

# Universidade de Vigo

## **Tecnoloxías de rexistro distribuído e Blockchain**

### Práctica 1 - Ejercicio 5

Por: Pablo Pérez Paramos y Alexandre Moinelo Rodríguez

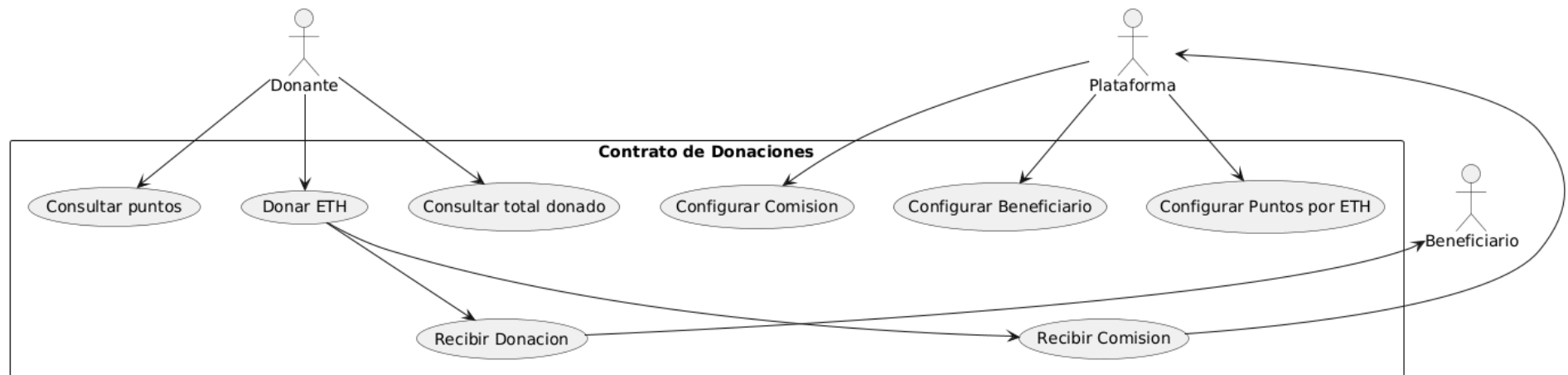
## Índice de contenidos

(1) Análisis y definición del escenario, debe requerir la funcionalidad proporcionada por una blockchain pública como Ethereum.....	3
(2) Diseño.....	4
(3) Implementación.....	5
(4) Pruebas.....	6

**(1) Análisis y definición del escenario, debe requerir la funcionalidad proporcionada por una blockchain pública como Ethereum.**

Hemos implementado un diseño de contrato en el que simulamos que el creador del contrato quiere destinar un porcentaje de su venta de Tokens mediante ETH a una donación. El diseño se basa en un comprador que adquiere Tokens y este precio en ETH se dividirá en la donación y la comisión, la comisión se quedará en la plataforma (La dirección que crea el contrato) y el porcentaje de la donación irá al beneficiario. La plataforma podrá cambiar de beneficiario o su porcentaje de comisión.

## (2) Diseño



### (3) Implementación

Te dejamos aquí nuestros enlaces a github:

[https://github.com/xMoiMoi/BC\\_Master\\_Ciber](https://github.com/xMoiMoi/BC_Master_Ciber)

