Atelier 3 : Blockchain

```python
import hashlib

# Données du bloc
transactions = (
    "Ahmed -> Ali : 6 BTC\n"
    "Ahmed -> Adam : 2 BTC"
)

# Hash Precedent
previous_hash = "abc123"

nonce = 0  # valeur initiale

print("Début du minage...\n")

while True:
    # Contenu du bloc
    block_data = transactions + previous_hash + str(nonce)

    # Calcul du hash SHA-256
    block_hash = hashlib.sha256(block_data.encode()).hexdigest()

    # Vérifier la difficulté (hash commence par '000')
    if block_hash.startswith("000"):
        print("Bloc miné avec succès !")
        print("Nonce trouvé :", nonce)
        print("Hash du bloc :", block_hash)
        break

    nonce += 1
```

Resultat :

```
Bloc miné avec succès !
Nonce trouvé : 5775
Hash du bloc : 000e0bc8481b828939e08c62f159733bb69cb65d42eeb8a0f726e8fea234ac6e
```

Projet : Smart Contract de Vote sur Blockchain

1. Introduction

Dans une blockchain, un vote est représenté par une transaction envoyée à un smart contract.
Ce smart contract permet d'organiser une élection de manière **décentralisée**, **transparente** et **immuable**.

L'objectif de ce projet est de concevoir, coder et déployer un **smart contract de vote** permettant :

- de définir une liste de candidats à la création,
- d'enregistrer les votes des participants,
- de garantir qu'un participant ne vote qu'une seule fois,
- de stocker les résultats de manière transparente sur la blockchain.

2. Environnement de travail

- Langage : **Solidity**
- Environnement de développement : **Remix IDE**
- Blockchain : **Blockchain locale (Remix VM)**

3. Conception du Smart Contract

Le smart contract doit assurer les fonctionnalités suivantes :

1. Stocker une liste de candidats
2. Enregistrer le vote d'un participant
3. Empêcher un participant de voter plus d'une fois
4. Permettre la consultation publique des résultats

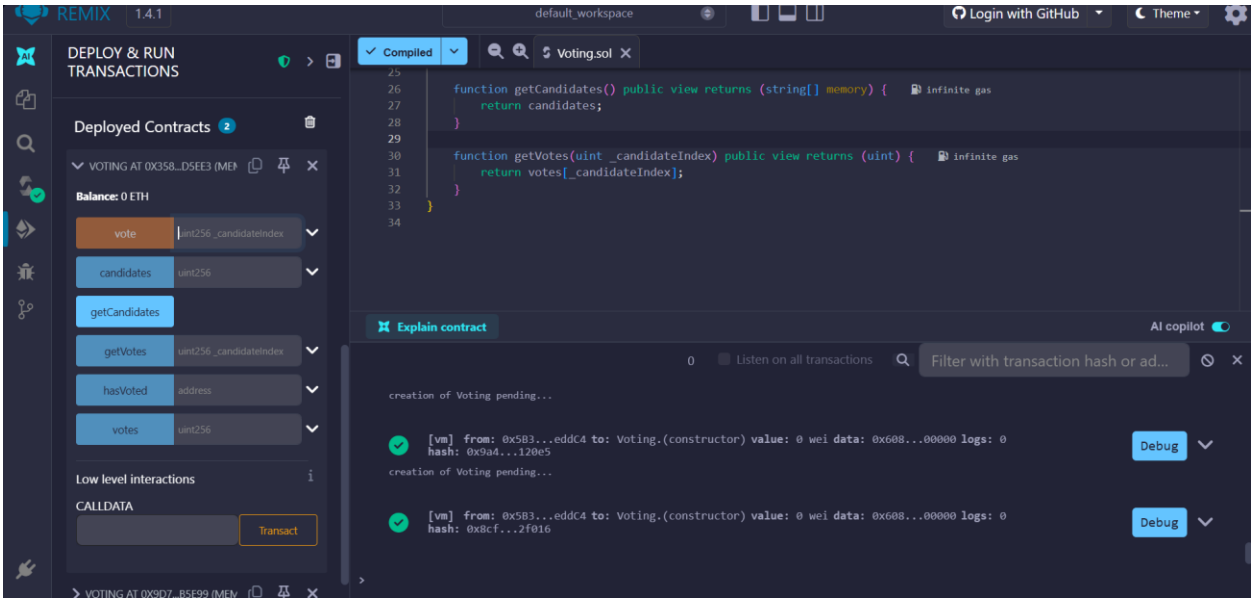
4. Implémentation du Smart Contract

4.1 Code Solidity

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Voting {      PUSH1 costs 3 gas - this line costs 18 gas - 2878424 gas left

    address public owner;

    // Liste des candidats
    string[] public candidates;

    // Nombre de votes par candidat
    mapping(uint => uint) public votes;

    // Vérifie si une adresse a déjà voté
    mapping(address => bool) public hasVoted;

    // Événement (pour la transparence)
    event VoteCasted(address voter, uint candidateIndex);

    // Constructeur : exécuté une seule fois au déploiement
    constructor(string[] memory _candidates) {      infinite gas 534800 gas
        owner = msg.sender;
        candidates = _candidates;
    }

    // Fonction pour voter
    function vote(uint _candidateIndex) public {      infinite gas
        require(!hasVoted[msg.sender], "Vous avez deja vote");
        require(_candidateIndex < candidates.length, "Candidat invalide");

        hasVoted[msg.sender] = true;
        votes[_candidateIndex] += 1;

        emit VoteCasted(msg.sender, _candidateIndex);
    }

    // Retourner la liste des candidats
    function getCandidates() public view returns (string[] memory) {      infinite gas
        return candidates;
    }

    // Retourner le nombre de votes d'un candidat
    function getVotes(uint _candidateIndex) public view returns (uint) {      infinite gas
        require(_candidateIndex < candidates.length, "Candidat invalide");
        return votes[_candidateIndex];
    }
}
```

Notre code Solidity a bien été compiler sans aucun problème 

Notre code a bien été Deployer sans aucun problème :



5. Tests et Validation

- Un compte peut voter une seule fois
- Toute tentative de double vote est rejetée
- Les votes sont correctement comptabilisés
- Les fonctions getVotes() et getCandidates() permettent à tout utilisateur de consulter les résultats
- Les données sont stockées de manière immuable sur la blockchain

6. Sécurité et transparence

- Le contrôle hasVoted empêche le vote multiple
- Les événements VoteCasted garantissent la traçabilité des votes
- La blockchain assure l'immuabilité et la transparence des résultats

7. Conclusion

Ce projet démontre comment un smart contract peut être utilisé pour implémenter un système de vote sécurisé et décentralisé.
Grâce à la blockchain, le processus est transparent, fiable et ne dépend d'aucune autorité centrale.