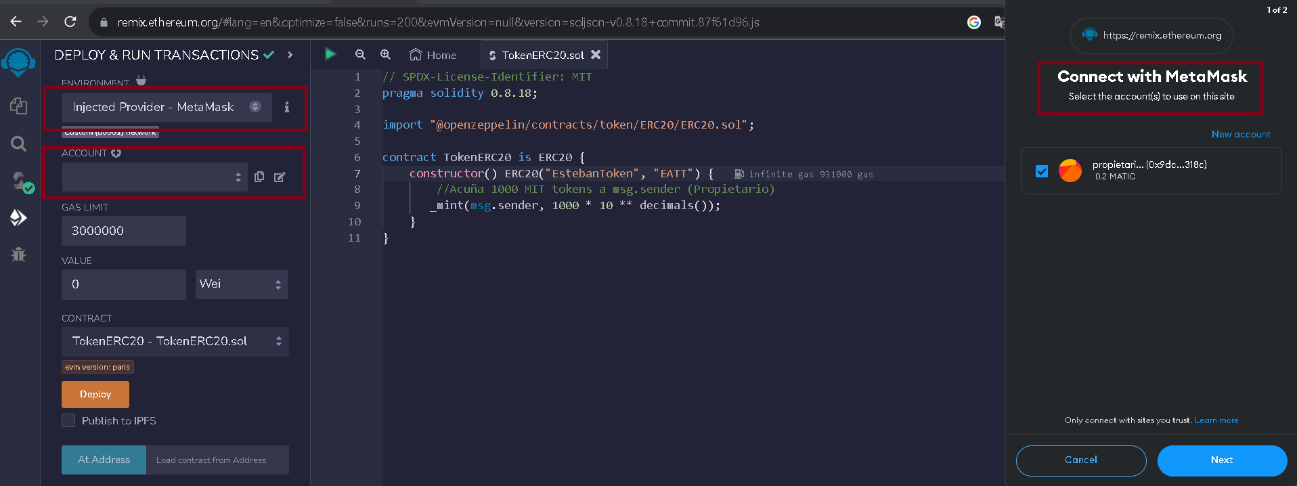
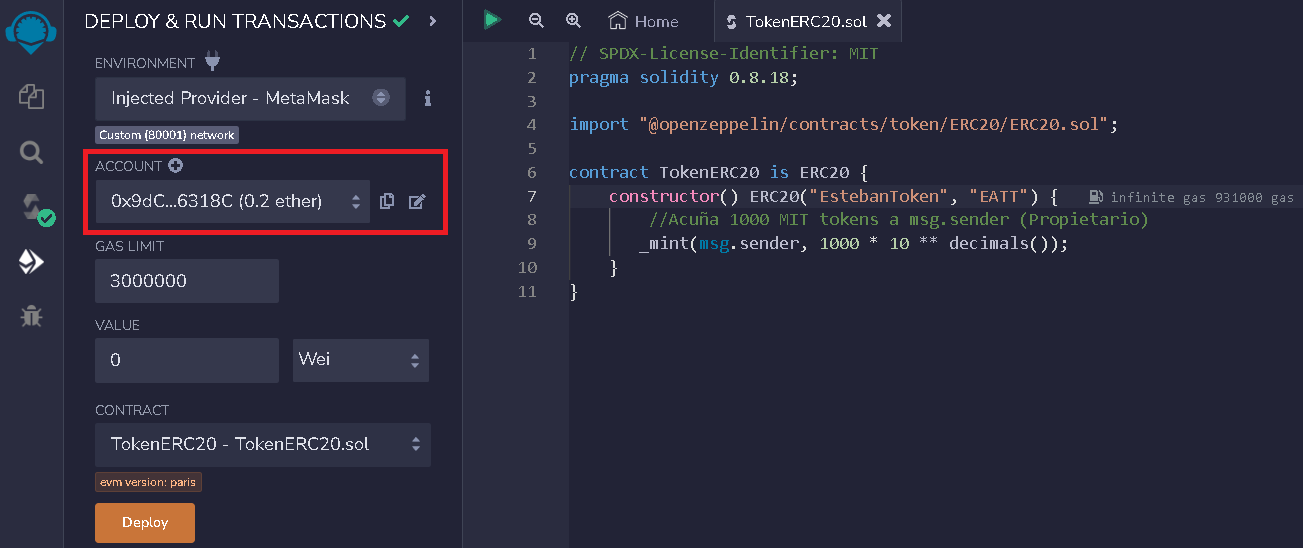
**LABORATORIO: Interactuando con los métodos del estándar ERC20**

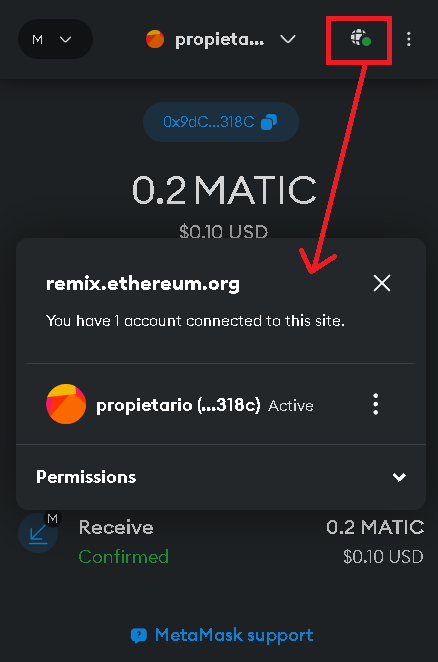
Vamos a usar esta billetera para deployar mi contrato en Remix:



Captura de pantalla de un celular

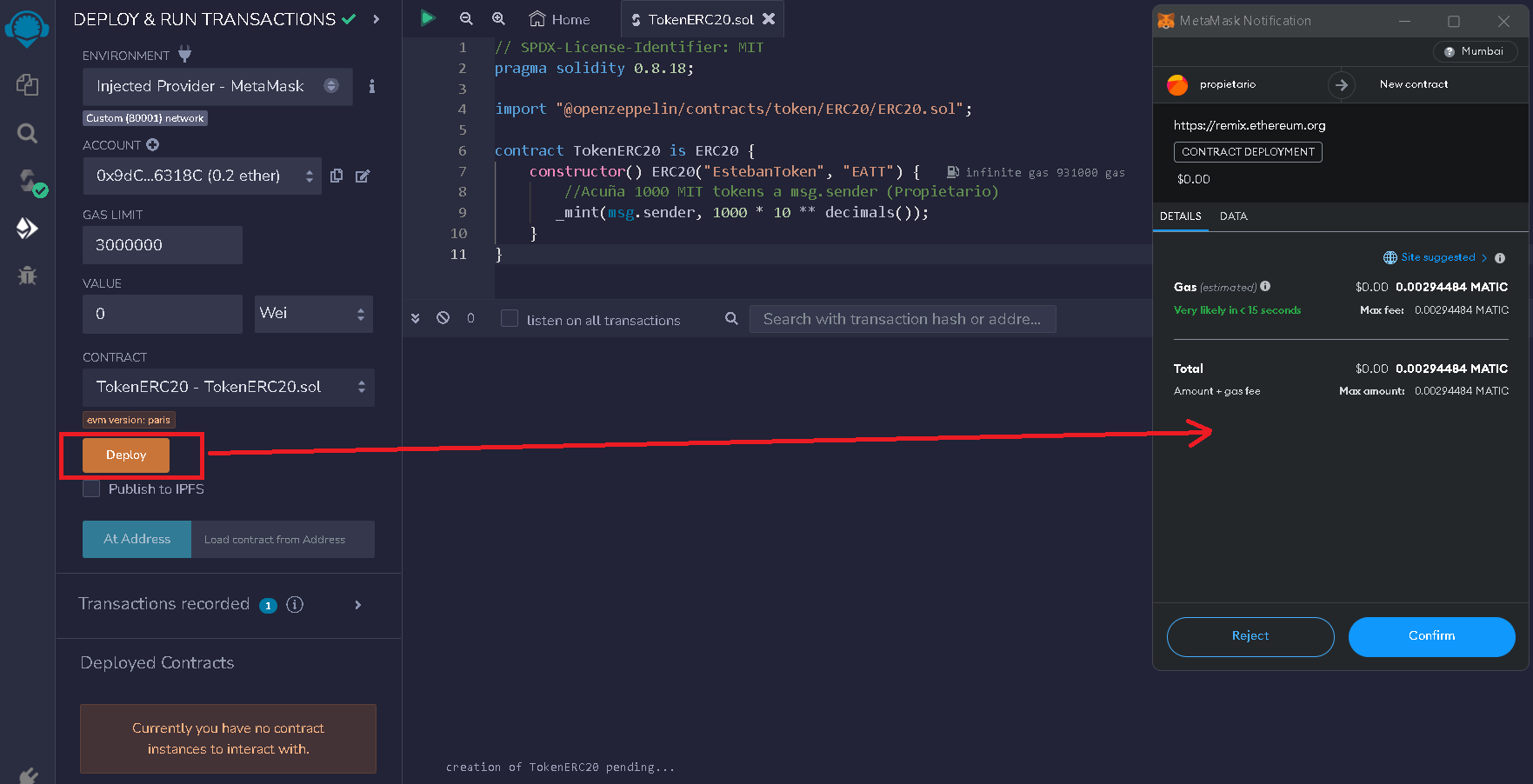
Descripción generada automáticamente





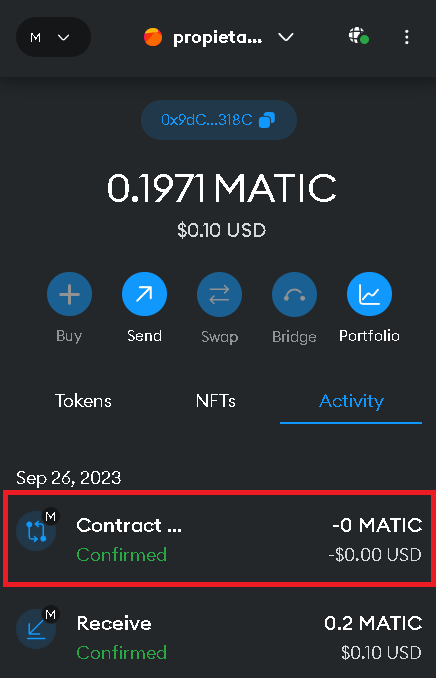
Interfaz de usuario gráfica, Texto, Aplicación

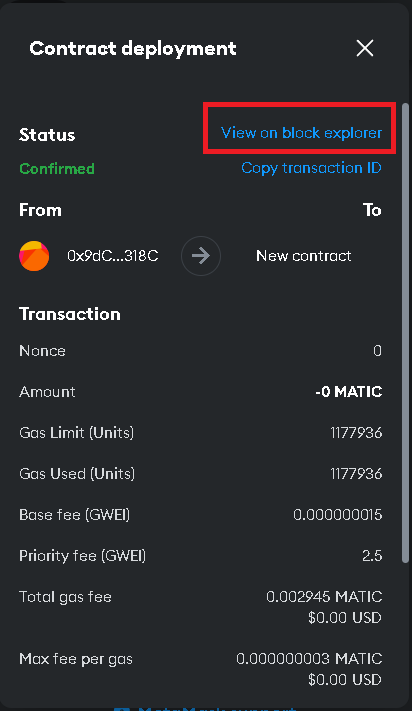
Descripción generada automáticamente

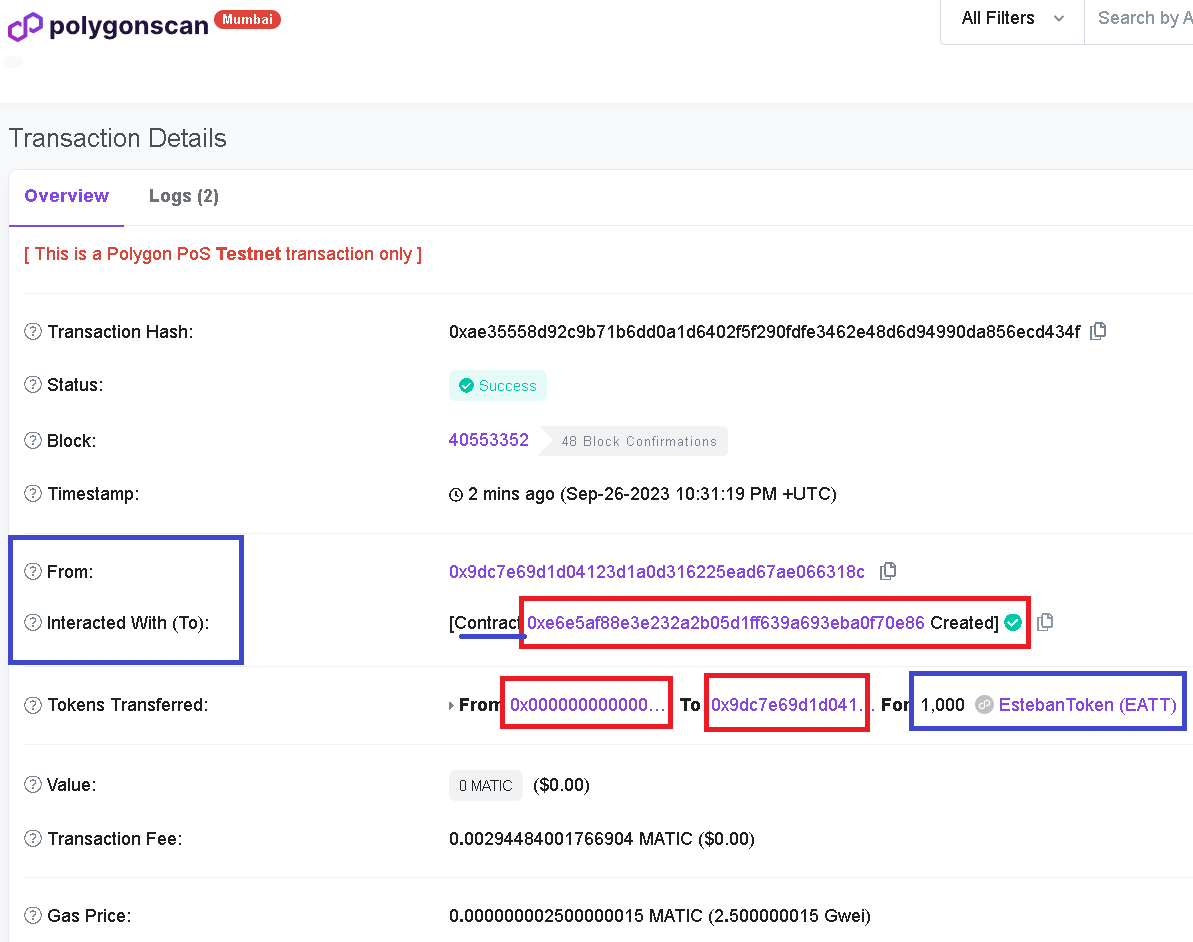


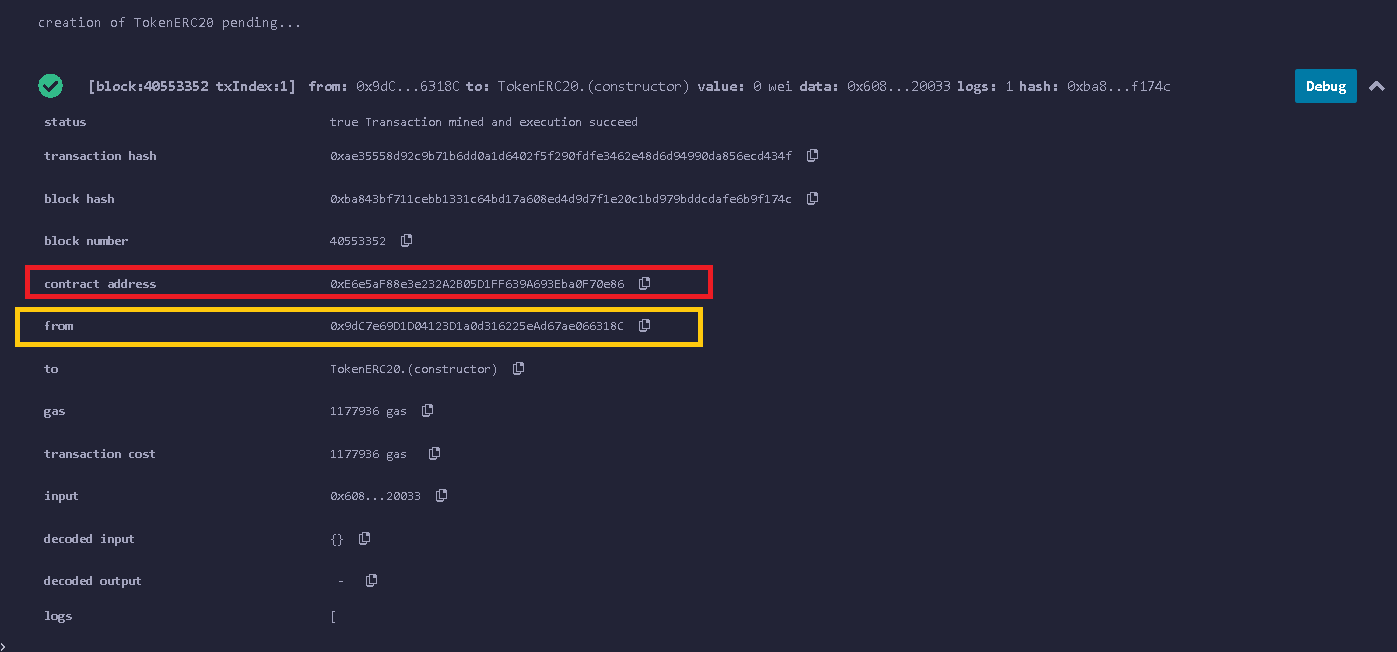
Captura de pantalla de un celular

Descripción generada automáticamente

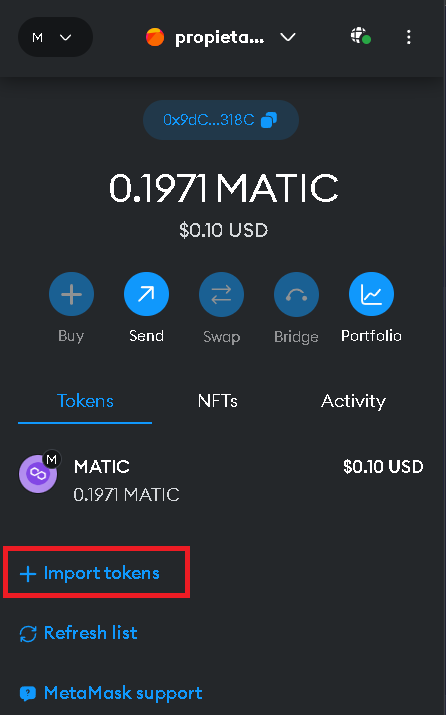








Para importar los tokens recibidos (solo visualmente en metamask):



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se pone el address del contrato recientemente deployado

Address del contrato: 0xE6e5aF88e3e232A2B05D1FF639A693Eba0F70e86

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Automáticamente te jala el Token Symbol y el Token decimal. Le damos en Add custom token:

Captura de pantalla de un celular

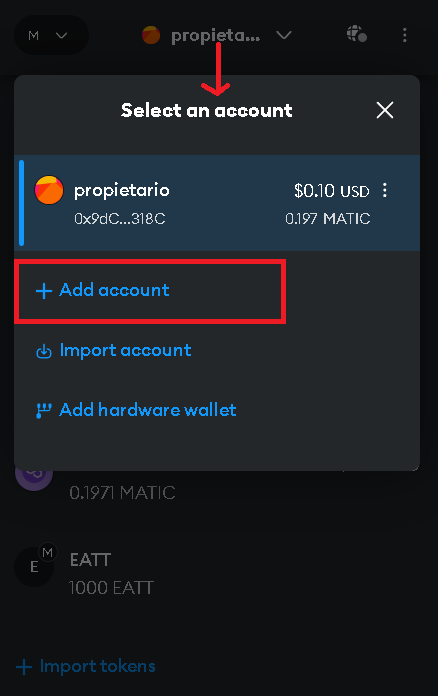
Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Pantalla de celular con imagen de la pantalla de un celular

Descripción generada automáticamente



Captura de pantalla de un teléfono celular

Descripción generada automáticamente

Después de crear “gastador” y “destinatario”:

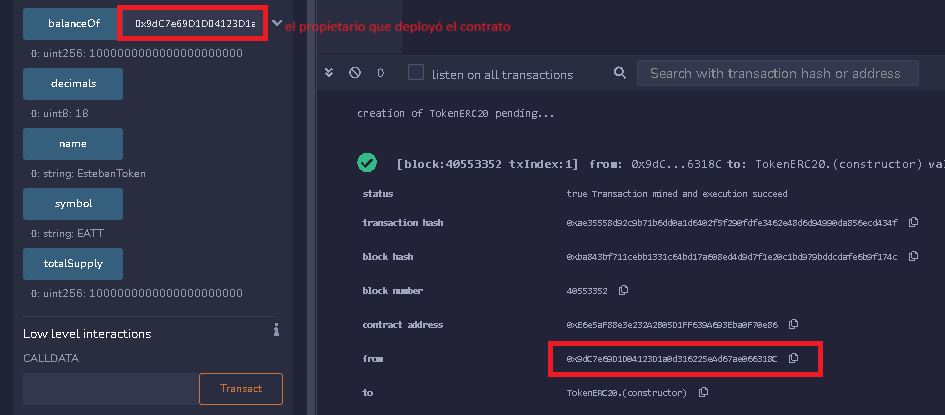
Address propietario: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

Address destinatario: 0x1b38Ba4AfcbA57f2F9E8827D29Ad7945e0b75a30

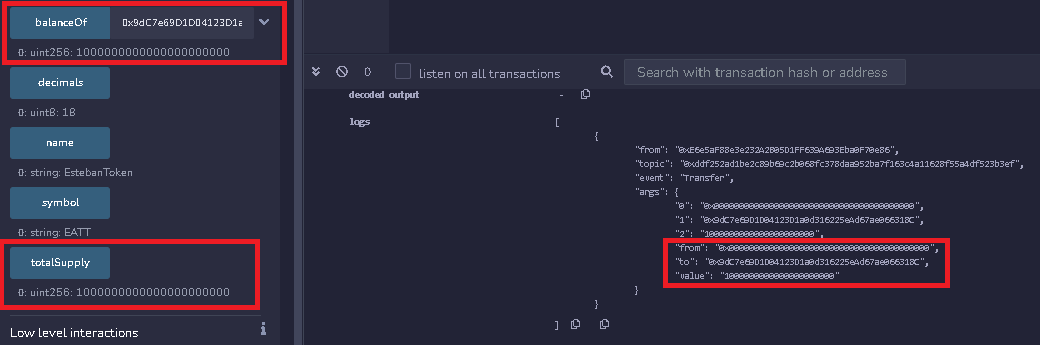
Address gastador: 0xA3a786FF773524f2aff0edED90230616A3a26de1