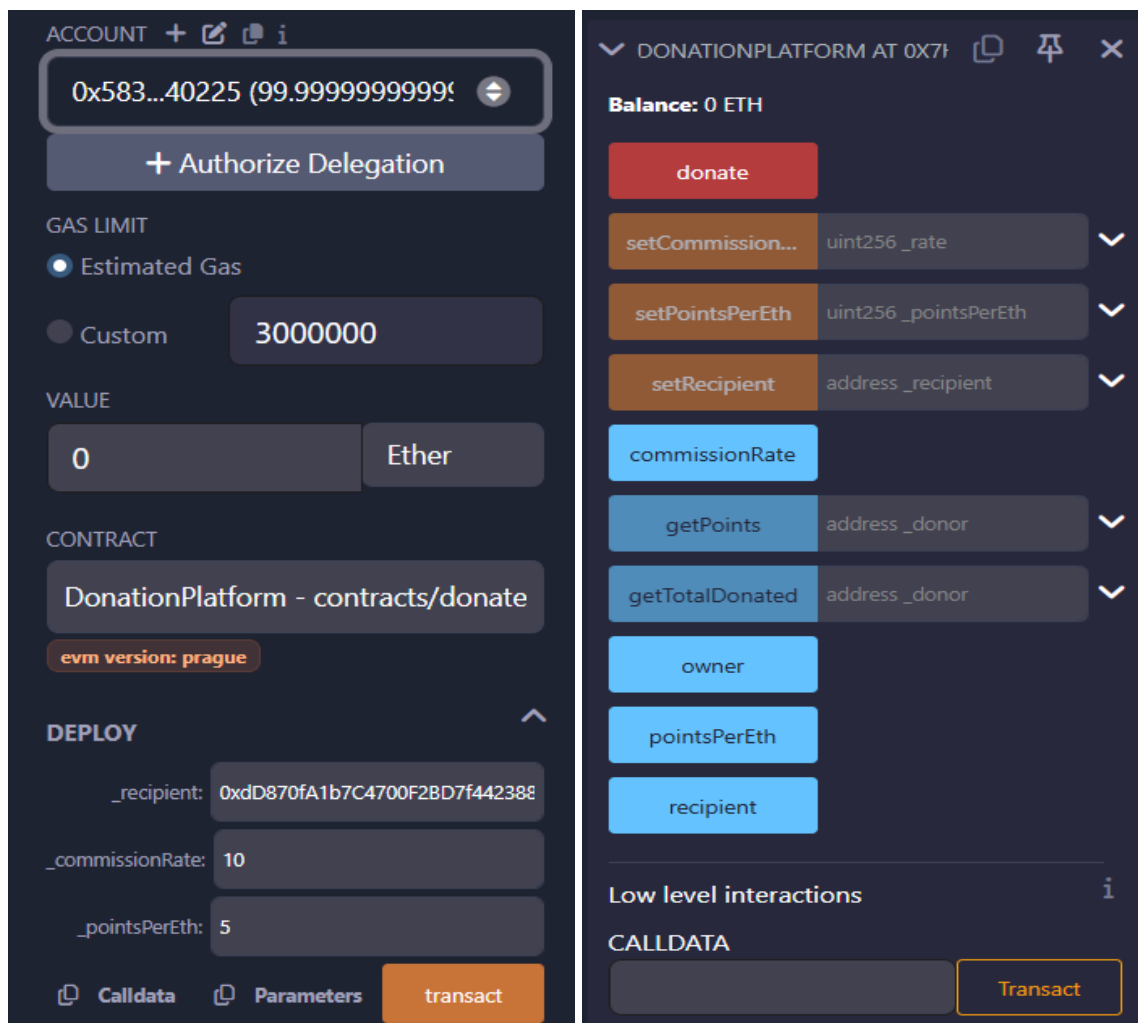
<https://github.com/przpablo/BC-practicas>

## (4) Pruebas

En esta primera imagen, mostramos la forma de crear un contrato. Con la cuenta `0x583031D1113aD414F02576BD6afaBfb302140225` creamos el contrato de forma que establecemos a la cuenta `0xD870fA1b7C4700F2BD7f44238821C26f7392148` como la que recibe las donaciones, establecemos también que el porcentaje de donación por compra de Token será del 10% y cuantos Tokens se llevará el comprador por cada ETH. Generando este decoded input: {

```
"address_recipient": "0xD870fA1b7C4700F2BD7f44238821C26f7392148",
"uint256_commissionRate": "10",
"uint256_pointsPerEth": "5"
```

}



**ACCOUNT** +   
 0x583...40225 (99.9999999999%)   
 + Authorize Delegation

**GAS LIMIT**   
☒ Estimated Gas   
☐ Custom 3000000

**VALUE**   
 0 Ether

**CONTRACT**   
 DonationPlatform - contracts/donate   
 evm version: prague

**DEPLOY**   
 \_recipient: 0xD870fA1b7C4700F2BD7f44238821C26f7392148   
 \_commissionRate: 10   
 \_pointsPerEth: 5   
 Calldata Parameters **transact**

