



# COMPTE RENDU DE TP :

## Développer, déployer et interagir avec un contrat intelligent sur Ethereum

Ce rapport comporte l'intégralité des recherches, manipulations et résultats obtenus à la cour du TP du 07/09/2019. Il aborde les différents points clef abordés au cours de ce dernier, étant :

- La prise en main des outils « Remix » et « Metamask ».
- Le développement d'un « Smart Contract ».
- Le déploiement d'un « Smart Contract ».
- Les interactions avec un « Smart Contract ».

# 1. Prise en main de l'outil Metamask

Dans un premier temps, nous devons générer un « Wallet » dans le but d'obtenir une paire de clef permettant de communiquer avec un réseau « Blockchain » publique basé sur « Ethereum » (Ethereum Sopssten). Pour ce faire, nous utiliserons l'outil « Metamask » dans un navigateur Firefox afin de pouvoir utiliser et traiter gratuitement des transactions.

Nous générons ainsi la clef publique ci-contre, que nous utiliserons pour l'ensemble des manipulations et transactions du TP :

*0xc447d09EE09ee8973d2e3D393495D42Fc771765d*

Afin de pouvoir effectuer des transferts et payer le coût des transactions que nous génèrerons, nous allons simuler l'envoi de tokens Ethereum sur notre clef privé par le biais de l'outil.

Nous pouvons ainsi observer, grâce à l'outil « Etherscan » que nous utiliserons pour l'analyse des transactions, l'envoi de la ressource vers notre wallet :

The screenshot displays the 'Transaction Details' page on Etherscan for a Ropsten Testnet transaction. The transaction is confirmed as 'Success'. It shows a transfer of 1 Ether from the address 0x81b7e08f65bdf5648606c89998a9cc8164397647 to the address 0xc447d09ee09ee8973d2e3d393495d42fc771765d. The transaction fee is 0.0000315 Ether, and the gas limit is 21,000. The input data field is empty, showing '0x'.

Transaction Details	
[ This is a Ropsten Testnet transaction only ]	
Transaction Hash:	0xe2a890f7119d40f8f34af02f7c3ca9c87981b67466e484d6864227d8b27dbe7a
Status:	Success
<b>Overview</b> State	
Timestamp:	8 mins ago (Sep-07-2020 08:17:51 AM +UTC)
From:	0x81b7e08f65bdf5648606c89998a9cc8164397647
To:	0xc447d09ee09ee8973d2e3d393495d42fc771765d
Value:	1 Ether (\$0.00)
Transaction Fee:	0.0000315 Ether (\$0.000000)
Gas Limit:	21,000
Gas Used by Transaction:	21,000 (100%)
Gas Price:	0.000000015 Ether (1.5 Gwei)
Nonce	34452179
Input Data:	0x

Figure 1 : Résumé de la transaction pour la génération de tokens

Transaction Details

Overview **State**

Advanced A set of information that represents the current **state** is updated when a transaction takes place on the network. The below is a summary of those changes :

Address	Before	After	State Difference
0x81b7e08f65bdf56486...	87,798,236.384047221349978134 Eth Nonce: 34452179	87,798,235.384015721349978134 Eth Nonce: 34452180	▼ 1.0000315
0xc447d09ee09ee8973...	1 Eth	2 Eth	▲ 1
0xd34912efb0e7fedaed... <small>Miner</small>	30,676.366958329013395319 Eth	30,676.366989829013395319 Eth	▲ 0.0000315

Figure 2 : Etat des crédits après transaction

Nous pouvons aussi observer que nous avons bien la génération d'un bloc chaîné, contenant l'intégralité des données attendues (Id du block, has du block et hash du block précédent, le « Nonce » ...) :

Block #8636025

Overview

[ This is a Ropsten Testnet block only ]

