

## Documentación Ejercicio 6 (Práctica 1)

Axel Valladares Pazó, Pedro Blanco Casal

### 1 Análisis y definición del escenario

El contrato inteligente "MiBazar" permite la compra y alquiler de skins para un videojuego, mientras que "MiToken" implementa un token ERC20 para realizar los pagos. El uso de blockchain garantiza la transparencia y seguridad en las transacciones, evitando manipulaciones y manteniendo un registro inmutable de las operaciones. La funcionalidad ofrecida por Ethereum es crucial para asegurar la propiedad y el historial de transacciones de los usuarios. De esta forma, ningún usuario podrá apropiarse de una skin que no le pertenece ni podrá alquilarla indefinidamente.

### 2 Diseño

#### 2.1 Caso de uso

El sistema facilita la compraventa y alquiler de skins, automatizando las transacciones y tratando a estas skins como tokens no fungibles, NFTs. Los usuarios deben registrarse en la plataforma y pueden usar el token ERC20 para comprar o alquilar skins. Cada skin tiene atributos como nombre, precio de compra, precio de alquiler, usuario actual, en caso de estar vendida o alquilada, y el propio estado. Finalmente es que se puedan usar estas skins o tokens de forma indefinida o no, en caso de ser alquiladas, en un videojuego en el que haya elementos personalizables.

#### 2.2 Contenido

##### 2.2.1 MiToken.sol:

Implementa un token ERC20 llamado "MiToken" con el símbolo "MTK". Se genera un suministro inicial de un millón de tokens asignados al creador del contrato.

##### 2.2.2 MiBazar.sol:

Define una estructura *Skin* con propiedades para el nombre, ID, precio de compra, precio de alquiler, hora de compra, duración del alquiler e ID del usuario propietario en caso de que lo tenga.

Mappings:

- *almacenSkins*: Relaciona el ID de una skin con su información.
- *usuarios*: Almacena la dirección de un usuario registrado con su identificador.

Funciones principales:

- *registrarUsuario*: Permite el registro de un usuario.
- *\_crearSkin*: Crea una nueva skin en la plataforma.
- *\_comprarSkin*: Permite comprar una skin, verificando que esté disponible y el usuario tenga el saldo necesario.
- *alquilarSkin*: Realiza el alquiler de una skin por un número determinado de días. Siempre y cuando que esta no esté vendida o ya alquilada.
- *\_datosSkin* y *\_devuelveEstado*: Permiten obtener datos sobre las skins y su estado.

Además de ciertas normas, como por ejemplo:

- El propietario del contrato, del Bazar, es el único que puede crear nuevas skins en este caso.
- Los usuarios deben registrarse antes de realizar compras o alquileres.
- Para poder comprarse una skin, el estado de esta debe ser "Disponible" y para ello, la hora de compra debe ser igual a 0 y el tiempo de alquiler debe ser 0 o menor al timestamp del momento en el que se ejecuta la función de compra.
- Antes de realizarse ningún gasto se comprueba que el usuario tenga suficientes fondos.

## 2.3 Usuarios del sistema

- Administrador: Responsable de crear skins y gestionar el contrato.
- Usuarios registrados: Pueden comprar o alquilar skins, utilizando el token MiToken.

## 3 Implementación

Se ha desarrollado todo el código introducido en este mismo repositorio en RemixIDE y teniendo en cuenta todo lo aprendido durante el desarrollo de la práctica 1.

## 4 Pruebas

A la hora de probar el código debemos compilar y ejecutar el contrato "miToken". Para así poder introducir tokens en el usuario y que pueda gustarlos en skins.

