جامعة عبد المالك السعدي
Université Abdelmalek Essaâdi

كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

# ROYAUME DU MAROC
# UNIVERSITÉ ABDELMALEK ESSAADI
## FACULTÉ DES SCIENCES ET TECHNIQUES - TANGER

CYCLE MASTER EN SCIENCES ET TECHNIQUES

FILIÈRE : INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNÉES

# BLOCKCHAIN

## (TWITTER DAPP)


**RÉALISÉE PAR :**
NOUIH Omar

**ENCADRÉ PAR :**
Pr. BENADBELOUAHAB Ikram


IASD
MASTER INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNÉES

**Année Universitaire 2024 - 2025**

# Introduction

Ce rapport présente le développement de l'application décentralisée (DApp) *Mini Twitter DApp*, qui exploite la technologie blockchain. Ce projet a été réalisé dans le cadre de l'Atelier 5 et utilise Solidity pour le développement de contrats intelligents, ainsi que HTML, JavaScript et Web3.js pour l'interface utilisateur.

# Technologies Utilisées

Le développement de *Mini Twitter DApp* repose sur les technologies suivantes :

- **Solidity**: Langage de programmation utilisé pour écrire des contrats intelligents sur la blockchain Ethereum.
- **Web3.js**: Bibliothèque JavaScript permettant d'interagir avec la blockchain Ethereum.
- **HTML/CSS/JavaScript**: Technologies pour le développement de l'interface utilisateur web.

# Contrat Intelligent

Le contrat intelligent constitue le backend de l'application *Mini Twitter DApp*. Il gère les fonctionnalités essentielles telles que la création, l'édition, le like et le dislike des publications.

## Fonctionnalités du Contrat

Le contrat offre les fonctionnalités suivantes :

- **publishPost**: Permet aux utilisateurs de créer de nouveaux posts.
- **editPost**: Donne la possibilité à l'auteur de modifier son post.
- **likePost** et **dislikePost**: Permettent aux utilisateurs de liker ou disliker des posts.
- **addComment**: Permet aux utilisateurs d'ajouter des commentaires sur les posts.
- **likeComment**: Autorise les utilisateurs à liker un commentaire.

## Exemple de Code

Voici un extrait du code du contrat intelligent :

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

contract MiniSocial {
    event NewPostEvent(uint indexed postId, string message, address indexed author, uint timestamp);
    event PostUpdatedEvent(uint indexed postId, string newMessage, uint lastModified);

    struct Comment {
        address author;
        string content;
        uint numLikes;
    }

    struct Post {
        uint id;
        string message;
        address author;
        uint numLikes;
        uint numDislikes;
        uint timestamp;
        uint lastModified;
        Comment[] comments;
    }
```

```
    Post[] private posts;
    mapping(uint => mapping(address => bool)) private hasLiked;
    mapping(uint => mapping(address => bool)) private hasDisliked;

    function publishPost(string memory _message) public { ... }
    function editPost(uint _postId, string memory _newMessage) public { ... }
    function likePost(uint _postId) public { ... }
    function dislikePost(uint _postId) public { ... }
    function addComment(uint _postId, string memory _content) public { ... }
    function likeComment(uint _postId, uint _commentId) public { ... }
}
```

# Développement Frontend

L'interface frontend permet aux utilisateurs d'interagir avec le contrat intelligent. Elle inclut des fonctions pour se connecter à MetaMask, créer, éditer, liker, disliker et visualiser des posts.

## Code HTML et JavaScript

Voici un extrait du code HTML et JavaScript utilisé pour l'interface :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Mini Twitter DApp</title>
    <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
</head>
<body>
    <h1>Mini Twitter DApp</h1>
    <button id="connectWalletBtn" onclick="connectWallet()">Connect Wallet</button>
    <p id="userAddress"></p>
    <h2>Add a New Post</h2>
    <textarea id="postMessage" placeholder="What's on your mind?" maxlength="140"></textarea>
    <button onclick="addPost()">Post</button>
    <h2>Feed</h2>
    <div id="feed"></div>

    <script type="text/javascript">
        let web3;
        let contract;
        const contractAddress = 'YOUR_CONTRACT_ADDRESS';
        const abi = [...];

        async function connectWallet() {
            if (window.ethereum) {
                web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: 'eth_requestAccounts' });
                contract = new web3.eth.Contract(abi, contractAddress);
                loadPosts();
            }
        }

        async function loadPosts() { ... }
        async function addPost() { ... }
        async function likePost(postId) { ... }
        async function dislikePost(postId) { ... }
        async function editPost(postId) { ... }
    </script>
</body>
</html>
```

Le code source complet du projet, y compris le contrat intelligent et l'application frontend, peut être trouvé dans le dépôt GitHub suivant :

https://github.com/OmarNouih/BLOCKCHAIN

# Déploiement et Tests

L'application a été déployée sur un réseau de test Ethereum local à l'aide de MetaMask et Ganache. Le contrat a été testé pour diverses fonctionnalités, notamment la publication, l'édition, le like et le commentaire des posts.

## Captures d'écran

Voici des captures d'écran de l'application en action.

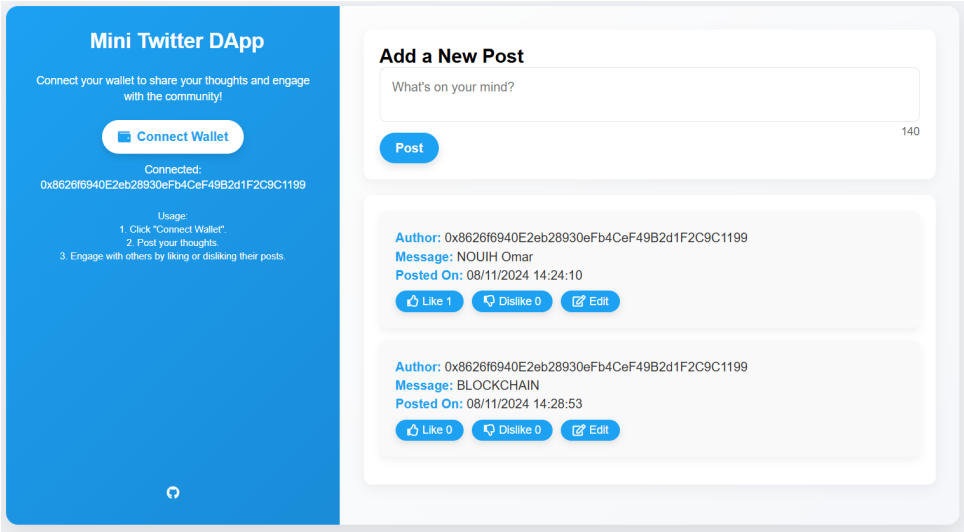


Figure 1: Capture d'écran de l'interface de la Mini Twitter DApp.

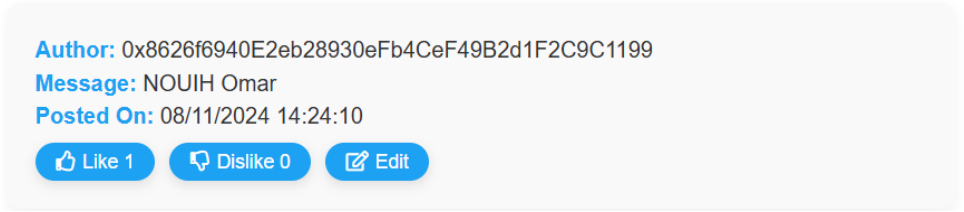


Figure 2: Exemple d'un post avec options de likes, dislikes et édition.

# Conclusion

Ce rapport présente le développement d'une DApp basée sur la blockchain, *Mini Twitter DApp*, qui utilise Solidity pour le développement de contrats intelligents et une interface web utilisant Web3.js. Cette application permet aux utilisateurs de créer des posts, de les éditer, d'ajouter des likes et des dislikes, ainsi que d'ajouter des commentaires, le tout stocké de manière immuable sur la blockchain.