



École Supérieure Polytechnique(ESP)  
Département Génie Informatique

# Rapport de projet

Module : Cryptographie et Blockchain

DIC2

*Gestion du Stockage de Documents dans les  
Systèmes Basés sur la Blockchain  
Groupe 1 : Évaluation du Stockage Off-Chain avec  
Hyperledger et IPFS*

Réalisé par :

- Hamidou Woury Ba
- Abdoulaye Wade Mané
- Elimane Ndiaye
- Ndèye Awa Diémé

Professeur :

- Mr Mendy

Année 2024/2025

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Hyperledger Fabric</b>	<b>5</b>
2.1	Présentation générale . . . . .	5
2.2	Caractéristiques principales . . . . .	5
2.3	Cas d'utilisation . . . . .	5
<b>3</b>	<b>InterPlanetary File System (IPFS)</b>	<b>5</b>
3.1	Présentation générale . . . . .	5
3.2	Caractéristiques principales . . . . .	5
3.3	Cas d'utilisation . . . . .	6
<b>4</b>	<b>Complémentarité entre Hyperledger Fabric et IPFS</b>	<b>6</b>
4.1	Intégration des deux technologies . . . . .	6
4.2	Cas d'usage concrets . . . . .	6
<b>5</b>	<b>Comparaison des mécanismes de consensus</b>	<b>6</b>
5.1	Proof of Work (PoW) . . . . .	6
5.2	Proof of Stake (PoS) . . . . .	6
<b>6</b>	<b>Analyser les cas d'utilisation d'IPFS avec la blockchain</b>	<b>7</b>
<b>7</b>	<b>Configuration de Hyperledger</b>	<b>9</b>
<b>8</b>	<b>Configuration du Réseau Hyperledger Fabric</b>	<b>9</b>
8.1	Démarrer le Réseau . . . . .	9
8.2	Créer un Channel . . . . .	10
<b>9</b>	<b>Structure du Projet</b>	<b>11</b>
<b>10</b>	<b>Fichiers</b>	<b>11</b>
10.1	Fichier go.mod . . . . .	11
10.2	Fichier go.sum . . . . .	11
10.3	Fichier main.go . . . . .	12
10.4	Fichier setOrg1Env.sh . . . . .	12
10.5	Fichier setOrg2Env.sh . . . . .	12
10.6	Appliquer les Variables d'Environnement et affichez le contenu du fichier de configuration pour Organisation 1 . . . . .	12
<b>11</b>	<b>Déploiement du Chaincode</b>	<b>12</b>
<b>12</b>	<b>Initialisation et Transactions</b>	<b>13</b>
12.1	Initialisation du Chaincode . . . . .	13
12.2	Lecture d'un Titre Foncier . . . . .	13
12.3	Création d'une Nouvelle Transaction . . . . .	13
12.4	Expliquer comment les smartcontracts sont soumis à Hyperledger . . . . .	14
12.5	Modification d'un Titre Foncier . . . . .	14
12.6	Valider que le ledger a bien mis à jour la donnée et a créer une nouvelle transaction de MAJ . . . . .	14
<b>13</b>	<b>Configuration de IPFS</b>	<b>14</b>
13.1	Téléchargement et installation d'IPFS . . . . .	14
13.2	Installation du package go-ipfs-api . . . . .	15
13.3	Création d'un fichier sous IPFS . . . . .	15
<b>14</b>	<b>Intégrer IPFS avec Hyperledger pour le stockage des documents.</b>	<b>16</b>
14.1	Modifications du fichier main.go . . . . .	16
14.2	Création d'un titre foncier avec IPFS . . . . .	16

14.2.1 Cas 1 : Hash valide . . . . .	16
14.2.2 Cas 1 : Hash invalide . . . . .	17
<b>15 Application cliente</b>	<b>18</b>
<b>16 Structure du Projet</b>	<b>18</b>
16.1 Explication des fichiers . . . . .	18
16.2 Installation et Exécution . . . . .	18
<b>17 Fonctionnalités de l'Application</b>	<b>19</b>
17.1 Page d'Accueil . . . . .	19
17.2 Ajouter un Nouveau Titre . . . . .	20
17.3 Modifier un Titre Foncier . . . . .	21
<b>18 Tests du Système</b>	<b>24</b>
18.1 Chargement de documents de tailles variées . . . . .	24
18.1.1 Méthodologie . . . . .	24
18.1.2 Résultats . . . . .	24
18.1.3 Graphiques des résultats . . . . .	25
18.1.4 Interprétation des résultats . . . . .	25
18.1.5 Conclusion . . . . .	25
18.2 Simulation des accès concurrents . . . . .	25
18.2.1 Installation d'Apache JMeter . . . . .	25
18.2.2 Création du Test de Charge . . . . .	25
18.2.3 Configuration du Plan de Test . . . . .	26
18.2.4 Ajout de Requêtes HTTP . . . . .	26
18.2.5 Ajout de l'Analyse des Résultats . . . . .	27
18.2.6 Exécution du Test de Charge . . . . .	27
18.2.7 Conclusion . . . . .	28
<b>19 Analyse des Données</b>	<b>28</b>
19.1 Mesurer le temps de réponse . . . . .	28
19.2 Sécurité . . . . .	29
19.2.1 Génération du certificat SSL/TLS . . . . .	29
19.2.2 Modification du serveur pour utiliser HTTPS . . . . .	29
19.2.3 Connexion sécurisée et validation . . . . .	29
19.3 Permanence des données . . . . .	30
<b>20 Calculer les coûts opérationnels comparés à une solution on-chain</b>	<b>30</b>
<b>21 Rédaction du Rapport</b>	<b>31</b>
21.1 Documentation du processus . . . . .	31
21.2 Découvertes et défis rencontrés . . . . .	31
<b>22 Recommandations</b>	<b>31</b>
<b>23 Conclusion</b>	<b>32</b>

**NB** : Pour plus de détails, vous pouvez consulter les codes sources sur le dépôt GitHub suivant :  
[Lien vers le dépôt GitHub](#).

# 1 Introduction

L'essor des technologies décentralisées a conduit à l'émergence de solutions innovantes pour la gestion des données et des transactions numériques. Parmi elles, Hyperledger Fabric et InterPlanetary File System (IPFS) jouent un rôle clé.

## 2 Hyperledger Fabric

### 2.1 Présentation générale

Hyperledger Fabric est une plateforme blockchain open-source développée par la Linux Foundation. Contrairement aux blockchains publiques comme Ethereum et Bitcoin, Fabric est permissionnée, ce qui signifie que seuls les participants autorisés peuvent rejoindre et interagir avec le réseau.

### 2.2 Caractéristiques principales

- **Architecture modulaire** : Fabric offre une structure modulaire permettant aux entreprises de personnaliser différents composants, notamment les mécanismes de consensus, la gestion des identités et l'interaction avec des bases de données externes.
- **Smart Contracts (Chaincode)** : Les smart contracts, appelés chaincodes, permettent d'automatiser les transactions sur le réseau en exécutant du code métier prédéfini.
- **Canaux privés** : Les participants peuvent créer des canaux privés pour restreindre l'accès aux transactions à un sous-ensemble spécifique du réseau, garantissant ainsi une meilleure confidentialité des données.
- **Modèle de consensus flexible** : Contrairement aux blockchains classiques qui reposent sur des mécanismes énergivores comme la Preuve de Travail (PoW), Fabric permet d'utiliser des protocoles de consensus plus efficaces et adaptés aux exigences des entreprises, comme Raft et PBFT (Practical Byzantine Fault Tolerance).

### 2.3 Cas d'utilisation

- **Finance** : Sécurisation des paiements interbancaires, gestion des identités numériques et traçabilité des transactions financières.
- **Supply Chain** : Suivi des produits à chaque étape de la chaîne logistique, assurant transparence et conformité réglementaire.
- **Santé** : Stockage sécurisé et traçabilité des dossiers médicaux pour garantir l'intégrité et la confidentialité des données patient.
- **Gouvernance** : Mise en place de registres sécurisés pour les titres fonciers, les dossiers juridiques et autres documents officiels.

## 3 InterPlanetary File System (IPFS)

### 3.1 Présentation générale

IPFS est un protocole pair-à-pair (P2P) de stockage et de partage de fichiers conçu pour rendre le web plus résilient et décentralisé. Il remplace le modèle traditionnel basé sur des serveurs centralisés en adoptant une approche où les fichiers sont répartis sur un réseau de nœuds interconnectés.

### 3.2 Caractéristiques principales

- **Stockage distribué** : Contrairement aux systèmes de stockage classiques, IPFS utilise un réseau distribué où les fichiers sont dupliqués sur plusieurs nœuds, réduisant ainsi les risques de perte de données et améliorant la résilience du réseau.
- **Adressage par hachage** : Chaque fichier stocké sur IPFS reçoit un identifiant unique basé sur son contenu (Content Identifier - CID). Cette méthode garantit l'intégrité des données et empêche leur altération.

- **Optimisation du réseau** : IPFS permet un partage efficace des fichiers en téléchargeant des fragments depuis plusieurs sources simultanément, ce qui réduit la bande passante nécessaire et accélère l'accès aux données.
- **Interopérabilité avec la Blockchain** : IPFS est souvent utilisé pour stocker des fichiers associés à des smart contracts sur des blockchains publiques comme Ethereum. Il permet de stocker des métadonnées de NFT (Non-Fungible Tokens) et d'autres données volumineuses sans surcharger la blockchain.

### 3.3 Cas d'utilisation

- **Stockage Web3** : Permet aux sites web décentralisés de fonctionner sans dépendre de serveurs centralisés.
- **NFT et blockchain** : Stockage sécurisé des métadonnées des NFT pour garantir leur authenticité et leur traçabilité.
- **Archivage de données** : Conservation à long terme de documents historiques, scientifiques et gouvernementaux.
- **Partage de fichiers résilient** : Facilite l'échange de données entre pairs sans passer par des infrastructures centralisées.

## 4 Complémentarité entre Hyperledger Fabric et IPFS

Hyperledger Fabric et IPFS peuvent être utilisés ensemble pour optimiser la gestion des transactions et du stockage des données. Cette complémentarité permet d'améliorer la scalabilité et la sécurité des applications blockchain en combinant les avantages de chaque technologie.

### 4.1 Intégration des deux technologies

- Hyperledger Fabric enregistre des références aux fichiers stockés sur IPFS, au lieu de stocker directement les fichiers dans la blockchain. Cela permet de réduire la charge et les coûts liés au stockage des données volumineuses.
- IPFS assure une disponibilité et une résilience accrues en décentralisant le stockage des fichiers, tout en permettant aux smart contracts de Hyperledger Fabric d'y accéder de manière sécurisée.

