

Máster Universitario en Tecnologías del Sector Financiero  
2019-2020

*Práctica 3*

“Aplicaciones sobre blockchain mediante contratos inteligentes”

D. Álvaro Andrés Suárez Alfonso

Maestro

Dr. José María de Fuentes García-Romero de Tejada

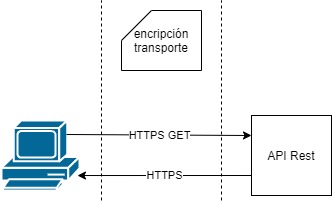
Puerta de Toledo, 2020

# Introducción

En este documento se plasma el desarrollo de la práctica 3 para la materia de Blockchain y Tecnologías de Seguridad, en la que con el uso de una aplicación Web se ejecutan diferentes funciones de un contrato inteligente para interactuar con una blockchain local en ambiente de desarrollo.

# Arquitectura de aplicación

La aplicación se basa en un API Rest desarrollada con NodeJS que esta securizada en la capa de transporte con el protocolo HTTPS como se muestra en el siguiente diagrama adhoc:



El API Rest consta de dos componentes: por un lado, el servidor que se encarga de exponer la aplicación en un servidor https y, por otro, la app que contiene los métodos que se consumen a través de una petición. La app consta de los siguientes métodos:

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Objetivo | Parámetros | Salida |
| getAddressess | Obtener las direcciones de la blockchain | N/A | Direcciones |
| getBalance | Obtener el balance de una dirección | from: Dirección a consultar | Balance disponible |
| depositFunds | Depositar balance en una dirección especifica | from: Dirección en la que se va a depositar wei: Ether que se van a depositar en unidad de medida wei | Confirmación de deposito |
| transferFunds | Transferir balance de una dirección a otra | from: Dirección ejecutante de la transferencia receiver: Dirección recibiente wei: Ether transferidos en unidad wei | Confirmación de transferencia |
| withdrawFunds | Extraer balance de una dirección | from: Dirección en a la que se va a extraer balance wei: Ether que se van a extraer en unidad de medida wei | Confirmación de extracción |

# Diseño contrato inteligente

El proyecto truffle consta de un contrato inteligente que simula un cajero automático para gestionar los fondos de las cuentas de forma básica y, como vamos a ver ahora, algunos métodos expuestos en el API tienen su función de contrato asociada:

|  |  |  |  |
| --- | --- | --- | --- |
| Función | Objetivo | Parámetros | Salida |
| deposit | Depositar balance en una dirección especifica | N/A | N/A |
| transfer | Transferir balance de una dirección a otra | receiver: Dirección recibiente amount: Ether transferidos en unidad wei | N/A |
| withdraw | Extraer balance de una dirección | wei: Ether que se van a extraer en unidad de medida wei | N/A |

Así mismo el contrato emite un evento asociado a cada función, de la siguiente forma:

|  |  |
| --- | --- |
| Evento | Parametros |
| Deposit | sender: Direccion remitente amount: Ether transferidos en unidad wei |
| Transfer | sender: Direccion remitente amount: Ether transferidos en unidad wei |
| Withdrawal | receiver: Direccion extractora de fondos wei: Ether que se van a extraer en unidad de medida wei |