**DONATIONPLATFORM AT 0x7f...**   
**Balance:** 0 ETH

**donate**

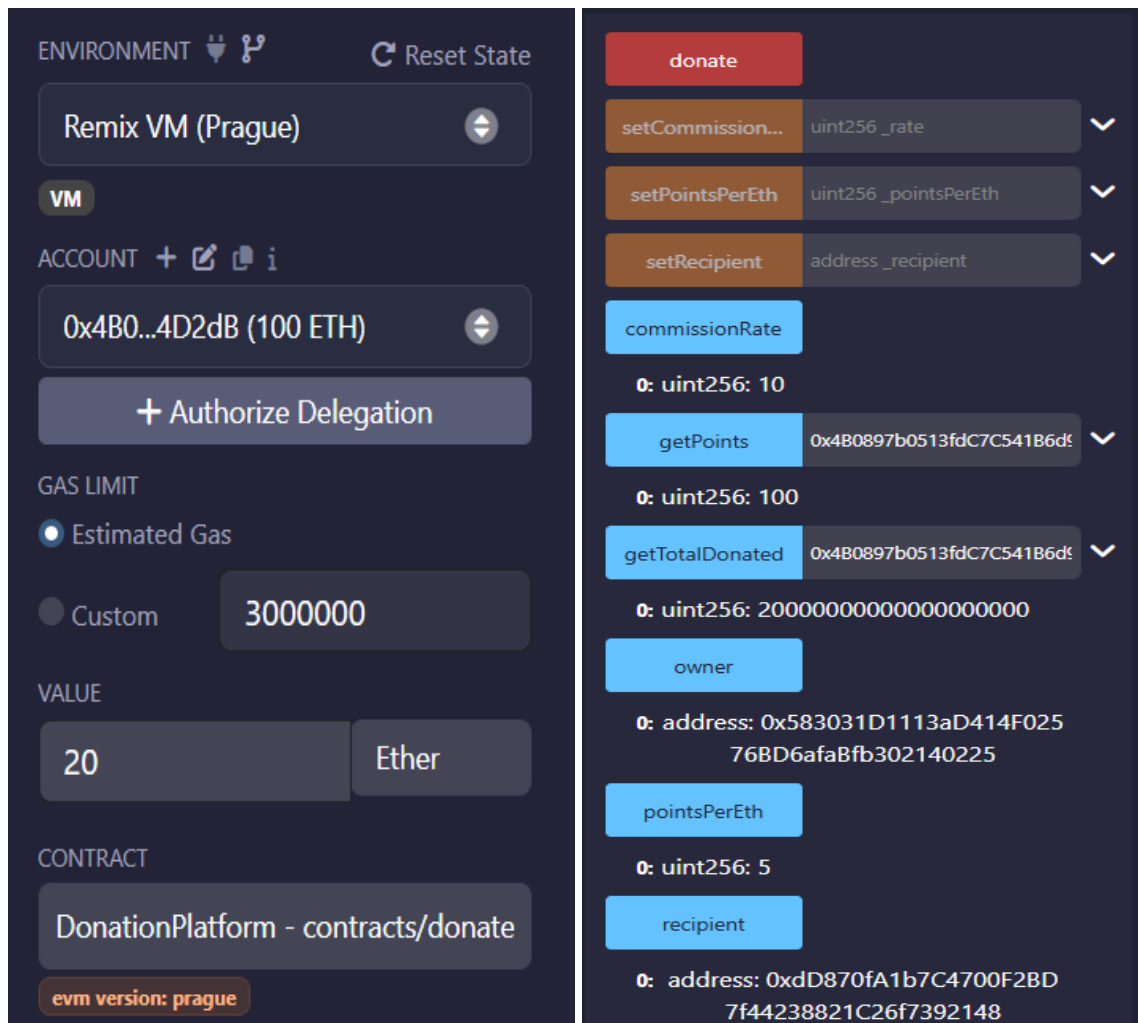
setCommissionRate uint256\_rate   
 setPointsPerEth uint256\_pointsPerEth   
 setRecipient address\_recipient

