ABI

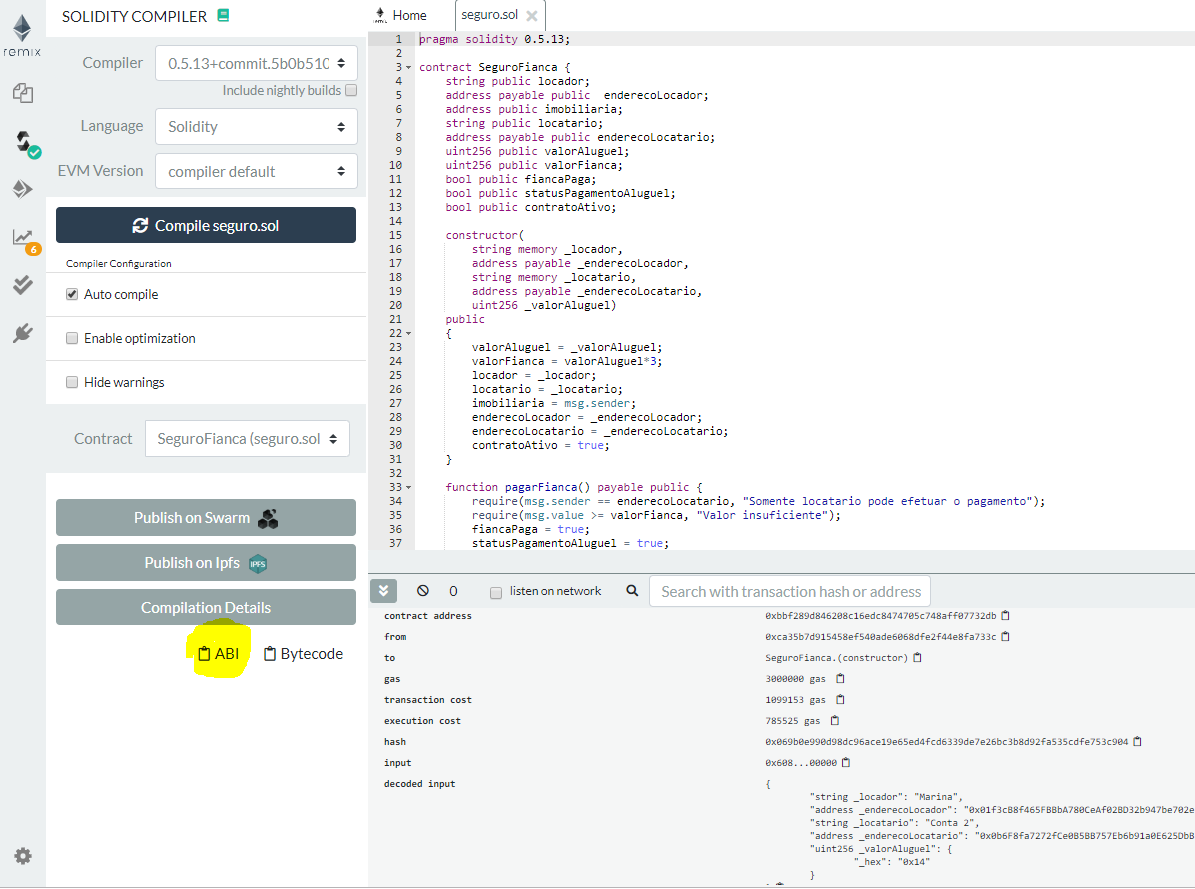
Solidity

HTML

JS

ABI – É a interface em Java script do Smart Contract. O Remix fornece o ABI na janela de compilação.