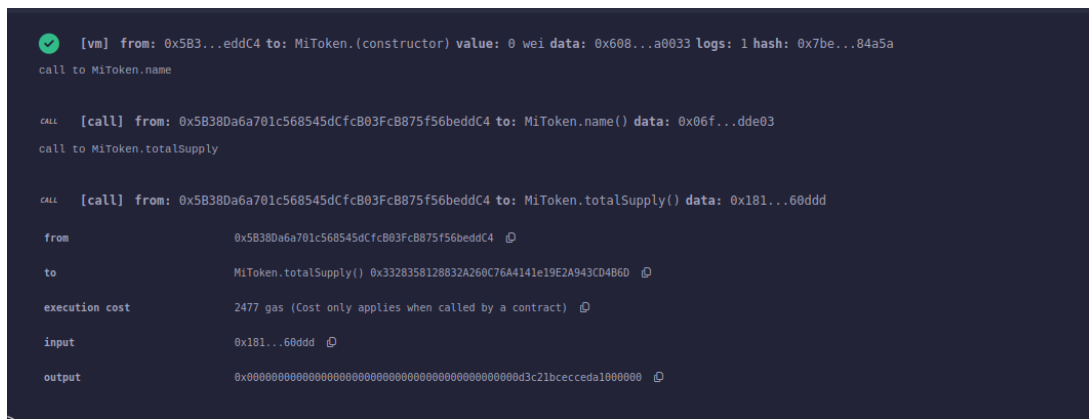


Figure 1: Resultado de compilar y ejecutar el contrato miToken.

Tras ello, hacemos lo mismo con el contrato "miBazar". Y ya tendríamos todo disponible, compilado y ejecutable para poder realizar las pruebas.

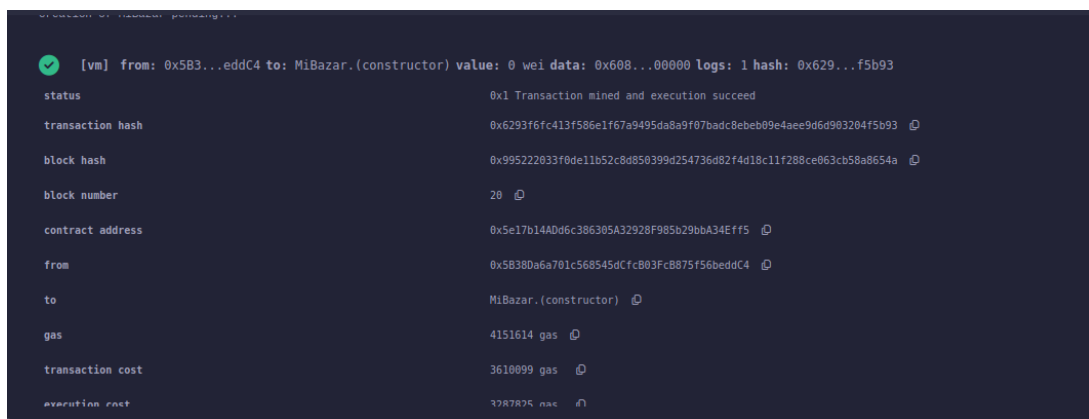


Figure 2: Resultado de compilar y ejecutar el contrato miBazar.

