

Université Cheikh Anta Diop de Dakar

Ecole Supérieure Polytechnique



**ECOLE SUPERIEURE  
POLYTECHNIQUE**

Département Génie Informatique

**A7 : Etude d'un système blockchain de consortium cas de  
Quorum  
Groupe 7**

**Présenté par :**

**Classe :** M2GLSI

Anta Diop MBAYE

Mame Meissa DIENG

Victorine Sady NDIAYE

Zeinebou Mohamed MAHMOUD

Année universitaire : 2022/2023

## **PLAN**

- I. Etude du Sujet
  - I.1 Qu'est-ce que la Blockchain ?
  - I.2 Blockchain dans l'administration des biens fonciers
- II. Présentation de Quorum
  - II.1 Definition de Quorum Blockchain
  - II.2 Comment fonctionne Quorum?
  - II.3 Les avantages et les limites de Quorum Blockchain
- III. Implémentation

## **I. Etude du sujet**

### **I.1 Qu'est-ce que la Blockchain ?**

La technologie Blockchain est un mécanisme de base de données avancé qui permet un partage transparent des informations au sein d'un réseau d'entreprises. Une base de données Blockchain stocke les données dans des blocs qui sont reliés entre eux dans une chaîne. Les données sont chronologiquement cohérentes, car vous ne pouvez pas supprimer ou modifier la chaîne sans le consensus du réseau. Par conséquent, vous pouvez utiliser la technologie Blockchain pour créer un grand livre inaltérable ou immuable pour le suivi des ordres, des paiements, des comptes et d'autres transactions. Le système dispose de mécanismes intégrés qui empêchent les entrées de transactions non autorisées et créent une cohérence dans la vue partagée de ces transactions.

### **I.2 Blockchain dans l'administration des biens fonciers**

Dans l'administration des biens fonciers la blockchain peut être utilisée pour :

- Stocker et gérer les enregistrements de propriété foncière de manière transparente, sécurisée et efficace ;
- Aider à éliminer les erreurs et les fraude dans les enregistrements fonciers ;
- Améliorer la transparence dans les transactions immobilières et faciliter la gestion des conflits liés à la propriété foncière ;
- Rendre le processus de transfert de propriété plus rapide et moins coûteux en automatisant de nombreuses étapes.

## **II. Présentation de Quorum**

### **II.1 Définition de Quorum Blockchain**

Quorum est à la base une plateforme de blockchain privée qui a été mise en place comme soft fork de l'Ethereum **par JP Morgan Chase** qui se trouve être une structure financière extrêmement développée dans le secteur. En gardant comme base les principales spécificités de l'ETH, le quorum a tout de même ajouté ses propres options à l'image de :

- La création de réseaux privés ;
- La confidentialité des échanges ;
- Un haut débit de transaction.

Ajoutons à cela que Quorum reste une création en open source, c'est-à-dire qu'il s'agit d'un programme web ouvert auquel peut contribuer n'importe quel autre développeur. Ayant comme base l'Ethereum, le Quorum présente le même langage de programmation à savoir Solidity qui est déjà connu par les nombreux utilisateurs de la crypto. Il en va de même pour les devises ou encore les jetons créés sur le projet qui seront compatibles avec les normes appliquées par l'ETH.

## II.2 Comment fonctionne Quorum?

Le fonctionnement de Quorum se base sur des nœuds validateurs qui sont autorisés à valider les transactions et à maintenir une copie de la chaîne de blocs. Pour valider une transaction, un certain nombre de nœuds doivent atteindre un consensus sur son validité. Ce nombre est défini par le Quorum.

Quorum offre également une confidentialité accrue en permettant aux parties impliquées dans une transaction de contrôler qui peut voir les détails de la transaction. Les transactions confidentielles sont cryptées et accessibles uniquement aux participants autorisés.