Block Height:	8636025 < >
Timestamp:	13 mins ago (Sep-07-2020 08:17:51 AM +UTC)
Transactions:	13 transactions and 7 contract internal transactions in this block
Mined by:	0xd34912efb0e7fedaed9390990d7ef623e01f4fa in 6 secs
Block Reward:	2.024766875 Ether (2 + 0.024766875)
Uncles Reward:	0
Difficulty:	551,199,604
Total Difficulty:	31,436,510,377,856,311
Size:	8,910 bytes
Gas Used:	2,561,916 (32.12%)
Gas Limit:	7,976,582
Extra Data:	poolin.com (Hex: 0x706f6f6c69e2e636f6d)
Hash:	0xc74a0ce0c824d431e36b462e7bfdd92b4f0687a3eb50a3b8d868f982c3281a0c
Parent Hash:	0xeabd71f60534cacf0480b95eadb03c7f4a5d806662d516d26b2111863f9b8844
Sha3Uncles:	0x1dcc4de8dec75d7aab85b567b6cccd41ad312451b948a7413f0a142fd40d49347
StateRoot:	0x807f00d1220a43dd21233f4605c728621fa3473730fcea7bc37b1952652aa1b1
Nonce:	0x49eb1a30057546a6

Figure 3 : Détail du block généré pour la transaction

Une fois le compte crédité, nous allons pouvoir effectuer des transactions à travers la blockchain, vers d'autres entités. Afin de vérifier si nous arrivons bien à générer des transactions, nous allons effectuer un transfert d'un montant de 1 ETH vers la clef publique suivante :

*0xc25a95A1D4a59A0E56f188f9C966A3Dad518100F*

Pour cela, nous utilisons Metamask et renseignons l'ensemble des informations demandées pour l'envoi des tokens :

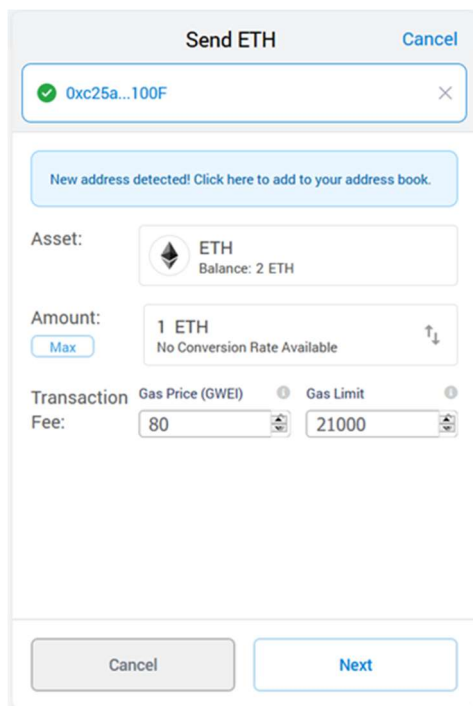


Figure 4 : Interface d'envoi d'ETH de l'outil Metamask

Nous pouvons en relever la transaction suivante, confirmant bien la présence dans la blockchain de l'envoi du montant préalablement indiqué :

The screenshot displays the 'Transaction Details' page for a transaction on the Ropsten Testnet. A red warning message at the top states: '[ This is a Ropsten Testnet transaction only ]'. The transaction is confirmed as 'Success' with a green checkmark. The transaction hash is 0x1fe9cce0fbb3f65d558d1c52a8bcd1fed9e18bf3dd91a1bb542e3d38c395f353. The transaction was sent from 0xc447d09ee09ee8973d2e3d393495d42fc771765d to 0xc25a95a1d4a59a0e56f188f9c966a3dad518100f. The value is 1 Ether (\$0.00) and the transaction fee is 0.00168 Ether (\$0.000000). The gas limit is 21,000, and the gas used is 21,000 (100%). The gas price is 0.00000008 Ether (80 Gwei). The nonce is 0. The input data is 0x. A link 'Click to see Less' is at the bottom left.

Field	Value
Transaction Hash	0x1fe9cce0fbb3f65d558d1c52a8bcd1fed9e18bf3dd91a1bb542e3d38c395f353
Status	Success
Timestamp	32 secs ago (Sep-07-2020 08:38:07 AM +UTC)
From	0xc447d09ee09ee8973d2e3d393495d42fc771765d
To	0xc25a95a1d4a59a0e56f188f9c966a3dad518100f
Value	1 Ether (\$0.00)
Transaction Fee	0.00168 Ether (\$0.000000)
Gas Limit	21,000
Gas Used by Transaction	21,000 (100%)
Gas Price	0.00000008 Ether (80 Gwei)
Nonce	0
Input Data	0x

Figure 5 : Détails de la transaction sur l'envoi des token ETH

## 2. Prise en main de l'outil Remix et déploiement d'un Smart Contract

Afin de générer un nouveau Smart Contract dans le réseau, nous nous baserons sur un déjà développé et disponible sur la plateforme Github par le biais de l'adresse suivante :

« [https://github.com/cozcan/TP\\_Election](https://github.com/cozcan/TP_Election) ». Ce dernier permet la déclaration de candidats ainsi qu'un système de vote pour ces derniers.

