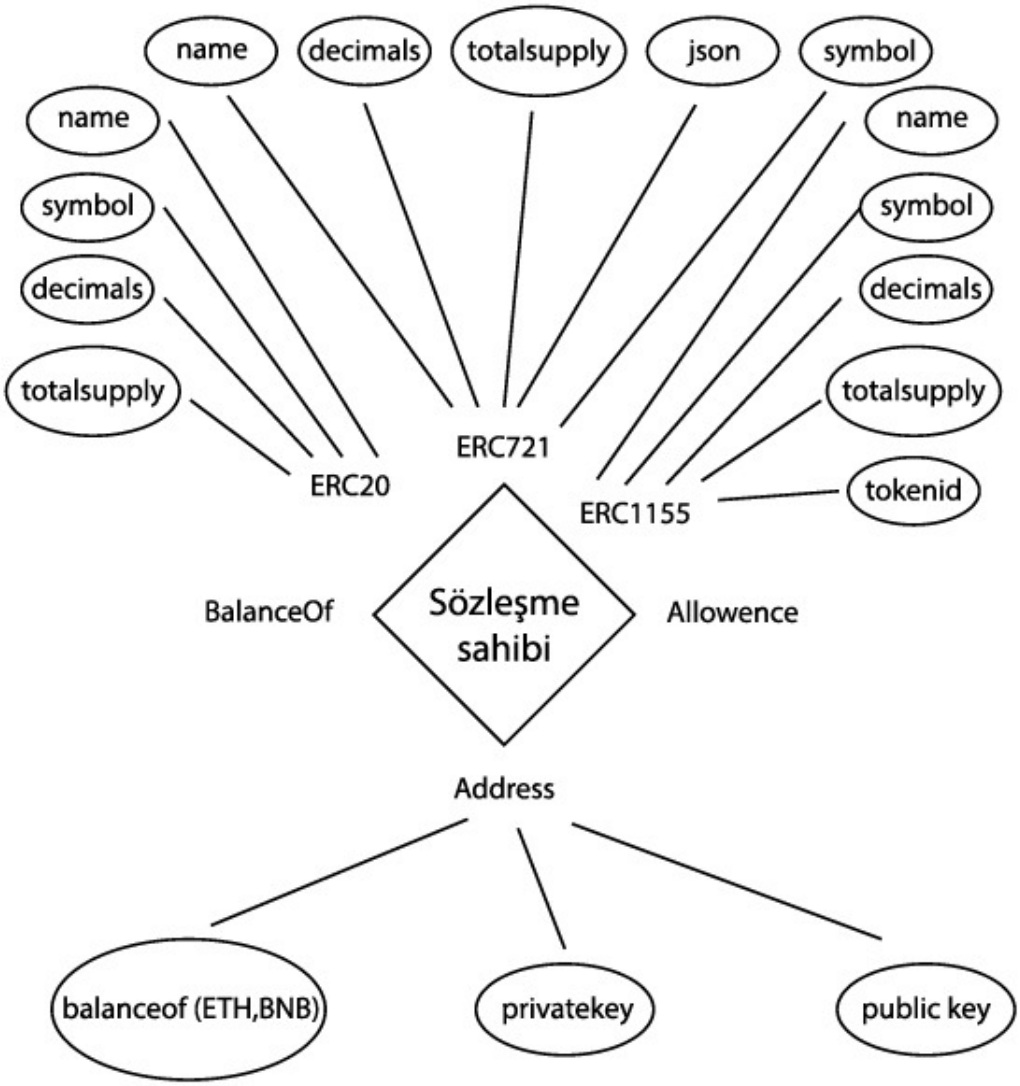
Bu kanca ile kötü niyetli işlemlerin gerçekleşmesini önleyebilir, imza sahibinin yani saldırganın kontrata erişimini engelleyebiliriz.(flag)

2 Akış Şemaları(Diagram Schemas)

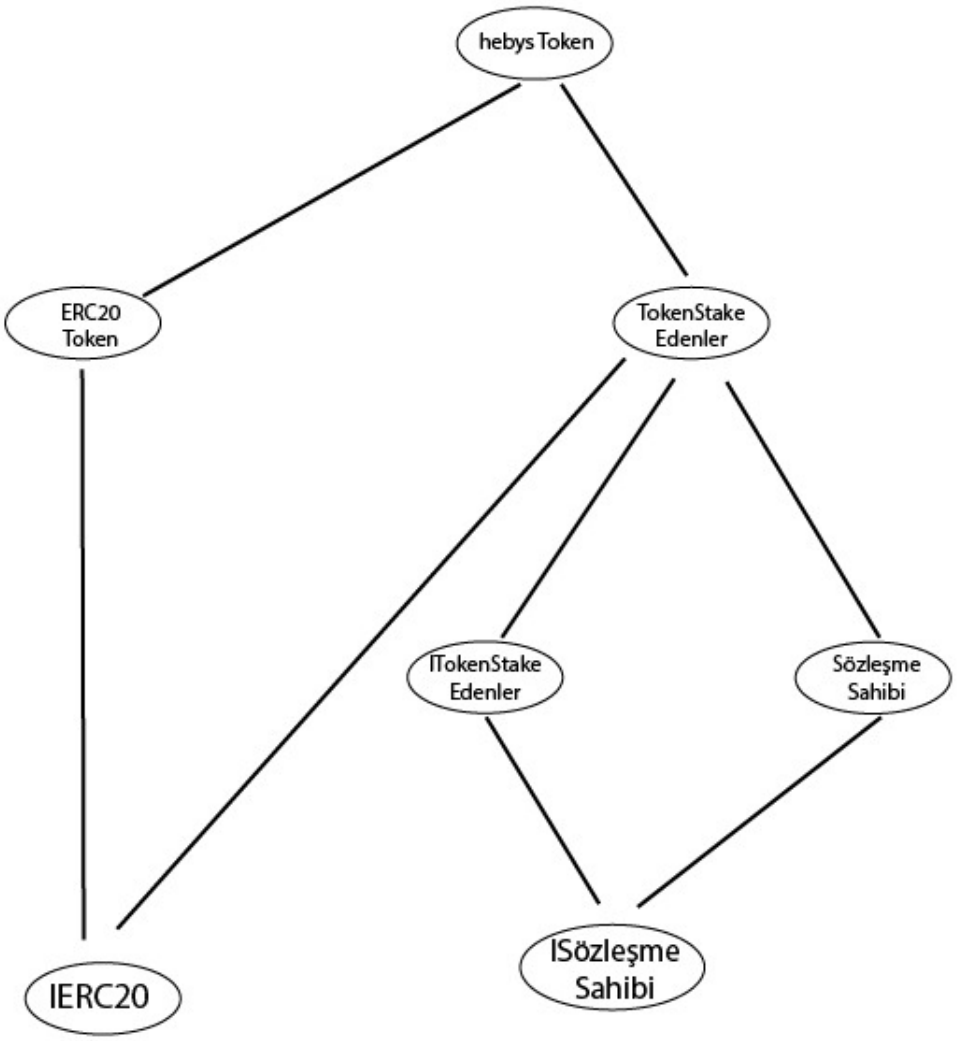
2.1 Kontrat Fonksiyonları, ve Çağırılan Sözleşmeler



2.2 Genel Akış Şeması



2.3 Stake Etme Fonksiyonu için Akış Senaryosu



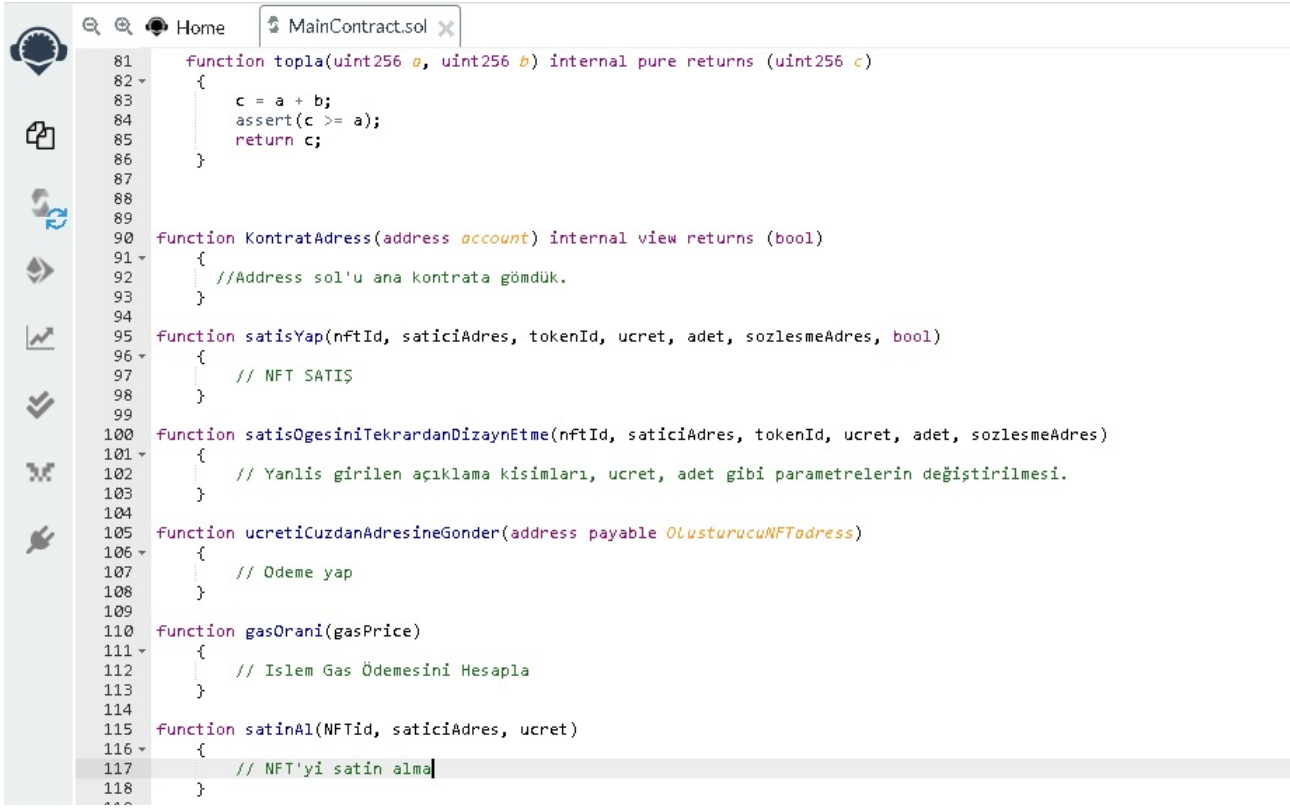
2.4 Ana Kontrat Kod Gövdesi(Main Contract Code Body)

Home | MainContract.sol

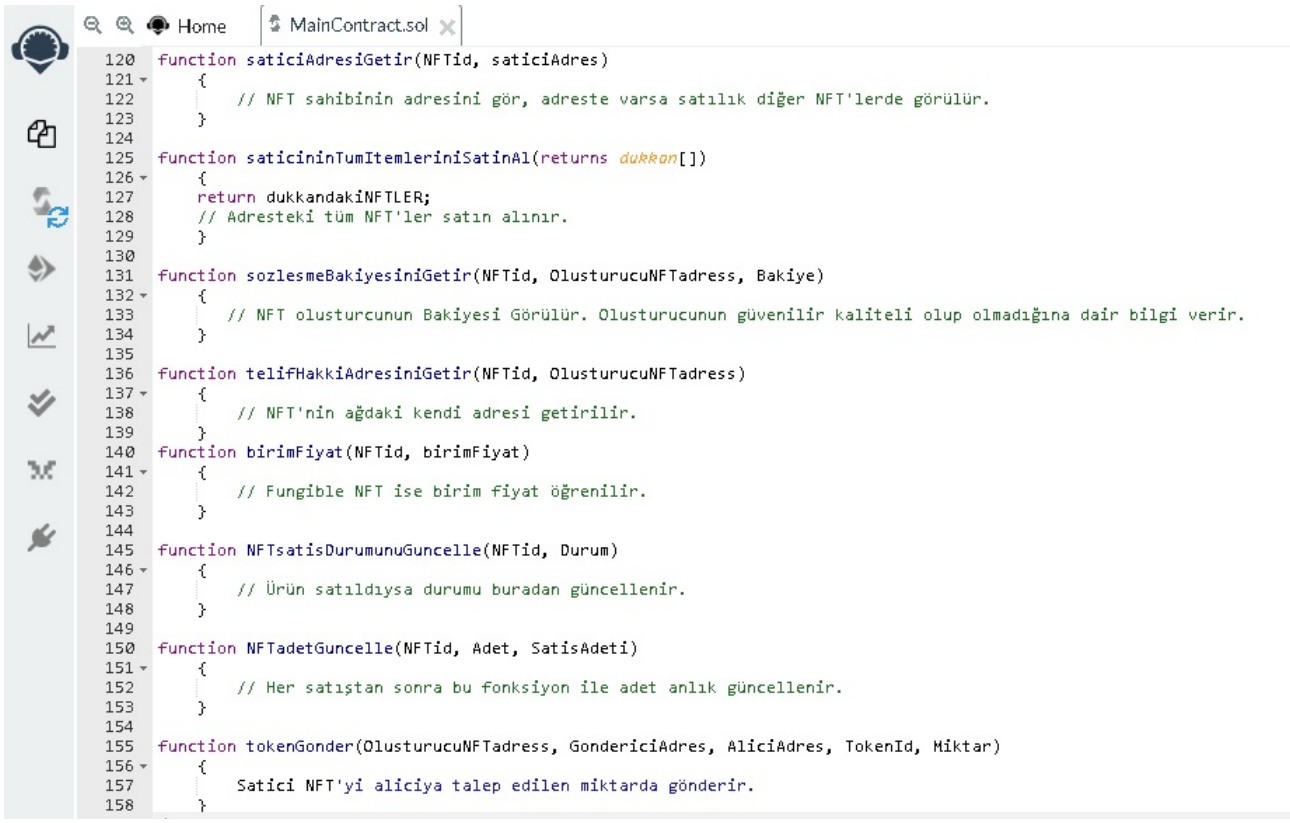
