

TRABALHO SMART CONTRACTS DEVELOPMENT

# PROFESSOR: GLAUBER DUARTE MONTEIRO

# CURSO: MBA EM BLOCKCHAIN – DEVELOPMENT & TECHNOLOGIES

# ALUNOS:

# BEATRIZ GRANCE RINN – RM 334670

# RAFAEL M STOCCO – RM 334725

# WAGNER M DOROTEIO – RM 333253

# SMART CONTRACT – REGISTRO DE VEICULOS

## Objetivo:

Registrar na rede ethereum um Smart Contract de registro de veículos com as seguintes informações;

* Tipo de Veiculo
* Modelo
* Cor
* Placa
* Renavan
* Valor

Smart Contract, possibilitará a consulta do veiculo pela placa e também a consulta de todos os veículos

registrados.

O veiculo será registrado na rede se a placa possuir 8 caracteres. Há uma validação que verifica se a placa possui o tamanho de 8 caracteres. Preferencia utilizar a seguinte mascara para a placa(ABC-9999).

## Critérios:

O Smart Contract deve atender os seguintes critérios;

* Desenvolver projeto de Smart Contract(s) Ethereum utilizando Truffle Suite ou Remix IDE
* Smart Contract deve conter pelo menos um método pago e um não pago
* Smart Contract deve conter pelo menos um método não público
* Smart Contract deve conter pelo menos uma struct, um mapping e um array
* Deploy em uma rede de teste (Ropsten, Kovan)
* Smart Contract(s) verificado(s) https://ropsten.etherscan.io/verifyContract
* ENS para o(s) Smart Contract(s)
* Projeto documentado e entregue no github

## Smart Contract : RegistraVeiculo2.sol

O contrato foi construído sob a plataforma solidity.

// SPDX-License-Identifier: MIT

pragma experimental ABIEncoderV2;

pragma solidity ^0.6.9;

contract RegistraVeiculo {

address private owner;

constructor () public {

owner = msg.sender;

}

modifier onlyOwner() {

require(msg.sender == owner, "Voce nao e o dono do contrato");

\_;

}

function contractAddress() public payable onlyOwner {

uint256 balance = address(this).balance;

(bool success, ) = msg.sender.call{value: balance}("");

require(success, "Erro na transferencia");

}

struct Veiculo {

string tipoveiculo;

string modelo;

string cor;

string placa;

string renavan;

uint valor;

}

mapping(address => Veiculo[]) public veiculos;

mapping(string => Veiculo) public findPlaca;

function salvaVeiculo( string memory \_tipoveiculo, string memory \_modelo, string memory \_cor,

string memory \_placa,string memory \_renavan,uint \_valor) public payable {

bool \_validaplaca = validaplaca(\_placa);

if (\_validaplaca==false){

require(\_validaplaca == false, "Placa do veiculo invalida, deve possiur 8 caracteres");

}else{

Veiculo memory \_veiculo = Veiculo(\_tipoveiculo, \_modelo, \_cor,\_placa,\_renavan,\_valor);

veiculos[msg.sender].push(\_veiculo);

findPlaca[\_placa] = \_veiculo;

}

}

function getVeiculos() public view returns(Veiculo[] memory) {

Veiculo[] memory retveiculos = veiculos[msg.sender];

return retveiculos;

}

function validaplaca(string memory \_placa) private pure returns (bool) {

if( bytes(\_placa).length == 8){

return true;