Afin de pouvoir modifier, compiler et soumettre les contrats au réseau, nous utiliserons l'outil Remix.

Une fois les fichiers développés sous l'IDE Solidity importés dans l'environnement, nous demandons à l'outil de compiler notre contrat afin de récupérer un fichier « ABI » (standard permettant l'interaction avec le contrat) et un fichier « Byte Code » (constructeur du contrat). Ces derniers sont disponibles avec ce document et seront utilisés pour la création du smart contract dans la blockchain.

Nous pouvons désormais demander le déploiement du contrat par le biais de l'outil. Ce dernier vas donc envoyer, faire valider et appliquer les fichiers générés afin de créer un nouveau smart contract aillant une adresse ip public propre. Nous pouvons relever la transaction de ce déploiement ci-dessous :

The screenshot displays the 'Transaction Details' for a smart contract deployment on the Ropsten Testnet. The transaction is successful and has been confirmed by 1 block.

Field	Value
Transaction Hash	0xb46d558ed51f7cb317195f2fc985f48fa5300328856b2b66dd67314e5ef37892
Status	Success
Block	8636333 (1 Block Confirmation)
Timestamp	32 secs ago (Sep-07-2020 08:58:53 AM +UTC)
To	[Contract 0xcaca723851462e2ba0469745e30c0fa492cbf5da Created]
Value	0 Ether (\$0.00)
Transaction Fee	0.000829809 Ether (\$0.000000)
Gas Limit	553,206
Gas Used by Transaction	553,206 (100%)
Gas Price	0.0000000015 Ether (1.5 Gwei)
Nonce	1
Input Data	0x608060405234801561001057600080fd5b50336000806101000a81548173ff021916908373ff1602179055506108ab806100606000396000f3fe608060405234801561001057600080fd5b506004361061007d5760003560e01c8063462e91ec1161005b578063462e91ec146101835780638da5cb5b1461023e578063a3ec138d14610272578063f2fde38b146102cc5761007d565b80630121b93ff146100825780632d35a8a2146100b05780633477ee2e146100ce575b600080fd5b6100ae6004803603602081101561009857600080fd5b8101908080359060200190929190505050610310565b005b6100b861042f565b604051808281526020019150506040518091

Figure 6 : Détails de la transaction pour le déploiement d'un smart contract

Nous pouvons constater que dans la transaction ci-dessus, nous avons :

- La présence de la clef publique du contrat :

**0xcaca723851462e2ba0469745e30c0fa492cbf5da**

- Un cout de transaction plus élevé que le coup de transaction fournit dans l'exemple du polycopié. En effet, les coûts de transactions sont bien plus élevés, puisque l'état de la blockchain est différente (les conditions différents). Lors de notre transaction le nombre de transaction est plus important et donc le coût de traitement de ces dernières augmente (plus de nœuds présents donc plus de transferts nécessaires).

### 3. Interactions avec un Smart Contract

Une fois le contrat déployé, nous devrions pouvoir interagir avec en lui envoyant des transactions passant en paramètre des valeurs traitées par ses fonctions. Pour plus de simplicités, nous utiliserons l'interface de l'outil.

Nous allons donc générer des utilisateurs au sein de ce dernier :

- Génération d'un candidat « PIPERAUD » :  
Nous pouvons relever la transaction suivante émise :

Transaction Details

[ This is a Ropsten Testnet transaction only ]

Transaction Hash: 0xf7918e74f83c6fcea60ed68728793d2cabccbe710b97461c089fba16c68613b

Status: Success

Block: 8636460 2 Block Confirmations

Timestamp: 50 secs ago (Sep-07-2020 09:13:12 AM +UTC)

From: 0xc447d09ee09ee8973d2e3d393495d42fc771765d

Overview State

Value: 0 Ether (\$0.00)

Transaction Fee: 0.000130059 Ether (\$0.000000)

Gas Limit: 88,189

Gas Used by Transaction: 86,706 (98.32%)

Gas Price: 0.0000000015 Ether (1.5 Gwei)

