EduChain

Plateforme d'Apprentissage Décentralisée basée sur la Blockchain

Rapport de Projet

Réalisé par :

Daoud ELLAILI
Salah-eddine ZITOUNI
Hamza SIHAMI
Mohammed EL GUERROUJ

16 mai 2025

Table des matières

Chapitre 1

Introduction

1.1 Présentation du projet

EduChain est une plateforme d'apprentissage décentralisée construite sur la blockchain Ethereum. Notre projet vise à révolutionner l'éducation en ligne en offrant un système transparent, sécurisé et sans intermédiaire pour créer, partager et certifier des connaissances.

La démocratisation de l'accès à l'éducation est au cœur de notre vision. En utilisant les technologies blockchain, nous avons créé un écosystème qui permet aux enseignants de partager leurs connaissances directement avec les apprenants, sans les frais d'intermédiaires traditionnellement associés aux plateformes d'apprentissage en ligne.

1.2 Vision et objectifs

Notre plateforme repose sur plusieurs objectifs fondamentaux :

— Démocratiser l'accès à l'éducation de qualité
— Éliminer les intermédiaires coûteux dans le processus éducatif
— Garantir la propriété intellectuelle des créateurs de contenu
— Créer un écosystème économique auto-suffisant pour l'apprentissage

1.3 Avantages clés

EduChain offre plusieurs avantages distinctifs par rapport aux plateformes d'apprentissage traditionnelles :

— **Transparence absolue :** Toutes les transactions et certifications sont enregistrées de manière immuable sur la blockchain
— **Désintermédiation complète :** Paiements directs des étudiants aux enseignants sans frais d'intermédiaire
— **Certification vérifiable instantanément :** Les diplômes et certificats peuvent être vérifiés par n'importe qui, n'importe quand
— **Protection de la propriété intellectuelle :** Droits des créateurs de contenu garantis par la blockchain
— **Système économique incitatif :** Récompenses en crypto-monnaie pour les enseignants et les apprenants

1.4 Public cible

Notre plateforme s'adresse à plusieurs catégories d'utilisateurs :

— Enseignants indépendants cherchant à monétiser leurs connaissances
— Étudiants à la recherche de certifications vérifiables
— Institutions éducatives souhaitant moderniser leur approche
— Professionnels en reconversion cherchant à acquérir de nouvelles compétences

Chapitre 2

Architecture du projet

```
testBlock/
└── educhain/
    ├── build/              # Contrats compilés
    ├── client/             # Application frontend
    │   ├── dist/           # Code compilé pour le déploiement
    │   ├── public/         # Fichiers statiques
    │   └── src/            # Code source frontend
    │       ├── components/ # Composants React
    │       ├── contexts/   # Contextes React
    │       ├── contracts/  # Intégration des contrats
    │       ├── pages/      # Pages de l'application
    │       └── utils/      # Utilitaires
    ├── contracts/          # Contrats intelligents Solidity
    │   └── EduChain.sol    # Contrat principal
    ├── migrations/         # Scripts de migration Truffle
    ├── test/               # Tests des contrats
    ├── package.json        # Dépendances du projet
    └── truffle-config.js   # Configuration Truffle
```

FIGURE 2.1 – Structure des dossiers

2.1 Vue d'ensemble

L'architecture d'EduChain repose sur un modèle en trois couches distinctes mais interconnectées, permettant une séparation claire des responsabilités tout en assurant une communication fluide entre les différents composants du système.

2.2 Couche blockchain (Backend)

La couche blockchain constitue le cœur de notre système :

— Smart contracts Solidity déployés sur la blockchain Ethereum

— Stockage décentralisé pour les métadonnées via IPFS (InterPlanetary File System)

— Oracle pour les interactions avec le monde extérieur

Cette couche assure l'immutabilité des transactions, la sécurité du système et la confiance entre les acteurs de la plateforme.

2.3 Couche middleware

La couche middleware sert d'interface entre la blockchain et l'application frontend :

— API Web3.js/Ethers.js pour interagir avec la blockchain

— Serveur Express.js pour les fonctionnalités hors chaîne

— Système de cache pour optimiser les performances

Cette couche permet d'abstraire la complexité de la blockchain pour les utilisateurs finaux tout en optimisant les performances du système.

2.4 Couche frontend

La couche frontend est l'interface utilisateur de notre plateforme :

— Application React avec Redux pour la gestion d'état

— Interface responsive adaptée mobile/desktop

— Intégration avec MetaMask et WalletConnect pour la gestion des wallets

2.5 Flux de données

Le flux de données dans notre architecture peut être représenté comme suit :

```
[Utilisateur] <-> [Interface React] <-> [Web3.js/API] <-> [Smart Contracts] <-> [
    Blockchain Ethereum] <-> [IPFS] <-> [Contenu d centralis ]
```

Ce schéma illustre comment les données circulent entre l'utilisateur et la blockchain, en passant par les différentes couches du système.

