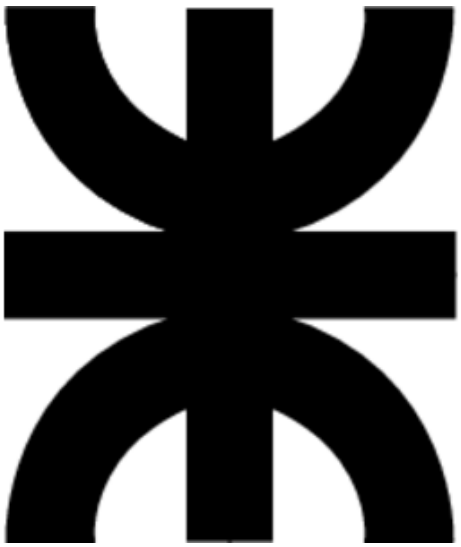


Universidad Tecnológica Nacional

Facultad Regional Villa María

Trabajo Práctico

Cátedra:

- Tecnologia de Redes Avanzada.

Docente:

- Ing Fernando Boiero

Alumno:

- Durelli, Nicolas-13037-nicolasarieldurelli@gmail.com

2021

Contrato

Creación

Como primer paso creamos un archivo llamado “Adoption.sol” que será el que tendrá la funcionalidad del contrato.

Una vez creado el archivo le incorporamos el siguiente código:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.7.3;

contract Adoption {
	address[16] public adopters;

	function adopt(uint petId) public returns(uint) {
		require(petId >= 0 && petId <= 15);
		adopters[petId] = msg.sender;
		return petId;
	}

	function getAdopters() public view returns(address[16] memory){
		return adopters;
	}

	function getAdopterOf(uint petId) public view returns (address) {
		return adopters[petId];
	}
}
```

Explicación

El contrato tiene un array de 16 (0-15) direcciones llamado adopters, que contiene las direcciones de quienes son lo que adoptan las mascotas, también posee 3 funciones:

- La función adopt() la cual recibe un parámetro de entrada, esta función permite “adoptar” a la mascota con el id ingresado, el cual debe estar entre 0 y 15, si esto se cumple se registra su dirección en la ubicación correspondiente en el array, esta función tiene costo.
- La función getAdopters() devuelve el array de direcciones de los “dueños” de las mascotas, esta función no tiene costo
- La función getAdoptersOf() la cual recibe un parámetro de entrada, esta función devuelve la dirección de la mascota ingresada como parámetro

Compilación

Una vez creado el contrato, como siguiente paso lo compilamos con el comando *“npm run build”.*

Como resultado se obtuvo el siguiente resultado:

> npm run clean && npm run compile
> npx hardhat clean
> npx hardhat compile

Compiling 1 file with 0.7.3
Generating typings for: 1 artifacts in dir: typechain for target: ethers-v5
Successfully generated 5 typings!
Compilation finished successfully

Test

Creación

Creamos un archivo llamado *"Adoption.spec.ts"* que será el que tendra el codigo del test para el contrato *"Adoption.sol*".

Con el archivo creado le incorporamos el siguiente código

```
import { expect } from "chai"
import { ethers } from "hardhat";

let signers;
let owner: any;
let alice: any;
let adoption: any;
const emptyAddress = '0x0000000000000000000000000000000000000000';

describe("Adoption", function () {
   beforeEach(async () => {
      signers = await ethers.getSigners();
      owner = signers[0];
      alice = signers[1];
      const Adoption = await ethers.getContractFactory("Adoption");
      adoption = await Adoption.deploy();
      await adoption.deployed();
   });
   it("should start without adopters", async () => {
      const adopters = await adoption.getAdopters();
      expect(adopters.every((address: string) => address === emptyAddress)).to.equal(true);
   });
   it('should adopt if id is between 0 and 15', async () => {
      await adoption.connect(alice).adopt(1).wait;
      const adopted = await adoption.connect(alice).getAdopterOf(1);
      expect(adopted).to.equal(alice.address);
```

```
    });
    it('should not adopt if id is not between 0 and 15', async () => {
        expect(adoption.connect(alice).adopt(16)).to.be.reverted;
    });
    it('should getAdopterOf if id is between 0 and 15', async () => {
        const adopter = await adoption.connect(alice).getAdopterOf(1);
        expect(adopter).to.equal(emptyAddress);
    })
    it('should not getAdopterOf if id is not between 0 and 15', async () => {
        expect(adoption.connect(alice).getAdopterOf(16)).to.be.reverted;
    })
});
```

Explicación

El test está compuesto por un grupo de pruebas, las cuales son:

- consultar adoptantes con el contrato sin direcciones
- probar adoptar mascotas con el id entre 0 y 15
- probar adoptar mascotas con el id fuera del rango entre 0 y 15
- consultar adoptante con id entre 0 y 15
- consultar adoptante con id fuera del rango de 0 y 15

Resultado

Una vez creado el test para el contrato, ejecutamos el comando *“npm test”*, y como resultado obtuvimos:

```
> npx hardhat test

No need to generate any newer typings.