Nonce Position: 2 22

Input Data:

```
Function: addCandidate(string name) ***
MethodID: 0x462e91ec
[0]: 0000000000000000000000000000000000000000000000000000000000000020
[1]: 0000000000000000000000000000000000000000000000000000000000000008
[2]: 50495045524155440000000000000000000000000000000000000000000000000000
```

View Input As

Click to see Less

Figure 7 : Détail de la transaction permettant l'appel à une fonction du Smart Contract



Suite à cette transaction, nous pouvons observer que le réseau à bien accepté et diffusé l'information. Nous avons désormais un nouvel utilisateur à l'id n°1 :



Figure 8 : Visualisation des données de l'ID 1 du Smart Contract sur l'outil Remix

- Génération d'un candidat « Other » :  
Nous pouvons relever la transaction suivante émise :

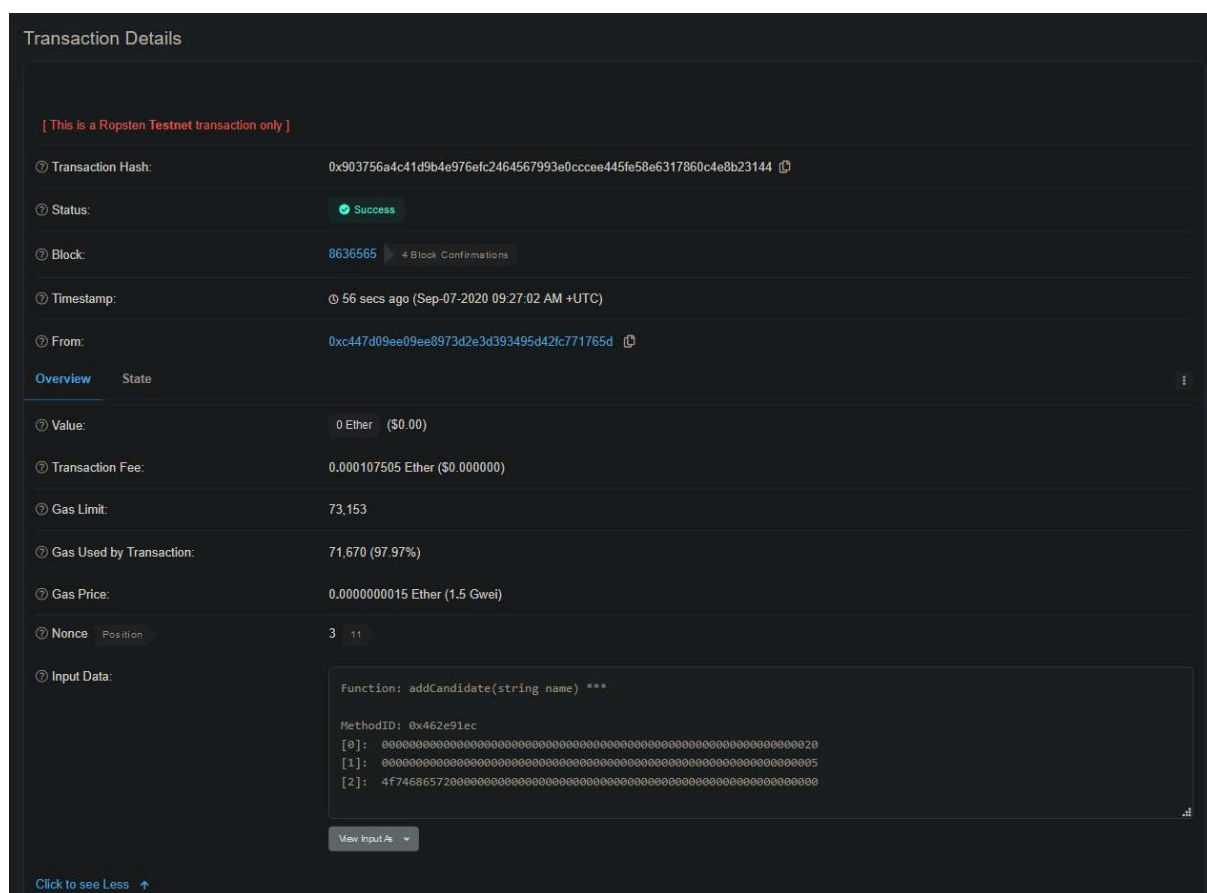


Figure 9 : Détails de la seconde transaction permettant l'appel à une fonction du Smart Contract



Nous pouvons de nouveau constater la génération d'un nouvel utilisateur au sein de la blockchain pour notre Smart Contract :



Figure 10 : Visualisation des données de l'ID 2 du Smart Contract sur l'outil Remix

Nous pouvons observer ici que nous avons bien une incrémentation du nombre de candidats enregistrés auprès de notre contrat sur le réseau.