commissionRate   
 getPoints address\_donor   
 getTotalDonated address\_donor

owner   
 pointsPerEth   
 recipient

**Low level interactions**   
**CALLDATA**   
 Transact

También se despliega el contrato de la segunda imagen con las funciones *Donate* (Explicamos como usarla a portiori), *setComission* (Para cambiar el % de comisión a donar), *setPointsPerEthI* (Esta función sólo está disponible para el creador del contrato), *setRecipient* (Para cambiar a la cuenta que se quiere donar), *comissionRate* (Muestra el % puesto a donar), *getTotalDonated* (Guarda la cantidad TOTAL de ETH que cada dirección ha donado al contrato), *getPoints* (Cada vez que donas recibes un punto aquí y consultando este mapping se puede ver cuantas donaciones lleva cierta dirección), *owner* (Muestra el creador del contrato y a quien le compramos los Tokens), *pointsPerEth* (Muestra los puntos que se consiguen por cada ETH comprado según el contrato) y *recipient* (Muestra la dirección de a quien se dona). En las siguientes capturas entraremos más en detalle de las funciones que consideramos más importantes o que pueden llevar a dudas.



The image shows the Remix IDE interface. On the left, the 'ENVIRONMENT' tab is active, showing 'Remix VM (Prague)' as the VM, '0x4B0...4D2dB (100 ETH)' as the account, and '3000000' as the gas limit. The 'VALUE' is set to '20' in 'Ether'. The 'CONTRACT' is 'DonationPlatform - contracts/donate'. On the right, a list of functions is displayed:

- donate** (red button)
- setCommission...** (orange button) with parameter `uint256 _rate`
- setPointsPerEth** (orange button) with parameter `uint256 _pointsPerEth`
- setRecipient** (orange button) with parameter `address _recipient`
- commissionRate** (blue button) with value `0: uint256: 10`
- getPoints** (blue button) with value `0x4B0897b0513fdC7C541B6d...`
- getTotalDonated** (blue button) with value `0: uint256: 20000000000000000000`
- owner** (blue button) with value `0: address: 0x583031D1113aD414F02576BD6afaBfb302140225`
- pointsPerEth** (blue button) with value `0: uint256: 5`
- recipient** (blue button) with value `0: address: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148`

En la primera imagen anterior realizamos la compra de 20ETH desde la dirección `0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB` al creador del contrato, es muy simple su uso, en value indicamos el número de ETH que enviamos al creador del contrato y clicando al *donate* que mostramos en una imagen anterior ya se realiza la transacción sin mayor problema. En la segunda imagen (Dcha.) entramos en detalle en las funciones explicadas anteriormente. *commissionRate*, *owner*, *pointsPerEth* y *recipient* no merecen más explicación que la captura de pantalla y la anteriormente dada, puesto que no tienen nada más importante. Sobre *getPoints* y *getTotalDonated* dada que la explicación anterior igual deja algo de dudas, esta imagen las despeja de todo tipo. En primer lugar, *getPoints* se le indica la dirección la cual queremos consultar (el comprador) y muestra los puntos conseguidos, teniendo en cuenta que por cada ETH indicamos que se consiguen 5 puntos el resultado es correcto. Por último, en *getTotalDonated* igual, indicamos al comprador, pero esta vez muestra los ETH que ha donado, el cual el resultado es también correcto.

En esta imagen indicamos cómo quedarían las direcciones tras la transacción realizada anteriormente, siendo `0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB` la encargada de la compra de ETH a la creadora del contrato que indicamos ya anteriormente cual es. En la dirección compradora podemos ver que desde los 100ETH con los que empiezan todas las direcciones ya solo le quedan 79.99999, puesto que, ha realizado una compra de tokens del contrato con 20ETH más el gasto de gas que lleva de por sí realizar esa transacción. En la dirección creadora del contrato podemos apreciar que recibe 20ETH menos el 10% y menos el gas gastado en la creación del contrato por eso le quedan 117.99999ETH y por último en la dirección que recibe las donaciones podemos apreciar como se le añadieron 2ETH al valor inicial, lo que viene siendo la suma del 10% de los 20ETH enviados en la transacción.