## II.3 Les avantages et les limites de Quorum Blockchain

Ce système a plusieurs avantages et caractéristiques :

- Sans intermédiaire : Les utilisateurs sont connectés les **uns aux autres** (système peer 2 peer, pair à pair).
- Conséquence directe du premier point, il n'y a **ni organe de contrôle ni tiers de confiance** (si ce n'est l'ensemble des utilisateurs), ce qui a de nombreux impacts et ouvre beaucoup de potentialités.
- **Sécurisée** et très difficilement piratable : Puisque chacun peut librement devenir un nœud du réseau, il deviendra lui-même hébergeur des données, ce qui ajoute à chaque fois un point de contrôle. Il faudrait pour pouvoir pirater le système posséder plus de la moitié des nœuds pour envoyer des informations erronées, ce qui devient très difficile dès qu'il y a plusieurs milliers d'utilisateurs du système.
- L'impossibilité d'effacer des données : C'est un autre avantage de la blockchain, toutes les informations stockées à chaque nouveau bloc ne peuvent plus être modifiées ou supprimées.

Les limites de Quorum incluent les éléments suivants :

- **Scalabilité** : Comme avec de nombreux autres systèmes de blockchain, Quorum peut avoir des problèmes de scalabilité lorsqu'il est soumis à une forte demande de transactions.
- **Complexité** : La mise en place et la gestion d'un réseau Quorum peuvent être complexes et nécessitent une expertise technique.
- **Confidentialité** : Bien que Quorum offre une confidentialité accrue par rapport à d'autres systèmes de blockchain publics, cela peut également présenter des défis en termes de protection de la vie privée et de conformité réglementaire.
- **Interopérabilité** : Quorum est conçu pour fonctionner en interne au sein d'une organisation, il peut donc être difficile d'interconnecter différents réseaux Quorum entre eux.
- **Coût** : L'utilisation de Quorum peut entraîner des coûts supplémentaires pour les infrastructures informatiques et les ressources humaines nécessaires pour le mettre en place et le gérer.

### III. Implémentation

On commence par créer un répertoire sur notre VM :

```
root@meissadieng-VirtualBox:~# mkdir quorum
root@meissadieng-VirtualBox:~# cd quorum/
root@meissadieng-VirtualBox:~/quorum#
```

Puis on installe le **quorum-wizard**

```
root@meissadieng-VirtualBox:~/quorum# sudo npm install -g quorum-wizard
/usr/bin/quorum-wizard -> /usr/lib/node_modules/quorum-wizard/build/index.js
+ quorum-wizard@1.3.3
added 176 packages from 125 contributors in 43.292s
root@meissadieng-VirtualBox:~/quorum#
```

Une fois installé, on peut lancer la commande suivante :

```
root@meissadieng-VirtualBox:~/quorum# quorum-wizard
?
Welcome to Quorum Wizard!

This tool allows you to easily create bash, docker, and kubernetes files to start up a quorum network.
You can control consensus, privacy, network details and more for a customized setup.
Additionally you can choose to deploy our chain explorer, Cakeshop, to easily view and monitor your network.

We have 3 options to help you start exploring Quorum:

  1. Quickstart - our 1 click option to create a 3 node raft network with tessera and cakeshop

  2. Simple Network - using pregenerated keys from quorum 7nodes example, this option allows you to choose the number of nodes (7 max), consensus mechanism, transaction manager, and the option to deploy cakeshop
```

L'installateur nous propose alors plusieurs choix, nous allons sélectionner '**Quickstart**' :

```
(Use arrow keys)
> Quickstart (3-node raft network with tessera and cakeshop)
  Simple Network
  Custom Network
  Generate from Existing Configuration
  Exit
```

Si tout s'est bien passé, vous devriez voir quelque le message '**Quorum network created**' comme ci-dessous :

```
Tessera Node 1 public key:
BULeR8JyUWhiuuCMU/HLA0Q5pzKYT+cHII3ZKBey3Bo=

Tessera Node 2 public key:
QfeDAys9MPDs2XHExtc84jKGHxZg/aj52DTh0vtA3Xc=

Tessera Node 3 public key:
1iTZde/ndBHvzhcl7V68x44Vx7pl8nwx9LqnM/AfJUg=

-----
-

Quorum network created

Run the following commands to start your network:

cd /root/quorum/network/3-nodes-quickstart
./start.sh

A sample simpleStorage contract is provided to deploy to your network
To use run ./runscript.sh public_contract.js from the network folder

A private simpleStorage contract was created with privateFor set to use Node 2'
s public key: QfeDAys9MPDs2XHExtc84jKGHxZg/aj52DTh0vtA3Xc=
```

Maintenant que tout est prêt, on peut aller lancer notre blockchain

```
root@meissadieng-VirtualBox:~/quorum# cd network/3-nodes-quickstart/
root@meissadieng-VirtualBox:~/quorum/network/3-nodes-quickstart# ./start.sh
```