Captura de pantalla de un teléfono celular

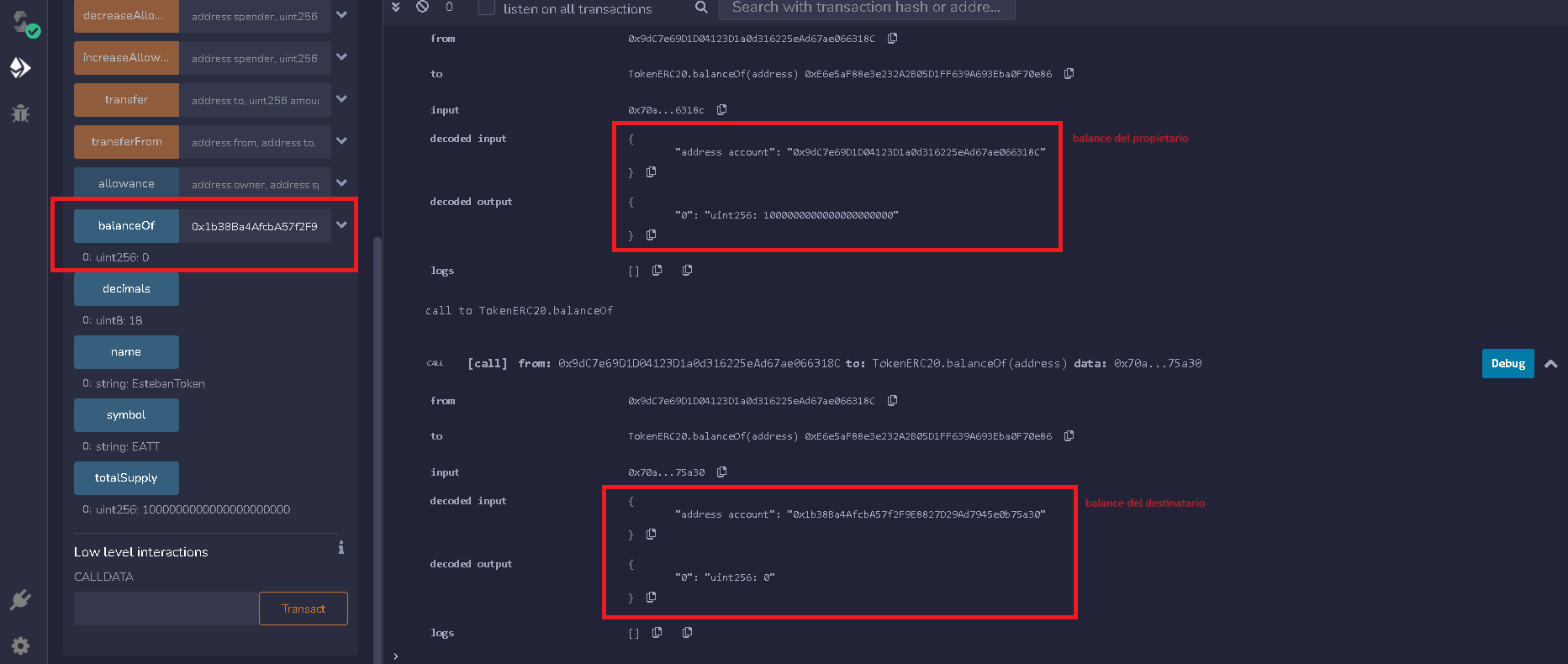
Descripción generada automáticamente

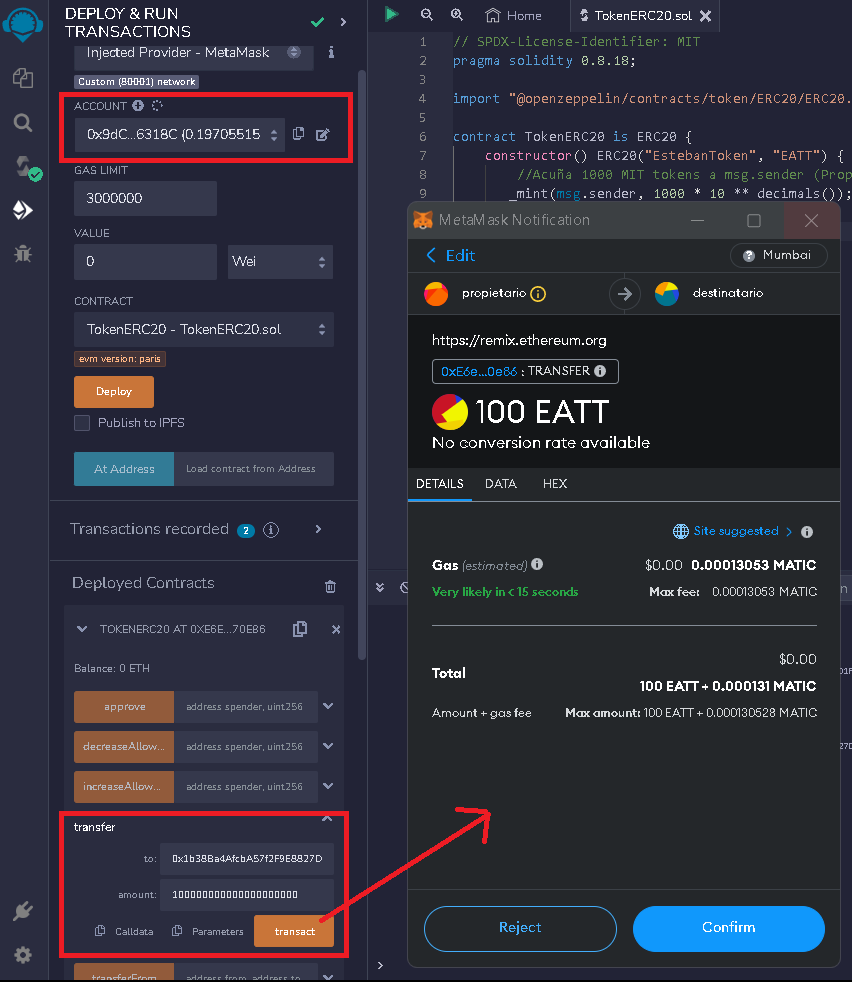


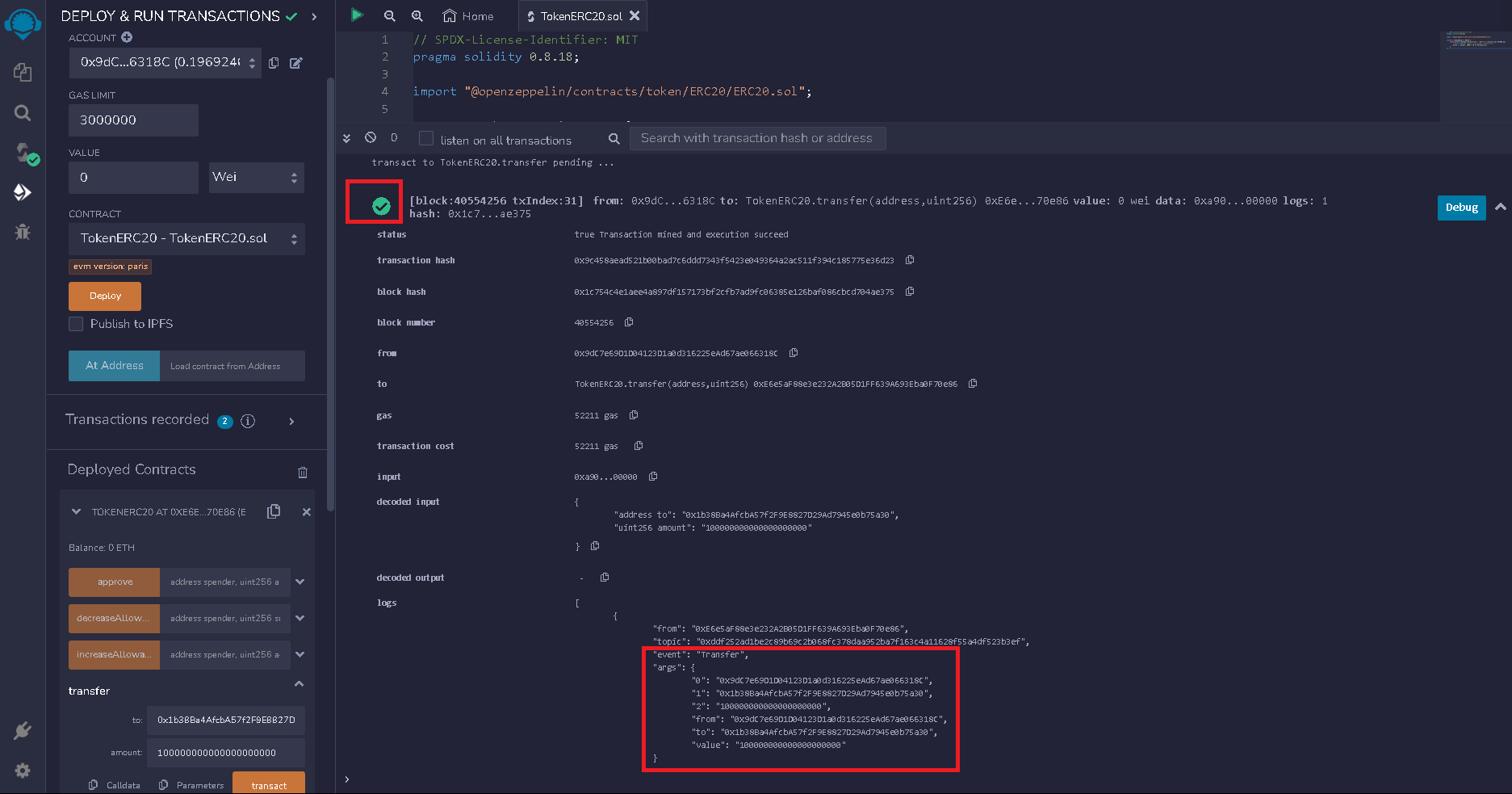
1. decimals(): Para obtener la cantidad de decimales del token, haga clic en el botón azul que dice decimals. El resultado será 18, que representa la unidad mínima divisible del token.
2. totalSupply(): Consulta el método totalSupply. Por ahora el valor es de 1000000000000000000000. Este número representa los 1000 tokens (con 18 decimales) que fueron acuñados al Propietario.



Consultando el balance de destinatario:







Después de la transferencia consultamos balances:

Propietario:

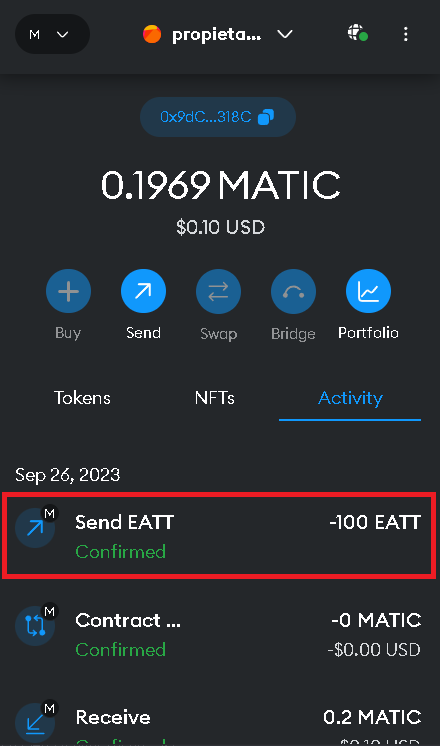
Texto

Descripción generada automáticamente

Destinatario:

Texto

Descripción generada automáticamente



Pantalla de celular con imagen de la pantalla de un celular

Descripción generada automáticamente

Importamos los tokens EATT en la cuenta Destinatario:

Pantalla de celular con imagen de la pantalla de un celular

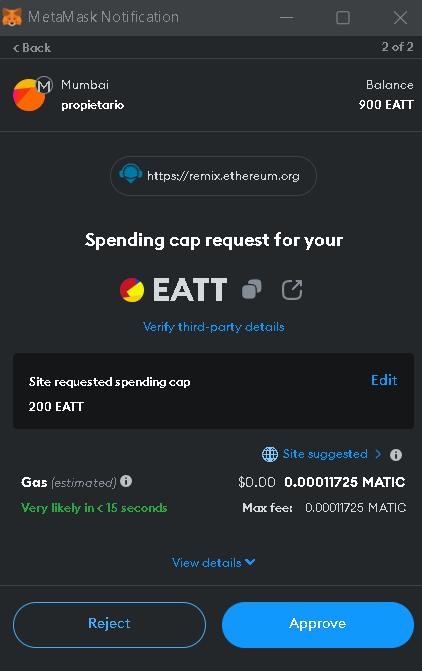
Descripción generada automáticamente con confianza media

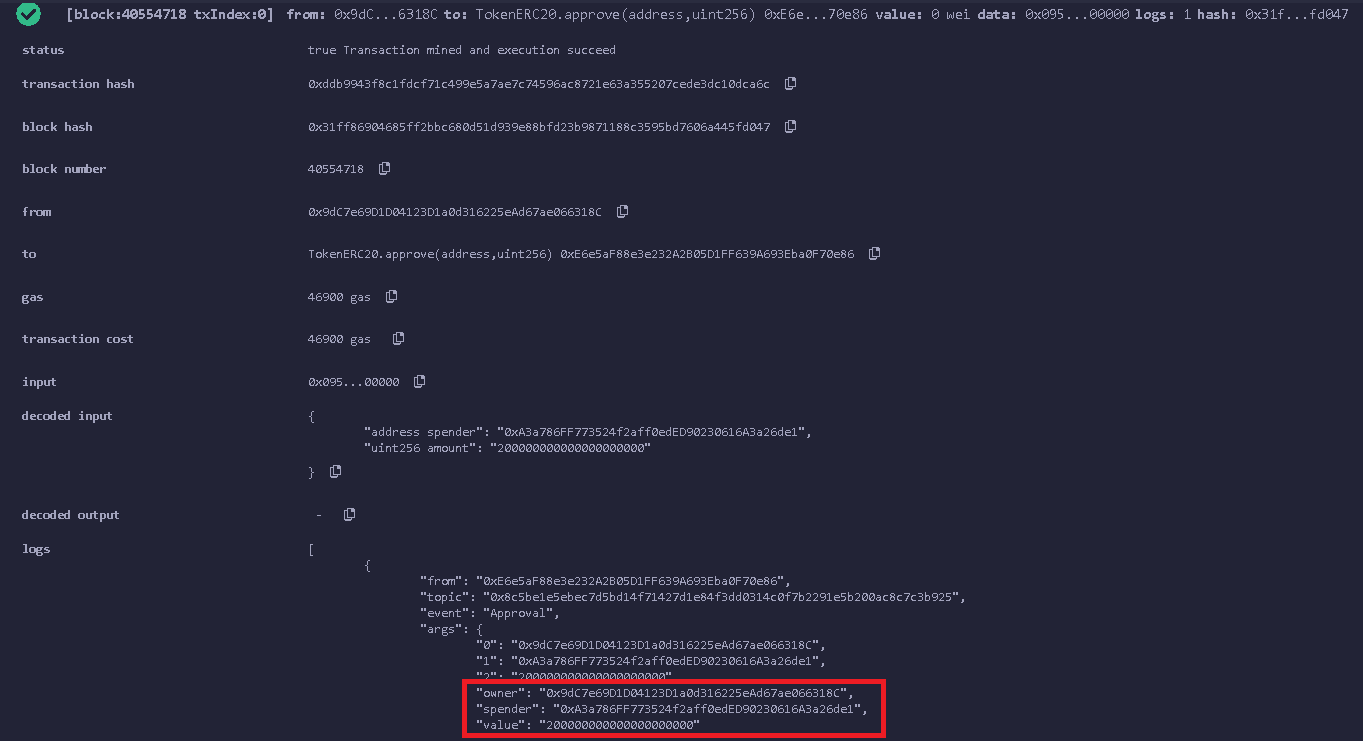
Vamos a darle permiso al Gastador sobre una cantidad de tokens del propietario:



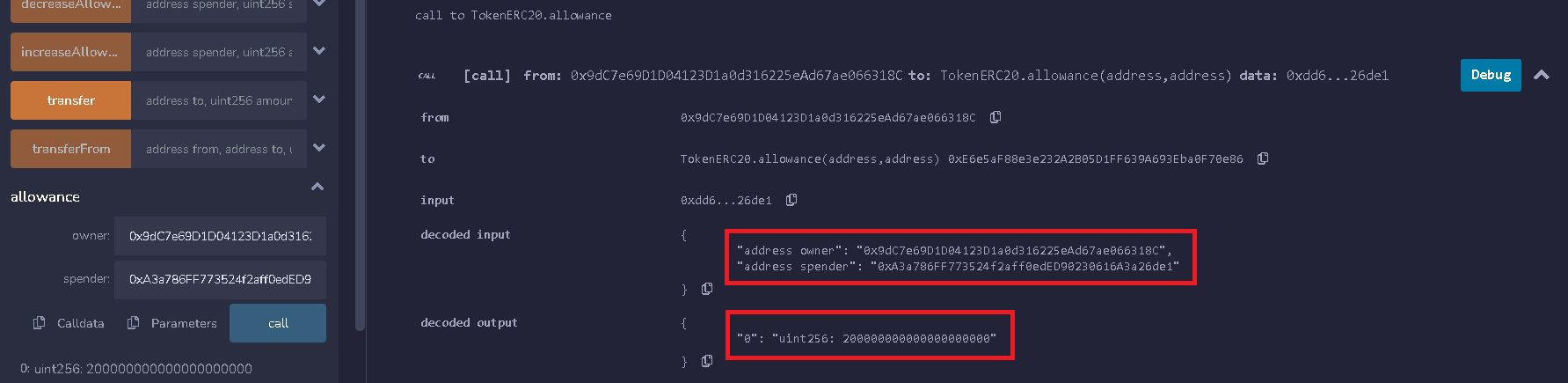
Interfaz de usuario gráfica, Aplicación

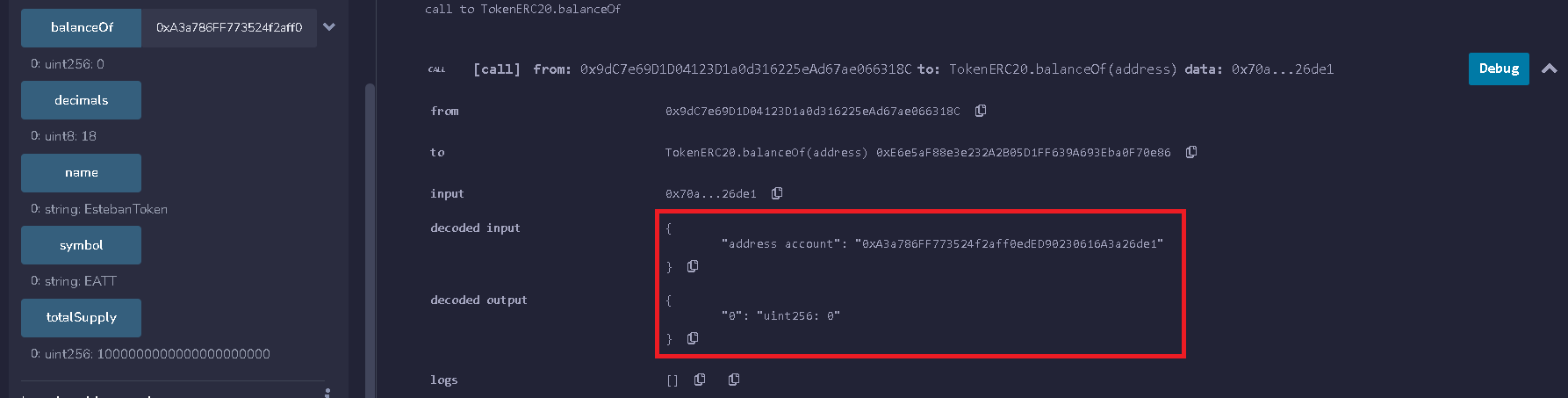
Descripción generada automáticamente





Solo se le ha dado al gastador realizar operaciones con esa cantidad de tokens a nombre del propietario, pero los saldos de ambos están iguales. No ha habido una transferencia.

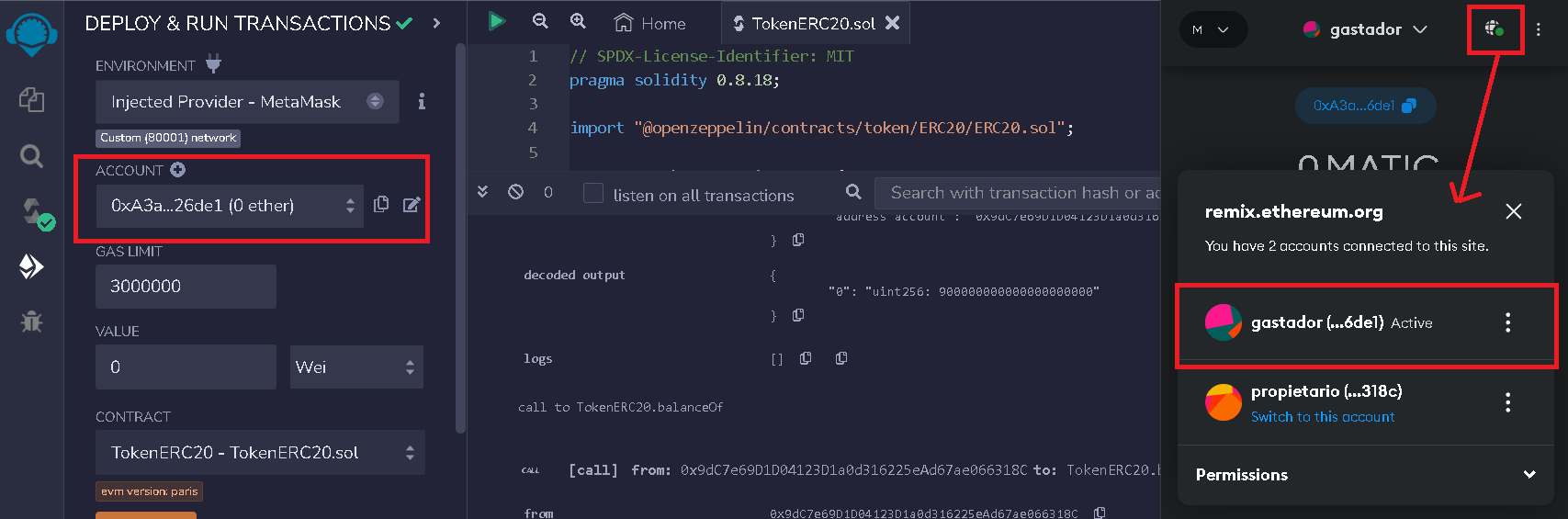




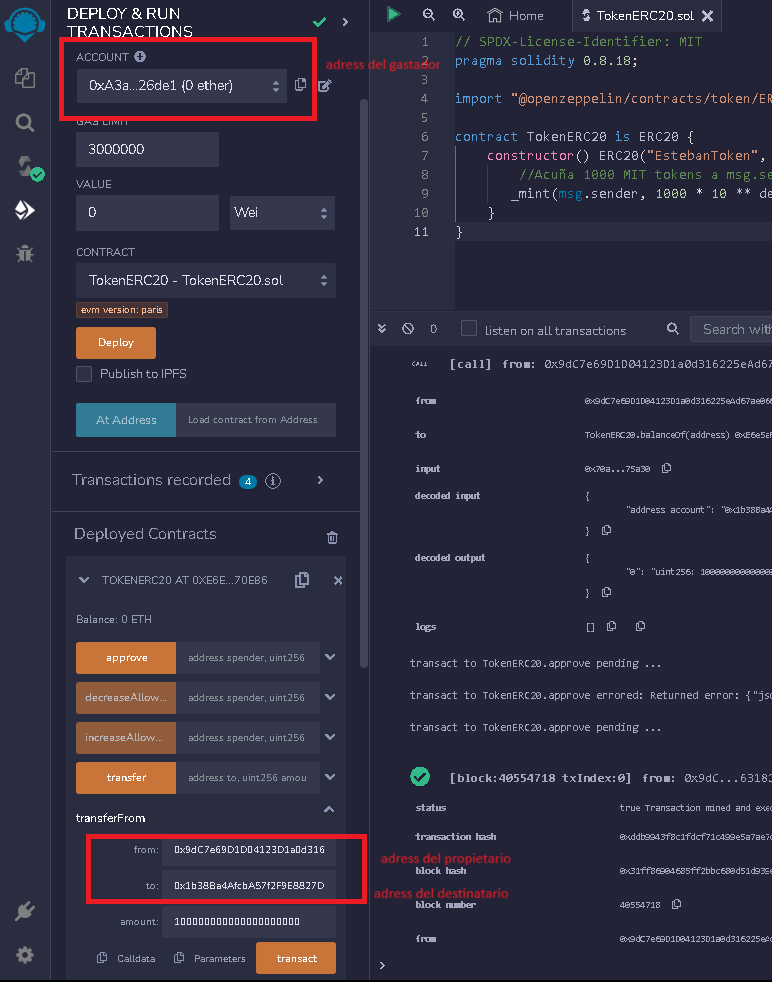
Ahora haremos uso de “transferFrom” que es transferir a nombre del propietario tokens a un destinatario. Para ello desde metamask, seleccionamos el account de Gastador:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



Le damos clic, para que ahora en REMIX sea la billetera de “gastador” quien procese la operación. Sin embargo de seguro le pedirá gas, así que deberíamos darle MATIC mediante el Faucett. Pero haremos el intento para ver que pasa, debería ser rechaza la transacción.

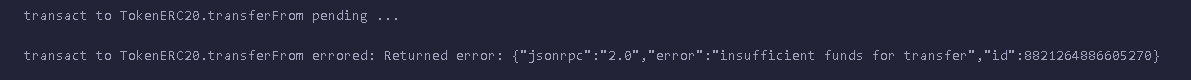


Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Efectivamente sale una advertencia.

Cancelo



Lleno de fondo con el Faucett al gastador. Pero el Faucett no me permte después de 24 Hrs.

Entonces desde metamask, hago una transferencia de propietario a gastador:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Le envío 0.001 MATIC

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Confirmo.

Pantalla de celular con imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Logotipo

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Ahora si Gastador tiene saldito:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Procedo a hacer la transferFrom en Remix:

Captura de pantalla de un celular

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Una captura de pantalla de un celular

Descripción generada automáticamente

Efectivamente se visualiza la transacción de gastador, el cual manda 100 EATT a la cuenta destinatario.