}else{

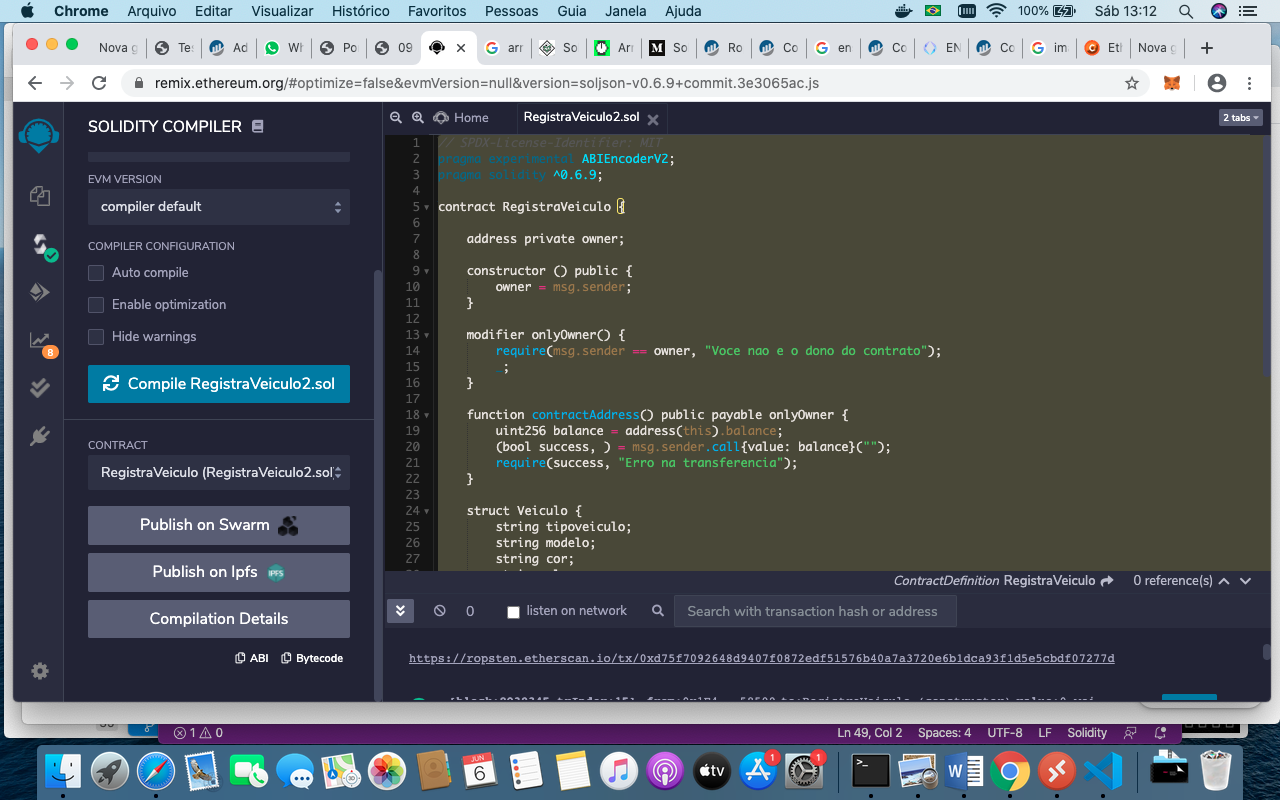
return false;

}

}

}

## Desenvolver projeto de Smart Contract(s) Ethereum utilizando Truffle Suite ou Remix IDE



* Smart Contract deve conter pelo menos um método pago e um não pago

Método Pago

function salvaVeiculo( string memory \_tipoveiculo, string memory \_modelo, string memory \_cor,

string memory \_placa,string memory \_renavan,uint \_valor) public payable {

Veiculo memory \_veiculo = Veiculo(\_tipoveiculo, \_modelo, \_cor,\_placa,\_renavan,\_valor);

veiculos[msg.sender].push(\_veiculo);

findPlaca[\_placa] = \_veiculo;

}

Método Não pago

function getVeiculos() public view returns(Veiculo[] memory) {

Veiculo[] memory retveiculos = veiculos[msg.sender];

return retveiculos;

}

* Smart Contract deve conter pelo menos um método não público
* function validaplaca(string memory \_placa) private pure returns (bool) {
* if( bytes(\_placa).length == 8){
* return true;
* }else{
* return false;
* }
* Smart Contract deve conter pelo menos uma struct, um mapping e um array

Struct

struct Veiculo {

string tipoveiculo;

string modelo;

string cor;

string placa;

string renavan;

uint valor;