**Creación de usuario:** Ejecutamos registrar usuario con un string como nombre, en este caso, *Pedro*.

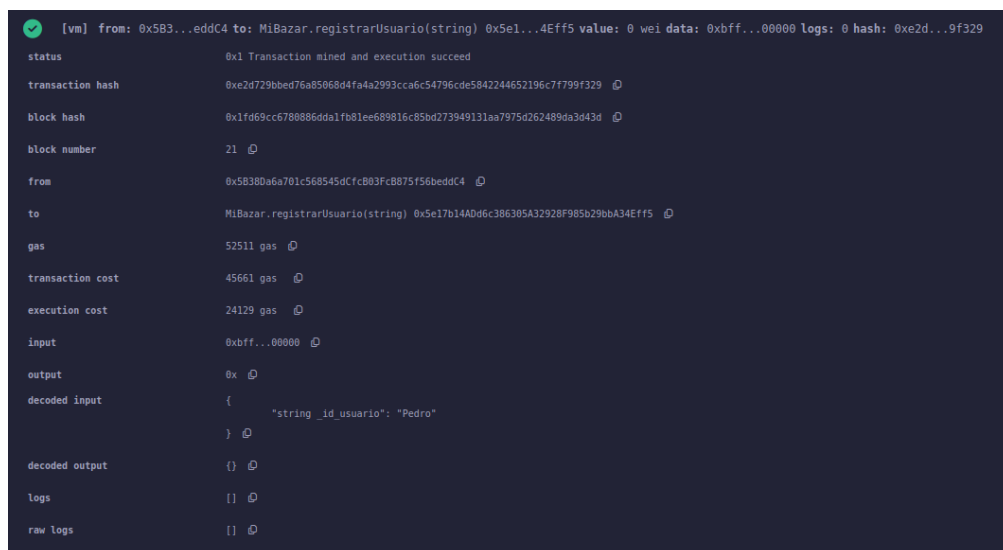
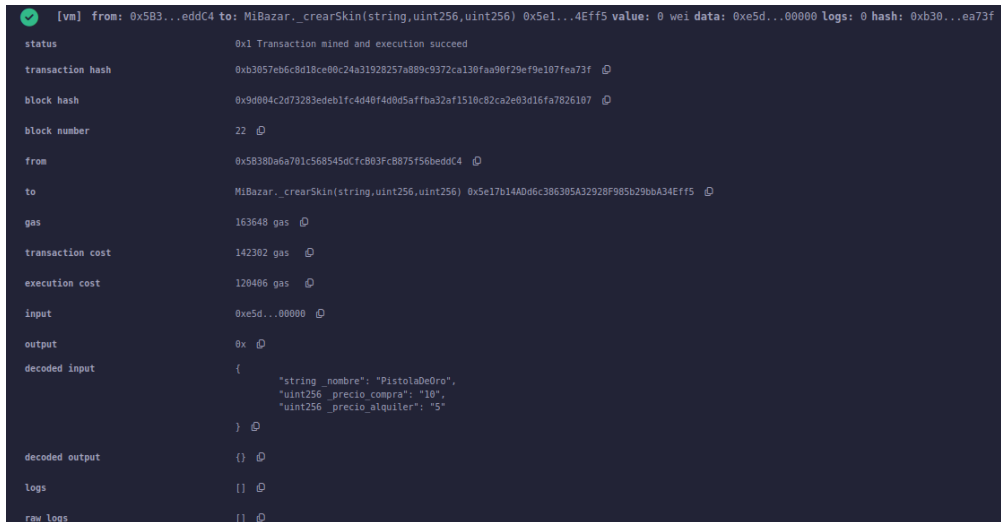


Figure 3: Resultado de registrar un usuario.

**Creación de skin:** Ejecutamos `_crearSkin` con un string como nombre, y un precio para la compra y otro para alquiler.



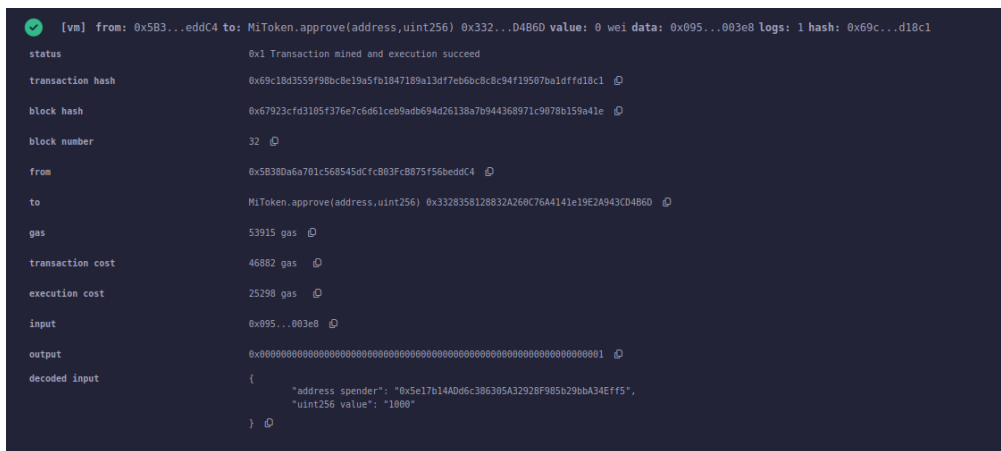
```

[vm] from: 0x583...eddC4 to: MiBazar._crearSkin(string,uint256,uint256) 0x5e1...4Eff5 value: 0 wei data: 0xe5d...00000 logs: 0 hash: 0xb30...ea73f
status                                0x1 Transaction mined and execution succeed
transaction hash                       0xb3057eb6c8d18ce08c24a31928257a809c9372ca130faa90f29ef9e107fea73f
block hash                            0x9d004c2d73283deb1fc4d40f4d0d5affba32af1518c82ca2e03d16fa7026107
block number                          22
from                                  0x58380a6a701c568545dcfcb03fcb875f56beddC4
to                                    MiBazar._crearSkin(string,uint256,uint256) 0x5e17b14ADd6c386305A32928F985b29bbA34Eff5
gas                                    163648 gas
transaction cost                       142302 gas
execution cost                         120406 gas
input                                  0xe5d...00000
output                                 0x
decoded input                          {
    "string_nombre": "PistolaDeOro",
    "uint256_precio_compra": "10",
    "uint256_precio_alquiler": "5"
}
decoded output                         {}
logs                                   []
raw logs                              []