### 4.2 Cas d'usage concrets

- **Gestion des dossiers médicaux** : Hyperledger Fabric enregistre les transactions relatives aux mises à jour des dossiers médicaux, tandis que IPFS stocke les documents médicaux eux-mêmes, assurant ainsi confidentialité et accessibilité.
- **Traçabilité des biens** : Fabric gère la validation et l'authenticité des transactions, et IPFS stocke les documents relatifs aux certifications et garanties de produits.
- **Gestion documentaire sécurisée** : Fabric permet de contrôler qui peut accéder à quels documents, tandis que IPFS assure un stockage distribué fiable.

## 5 Comparaison des mécanismes de consensus

### 5.1 Proof of Work (PoW)

Le PoW, utilisé par Bitcoin, repose sur la résolution de problèmes mathématiques complexes. Ce processus demande une importante puissance de calcul, ce qui le rend très sécurisé mais aussi énergivore. Ce mécanisme convient mieux aux blockchains publiques ouvertes, mais il est limité en termes de scalabilité et de temps de traitement.

### 5.2 Proof of Stake (PoS)

Le PoS, adopté par Ethereum 2.0, sélectionne les validateurs en fonction de la quantité de crypto-monnaies qu'ils bloquent en staking. Ce mécanisme réduit considérablement la consommation

d'énergie par rapport au PoW tout en augmentant la rapidité des transactions. Toutefois, il peut présenter un risque de centralisation, car les validateurs les plus riches ont davantage de chances de participer.

**Tableau comparatif**

Critère	Hyperledger Fabric	Proof of Work (PoW)	Proof of Stake (PoS)
Type de Blockchain	Permissioned	Permissionless	Permissionless
Mécanisme de Consensus	Kafka, Raft, BFT	Proof of Work	Proof of Stake
Participants	Membres autorisés	Tout public	Tout public
Énergie Consommée	Faible	Très élevée	Faible à modérée
Vitesse des Transactions	Très rapide	Lente	Modérée à rapide
Sécurité	Élevée	Élevée	Modérée à élevée
Scalabilité	Très scalable	Limité	Scalable
Cas d'Utilisation	Entreprises	Cryptomonnaies	Cryptomonnaies publiques

## 6 Analyser les cas d'utilisation d'IPFS avec la blockchain

Dans cette partie, nous allons essayer d'énumérer quelques applications clés de l'IPFS dans l'espace blockchain en donnant pour chaque application un cas d'utilisation :

### Stockage de fichiers décentralisé :

*Cas d'utilisation :* Les réseaux blockchain génèrent une quantité importante de données, y compris les transactions et les détails des contrats intelligents. Le stockage de ces données sur un système de fichiers décentralisé tel que IPFS garantit qu'elles sont distribuées sur un réseau de nœuds plutôt que de s'appuyer sur un serveur central.

### Adressage de contenu immuable :

*Cas d'utilisation :* L'adressage de contenu dans IPFS attribue un hachage cryptographique unique à chaque élément de contenu. Les projets blockchain peuvent utiliser cette fonctionnalité pour créer des références inviolables aux documents, garantissant ainsi que le contenu reste immuable au fil du temps.

### Applications décentralisées (DApps)

*Cas d'utilisation :* Les DApps nécessitent souvent du stockage pour diverses ressources, telles que des images, des vidéos et d'autres médias. IPFS peut être intégré dans les DApps pour fournir une solution de stockage décentralisée et distribuée pour ces actifs.

### Interopérabilité entre les blockchains :

*Cas d'utilisation :* Différents réseaux blockchain peuvent utiliser IPFS comme protocole commun pour le stockage décentralisé de fichiers, favorisant ainsi l'interopérabilité. Cela permet le partage et la récupération transparents de données entre des écosystèmes blockchain disparates.

### Réseaux de distribution de contenu (CDN) :

*Cas d'utilisation :* IPFS peut être utilisé comme un CDN (Content Delivery Network) décentralisé pour distribuer du contenu, tel que des fichiers multimédias, sur un réseau de nœuds. Ceci est particulièrement utile pour les projets basés sur la blockchain qui nécessitent une diffusion de contenu efficace et décentralisée.

### Horodatage immuable :

*Cas d'utilisation :* IPFS peut être utilisé pour créer des horodatages immuables pour les documents et les données stockés sur la blockchain. Cet horodatage garantit un enregistrement transparent et vérifiable du moment où des informations spécifiques ont été ajoutées au réseau IPFS.

### **Gestion de la chaîne d'approvisionnement :**

*Cas d'utilisation :* L'intégration d'IPFS dans les solutions de chaîne d'approvisionnement basées sur la blockchain garantit que les documents, les certifications et les détails des transactions pertinents sont stockés en toute sécurité et facilement récupérables de manière décentralisée.

### **Transactions de pair à pair :**

*Cas d'utilisation :* IPFS peut faciliter les transactions peer-to-peer en permettant aux utilisateurs d'échanger des fichiers directement sans dépendre d'un intermédiaire centralisé. Cela est particulièrement pertinent dans les scénarios où les parties doivent partager de grands ensembles de données en toute sécurité.

## 7 Configuration de Hyperledger

Pour configurer Hyperledger, il est nécessaire d'installer les composants suivants :

- Docker et Docker Compose

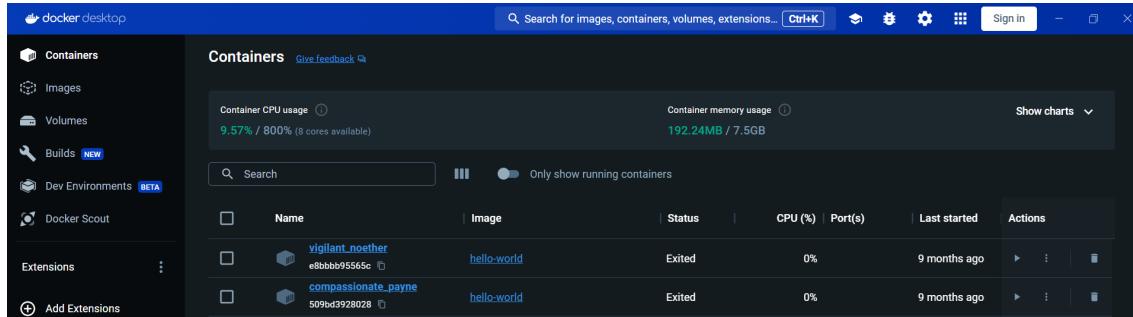


FIGURE 1 – Docker et Docker Compose

- Go

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ go version
go version go1.20.5 linux/amd64
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 2 – Go version

- Fabric Samples, Binaries et Docker Images

On a téléchargé la dernière version d'échantillons de fabric et d'images Docker avec cette commande

```
curl -sSL https://bit.ly/2ysb0FE | bash -s
```

```
hamidou@Hamidou-Ba:~/go/src/github.com$ curl -sSL https://bit.ly/2ysb0FE | bash -s
Clone hyperledger/fabric-samples repo
==== Cloning hyperledger/fabric-samples repo
Cloning into 'fabric-samples'...
remote: Enumerating objects: 14487, done.
remote: Counting objects: 100% (153/153), done.
remote: Compressing objects: 100% (99/99), done.
Receiving objects: 54% (7823/14487), 6.45 MiB | 2.32 MiB/s
```

FIGURE 3 – Hyperledger Fabric

Une fois ces éléments installés, il faut configurer un réseau blockchain de test en utilisant les scripts fournis dans Fabric Samples et nos propres scripts.

## 8 Configuration du Réseau Hyperledger Fabric

Avant de lancer une transaction, nous devons configurer et démarrer le réseau Hyperledger Fabric.

### 8.1 Démarrer le Réseau

Exécutez la commande suivante pour démarrer le réseau :

```
./network.sh up
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ sudo ./network.sh up
[sudo] password for hamidou:
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=v2.5.10
DOCKER_IMAGE_VERSION=v2.5.10
/home/hamidou/go/src/github.com/fabric-samples/test-network/..../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
[+] Running 7/7
  ✓ Network fabric_test          Created      0.4s
  ✓ Volume "compose_orderer.example.com" Created      0.1s
  ✓ Volume "compose_peer0.org1.example.com" Created      0.0s
  ✓ Volume "compose_peer0.org2.example.com" Created      0.0s
  ✓ Container peer0.org2.example.com   Started      0.8s
  ✓ Container orderer.example.com    Started      0.8s
  ✓ Container peer0.org1.example.com   Started      0.8s
CONTAINER ID  IMAGE           COMMAND      CREATED      STATUS      PORTS
S             7181d16b35f5  hyperledger/fabric-peer:latest "peer node start"  3 seconds ago  Up Less than a second  0.0.

```

FIGURE 4 – Démarrage du réseau

Vérifiez que les conteneurs sont bien en cours d'exécution :

```
docker ps
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ docker ps
CONTAINER ID  IMAGE           COMMAND      CREATED      STATUS      PORTS
              NAMES
7181d16b35f5  hyperledger/fabric-peer:latest "peer node start"  4 minutes ago  Up 4 minutes  0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp
e4415cccc9baa  hyperledger/fabric-peer:latest "peer node start"  4 minutes ago  Up 4 minutes  0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp
bfe6696ebb86   hyperledger/fabric-orderer:latest "orderer"      4 minutes ago  Up 4 minutes  0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 5 – Conteneurs en cours d'exécution

## 8.2 Créer un Channel

Créez un channel en exécutant :