```
pragma solidity ^0.5.0;


import "./IERC1155.sol";                    // Aşağıdaki fonksiyonların çalışabilmesi için IDE içine gömülü olan InterfaceERC1155 çağırılmalıdır.
import "./Migrations.sol";                  // VERSIYON YUKSELTMELERI ICIN MIGRATIONS.SOL CAGIRILMALIDIR. ÖRNEĞİN; ETH 2.0 GÖÇÜ
import "./ERC1155AllowanceWrapper.sol";     // COIN-TOKEN WRAP ETME ////////////////////////////
import "./ERC1155Mintable.sol";             // TOKEN BASMA
import "./ERC1155MixedFungible.sol";        // DEGISTIRILEBILIR TOKEN OLUSTURMA
import "./ERC1155MixedFungibleMintable.sol"; // DEGISTIRILEBILIR TOKEN BASMA
import "./ERC1155MockReceiver.sol";         // TOKEN TRANSFER TESTI
import "./Common.sol";                      //  ORTAK TOPLU AKTARIMLAR VE TOPLU SOZLESME CAGRILARI ICIN COMMON.SOL CAGIRILIR.


function guvenliTransfer(address _from, address _to, uint256 _id, uint256 _value, bytes calldata _data) external
    {
        // güvenli transfer fonksiyonu

    }

function guvenliTopluAktarim(address _from, address _to, uint256[] calldata _ids, uint256[] calldata _values, bytes calldata _data) external
    {
        //toplu aktarım fonksiyonu (Batch Transfer)
    }

function bakiyeDurum(address _owner, uint256 _id) external view returns (uint256) {
        //Cüzdan bakiyesi sorgulama
    }

 function topluBakiyeSorgulama(address[] calldata _owners, uint256[] calldata _ids) external view returns (uint256[] memory)
    {
    //Şartlara göre toplu cüzdan bakiyesi sorgulama fonksiyonu. Örneğin; "Hesabında 100'den fazla hebys tutan cüzdanları ara".