}

Mapping

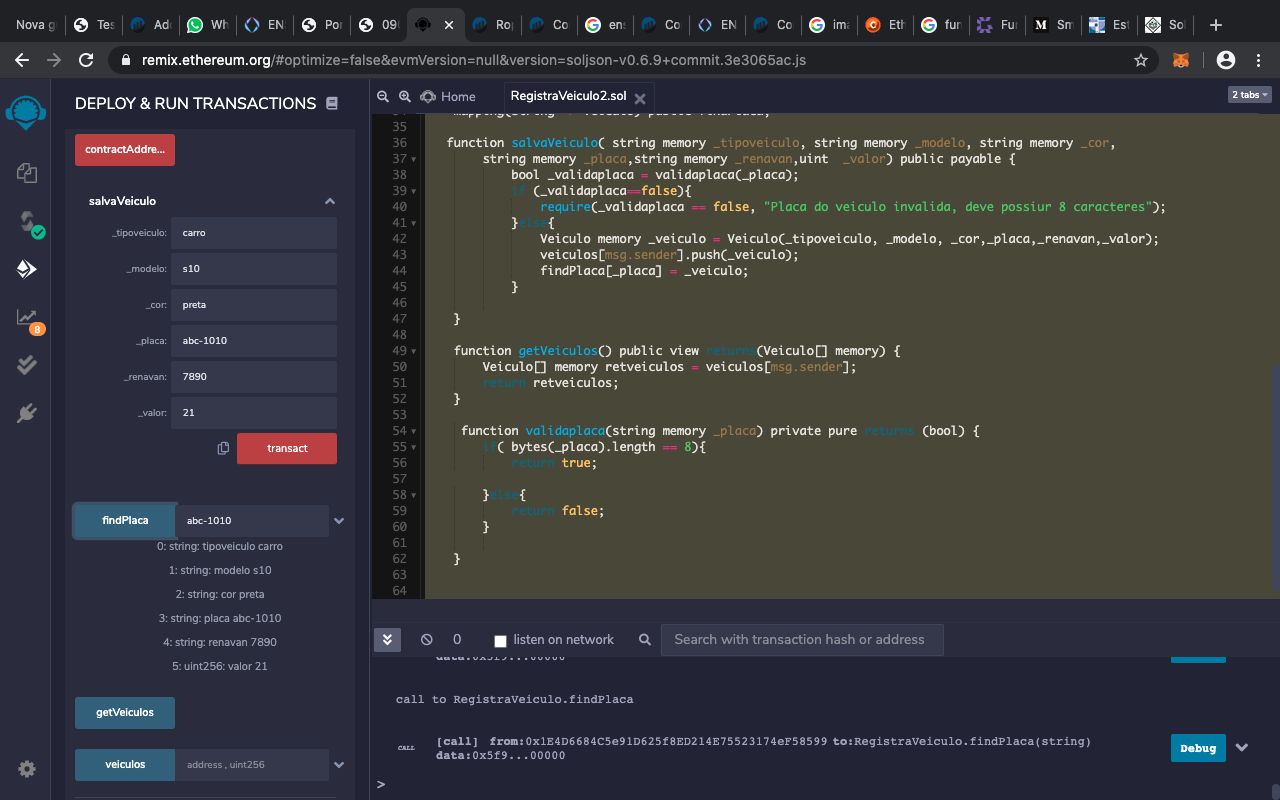
mapping(address => Veiculo[]) public veiculos;

mapping(string => Veiculo) public findPlaca;