Lorsque le script se termine :

```
Cakeshop is not yet listening on http
Waiting until Cakeshop is running...
Cakeshop is not yet listening on http
Waiting until Cakeshop is running...
Cakeshop is not yet listening on http
Waiting until Cakeshop is running...
Cakeshop started at http://localhost:8999
Successfully started Quorum network.
-----
-

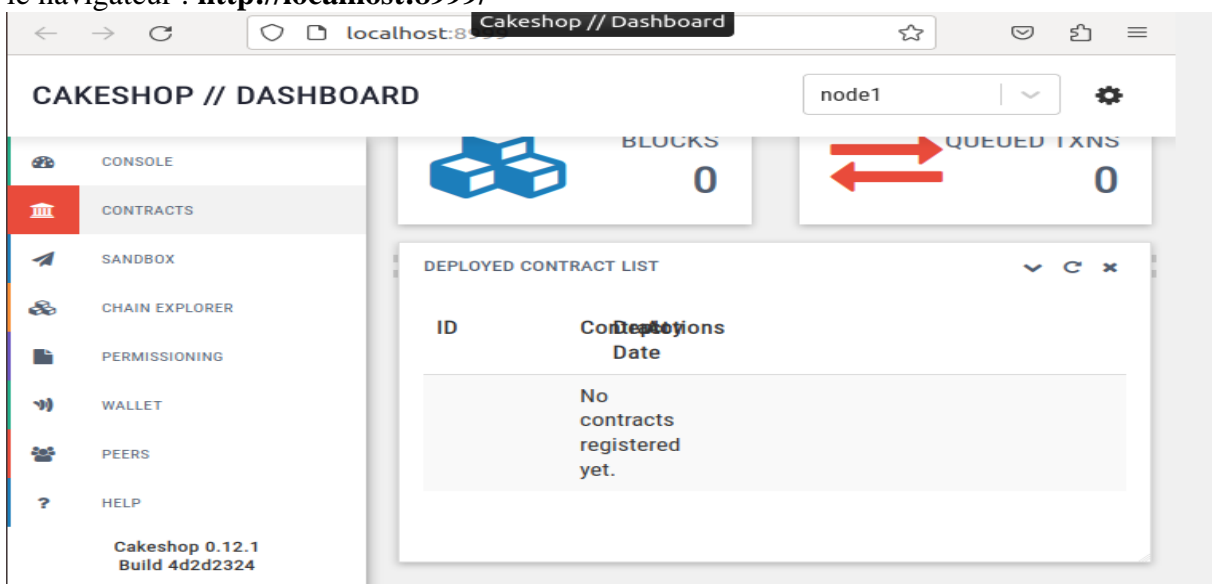
Tessera Node 1 public key:
BULeR8JyUWhiuuCMU/HLA0Q5pzkyT+cHII3ZKBey3Bo=

Tessera Node 2 public key:
QfeDAys9MPDs2XHExtc84jKGHxZg/aj52DTh0vtA3Xc=

Tessera Node 3 public key:
1iTZde/ndBHvzhcl7V68x44Vx7pl8nwx9LqnM/AfJUg=
-----
-

root@meissadieng-VirtualBox:~/quorum/network/3-nodes-quickstart#
```

Pour l'utiliser, si l'étape précédente s'est bien passée, il suffit d'ouvrir l'adresse suivante dans le navigateur : <http://localhost:8999/>



- **Déploiement d'un premier réseau et exécution d'une transaction**

Nous allons maintenant déployer un premier réseau et exécuter une transaction

Voici le code du contrat :

```

1  pragma solidity >=0.4.22 <0.9.0;
2
3  contract Message {
4
5      string lemessage;
6
7      constructor(string memory _messageoriginal) public {
8          lemessage = _messageoriginal;
9      }
10
11     function definirMessage(string memory _nouveaumessage) public{
12         lemessage = _nouveaumessage;
13     }
14
15     function voirMessage() public view returns (string memory){
16         return lemessage;
17     }
18
19 }

```

The screenshot shows the Remix IDE interface. At the top, there's a toolbar with icons for navigation, compilation, and help. Below the toolbar, a 'Compile' button is visible next to a dropdown menu showing 'node1' and a gas price of '#22'. The main interface is divided into several panels:

- Choose Contract:** This panel has two sections. 'From Deployed Contracts:' has an empty dropdown. 'Or Deploy From Editor:' has a dropdown menu with 'Message' selected. Below this, the 'Private For' section shows a dropdown with 'BULeR...' and a 'Deploy' button. At the bottom, there's a text input field containing 'test' and another section 'Or Enter Address:' with a text input field containing '0xe273fd8a6deb07c4bd7226'.
- Accounts:** This panel shows a single account with the address '0xed9d02...' and a balance of '10000000000.00'.
- Contract State:** This panel shows the state of the deployed contract, with a text input field containing 'payload'.
- Paper Tape:** This panel shows a log of transactions. It contains two entries:
  - [compile] Compiling 03:38:10 PM Untitled.sol
  - [deploy] Contract 03:38:17 PM 'Message' (test)



Transact

FROM ADDRESS

0xed9d02e382b3...

Private For

BULeR... x x

definirMessage

Transact

test

voirMessage

Read

[txn] 0x9720ef88 was 03:39:28 PM committed in block #24

[read] voirMessage() => 03:39:32 PM "test"

Cakeshop // Dashboard

Cakeshop // Sandbox

localhost:8999

node1

CONSOLE

CONTRACTS

SANDBOX

CHAIN EXPLORER

PERMISSIONING

WALLET

PEERS

HELP

▶

NODE STATUS

Running

PEERS

3

BLOCKS

7

QUEUED TXNS

0

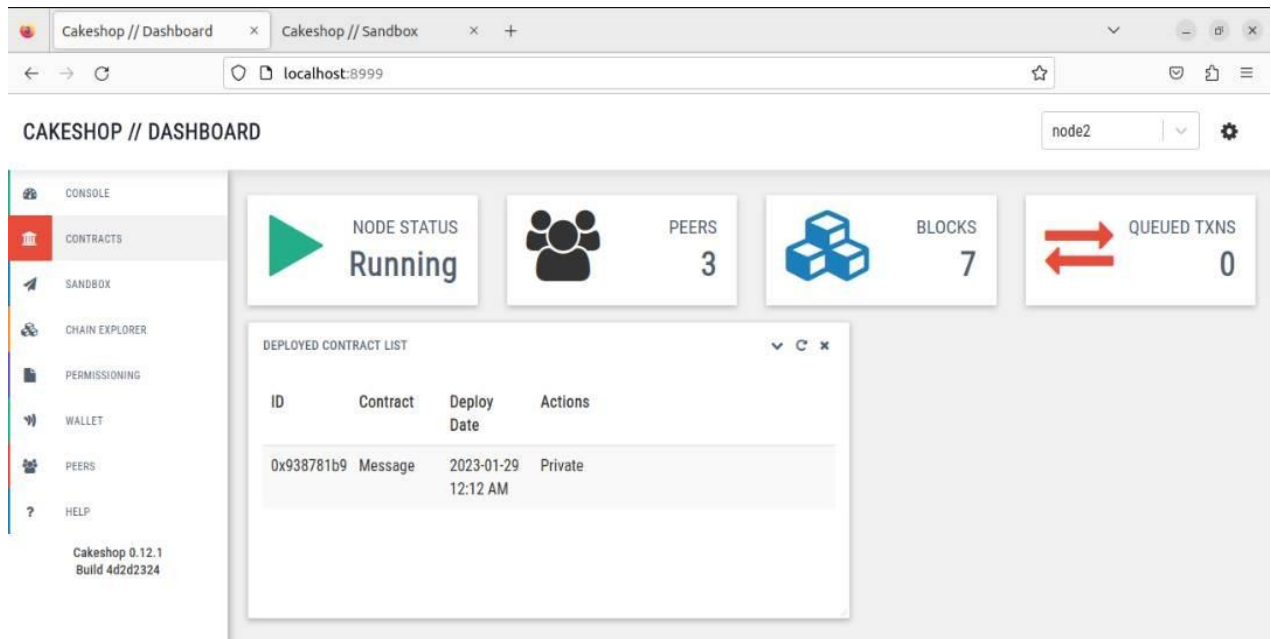
DEPLOYED CONTRACT LIST

ID	Contract	Deploy Date	Actions
0x938781b9	Message	2023-01-29 12:12 AM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>