  Adoption
    √ should start without adopters
    √ should adopt if id is between 0 and 15
    √ should not adopt if id is not between 0 and 15
    √ should getAdopterOf if id is between 0 and 15
    √ should not getAdopterOf if id is not between 0 and 15

  5 passing (18s)
```

Deployado

Rinkeby

Para configurar la red de ethereum de pruebas “Rinkeby” en el archivo hardhat.config.ts agregamos el siguiente código

```
. rinkeby: {
    url: "https://eth-rinkeby.alchemyapi.io/v2/4_s8qQ4ie8MKleCqfUfYPTJQUmZ1LjIZ",
    accounts: [`0x`],
    gasPrice: 8000000000
  }
```

Donde en “url” es nuestra dirección de alchemy, y “accounts” es nuestra clave privada de la billetera

Script

Una vez configurada la red “Rinkeby” creamos un archivo llamado deploy.ts, este archivo se va a encargar de deployar contratos, este archivo contiene el siguiente código:

```
import { ethers } from "hardhat";

async function main() {
  const Adoption = await ethers.getContractFactory("Adoption")
  const contract = await Adoption.deploy()
  console.log("Contract deployed to address:", contract.address)
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error)
  process.exit(1)
})
```

Este script va a tomar el contrato lo va a deployar y va a devolvernos la dirección del mismo.

Deployado

Una vez con el contrato creado, testeado, con la red configurada y creado el script necesario para deployar, procedemos a ejecutar el siguiente comando:

```
npx hardhat run scripts/deploy.ts --network rinkeby
```

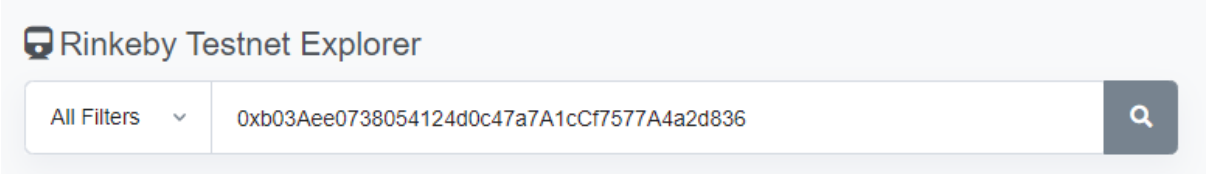
Como resultado obtenemos la dirección del contrato.

