// SPDX-License-Identifier: MIT

pragma solidity ^0.8.2;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

import "@openzeppelin/contracts/utils/Counters.sol";

import "@imtbl/imx-contracts/contracts/Mintable.sol";

contract MyNFT is ERC721, ERC721URIStorage, Ownable, Mintable {

using Counters for Counters.Counter;

Counters.Counter private \_tokenIdCounter;

constructor(address \_imx) Mintable(\_imx) {}

constructor() ERC721("MyNFT", "MNFT") {}

function \_baseURI() internal pure override returns (string memory) {

return "https://i.ibb.co/kQCYnPS/photo-2021-11-29-18-44-36.jpg";

}

function safeMint(address to, string memory uri) public onlyOwner {

uint256 tokenId = \_tokenIdCounter.current();

\_tokenIdCounter.increment();

\_safeMint(to, tokenId);

\_setTokenURI(tokenId, uri);

}

function \_mintFor(address to, uint256 id, bytes memory) internal override {

require(to != address(0), "ERC721: mint to the zero address");

require(!\_exists(tokenId), "ERC721: token already minted");

\_beforeTokenTransfer(address(0), to, tokenId);

emit Transfer(address(0), to, tokenId);

}

function mint(address to) public returns (uint256) {

require(\_tokenIdCounter.current)

}

// The following functions are overrides required by Solidity.

function \_burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {

super.\_burn(tokenId);

}

function tokenURI(uint256 tokenId)

public

view

override(ERC721, ERC721URIStorage)

returns (string memory)

{

return super.tokenURI(tokenId);

}

}