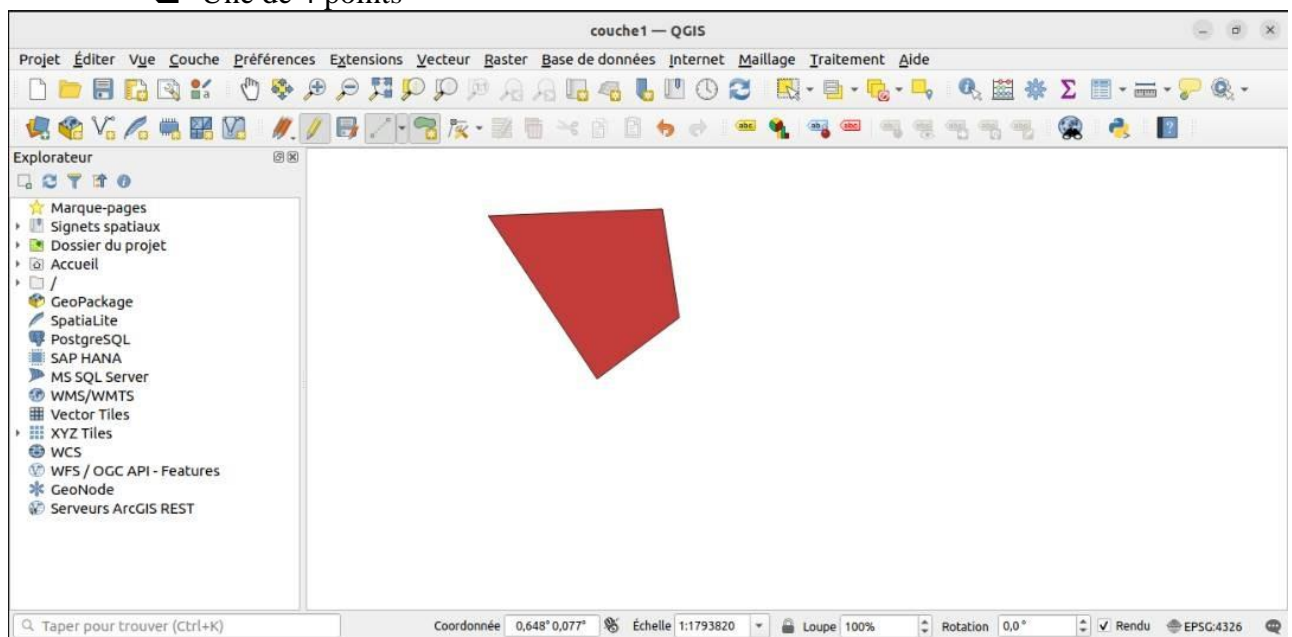
Cakeshop 0.12.1

Build 4d2d2324

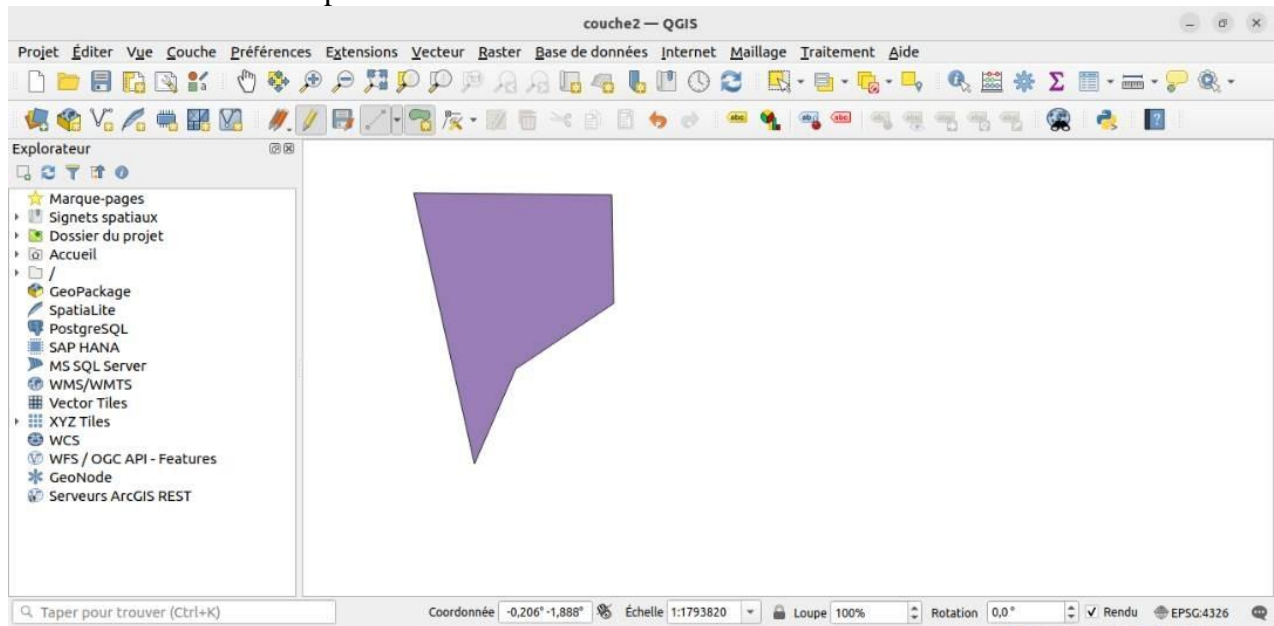
