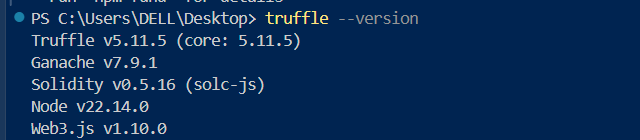
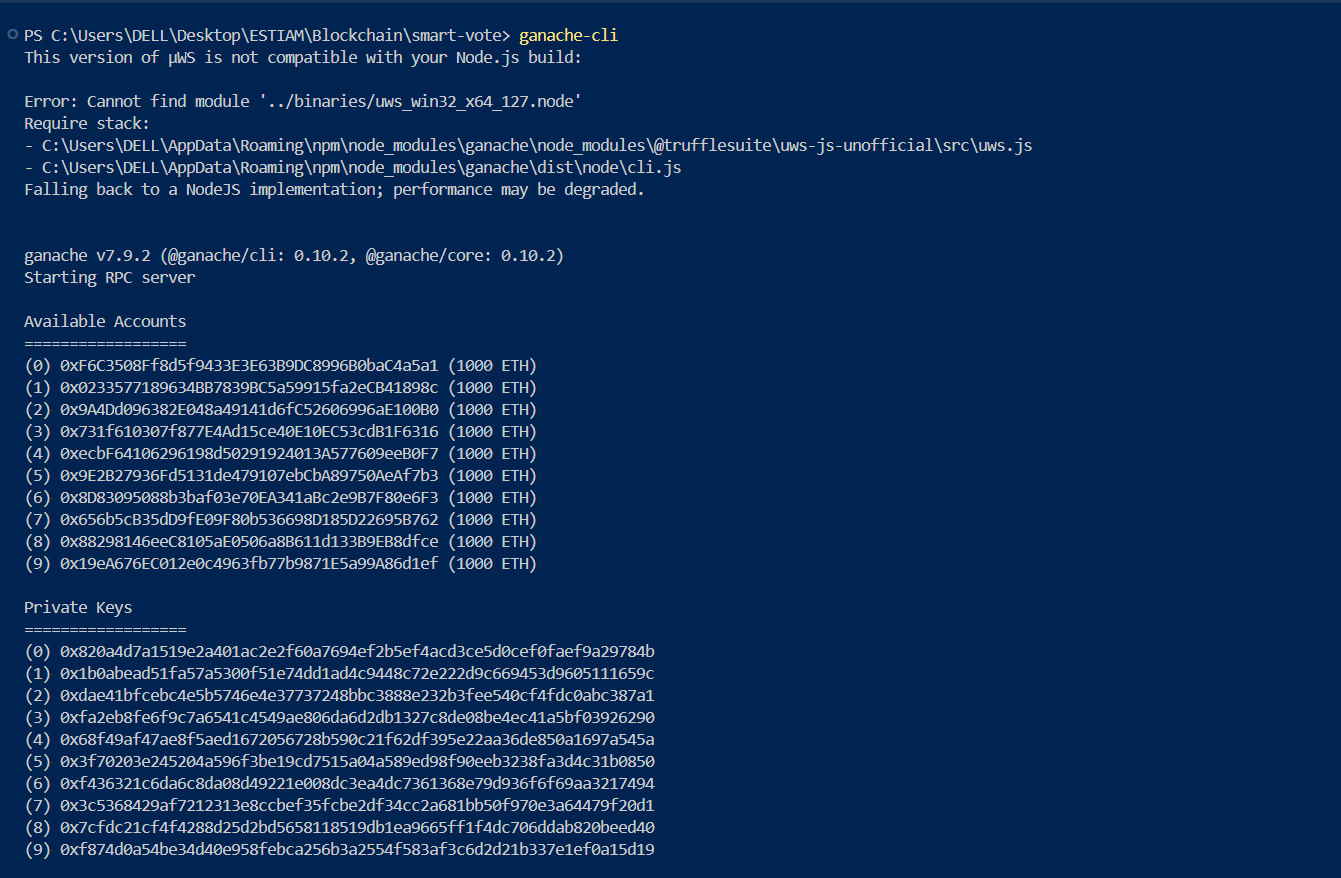
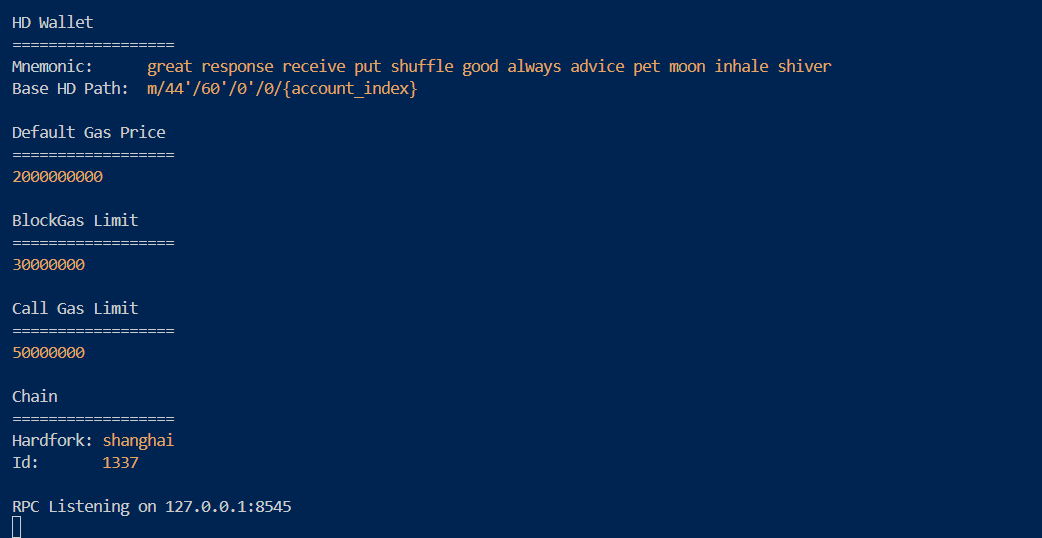
**Tâche 5 - Test Smart Contract avec Truffle (Salamata)**  
   - Écrire et exécuter tests unitaires (Truffle)  
   - Tester cas limites (double vote, votes invalides)

Versions utilisées



Démarrer la blockchain locale

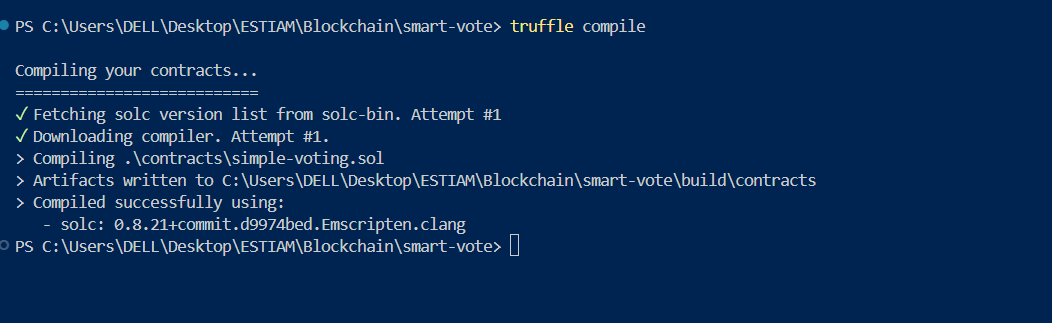




Le serveur RPC tourne bien sur **127.0.0.1:8545** avec 10 comptes à 1000 ETH chacun.

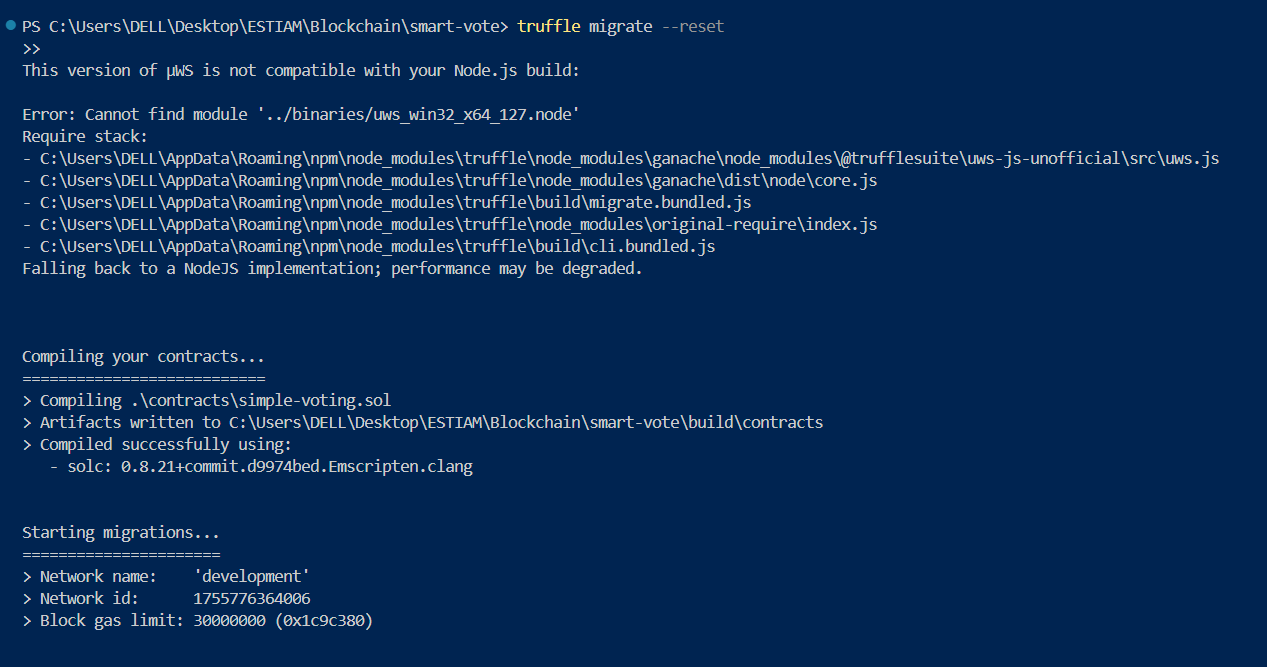
Dans un autre terminal :

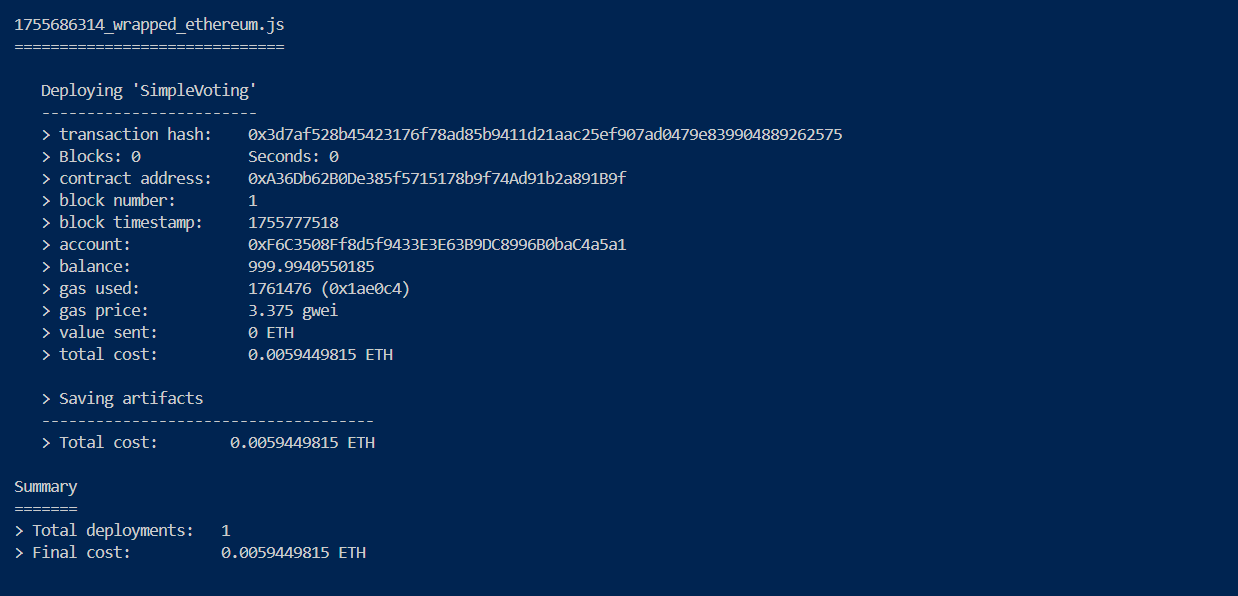
Compiler les contrats



* Compilation réussie
* Artifacts générés dans *build/contracts*

Déployer les contrats

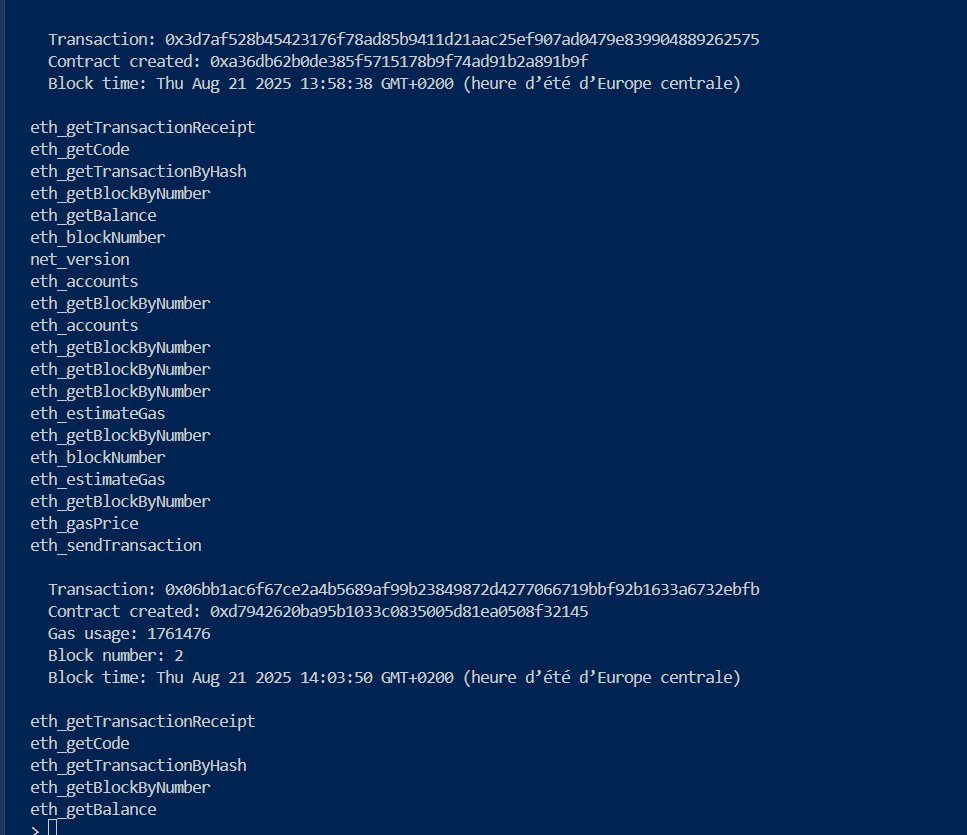




Résumé du déploiement :

* Adresse du contrat : 0xA36Db62B0De385f5715178b9f74Ad91b2a891B9f
* Compte déployeur : 0xF6C3508Ff8d5f9433E3E63B9DC8996B0baC4a5a1
* Coût du déploiement : ~0.0059 ETH (en local, donc gratuit en pratique sur Ganache)

Dans le terminal Ganache :



Tests pour VotingSystem.test.js

***1. Déploiement***

**Objectif :** Vérifier que le contrat est bien initialisé et que ses paramètres de base sont corrects.

* **Contrôle du déploiement :** Le contrat possède bien une adresse valide après le déploiement.
* **Propriétaire défini :** L’adresse qui a déployé le contrat est bien enregistrée comme propriétaire.
* **Candidats initialisés :** Le nombre de candidats et leurs noms sont correctement stockés.

***2. Gestion de la phase de vote***

**Objectif :** Contrôler qui peut ouvrir ou fermer le vote.

* **Ouverture par le propriétaire :** Seul le propriétaire peut ouvrir le vote.
* **Interdiction pour non-propriétaire :** Les autres adresses ne peuvent pas ouvrir le vote.
* **Fermeture par le propriétaire :** Seul le propriétaire peut fermer le vote.

***3. Processus de vote***

**Objectif :** Vérifier le fonctionnement du vote et l’intégrité des données.

* **Votes valides :** Un votant peut voter pour un candidat existant, le vote est comptabilisé et l’événement est émis.
* **Prévention du double vote :** Une même adresse ne peut pas voter deux fois.
* **Votes invalides :** Les votes pour des candidats inexistants sont rejetés.
* **Comptage correct :** Le total des votes pour chaque candidat est calculé correctement même avec plusieurs votants.

***4. Résultats et requêtes***

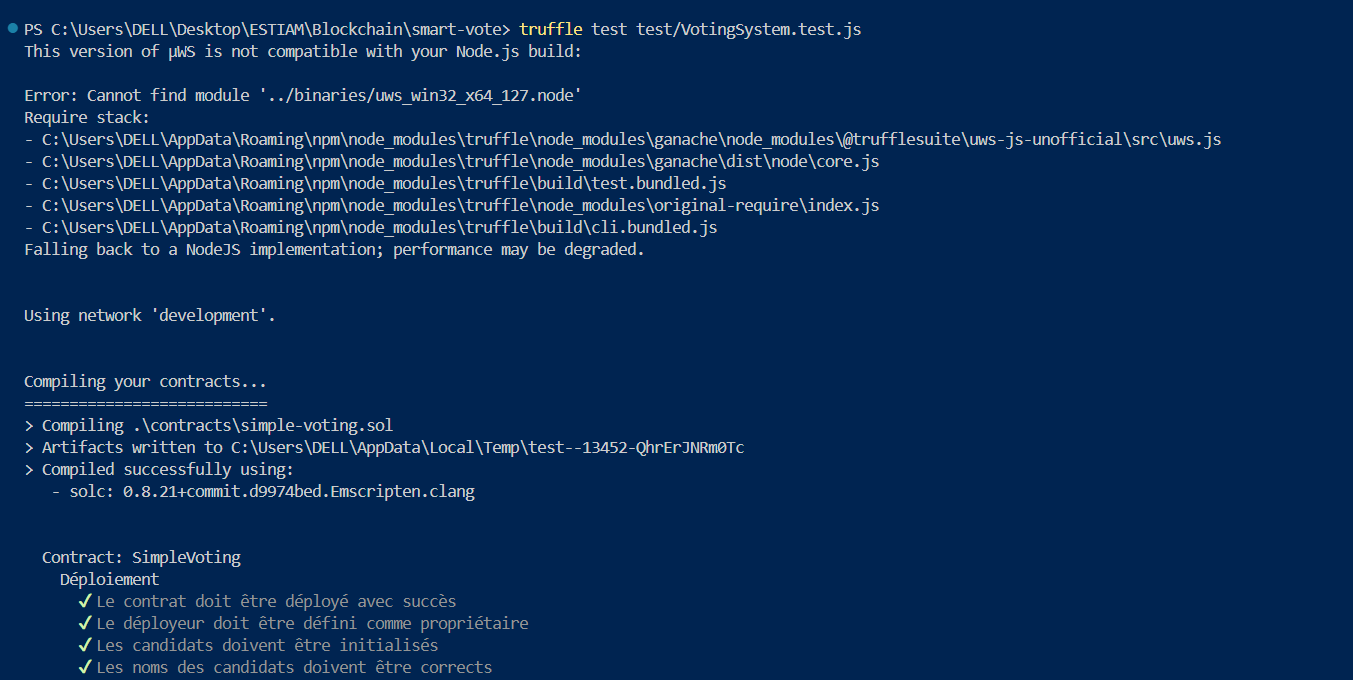
**Objectif :** Vérifier que les résultats et le statut des votants sont accessibles et corrects.

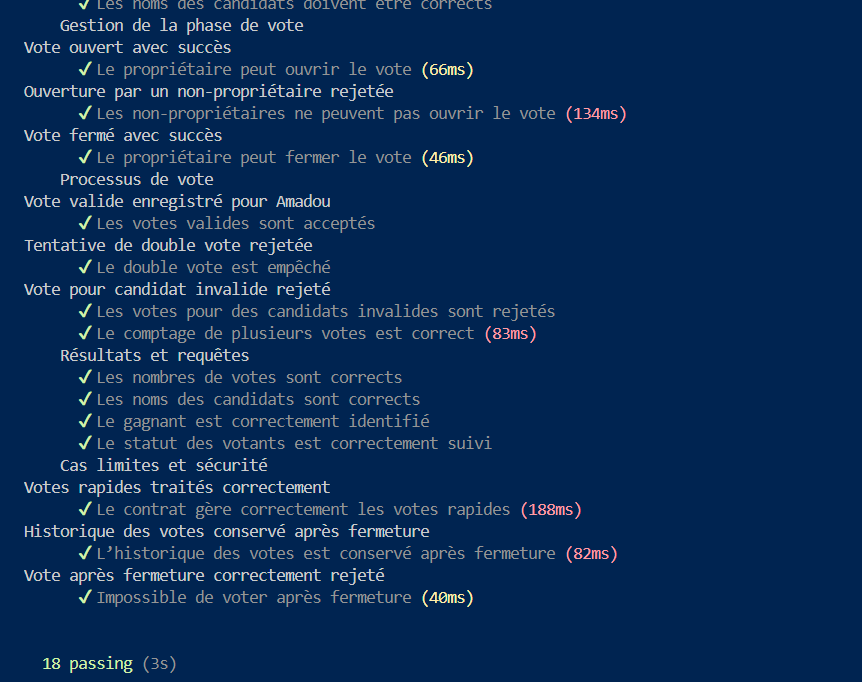
* **Nombre de votes correct :** Le comptage des votes pour chaque candidat est exact.
* **Noms des candidats corrects :** Les noms restent conformes à l’initialisation.
* **Gagnant correctement identifié :** Le candidat avec le plus de votes est bien déterminé.
* **Statut des votants :** Chaque votant est correctement marqué comme ayant voté ou non.

***5. Cas limites et sécurité***

**Objectif :** Tester le comportement du contrat dans des situations extrêmes ou concurrentes.

* **Votes rapides simultanés :** Le contrat peut traiter plusieurs votes en même temps sans perte ni corruption de données.
* **Historique conservé après fermeture :** Les votes restent accessibles même après la fermeture du vote.
* **Vote impossible après fermeture :** Toute tentative de voter après la fermeture est rejetée.





Tests pour FailuresCases.test.js (Double vote, sécurité, etc.)

***1. Tests de sécurité - Contrôle d’accès***

**Objectif** : Vérifier que seules les adresses autorisées (le propriétaire) peuvent effectuer des actions administratives comme ouvrir ou fermer le vote.

* **Test principal** : Empêcher qu’un attaquant (adresse non propriétaire) ouvre ou ferme le vote.
* **Pourquoi c’est important** : Évite qu’un utilisateur malveillant manipule le déroulement du vote.

***2. Tests anti-double vote***

**Objectif** : Empêcher qu’une même adresse vote plusieurs fois.

* **Test** **1** : Vérifier qu’une adresse ne peut pas voter deux fois pour le même ou un autre candidat.
* **Test** **2** : Vérifier que le vote initial est bien comptabilisé et qu’un second vote échoue.
* **Pourquoi** **c’est** **important** : Garantit l’intégrité et l’équité des résultats.

***3. Tests d’entrées invalides***

**Objectif** : Empêcher les votes pour des candidats inexistants.

* **Test** **principal** : Tenter de voter avec des indices de candidats invalides (ex. 3, 999, 100).
* **Pourquoi** **c’est** **important** : Protège le système contre les erreurs ou les tentatives de fraude avec des identifiants invalides.

4**. Tests de contrôle des phases**

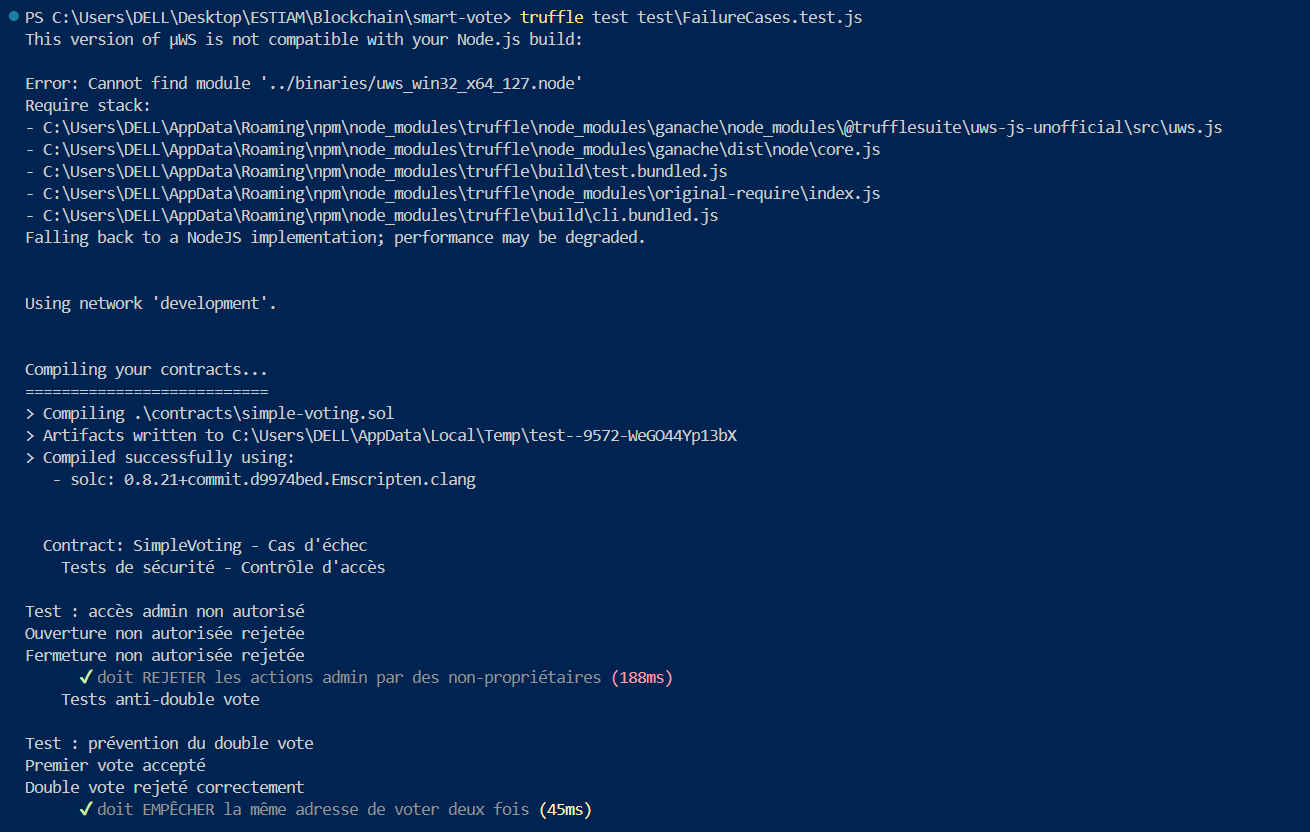
**Objectif** : Vérifier que les votes ne peuvent être déposés qu’entre l’ouverture et la fermeture du vote.

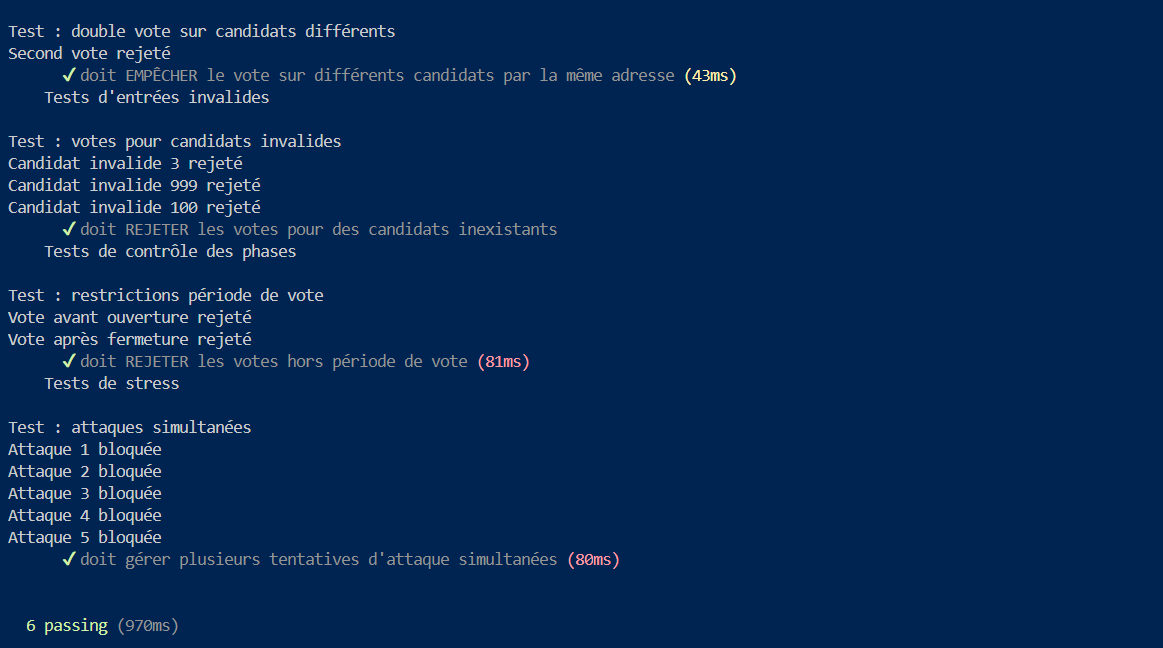
* **Test** **principal** : Empêcher de voter avant l’ouverture et après la fermeture.
* **Pourquoi** **c’est** **important** : Assure que le vote respecte les règles temporelles définies.

5. **Tests de stress / attaques simultanées**

**Objectif** : Tester la résistance du contrat face à des attaques ou des actions concurrentes.

* **Test** **principal** : Lancer plusieurs votes simultanés depuis la même adresse pour vérifier que tous échouent.
* **Pourquoi** **c’est** **important** : Évite les problèmes d’intégrité des votes en cas d’attaques ou de transactions concurrentes.





Test final de tous les fichiers

Commande : Truffle test