Array

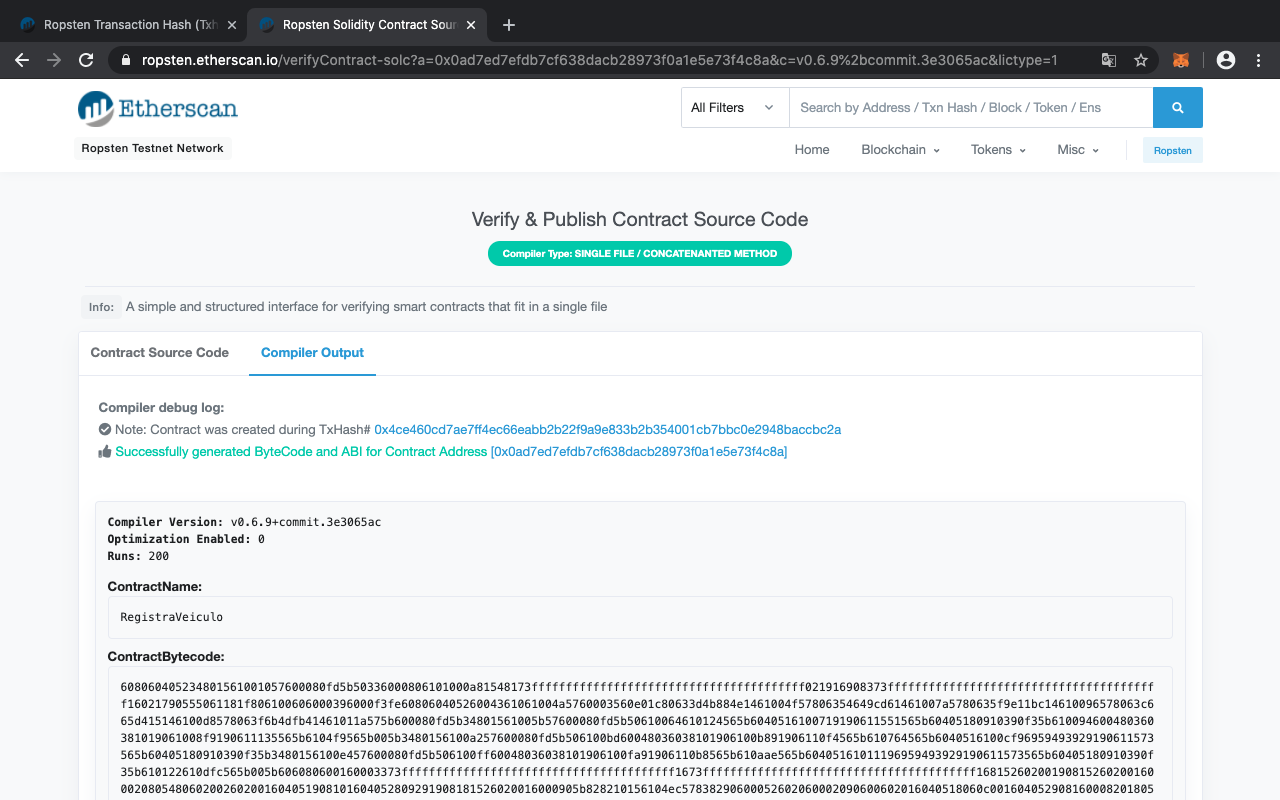
mapping(address => Veiculo[]) public veiculos;

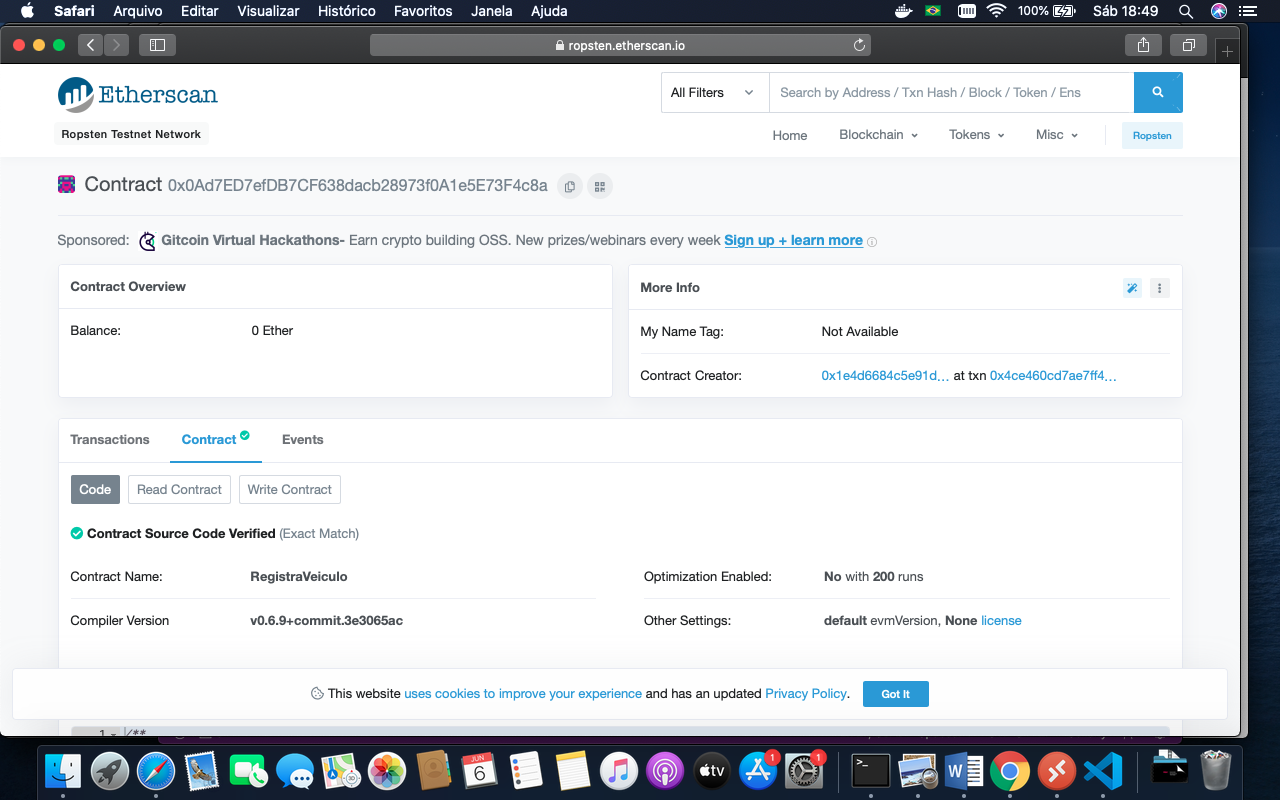
* Deploy em uma rede de teste (Ropsten, Kovan)



