Placeholder para 5.1_Termo_Emissao_Founders_Edition.pdf
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

/**
 * @title MAUAX Founders Edition NFT
 * @author GOS3 Team
 * @notice Este contrato gerencia os 100 Tokens Não Fungíveis (NFTs) de Governança
 * da "Founder's Edition" (MAUAX-G) do ecossistema MAUAX.
 * @dev A propriedade do contrato (ownership) deve ser transferida para um Gnosis Safe Multisig
 * imediatamente após a implantação. Este cofre representa o controle conjunto do
 * Consórcio Proponente (MEX Energia e Oliveira & Oliveira).
 */
contract MauaxFoundersNFT is ERC721, Ownable {
    using Strings for uint256;

    // A URL base para os metadados (arte, descrição) dos NFTs no IPFS.
    // Exemplo: "ipfs://Qm.../"
    string private _baseTokenURI;

    // Quantidade máxima e imutável de NFTs para esta série.
    uint256 public constant MAX_SUPPLY = 100;

    /**
     * @dev O construtor define o nome e o símbolo do token.
     * @param baseURI A URL base para os metadados no IPFS, a ser fornecida no momento da implantação.
     */
    constructor(string memory baseURI) ERC721("MAUAX Governance Token", "MAUAX-G") Ownable(msg.sender) {
        _baseTokenURI = baseURI;
    }

    /**
     * @notice Emite todos os 100 NFTs de uma vez para a carteira Cofre (Tesouraria).
     * @dev Esta função deve ser chamada apenas uma vez pelo deployer inicial. A propriedade
     * do contrato deve então ser transferida para a Multisig.
     * @param treasuryAddress O endereço da Gnosis Safe Multisig que servirá como custodiante inicial.

```
     */
    function mintAllToTreasury(address treasuryAddress) external onlyOwner {
        require(treasuryAddress != address(0), "ERC721: Mint to the zero address");
        // Garante que a emissão total só possa acontecer uma vez.
        require(totalSupply() == 0, "Initial mint has already been performed");

        for (uint256 i = 1; i <= MAX_SUPPLY; i++) {
            _safeMint(treasuryAddress, i);
        }
    }

    /**
     * @notice Retorna a URI dos metadados para um determinado token ID.
     * @dev Concatena a baseURI com o ID do token e a extensão .json, seguindo o padrão de
metadados.
     */
    function tokenURI(uint256 tokenId) public view override returns (string memory) {
        require(_exists(tokenId), "ERC721: URI query for nonexistent token");
        return string(abi.encodePacked(_baseTokenURI, tokenId.toString(), ".json"));
    }

    /**
     * @notice Permite que o dono do contrato (a Multisig) atualize a URL base dos metadados.
     * @dev Útil para revelação de arte ou atualizações futuras nos metadados.
     */
    function setBaseURI(string memory baseURI) external onlyOwner {
        _baseTokenURI = baseURI;
    }

    /**
     * @notice Transfere a propriedade do contrato para um novo endereço.
     * @dev Função CRÍTICA para transferir o controle administrativo para o Gnosis Safe Multisig.
     * Esta ação deve ser a primeira executada pelo deployer após a implantação.
     */
    function transferContractOwnership(address newOwner) external onlyOwner {
        transferOwnership(newOwner);
    }
}
```