9



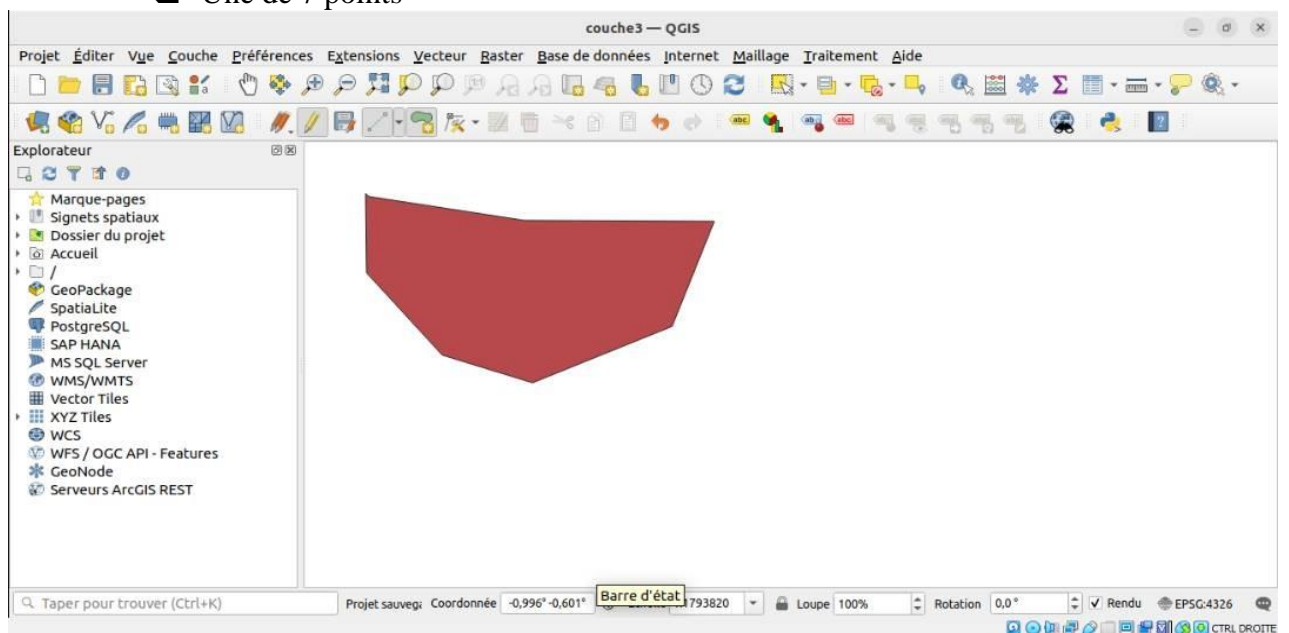
- **Création des parcelles avec QGIS**
  - ☐ Une de 4 points