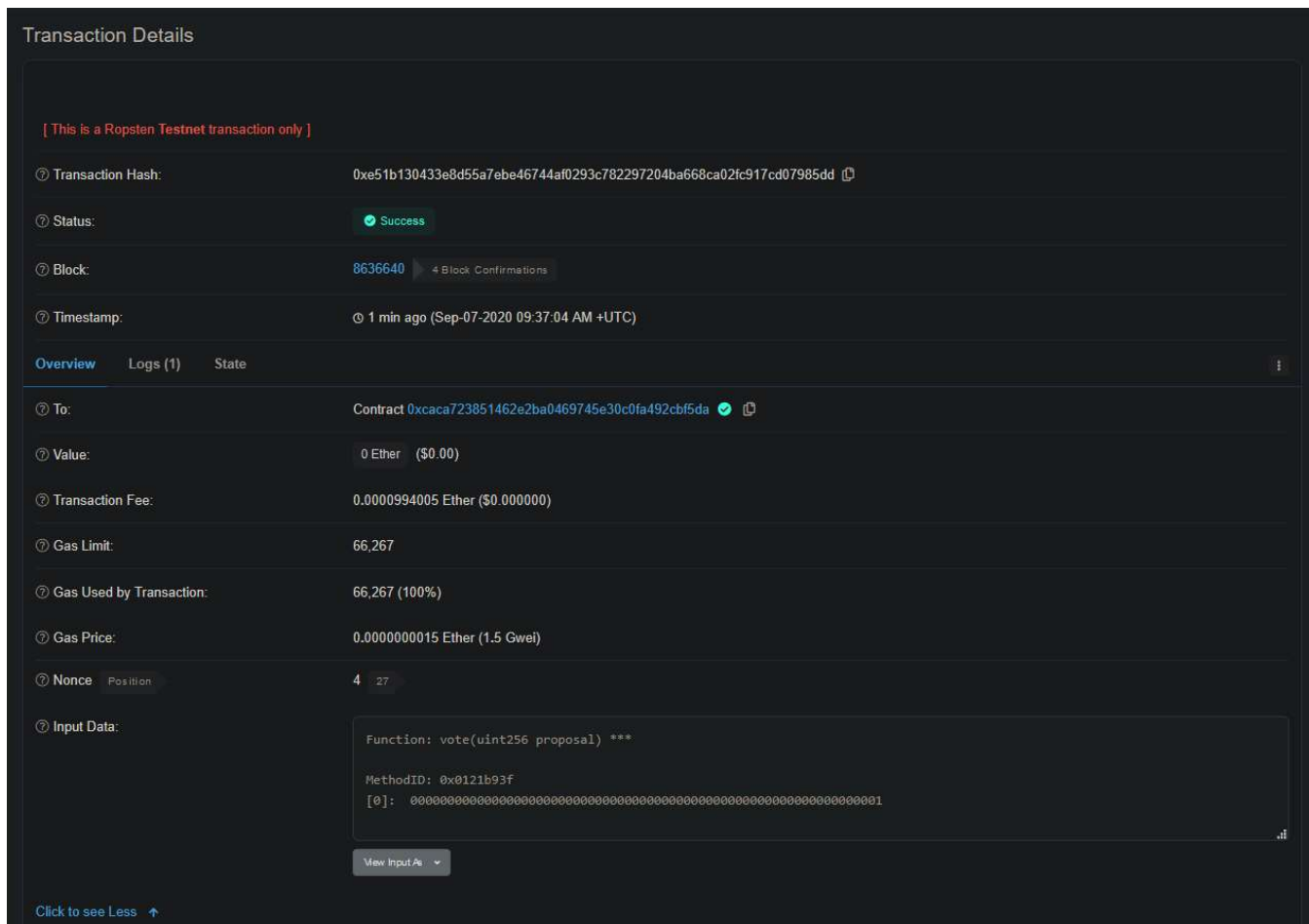
Suite à la génération de ces deux candidats, nous pouvons bien observer que le contrat suit ce que nous avons généré. Ce dernier génère bien la création de candidats lors de l'appel à cette fonctionnalité avec l'intégralité des valeurs nécessaire (Id, Nom, Nombre de votes) et retourne bien le nombre total de vote. Nous pouvons aussi observer la présence de l'adresse du détenteur du contrat (aillant les privilèges sur ce dernier) :

***0xc447d09EE09ee8973d2e3D393495D42Fc771765d***

Etant le propriétaire du contrat, il est donc normal que nous y retrouvions notre adresse publique.