Vemos el log de remix

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En metamask cambiamos de cuenta del propietario:

Pantalla de celular con imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

El no hizo la transacción, no aparece aquí sino en gastador como lo vimos anteriormente

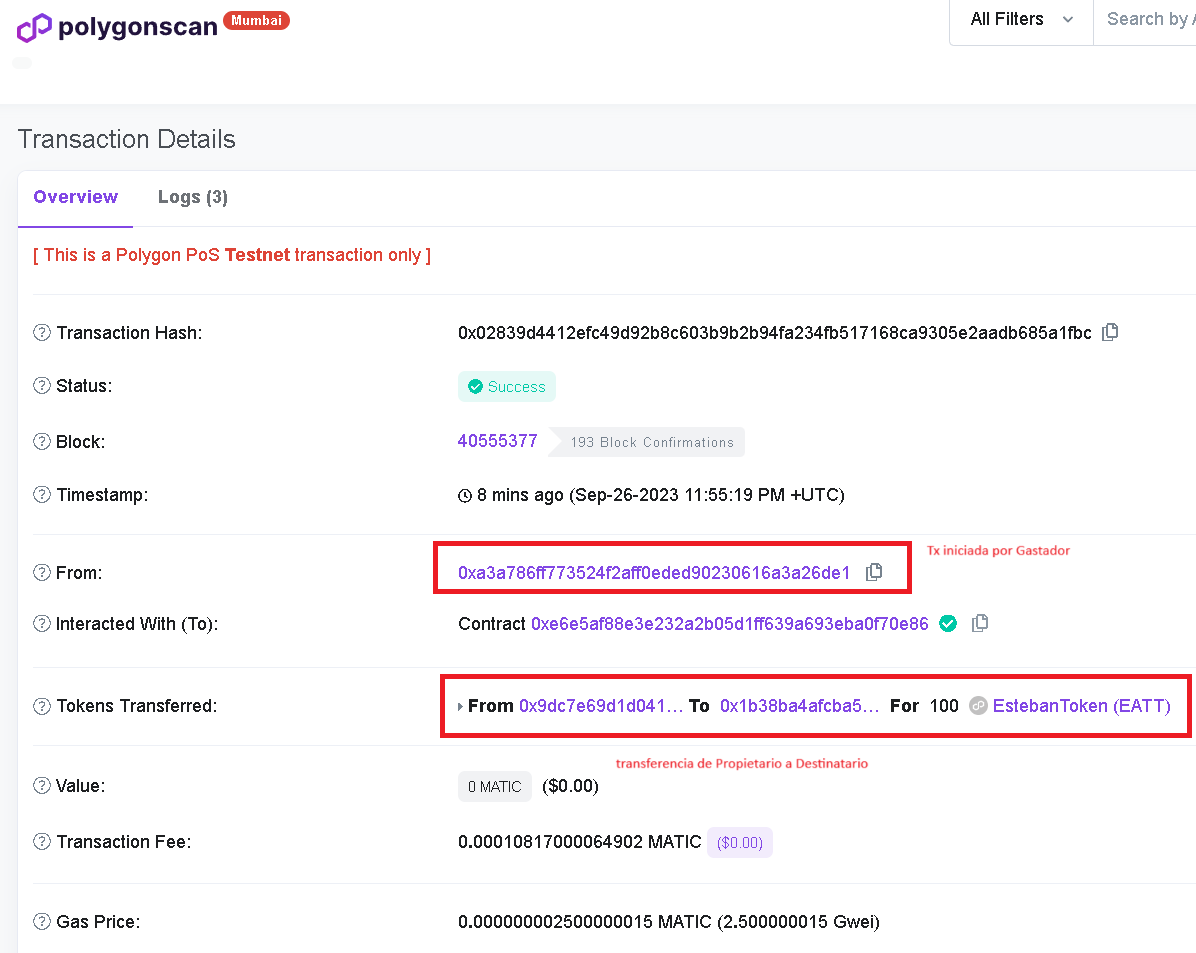
Vemos su saldo:

Pantalla de celular con imagen de la pantalla de un celular

Descripción generada automáticamente

Disminuyó eso si su saldo

Veamos en el block explorer, para ello regresamos a la cuenta de Gastador -> Activity, seleccionamos la transacción de envio de EATT de -100 EATT y damos clic en block explorer:



Verificar que Gastador ya no tiene 200 sino 100 de permiso:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ahora llamo a increaseAllowance para darle mas permisos al Gastador

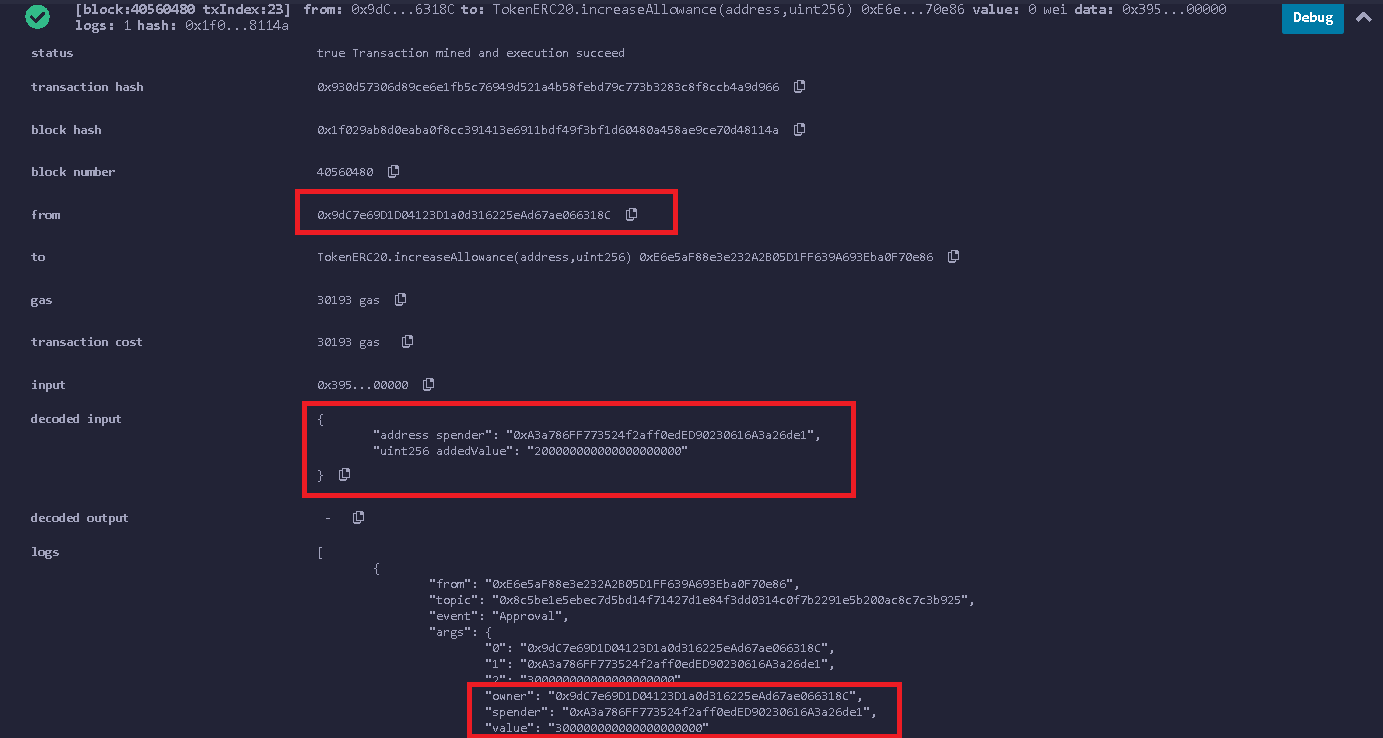
Para ello primero al caller o tx.origin de la operación lo cambiamos al Propietario (en Remix estaba con el de Gastador)

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



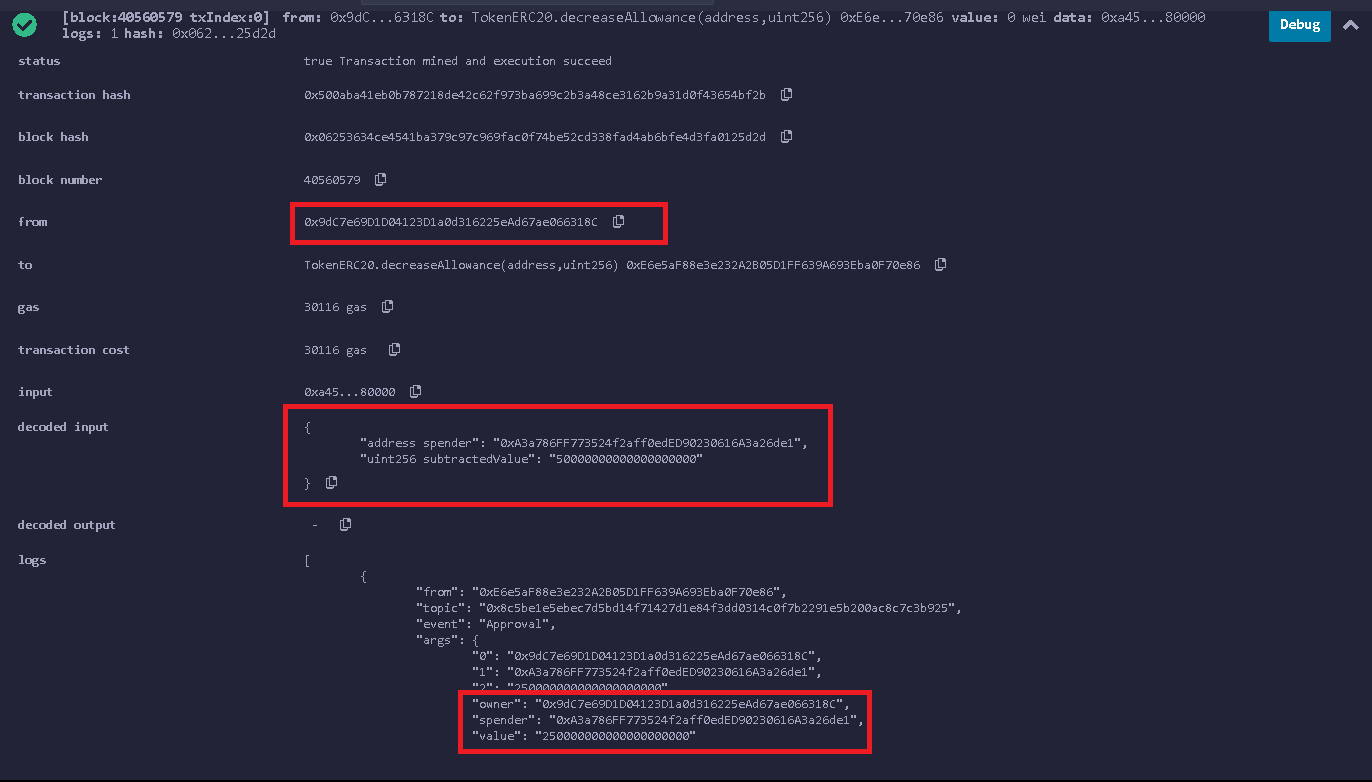
Ahora disminuyo el allowance:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente



**En conclusión:**

1. **approve**, **increaseAllowance** y **decreaseAllowance**

**decreaseAllowance**: podría usarse el mismo **increaseAllowance** pasándole el monto en negativo.

El **approve** puede ser sustituido por **increaseAllowance**

En resumen:

Existen 3 métodos: **approve**, **increaseAllowance**, **decreaseAllowance**

    function approve(address spender, uint256 amount) public virtual override returns (bool) {

        address owner = \_msgSender();

        \_approve(owner, spender, amount);

        return true;

    }

    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {

        address owner = \_msgSender();

        \_approve(owner, spender, allowance(owner, spender) + addedValue);

        return true;

    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {

        address owner = \_msgSender();

        uint256 currentAllowance = allowance(owner, spender);

        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");

        unchecked {

            \_approve(owner, spender, currentAllowance - subtractedValue);

        }

        return true;

    }

Yo me quedaría solo con uno: **setAllowance** (tanto inicialmente como a la hora de incrementar o disminuir su permiso de Q tokens, con la validación de que no sea un permiso de Q tokens negativo).

Todos los dueños o propietarios de tokens pueden llamar a este método (no solo el owner del contrato), solo para precisar.

\* El termino owner aquí se refiere a los propietarios de los tokens. Pareciera que se refiriese al propietario del contrato quien solo seria el que deployó el contrato o dueño del contrato.

1. **allowance**

    function allowance(address owner, address spender) public view virtual override returns (uint256) {

        return \_allowances[owner][spender];

    }

El método allowance recibe el parámetro owner (dueño de los tokens) y spender (el gastador a quien se le dará permisos de usarlos a nombre del dueño).

¿Porque 2 parámetros y no 1?

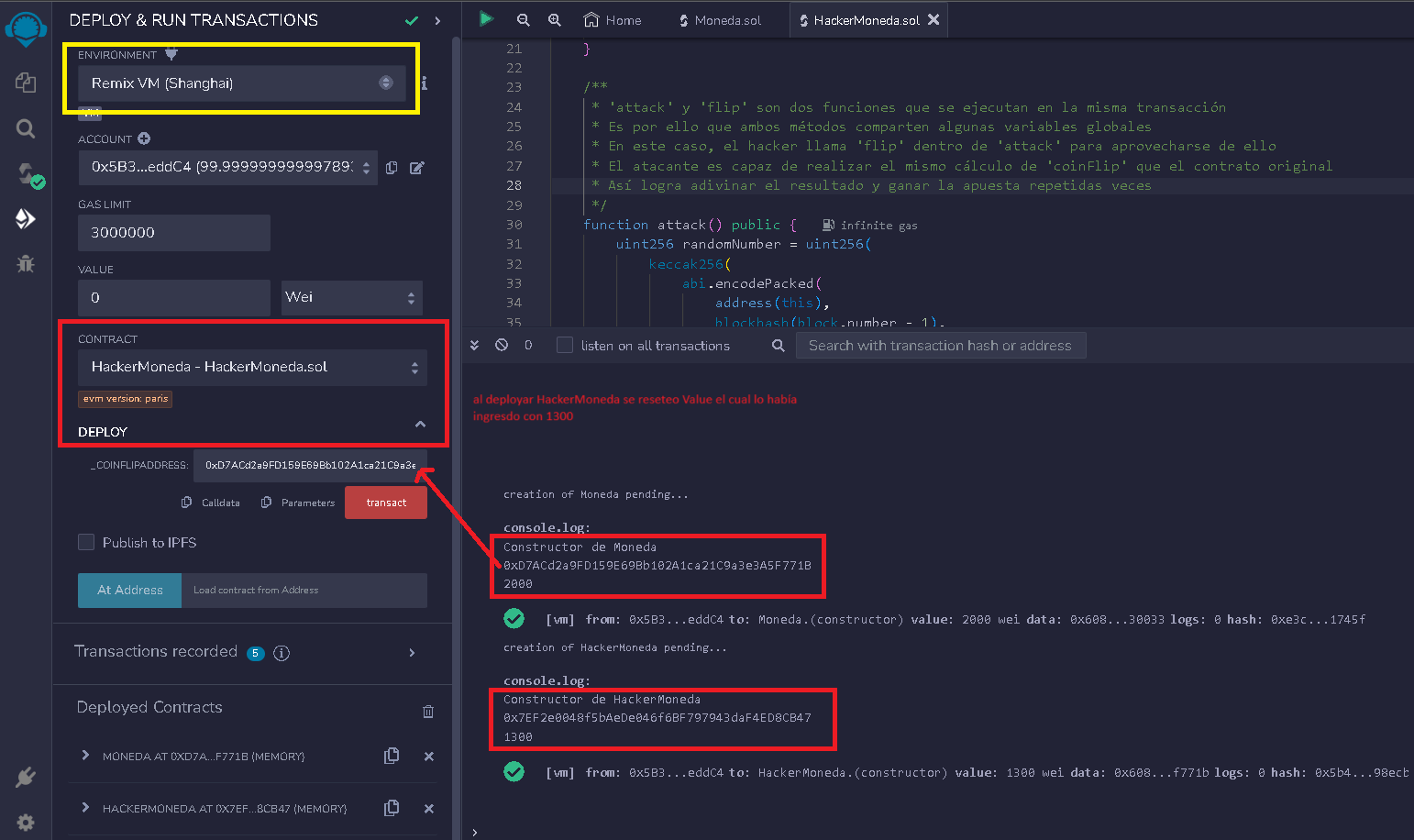
- Si lo llama el owner o dueño de los tokens, pareciera que estuviese de más ese primer parámetro.

- Si lo llama el spender o gastador, pareciera que estuviese de más el 2do parámetro.

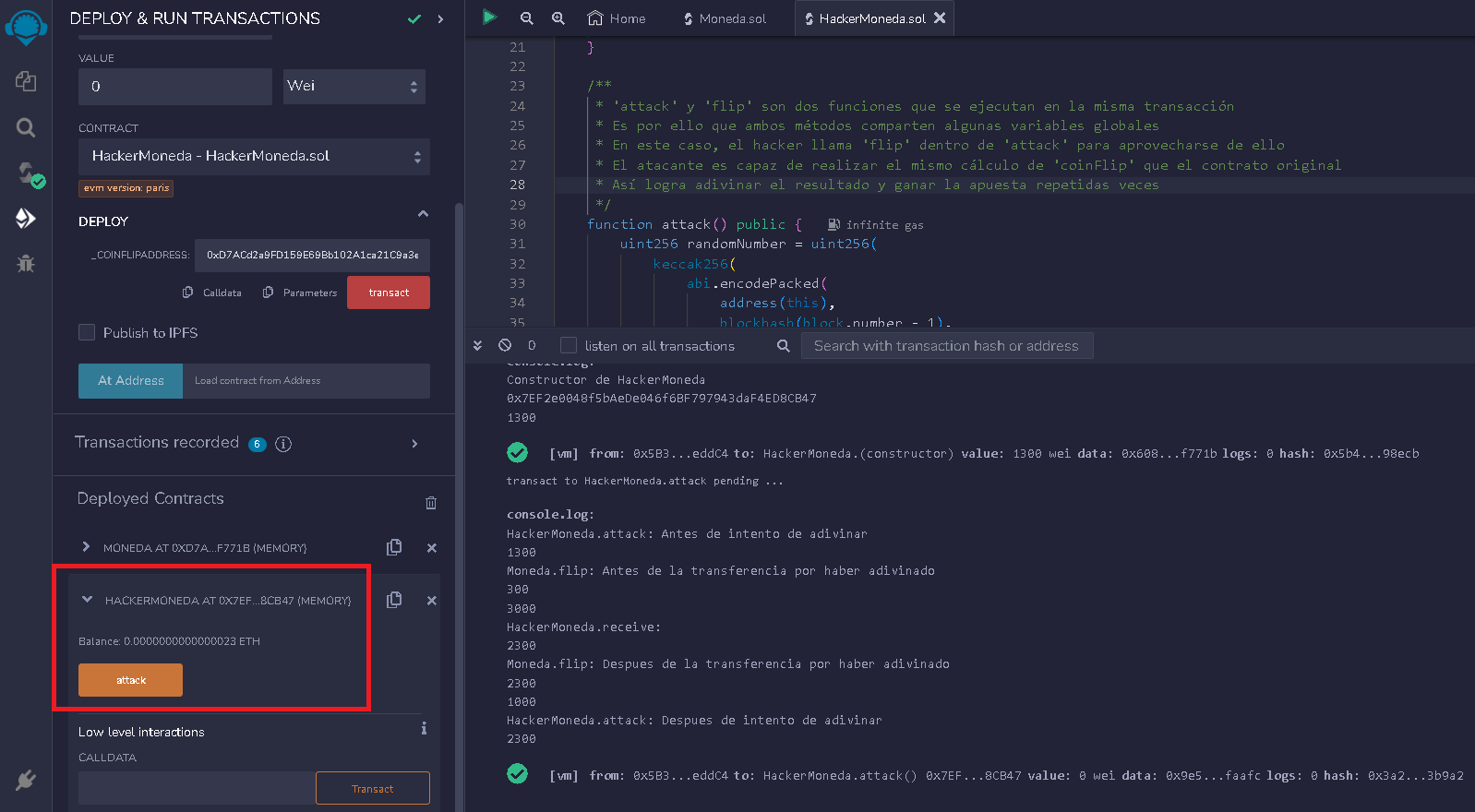
Para no crear 2 métodos por separado, uno para que lo llame el dueño y otro el spender, es que el método allowance recibe 2 parámetros.

Adicionalmente, aunque algo innecesario, sería que cualquier persona podría llamar a este método para averiguar el permiso que tenga cualquier account sobre tokens de un tercero.

**LABORATORIO: DEVELANDO NÚMEROS RANDOM EN SOLIDITY**



Al atackar:



Los addresses:

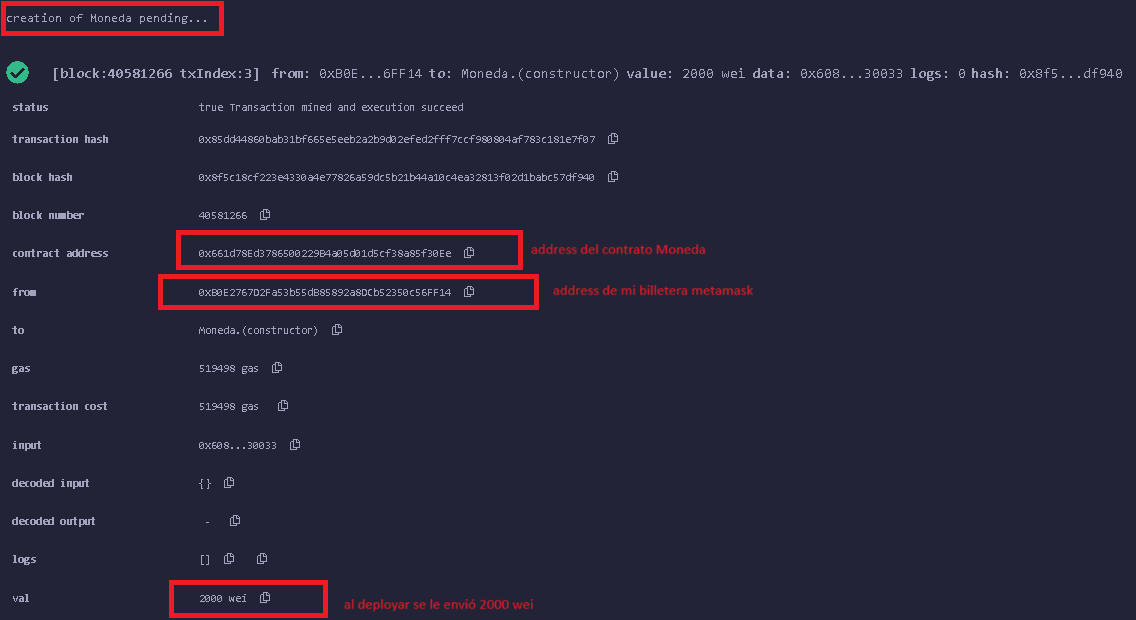
**Moneda**

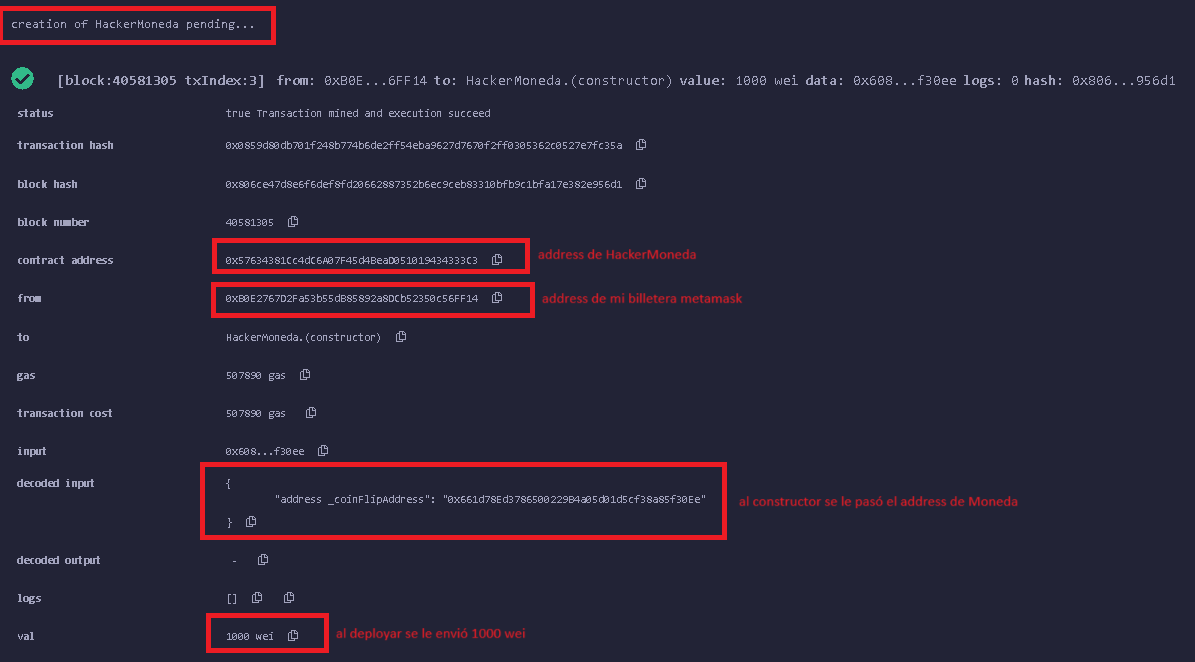
**0xD7ACd2a9FD159E69Bb102A1ca21C9a3e3A5F771B**

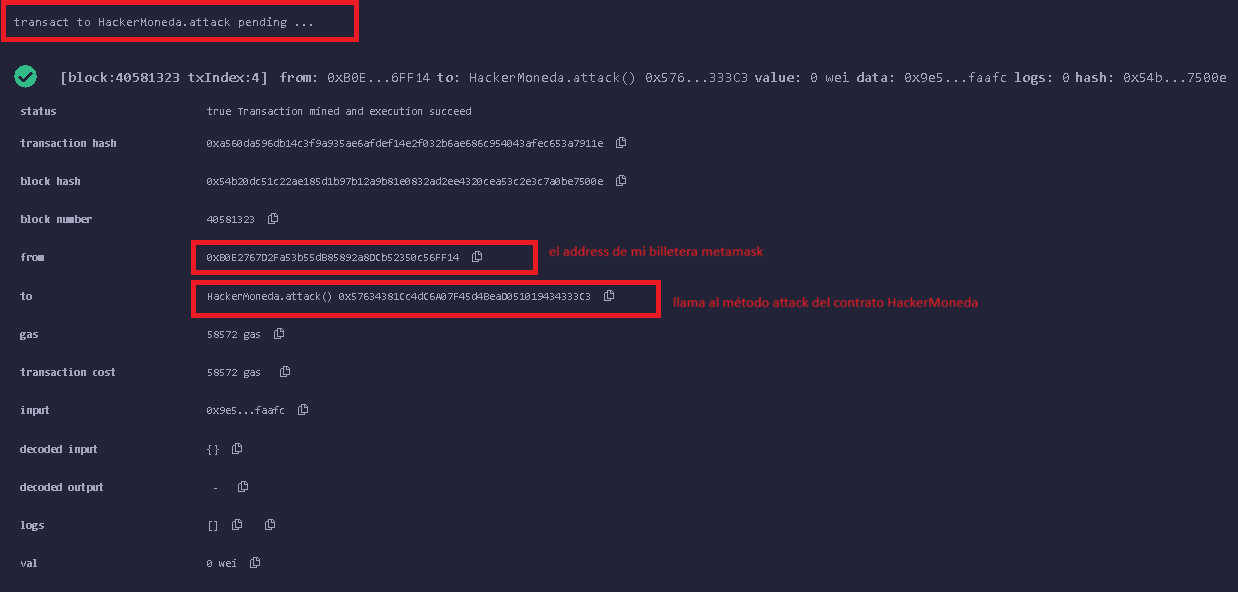
**HackerMoneda**

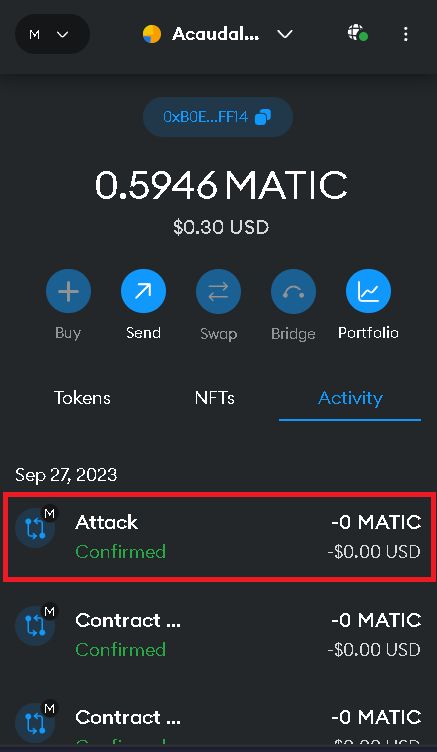
**0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47**

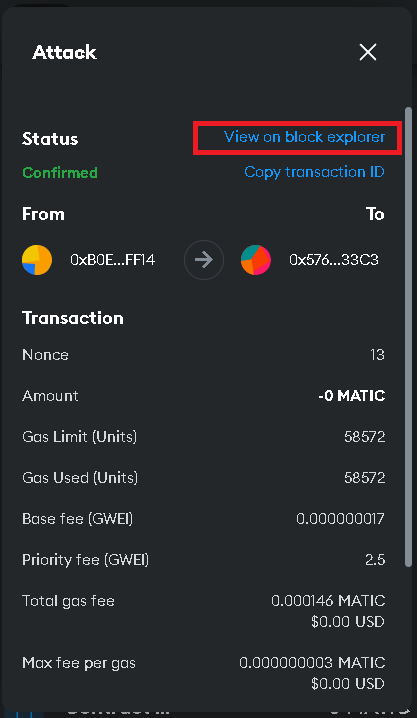
Lo volví a hacer usando mi billetera de Metamask en lugar de usar el de Remix VM (Shangai)