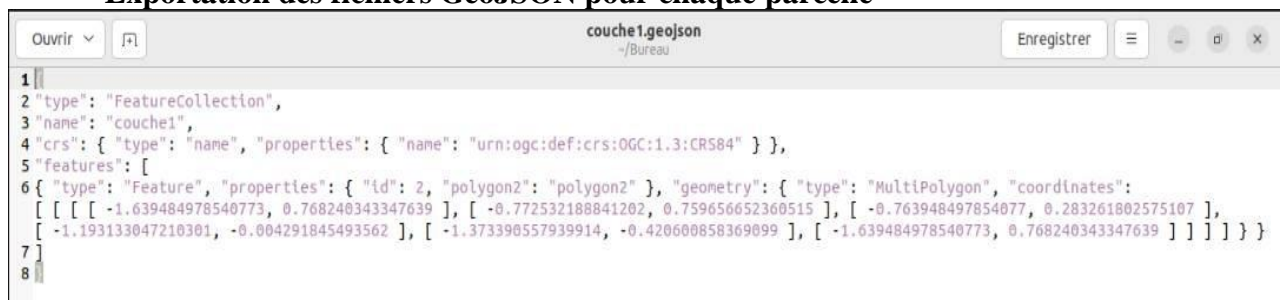
☐ Une de 5 points



☐ Une de 7 points



- **Exportation des fichiers GeoJSON pour chaque parcelle**



```

1
2 "type": "FeatureCollection",
3 "name": "couche2",
4 "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
5 "features": [
6 { "type": "Feature", "properties": { "id": 2, "polygon2": "polygon2" }, "geometry": { "type": "MultiPolygon", "coordinates":
7 [ [ [ [ -1.639484978540773, 0.768240343347639 ], [ -0.772532188841202, 0.759656652360515 ], [ -0.763948497854077, 0.283261802575107 ],
8 [ [ -1.193133047210301, -0.004291845493562 ], [ -1.373390557939914, -0.420600858369099 ], [ -1.639484978540773, 0.768240343347639 ] ] ] ] }

```

```

1
2 "type": "FeatureCollection",
3 "name": "couche3",
4 "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
5 "features": [
6 { "type": "Feature", "properties": { "id": 3, "polygon3": "polygon3" }, "geometry": { "type": "MultiPolygon", "coordinates":
7 [ [ [ [ -1.167381974248927, 0.648068669527897 ], [ -0.334763948497854, 0.643776824034335 ], [ -0.51931330472103, 0.171673819742489 ],
8 [ [ -1.128755364806867, -0.081545064377682 ], [ -1.523605150214592, 0.042918454935622 ], [ -1.854077253218884, 0.412017167381974 ],
9 [ [ -1.858369098712446, 0.768240343347639 ], [ -1.84549356223176, 0.755364806866953 ], [ -1.167381974248927, 0.648068669527897 ] ] ] ] }

```

- **Création d'une transaction avec un payload constitué du GeoJSON ainsi créer**

Voici le code du contrat :

The screenshot shows a web-based IDE interface for deploying a Solidity contract. On the left, the contract code is displayed in a text editor with the following content:

```

1 pragma solidity ^0.8.0;
2
3 contract GeoJSONTransaction {
4     string public payload;
5
6     function setPayload(string memory _payload) public {
7         payload = _payload;
8     }
9 }

```

On the right side of the interface, there are several panels:

- Choose Contract:** A dropdown menu showing "From Deployed Contracts:" and "Or Deploy From Editor:". Below it, a "Private For" field contains the address "BULeR8JyUWhiu..." with a dropdown arrow.
- Accounts:** A table showing account information, including the address "0xed9d02e38..." and a balance of "1000000000.00".
- Contract State:** A panel showing the state of the contract.
- Paper Tape:** A panel showing a list of deployment events. The first event is a "Post" message with the cause "http+unix://c/send: net/http: timeout awaiting response headers". Below this, there are three "deploy" events for the contract "GeoJSONTransaction", each with a timestamp of "04:04:42 PM" or "04:04:47 PM".

At the bottom right, there is a "Deploy" button and a field labeled "Or Enter Address:" with the text "address" inside.