Nous allons désormais pouvoir tester la fonctionnalité de vote afin d'observer les interactions lorsque nous ou un utilisateur externe (clef publique différente) tente d'interagir avec notre contrat :

- Emission d'un vote pour le premier candidat de la liste (ID 1) :  
Nous pouvons donc relever la transaction suivante pour l'émission d'un vote :



Transaction Details

[ This is a Ropsten Testnet transaction only ]

Transaction Hash: 0xe51b130433e8d55a7ebe46744af0293c782297204ba668ca02fc917cd07985dd

Status: Success

Block: 8636640 4 Block Confirmations

Timestamp: 1 min ago (Sep-07-2020 09:37:04 AM +UTC)

Overview Logs (1) State

To: Contract 0xcaca723851462e2ba0469745e30c0fa492cbf5da

Value: 0 Ether (\$0.00)

Transaction Fee: 0.0000994005 Ether (\$0.000000)

Gas Limit: 66,267

Gas Used by Transaction: 66,267 (100%)

Gas Price: 0.0000000015 Ether (1.5 Gwei)

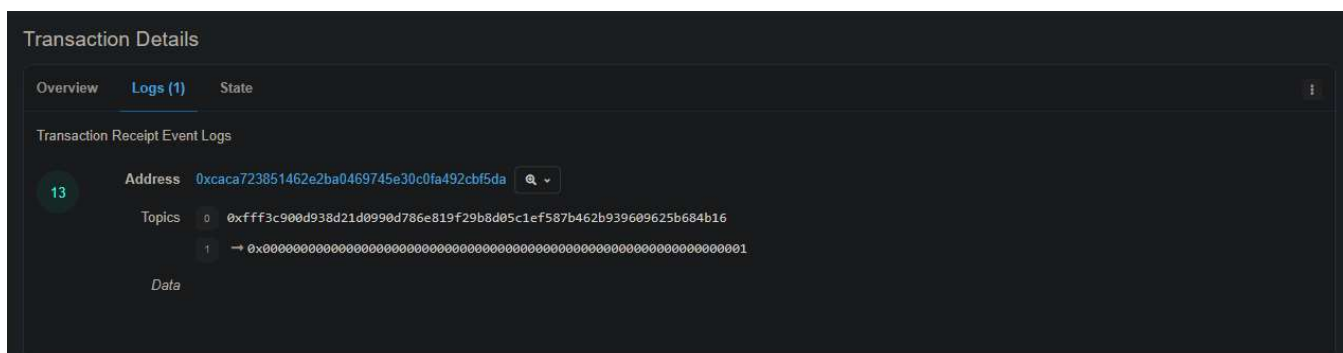
Nonce: 4 Position: 27

Input Data:

```
Function: vote(uint256 proposal) ***
MethodID: 0x0121b93f
[0]: 0000000000000000000000000000000000000000000000000000000000000001
```

Click to see Less

Figure 11 : Détails de la transaction permettant l'appel à une fonction de vote du Smart Contract



Transaction Details

Overview Logs (1) State

Transaction Receipt Event Logs

13 Address: 0xcaca723851462e2ba0469745e30c0fa492cbf5da

Topics: 0 0x0001

1 → 0x0001

Data

Figure 12 : Traces générées par le contrat lors de l'exécution de la fonction

Nous pouvons observer ci-dessus que nous avons bien eu une réaction au sein du code du contrat avec la modification d'une valeur. Lorsque que nous vérifions l'état de notre candidat par le biais de l'interface graphique, nous pouvons aussi observer ce changement :



Figure 13 : Visualisation des données de l'ID 1 du Smart Contract sur l'outil Remix

De plus, nous pouvons aussi observer que le nombre total de votes a été automatique incrémenté.