```

Figure 4: Resultado de crear una skin.

**Permitir al usuario gastar tokens** Ejecutamos la función `approve` de "miToken" introduciendo como parámetros el address del usuario que nos devuelve al ejecutar su creación y la cantidad de tokens que le permitiremos usar.



```

[vm] from: 0x583...eddC4 to: MiToken.approve(address,uint256) 0x332...D4B6D value: 0 wei data: 0x095...003e8 logs: 1 hash: 0x69c...d18c1
status                                0x1 Transaction mined and execution succeed
transaction hash                       0x69c18d3559f98bc8e19a5fb1847189a13d7eb6bc8c8c94f19507ba1dffd18c1
block hash                            0x67923cfd3105f376e7c6d61ceb9adb694d26138a7b944368971c9078b159a41e
block number                          32
from                                  0x58380a6a701c568545dcfcb03fcb875f56beddC4
to                                    MiToken.approve(address,uint256) 0x3328358128832A260C76A4141e19E2A943C04B6D
gas                                    53915 gas
transaction cost                       46882 gas
execution cost                         25290 gas
input                                  0x095...003e8
output                                 0x0000000000000000000000000000000000000000000000000000000000000001
decoded input                          {
    "address spender": "0x5e17b14ADd6c386305A32928F985b29bbA34Eff5",
    "uint256 value": "1000"
}

```

Figure 5: Resultado de aprobar tokens.

**Comprar o alquilar una skin** Ejecutamos la función `_comprarSkin` introduciendo como parámetros el ID de la skin, por ejemplo "0", en la skin creada anteriormente y el nombre del usuario, al igual que con alquiler, donde se añade también el número de días.

```
[vm] from: 0x583...eddC4 to: MiBazar._comprarSkin(uint256,string) 0x5e1...4Eff5 value: 0 wei data: 0x54c...00000 logs: 1 hash: 0xcae...f4696

status
  0x1 Transaction mined and execution succeed

transaction hash
  0xcae4f3da9d887910273b992268c6bca8d78e2bcb0d9c6106d0d414098ff4696

block hash
  0xf4cdc32898725b5beb11457ef8f9eb97c922eca18e3a34779b9acala8ba4e54ac

block number
  40

from
  0x58380a6a701c568545dcfc803fc8875f56beddC4

to
  MiBazar._comprarSkin(uint256,string) 0x5e17b14ADd6c386305A32928F985b29bbA34eff5

gas
  117225 gas

transaction cost
  96334 gas

execution cost
  77462 gas

input
  0x54c...00000

output
  0x

decoded input
  {
    "uint256_id": "1",
    "string_id_usuario": "Pedro"
  }

decoded output
  {}

logs
  [
    {
      "from": "0x3328358128832A260C764141c19E24943C04860",
      "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
      "event": "Transfer",
      "args": {
        "0": "0x58380a6a701c568545dcfc803fc8875f56beddC4",
        "1": "0x58380a6a701c568545dcfc803fc8875f56beddC4",
        "2": "25",
        "from": "0x58380a6a701c568545dcfc803fc8875f56beddC4",
        "to": "0x58380a6a701c568545dcfc803fc8875f56beddC4",
        "value": "25"
      }
    }
  ]
```

