

Compte Rendu du TP : Smart Contract pour Vote Électronique

Fait par :

**CHEICK OUMAR THERA
NVARA ROKU JUAN ANTONIO**

Objectif

L'objectif de ce TP est de développer une application de vote électronique sécurisée sur Ethereum en utilisant :

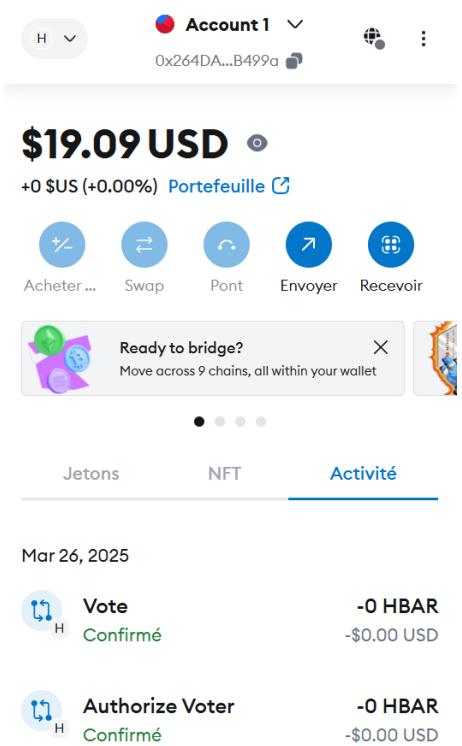
- **Solidity** pour le smart contract,
- **Streamlit** pour l'interface utilisateur,
- **Web3.py** pour l'interaction avec la blockchain.

Étape 1 : Préparer l'Environnement

1. Installer MetaMask et créer un portefeuille

MetaMask permet de gérer les identités et transactions blockchain directement depuis un navigateur. L'obtention d'ETH de test via un faucet est essentielle pour simuler des transactions sans coût réel, facilitant le développement et les tests de smart contracts.

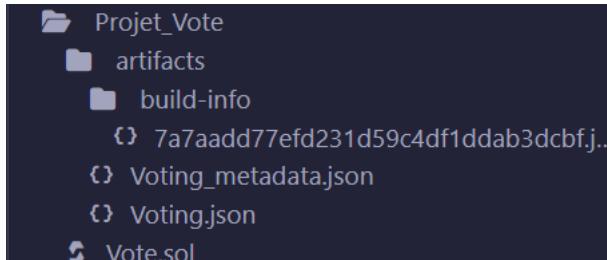
- Télécharger MetaMask depuis <https://metamask.io/>.
- Créer un nouveau wallet et sauvegarder la seed phrase (**Ne pas partager !**).
- Ajouter des ETH de test depuis un faucet.



(Capture d'écran de l'extension MetaMask installée et configurée.)

2. Configurer Remix pour écrire le Smart Contract

- Accéder à [Remix Ethereum](#).
- Créer un nouveau fichier Solidity nommé **Vote.sol**.
- Écrire et tester le smart contract.



(Capture d'écran de Remix avec le fichier *Vote.sol* ouvert.)

Étape 2 : Développer le Smart Contract Solidity

Le smart contract intègre plusieurs mécanismes de sécurité :

- Un registre des électeurs autorisés pour prévenir le vote multiple
- Un système de vérification d'identité par adresse blockchain
- Un mécanisme de vote unique par électeur, garanti par la blockchain

1. Écriture du Smart Contract

Le smart contract doit permettre :

- D'ajouter des candidats.
- D'enregistrer les électeurs autorisés.
- De permettre aux électeurs de voter une seule fois.
- De compter les votes et afficher le gagnant.

(Code du smart contract Solidity sur GitHub.)

2. Déployer le contrat sur un testnet

- Compiler le contrat dans Remix.
- Connecter MetaMask.
- Déployer et récupérer l'adresse du contrat.

(Capture d'écran montrant le déploiement réussi du contrat avec l'adresse récupérée.)

Étape 3 : Développer l'Interface Web avec Streamlit

L'interface simule un système de vote réel avec :

- Authentification des comptes
- Vérification des droits de vote
- Enregistrement sécurisé des votes
- Visualisation dynamique des résultats

Note : Cette version utilise Hedera Hashgraph, une alternative à Ethereum offrant des transactions plus rapides et plus économiques.

1. Installer Streamlit et Web3.py

Installer les dépendances nécessaires avec :

```
pip install streamlit web3
```

2. Crée une interface permettant de :

- Connecter MetaMask.
- Ajouter des candidats.
- Voter pour un candidat.
- Voir le gagnant en temps réel.

Code Python de l'interface Streamlit.

```
import streamlit as st
import requests
import json

# Configuration de Hedera Testnet
HEDERA_API_URL = "https://testnet.mirrornode.hedera.com/api/v1"
HEDERA_ACCOUNT_ID = "0.0.5767868" # Remplace par ton ID Hedera
PRIVATE_KEY =
"21cdcf5694fafeb1b80e1a65c3904595391fe8decb52b5255e9146e3ae34d9d3" #
Clé privée admin

st.title("🗳️ Système de vote basé sur Hedera")

# Stockage des votes (simulation)
if "votes" not in st.session_state:
    st.session_state.votes = {"Alice": 0, "Bob": 0, "Charlie": 0}

# Entrée de l'utilisateur
account = st.text_input("❖ Entrez votre adresse Hedera (ex: 0.0.12345):")
if not account.startswith("0.0."):
    st.error("⚠️ Adresse Hedera invalide.")
    st.stop()

st.write(f"☑️ Bienvenue, {account}")

# Vérification du compte sur Hedera
response = requests.get(f"{HEDERA_API_URL}/accounts/{account}")
if response.status_code != 200:
    st.error("⚠️ Le compte Hedera n'existe pas.")
    st.stop()

# Vérification admin
if st.checkbox("👤 Se connecter en tant qu'administrateur"):
    admin_private_key = st.text_input("🔑 Entrez votre clé privée d'admin:", type="password")
    if admin_private_key and st.button("☑️ Autoriser un électeur"):
        voter_address = st.text_input("❖ Adresse Hedera de l'électeur à autoriser:")
        if voter_address.startswith("0.0."):
            st.success(f"🎉 Électeur {voter_address} autorisé avec succès ! (Simulation)")

# Liste des candidats
candidates = ["Alice", "Bob", "Charlie"]
selected_candidate = st.selectbox("📋 Choisissez un candidat pour voter:", candidates)

# Vérification de l'autorisation de voter (simulation)
is_authorized = True # À remplacer par une vraie autorisation

if not is_authorized:
    st.error("⚠️ Vous n'êtes pas autorisé à voter.")
else:
    private_key = st.text_input("🔑 Entrez votre clé privée pour voter:", type="password")
```

```
if st.button("🗳️ Voter maintenant") and private_key:  
    # Ajout du vote (simulation)  
    st.session_state.votes[selected_candidate] += 1  
    st.success(f"☑️ Vote enregistré pour {selected_candidate} !")  
  
# Affichage du gagnant en temps réel  
st.subheader("📊 Résultats en temps réel")  
leader = max(st.session_state.votes, key=st.session_state.votes.get)  
st.write(f"🏆 Le gagnant actuel est **{leader}** avec  
**{st.session_state.votes[leader]}** votes !")  
  
st.bar_chart(st.session_state.votes)
```

🗳️ Système de vote basé sur Hedera

- ◆ Entrez votre adresse Hedera (ex: 0.0.12345):

0.0.5767868

✓ Bienvenue, 0.0.5767868

🔒 Se connecter en tant qu'administrateur

👤 Choisissez un candidat pour voter:

Alice



🔑 Entrez votre clé privée pour voter:

.....

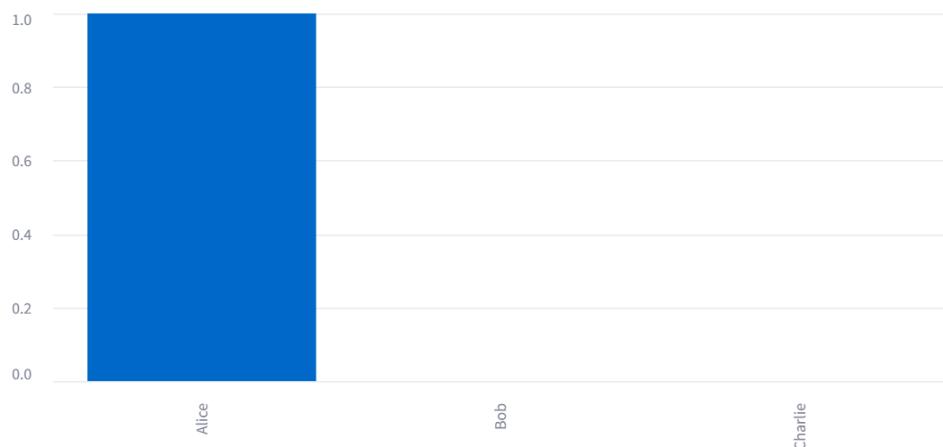


🗳️ Voter maintenant

Vote enregistré pour Alice !

📊 Résultats en temps réel

🏆 Le gagnant actuel est Alice avec 1 votes !



(Capture d'écran de l'interface web en fonctionnement.)

Étape 4 : Tester avec Deux Comptes MetaMask

Effectuer un test complet avec :

1. **Compte 1 (Admin)** : Ajoute des candidats et autorise des électeurs.
2. **Compte 2 (Électeur)** : Vote pour un candidat.
3. **Compte 1 (Admin)** : Vérifie le gagnant.

Code python du test complet

```
import streamlit as st
import requests
import json

# Configuration de Hedera Testnet
HEDERA_API_URL = "https://testnet.mirrornode.hedera.com/api/v1"
HEDERA_ACCOUNT_ID = "0.0.5767868" # Remplace par ton ID Hedera
PRIVATE_KEY =
"21cdcf5694fafeb1b80e1a65c3904595391fe8decb52b5255e9146e3ae34d9d3" # Clé
privée admin

st.title("🗳️ Système de vote basé sur Hedera")

# Stockage des votes (simulation)
if "votes" not in st.session_state:
    st.session_state.votes = {"Alice": 0, "Bob": 0, "Charlie": 0}

# Initialisation des comptes autorisés
```

```

if "authorized_voters" not in st.session_state:
    st.session_state.authorized_voters = {"0.0.5777081", "0.0.5767675",
"0.0.902"}

# Entrée de l'utilisateur
account = st.text_input("⚠ Entrez votre adresse Hedera (ex: 0.0.12345):")
if not account.startswith("0.0."):
    st.error("⚠ Adresse Hedera invalide.")
    st.stop()

st.write(f"✅ Bienvenue, {account}")

# Vérification du compte sur Hedera
response = requests.get(f"{HEDERA_API_URL}/accounts/{account}")
if response.status_code != 200:
    st.error("⚠ Le compte Hedera n'existe pas.")
    st.stop()

# Vérification admin
if st.checkbox("🔒 Se connecter en tant qu'administrateur"):
    admin_private_key = st.text_input("🔑 Entrez votre clé privée d'admin:", type="password")
    if admin_private_key and st.button("✅ Autoriser un électeur"):
        voter_address = st.text_input("⚠ Adresse Hedera de l'électeur à autoriser:")
        if voter_address.startswith("0.0."):
            st.session_state.authorized_voters.add(voter_address)
            st.success(f"🎉 Électeur {voter_address} autorisé avec succès ! (Simulation)")

# Liste des candidats
candidates = ["Alice", "Bob", "Charlie"]
selected_candidate = st.selectbox("📋 Choisissez un candidat pour voter:", candidates)

# Vérification de l'autorisation de voter (simulation)
is_authorized = account in st.session_state.authorized_voters

if not is_authorized:
    st.error("⚠ Vous n'êtes pas autorisé à voter.")
else:
    private_key = st.text_input("🔑 Entrez votre clé privée pour voter:", type="password")
    if st.button("📋 Voter maintenant") and private_key:
        # Ajout du vote (simulation)
        st.session_state.votes[selected_candidate] += 1
        st.success(f"✅ Vote enregistré pour {selected_candidate} !")

# Affichage des résultats en temps réel
st.subheader("📊 Résultats en temps réel")
leader = max(st.session_state.votes, key=st.session_state.votes.get)
st.write(f"🎉 Le gagnant actuel est **{leader}** avec **{st.session_state.votes[leader]} votes** !")

st.bar_chart(st.session_state.votes)

```

Système de vote basé sur Hedera

- ◆ Entrez votre adresse Hedera (ex: 0.0.12345):

0.0.5767868

Bienvenue, 0.0.5767868

 Se connecter en tant qu'administrateur

 Entrez votre clé privée d'admin:

.....



Autoriser un électeur

- ◆ Ajoutez un nouveau candidat:

Juan

 Ajouter candidat

 Candidat Juan ajouté avec succès ! (Simulation)

Système de vote basé sur Hedera

- ◆ Entrez votre adresse Hedera (ex: 0.0.12345):

0.0.5777081

Bienvenue, 0.0.5777081

 Se connecter en tant qu'administrateur

 Choisissez un candidat pour voter:

Alice



 Entrez votre clé privée pour voter:

.....

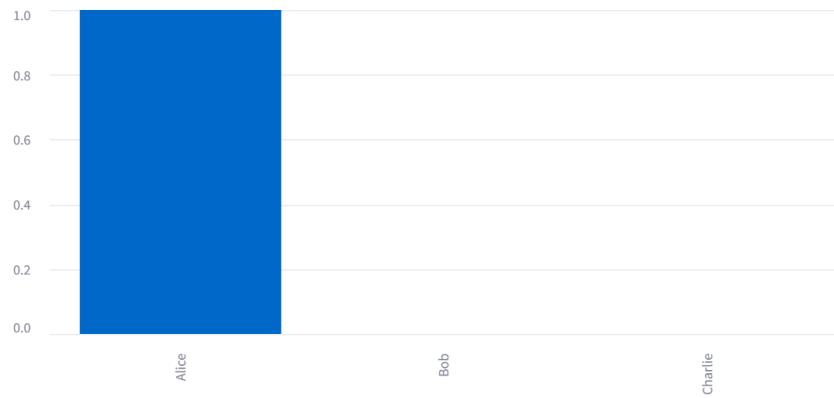


 Voter maintenant

Vote enregistré pour Alice !

Résultats en temps réel

🏆 Le gagnant actuel est Alice avec 1 votes !



(Capture d'écran des différentes étapes du test.)

Conclusion : Ce travail pratique illustre la puissance des technologies blockchain dans la modernisation des processus démocratiques. En combinant smart contracts, interface web et authentification sécurisée, nous avons démontré comment la technologie peut garantir transparence, intégrité et accessibilité du vote électronique.