- Emission d'un vote pour le premier candidat de la liste (ID1), en provenance d'une autre clef public :

Nous pouvons donc relever la transaction suivante émise par le détenteur de la clef public :

*0xc8134fa8c874359e9aa8ecd005af6a409446a59a*

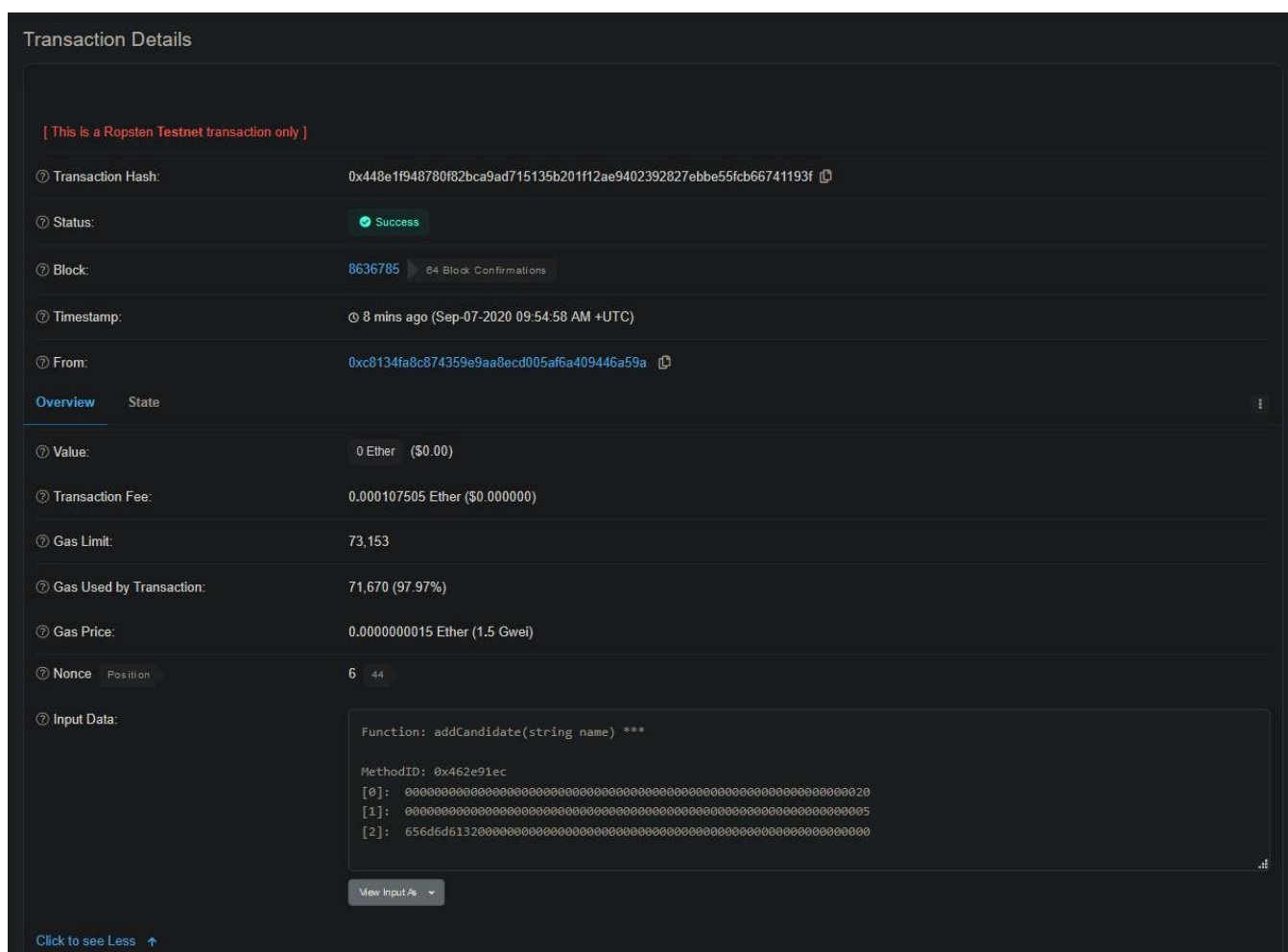


Figure 14 : Détails de la transaction permettant l'appel à une fonction de vote du Smart Contract depuis une autre clef public

Nous pouvons donc observer que nous avons bien eu un vote émis par le propriétaire d'une autre clef sur notre contrat.