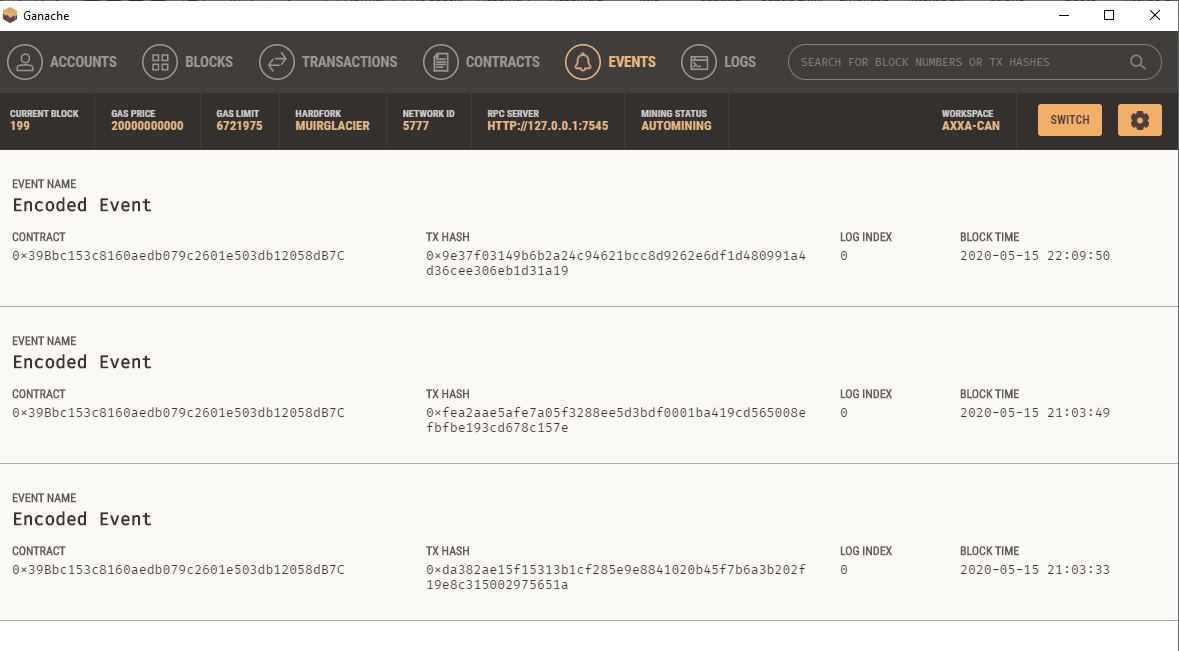
# Testing

Se hicieron distintas pruebas consumiendo los métodos del contrato para verificar su funcionalidad:

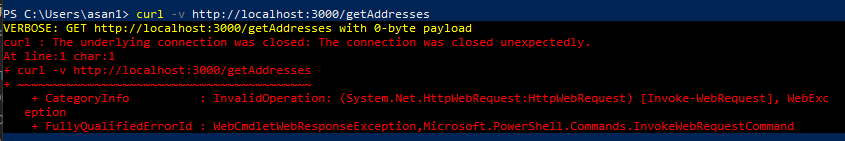
Se verificaron transacciones en 0 para el contrato:



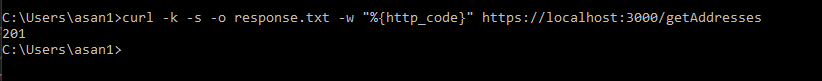
Se verificaron 0 eventos para el contrato ATM:



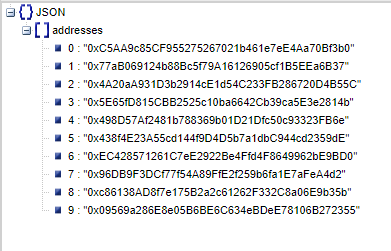
Verificación de denegación para canal http:



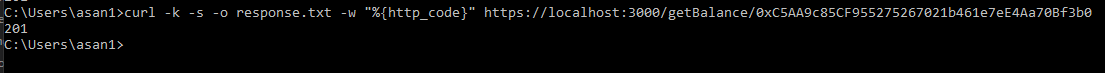
Se obtienen las direcciones de la red:



Y en el fichero response.txt se encuentran las siguientes direcciones:



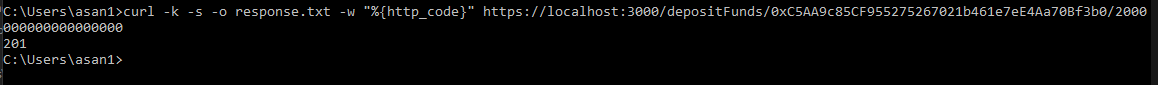
Ahora obtenemos el balance de la primera dirección:



{"balance":"20000000000000000"}

Que corresponden a 0.02 ether.