## ❑ Transaction pour la couche 1

localhost:8999/sandbox.html

Or Enter Address:  
0xd9d64b7dc034fafdba5dc2902875a67

Paper rape  
[state] payload: "" => "{ \"type\": \"04:11:25 PM  
\"FeatureCollection\", \"name\": \"couche1\",  
\"crs\": { \"type\": \"name\", \"properties\": {  
\"name\": \"urn:ogc:def:crs:OGC:1.3:CRS84\" } }

Transact

FROM ADDRESS  
0xd9d02e382b34818e88... ▾

Private For  
BULeR8JyUWhiuu... x ▾

payload Read

setPayload Transact

i773, 0.768240343347639 ]]]]]]]

## ❑ Transaction pour la couche 2

Or Enter Address:  
0xd9d64b7dc034fafdba5dc2902875a67

[state] payload: "{ \"type\": \"04:16:58 PM  
\"FeatureCollection\", \"name\": \"couche2\",  
\"crs\": { \"type\": \"name\", \"properties\": {  
\"name\": \"urn:ogc:def:crs:OGC:1.3:CRS84\" } }

Transact

FROM ADDRESS  
0xd9d02e382b34818e88... ▾

Private For  
BULeR8JyUWhiuu... x ▾

payload Read

setPayload Transact

i927, 0.648068669527897 ]]]]]]]

## ❑ Transaction pour la couche 3

Or Enter Address:

**Transact**

**FROM ADDRESS**

**Private For**

**payload** Read

**setPayload** Transact

[state] payload: "" => "{ \"type\": \"FeatureCollection\", \"name\": \"couche3\", \"crs\": { \"type\": \"name\", \"properties\": { \"name\": \"urn:ogc:def:crs:OGC:1.3:CRS84\" } }

Ci-dessous nous avons les transactions créées :

CAKESHOP // DASHBOARD

node1

CONSOLE

CONTRACTS

SANDBOX

CHAIN EXPLORER

PERMISSIONING

WALLET

PEERS

HELP

Cakeshop 0.12.1  
Build 4d2d2324

**NODE STATUS**  
Running

**PEERS**  
3

**BLOCKS**  
10

**QUEUED TXNS**  
0

DEPLOYED CONTRACT LIST

ID	Contract	Deploy Date	Actions
0x1932c48b	Message	2023-01-30 12:20 PM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>
0x8a5e2a63	GeoJSONTransaction	2023-01-30 04:04 PM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>
0x9d13c6d3	GeoJSONTransaction	2023-01-30 04:04 PM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>
0xa501afd7	GeoJSONTransaction	2023-01-30 04:23 PM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>
0xd9d64b7d	GeoJSONTransaction	2023-01-30 04:04 PM	<a href="#">Transact</a> <a href="#">Details</a> <a href="#">Current State</a>



- Création d'une transaction avec un payload constitué du hash de la parcelle à 4 sommets :

```

+ Code + Texte
Reconnecter Modification ^

!pip install polygeohasher

from polygeohasher import polygeohasher
import geopandas as gpd

gdf = gpd.read_file("couche1.geojson") # read your geometry file here

primary_df = polygeohasher.create_geohash_list(gdf, 6, inner=False) # returns a dataframe with list of ge
secondary_df = polygeohasher.geohash_optimizer(primary_df, 5, 6, 6) # returns optimized list of geohash
polygeohasher.optimization_summary(primary_df, secondary_df) #creates a summary of first and second outp
geo_df = polygeohasher.geohashes_to_geometry(secondary_df, "optimized_geohash_list") # return geometry 1
geo_df.to_file("polygon1.geojson", driver = "GeoJSON") #write file in your favorite spatial file format

Downloading geopandas-0.12.2-py3-none-any.whl (1.1 MB)
1.1/1.1 MB 17.2 MB/s eta 0:00:00
Collecting Geohash
Downloading Geohash-1.0.tar.gz (15 kB)
Preparing metadata (setuptools) done
✓ 14 s terminée à 16:59

```

```

1 pragma solidity ^0.8.0;
2
3 contract GeoJSONTransaction {
4     string public payload;
5
6     function setPayload(string memory _payload) public {
7         payload = _payload;
8     }
9 }

```

**Private For**

BULeR8JyUWhluu... x

Deploy

**Or Enter Address:**

0xfc1684a8af3e6bf532482c356e5b2ce

**Paper Tape**

```

{
  "polygon2": {
    "type": "Polygon",
    "coordinates": [
      [
        [
          -1.065673828125, 0.7470703125
        ],
        [
          -1.0546875, 0.7470703125
        ],
        [
          -1.0546875, 0.7525634765625
        ],
        [
          -1.065673828125, 0.7525634765625
        ],
        [
          -1.065673828125, 0.7470703125
        ]
      ]
    ]
  }
}

```

**Transact**

**FROM ADDRESS**

0xed9d02e382b34818e88... v

**Private For**

BULeR8JyUWhluu... x