```
Contract deployed to address: 0xb03Aee0738054124d0c47a7A1cCf7577A4a2d836
```

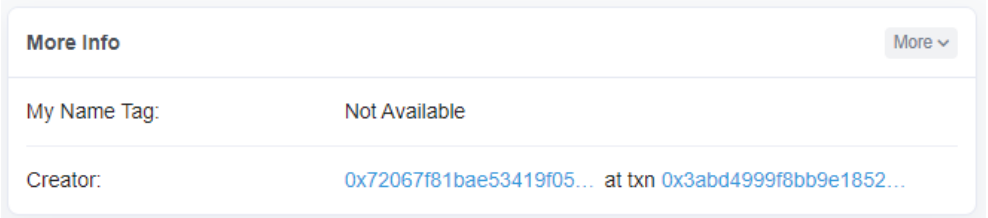
Interacción con el contrato

Etherscan

Si al ingresar a "https://rinkeby.etherscan.io/" en su buscador pegamos la dirección del contrato que acabamos de deployar "0xb03Aee0738054124d0c47a7A1cCf7577A4a2d836" vamos a obtener la información del mismo

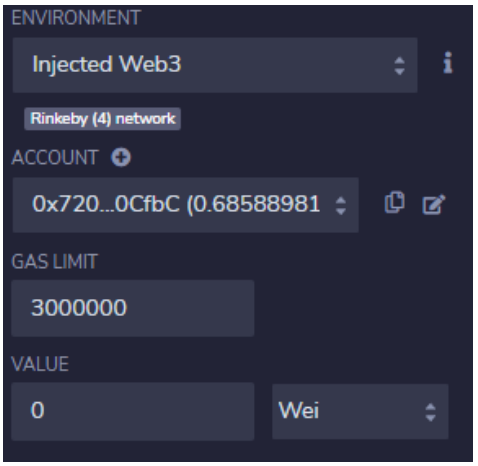


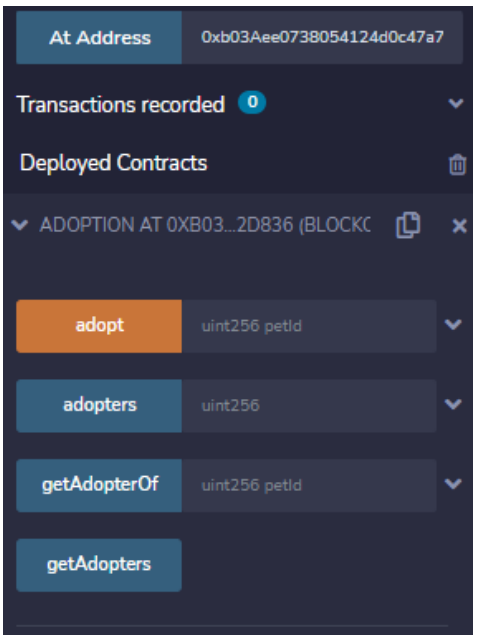
Como la dirección del creador del contrato, o el Hash de la transacción cuando se creó el contrato



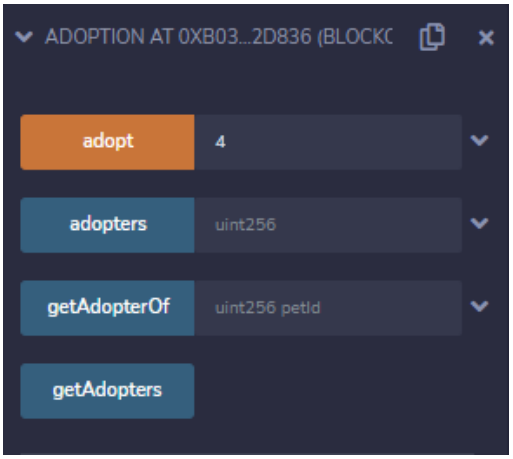
Remix ethereum IDE

También podemos interactuar con el contrato a través de "https://remix.ethereum.org/", ingresando su dirección y conectando una cuenta (en este caso a través de metamask).





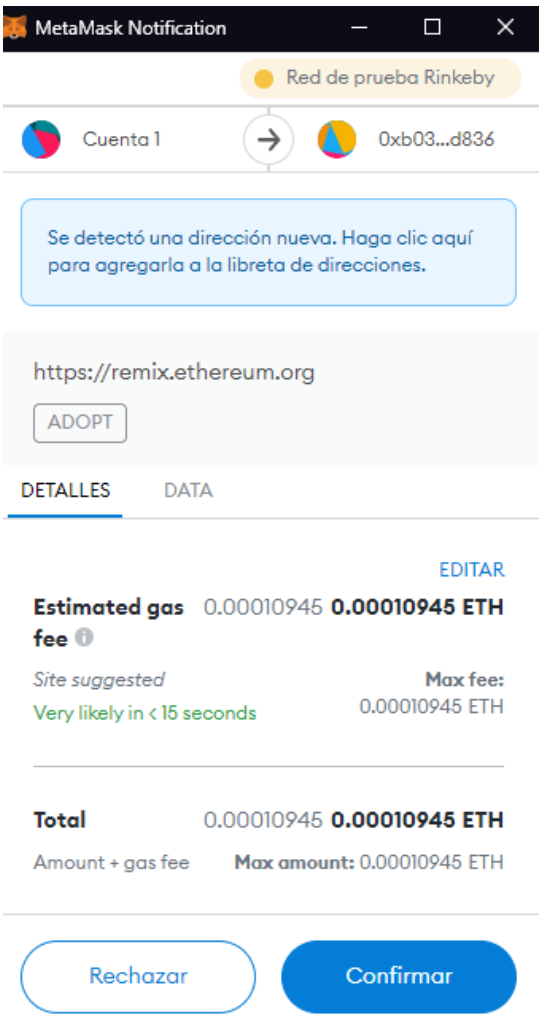
Podemos adoptar una mascota (por ejemplo la N° 5), entonces escribimos su id (el id es el N° 4 ya que empieza en 0), y apretamos adoptar



En ese momento podemos ver que en la consola sale el siguiente mensaje

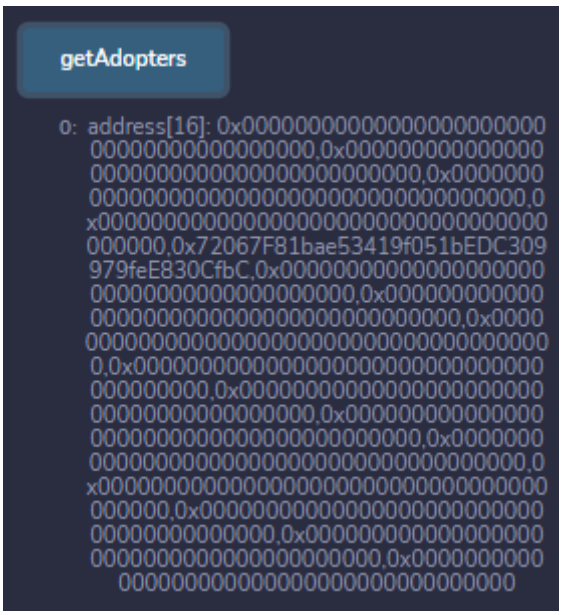
```
transact to Adoption.adopt pending ...
```

En este caso MetaMask nos avisa de la transacción con el siguiente mensaje, permitiéndonos aceptar o rechazar adoptar a la mascota N° 5, indicando el fee de la transacción



Una vez aceptada la transacción en la consola de Remix aparece como exitosa la transacción

Podemos verificar que fue exitosa con las otras funciones del contrato, al activar la función *"getAdopters"*, en este caso, nos devuelve qué direcciones son las dueñas de las mascotas, podemos ver que la N° 5 tiene nuestra dirección, mientras que las otras tienen direcciones con 0´s.



en este caso la transacción se realiza sin ningún fee, sin metamask.