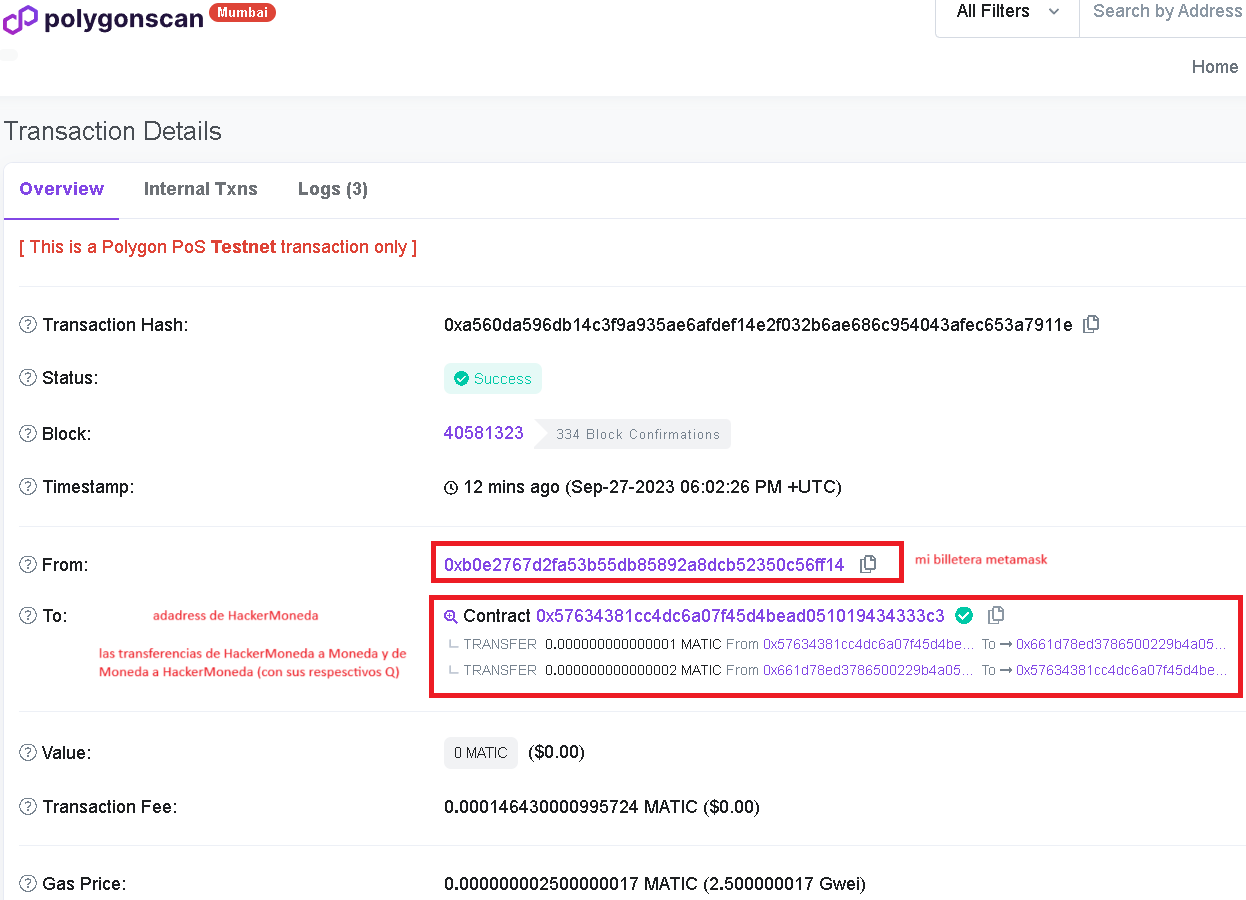












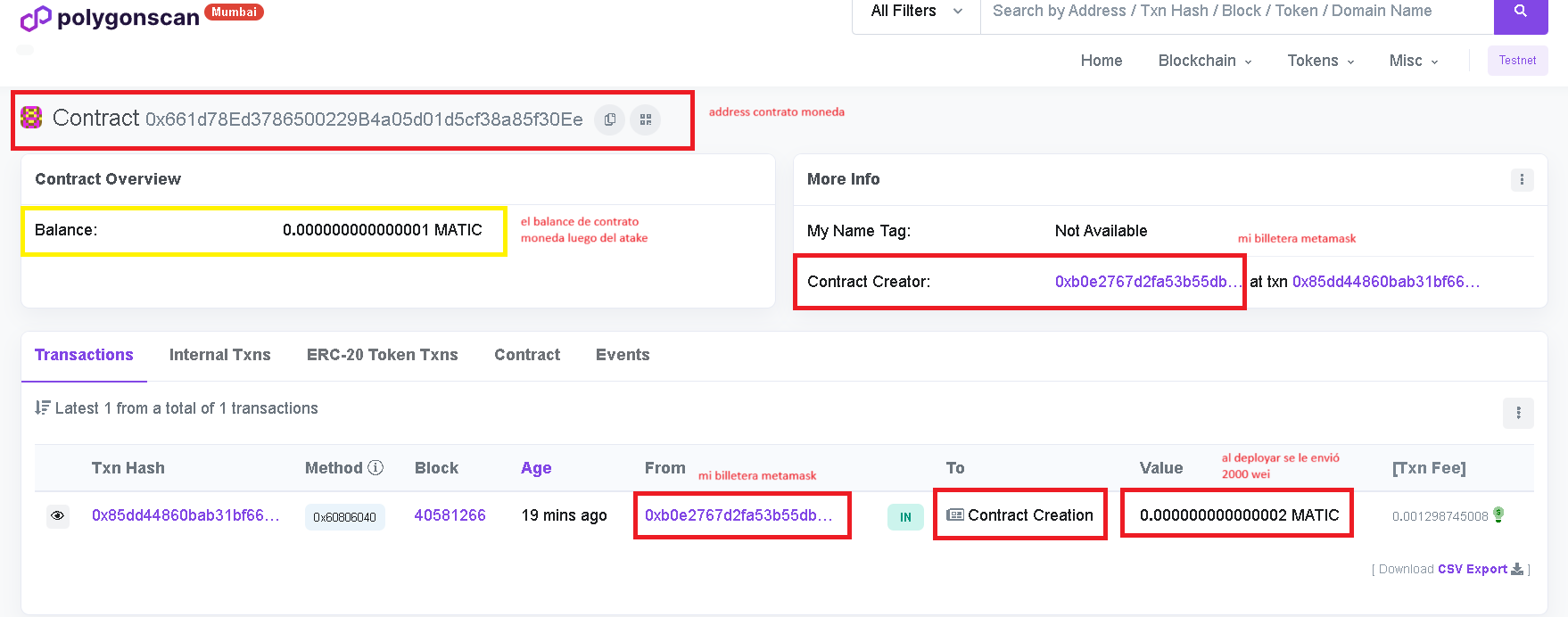
Los addresses:

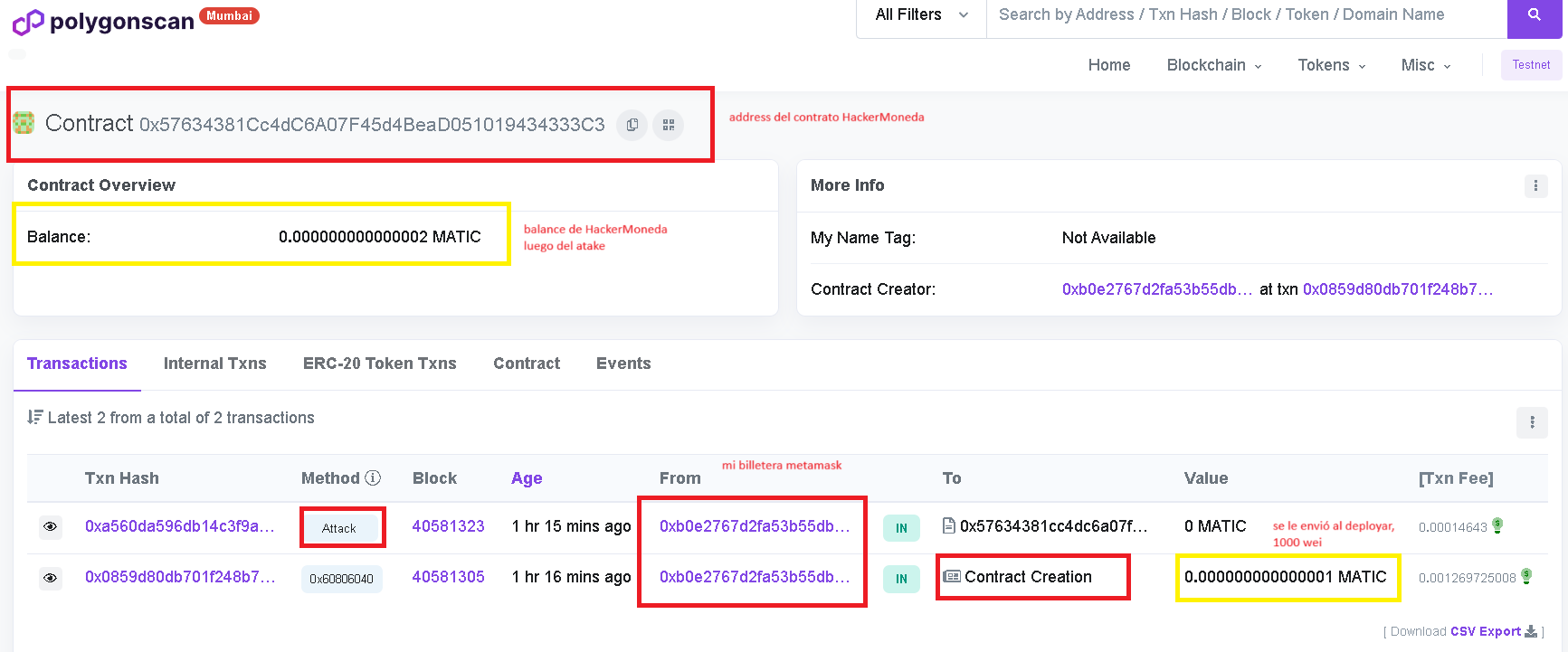
**Moneda**

0x661d78Ed3786500229B4a05d01d5cf38a85f30Ee

**HackerMoneda**

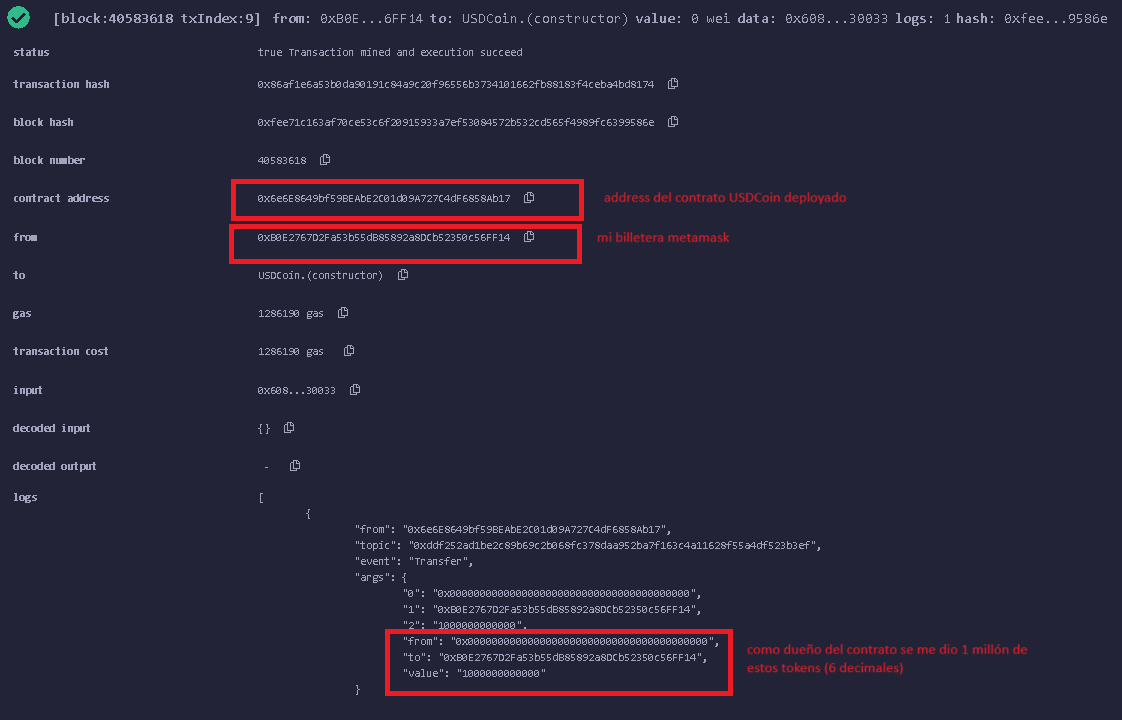
0x57634381Cc4dC6A07F45d4BeaD051019434333C3





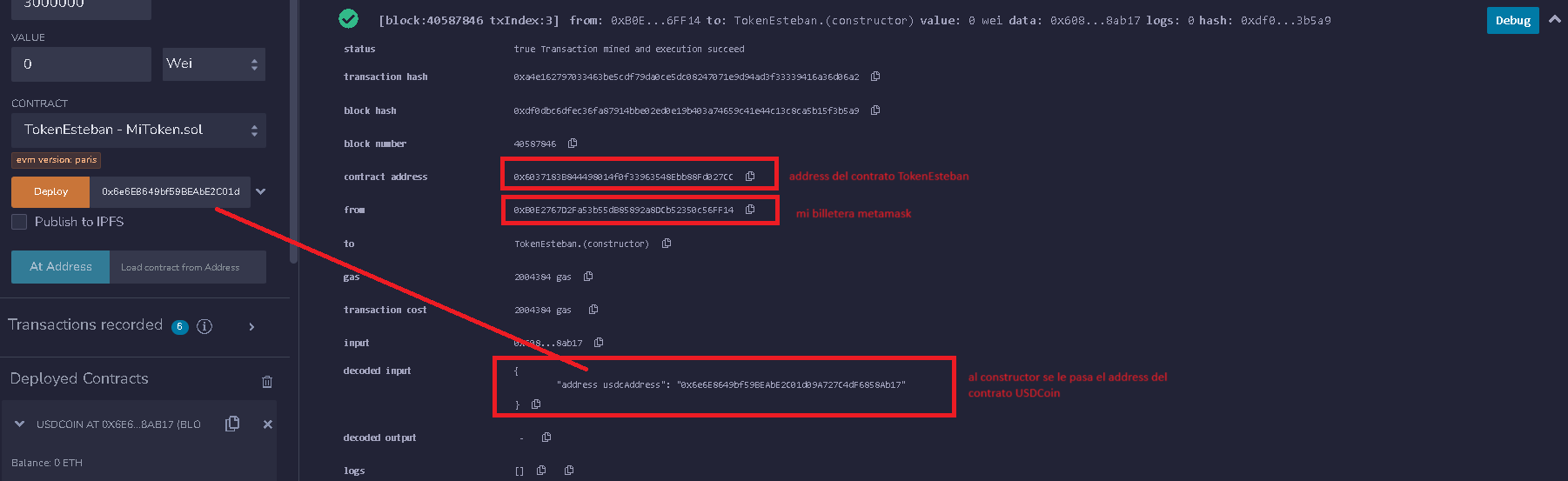
**LABORATORIO: Contrato de compra y venta de Tokens (ERC20) usando monedas estables (e.g. USDC)**

Deployamos el contrato **USDCoin**



A mi cuenta metamask, como dueño del contrato USDCoin, se me dio 1 millon de tokens USDCoin.

Address del contrato USDCoin: 0x6e6E8649bf59BEAbE2C01d09A727C4dF6858Ab17

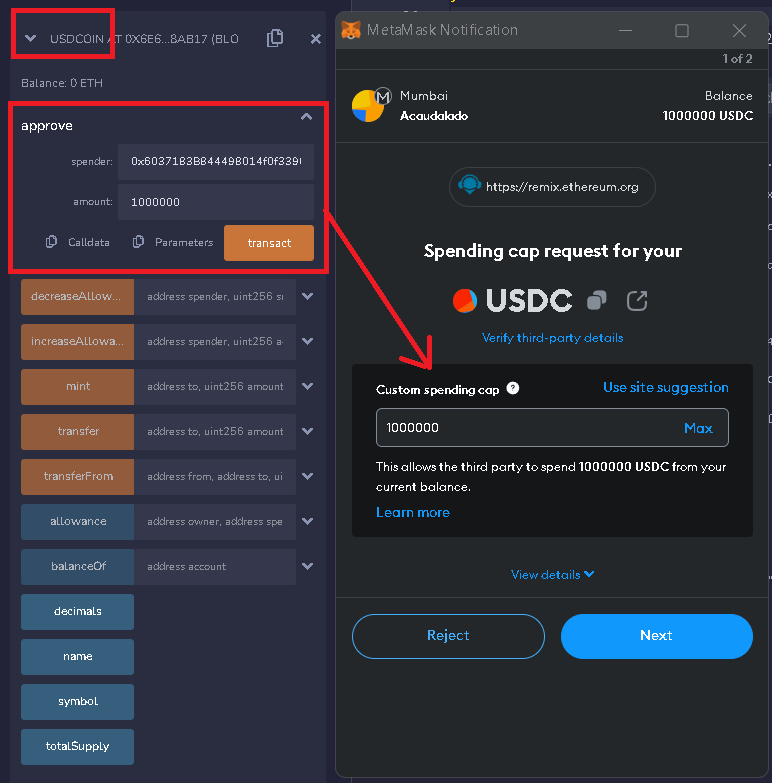


Address del contrato TokenEsteban: 0x6037183B844498014f0f33963548Ebb88Fd027CC

Address de metamask (comprador): 0xB0E2767D2Fa53b55dB85892a8DCb52350c56FF14

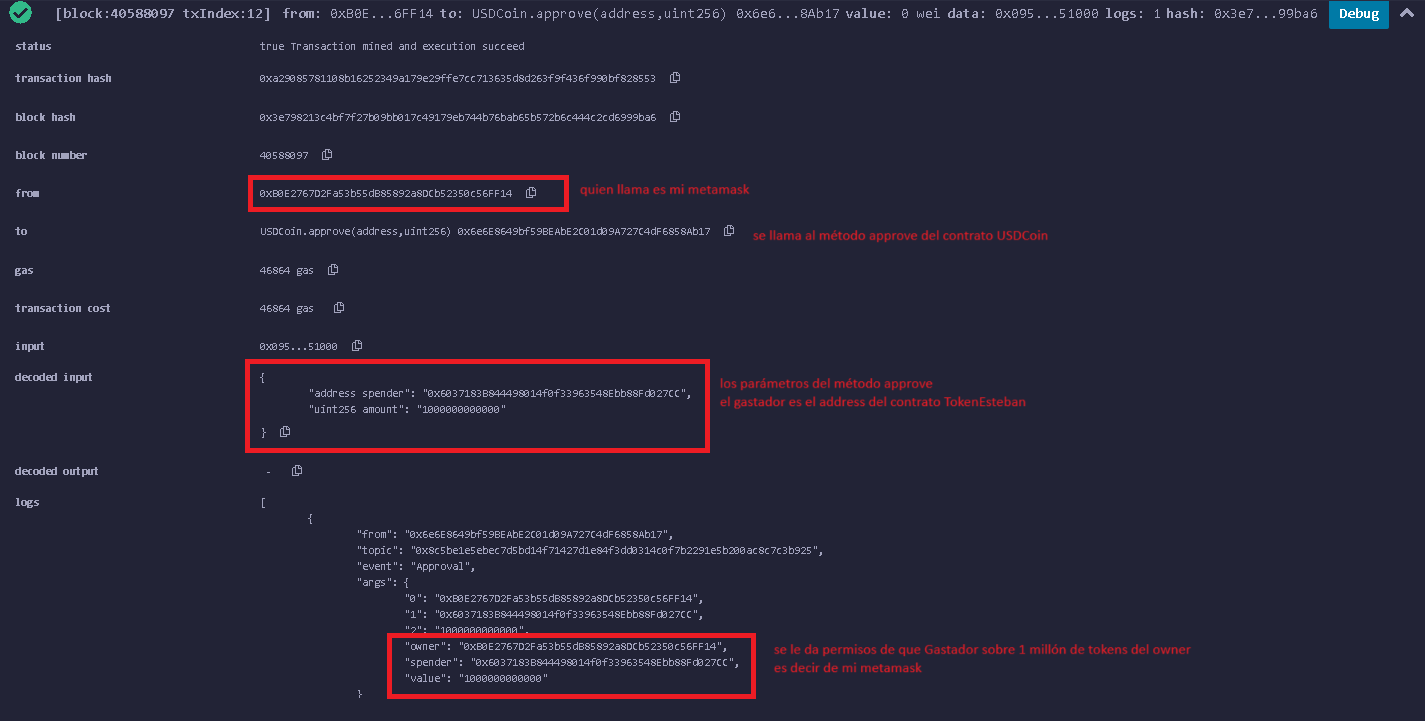
Para que mi cuenta metamask pueda comprar tokens TokenEsteban, primero debe darle permisos de que el contrato TokenEsteban pueda manejar sus tokens de USDCoin.

Para ello en el contrato USDCoin, mi metamask llama al approve, pasandole como gastador el address del contrato TokenEsteban y un monto, en este caso, 1 millon de tokens USDCoin.



Captura de pantalla de un celular

Descripción generada automáticamente



Ahora el comprador llama al contrato de TokenEsteban para comprar su token con 100$

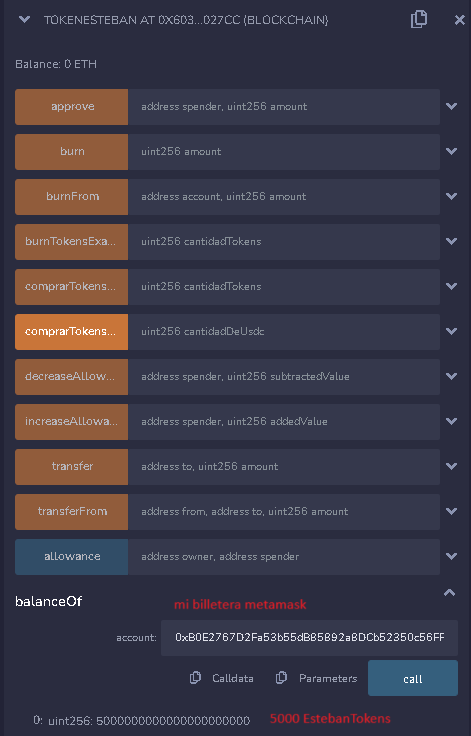
Interfaz de usuario gráfica

Descripción generada automáticamente

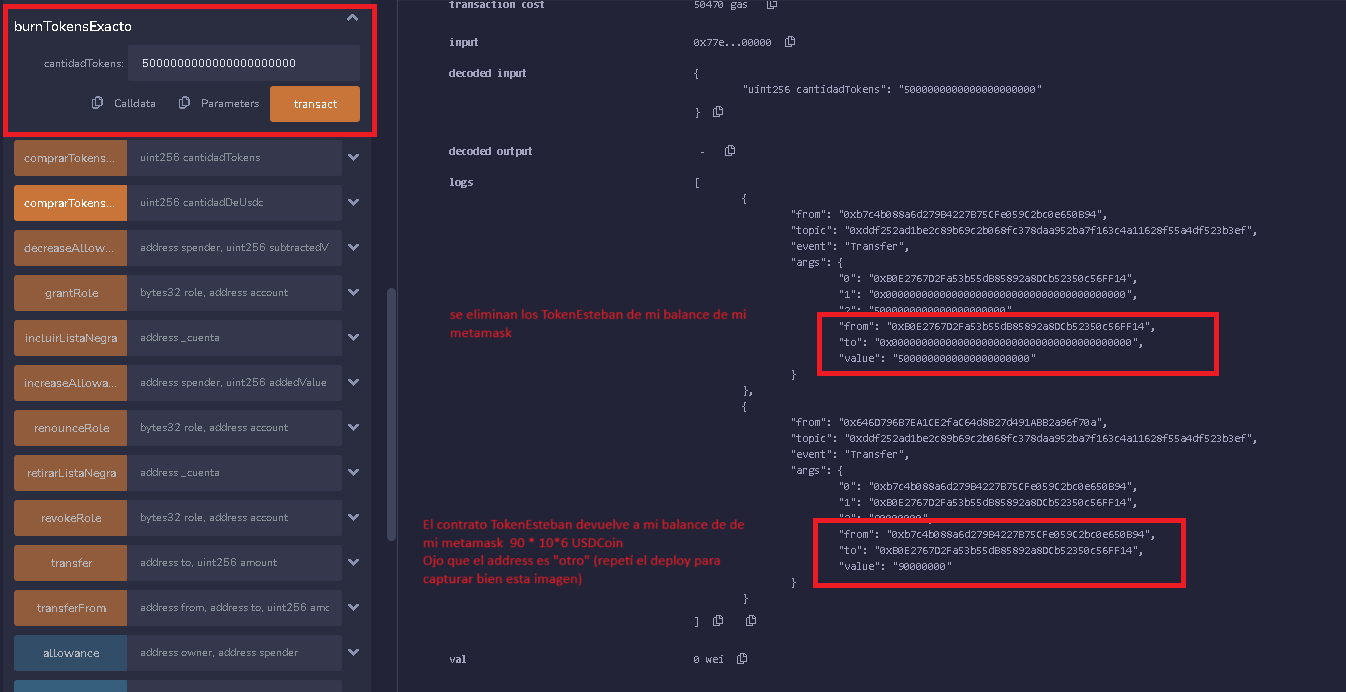
Texto

Descripción generada automáticamente





Ahora los quemo: (se corrigió el código para que se calcule bien el pago del impuesto)

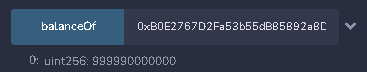


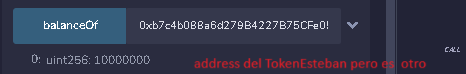
Consulto mi balance de TokenEsteban:

Interfaz de usuario gráfica, Aplicación

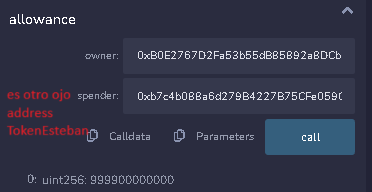
Descripción generada automáticamente

Consulto mi balance de USDCoin:





El allowance de Usccoin del TokenEsteban bajo en 100:



Ahora quiero comprar 4500 \* 10\*\*18 TokenEsteban:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



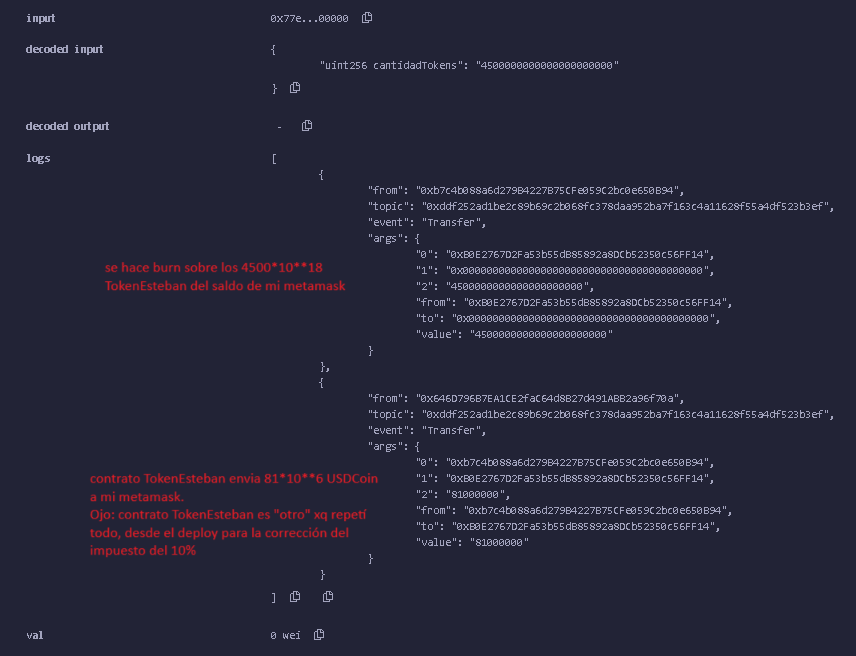
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ahora hacemos una “devolución” de esos 4500 \*10\*\*18 TokenEsteban

Texto

Descripción generada automáticamente



Balance de TokenEsteban:

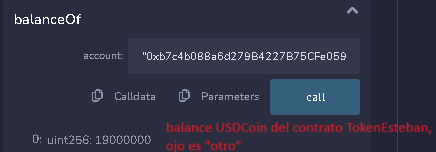
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Balance de USDCoin:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Paso 17 corregir:

Cabe notar que 9000000 es el 10% de 90000000 USDC que fue lo que pagó para obtener 4500000000000000000000 tokens MTLMRR. Ello demuestra que se aplicó el impuesto del 10%.