No caso do Smart Contract feito no Remix, temos uma opção de abrir o ABI, o qual conversará com o JAVA:



Segue o ABI do contrato de Seguro Fiança no Remix:

[

{

"constant": false,

"inputs": [],

"name": "fimDoContrato",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"constant": false,

"inputs": [

{

"internalType": "string",

"name": "\_descricao",

"type": "string"

}

],

"name": "mudaStatusPagamento",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"constant": false,

"inputs": [],

"name": "pagarFianca",

"outputs": [],

"payable": true,

"stateMutability": "payable",

"type": "function"

},

{

"inputs": [

{

"internalType": "string",

"name": "\_locador",

"type": "string"

},

{

"internalType": "address payable",

"name": "\_enderecoLocador",

"type": "address"

},

{

"internalType": "string",

"name": "\_locatario",

"type": "string"

},

{

"internalType": "address payable",

"name": "\_enderecoLocatario",

"type": "address"

},

{

"internalType": "uint256",

"name": "\_valorAluguel",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "nonpayable",

"type": "constructor"

},

{

"constant": true,

"inputs": [],

"name": "contratoAtivo",

"outputs": [

{

"internalType": "bool",

"name": "",

"type": "bool"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "enderecoLocador",

"outputs": [

{

"internalType": "address payable",

"name": "",

"type": "address"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "enderecoLocatario",

"outputs": [

{

"internalType": "address payable",

"name": "",

"type": "address"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "fiancaPaga",

"outputs": [

{

"internalType": "bool",

"name": "",

"type": "bool"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "imobiliaria",

"outputs": [

{

"internalType": "address",

"name": "",

"type": "address"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "locador",

"outputs": [

{

"internalType": "string",

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "locatario",

"outputs": [

{

"internalType": "string",

"name": "",

"type": "string"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "statusPagamentoAluguel",

"outputs": [

{

"internalType": "bool",

"name": "",

"type": "bool"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "valorAluguel",

"outputs": [

{

"internalType": "uint256",

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "valorFianca",

"outputs": [

{

"internalType": "uint256",

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

}

]

Para fazermos o vínculo do Smart Contract para o HTML, em seguro.html, primeiro importamos a biblioteca do Ethers.js:

<script charset="utf-8" src="https://cdn.ethers.io/scripts/ethers-v4.min.js" type="text/javascript"></script>

E depois importamos o ABI do Smart Contract:

<script src="javascript/abi.js"></script>

Depois, vinculamos ao Ether JS o “provedor de conexão”, o “ABI” tudo o que tem público no SC, e o “endereço do SC na rede Ethereum correspondente.

var enderecoContrato = "0x891710b3475b3076Dd1F9C6a6A9b1f87BA5e0517";

var provedor = ethers.getDefaultProvider('rinkeby');

var contrato = new ethers.Contract(enderecoContrato, abiContrato, provedor);

provedor de conexão

 ABI” tudo o que tem público no SC

 endereço do SC na rede Ethereum correspondente

Ethers JS

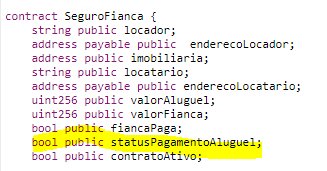
O Status de pagamento do Aluguel é uma função do SC:

contrato.statusPagamentoAluguel()

.then( (resultado) => {

campoStatus.innerHTML = resultado;

})



PROMISSES:

Funcao ( )

.then (função a ser chamada em caso de sucesso – em geral aqui vem uma arrow function)

.catch (função a ser chamada em caso de erro – em geral aqui vem uma arrow function também)

O “Then” vem de uma promessa de resultado do JS (certo-resultado ou errado –catch). Leia-se: “Quando eu trouxer da Blockchain (Status do Pagamento do Aluguel), então vou te trazer a função “resultado” no HTML (campoStatus.innerHTML), e a resposta pode ser um resultado ou ser um erro”

Antes era da seguinte forma:

**Function Xpto (Resultado) {**

**}**

Então, usa-se o Array function, que é um jeito mais curto de escrever o que está acima:

**(resultado) => {**

**}**

.catch( (err) => {

console.error(err);

campoStatus.innerHTML = err;

});

Caso dê algum problema, o “catch” buscará todos as funções que não funcionaram, e ele apresenta o erro (err).