    }

function transferOnayi(address _operator, bool _approved) external
    {

     // Tokeni harcayacak ve kullanacak kişi yani token sahibi tarafından onay durumu sorgusu.
    }
```

Home | MainContract.sol

```
function transferKontrolu(address _operator, address _from, address _to, uint256 _id, uint256 _value, bytes memory _data) internal
    {

        // Transfer yapılmak istenen cüzdan adresinin kontrolu gerçekleştirilir. Örneğin cüzdan adresinde bilinmeyen karakterler varsa,
        // işlem false olarak geri döner.
    }



interface ERC1155Metadata_URI {

    // Metadata bilgileri için ERC1155 JSON dosyası oluşturma fonksiyonu
    function uri(uint256 _id) external view returns (string memory);
}


//////////////////////////////////////////// SAFEMATHI ÇAĞIRMAYIP KENDİMİZ KONTRATA GÖMÜYORUZ ///////////////////////////////////

function carp(uint256 a, uint256 b) internal pure returns (uint256 c)
    {
        if (a == 0) {
            return 0;
    }
        c = a * b;
        assert(c / a == b);
        return c;
    }

    function bol(uint256 a, uint256 b) internal pure returns (uint256)
    {
        return a / b;
    }

    function cikar(uint256 a, uint256 b) internal pure returns (uint256)
    {
        assert(b <= a);
        return a - b;
    }
```




```
81      function topla(uint256 a, uint256 b) internal pure returns (uint256 c)
82      {
83          c = a + b;
84          assert(c >= a);
85          return c;
86      }
87
88
89
90  function KontratAdress(address account) internal view returns (bool)
91      {
92        //Address sol'u ana kontrata gömdük.
93      }
94
95  function satisYap(nftId, saticiAdres, tokenId, ucret, adet, sozlesmeAdres, bool)
96      {
97          // NFT SATIŞ
98      }
99
100 function satisOgesiniTekrardanDizaynEtme(nftId, saticiAdres, tokenId, ucret, adet, sozlesmeAdres)
101     {
102         // Yanlis girilen açıklama kisimları, ucret, adet gibi parametrelerin değiştirilmesi.
103     }
104
105 function ucretiCuzdanAdresineGonder(address payable OlusturucuNFTadress)
106     {
107         // Odeme yap
108     }
109
110 function gasOrani(gasPrice)
111     {
112         // Islem Gas Ödemesini Hesapla
113     }
114
115 function satinAl(NFTid, saticiAdres, ucret)
116     {
117         // NFT'yi satin alma
118     }
```




```
120 function saticiAdresiGetir(NFTid, saticiAdres)
121     {
122         // NFT sahibinin adresini gör, adreste varsa satılık diğer NFT'lerde görülür.
123     }
124
125 function saticininTumItemleriniSatinAl(returns dukkan[])
126     {
127     return dukkandakiNFTLER;
128     // Adresteki tüm NFT'ler satın alınır.
129     }
130
131 function sozlesmeBakiyesiniGetir(NFTid, OlusturucuNFTadress, Bakiye)
132     {
133        // NFT olusturcunun Bakiyesi Görülür. Olusturucunun güvenilir kaliteli olup olmadığına dair bilgi verir.
134     }
135
136 function telifHakkiAdresiniGetir(NFTid, OlusturucuNFTadress)
137     {
138         // NFT'nin ağdaki kendi adresi getirilir.
139     }
140 function birimFiyat(NFTid, birimFiyat)
141     {
142         // Fungible NFT ise birim fiyat öğrenilir.
143     }
144
145 function NFTsatisDurumunuGuncelle(NFTid, Durum)
146     {
147         // Ürün satıldıysa durumu buradan güncellenir.
148     }
149
150 function NFTadetGuncelle(NFTid, Adet, SatisAdeti)
151     {
152         // Her satıştan sonra bu fonksiyon ile adet anlık güncellenir.
153     }
154
155 function tokenGonder(OlusturucuNFTadress, GondericiAdres, AliciAdres, TokenId, Miktar)
156     {
157         Satici NFT'yi aliciya talep edilen miktarda gönderir.
158     }
```