Debe decir:

Cabe notar que 9000000 es el 10% de 90000000 USDC que fue lo que pagó **por** **devolución de los** 4500000000000000000000 tokens MTLMRR. Ello demuestra que se aplicó el impuesto del 10%.

Hay un ERROR, en la devolución, no se devuelve el 90% (100% - 10% del impuesto) sino solo el 10%.

Se volvió a correr el ejercicio porque el profe pide que la comisión sea ahora de 30% además de poner condición al hacer la transferencia mediante una lista negra.

Address del contrato USDCoin: 0xFbeDbE14D31B9c3d67b5C6b1fbC99A1755691568

Address del contrato TokenEsteban: 0xDF8347383306F190734C908b020907924DaE8863

Address de metamask (comprador): 0xB0E2767D2Fa53b55dB85892a8DCb52350c56FF14

Le doy approve de USDCoin de mi metamask sobre el contrato TokenEsteban por 1000000000000

Agrego a la lista negra de TokenEsteban a mi metamask, e intento comprar TokenEsteban con mi metamask el valor de $100.

La pregunta seria como adivino que saldría el error sin antes enviar la transacción:

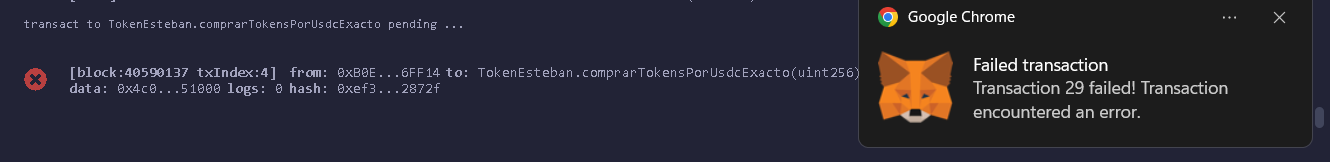
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Le doy Send Transaction:

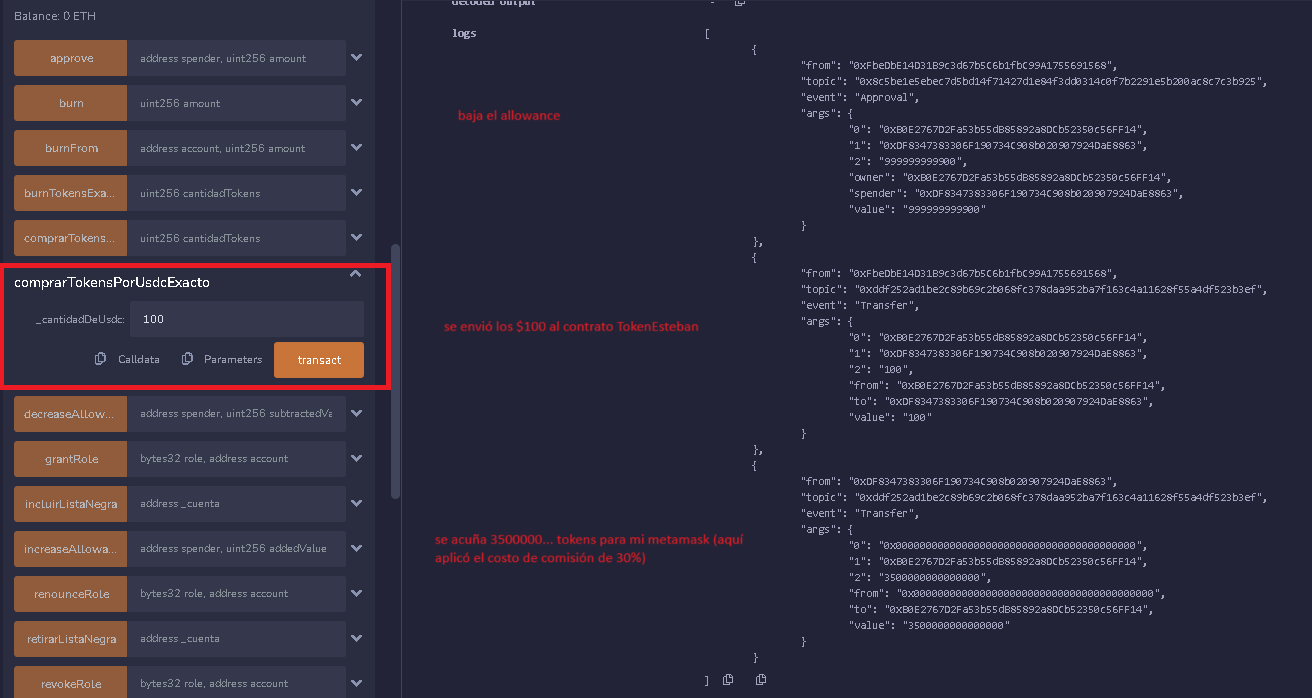
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



Lo retiro de la lista negra

Vuelvo a comprarTokenes con $100:



Vuelvo a comprarTokenes por 100 Tokens:

Interfaz de usuario gráfica, Texto, Aplicación

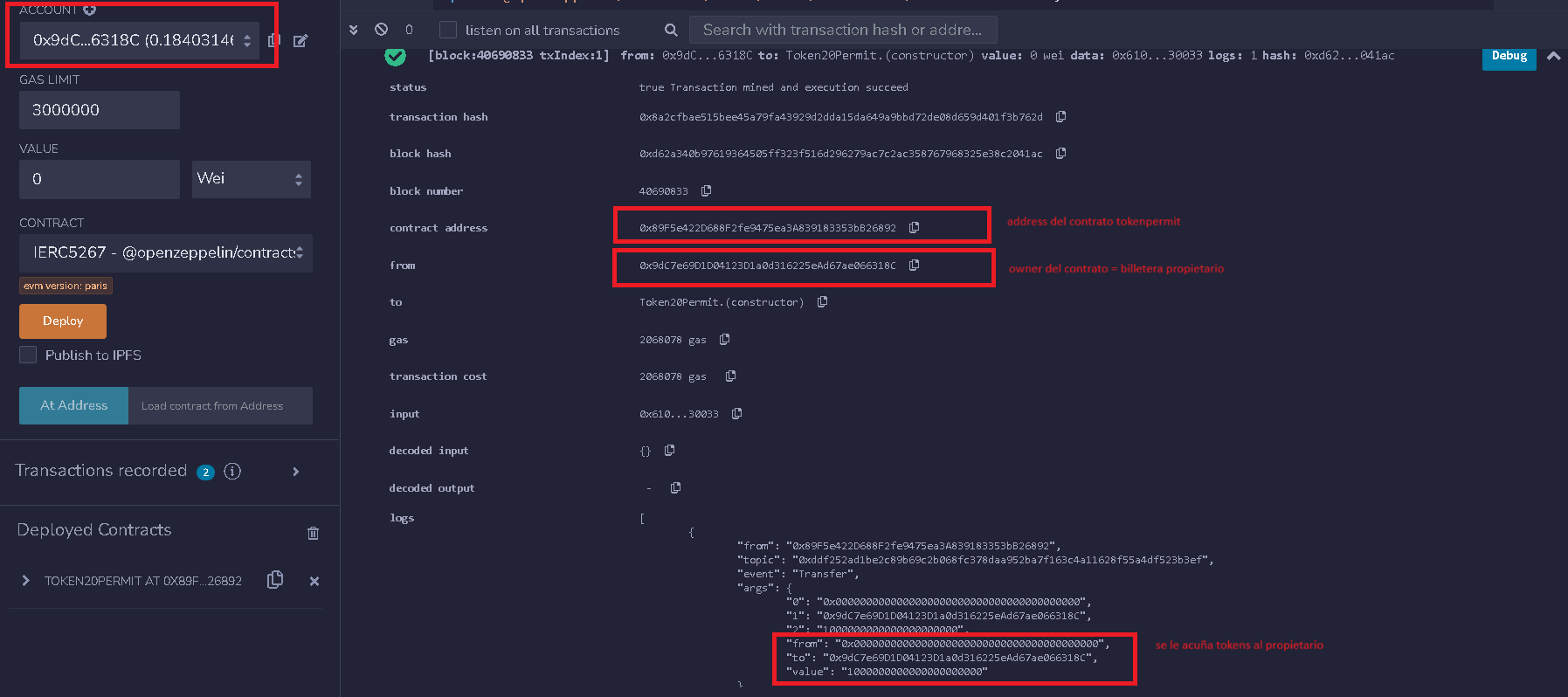
Descripción generada automáticamente

**LABORATORIO: ERC20Permit: Aprobaciones sin gas mediante firmas**

Address metamask Propietario: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

Address metamask Gastador: 0xA3a786FF773524f2aff0edED90230616A3a26de1

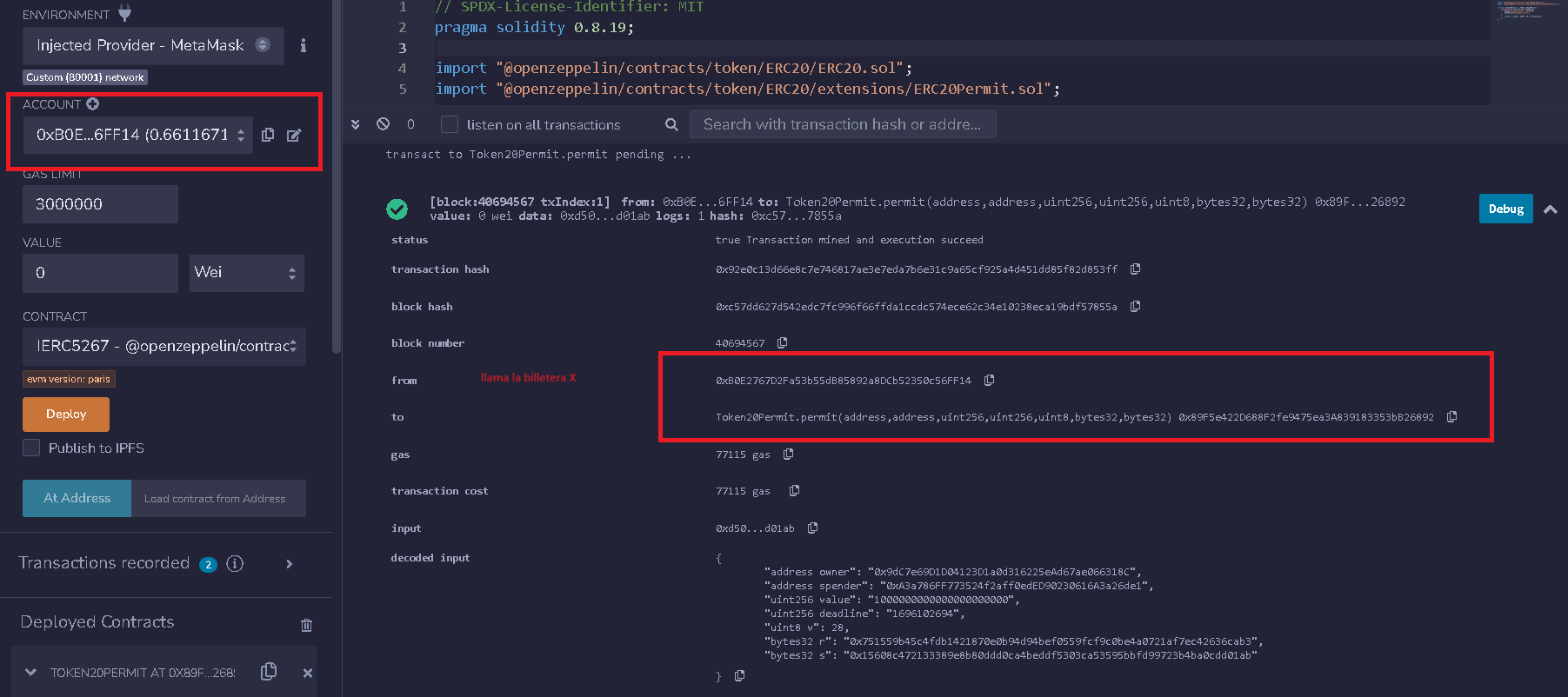
Address metamask X: 0xB0E2767D2Fa53b55dB85892a8DCb52350c56FF14

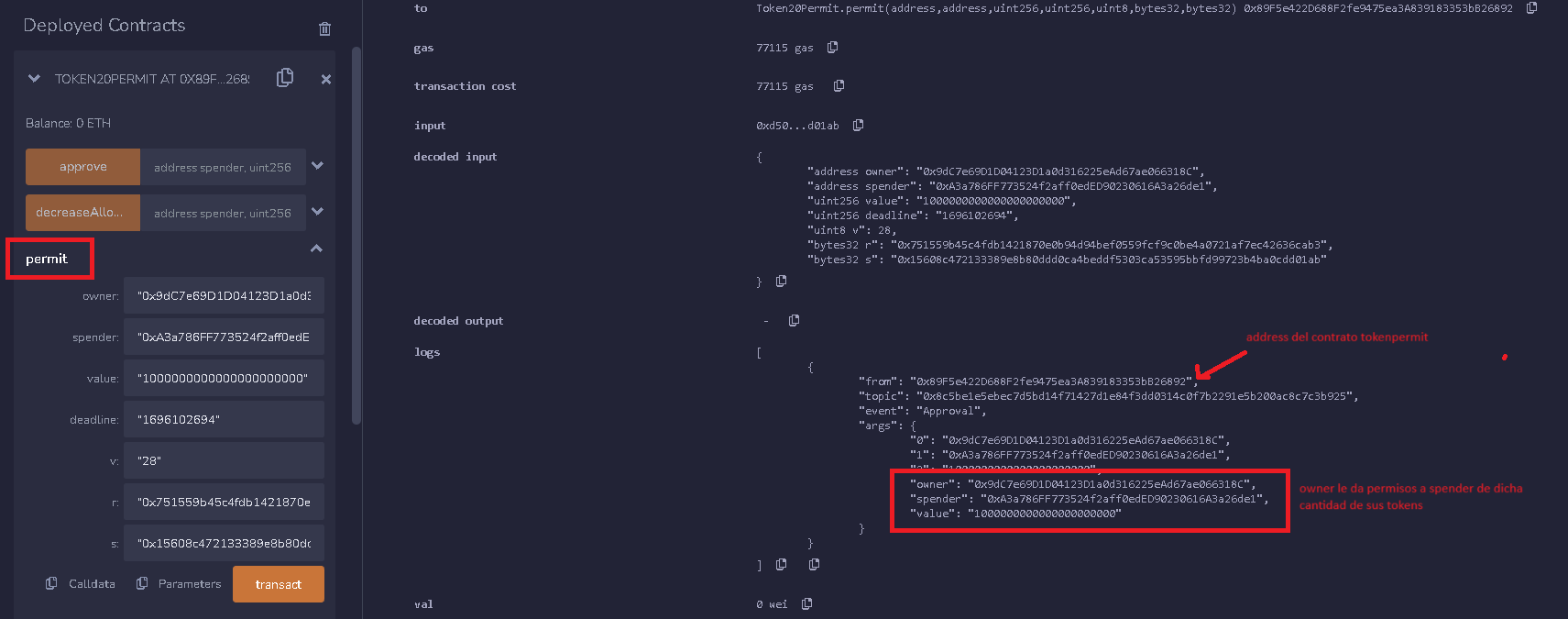


Address Contrato publicado Token20Permit: 0x89F5e422D688F2fe9475ea3A839183353bB26892

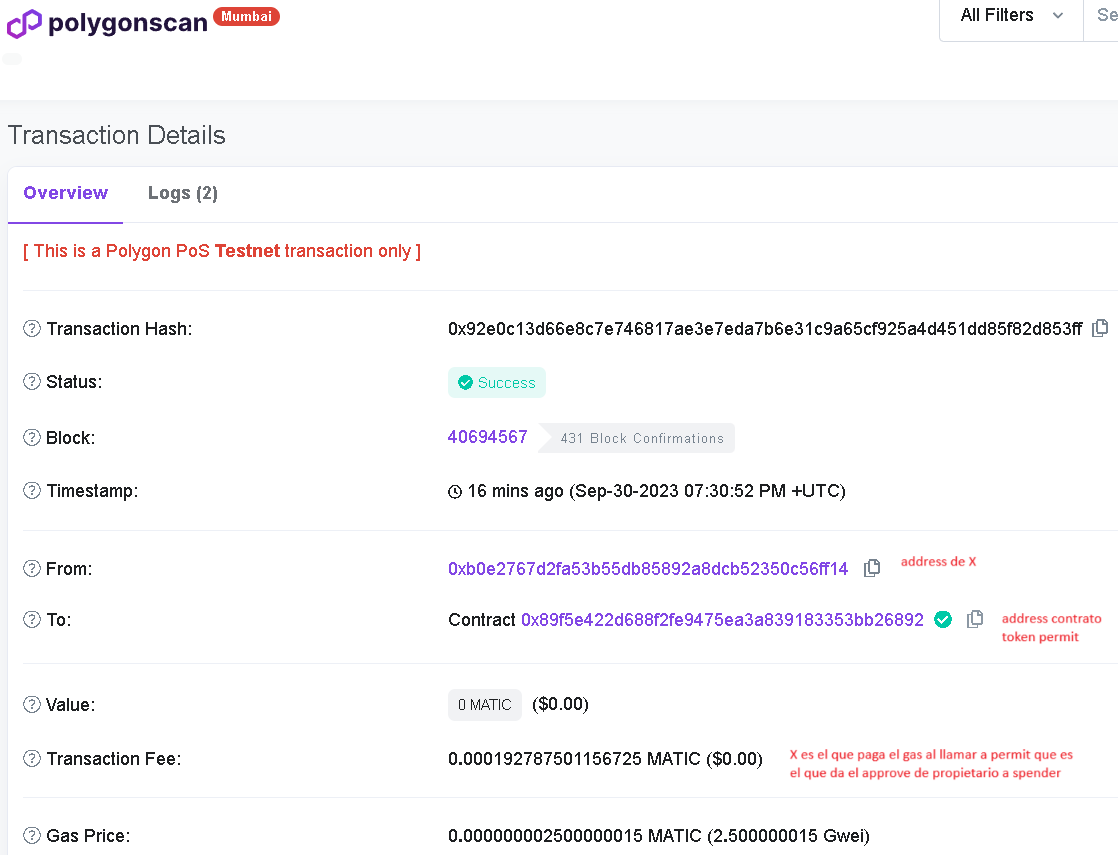
npx hardhat --network mumbai run scripts/erc20Permit.js

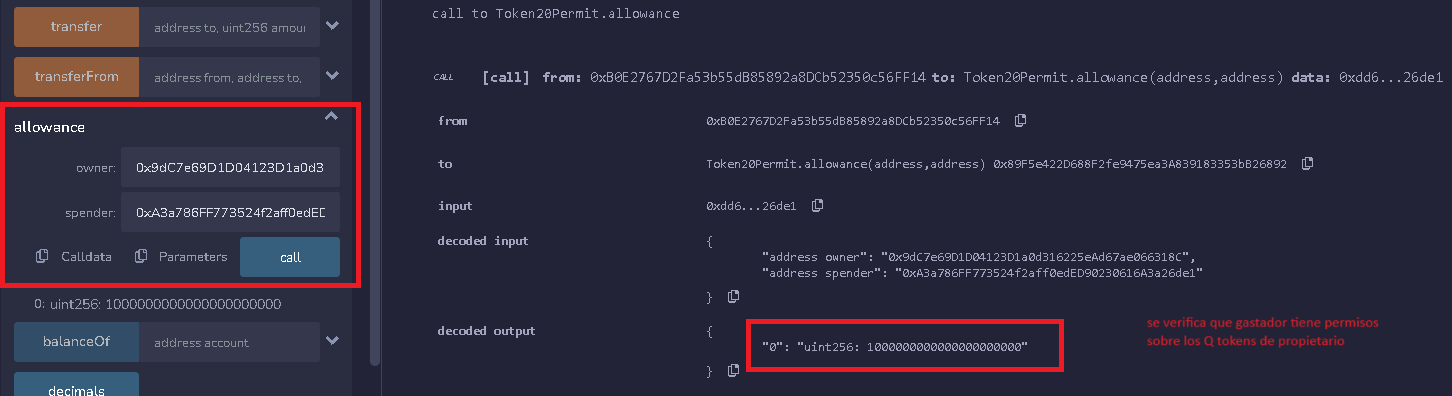
|  |
| --- |
| ownerAddress: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C  spenderAddress: 0xA3a786FF773524f2aff0edED90230616A3a26de1  value: 1000000000000000000000  deadline: 1696102694  v: 28  r: 0x751559b45c4fdb1421870e0b94d94bef0559fcf9c0be4a0721af7ec42636cab3  s: 0x15608c472133389e8b80ddd0ca4beddf5303ca53595bbfd99723b4ba0cdd01ab  Address recuperada de firma: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C |





Hash de la transacción (permit): 0x92e0c13d66e8c7e746817ae3e7eda7b6e31c9a65cf925a4d451dd85f82d853ff





**LABORATORIO: Interactuando con los métodos del estándar ERC721**

Al deployar NFTERC721:

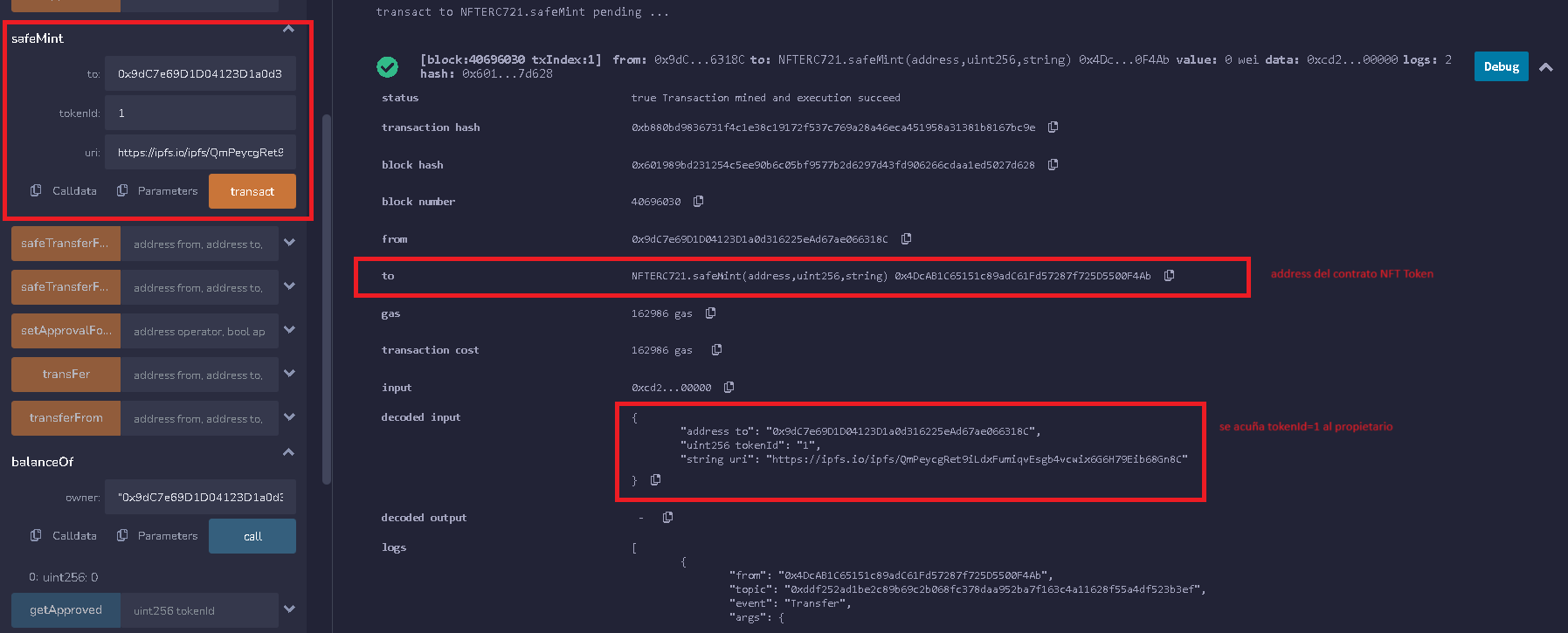
Address propietario: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