Luego, hacemos un depósito de 2 eth:



Y verificamos el balance nuevamente:



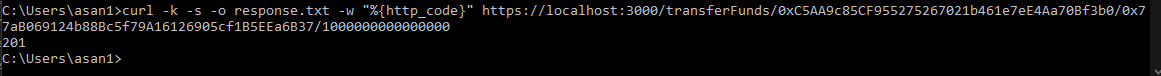
{"balance":"2020000000000000000"}

Ahora verificamos el balance de la segunda cuenta:



{"balance":"0"}

Y hacemos una transferencia a esta cuenta por 0.001 eth desde nuestra primera cuenta:



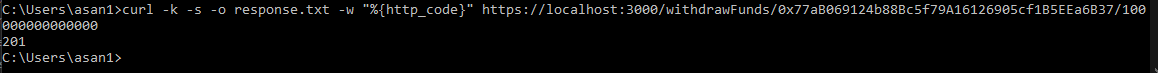
Verificamos el balance de la primera cuenta:

{"balance":"2019000000000000000"}

Verificamos el balance de la segunda cuenta:

{"balance":"1000000000000000"}

Ahora vamos a extraer 0.0001 eth de la segunda cuenta:



Y verificamos su balance:

{"balance":"900000000000000"}

Ahora vamos a intentar extraer 0.01 eth de la segunda cuenta:

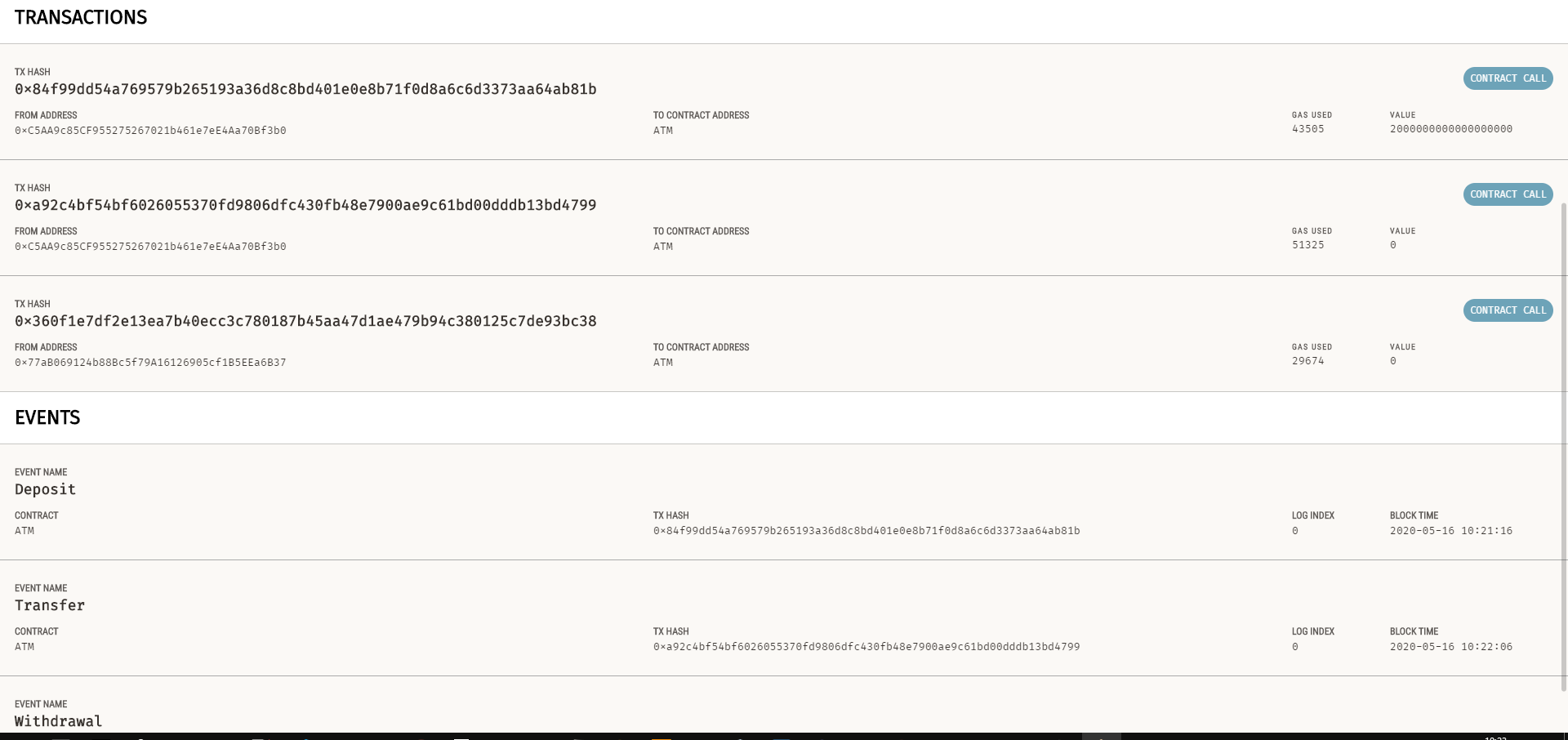
Y nos retorna:

Error: Returned error: VM Exception while processing transaction: revert Insufficient funds

Volvemos a verificar el balance:

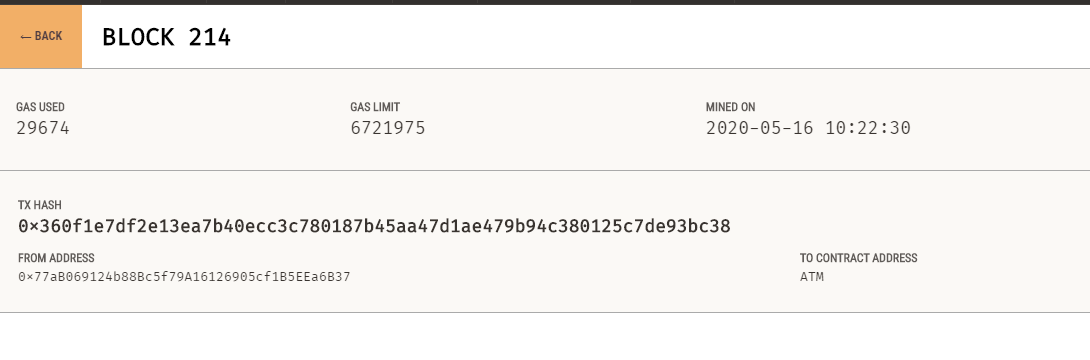
{"balance":"900000000000000"}

En la interfaz del contrato se puede ver el log de transacciones y eventos:

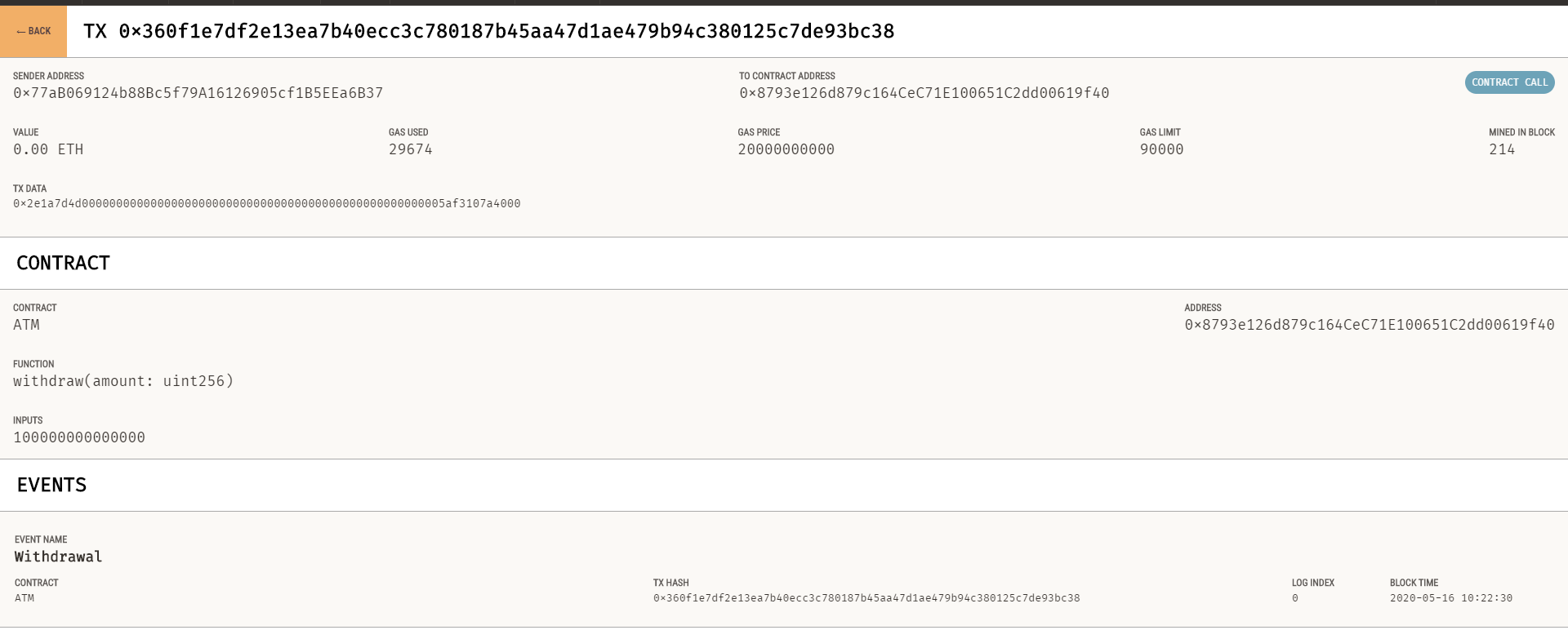




Igualmente se visualizan los bloques de estas transacciones, por ejemplo:



La cual corresponde a un retiro (función withdraw del contrato):



# Instrucciones de despliegue de aplicación

Prerrequisitos:

* Tener como mínimo nodeJS v12.14 instalado, el cual se puede descargar de <https://nodejs.org/es/>
* Establecer la URL donde se encuentra la blockchain en el fichero app.js:

const url = 'HTTP://127.0.0.1:7545';

Descomprimir el fichero practica3 y dentro de la carpeta node\_project ejecutar el siguiente comando:

npm install

Este comando instalará las dependencias necesarias para correr la aplicación, las cuales están referenciadas en package.json

Posterior a que acabe la instalación, correr el siguiente comando:

npm run start:server

Este comando correrá el fichero server.js el cual inicializa el API y la mantiene arriba para que pueda ser consumida desde el puerto 3000.

Ejemplos de consumo del API:

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/getAddresses

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/getBalance/0xC5AA9c85CF955275267021b461e7eE4Aa70Bf3b0

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/depositFunds/0xC5AA9c85CF955275267021b461e7eE4Aa70Bf3b0/2000000000000000000

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/getBalance/0x77aB069124b88Bc5f79A16126905cf1B5EEa6B37

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/withdrawFunds/0x77aB069124b88Bc5f79A16126905cf1B5EEa6B37/10000000000000000

# Observaciones y dificultades

Se presentaron dificultades en el uso del contrato y la verificación del balance en Ganache, y se encuentra en las funciones transfer y deposit del contrato ATM las cuales se pueden verificar a la hora de pedir el balance desde el contrato, pero no se puede verificar desde la interfaz grafica de ganache. A continuación, la implementación del contrato:

pragma solidity 0.5.16;

contract ATM {

    mapping(address => uint) public balances;

    event Deposit(address sender, uint amount);

    event Withdrawal(address receiver, uint amount);

    event Transfer(address sender, address receiver, uint amount);

    function deposit() public payable {

        emit Deposit(msg.sender, msg.value);

        balances[msg.sender] += msg.value;

    }

    function withdraw(uint amount) public {

        require(balances[msg.sender] >= amount, "Insufficient funds");

        emit Withdrawal(msg.sender, amount);

        balances[msg.sender] -= amount;