Figure 6: Resultado de comprar skins.

```
[vm] from: 0x583...eddC4 to: MiBazar.alquilarSkin(uint256,string,uint256) 0x5e1...4Eff5 value: 0 wei data: 0x000...00000 logs: 1 hash: 0x5ec...8f59c

status
  0x1 Transaction mined and execution succeed

transaction hash
  0x5ecd4d7ecd706d4fb4b649722a0e5a6f74ee7095fa3147071c812c4638f59c

block hash
  0xec72cdab35ee5c62ea885fe8806e6f130654ee758f8a0f1d247cae9923ff5a28

block number
  37

from
  0x58380a6a701c568545dcfc803fc8875f56beddC4

to
  MiBazar.alquilarSkin(uint256,string,uint256) 0x5e17b14ADd6c386305A32928F985b29bbA34eff5

gas
  141233 gas

transaction cost
  117211 gas

execution cost
  90223 gas

input
  0x000...00000

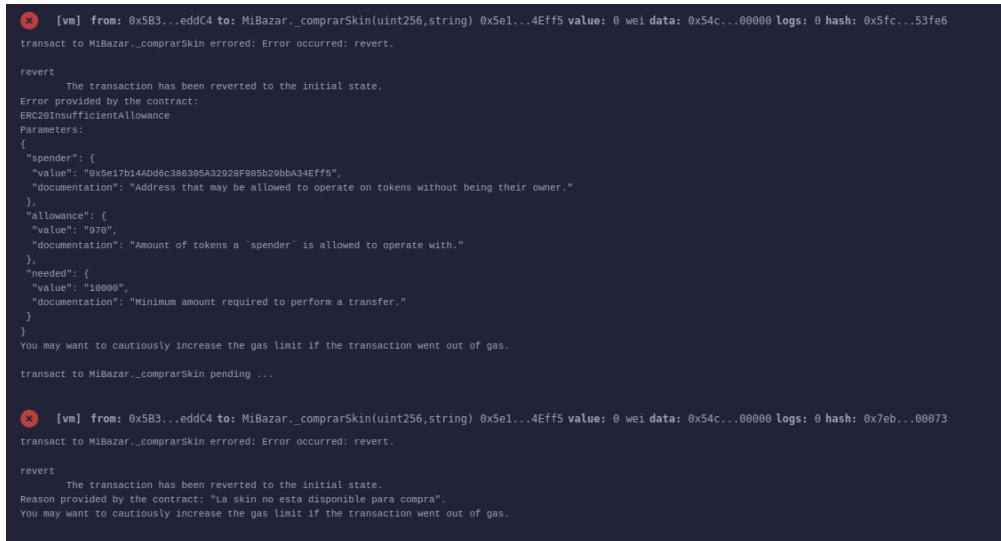
output
  0x

decoded input
  {
    "uint256_id": "0",
    "string_id_usuario": "Pedro",
    "uint256_dias": "1"
  }

... ..
```

Figure 7: Resultado de alquilar skins.

**Prueba de fallo** Ejecutamos la función `_comprarSkin` Sabiendo que la skin elegida tiene un costo demasiado alto o ya ha sido vendida. Como podemos ver en la siguiente figura:



```
[vm] from: 0x583...eddC4 to: MiBazar._comprarSkin(uint256,string) 0x5e1...4Eff5 value: 0 wei data: 0x54c...00000 logs: 0 hash: 0x5fc...53fe6
transact to MiBazar._comprarSkin errored: Error occurred: revert.

revert
  The transaction has been reverted to the initial state.
Error provided by the contract:
ERC20InsufficientAllowance
Parameters:
{
  "spender": {
    "value": "0x5e17b14ADd6c386385A32928F985b20bbA344Eff5",
    "documentation": "Address that may be allowed to operate on tokens without being their owner."
  },
  "allowance": {
    "value": "970",
    "documentation": "Amount of tokens a 'spender' is allowed to operate with."
  },
  "needed": {
    "value": "10000",
    "documentation": "Minimum amount required to perform a transfer."
  }
}
You may want to cautiously increase the gas limit if the transaction went out of gas.

transact to MiBazar._comprarSkin pending ...

[vm] from: 0x583...eddC4 to: MiBazar._comprarSkin(uint256,string) 0x5e1...4Eff5 value: 0 wei data: 0x54c...00000 logs: 0 hash: 0x7eb...00073
transact to MiBazar._comprarSkin errored: Error occurred: revert.

revert
  The transaction has been reverted to the initial state.
Reason provided by the contract: "La skin no esta disponible para compra".
You may want to cautiously increase the gas limit if the transaction went out of gas.
```

Figure 8: Resultado de comprar una skin demasiado cara y de comprar una ya vendida.