`0x4B0...4D2dB (79.999999999999796758 ETH)`

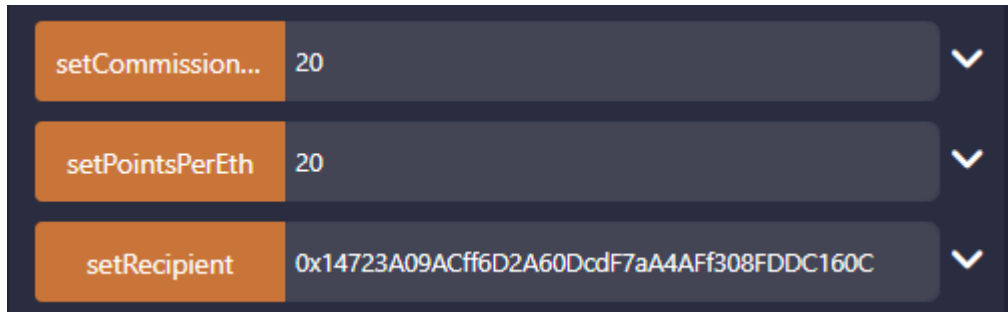
`0x583...40225 (117.999999999997749258 ETH)`

`0xD8...92148 (102 ETH)`



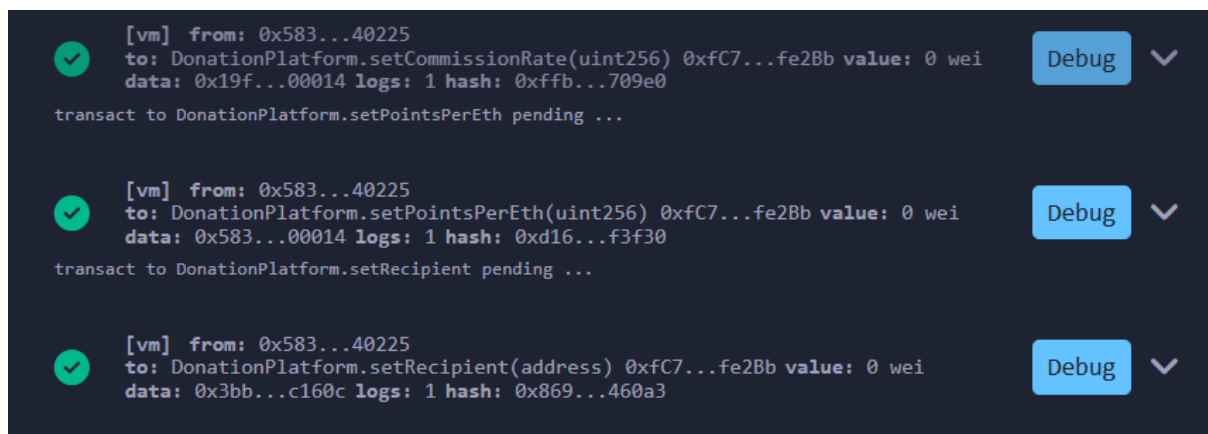
Ahora mostraremos el uso de *sets*. En primer lugar usando *setPointsPerEth* con el creador del contrato introduciendo como valor 20, obtenemos el siguiente decoded input: {

```
"uint256 _pointsPerEth": "20"
}
```



setCommission...	20	▼
setPointsPerEth	20	▼
setRecipient	0x14723A09ACff6D2A60DcdF7aA4AF308FDDC160C	▼

También desde el creador del contrato, queremos hacer una donación mayor y a otra cuenta en la operación podremos, dicha donación se realizará a la siguiente cuenta *0x14723A09ACff6D2A60DcdF7aA4AF308FDDC160C* y con un % del 20. En la captura se aprecia cómo se modifican los valores correctamente.

