Hola Sergio:

1. Ante todo, quiero comentarte que la programación no es mi fuerte. No lo consideres cómo una excusa. El tema me apasiona, pero es una “pata” que me falta y estoy en ello. Soy de la generación del Basic, o sea que con eso te lo digo todo; pero estudio en paralelo a este máster los fundamentos de programación en JavaScript, Node.js y Solidity.

Bueno…ya me dirás si he cometido muchas burradas. Un saludo.

1. Como tengo instalado truffle en ambiental, lo he usado junto con ganache para compilar y “jugar” con la propia consola de truffle.
2. El fichero zip que te adjunto ya tiene instalado todas las carpetas y ficheros necesarios. Abrimos ganache y ejecutamos :

**truffle deploy - -network development**

Nota-1.- En el fichero zip también está el contrato que se despliega por defecto al inicializar truffle.

Nota2.- En las instrucciones en el que hay dos guiones seguidos, no basta con un copiar y pegar. Los dos guiones seguidos en Word, se solapan.

para que se despliegue el contrato en la consola y en la red Ethereum de ganache. Como ya sabes, truffle ya incorpora web3.

1. Modificamos el contenido del fichero **truffle-config.js** (este aparece al ejecutar **truffle init**) donde he definido una red de desarrollo **development** para trabajar con el servidor de ganache (le he incorporado un **host**, el puerto de ganache, una **id** y un **gas).**
2. Creamos un fichero de migraciones en donde le indicamos que queremos usar el artefacto de nuestro smart contract.
3. Aunque seguro que sabrás como hacerlo, para facilitarte jugar con el smart contract, indico algunas instrucciones. Basta con un copiar y pegar.
4. Abrimos la carpeta con VSC (si quieres, lo puedes hacer desde la consola).
5. Abrimos ganache. Yo uso mi propio **mnemonic** pero, por defecto, te autogenera las cuentas.
6. En la terminal de VSC (o consola), dentro de la carpeta:

**truffle deploy - - network development**

1. En el caso de que por cualquier circunstancia cerráramos ganache, para volver a desplegar el contrato, abrimos ganache y ejecutamos:

**truffle deploy - - reset network development**

6. Abrimos la consola de truffle

**truffle console network development-> truffle(development)>**

**En la consola:**

* 1. Obtención de la dirección:

**FabricaProfesoresZombies.address**

**web3.eth.getAccounts(function(err, data) {accounts = data});**

El sistema nos devolverá **undefined**

Ejecutamos **accounts,** se nos desplegarán en la consola, las 10 cuentas de Ganache.

* 1. Para obtener el balance de una cuenta en concreto:

**web3.eth.getBalance (accounts[0]);**

* 1. Información del balance de una cuenta concreta de forma legible (no BigNumber):

**web3.eth.getBalance (accounts[0]).toNumber();**

* 1. Conversión de divisa para pasar de wei a ether de la cuenta:

**web3.fromWei(web3.eth.getBalance(accounts[0]), 'ether');**

* 1. Conversión de divisa para pasar de wei a ether de la cuenta de forma legible:

**web3.fromWei(web3.eth.getBalance(accounts[0]).toNumber(), 'ether');**

* 1. Ejecutar una transacción enviando divisas desde una cuenta a otra

**web3.eth.sendTransaction({ from: accounts[0], to: accounts[1], value: web3.toWei (15, 'ether') });**

* 1. Para conocer los detalles de la transacción

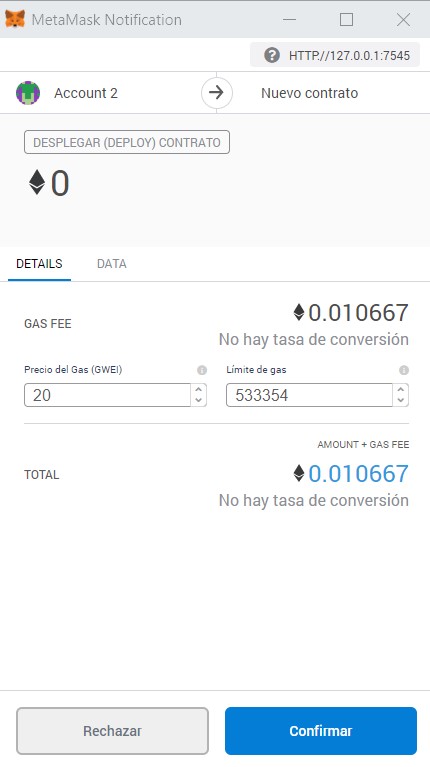
**web3.eth.getTransaction('id de la transacción anterior)'**