```
sudo ./network.sh createChannel
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ sudo ./network.sh createChannel
[sudo] password for hamidou:
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
Bringing up network
LOCAL_VERSION=v2.5.10
DOCKER_IMAGE_VERSION=v2.5.10
[+] Running 3/0 :
  ✓ Container peer0.org1.example.com  Running          0.0s
  ✓ Container orderer.example.com    Running          0.0s.
  ✓ Container peer0.org2.example.com  Running          0.0s
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
7181d16b3f5f      hyperledger/fabric-peer:latest   "peer node start"   56 minutes ago   Up 56 minutes   0.0.0.0:9051→9051/tcp, 7051/tcp, 0.0.0.0:9
94415cc9bba      hyperledger/fabric-peer:latest   "peer node start"   56 minutes ago   Up 56 minutes   0.0.0.0:7051→7051/tcp, 0.0.0.0:9444→9444
/tcp              bfe6696eb86      hyperledger/fabric-orderer:latest "orderer"         56 minutes ago   Up 56 minutes   0.0.0.0:7050→7050/tcp, 0.0.0.0:7053→7053
/tcp              0.0.0.0:9413→9413/tcp   orderer.example.com
{"  
  "Admins": [{"  
    "mod_policy": {"  
      "policy": null,  
      "version": "0"  
    },  
    "Endorsement": {"  
      "mod_policy": {"  
        "policy": null,  
        "version": "0"  
      },  
      "version": "0"  
    },  
    "Readers": [{"  
      "mod_policy": {"  
        "policy": null,  
        "version": "0"  
      },  
      "Writers": [{"  
        "mod_policy": {"  
          "policy": null,  
          "version": "0"  
        },  
        "version": "0"  
      }]  
    },  
    "values": [{"  
      "MSP": {"  
        "mod_policy": {"  
          "policy": null,  
          "version": "0"  
        },  
        "value": null,  
        "version": "0"  
      }]  
    }],  
    "policies": {},  
    "values": {}  
  },  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  
    "version": "0"  
  }],  
  "mod_policy": {"  
    "policy": null,  
    "version": "0"  
  },  
  "policies": {},  
  "values": {}  
},  
  {"  


```

FIGURE 6 – Channel

## 9 Structure du Projet

Dans le répertoire `fabric-samples`, nous avons créé un dossier nommé `titre-foncier-chaincode` contenant les fichiers suivants :

- `chaincode`
  - `go.mod` : Dépendances du projet.
  - `go.sum` : Hash des dépendances.
  - `main.go` : Code principal du chaincode.
- `scripts`
  - `setOrg1Env.sh` : Script de configuration pour l'Organisation 1.
  - `setOrg2Env.sh` : Script de configuration pour l'Organisation 2.

## 10 Fichiers

### 10.1 Fichier `go.mod`

```
/fabric-samples/titre-foncier-chaincode/chaincode/go.mod
```

```
/fabric-samples/titre-foncier-chaincode/chaincode/go.sum
```

Après avoir écrit le fichier `go.mod`, exécutez la commande suivante pour télécharger les dépendances :

```
go mod tidy
```

Cette commande télécharge les packages manquants et met à jour le fichier `go.sum`.

```

root@Hamidou-Ba:/home/hamidou/go/src/github.com/fabric-samples/chaincode/land-title# go mod tidy
go: finding module for package github.com/hyperledger/fabric-contract-api-go/contractapi
go: downloading github.com/hyperledger/fabric-contract-api-go v1.2.2
go: found github.com/hyperledger/fabric-contract-api-go/contractapi in github.com/hyperledger/fabric-contract-api-go v1.2.2
go: downloading github.com/hyperledger/fabric-chaincode-go v0.0.0-20230731094759-d626e9ab09b9
go: downloading github.com/hyperledger/fabric-protos-go v0.3.0
go: downloading github.com/stretchr/testify v1.10.0
go: downloading github.com/xipipuu/gojonschema v1.2.0
go: downloading github.com/go-openapi/spec v0.21.0
go: downloading github.com/gobuffalo/packr v1.38.1
go: downloading github.com/golang/protobuf v1.5.3
go: downloading google.golang.org/grpc v1.67.8
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading google.golang.org/protobuf v1.36.1
go: downloading github.com/xipipuu/gojsonreference v0.0.0-20180127040603-bd5ef7bd5415
go: downloading github.com/gobuffalo/envd v1.10.2
go: downloading github.com/gobuffalo/middleware v1.0.2
go: downloading gopkg.in/yaml.v2 v2.4.1
go: downloading github.com/google/go-cmp v0.5.5
go: downloading github.com/go-openapi/jsonpointer v0.21.0
go: downloading github.com/go-openapi/jsonreference v0.21.0
go: downloading github.com/go-openapi/swag v0.23.0
go: downloading golang.org/x/net v0.28.0
go: downloading google.golang.org/genproto/googleapis/rpc v0.0.0-20240814211410-ddb44dafa142

```

FIGURE 7 – Télécharger les packages manquants

### 10.3 Fichier main.go

```
/fabric-samples/titre-foncier-chaincode/chaincode/main.go
```

### 10.4 Fichier setOrg1Env.sh

```
/fabric-samples/titre-foncier-chaincode/scripts/setOrg1Env.sh
```

### 10.5 Fichier setOrg2Env.sh

```
/fabric-samples/titre-foncier-chaincode/scripts/setOrg2Env.sh
```

### 10.6 Appliquer les Variables d’Environnement et affichez le contenu du fichier de configuration pour Organisation 1

Chargez les scripts de configuration des organisations :

```
source ../titre-foncier-chaincode/scripts/setOrg1Env.sh
source ../titre-foncier-chaincode/scripts/setOrg2Env.sh
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ source ../titre-foncier-chaincode/scripts/setOrg1Env.sh
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ echo $CORE_PEER_TLS_ENABLED
echo $CORE_PEER_LOCALMSPID
echo $CORE_PEER_TLS_ROOTCERT_FILE
echo $CORE_PEER_MSPCONFIGPATH
echo $CORE_PEER_ADDRESS
true
Org1MSP
/home/hamidou/go/src/github.com/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
/home/hamidou/go/src/github.com/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
localhost:7051

```

FIGURE 8 – Variables d’Environnement

## 11 Déploiement du Chaincode

Exécutez la commande suivante pour déployer le chaincode sur le réseau :

```
./network.sh deployCC -ccn titre-foncier -ccp ../titre-foncier-chaincode/chaincode
-ccl go
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ ./network.sh deployCC -ccn titre-foncier -ccp ../titre-foncier-chaincode/chaincode -ccl go
Using docker and docker-compose
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: titre-foncier
- CC_SRC_PATH: ../titre-foncier-chaincode/chaincode
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: auto
- CC_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: ''
- MAX_RETRY: 5
- VERBOSE: false
executing with the following
- CC_NAME: titre-foncier
- CC_SRC_PATH: ../titre-foncier-chaincode/chaincode
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
Rendering Go dependencies at ../titre-foncier-chaincode/chaincode
~/go/src/github.com/fabric-samples/titre-foncier-chaincode ~/go/src/github.com/fabric-samples/test-network
go: downloading github.com/go-openapi/spec v0.20.9
go: downloading google.golang.org/protobuf v1.31.0
go: downloading google.golang.org/grpc v1.59.0
go: downloading github.com/go-openapi/jsonreference v0.20.2
go: downloading github.com/go-openapi/openapitools v0.20.0
go: downloading github.com/golang/protobuf v0.22.4
go: downloading google.golang.org/genproto/googleapis/rpc v0.0.0-20230303175426-d783a09b4405
go: downloading google.golang.org/x/net v0.17.0
go: downloading google.golang.org/x/sys v0.14.0
go: downloading google.golang.org/x/mod v0.14.0
go: downloading google.golang.org/x/text v0.14.0
go: mkdir /home/hamidou/go/src/github.com/fabric-samples/titre-foncier-chaincode/vendor: permission denied
~/go/src/github.com/fabric-samples/test-network
2025-02-18 21:48:24.688 +0000 UTC [chaincodeCmd] ClientWait → txid [4762884b9bf52c47d50409ab2914142279d838a7295155342e89148446351aecd] committed with status (VALID) at localhost:7051
2025-02-18 21:48:24.706 +0002 [chaincodeCmd] ClientWait → txid [4762884b9bf52c47d50409ab2914142279d838a7295155342e89148446351acc] committed with status (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name titre-foncier
+ res=0
Committed chaincode definition for chaincode 'titre-foncier' on channel 'mychannel':
Version: 1, Sequence: 1, Endorsement Plugin: esc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle.chaincode querycommitted --channelID mychannel --name titre-foncier
+ res=0
Committed chaincode definition for chaincode 'titre-foncier' on channel 'mychannel':
Version: 1, Sequence: 1, Endorsement Plugin: esc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Chaincode definition successful on peer0.org2 on channel 'mychannel'
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ 

```

FIGURE 9 – Déploiement

## 12 Initialisation et Transactions

### 12.1 Initialisation du Chaincode

Après le déploiement, initialisez le chaincode :

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode invoke \
-o localhost:7050 \
--tls \
--cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" \
-C mychannel \
-n titrefoncier \
--peerAddresses localhost:7051 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" \
--peerAddresses localhost:9851 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" \
-c '{"function":"InitLedger","Args":[]}'
2025-02-18 23:41:08.901 +0000 UTC [chaincodeCmd] chaincodeInvokeOrQuery → Chaincode invoke successful. result: status:200
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ 

```

FIGURE 10 – Inialiser

### 12.2 Lecture d'un Titre Foncier

Vérifiez si l'initialisation a réussi en lisant le titre foncier 1 :

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode query -C mychannel -n titrefoncier -c '{"function":"LireTitreFoncier","Args":["1"]}' \
{"id": "1", "proprietaire": "Alice", "description": "Maison à Paris"} \
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ 

```

FIGURE 11 – Titre foncier 1

### 12.3 Cr ation d'une Nouvelle Transaction

Ajoutez un titre foncier :

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode invoke \
-o localhost:7050 \
--ordererTLSHostnameOverride orderer.example.com \
--tls \
--cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" \
-G mychannel \
-n titre-foncier \
-peerAddresses localhost:7051 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" \
-peerAddresses localhost:9051 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" \
-c '{"function":"CreerTitreFoncier","Args":["3","Hamidou Woury Ba","Appartement à Grand-Dakar"]}' \
2025-02-18 23:46:30.292 GMT 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery → Chaincode invoke successful. result: status:200
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 12 – Titre foncier 3

Vérifiez si l'ajout a été effectué :

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode query -C mychannel -n titre-foncier -c '{"function":"LireTitreFoncier","Args":["3"]}' \
{"id": "3", "proprietaire": "Hamidou Woury Ba", "description": "Appartement à Grand-Dakar"} \
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 13 – Titre foncier 3 Verification

## 12.4 Expliquer comment les smartcontracts sont soumis à Hyperledger

- Les smart contracts (chaincodes) dans Hyperledger Fabric sont soumis via un processus structuré : écriture, empaquetage, installation, approbation, engagement et test.
- Ils sont essentiels pour définir la logique métier et interagir avec le ledger.
- Le cycle de vie des chaincodes permet de gérer leur déploiement et leurs mises à jour de manière sécurisée.

## 12.5 Modification d'un Titre Foncier

Modifiez la valeur d'un élément (Propriétaire et Description) :

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode invoke \
-o localhost:7050 \
--ordererTLSHostnameOverride orderer.example.com \
--tls \
--cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" \
-G mychannel \
-n titre-foncier \
-peerAddresses localhost:7051 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" \
-peerAddresses localhost:9051 \
--tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" \
-c '{"function":"ModifierTitreFoncier","Args":["3","Nouveau Propriétaire","Nouvelle Description"]}' \
2025-02-19 00:07:25.004 GMT 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery → Chaincode invoke successful. result: status:200
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode query -C mychannel -n titre-foncier -c '{"function":"LireTitreFoncier","Args":["3"]}' \
{"id": "3", "proprietaire": "Nouveau Propriétaire", "description": "Nouvelle Description"} \
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 14 – Modifier Titre foncier 3

## 12.6 Valider que le ledger a bien mis à jour la donnée et a créer une nouvelle transaction de MAJ

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode query -C mychannel -n titre-foncier -c '{"function":"GetHistoryForTitreFoncier","Args":["3"]}' \
[{"txId": "76d72aa158d5c97f1d0dd078c1d1250de685bf7070fdad4122f0811a9a28d4", "timestamp": "2025-02-19T00:07:24Z", "isDelete": false, "value": {"id": "3", "proprietaire": "Nouveau Propriétaire", "description": "Nouvelle Description"}}, {"txId": "b8885dd4c2ccdb16efc56hd699516965efal696ba44d3e9f25cc61324170c49", "timestamp": "2025-02-19T00:07:01Z", "isDelete": false, "value": {"id": "3", "proprietaire": "Hamidou Woury Ba", "description": "Appartement à Grand-Dakar"}]] \
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 15 – Ledger mis à jour

## 13 Configuration de IPFS

### 13.1 Téléchargement et installation d'IPFS

Pour commencer, nous téléchargeons IPFS depuis le site officiel. Une fois le fichier téléchargé, nous le décompressons à l'aide de la commande tar. Ces étapes sont illustrées ci-dessous :