* Smart Contract(s) verificado(s) <https://ropsten.etherscan.io/verifyContract>

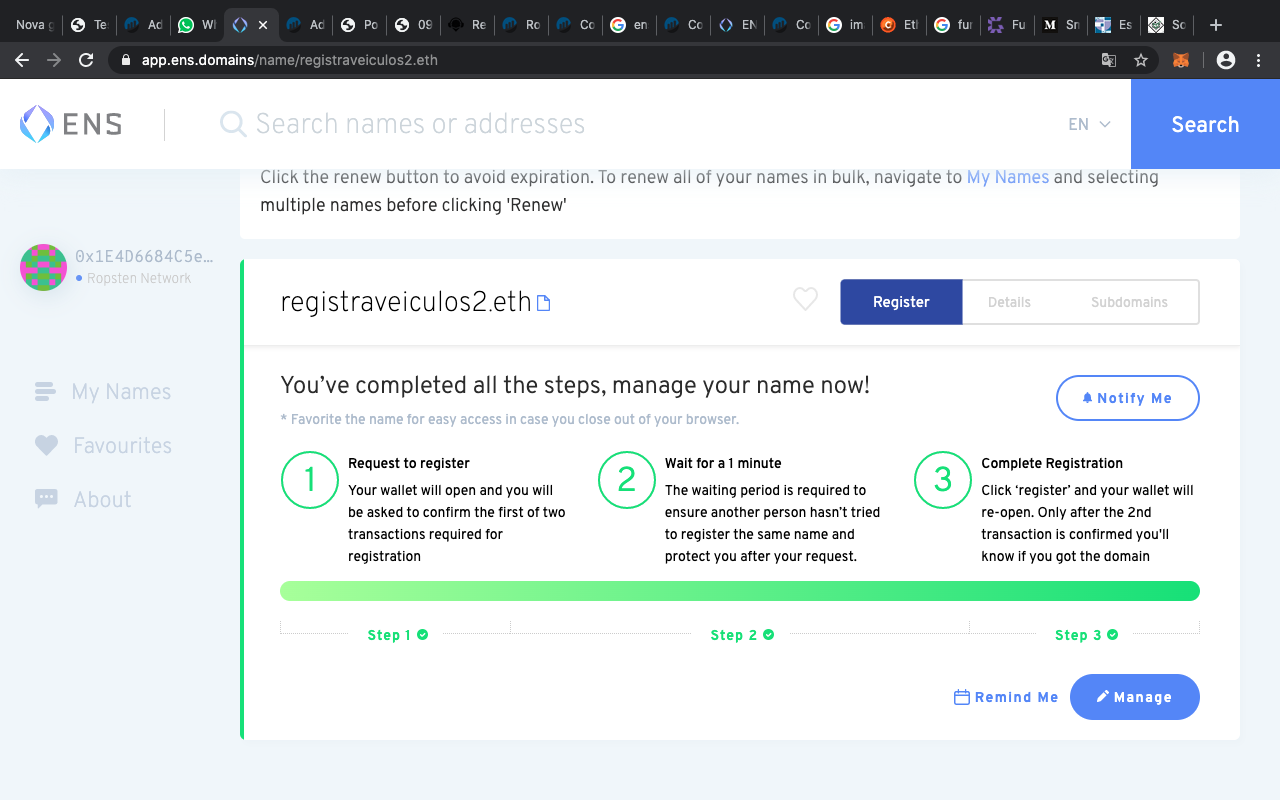
<https://ropsten.etherscan.io/verifyContract-solc?a=0x0ad7ed7efdb7cf638dacb28973f0a1e5e73f4c8a&c=v0.6.9%2bcommit.3e3065ac&lictype=1>