    }

    function transfer(address receiver, uint amount) public {

        require(balances[msg.sender] >= amount, "Insufficient funds");

        emit Transfer(msg.sender, receiver, amount);

        balances[msg.sender] -= amount;

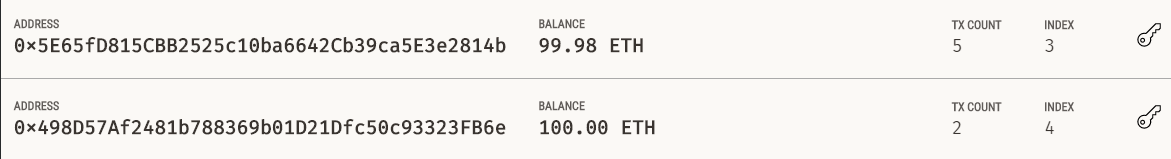
        balances[receiver] += amount;

    }

}

Como se puede ver hay una variable publica que apunta al storage de balances por address.

Inicialmente tenemos las dos siguientes direcciones con sus balances:



Inicialmente produzco un withdraw de la segunda cuenta para verificar que se vea reflejado en Ganache:

Extraigo 0.1 eth:

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/withdrawFunds/0x498D57Af2481b788369b01D21Dfc50c93323FB6e/100000000000000000

y en logs sale error:

 [10:02:29 PM] Transaction: 0x20a6cdfe956d4fed7bd849823b91038208087a33d6f52daf26e7c2ca046b4d8f

 [10:02:29 PM] Gas usage: 22518

 [10:02:29 PM] Block Number: 220

 [10:02:29 PM] Block Time: Tue May 19 2020 22:02:29 GMT+0200 (hora de verano de Europa central)

 [10:02:29 PM] Runtime Error: revert

 [10:02:29 PM] Revert reason: Insufficient funds

Al parecer el balance solo se puede hacer un withdraw si la dirección cuenta con balance dentro del contrato, ya que al hacer la verificación del balance para esta dirección usando el contrato obtengo lo siguiente:

{"balance":"0"}

De igual forma sucede con la función transfer, ya que si la dirección no tiene fondos en el contrato no puede hacer la transferencia a menos que le ingrese fondos con la función de deposit.

He intentado cambiar el contrato de forma que las funciones withdraw y transfer implementen de payable. Igualmente desde la api de Nodejs en el cual uso la abstracción de web3js para instanciar el smart contract, he llamado los métodos del contrato de dos formas que veo son las únicas posibles; method.NombreMetodo.call y method.NombreMetodo.send, obteniendo el mismo resultado. También he borrado la carpeta build de truffle y vuelto a compilar y desplegar el contrato.

Verifico también las transacciones en ganache, los eventos, el contador de bloques y puedo llegar a la conclusión que quedan registros de ellos solo si la ejecución de la función no envía error, es decir, si hago el depósito desde:



curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/depositFunds/0x498D57Af2481b788369b01D21Dfc50c93323FB6e/100000000000000000



La transacción va en +1, se puede ver que el deposito se hace en el contrato al verificar el balance para esa dirección:

{"balance":"100000000000000000"}

Y ahora si puedo hacer la transferencia sin error:

curl -k -s -o response.txt -w "%{http\_code}" https://localhost:3000/transferFunds/0x498D57Af2481b788369b01D21Dfc50c93323FB6e/0x77aB069124b88Bc5f79A16126905cf1B5EEa6B37/100000000000000000

 [10:34:39 PM] eth\_sendTransaction

 [10:34:39 PM] Transaction: 0x4d42304bc6cf95263bdda06c3e1849d40115926ea05084eb0c457533343d7c2d

 [10:34:39 PM] Gas usage: 21325

 [10:34:39 PM] Block Number: 222

 [10:34:39 PM] Block Time: Tue May 19 2020 22:34:39 GMT+0200 (hora de verano de Europa central)

 [10:34:39 PM] eth\_getBlockByNumber