```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.6.0) (proxy/Proxy.sol)

pragma solidity ^0.8.0;

/**
 * @dev This abstract contract provides a fallback function that delegates all calls to another contract using the EVM
 * instruction `delegatecall`. We refer to the second contract as the _implementation_ behind the proxy, and it has to
 * be specified by overriding the virtual {_implementation} function.
 *
 * Additionally, delegation to the implementation can be triggered manually through the {_fallback} function, or to a
 * different contract through the {_delegate} function.
 *
 * The success and return data of the delegated call will be returned back to the caller of the proxy.
 */
abstract contract Proxy {
    /**
     * @dev Delegates the current call to `implementation`.
     *
     * This function does not return to its internal call site, it will return directly to the external caller.
     */
    function _delegate(address implementation) internal virtual {
        assembly {
            // Copy msg.data. We take full control of memory in this inline assembly
            // block because it will not return to Solidity code. We overwrite the
            // Solidity scratch pad at memory position 0.
            calldatacopy(0, 0, calldatasize())

            // Call the implementation.
            // out and outsize are 0 because we don't know the size yet.
            let result := delegatecall(gas(), implementation, 0, calldatasize(), 0, 0)

            // Copy the returned data.
            returndatacopy(0, 0, returndatasize())

            switch result
            // delegatecall returns 0 on error.
            case 0 {
                revert(0, returndatasize())
            }
            default {
                return(0, returndatasize())
            }
        }
    }

    /**
     * @dev This is a virtual function that should be overridden so it returns the address to which the fallback function
     * and {_fallback} should delegate.
     */
    function _implementation() internal view virtual returns (address);

    /**
     * @dev Delegates the current call to the address returned by `_implementation()`.
     *
     * This function does not return to its internal call site, it will return directly to the external caller.
     */
    function _fallback() internal virtual {
        _beforeFallback();
        _delegate(_implementation());
    }

    /**
     * @dev Fallback function that delegates calls to the address returned by `_implementation()`. Will run if no other
     * function in the contract matches the call data.
     */
    fallback() external payable virtual {
        _fallback();
    }

    /**
     * @dev Fallback function that delegates calls to the address returned by `_implementation()`. Will run if call data
     * is empty.
     */
    receive() external payable virtual {
        _fallback();
    }

    /**
     * @dev Hook that is called before falling back to the implementation. Can happen as part of a manual `_fallback`
     * call, or as part of the Solidity `fallback` or `receive` functions.
     *
     * If overridden should call `super._beforeFallback()`.
     */
    function _beforeFallback() internal virtual {}
}
```