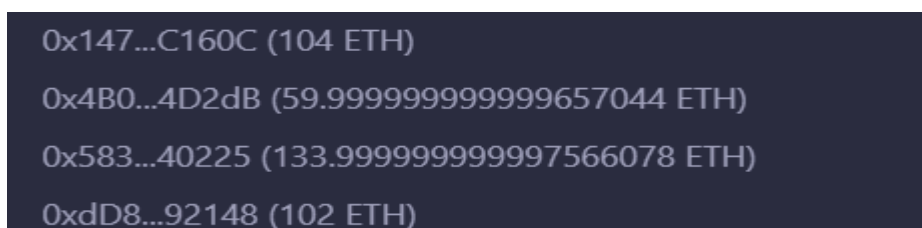


```
[vm] from: 0x583...40225
to: DonationPlatform.setCommissionRate(uint256) 0xfc7...fe2Bb value: 0 wei
data: 0x19f...00014 logs: 1 hash: 0xffb...709e0
Debug ▼
transact to DonationPlatform.setPointsPerEth pending ...

[vm] from: 0x583...40225
to: DonationPlatform.setPointsPerEth(uint256) 0xfc7...fe2Bb value: 0 wei
data: 0x583...00014 logs: 1 hash: 0xd16...f3f30
Debug ▼
transact to DonationPlatform.setRecipient pending ...

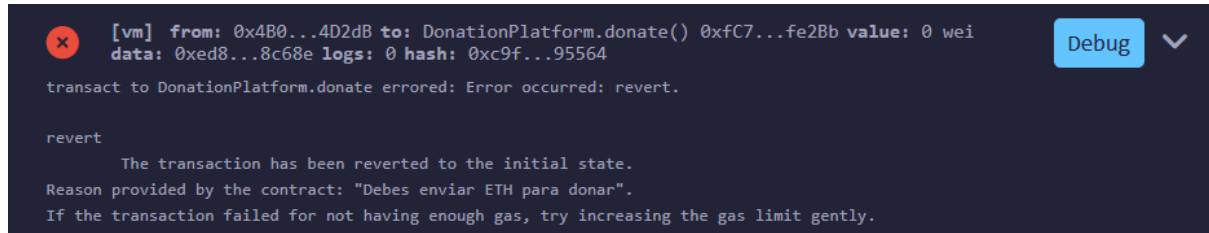
[vm] from: 0x583...40225
to: DonationPlatform.setRecipient(address) 0xfc7...fe2Bb value: 0 wei
data: 0x3bb...c160c logs: 1 hash: 0x869...460a3
Debug ▼
```

Después de modificar dichos valores, vamos a realizar otra compra de tokens desde nuestra dirección compradora, también de 20ETH para apreciar las diferencias en las cuentas que reciben las donaciones. Como podemos ver en la imagen se ha realizado todo de forma correcta, las diferentes direcciones tienen los valores de ETH que deberían de tener después de las transacciones realizadas



```
0x147...C160C (104 ETH)
0x4B0...4D2dB (59.99999999999999657044 ETH)
0x583...40225 (133.999999999999997566078 ETH)
0xD8...92148 (102 ETH)
```

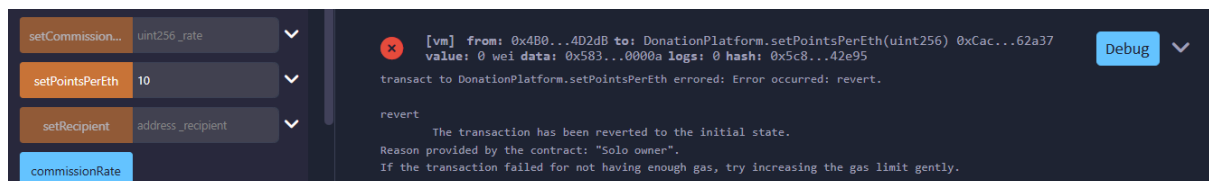
Ahora vamos a entrar en los posibles mensajes de error. Al realizar una compra de 0 ETH recibimos el mensaje “Debes enviar ETH para donar”.



```
[vm] from: 0x4B0...4D2dB to: DonationPlatform.donate() 0xFC7...fe2Bb value: 0 wei
data: 0xed8...8c68e logs: 0 hash: 0xc9f...95564
transact to DonationPlatform.donate errored: Error occurred: revert.

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Debes enviar ETH para donar".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```

Intentando realizar cualquier Set con alguien que no sea el owner nos salta el siguiente error.



```
[vm] from: 0x4B0...4D2dB to: DonationPlatform.setPointsPerEth(uint256) 0xCac...62a37
value: 0 wei data: 0x583...0000a logs: 0 hash: 0x5c8...42e95
transact to DonationPlatform.setPointsPerEth errored: Error occurred: revert.

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Solo owner".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
```

Por último con la ayuda de la IA generativa hemos generado un archivo que realiza pruebas unitarias llamado *donate\_test.sol* el cual lo incluimos en la carpeta del ejercicio 5.