## 4. Gestion et administration d'un Smart Contract

Nous pouvons désormais administrer le Smart Contract utilisé jusqu'à afin de céder les droit à un autre utilisateur de la blockchain. Pour ce faire, nous utiliserons de nouveau l'interface Remix e récupérant l'adresse public d'un autre utilisateur et en la renseignant dans le champ approprié. Nous utiliserons l'adresse publique ci-contre :

0xc8134fa8c874359e9aa8ecd005af6a409446a59a

Nous pouvons observer les détails de la transaction émis sur le réseau :

**Transaction Details**

[ This is a Ropsten Testnet transaction only ]

Transaction Hash: 0x2ce282af9f223c111b61c63e4f6ab2fd8d010b95943f32b3530c21d943fad769

Status: Success

Block: 8636996 4 Block Confirmations

Timestamp: 59 secs ago (Sep-07-2020 10:21:31 AM +UTC)

**Overview** Logs (1) State

To: Contract 0xcaca723851462e2ba0469745e30c0fa492cbf5da

Value: 0 Ether (\$0.00)

Transaction Fee: 0.000046323 Ether (\$0.000000)

Gas Limit: 30,882

Gas Used by Transaction: 30,882 (100%)

Gas Price: 0.000000015 Ether (1.5 Gwei)

Nonce: 6

Input Data:

```
Function: transferOwnership(address newOwner) ***
MethodID: 0xf2fde38b
[0]: 000000000000000000000000c8134fa8c874359e9aa8ecd005af6a409446a59a
```

View Input As

Click to see Less

Figure 15 : Détails de la transaction permettant le transfert de propriété de ce dernier

**Transaction Details**

**Overview** Logs (1) State

Transaction Receipt Event Logs

Address: 0xcaca723851462e2ba0469745e30c0fa492cbf5da

Topics:

- 0: 0x8be0079c53165914134cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0
- 1: → 0x000000000000000000000000c447d09ee09ee8973d2e3d393495d42fc771765d
- 2: → 0x000000000000000000000000c8134fa8c874359e9aa8ecd005af6a409446a59a

Data

Figure 16 : Traces générées par le contrat lors de l'appel à la fonction de transfert de propriété

Nous pouvons observer que nous avons bien eu l'appel à une méthode de notre contrat et pouvons relever la présence des clefs publiques respectives de l'émetteur et du récepteur des privilèges.

De plus, nous pouvons observer par le biais de l'interface graphique un changement dans la section propriétaire :



Figure 17 : Visualisation du propriétaire du Smart Contract sur l'outil Remix

Nous pouvons donc conclure que nous avons bien effectué et diffusé le transfert des propriétés de notre contrat à un autre tiers.

Aillant vue la présence de propriété sur les contrats et la possibilité de leurs transfert, nous pouvons désormais manipuler les fonctions afin de leurs ajouter des sécurités et restrictions. En effet, nous pouvons au sein du code, lors de la déclaration des méthodes du contrat, mettre des restrictions sur l'utilisation de ces dernières. Comme le montre l'exemple ci-dessous, nous pouvons déclarer la stricte utilisation de notre méthode au propriétaire du contrat :

```
function addCandidate (string memory _name) public {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
```

Figure 18 : Extrait de code d'une méthode sans restrictions d'utilisations

```
function addCandidate (string memory _name) public onlyOwner {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
```

Figure 19 : Extrait de code d'une méthode avec restriction d'utilisation à « onlyOwner »

Nous pouvons donc rajouter une sorte d'héritage à notre fonction, qu'y sera appelée lorsque la méthode « addCandidate » sera utilisée.

Nous déclarons donc une méthode « modifier » qu'y sera appelée avant son exécution :

```
/**
 * @dev Throws if called by any account other than the owner.
 */
modifier onlyOwner() {
    require(msg.sender == owner, "Not authorized operation");
    _;
}
```

Figure 20 : Extrait de la méthode de control d'utilisation des méthodes du contrat

La déclaration ci-dessus va donc vérifier au sein de la requête émise que la clef public correspond à celle enregistrée dans les données du contrat (celle de la clef propriétaire), et rejettera avec un message d'erreur toute transaction utilisant une autre clef.

Afin de pouvoir tester ce fonction, nous aurions pu de nouveau émettre une demande d'ajout de candidat auprès du contrat avec la clef public propriétaire et une autre clef d'un autre tiers.