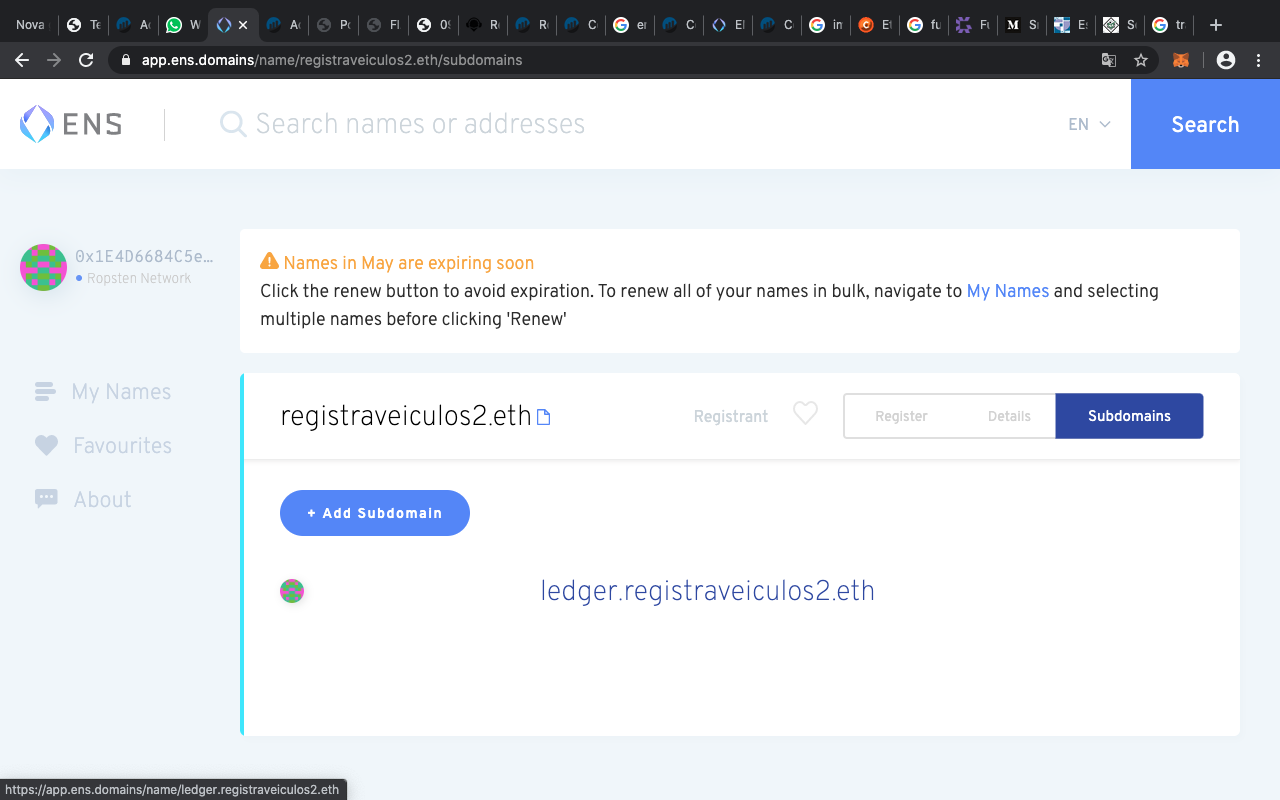




* ENS para o(s) Smart Contract(s)

registraveiculos2.eth





ledger.registraveiculos2.eth