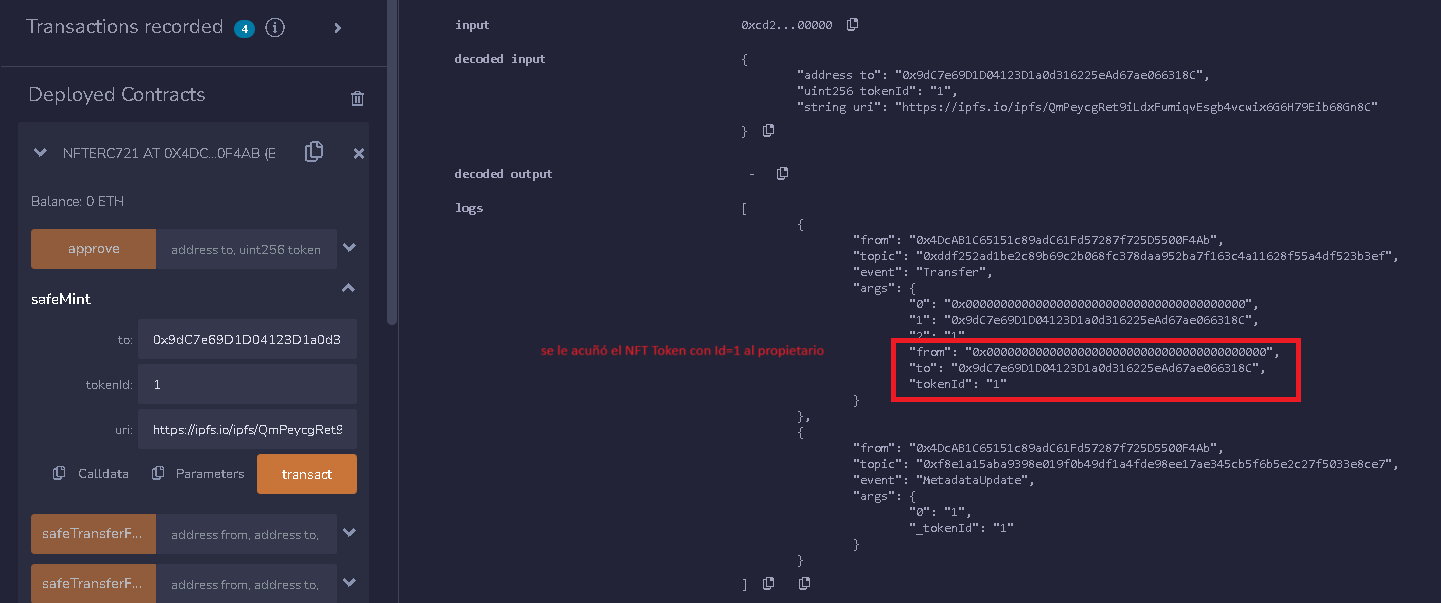
Address destinatario: 0x1b38Ba4AfcbA57f2F9E8827D29Ad7945e0b75a30

Address operador: 0x2C2aC6111276691aE07178860B5CFbcC3a34832E

Address Contrato NFT Token: 0x4DcAB1C65151c89adC61Fd57287f725D5500F4Ab

Llamamos a safemint:





Para poder ver tus NFT Acuñados ingresa a Opensea Testnets, haz clic en Conectar Modenero luego en firmar para que Opensea gestione tus NFT:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Accept y sign:

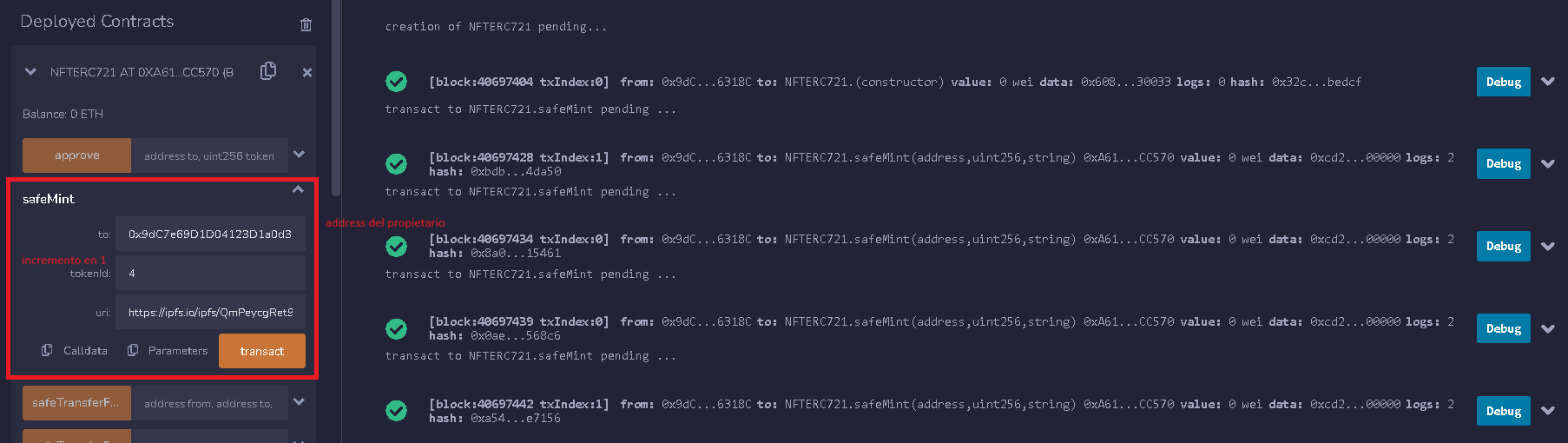
Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Volví a deployar desde REMIX el contrato Esteban NFT Token

Address Contrato NFT Token: 0xA614Fbad1f5e6B3C17E55e686753b4bbA19CC570

Luego llamé 4 veces a safemint:



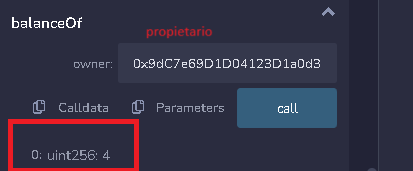
Voy a Opensea:

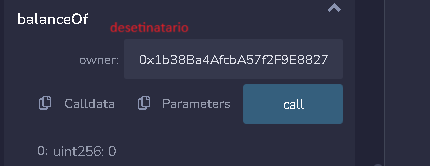
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

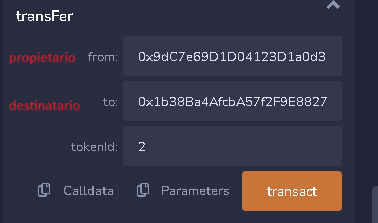




Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

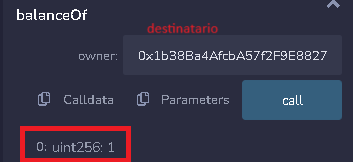
Transfer: de propietario a destinatario (tokenId=2)

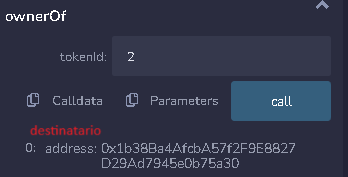


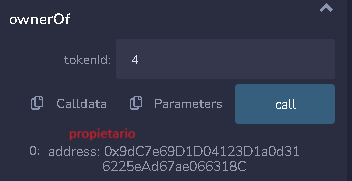


Vemos sus balances y consulta de ownerOf:



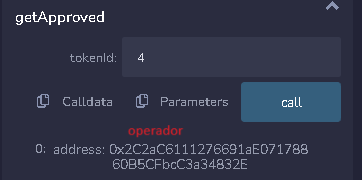






Approve de Propietario hacia Operador (un NFT especifico), ejem TokenId=4:





Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

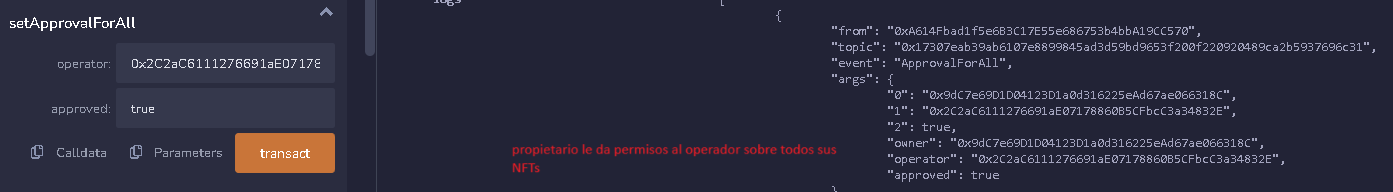
setApprovalForAll (el propietario le da permisos sobre todos sus NFTs a operador):

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

\* Individualmente no se refleja el permiso (extraño, pues debería)

Balances antes de que operador transfiera en nombre del propietario, un tokenId a Destinatario.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Una captura de pantalla de un celular

Descripción generada automáticamente

TransferFrom: de que operador transfiera en nombre del propietario, un tokenId a Destinatario

(de nuevo todo de nuevo:

Address propietario: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

Address destinatario: 0x1b38Ba4AfcbA57f2F9E8827D29Ad7945e0b75a30

Address operador: 0x2C2aC6111276691aE07178860B5CFbcC3a34832E

Address Contrato NFT Token: 0x2a28121a8eFa6e131088356AE287f34344bF1888

Captura de pantalla de un celular

Descripción generada automáticamente

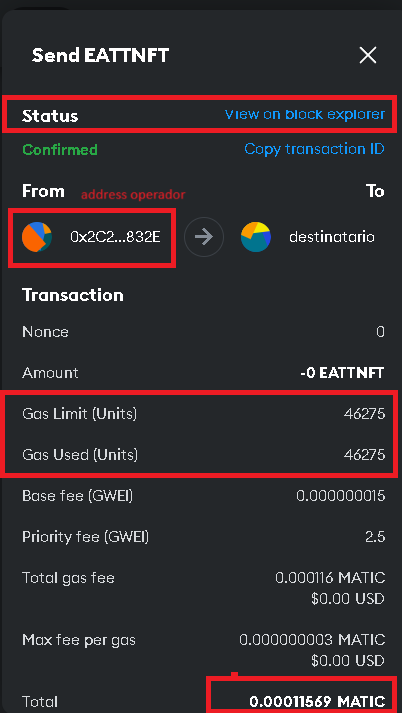


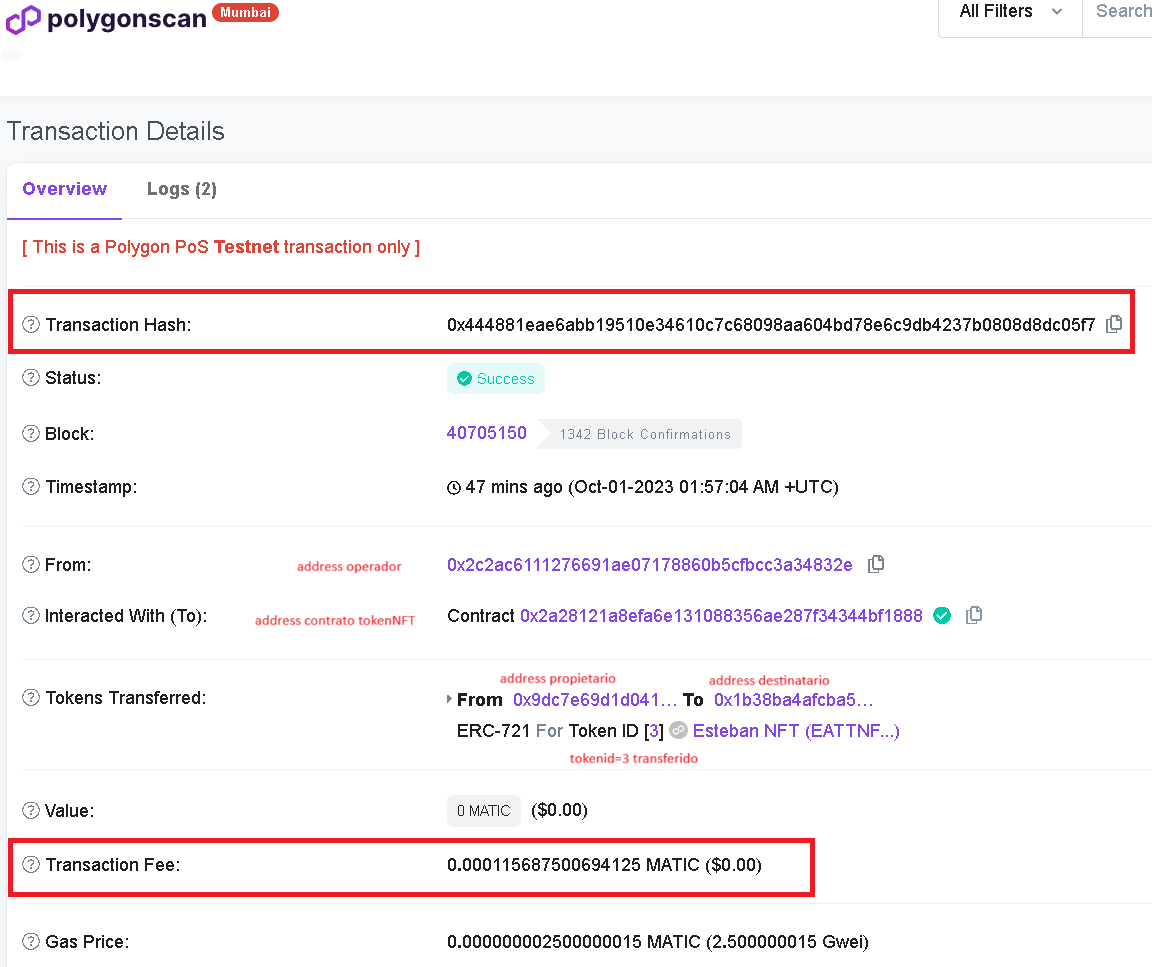


Vemos quien se cobró el gas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



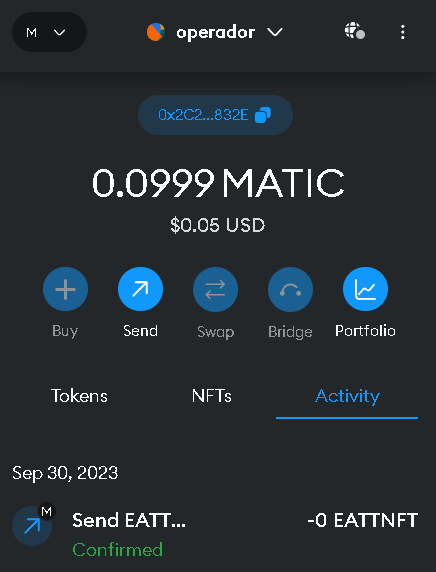


El gas se cobró de la billetera de operador.

Su saldo antes del TransferFrom era 0.1

El gas es: 0.00011569

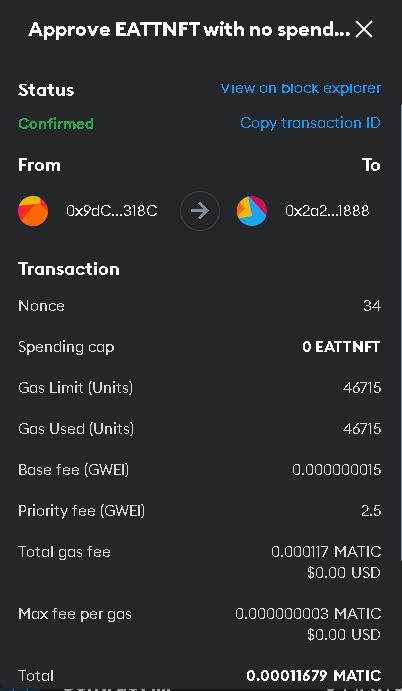
El saldo después es: 0.09988431



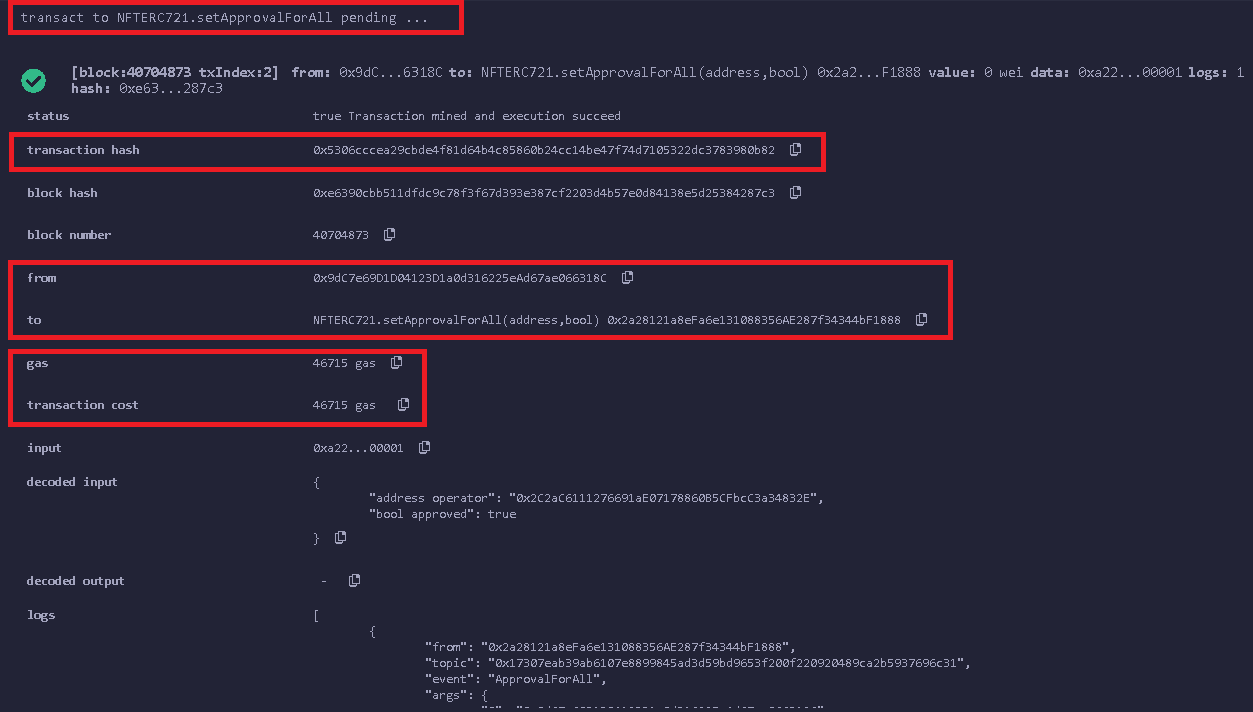
La billetera de propietario no sufrió gastos de la tx de transferfrom iniciada por operador.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente







Conclusión: hay un error en el paso 20

Como se puede apreciar en el gráfico del método approve, el Propietario concede autorización al Operador para administrar el NFT en especifico en su nombre, pero no se realiza la transferencia en ese momento. **Cuando el Operador decida llevar a cabo la transferencia del NFT, el costo de la transacción se deducirá de la billetera del Propietario, mientras que la del Operador mantendrá su balance intacto**. Es importante destacar que la propiedad de los NFTs no cambia en este proceso; los dueños de los NFTs siguen siendo los mismos.

Conclusión: hay un error en el paso 20

* ¿Qué es lo que más te llamó la atención de este estándar?
* ¿Alguna funcionalidad que crees que está faltando? ¿Qué quitarías o añadirías?

**LABORATORIO: ERC721 Enumerable: Enumeración de NFT**

Address propietario: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

Address destinatario: 0x1b38Ba4AfcbA57f2F9E8827D29Ad7945e0b75a30

Address operador: 0x2C2aC6111276691aE07178860B5CFbcC3a34832E

Address Contrato NFT Token: 0x96cFA4bC640168139c4720c1015aE131e7f8827D

Resultados:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El 20 deberia aparecer al último.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**DESAFIO 9:**

Address del contrato AirdropOneCrossChain (en mumbai): 0x3e5929B5A832a0Cf6b4Cb7C3DcC10013e44a8c42

Address del contrato MiPrimerTokenCrossChain (en sepolia): 0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2

<https://defender.openzeppelin.com/>

Luego de registrarme y loguearme, registro un Relay:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Add first Relayer:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Create:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Address del relayer (Sepolia): 0x21159912C5D49aE5e282346fBD921aE44248Bd64

Le envío algunos SepoliaETH pues su saldo de de 0. Lo necesitará para pagar el gas cuando invoque a mint del contrato Token.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Listo.

Ahora agregamos el autotask:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Add first Autotask:

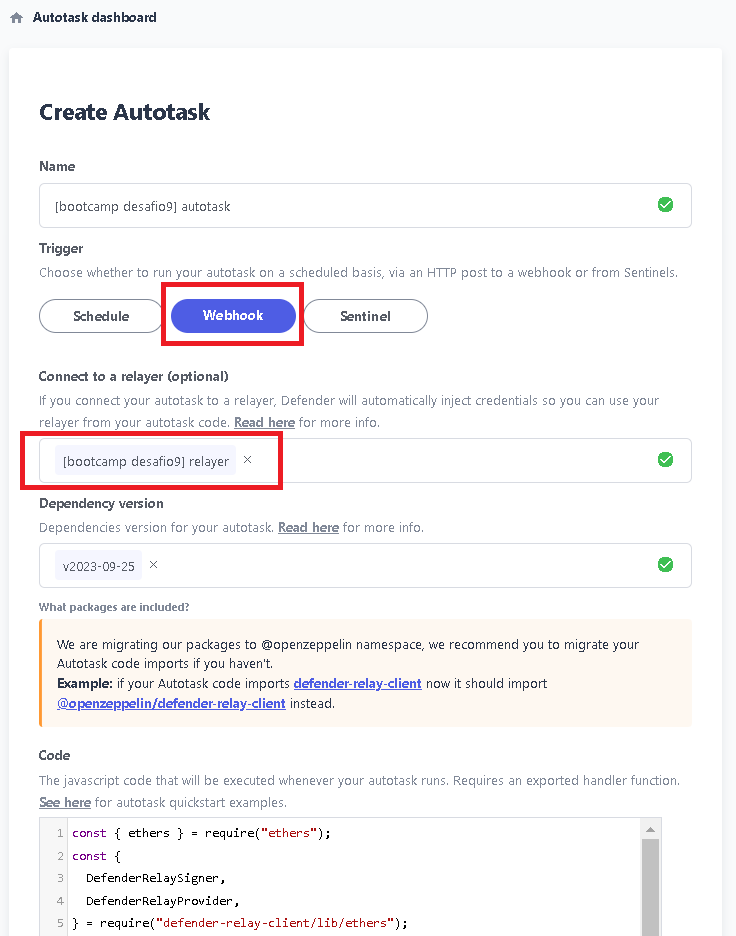


Imagen que contiene Texto

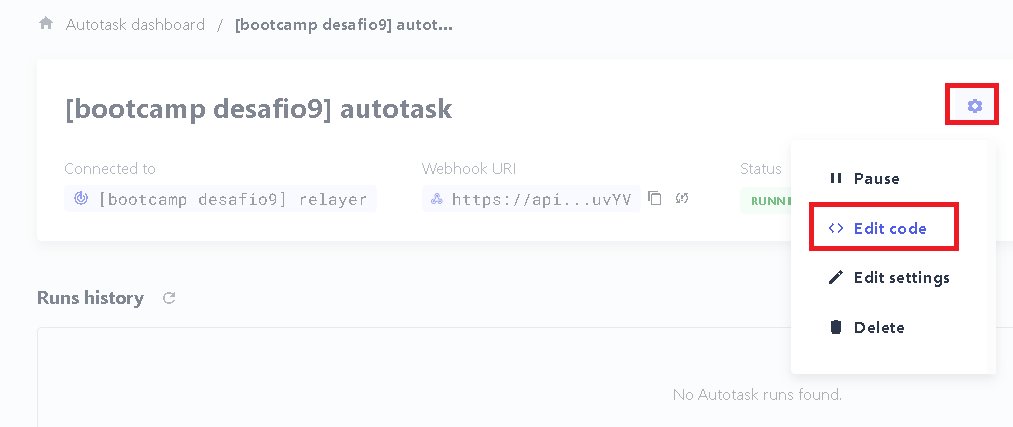
Descripción generada automáticamente

Create:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Edito el autotask pues me olvidé en modificar en el script el valor de miPrimerTokenAdd = 0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahora agregamos el sentinel:

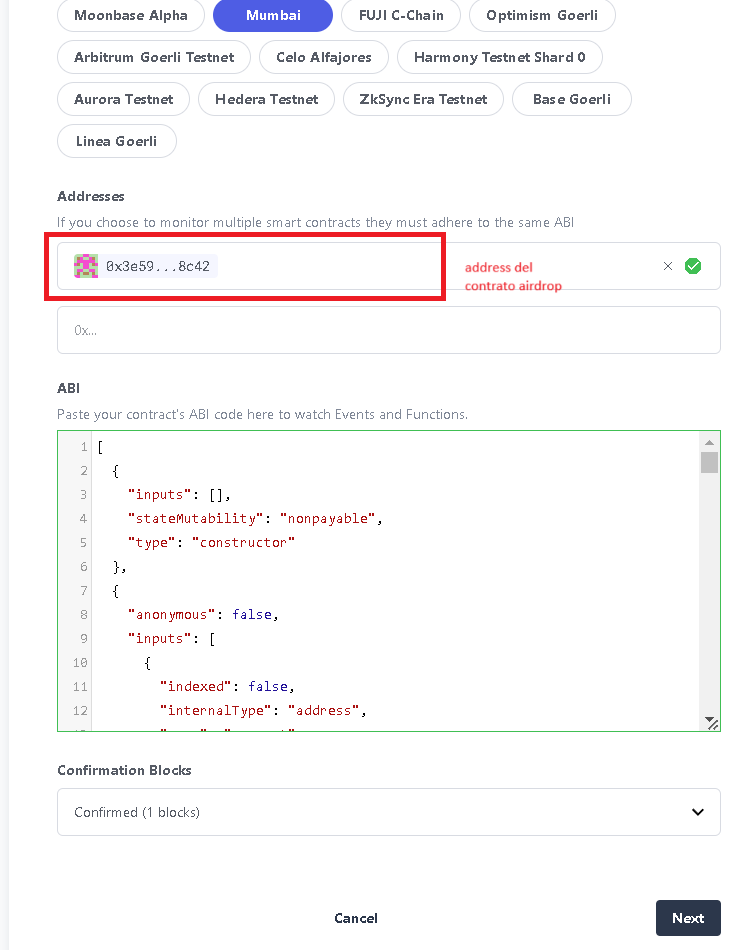
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Add first Sentinel:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente



Next:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Create Sentinel:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

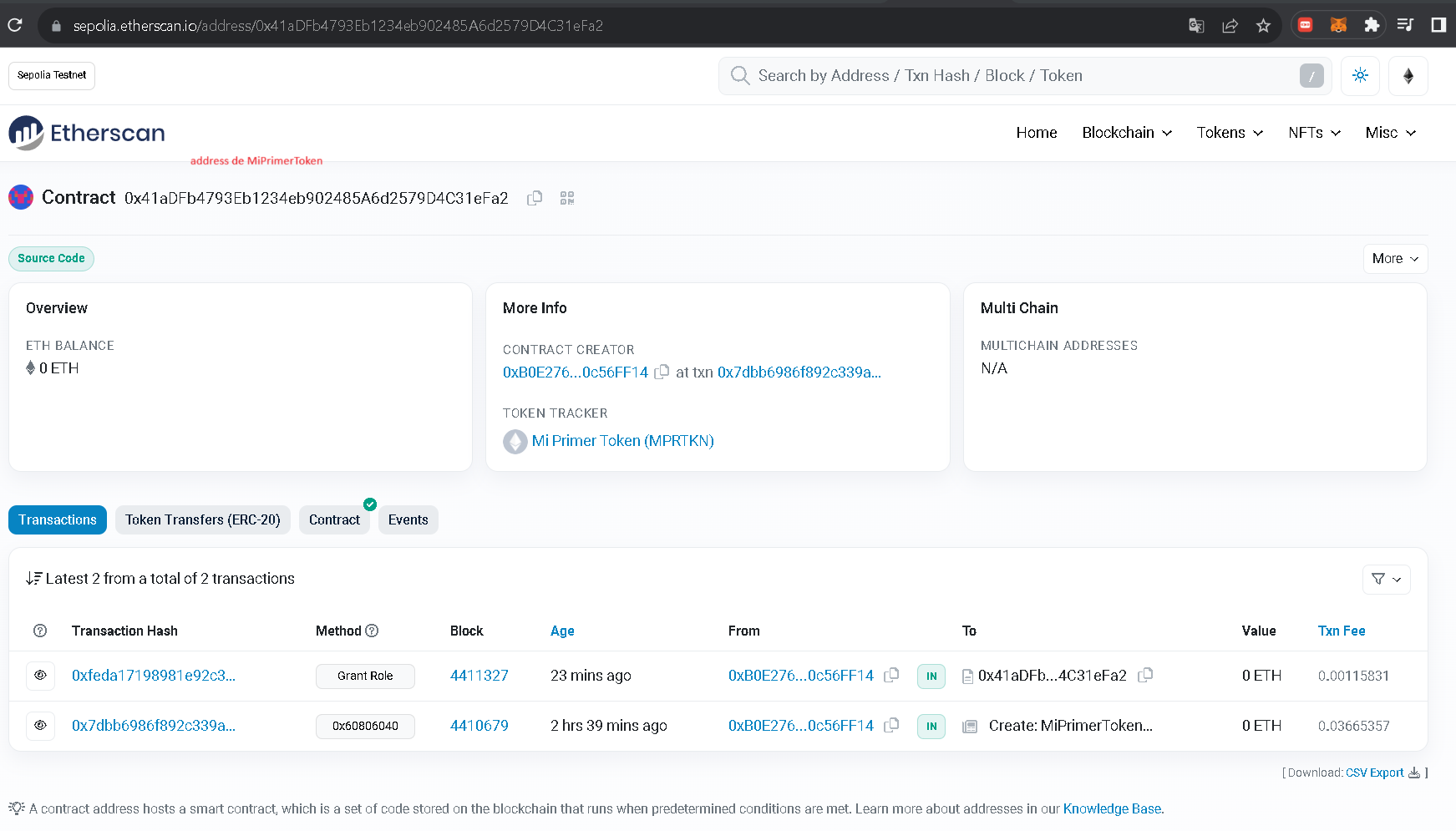
Descripción generada automáticamente

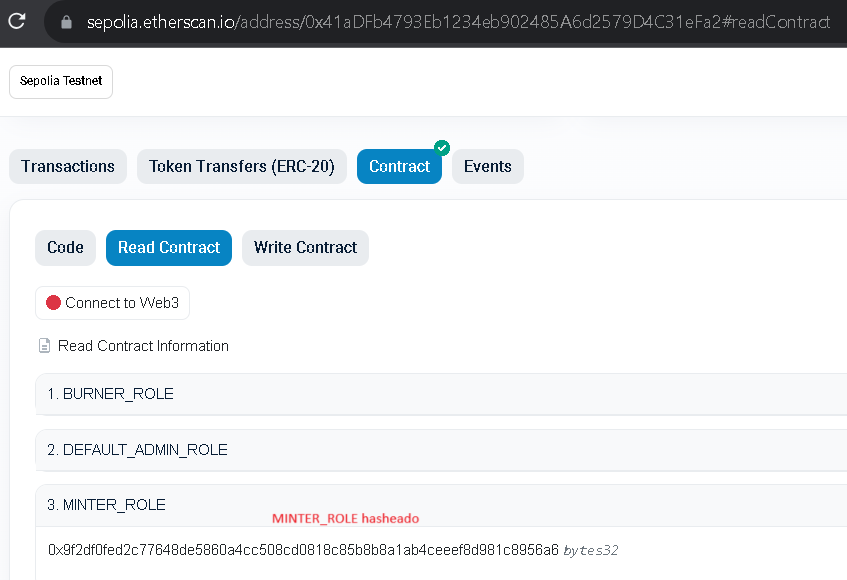
El sentinel está monitoreando al address del Airdrop en Mumbai: 0x3e5929B5A832a0Cf6b4Cb7C3DcC10013e44a8c42

Ahora hay que darle permisos de MINTER al relayer sobre el contrato del Token

Address del contrato MiPrimerTokenCrossChain (en sepolia): 0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2

Voy a: <https://sepolia.etherscan.io/address/0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2>



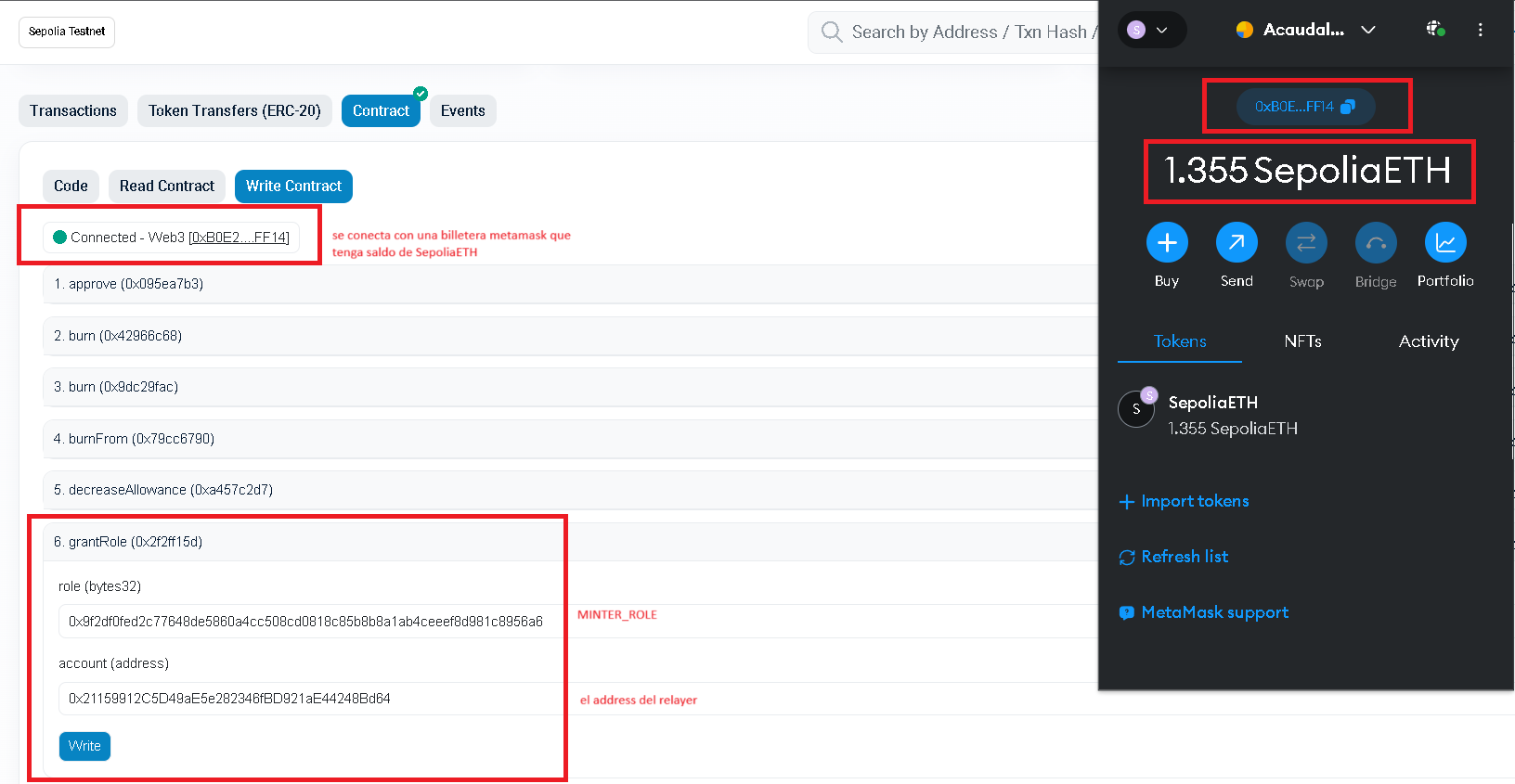


El MINTER\_ROLE es: 0x9f2df0fed2c77648de5860a4cc508cd0818c85b8b8a1ab4ceeef8d981c8956a6

Address del relayer (Sepolia): 0x21159912C5D49aE5e282346fBD921aE44248Bd64

Procedo a darle el rol de MINTER\_ROLE al relayer, con los dos datos anteriores:

Para ello conecto una billetera que tenga saldo de SepoliaETH.



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ya se le dio al relayer el minter\_role para que llame al mint del token.

Para efectos de ver el resultado final verifico el balanceOf de una de mis billeteras sobre este Token: 0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

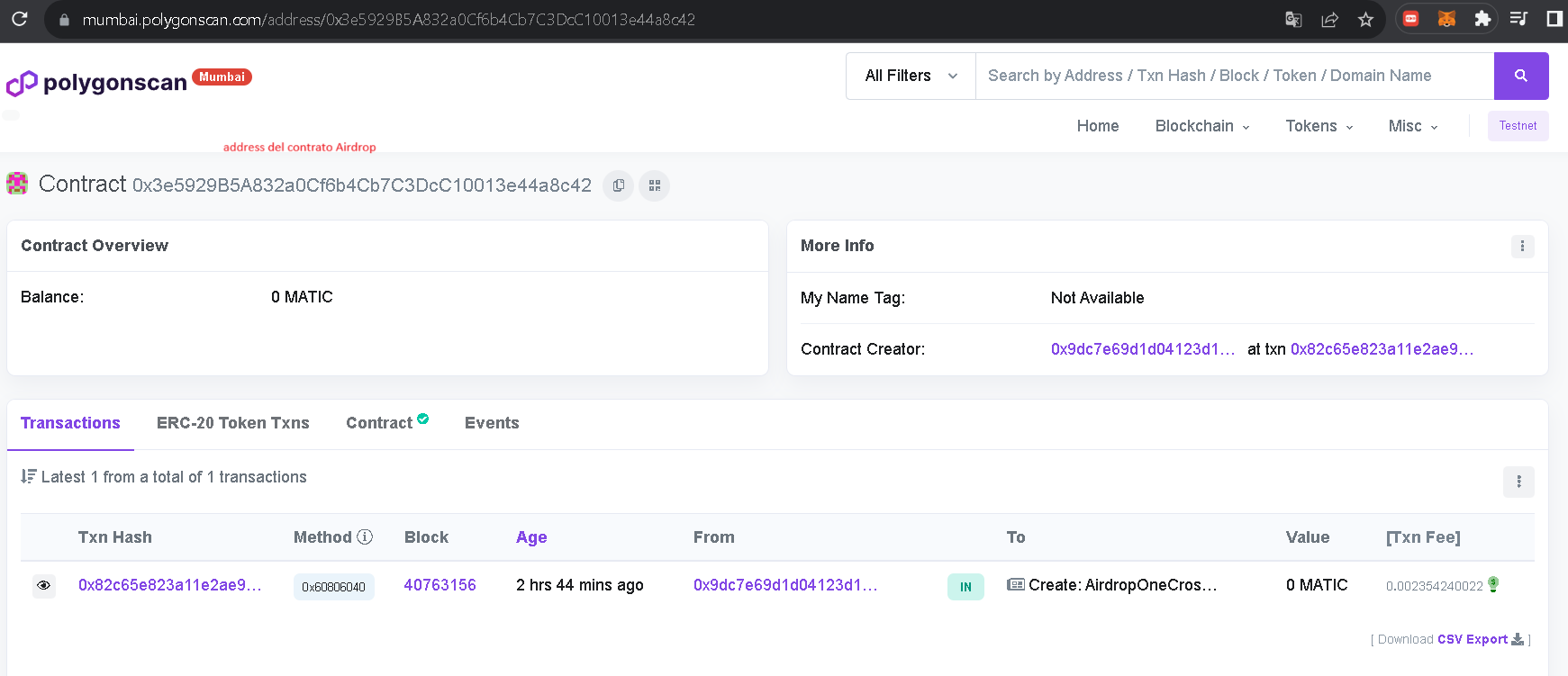
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ahora me voy al contrato del Airdrop (Mumbai):

0x3e5929B5A832a0Cf6b4Cb7C3DcC10013e44a8c42

Voy a: <https://mumbai.polygonscan.com/address/0x3e5929B5A832a0Cf6b4Cb7C3DcC10013e44a8c42>



Voy al contrato al método participateInAirdrop:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para ello conecto una billetera de Mumbai que participará en el airdrop.

Uso la billetera el cual verifique su balanceOf del Token:

0x9dC7e69D1D04123D1a0d316225eAd67ae066318C

Interfaz de usuario gráfica, Aplicación

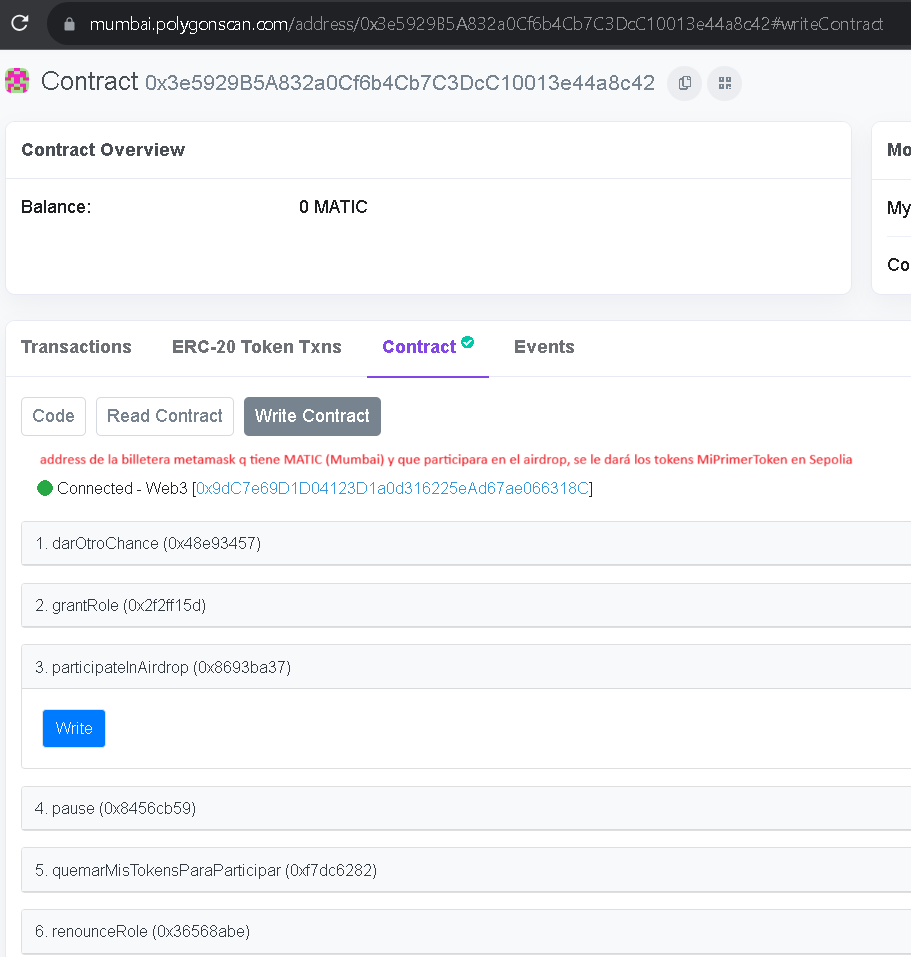
Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Captura de pantalla de un celular

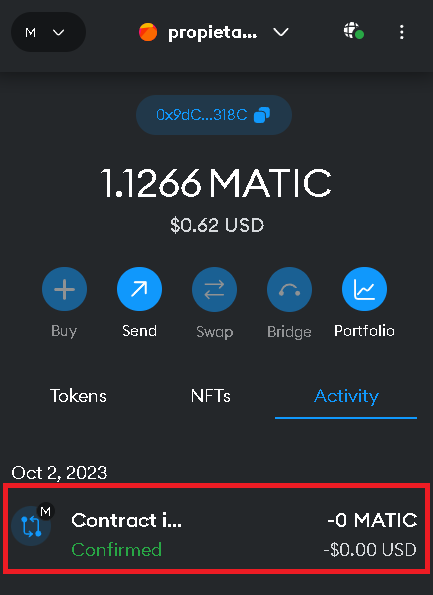
Descripción generada automáticamente

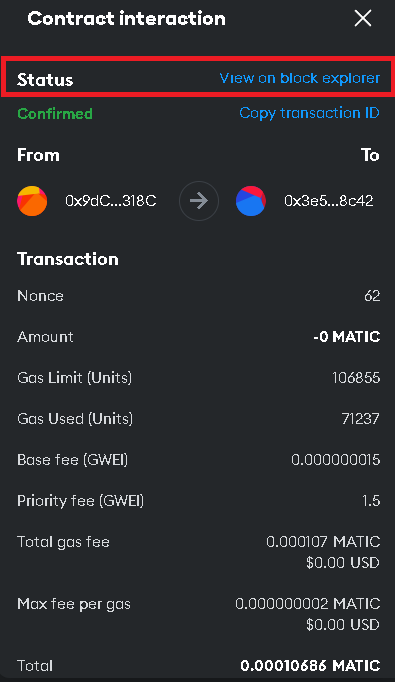


Le doy clic en “Write” y confirmo de mi metamask:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente





Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Transaction Hash:

**0xacc91064fd9797151370acfbb51c8552172ace8671032a93fe4546d5d7de6670**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Le doy en View your transaction, sale la misma pagina anterior.

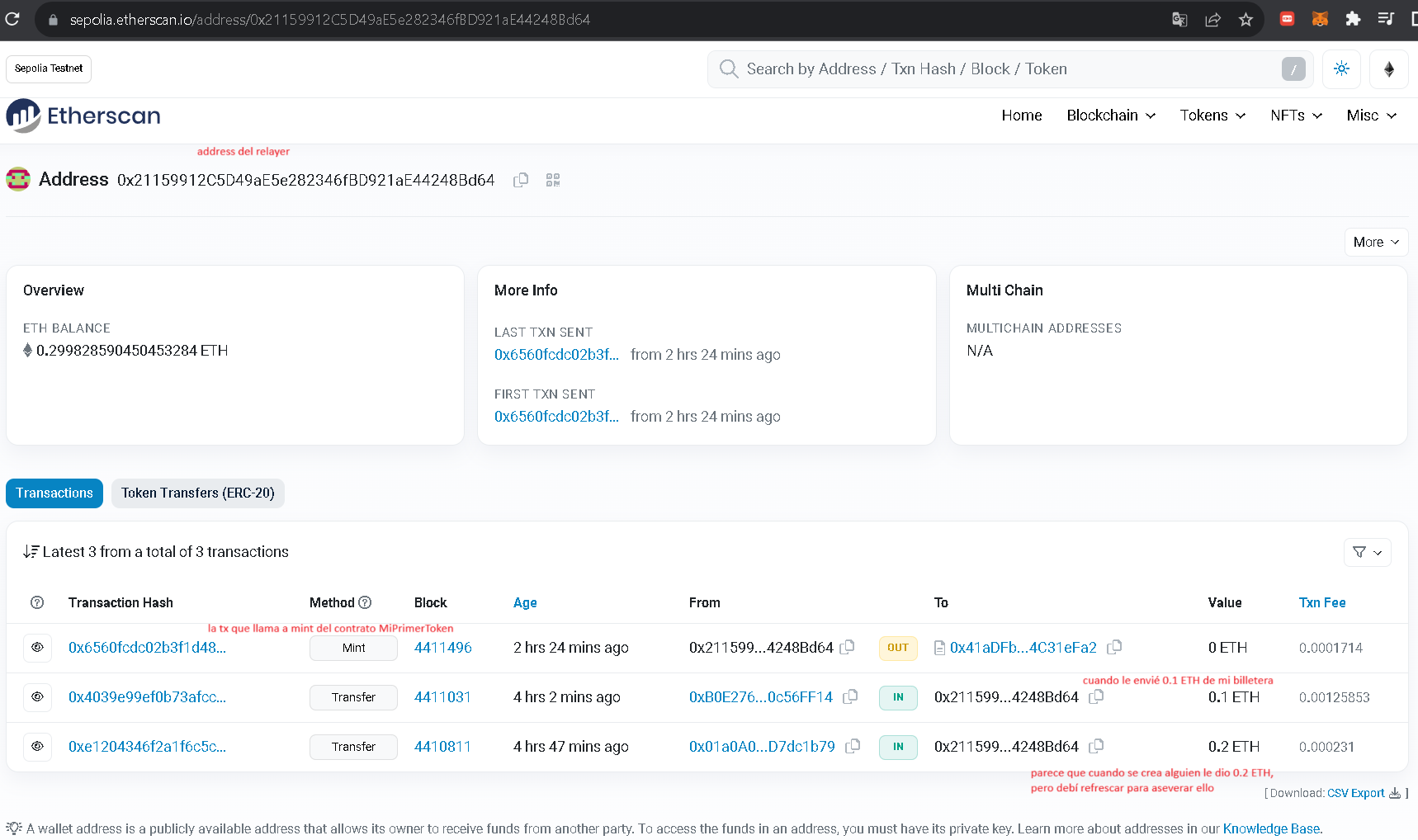
Ahora me fijo en la transacción que sale desde el relayer hacia el contrato Token (Sepolia)

Address del relayer (en sepolia): 0x21159912C5D49aE5e282346fBD921aE44248Bd64

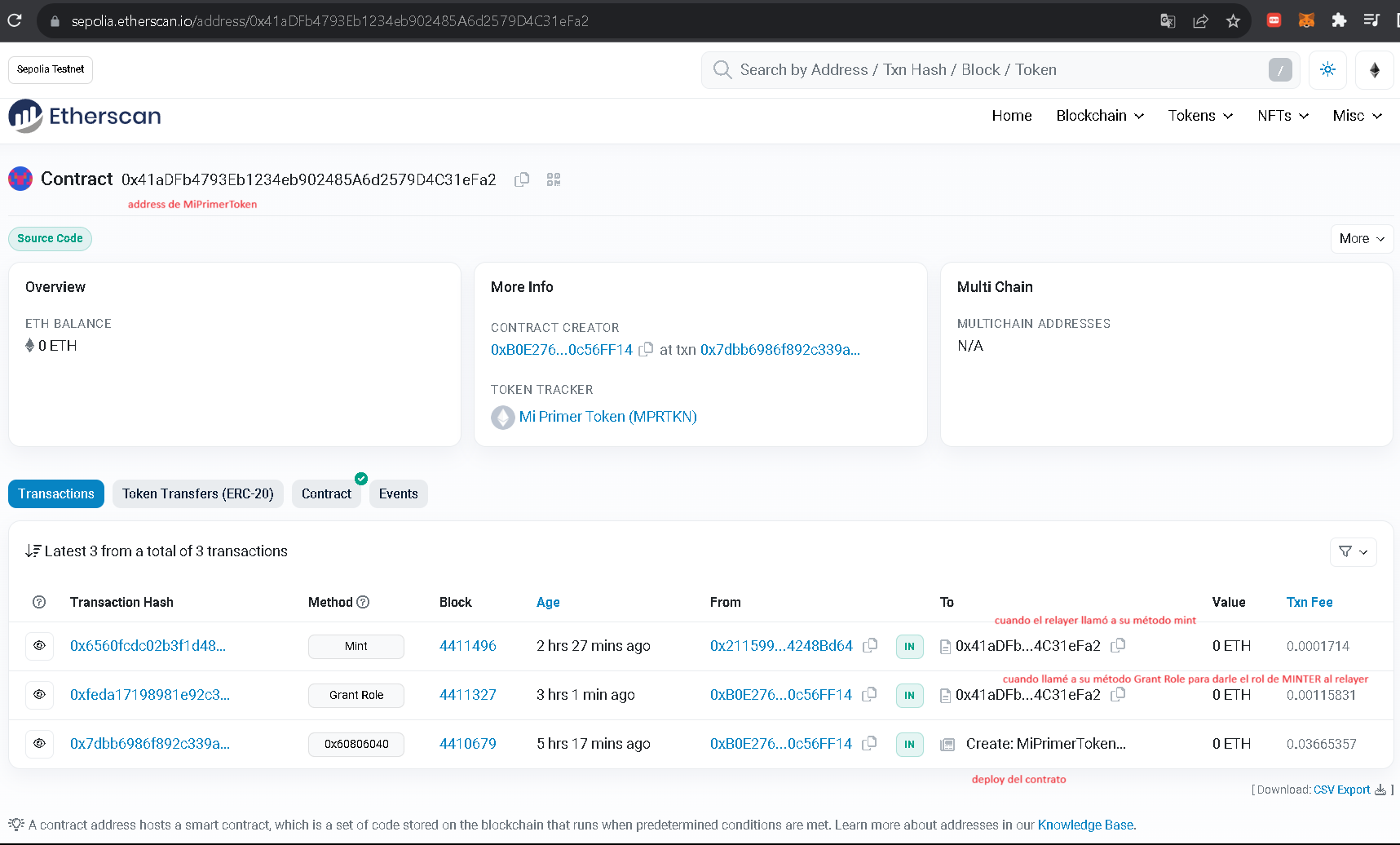
Address del contrato Token (en sepolia): 0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2

