// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";

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

contract VerifiedEventTickets is ERC721Enumerable, Ownable {

uint256 public ticketPrice;

uint256 public maxTickets;

uint256 public ticketsSold;

string public baseTokenURI;

mapping(uint256 => bool) public ticketUsed;

event TicketPurchased(address indexed buyer, uint256 indexed ticketId);

event TicketVerified(uint256 indexed ticketId);

constructor(

string memory name,

string memory symbol,

uint256 \_ticketPrice,

uint256 \_maxTickets,

string memory \_baseTokenURI

) ERC721(name, symbol) {

ticketPrice = \_ticketPrice;

maxTickets = \_maxTickets;

baseTokenURI = \_baseTokenURI;

}

function buyTicket() external payable {

require(ticketsSold < maxTickets, "All tickets sold");

require(msg.value >= ticketPrice, "Insufficient payment");

uint256 ticketId = ticketsSold;

ticketsSold++;

\_safeMint(msg.sender, ticketId);

emit TicketPurchased(msg.sender, ticketId);

}

function verifyTicket(uint256 ticketId) external onlyOwner {

require(\_exists(ticketId), "Ticket does not exist");

require(!ticketUsed[ticketId], "Ticket already verified");

ticketUsed[ticketId] = true;

emit TicketVerified(ticketId);

}

function isTicketUsed(uint256 ticketId) external view returns (bool) {

return ticketUsed[ticketId];

}

function withdrawFunds() external onlyOwner {

payable(owner()).transfer(address(this).balance);

}

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

return baseTokenURI;

}

}