```

hamidou@Hamidou-Ba:~$ wget https://dist.ipfs.io/go-ipfs/v0.9.0/go-ipfs_v0.9.0_linux-amd64.tar.gz
--2025-02-19 10:31:49-- https://dist.ipfs.io/go-ipfs/v0.9.0/go-ipfs_v0.9.0_linux-amd64.tar.gz
Resolving dist.ipfs.io (dist.ipfs.io) ... 209.94.90.1, 2602:fea2:2::1
Connecting to dist.ipfs.io (dist.ipfs.io)|209.94.90.1|:443... connected.
HTTP request sent, awaiting response ... 301 Moved Permanently
Location: https://dist.ipfs.tech/go-ipfs/v0.9.0/go-ipfs_v0.9.0_linux-amd64.tar.gz [following]
--2025-02-19 10:31:49-- https://dist.ipfs.tech/go-ipfs/v0.9.0/go-ipfs_v0.9.0_linux-amd64.tar.gz
Resolving dist.ipfs.tech (dist.ipfs.tech) ... 209.94.78.1, 2602:fea2:3::1
Connecting to dist.ipfs.tech (dist.ipfs.tech)|209.94.78.1|:443... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 25946602 (25M) [application/gzip]
Saving to: 'go-ipfs_v0.9.0_linux-amd64.tar.gz'

go-ipfs_v0.9.0_linux-amd64.tar.gz      100%[=====]   24.74M  211KB/s    in 3m 22s
2025-02-19 10:35:24 (126 KB/s) - 'go-ipfs_v0.9.0_linux-amd64.tar.gz' saved [25946602/25946602]

hamidou@Hamidou-Ba:~$ 

```

FIGURE 16 – Téléchargement et installation d’IPFS

```

hamidou@Hamidou-Ba:~$ tar -xvf go-ipfs_v0.9.0_linux-amd64.tar.gz
go-ipfs/LICENSE
go-ipfs/LICENSE-APACHE
go-ipfs/LICENSE-MIT
go-ipfs/README.md
go-ipfs/install.sh
go-ipfs/api/
hamidou@Hamidou-Ba:~$ ls
Crypto          'Crypto - Raccourci (1).lnk'  MonDossier  go-ipfs'           gol.23.6.linux-amd64.tar.gz  snap
'Crypto - Raccourci (2).lnk' 'Crypto - Raccourci.lnk'  exit        go-ipfs_v0.9.0_linux-amd64.tar.gz  install-fabric.sh
'Crypto - Raccourci (3).lnk'  Miniforge3-linux-x86_64.sh  go         gol.20.5.linux-amd64.tar.gz  miniforge3
hamidou@Hamidou-Ba:~$ cd go-ipfs
hamidou@Hamidou-Ba:~/go-ipfs$ ls
LICENSE LICENSE-APACHE LICENSE-MIT README.md install.sh ipfs
hamidou@Hamidou-Ba:~/go-ipfs$ sudo bash install.sh
[sudo] password for hamidou:
Moved ./ipfs to /usr/local/bin
hamidou@Hamidou-Ba:~/go-ipfs$ ipfs --version
ipfs version 0.9.0
hamidou@Hamidou-Ba:~/go-ipfs$ 

```

FIGURE 17 – Décompression du fichier IPFS

### 13.2 Installation du package go-ipfs-api

Pour interagir avec IPFS depuis un projet Go, nous installons le package **go-ipfs-api** en utilisant la commande suivante :

```
go get github.com/ipfs/go-ipfs-api
```

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/titre-foncier-chaincode$ go get github.com/ipfs/go-ipfs-api
go: downloading github.com/ipfs/go-ipfs-api v0.7.0
go: downloading github.com/ibox/boxo v0.12.0
go: downloading github.com/blang/semver/v4 v4.0.0
go: downloading github.com/libp2p/go-libp2p v8.26.3
go: downloading github.com/mitchellh/go-homedir v1.1.0
go: downloading github.com/multiformats/go-multipaddr v0.8.0
go: downloading github.com/multiformats/go-multiparse v0.2.0
go: downloading github.com/mr-tron/base58 v1.2.0
go: downloading github.com/multiformats/go-base32 v0.1.0
go: downloading github.com/multiformats/go-base36 v0.2.0
go: downloading github.com/multiformats/go-cid v0.4.1
go: downloading github.com/multiformats/go-multihash v0.2.3
go: downloading github.com/multiformats/go-varint v0.0.7
go: downloading github.com/libp2p/go-flow-metrics v0.1.0
go: downloading github.com/multiformats/go-multicodec v0.9.0
go: downloading github.com/multiformats/go-multistream v0.4.1
go: downloading github.com/dcrdc/dcdec/secp256kl/v4 v4.1.0
go: downloading github.com/minio/sha256-simd v1.0.0
go: downloading github.com/libp2p/go-buffer-pool v0.1.0
go: downloading github.com/crackcomm/go-gitignore v0.0.0-20170627025303-887ab5e44cc3

```

FIGURE 18 – Installation du package go-ipfs-api

### 13.3 Crédit d’un fichier sous IPFS

Une fois IPFS configuré, nous pouvons ajouter un fichier au réseau IPFS en utilisant la commande suivante

```
ipfs add add.txt
```

Cette commande génère un hash unique pour le fichier, qui est ensuite stocké sur le réseau IPFS. Le résultat de cette opération est illustré ci-dessous :

```

hamidou@Hamidou-Ba:~/IPFS$ nano add.txt
hamidou@Hamidou-Ba:~/IPFS$ ls
add.txt test.txt
hamidou@Hamidou-Ba:~/IPFS$ ipfs add add.txt
added QmULFEn4WhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4
22 B / 22 B [=====] 100.00%
hamidou@Hamidou-Ba:~/IPFS$ cat QmULFEn4WhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4
cat: QmULFEn4WhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4: No such file or directory
hamidou@Hamidou-Ba:~/IPFS$ ipfs cat QmULFEn4WhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4
Sauvegarder dans IPFS
hamidou@Hamidou-Ba:~/IPFS$ 

```

FIGURE 19 – Ajout fichier dans IPFS

## 14 Intégrer IPFS avec Hyperledger pour le stockage des documents.

### 14.1 Modifications du fichier main.go

Pour intégrer IPFS avec Hyperledger, nous avons modifié le fichier main.go en ajoutant un champ DocumentHash dans la structure TitreFoncier. Ce champ stocke le hash généré lors de l'ajout d'un document sur IPFS.

```

type TitreFoncier struct {
    ID          string `json:"id"`
    Proprietaire string `json:"proprietaire"`
    Description string `json:"description"`
    DocumentHash string `json:"documentHash"`
}

```

#### Méthode CreerTitreFoncier et ModifierTitreFoncier

Les méthodes **CreerTitreFoncier** et **ModifierTitreFoncier** ont été modifiées pour inclure le champ **DocumentHash** lors de la création et de la modification d'un titre foncier. Cette modification permet d'associer un hash de document à chaque titre foncier, garantissant ainsi l'intégrité et la traçabilité des documents associés.

- **CreerTitreFoncier** : Cette méthode permet désormais de créer un titre foncier avec les informations suivantes : **ID**, **Propriétaire**, **Description**, et un nouveau champ **DocumentHash** pour lier un document à ce titre foncier.
- **ModifierTitreFoncier** : La méthode de modification permet maintenant de mettre à jour le champ **DocumentHash** afin que les modifications apportées à un titre foncier incluent également la mise à jour du hash associé au document.

/fabric-samples/titre-foncier-chaincode/chaincode/main.go

### 14.2 Création d'un titre foncier avec IPFS

#### 14.2.1 Cas 1 : Hash valide

Lors de la création d'un titre foncier avec un hash valide, le document est correctement associé au titre foncier. Voici un exemple de commande et le résultat obtenu :

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode invoke \
    -o localhost:7050 \
    --ordererTLSHostnameOverride orderer.example.com \
    --tls \
    --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.e
xample.com-cert.pem" \
    -C mychannel \
    -n titre-foncier \
    --peerAddresses localhost:7051 \
    --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
" \
    --peerAddresses localhost:9051 \
    --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
" \
    -c '{"function":"CreerTitreFoncier","Args":["8", "Abdoulaye Wade", "Villa à Ziguinchor", "QmULFEen4WVhnxZ8DM8y97RPzJ.
3xhSUsG64eo2ieKGgzSQ4"]}' \
2025-02-28 20:54:43.983 GMT 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery → Chaincode invoke successful. result: s
tatus:200
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode query -C mychannel -n titre-foncier \
    -c '{"function":"LireTitreFoncier","Args":["8"]}' \
    {"id": "8", "proprietaire": "Abdoulaye Wade", "description": "Villa à Ziguinchor", "documentHash": "QmULFEen4WVhnxZ8DM8y97RPz
J3xhSUsG64eo2ieKGgzSQ4"} \
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 20 – Titre foncier avec hash

#### 14.2.2 Cas 1 : Hash invalide

Si le hash fourni n'existe pas sur IPFS, la création du titre foncier échoue, et un message d'erreur est retourné. Voici un exemple :

```

hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$ peer chaincode invoke \
    -o localhost:7050 \
    --ordererTLSHostnameOverride orderer.example.com \
    --tls \
    --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.e
xample.com-cert.pem" \
    -C mychannel \
    -n titre-foncier \
    --peerAddresses localhost:7051 \
    --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
" \
    --peerAddresses localhost:9051 \
    --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
" \
    -c '{"function":"CreerTitreFoncier","Args":["9", "Salif", "Une nouvelle Villa", "QmULFEen4WVhnxZ8DM8y97RPzJ3xhSUsG64.
eo2ieKGgzSQ4"]}' \
Error: endorsement failure during invoke. response: status:500 message:"erreur lors de la v\303\251rification du docu
ment sur IPFS: erreur IPFS: 500 Internal Server Error"
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network$
```

FIGURE 21 – Titre foncier avec hash incorrect

On voit bien qu'on nous dit erreur lors de la création du document sur IPFS.

## 15 Application cliente

Cette application est un **client web** permettant de gérer des **titres fonciers** à travers une solution basée sur **Hyperledger Fabric**. Elle permet aux utilisateurs d'ajouter, de modifier et de consulter des titres fonciers enregistrés sur la blockchain. En tant qu'application cliente, elle interagit avec un réseau blockchain privé géré par Hyperledger Fabric pour garantir la sécurité et l'intégrité des données.

## 16 Structure du Projet

Le projet est organisé comme suit :

```
titre-foncier-app/
|
+-- public/                      # Fichiers statiques (HTML, CSS, JS)
|   +-- index.html                # Page d'accueil, affiche la liste des titres fonciers
|   +-- ajout-titre.html          # Formulaire pour ajouter un nouveau titre foncier
|   +-- modifier-titre.html      # Formulaire pour modifier un titre foncier existant
|   +-- styles.css                # Styles CSS
|   +-- app.js                    # Script frontend pour gérer le formulaire
|
+-- server.js                   # Serveur Node.js pour gérer les requêtes
+-- enrollUser.js               # Script pour enregistrer un utilisateur sur Fabric
+-- connection.json             # Fichier de configuration pour se connecter à Fabric
+-- wallet/                      # Dossier pour stocker les identités
```

### 16.1 Explication des fichiers

#### Public Directory

Le répertoire **public** contient les fichiers statiques nécessaires à l'application web :

- **index.html** : Page d'accueil affichant la liste des titres fonciers existants.
- **ajout-titre.html** : Formulaire pour ajouter un nouveau titre foncier dans la blockchain.
- **modifier-titre.html** : Formulaire pour modifier un titre foncier existant.
- **styles.css** : Fichier CSS pour styliser l'application web.
- **app.js** : Script JavaScript qui interagit avec le frontend et gère la logique du formulaire.

#### Backend Files

Le répertoire principal contient les fichiers backend suivants :

- **server.js** : Fichier principal qui lance un serveur Node.js et gère les requêtes HTTP pour interagir avec la blockchain.
- **enrollUser.js** : Ce script permet d'enregistrer un utilisateur sur Hyperledger Fabric et d'ajouter l'identité à un wallet local.
- **connection.json** : Ce fichier contient les informations nécessaires pour se connecter à l'instance Hyperledger Fabric, notamment les adresses des peers, les certificats TLS, etc.
- **wallet/** : Dossier où les identités des utilisateurs sont stockées.

### 16.2 Installation et Exécution

Pour installer et configurer l'application, voici les étapes à suivre :

```
npm init -y
npm install fabric-network
npm install fabric-ca-client
```

Ensuite, le fichier **connection.json** est copié depuis le répertoire de test du réseau Fabric :

```
cp /home/hamidou/go/src/github.com/fabric-samples/test-network/organizations/
    peerOrganizations/org1.example.com/connection-org1.json /home/hamidou/go/src/
    github.com/fabric-samples/titre-foncier-app/connection.json
```

## Exécution du Projet

L'exécution de l'application se fait en deux étapes :

1. Lancer le script `enrollUser.js` pour enregistrer un utilisateur sur Hyperledger Fabric :

```
node enrollUser.js
```

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/titre-foncier-app$ node enrollUser.js
Identité "user1" enregistrée avec succès dans le wallet.
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/titre-foncier-app$
```

FIGURE 22 – Résultat exécution

2. Démarrer le serveur Node.js avec `server.js` :

```
node server.js
```

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/titre-foncier-app$ node server.js
Serveur en écoute sur http://localhost:3000
-
```

FIGURE 23 – Résultat exécution

Une fois le serveur démarré, vous pouvez accéder à l'application en vous rendant à l'URL suivante dans votre navigateur :

<http://localhost:3000>

## 17 Fonctionnalités de l'Application

L'application permet d'afficher la liste des titres fonciers existants, d'ajouter de nouveaux titres, et de modifier des titres spécifiques. Voici un aperçu de la page d'accueil et des autres fonctionnalités.

### 17.1 Page d'Accueil

La page d'accueil affiche la liste des titres fonciers qui ont été ajoutés à la blockchain. Elle permet également d'ajouter de nouveaux titres via un bouton "Ajouter" et de modifier les titres existants via des liens spécifiques pour chaque titre.

The screenshot shows a web application titled "Gestion des Titres Fonciers". At the top, there is a button labeled "Ajouter un titre foncier". Below it, a section titled "Liste des Titres Fonciers" contains a green button labeled "Rafraîchir la liste". A table lists eight land titles with columns for ID, Propriétaire, Description, Document Hash, and Actions (each with a "Modifier" link). The data is as follows:

ID	Propriétaire	Description	Document Hash	Actions
1	Alice	Maison à Paris	QmVNqRfyeQAiiinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
2	Bob	Appartement à Lyon		<a href="#">Modifier</a>
3	Modou Coumba	Villa à Sally	QmYGY7iljPNGYQFSJsnclCDkZUGXayz7kfftBt14GqBY5u	<a href="#">Modifier</a>
4	Hamidou Woury	Villa	QmYGY7iljPNGYQFSJsnclCDkZUGXayz7kfftBt14GqBY5u	<a href="#">Modifier</a>
5	Salif	Villa à Las Vegas	QmVNqRfyeQAiiinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
6	Amath	Une Villa de Luxe	QmVNqRfyeQAiiinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
7	Mariama	Villa à California	Qmdzm26Qzm8XBIGkFTkQBeKJBq56kiJvNY22NvWYSTcWQF	<a href="#">Modifier</a>
8	Abdoulaye Wade	Villa à Ziguinchor	QmULFEn4WVhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4	<a href="#">Modifier</a>

FIGURE 24 – Page d'accueil affichant les titres fonciers

## 17.2 Ajouter un Nouveau Titre

Un utilisateur peut ajouter un nouveau titre foncier via un formulaire. Le formulaire permet d'entrer les détails du titre foncier, et une fois soumis, les données sont envoyées à la blockchain via Hyperledger Fabric.

The screenshot shows a form titled "Ajouter un Titre Foncier". It includes fields for "ID" (with value "9"), "Propriétaire" (with value "Mamadou Mactar"), "Description" (with value "Villa à Saint-Louis"), and "Document Hash" (with value "QmVNqRfyeQAiiinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN"). At the bottom, there is a green button labeled "Créer le titre foncier" and a link "← Retour à la liste".

FIGURE 25 – Formulaire d'ajout d'un titre foncier

Lorsqu'on clique sur **Créer le titre foncier**, un message de succès est affiché en bas. Voir sur l'image ci-dessous :

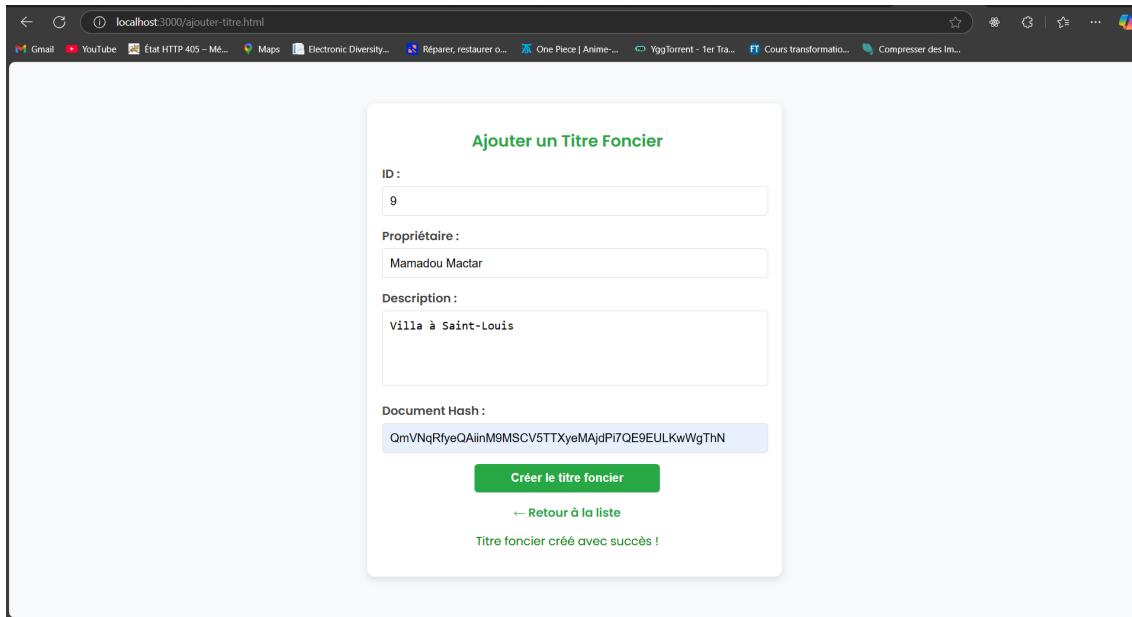


FIGURE 26 – Message indicant que l'ajout a réussi

Lorsqu'on clique sur **Retour à la liste**, on est redirigé vers la page d'accueil et on voit bien que le titre foncier a été créé.

The screenshot shows a table titled "Gestion des Titres Fonciers" listing nine land titles. The columns are:

ID	Propriétaire	Description	Document Hash	Actions
1	Alice	Maison à Paris	QmVNqRfyeQAlinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
2	Bob	Appartement à Lyon	QmYGe7ijPNQYQFSJsncuCDkZUGXayz7kfBt14GoBY5u	<a href="#">Modifier</a>
3	Modou Coumba	Villa à Sally	QmYGe7ijPNQYQFSJsncuCDkZUGXayz7kfBt14GoBY5u	<a href="#">Modifier</a>
4	Hamidou Woury	Villa	QmYGe7ijPNQYQFSJsncuCDkZUGXayz7kfBt14GoBY5u	<a href="#">Modifier</a>
5	Salif	Villa à Las Vegas	QmVNqRfyeQAlinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
6	Amath	Une Villa de Luxe	QmVNqRfyeQAlinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
7	Mariama	Villa à California	Qmdzm26Qzm8XBIGkFTkBeKJBq56kiJvNY22nWVYSTcWQF	<a href="#">Modifier</a>
8	Abdoulaye Wade	Villa à Ziguinchor	QmULFen4WVhnxZ8DM8y97RPzJ3xhSUSG64eo2ieKGgzSQ4	<a href="#">Modifier</a>
9	Mamadou Mactar	Villa à Saint-Louis	QmVNqRfyeQAlinM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>

FIGURE 27 – Retour à la page d'accueil

### 17.3 Modifier un Titre Foncier

L'application permet également de modifier un titre foncier existant en cliquant sur un lien "Modifier" associé à chaque titre. Un formulaire pré-rempli s'affiche alors pour permettre les modifications nécessaires.

On clique sur le lien **Modifier** du titre foncier créé. On est redirigé vers la page de modifier avec l'id du titre foncier que l'on veut modifier (ceci est visible dans l'url de la requête). Le formulaire de modification s'affiche avec les champs pré-remplis. Voir sur la figure ci-dessous :

FIGURE 28 – Formulaire de modification d'un titre foncier

Lorsqu'on clique sur **Modifier le titre foncier**, un message de succès est affiché en bas. Voir sur l'image ci-dessous :

FIGURE 29 – Message indicant que la modification a réussi

Lorsqu'on clique sur **Retour à la liste**, on est redirigé vers la page d'accueil et on voit bien que le titre foncier a été modifié.

Gestion des Titres Fonciers

Ajouter un titre foncier

Liste des Titres Fonciers

Rafraîchir la liste

ID	Propriétaire	Description	Document Hash	Actions
1	Alice	Maison à Paris	QmVNqRfyeQAiiM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
2	Bob	Appartement à Lyon	QmYGY7iljPNGYQFSJsnccuCDkZUGXayz7kftBtI4GqBY5u	<a href="#">Modifier</a>
3	Modou Coumba	Villa à Sally	QmYGY7iljPNGYQFSJsnccuCDkZUGXayz7kftBtI4GqBY5u	<a href="#">Modifier</a>
4	Hamidou Woury	Villa	QmYGY7iljPNGYQFSJsnccuCDkZUGXayz7kftBtI4GqBY5u	<a href="#">Modifier</a>
5	Salif	Villa à Las Vegas	QmVNqRfyeQAiiM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
6	Amath	Une Villa de Luxe	QmVNqRfyeQAiiM9MSCV5TTXyeMAjdPi7QE9EULKwWgThN	<a href="#">Modifier</a>
7	Mariama	Villa à California	Qmdzm26QZm8XBIGkFTkQBBeKJBq56kjvNY22NvWYSTcWQF	<a href="#">Modifier</a>
8	Abdoulaye Wade	Villa à Ziguinchor	QmULFen4WVhnxZ8DM8y97RPzJ3xhSUsG64eo2ieKGgzSQ4	<a href="#">Modifier</a>
9	Kalidou Koulibaly	Villa à Saint-Louis	Qmdzm26QZm8XBIGkFTkQBBeKJBq56kjvNY22NvWYSTcWQF	<a href="#">Modifier</a>

FIGURE 30 – Message indiquant que la modification a réussi

## 18 Tests du Système

### 18.1 Chargement de documents de tailles variées

L'objectif est d'évaluer la performance et la scalabilité du système en chargeant des fichiers de différentes tailles. Les tests sont réalisés avec des fichiers de tailles variées : 1MB, 10MB et 100MB. Ces fichiers sont utilisés pour mesurer le temps de traitement et la consommation mémoire du système.

#### 18.1.1 Méthodologie

- Test sur des fichiers de 1MB, 10MB et 100MB

```
hamidou@Hamidou-Ba:~/IPFS/Fichier$ dd if=/dev/urandom of=file1mb.txt bs=1M count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00731859 s, 143 MB/s
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ls
file1mb.txt
hamidou@Hamidou-Ba:~/IPFS/Fichier$ dd if=/dev/urandom of=file10mb.txt bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0344721 s, 304 MB/s
hamidou@Hamidou-Ba:~/IPFS/Fichier$ dd if=/dev/urandom of=file100mb.txt bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.499348 s, 210 MB/s
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ls
file100mb.txt file10mb.txt file1mb.txt
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ls -l
total 113664
-rw-r--r-- 1 hamidou hamidou 104857600 Mar 1 13:49 file100mb.txt
-rw-r--r-- 1 hamidou hamidou 10485760 Mar 1 13:49 file10mb.txt
-rw-r--r-- 1 hamidou hamidou 1048576 Mar 1 13:49 file1mb.txt
hamidou@Hamidou-Ba:~/IPFS/Fichier$
```

FIGURE 31 – Création des fichiers

```
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ls
file100mb.txt file10mb.txt file1mb.txt
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ipfs add file1mb.txt
added QmTwonFRj5FxVmocMJCMoFhjkadh8idHuzPC8acXdfkYN file1mb.txt
 1.00 MiB / 1.00 MiB [=====] 100.00%
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ipfs add file10mb.txt
added QmRpWMRsijuojujFAF4xUq3Twb1ZLthpcr2Suk966GJ521a file10mb.txt
 10.00 MiB / 10.00 MiB [=====] 100.00%
hamidou@Hamidou-Ba:~/IPFS/Fichier$ ipfs add file100mb.txt
added QmcVkg8XeAsg6uXvGvnM57Z3UdtsgXZkvsgM1RkV9fpugef file100mb.txt
 100.00 MiB / 100.00 MiB [=====] 100.00%
hamidou@Hamidou-Ba:~/IPFS/Fichier$
```

FIGURE 32 – Ajout des fichiers dans IPFS

- Format testé : .txt
- Mesure du temps de traitement et de la consommation mémoire
- Utilisation d'un script `executionTime-Memoire.sh` pour collecter les données, qui sont ensuite analysées via un script Python `executionTime-Memoire_results.py`

#### 18.1.2 Résultats

Les résultats du test sont présentés dans le tableau suivant :

Taille du fichier	Temps de traitement (s)	Consommation mémoire (MB)
1 MB	0.37	44.53
10 MB	1.88	44.43
100 MB	14.92	42.74

TABLE 1 – Résultats des tests de performance

```
hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network/Python$ python3 executionTime-Memoire_results.py
/home/hamidou/.local/lib/python3.10/site-packages/tensorflow/python/_pywrap_tensorflow.so: warning: libAves30, version 0.0.0 not found (or not built-in)
warning: while loading shared library tensorflow.python._pywrap_tensorflow: libAves30: cannot open shared object file: No such file or directory
/home/hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network/Python$ python3 executionTime-Memoire_results.py
/home/hamidou/.local/lib/python3.10/site-packages/tensorflow/python/_pywrap_tensorflow.so: warning: libAves30, version 0.0.0 not found (or not built-in)
warning: while loading shared library tensorflow.python._pywrap_tensorflow: libAves30: cannot open shared object file: No such file or directory
/home/hamidou@Hamidou-Ba:~/go/src/github.com/fabric-samples/test-network/Python$
```

FIGURE 33 – Résultat exécution

### 18.1.3 Graphiques des résultats

Pour mieux comprendre la relation entre la taille des fichiers, le temps de traitement et la consommation mémoire, nous avons généré les graphiques suivants.

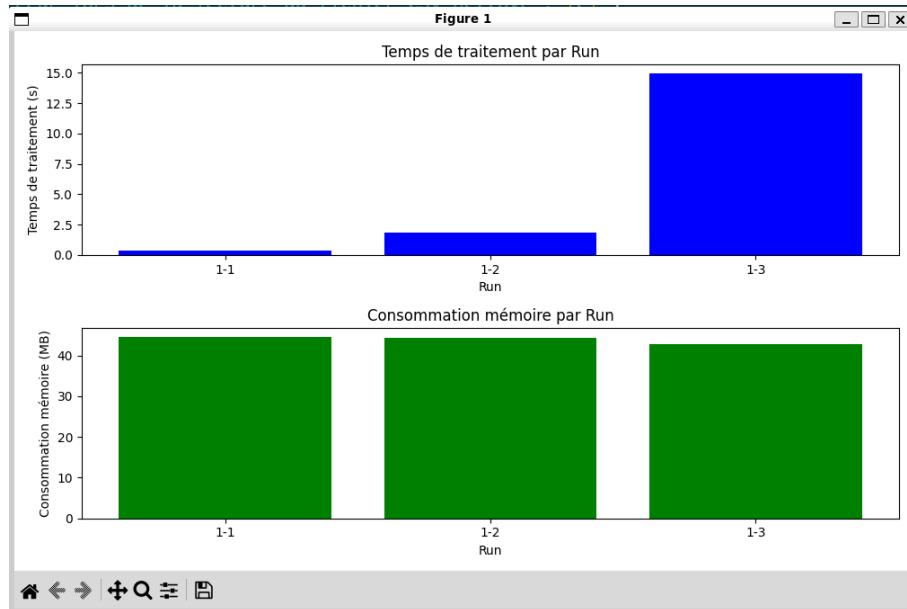


FIGURE 34 – Temps de traitement (en secondes) et Consommation mémoire (en MB) pour différentes tailles de fichier.

### 18.1.4 Interprétation des résultats

**Temps de traitement :** Le temps de traitement augmente significativement avec la taille du fichier :

- 1 Mo : 0.37 s
- 10 Mo : 1.88 s (5 fois plus long que 1 Mo)
- 100 Mo : 14.92 s (40 fois plus long que 1 Mo)

**Consommation mémoire :** La consommation mémoire reste relativement stable, variant entre 42.74 MB et 44.53 MB. Cela suggère que la mémoire utilisée ne dépend pas fortement de la taille des fichiers traités.

### 18.1.5 Conclusion

Les tests montrent que la taille des fichiers a un impact direct sur le temps de traitement, mais peu sur la consommation mémoire. Cela peut indiquer que le système est bien optimisé en termes de gestion mémoire, mais pourrait nécessiter des améliorations pour mieux gérer les fichiers de grande taille en termes de temps de traitement.

## 18.2 Simulation des accès concurrents

L'objectif est d'évaluer la résilience et la réactivité du système sous forte charge.

### 18.2.1 Installation d'Apache JMeter

Télécharger et installer Apache JMeter depuis le site officiel : <https://jmeter.apache.org/>. L'installation est simple et ne nécessite aucune configuration particulière.

### 18.2.2 Création du Test de Charge

La création d'un test de charge dans Apache JMeter consiste à simuler des utilisateurs accédant simultanément à votre application pour évaluer ses performances sous une charge élevée.

### 18.2.3 Configuration du Plan de Test

Le *plan de test* dans Apache JMeter est la première étape pour structurer notre simulation de charge. Nous avons configuré un *Thread Group* pour représenter un ensemble d'utilisateurs simulés accédant à l'application en même temps.

1. Dans la fenêtre principale de JMeter, faites un clic droit sur **Test Plan** et sélectionnez **Add -> Threads -> Thread Group**.
2. Dans le **Thread Group**, nous avons défini les paramètres suivants :
  - **Number of Threads (users)** : Nous avons simulé **50 utilisateurs** qui effectuent des requêtes simultanément.
  - **Ramp-Up Period** : Nous avons configuré une période de montée en charge de **10 secondes**, ce qui signifie que les 50 utilisateurs seront ajoutés progressivement pendant cette période.
  - **Loop Count** : Le nombre de fois que chaque utilisateur effectuera la requête a été défini sur **10**, soit chaque utilisateur enverra 10 requêtes pendant l'exécution du test.

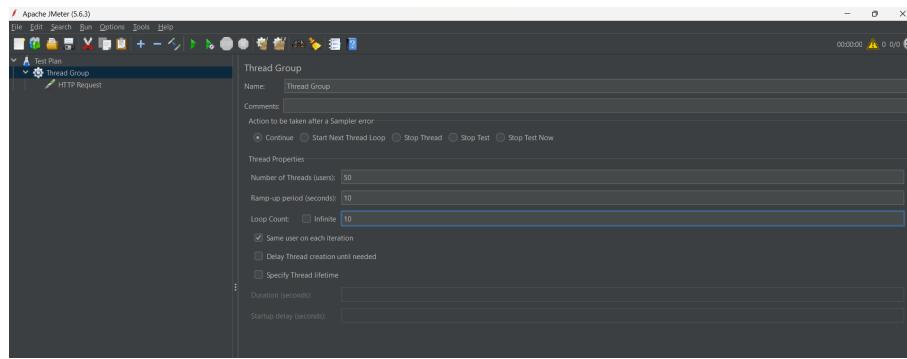


FIGURE 35 – Configuration du plan de test.

### 18.2.4 Ajout de Requêtes HTTP

Nous avons ajouté une requête HTTP pour l'endpoint `/listerTitresFonciers`, qui permet de récupérer la liste des titres fonciers depuis le système. Cette requête est envoyée par chaque utilisateur simulé.

1. Faites un clic droit sur le **Thread Group** et sélectionnez **Add -> Sampler -> HTTP Request**.
2. Dans la configuration de la requête HTTP, nous avons défini les paramètres suivants :
  - **Server Name or IP** : `localhost`, car l'application est déployée localement.
  - **Port Number** : `3030`, le port sur lequel l'application tourne.
  - **Method** : `GET`, car nous souhaitons récupérer des données de manière passive (sans modification côté serveur).
  - **Path** : `/listerTitresFonciers`, l'endpoint spécifique que nous testons.

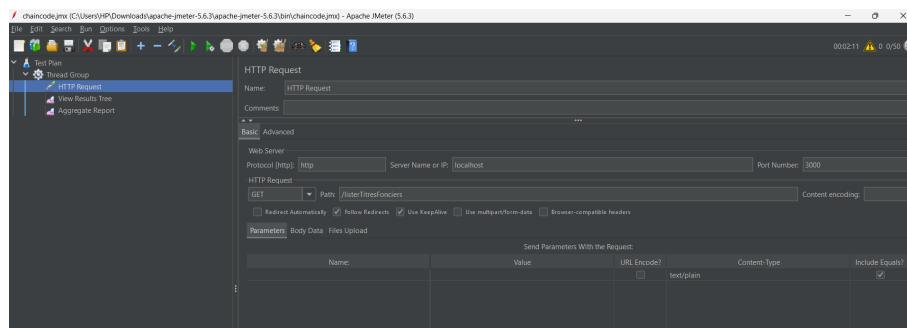


FIGURE 36 – Ajout de Requêtes HTTP

### 18.2.5 Ajout de l'Analyse des Résultats

Pour analyser les résultats du test de charge, nous avons ajouté plusieurs *Listeners*, qui sont des outils permettant de visualiser les données de performance pendant l'exécution du test.

1. On a ajouté un **Listener** comme **View Results Tree** pour observer chaque requête et réponse en détail pendant le test.
2. On a également ajouté un **Summary Report** pour obtenir une vue d'ensemble des performances du test, avec des statistiques telles que le temps moyen de réponse, le taux d'erreur, et la latence.

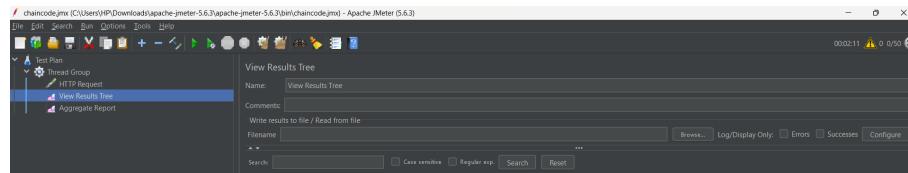


FIGURE 37 – Ajout de Requêtes HTTP

### 18.2.6 Exécution du Test de Charge

Après l'exécution du test de charge sur l'endpoint `/listerTitresFonciers`, nous avons analysé les performances à l'aide du rapport agrégé de JMeter. Les captures d'écran ci-dessous présentent les résultats obtenus :

Résultats dans **Views Results Tree** :

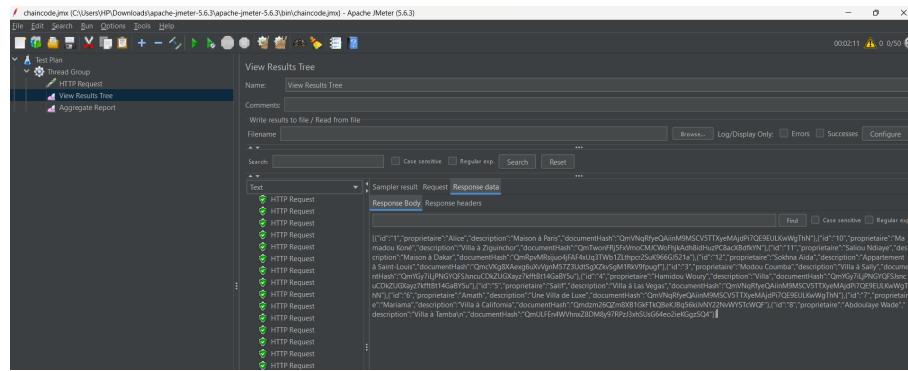


FIGURE 38 – Views Results Tree

#### Succès des Requêtes :

La capture d'écran du *View Results Tree* montre que toutes les requêtes HTTP ont abouti avec succès (icônes vertes). Cela indique que le serveur a bien répondu aux requêtes et que l'endpoint fonctionne correctement sans erreurs majeures.

#### Stabilité et Scalabilité :

Le test a permis d'évaluer la stabilité de l'application face à une montée en charge progressive sur une période de **10 secondes**. L'absence d'échecs dans la capture d'écran suggère que le système a bien géré les **500 requêtes** (50 utilisateurs × 10 requêtes chacun).

#### Détails des Données Renvoyées :

Dans l'onglet *Response Data*, les résultats montrent que l'endpoint retourne une liste de titres fonciers sous format **JSON**, contenant des attributs comme **id**, **propriétaire**, **description** et **documentHash**. Cela confirme que l'API fonctionne comme attendu.

## Résultats dans **Aggregate Report** :



FIGURE 39 – Aggregate Report

Les résultats obtenus indiquent les métriques suivantes :

- **Nombre total de requêtes exécutées** : 500.
- **Temps de réponse moyen** : 11 979 ms.
- **Médiane du temps de réponse** : 12 202 ms.
- **Temps de réponse au 90<sup>ème</sup> percentile** : 14 462 ms.
- **Temps de réponse au 95<sup>ème</sup> percentile** : 15 701 ms.
- **Temps de réponse au 99<sup>ème</sup> percentile** : 21 247 ms.
- **Temps de réponse minimum** : 2 191 ms.
- **Temps de réponse maximum** : 23 033 ms.
- **Pourcentage d'erreurs** : 0.00%.
- **Débit (Throughput)** : 3.8 requêtes par seconde.
- **Données reçues** : 5.98 KB/sec.
- **Données envoyées** : 0.73 KB/sec.

L'absence d'erreurs (0.00%) indique que toutes les requêtes ont été exécutées avec succès. Toutefois, le temps de réponse moyen de 11 979 ms montre que l'endpoint présente un certain niveau de latence sous charge.

### 18.2.7 Conclusion

Cette simulation d'accès concurrents a permis d'évaluer la capacité de notre application à supporter une charge élevée. En utilisant Apache JMeter, nous avons pu tester la résilience et la réactivité du système, identifier les points faibles, et proposer des améliorations pour garantir une meilleure performance sous charge.

## 19 Analyse des Données

### 19.1 Mesurer le temps de réponse

L'évaluation du temps de réponse a déjà été réalisée dans la section "Simulation des accès concurrents pour évaluer la résilience et la réactivité". En utilisant JMeter, nous avons simulé 50 utilisateurs envoyant chacun 10 requêtes à l'endpoint /listerTitresFonciers. Les résultats obtenus via **Aggregate Report** indiquent les temps de réponse moyens, médians et maximaux des requêtes HTTP.

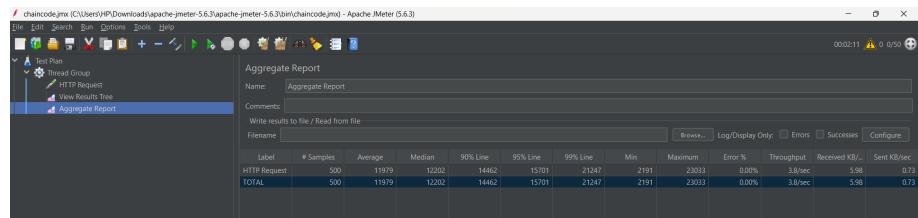


FIGURE 40 – Aggregate Report

## 19.2 Sécurité

Avant la mise en place du protocole HTTPS, notre serveur fonctionnait uniquement sous HTTP, ce qui exposait les échanges de données à des risques d'interception (attaque Man-in-the-Middle). Afin de renforcer la sécurité des échanges entre le client et le serveur, nous avons configuré notre serveur pour utiliser HTTPS en générant un certificat SSL/TLS auto-signé.

### 19.2.1 Génération du certificat SSL/TLS

La première étape a consisté à générer une paire de clés privée et publique ainsi qu'un certificat auto-signé en utilisant OpenSSL. La commande suivante a été exécutée dans le terminal :

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout key.pem -out cert.pem
```

Lors de l'exécution, plusieurs informations ont été demandées, notamment :

- **Country Name (2 letter code)** : SN
- **State or Province Name** : Senegal
- **Locality Name** : Dakar
- **Organization Name** : ESP
- **Organizational Unit Name** : sn
- **Common Name** : localhost

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
  
Country Name (2 letter code) [AU]:SN  
State or Province Name (full name) [Some-State]:Senegal  
Locality Name (eg, city) []:Dakar  
Organization Name (eg, company) [Internet Widgets Pty Ltd]:ESP  
Organizational Unit Name (eg, section) []:sn  
Common Name (e.g. server FQDN or YOUR name) []:localhost  
Email Address []:hamidouwouryba@esp.sn
```

FIGURE 41 – Génération des clés

### 19.2.2 Modification du serveur pour utiliser HTTPS

Une fois le certificat généré, le fichier `server.js` a été modifié pour activer HTTPS en intégrant le module `https` de Node.js. Le code se trouve dans le dépôt github sur le chemin :

```
/fabric-samples/titre-foncier-app/server.js
```

### 19.2.3 Connexion sécurisée et validation

Après le redémarrage du serveur, nous avons accédé à l'application via l'URL `https://localhost:3443` car on a changé le port de 3000 à 3443 dans le code. Un avertissement de sécurité est affiché sur le navigateur car le certificat est auto-signé, mais la connexion est bien établie sous HTTPS.

ID	Propriétaire	Description	Document Hash	Actions
1	Alice	Maison à Paris	QmVNqRfyeQAinM9MSCV5TTXyeMAjdPi7QE9EULkwWgThN	<a href="#">Modifier</a> <a href="#">Supprimer</a>
10	Mamadou Koné	Villa à Ziguinchor	QmTwonfRj5xv/moCMjCWoFhjkAchd8idHuZpc8acX8dfkYN	<a href="#">Modifier</a> <a href="#">Supprimer</a>
11	Saliou Ndiaye	Maison à Dakar	QmRpvmRsijuo4jFA4xUqz3TwbIzLthpcr2SuK966gJ521a	<a href="#">Modifier</a> <a href="#">Supprimer</a>
13	Abdoulaye Wade	Une nouvelle description	QmULfEn4WVhmz28DM8y97RPzJ3xhSUs7G64eo2ieK0gZSQ4	<a href="#">Modifier</a> <a href="#">Supprimer</a>
3	Modou Coumba	Villa à Sally	QmYgy7iljPNgyQFSJsnclCDkzUGXayz7kfftBt4GqBY5u	<a href="#">Modifier</a> <a href="#">Supprimer</a>
4	Hamidou Woury	Villa	QmYgy7iljPNgyQFSJsnclCDkzUGXayz7kfftBt4GqBY5u	<a href="#">Modifier</a> <a href="#">Supprimer</a>
5	Salif	Villa à Las Vegas	QmVNqRfyeQAinM9MSCV5TTXyeMAjdPi7QE9EULkwWgThN	<a href="#">Modifier</a> <a href="#">Supprimer</a>
6	Amath	Une Villa de Luxe	QmVNqRfyeQAinM9MSCV5TTXyeMAjdPi7QE9EULkwWgThN	<a href="#">Modifier</a> <a href="#">Supprimer</a>
7	Mariama	Villa à California	Qmdzm28Qz8m8XBIGkFTkQ8ekJBq58kJuNY22NwYSTcWQF	<a href="#">Modifier</a> <a href="#">Supprimer</a>
8	Abdoulaye Wade	Villa à Tambac	QmULfEn4WVhmz28DM8y97RPzJ3xhSUsG64eo2ieK0gZSQ4	<a href="#">Modifier</a> <a href="#">Supprimer</a>

FIGURE 42 – Visualisation du résultat

On voit bien dans l'url **https**, grâce à cette configuration, les échanges de données entre le client et le serveur sont désormais chiffrés, garantissant une meilleure confidentialité et protection contre les attaques réseau.

### 19.3 Permanence des données

La permanence des données a également été évaluée lors de la simulation des accès concurrents. Dans JMeter (Views Results Tree, onglet Response Data), nous avons vérifié que les données retournées restent cohérentes malgré la charge élevée et qu'il n'y a pas de corruption des informations.

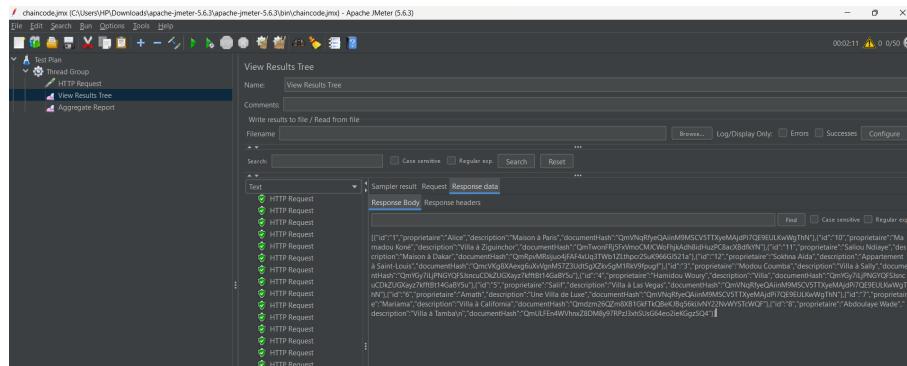


FIGURE 43 – Views Results Tree

De plus, nous avons redémarré le serveur et relancé des requêtes pour confirmer que les données étaient toujours accessibles et intactes après une interruption.

## 20 Calculer les coûts opérationnels comparés à une solution on-chain

Pour évaluer les coûts opérationnels de notre solution, nous avons comparé une approche **on-chain** utilisant **Hyperledger Fabric** avec une approche **off-chain**, où les documents sont stockés via **IPFS**.

### Coûts de l'Infrastructure

- On-Chain** : L'utilisation de Hyperledger Fabric nécessite le déploiement et la gestion de plusieurs noeuds, ce qui implique des coûts liés aux serveurs, à la bande passante et à la gestion des identités.

- **Off-Chain (IPFS)** : Le stockage des documents via IPFS réduit les coûts d'infrastructure, car seuls les hachages des fichiers sont enregistrés sur la blockchain, limitant ainsi l'espace nécessaire sur le registre distribué.

### **Coûts de Maintenance**

- **On-Chain** : La maintenance de Hyperledger Fabric requiert une gestion continue des smart contracts et de la gouvernance du réseau.
- **Off-Chain (IPFS)** : Moins coûteux en termes de maintenance, car les fichiers sont décentralisés et distribués, nécessitant moins d'intervention pour la gestion du stockage.

### **Coûts de Transactions**

- **On-Chain** : Chaque transaction impliquant le stockage de données entraîne des frais computationnels et de consensus.
- **Off-Chain (IPFS)** : Seuls les hachages sont inscrits sur la blockchain, réduisant ainsi les frais de transaction.

### **Coûts de Sécurité**

- **On-Chain** : Hyperledger Fabric offre un niveau de sécurité élevé grâce aux permissions et au contrôle d'accès, mais cela peut impliquer des coûts supplémentaires en gestion des identités.
- **Off-Chain (IPFS)** : La sécurité des fichiers dépend de la disponibilité des nœuds IPFS et de l'intégrité assurée par le hachage stocké sur la blockchain.

## **21 Rédaction du Rapport**

### **21.1 Documentation du processus**

Le projet a suivi une approche structurée comprenant les étapes suivantes :

- Configuration et déploiement de Hyperledger Fabric sur un environnement local.
- Intégration de l'API avec les endpoints permettant la gestion des titres fonciers.
- Simulation des accès concurrents via JMeter pour mesurer les performances.
- Analyse des résultats en termes de temps de réponse, résilience et cohérence des données.
- Comparaison des coûts d'une solution on-chain vs. off-chain.

### **21.2 Découvertes et défis rencontrés**

Les principaux constats et défis identifiés lors de l'expérimentation sont :

- **Performances** : L'API a montré des temps de réponse satisfaisants, mais la charge élevée a révélé des goulots d'étranglement au niveau des requêtes simultanées.
- **Sécurité** : L'accès sans authentification aux endpoints critiques pourrait représenter un risque de violation de données.
- **Permanence des données** : IPFS assure la persistance des fichiers mais n'offre pas de garantie sur la durée de stockage sans réplication active.
- **Coûts** : Le stockage on-chain est significativement plus cher que la solution off-chain utilisant IPFS.

## **22 Recommandations**

Suite aux tests effectués, les recommandations suivantes sont formulées :

- Optimiser la gestion des requêtes pour réduire les latences sous forte charge.
- Mettre en place une authentification robuste pour protéger les endpoints sensibles.
- Assurer la réplication des fichiers IPFS sur plusieurs nœuds pour garantir leur durabilité.
- Privilégier un stockage hybride combinant on-chain pour les métadonnées et off-chain pour les fichiers volumineux.

## **23 Conclusion**

Ce projet a permis de mettre en place une application basée sur Hyperledger Fabric pour la gestion des titres fonciers, en intégrant IPFS pour la gestion décentralisée des documents. Grâce aux scripts d'automatisation, le déploiement et l'utilisation du système sont simplifiés, facilitant ainsi la gestion des transactions sur la blockchain.

L'implémentation de ce projet a démontré l'intérêt des technologies blockchain dans des domaines nécessitant transparence, sécurité et traçabilité des données. Toutefois, des améliorations peuvent être envisagées, notamment l'optimisation des performances, l'ajout de nouvelles fonctionnalités et l'intégration d'autres mécanismes de sécurité.