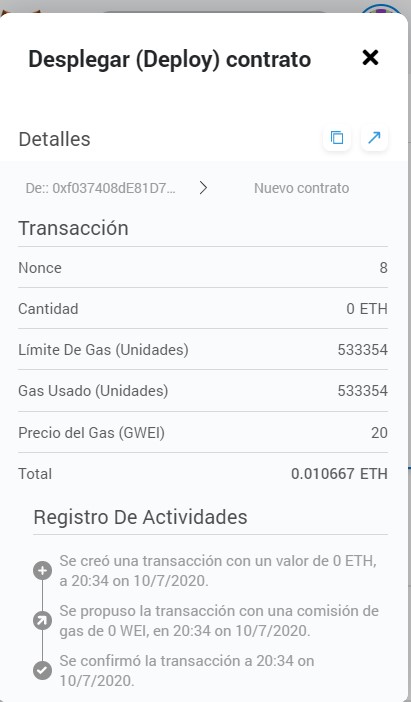
* 1. Obtención de más detalles y observación del status: ‘0X1’ => correcto)

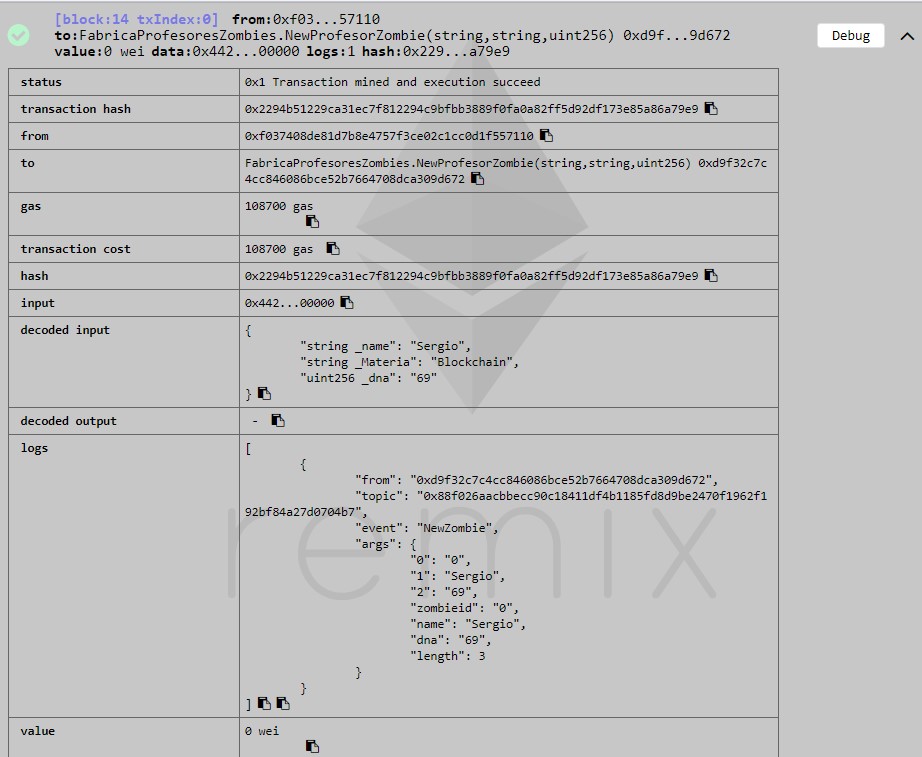
**web3.eth.getTransactionReceipt(('id de la transacción anterior)'**

1. **Desplegando el contrato. Lo he hecho de 2 maneras en el Enviroment**
   1. Con **JavaScript VM.**
   2. **Injected Web** de la siguiente manera:

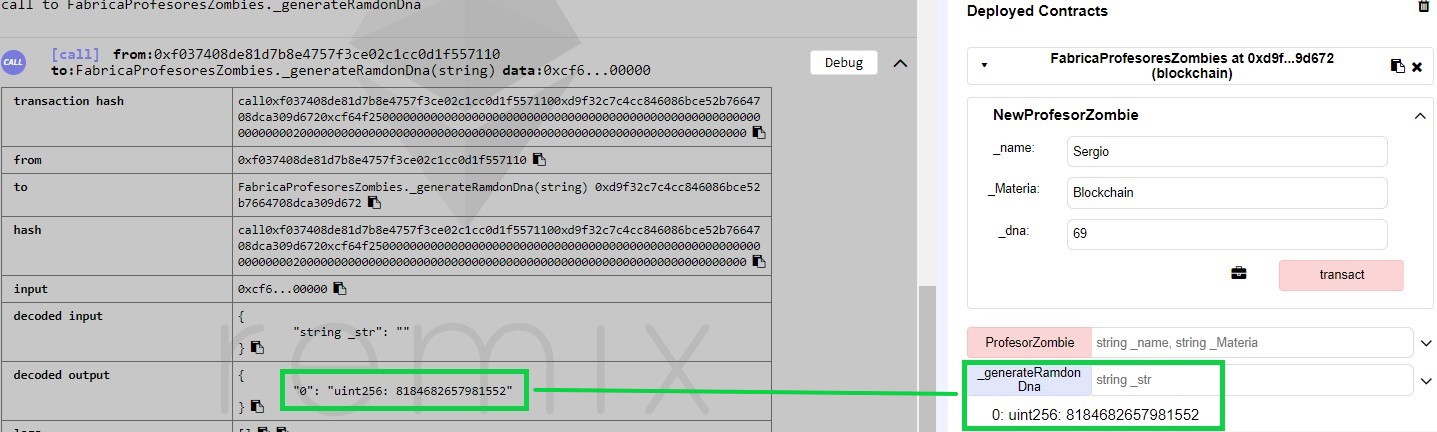
* Desde Metamask importo de Ganache una cuenta con 100 ether.
* Ejecuto Deploy y confirmo en Metamask

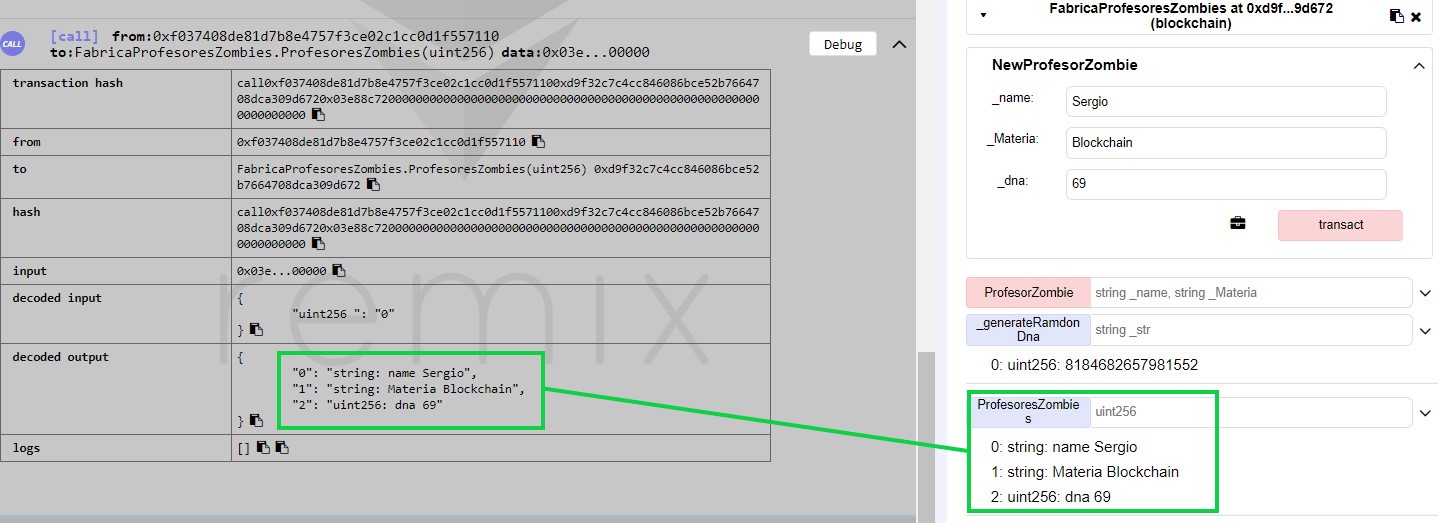






**Random DNA**



**ProfesoresZombies**

Sigo con Loom. Un saludo