Lo busco ambos en el scan de sepolia:

<https://sepolia.etherscan.io/address/0x21159912C5D49aE5e282346fBD921aE44248Bd64>



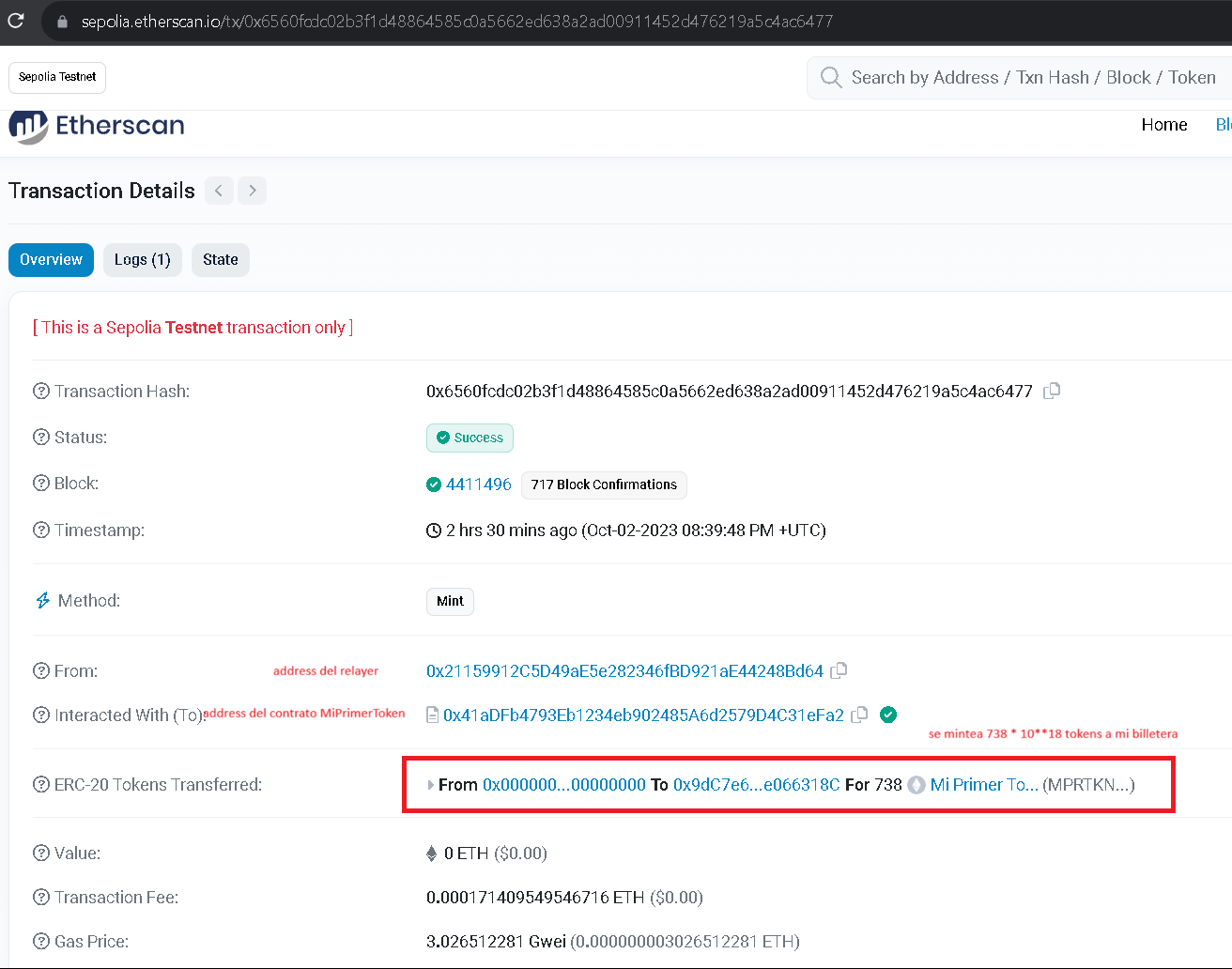
<https://sepolia.etherscan.io/address/0x41aDFb4793Eb1234eb902485A6d2579D4C31eFa2>



Copió el HASTA de la transacción cuando el relayer llamó al mint de MiPrimerToken:

Transaction Hash:

**0x6560fcdc02b3f1d48864585c0a5662ed638a2ad00911452d476219a5c4ac6477**



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## [Tarea](https://github.com/Blockchain-Bites/batch-02-bootcamp-student/blob/main/contracts/desafios/Desafio_9.md#tarea)

1. Pega aquí la transacción hash de Mumbai luego de participar en Airdrop:

**0xacc91064fd9797151370acfbb51c8552172ace8671032a93fe4546d5d7de6670**

1. Pega aquí la transacción hash de Sepolia que acuña los tokens ERC20:

**0x6560fcdc02b3f1d48864585c0a5662ed638a2ad00911452d476219a5c4ac6477**

**LABORATORIO: Contratos Actualizables**

npx hardhat --network mumbai run contracts\laboratorios\lab\_upgradeable\scripts\deployProxy.js

Compiled 3 Solidity files successfully

Address del Proxy es: 0xAB95222Fb6268eabc114fd2e32808FaD1143C3c0

Address de Impl es: 0x3E97B789898048287226bA651fCeeef82778c115

Successfully submitted source code for contract

contracts/laboratorios/lab\_upgradeable/contracts/EAGTokenUpgradeable.sol:EAGTokenUpgradeable at 0x3E97B789898048287226bA651fCeeef82778c115

for verification on the block explorer. Waiting for verification result...

Successfully verified contract EAGTokenUpgradeable on the block explorer.

https://mumbai.polygonscan.com/address/0x3E97B789898048287226bA651fCeeef82778c115#code



Me voy a Contract:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Voy a las pestañas de Read Contract y Write Contract:

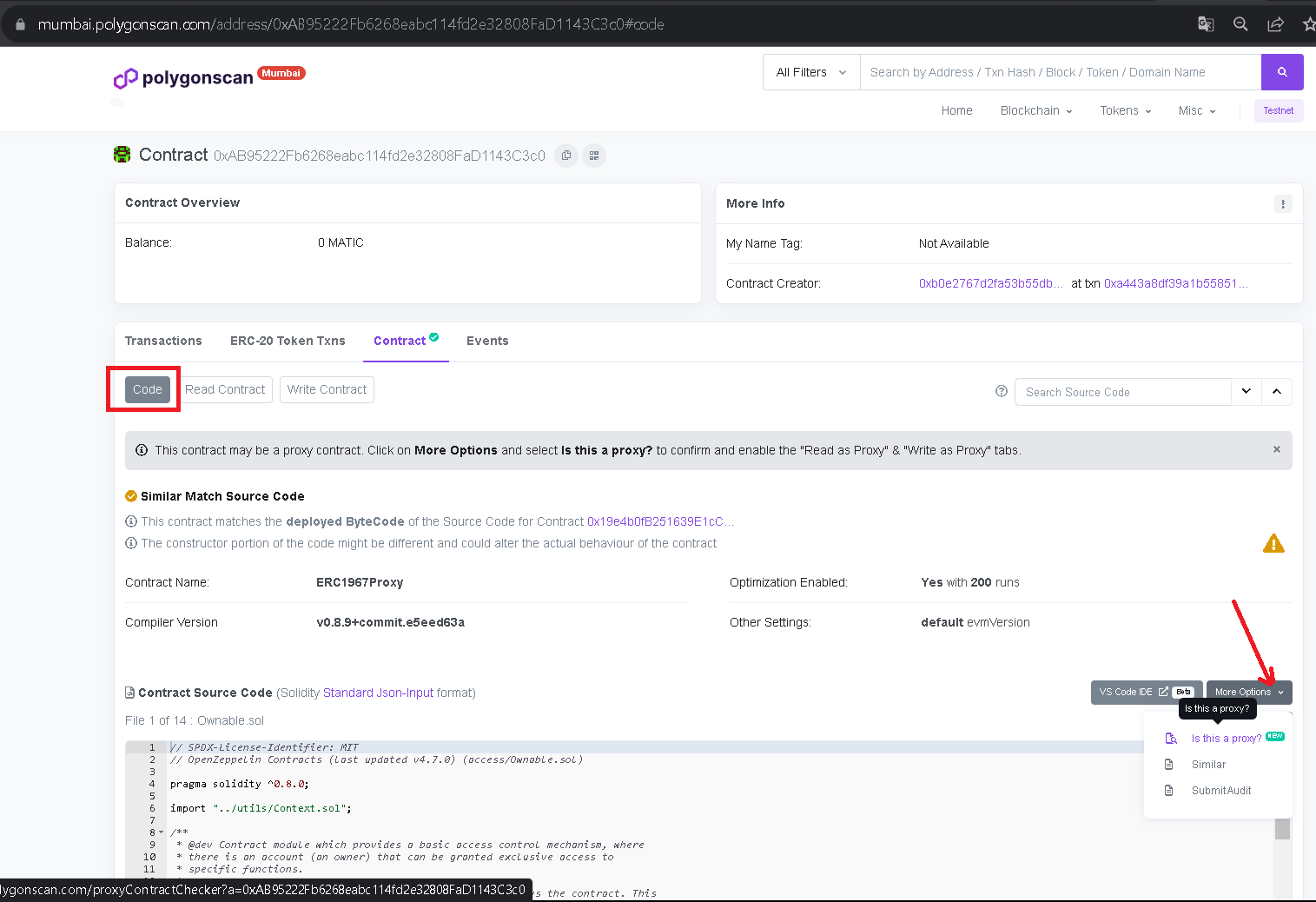
Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Teams

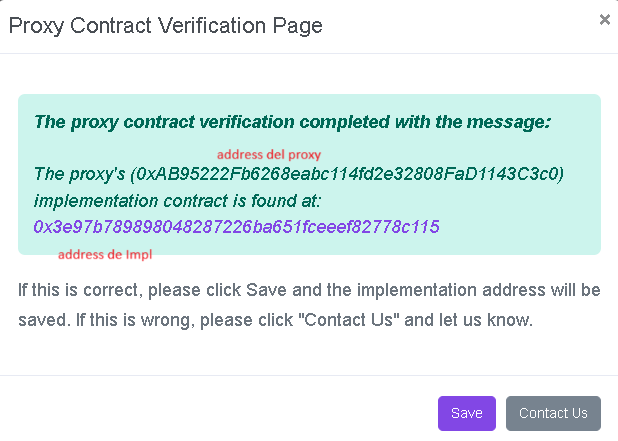
Descripción generada automáticamente

Le doy clic en “More Options”:

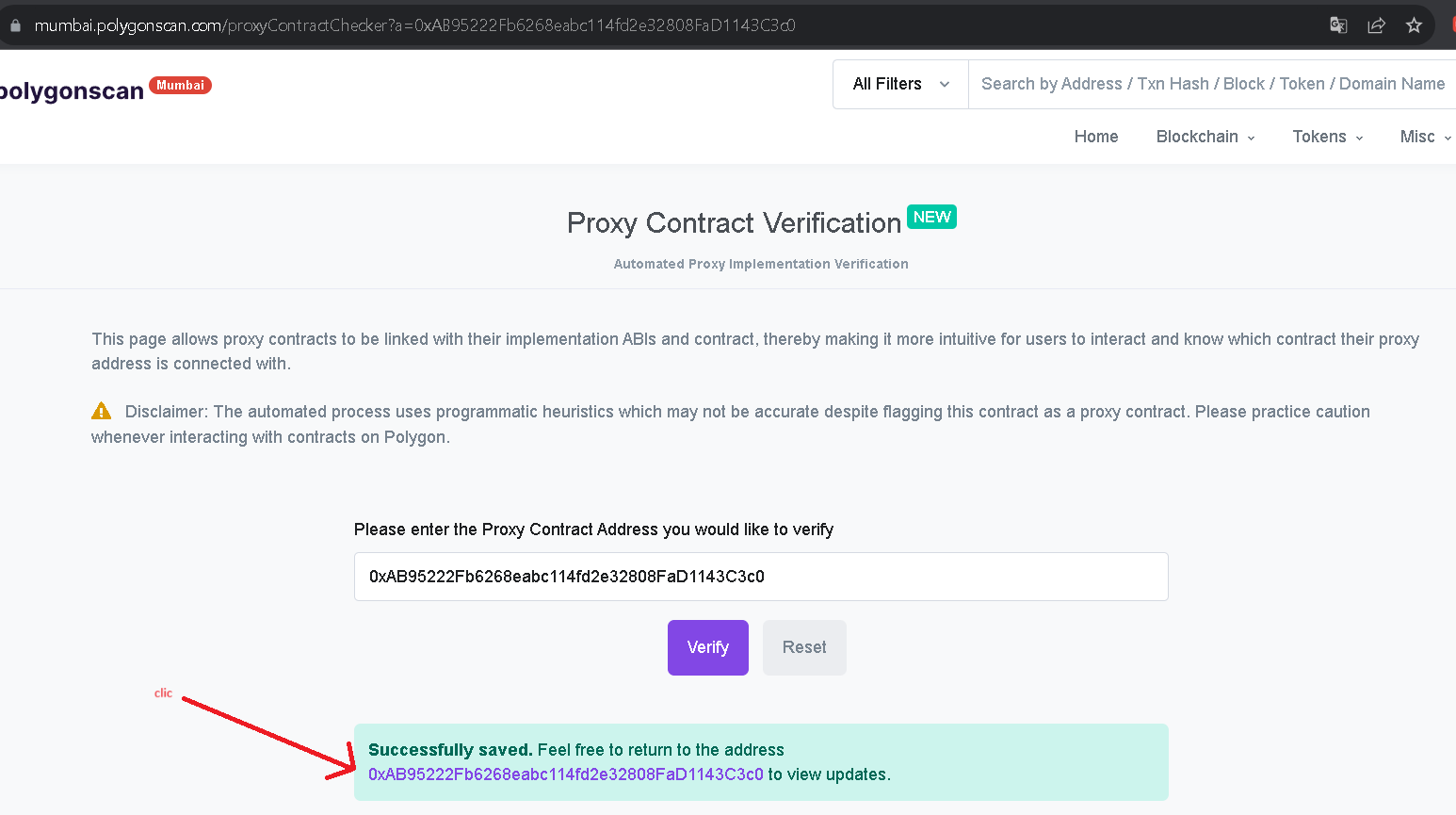


Interfaz de usuario gráfica, Aplicación, Teams

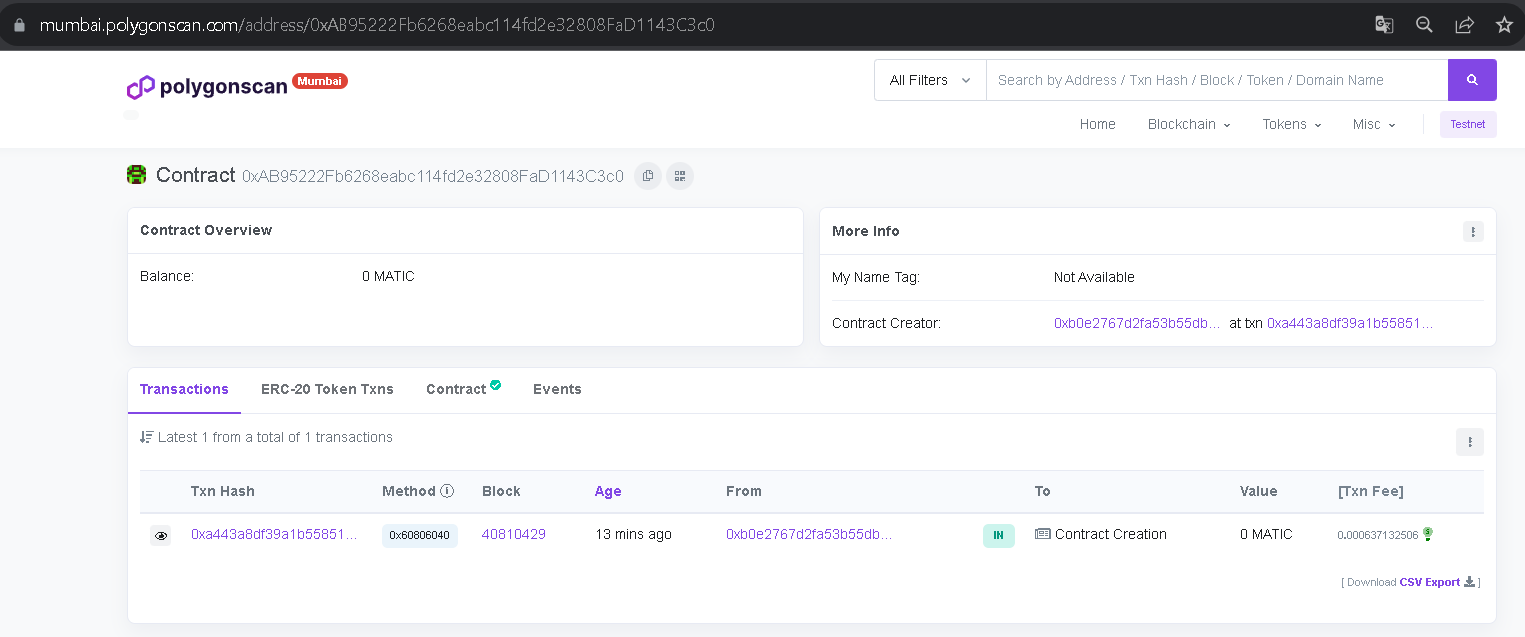
Descripción generada automáticamente



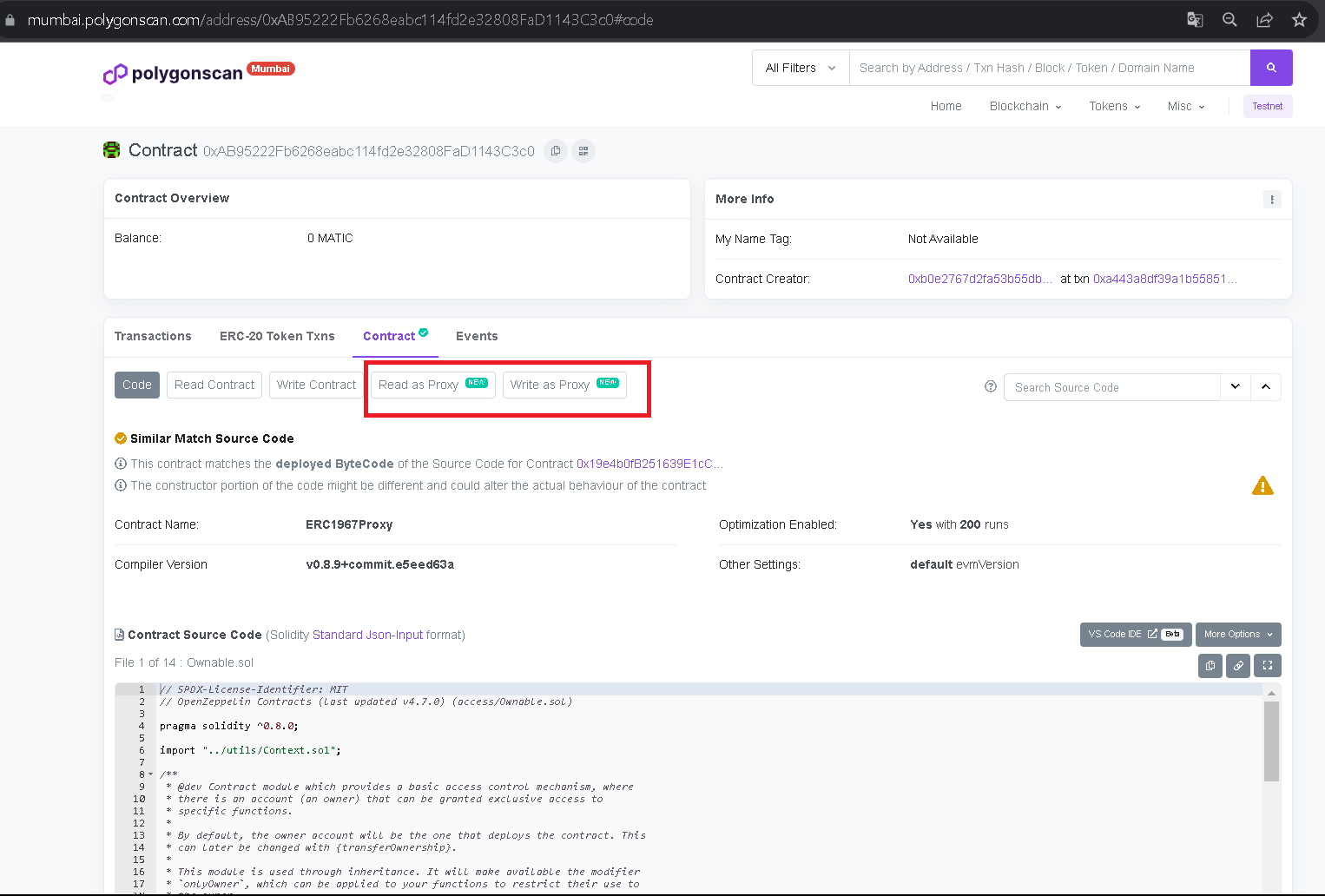
Clic en Save:



Regreso al contrato Proxy:



Me voy a Contract:



Voy a Read as Proxy:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

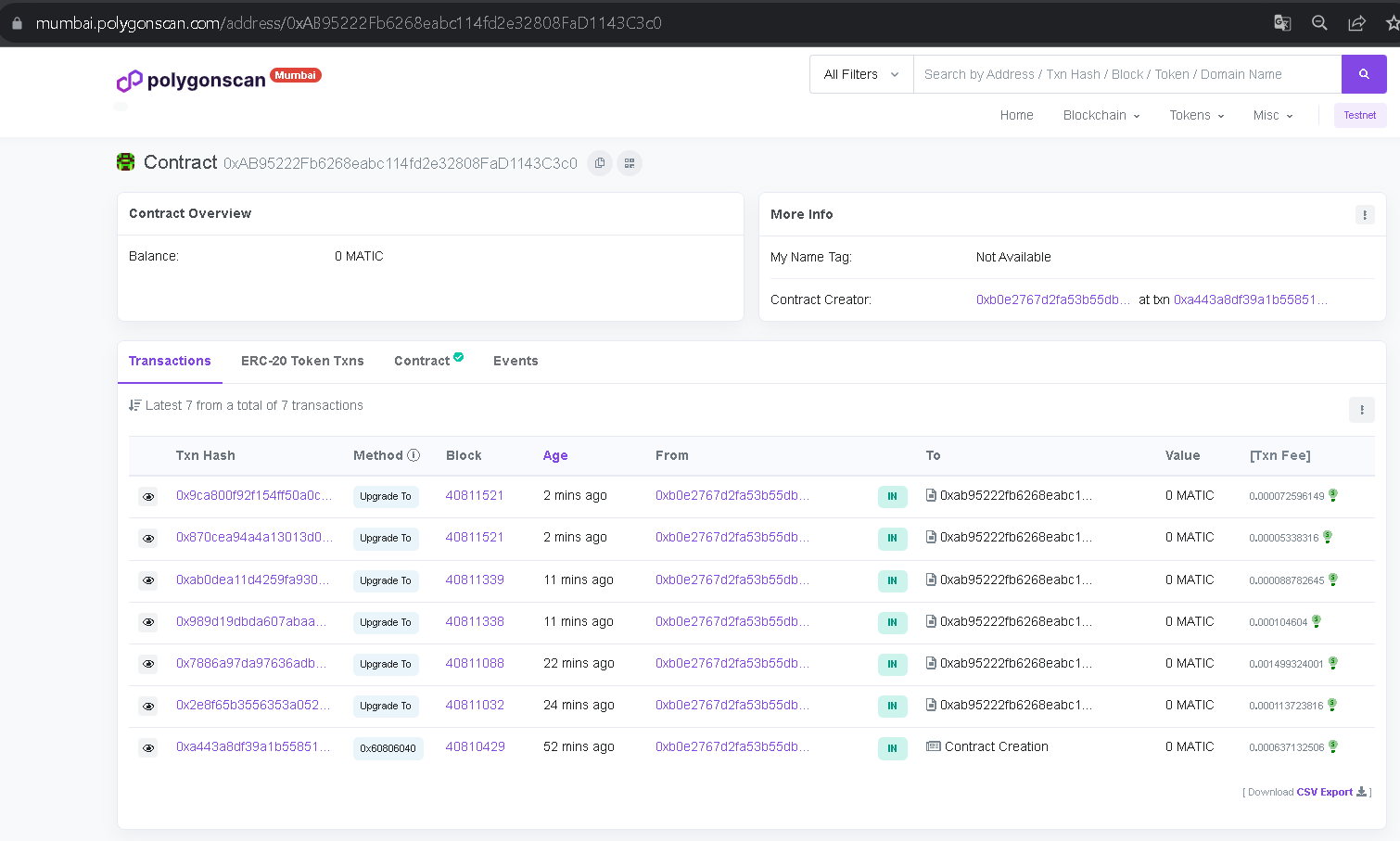
Voy a Write as Proxy:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ahora implemento una nueva versión (ejem V2):

Ejecutamos el script (varias veces intentando hacer la confirmación de bloques pero no pude hacerlo):



Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

Address de ImplV2: 0x5B77eB2F5696dB77051DcA03eD2b2689906CaEb5

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En ReadContract ya aparece el nuevo método:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En WriteContract igual:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente