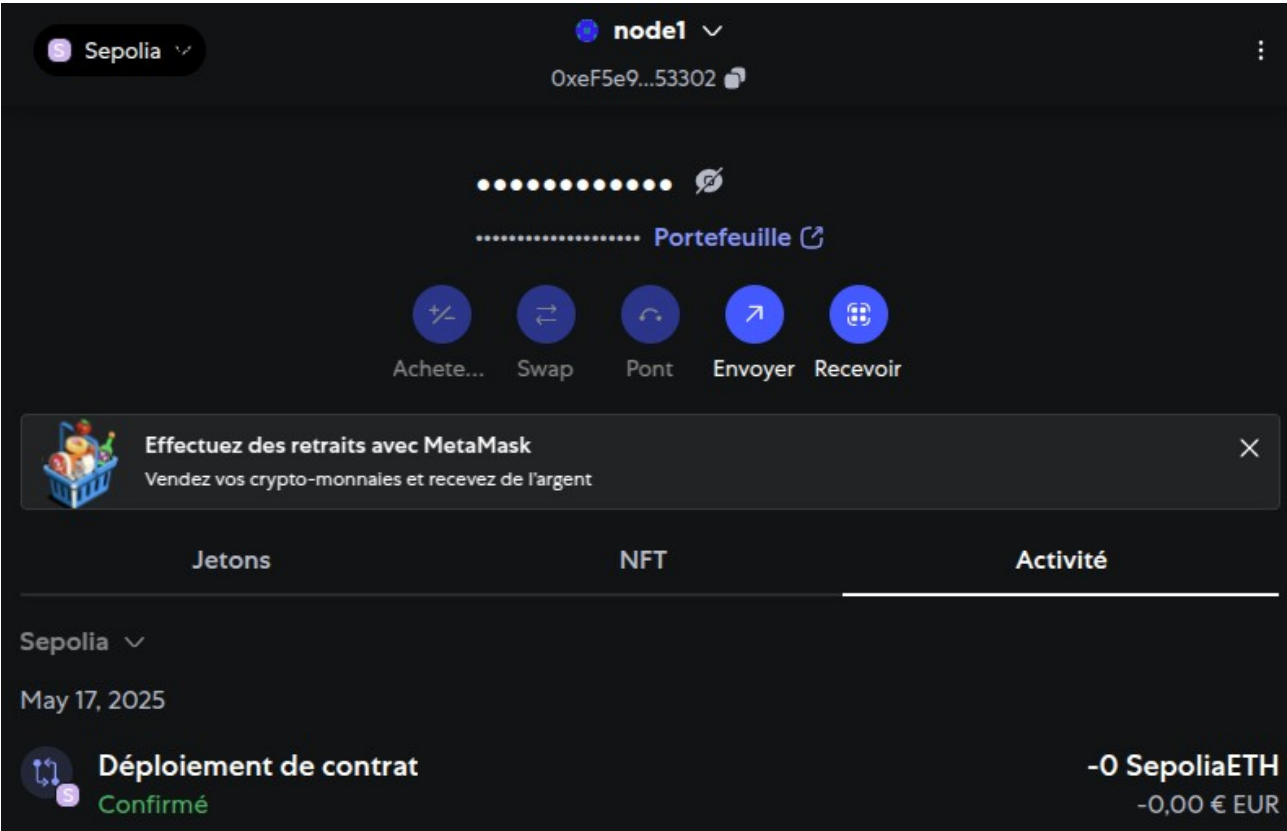


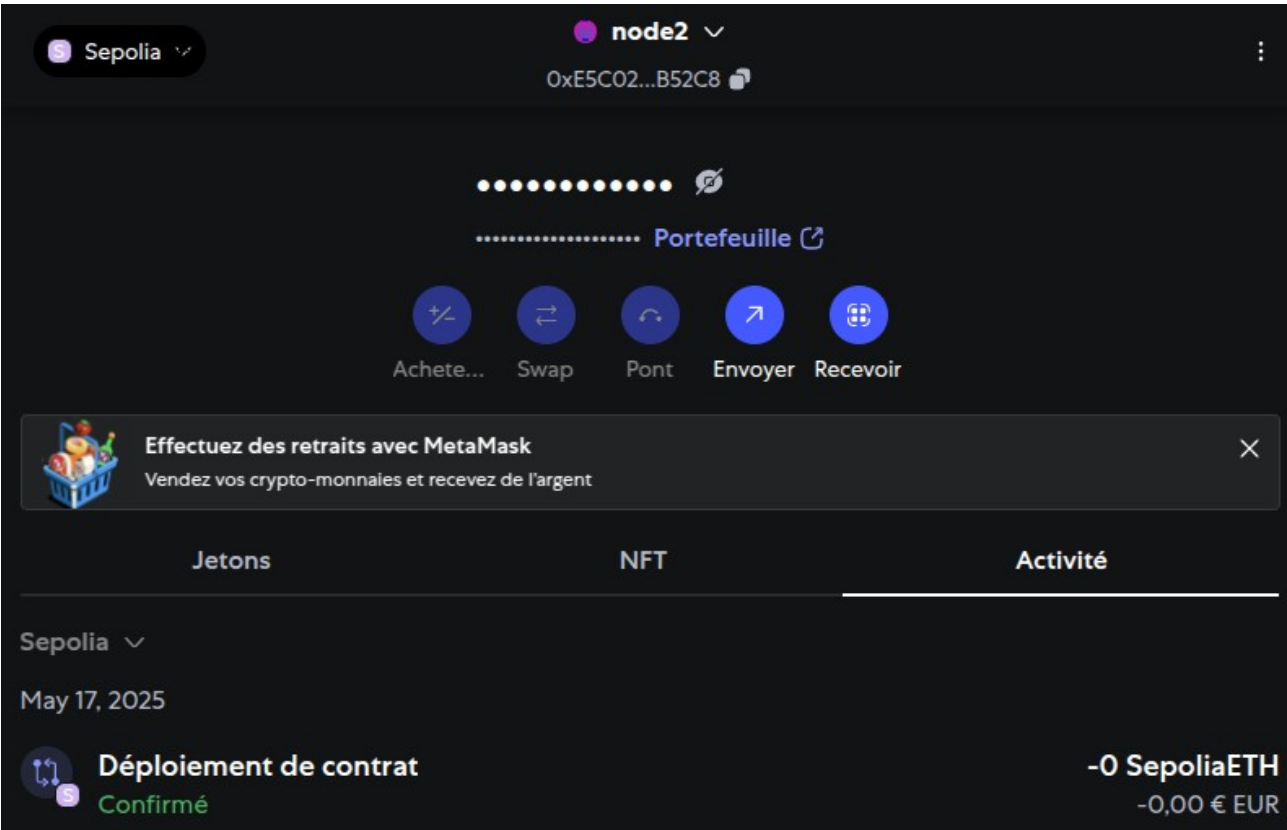
# AttestationRegistry

contract address	0x53e547ce7579b8483e6302ce25369469309c8889
------------------	--------------------------------------------



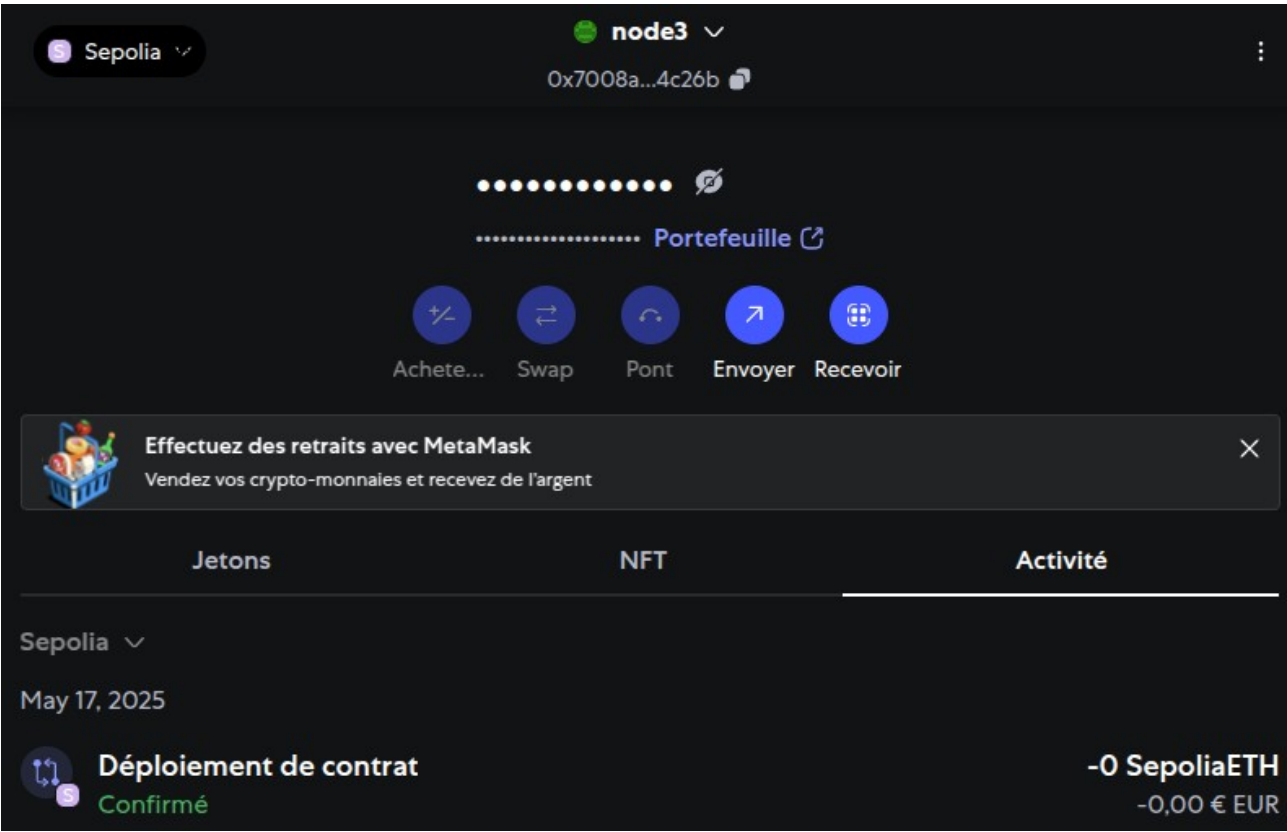
# ConsensusLog

contract address	0xae0cdce0608e385c45c90c6faad7f1de785cbea9
------------------	--------------------------------------------



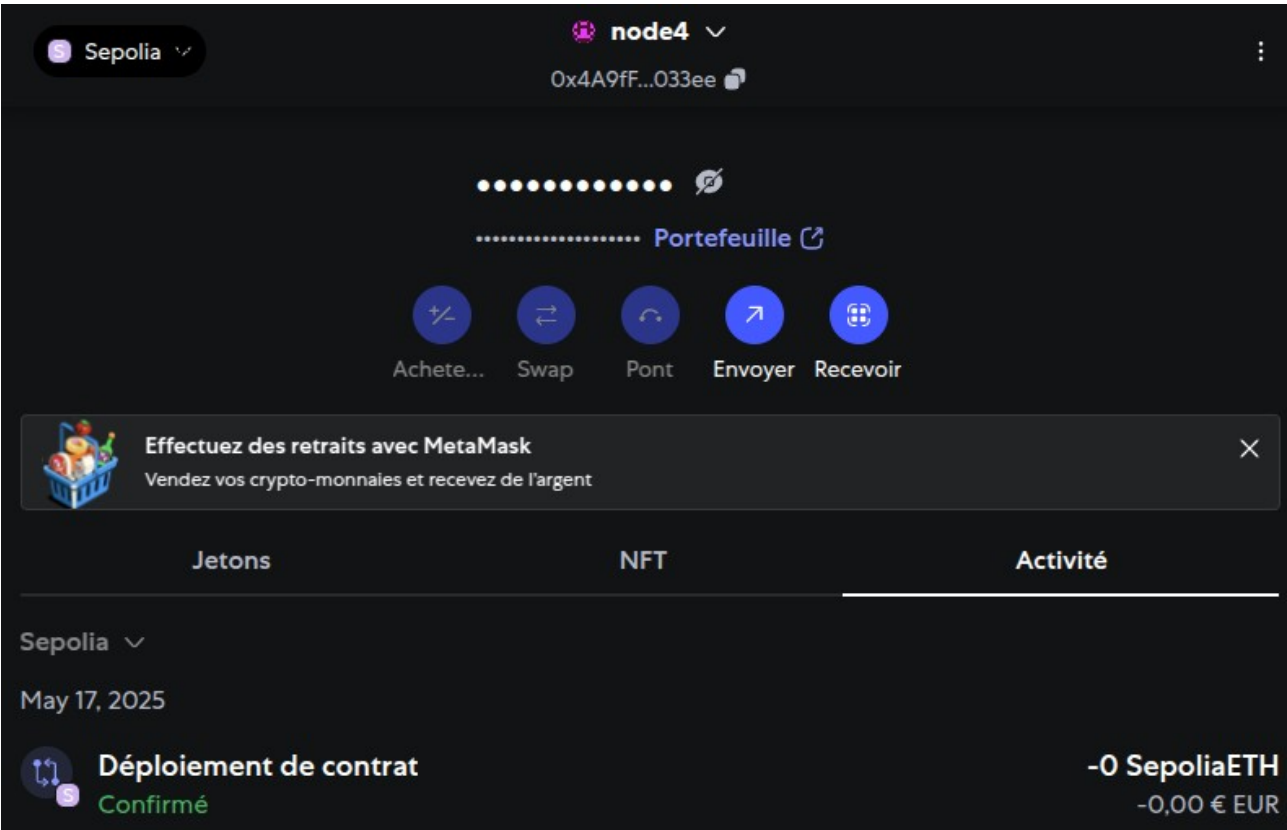
ReputationManager

contract address	0xd4db9b88778fdd9664cdde155340585118cbeeac
------------------	--------------------------------------------



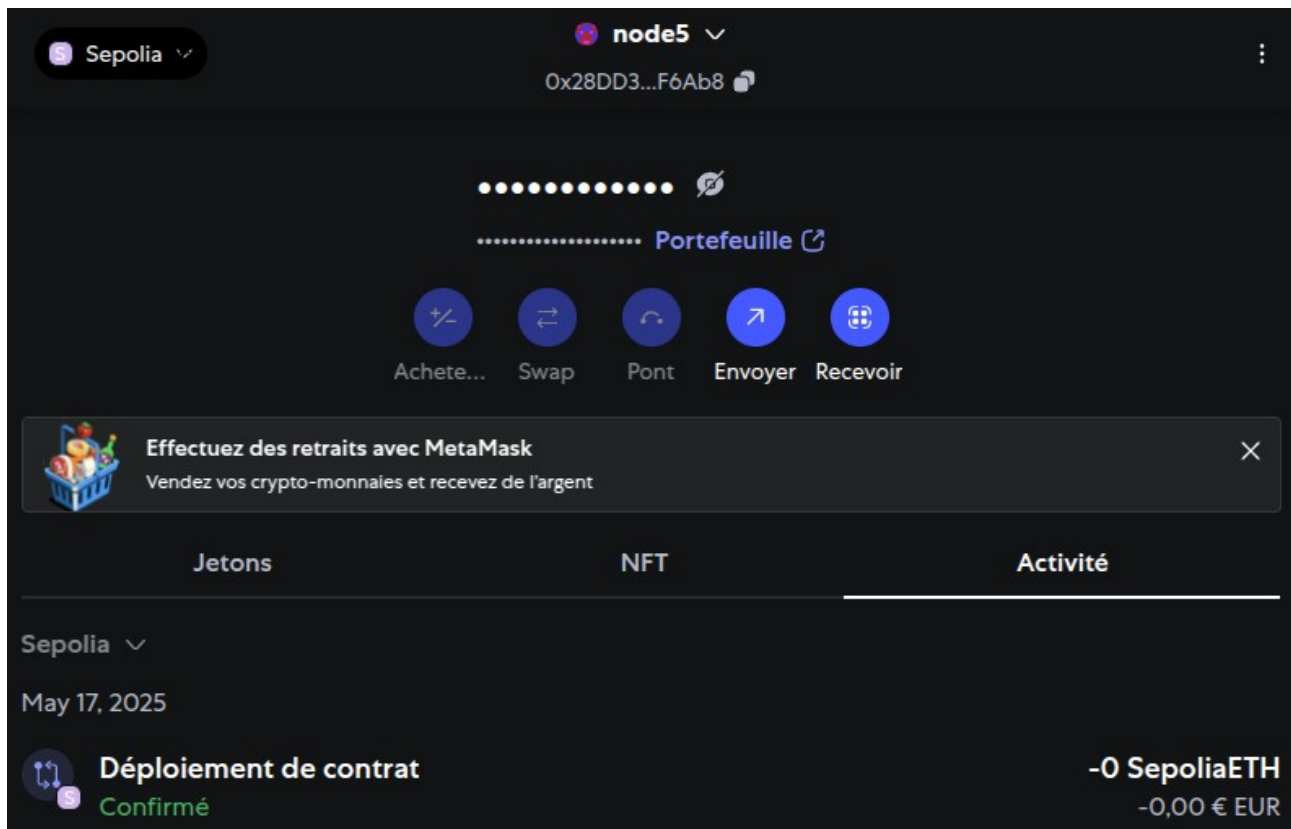
ResourceLedger

contract address	0xcb8a5538631201736ef48aec47f5ac1dad02939a
------------------	--------------------------------------------



SliceRequestManager

contract address	0x77a034e22aa2a65b5f8ab7e8438ef5f756f8b419
------------------	--------------------------------------------



```
nabster@Cisco:~/projet-netchain$ ls -a
.  ..  .abis  .env  index.js  node_modules  package.json  package-lock.json
nabster@Cisco:~/projet-netchain$
```

# Partie Simulation

## 1. DEMANDE DE SLICE

```
• nabster@Cisco:~/projet-netchain$ node index.js

=== DÉBUT DE LA SIMULATION ===

=== 1. DEMANDE DE SLICE ===
[2025-05-18T21:54:24.000Z] Client: node1
• Type: IA (Bronze)
• ID Requête: 38
• TX Hash: 0x63148323366f4ceca3cbd9aa5efb931f302cbda81216eb1660f64b3e790d97e7
```

**Signification :** Un client (node1) initie une demande de ressources.

- **Détails :**
  - **Type :** Domaine (IA) et niveau de qualité de service (Bronze) demandés.
  - **ID Requête :** Identifiant unique de la demande dans le contrat.
  - **TX Hash :** Preuve blockchain de la transaction.

## 2. ENREGISTREMENT CANDIDAT

```
=== 2. ENREGISTREMENT CANDIDAT ===
[2025-05-18T21:54:28.000Z] Candidat: node2
• Statut: Déjà enregistré
```

**Signification :** Un nœud (node2) se propose pour répondre à la demande.

- **Détails :**
  - **Statut :** Indique si le nœud est nouveau ou déjà enregistré dans le réseau.

## 3. DÉCLARATION DES RESSOURCES

```
=== 3. DÉCLARATION DES RESSOURCES ===
[2025-05-18T21:54:36.000Z] Candidat: node2
• CPU: 9 cores
• Mémoire: 5316 MB
• TX Hash: 0xca57f8638c2f3e4f9b2455af4b30aff479b9bb8b6ac6f3ba0f527fec2d4cf3aa
```

**Signification :** Le candidat déclare ses ressources disponibles.

- **Détails :**
  - **CPU/Mémoire :** Capacités techniques du nœud.
  - **TX Hash :** Preuve de la déclaration sur la blockchain.

#### 4. ÉVALUATIONS DES VOTANTS

```
=== 4. ÉVALUATIONS DES VOTANTS ===
[2025-05-18T21:54:48.000Z] Votant: node5
• Note: 4/5
• TX Hash: 0x23b7f7f6d3d41391bdcc8a20a9cf9f8f6fa4dc1632dc8b10a56ecd5f072834bf
[2025-05-18T21:55:00.000Z] Votant: node4
• Note: 5/5
• TX Hash: 0x5f1044f3f51237a781d088b49f32a98e3fe2e8d26ae42af79be8b73500f423ff
[2025-05-18T21:55:12.000Z] Votant: node3
• Note: 5/5
• TX Hash: 0x196f870444c9fb633c2a72beda0cbf31079f24e517a4e2e3fb72af669f7e6c8b

[REPUTATION FINALE] Candidat node2: 4/5
```

**Signification** : Les autres nœuds évaluent la réputation du candidat.

- **Détails** :
  - **Note** : Score attribué par chaque votant (sur 5).
  - **TX Hash** : Preuve de chaque évaluation.
  - **REPUTATION FINALE** : Moyenne des notes, déterminant l'éligibilité du candidat.

#### 5. SOUMISSION DE PROPOSITION

```
=== 5. SOUMISSION DE PROPOSITION ===
[2025-05-18T21:55:24.000Z] Candidat: node2
• ID Proposition: 30
• Contenu: req=38,node=0xE5C02a3A0b97d20Bd3e64bf8702732f6289B52C8
• TX Hash: 0x4ee7eae64bed1895582587ecca2b8d385502206c33eef8c4496a5428e8a6fd09
```

**Signification** : Le candidat soumet une proposition formelle pour la demande.

- **Détails** :
  - **ID Proposition** : Identifiant unique de la proposition.
  - **Contenu** : Lien entre la demande (req=38) et le nœud candidat.

#### 6. PROCESSUS DE VOTE

```
=== 6. PROCESSUS DE VOTE ===
[2025-05-18T21:55:36.000Z] Votant: node5
• Vote: POUR
• TX Hash: 0x4f783be65e723f3884c5f40e3bfab2648dc5846fe8e48714b9601507747b724f
[2025-05-18T21:55:48.000Z] Votant: node4
• Vote: POUR
• TX Hash: 0xc31af6ea72536287484a54886b511e03aa523a291c02b86c8c8411f09624828f
[2025-05-18T21:56:00.000Z] Votant: node3
• Vote: POUR
• TX Hash: 0x75d7f23e91e306448dfd3449f154088672840d51257cb8958b85612fc8cef65f
```

**Signification** : Les votants décident d'accepter ou rejeter la proposition.

- **Détails** :

- **Vote** : POUR ou CONTRE.
- **TX Hash** : Preuve de chaque vote.

## 7. RÉSULTAT FINAL

```
=== 7. RÉSULTAT FINAL ===

[ANALYSE]
• Votes requis: 2/3
  -> Obtenus: 3 (✓ SUFFISANT)
• Réputation minimale: 4
  -> Actuelle: 4 (✓ SUFFISANTE)
• Consensus: ACCEPTÉE

[DÉCISION FINALE]
• Statut: ● APPROUVÉE
• Requête 38: ● ACCEPTÉE
• Détails: IA (Bronze)

🕒Durée totale: 117.81 secondes
```

**Signification** : Synthèse des résultats et décision finale.

- **Détails** :
  - **[ANALYSE]** : Vérification des conditions :
    - Votes suffisants ( $\geq 2/3$  ici).
    - Réputation minimale atteinte ( $\geq 4$ ).
  - **[DÉCISION FINALE]** :
    - **Statut** : Résultat global (●/●).
    - **Requête** : Statut spécifique de la demande.
    - **Détails** : Rappel du type de demande.
  - **Durée totale** : Temps d'exécution complet de la simulation.

### Cas 2 : Demande Refusée faute de vote favorable

```
=== 7. RÉSULTAT FINAL ===

[ANALYSE]
• Votes requis: 2/3
  -> Obtenus: 1 (✗ INSUFFISANT)
• Réputation minimale: 4
  -> Actuelle: 4 (✓ SUFFISANTE)
• Consensus: REJETÉE

[DÉCISION FINALE]
• Statut: ● REJETÉE
• Requête 37: ● REJETÉE
• Détails: Blockchain (Platinum)

🕒Durée totale: 108.33 secondes
```



### Cas 3 : Demande Refusée faute de mauvaise réputation

=== 7. RÉSULTAT FINAL ===

[ANALYSE]

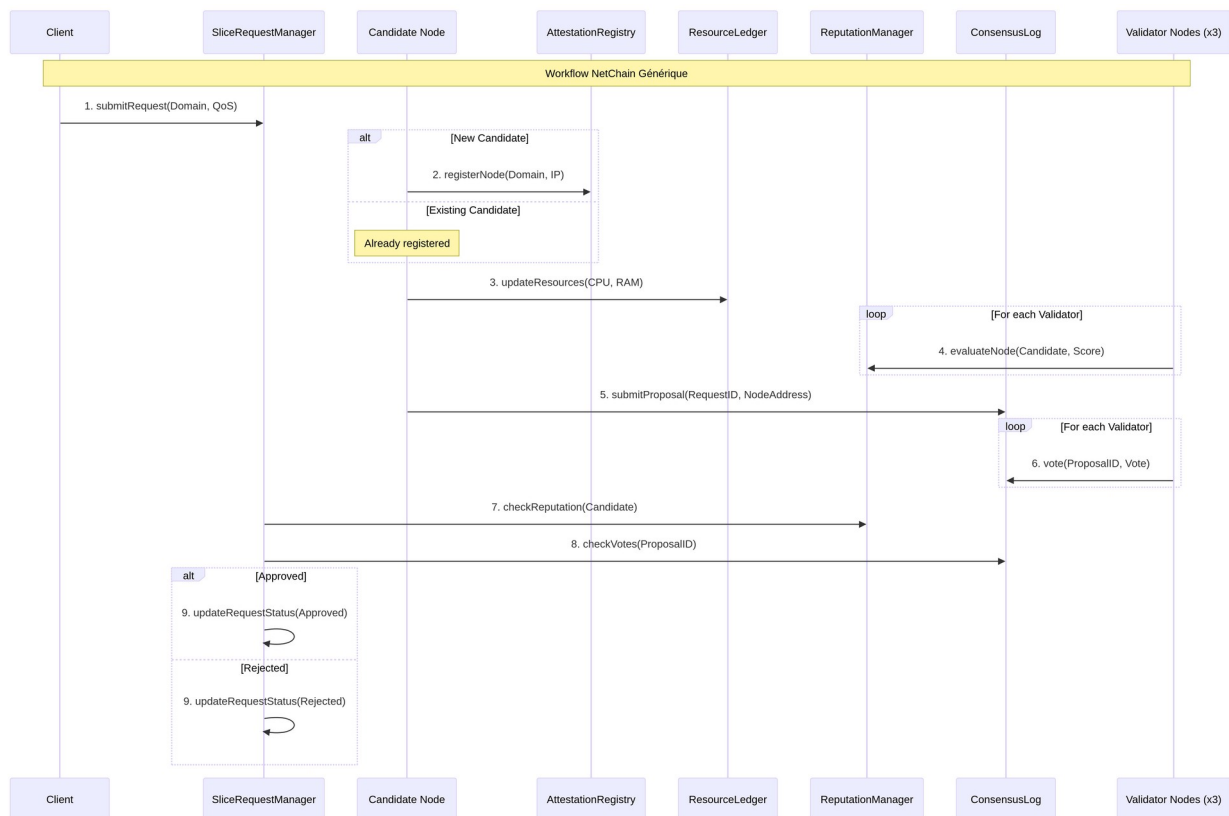
- Votes requis: 2/3
  - > Obtenus: 2 (✓ SUFFISANT)
- Réputation minimale: 4
  - > Actuelle: 3 (✗ INSUFFISANTE)
- Consensus: REJETEE

[DÉCISION FINALE]

- Statut: ● REJETÉE
- Requête 35: ● REJETÉE
- Détails: VR (Silver)

🕒Durée totale: 104.22 secondes

### Workflow



## Étapes Détaillées

### 1. Demande de Slice (Client)

- **Action** : Un client initie une demande de ressources
- **Contrat** : SliceRequestManager
- **Données** :
  - Type de service (IA/Cloud/etc.)
  - Niveau de QoS (Bronze/Silver/etc.)
- **Blockchain** : Transaction enregistrée

### 2. Enregistrement Candidat

- **Condition** : Le nœud candidat vérifie s'il est déjà enregistré
  - **Si nouveau** :
    - Envoie son IP + preuve d'attestation
    - Contrat AttestationRegistry
  - **Si existant** : Passe à l'étape 3

### 3. Déclaration des Ressources

- **Contrat** : ResourceLedger
- **Données** :

- CPU cores disponibles
- Mémoire RAM
- **But** : Montrer sa capacité à satisfaire la demande

#### **4. Évaluation par les Validateurs (x3)**

- **Contrat** : ReputationManager
- **Mécanisme** :
  - Chaque validateur note le candidat (1-5)
  - Réputation = Moyenne des notes
- **Seuil** :  $\geq 4/5$  pour être éligible

#### **5. Proposition du Candidat**

- **Contrat** : ConsensusLog
- **Contenu** :
  - Lien explicite entre la demande et le candidat
  - Format : req=[ID], node=[Adresse]



#### **6. Vote des Validateurs (x3)**

- **Options** :
  - POUR (Accepté)
  - CONTRE (Rejeté)
- **Quorum** :  $\geq 2$  votes favorables sur 3

#### **7-8. Vérifications Automatiques**

- **Double-check** :
  1. ReputationManager : Vérifie si score  $\geq 4$
  2. ConsensusLog : Compte les votes favorables
- **Blockchain** : Lecture des données sans transaction

#### **9. Décision Finale**

- **Conditions de succès** :
  - Réputation OK **ET** Votes OK →  **Accepté**
  - Sinon →  **Rejeté**
- **Action** : Mise à jour du statut dans SliceRequestManager

## Règles Métier Clés

### 1. Consensus :

- Requiert **deux conditions cumulatives** :
  - Réputation minimale (configurable, 4 par défaut)
  - Majorité de votes ( $\geq 2/3$ )

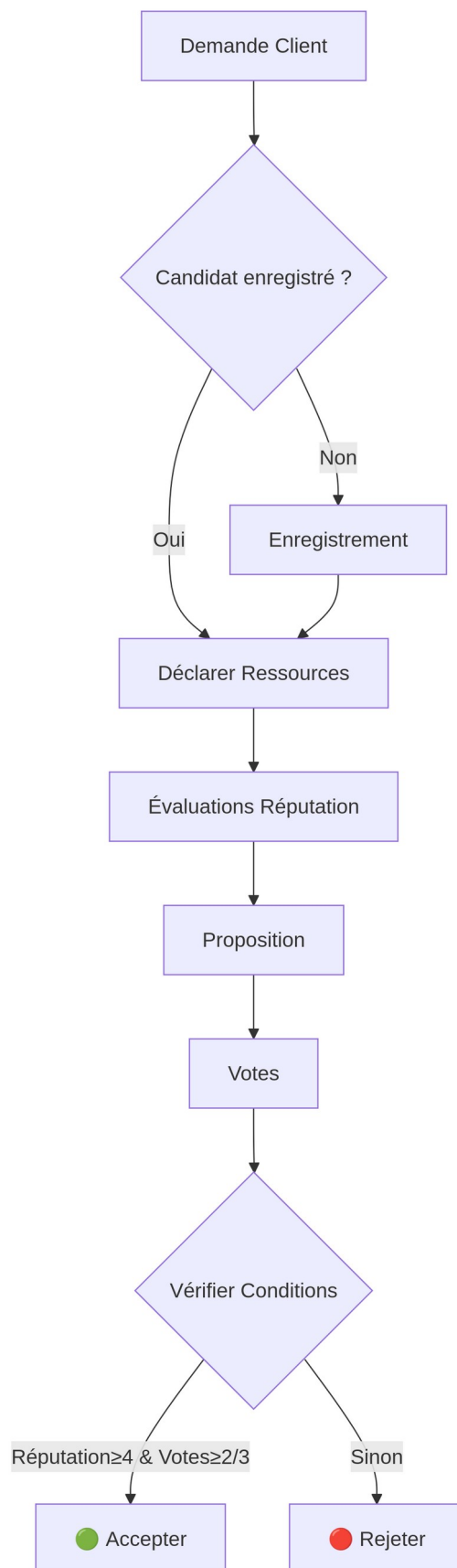
### 2. Immuabilité :

- Toutes les étapes critiques sont enregistrées sur la blockchain via :
  - TX Hash pour les écritures
  - Appels `view` pour les lectures

### 3. Séquencement :

- L'ordre est **strict** (ex: impossible de voter avant la proposition)
- Géré par les modificateurs de contrat (`onlyRegisteredNode`, etc.)

## Exemple de la logique décentralisée de NetChain



# Mise en place d'une Interface Graphique



## Architecture Technique

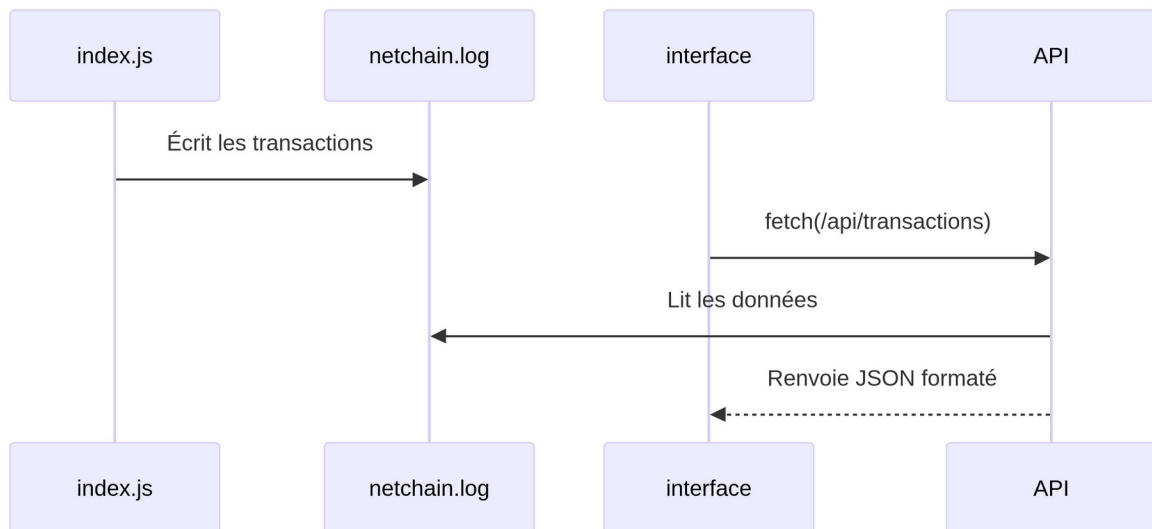
### 1. Backend (Node.js/Express) :

- Lit le fichier `netchain.log`
- Expose une API REST sur le port 4000

### 2. Frontend (React) :

- Affiche les données de l'API
- Met à jour automatiquement

### Flux de données :



### Fonctionnalités de l'interface :

- Affichage chronologique inversé (récent → ancien)
- Codage couleur (vert/rouge pour acceptation/rejet)
- Résumé lisible des événements
- Accès aux hashes de transactions
- Détails complets au clic

### Statistiques Utiles

L'interface permet de visualiser :

- Taux d'acceptation/rejet
- Temps moyen de traitement
- Répartition des votes
- Performance des nœuds

🔒 <http://localhost:3000>



# NetChain Explorer

<b>FINAL_DECISION</b>	17:58:49
✅ APPROUVÉ (4/5 rep, 3/3 votes)	
<b>VOTE_CAST</b>	17:58:40
<pre>{ "event": "VOTE_CAST", "proposalId": 35, "timestamp": "2025-05-21T15:58:40.562Z", "txHash": "0x3cdea7f1ba8d9833ca50eada0f8eeb29746f4b5f3eec761897c59daab613b433", "vote": "FOR", "voter": "node3" }</pre>	
TX: 0x3cdea7f1ba...b433	
<b>VOTE_CAST</b>	17:58:27
<pre>{ "event": "VOTE_CAST", "proposalId": 35, "timestamp": "2025-05-21T15:58:27.957Z", "txHash": "0xf0f909b63a0d77913865faad75bbfe8640f28e7f280ef20c1dac6e03c4cf1de8", "vote": "FOR", "voter": "node2" }</pre>	
TX: 0xf0f909b63a...1de8	
<b>VOTE_CAST</b>	17:58:15
<pre>{ "event": "VOTE_CAST", "proposalId": 35, "timestamp": "2025-05-21T15:58:15.345Z", "txHash": "0xfa6496b219fa7f5034d915cc08a4252cb66ba79de3b8e8ee315312da21932378", "vote": "FOR", "voter": "node5" }</pre>	
TX: 0xfa6496b219...2378	

## conclusion