Chapitre 3

Smart Contracts

3.1 EduChain.sol

Le cœur de notre plateforme repose sur des smart contracts Solidity déployés sur la blockchain Ethereum. Le contrat principal, `EduChain.sol`, orchestre l'ensemble de la plateforme. Il est basé sur Solidity 0.8.0+ et utilise les bibliothèques OpenZeppelin pour garantir la sécurité et la conformité aux standards.

3.1.1 Structures de données principales

Course (Cours)

```
struct Course {
    uint256 id;                 // Identifiant unique du cours
    address teacher;            // Adresse Ethereum de l'enseignant
    string title;               // Titre du cours
    string description;         // Description d taill e
    uint256 price;              // Prix en wei
    string contentUrl;          // URL du contenu (IPFS ou autre)
    uint256 moduleCount;        // Nombre de modules dans le cours
    uint256 enrollmentCount;    // Nombre d' tudiants  inscrits
    uint256 creationTimestamp;  // Horodatage de cr ation
    bool isActive;              // Statut du cours (actif/inactif)
    mapping(address => bool) enrolledStudents; //  tudiants  inscrits
}
```

StudentProgress (Progression de l'étudiant)

```
struct StudentProgress {
    bool isEnrolled;                 // Statut d'inscription de l' tudiant
    uint256 enrollmentDate;          // Date d'inscription (timestamp)
    mapping(uint256 => bool) completedModules; // Modules compl t s (par leur ID)
    uint256 completedModuleCount;    // Nombre total de modules compl t s
    bool coursePassed;               // Statut de r ussite du cours
    uint256 lastActivityDate;        // Date de la derni re activit  (timestamp)
}
```

Certificate (Certificat)

```
struct Certificate {
    uint256 id;                 // Identifiant unique du certificat
    address student;            // Adresse de l' tudiant
    uint256 courseId;           // ID du cours concern
    uint256 timestamp;          // Date d' mission  (timestamp)
    bool issued;                // Statut d' mission  du certificat
    string metadataUrl;         // Lien vers les m tadonn es (IPFS ou autre)
    bytes signature;            // Signature cryptographique du certificat
}
```

3.1.2 Événements

Les événements permettent de suivre les actions importantes sur la blockchain :

```
event CourseCreated(
    uint256 indexed courseId,         // Identifiant du cours
    address indexed teacher,          // Adresse de l'enseignant
    string title,                     // Titre du cours
    uint256 price                     // Prix en wei
);

event StudentEnrolled(
    uint256 indexed courseId,         // Cours concern
    address indexed student,          //  tudiant  inscrit
    uint256 enrollmentDate            // Date d'inscription (timestamp)
);

event ModuleCompleted(
    uint256 indexed courseId,         // Cours concern
    address indexed student,          //  tudiant  concern
    uint256 moduleId                  // Identifiant du module compl t
);

event CourseCompleted(
    uint256 indexed courseId,         // Cours termin
    address indexed student           //  tudiant  ayant compl t  le cours
);

event CertificateIssued(
    uint256 indexed certificateId,    // ID du certificat
    address indexed student,          //  tudiant  concern
    uint256 indexed courseId          // Cours concern
);

event CourseUpdated(
    uint256 indexed courseId,         // Cours mis   jour
    string newTitle,                  // Nouveau titre
    uint256 newPrice                  // Nouveau prix
);

event TeacherPaid(
    address indexed teacher,          // Enseignant r mun r
    uint256 amount,                   // Montant vers
```

```
    uint256 indexed courseId          // Cours concern
);
```

3.1.3 Fonctions principales

Gestion des cours

```
function createCourse(
        string memory _title,
        string memory _description,
        uint256 _price,
        string memory _contentUrl,
        uint256 _moduleCount
    ) public returns (uint256) {
        require(_moduleCount > 0, "Course must have at least one module");

        _courseIds.increment();
        uint256 newCourseId = _courseIds.current();

        Course storage newCourse = courses[newCourseId];
        newCourse.id = newCourseId;
        newCourse.teacher = msg.sender;
        newCourse.title = _title;
        newCourse.description = _description;
        newCourse.price = _price;
        newCourse.contentUrl = _contentUrl;
        newCourse.moduleCount = _moduleCount;

        teacherCourses[msg.sender].push(newCourseId);

        emit CourseCreated(newCourseId, msg.sender, _title, _price);
        return newCourseId;
    }
```

Gestion des inscriptions

```
    function enrollInCourse(uint256 _courseId) public payable {
        Course storage course = courses[_courseId];
        require(course.id == _courseId, "Course does not exist");
        require(msg.value >= course.price, "Insufficient payment");
        require(!studentProgress[_courseId][msg.sender].isEnrolled, "Already
            enrolled");

        // Initialize the student progress
        StudentProgress storage progress = studentProgress[_courseId][msg.sender];
        progress.isEnrolled = true;
        progress.completedModuleCount = 0;
        progress.coursePassed = false;

        // Send payment to the teacher
        payable(course.teacher).transfer(msg.value);
```

```
15
16          // Add course to student's courses
17          studentCourses[msg.sender].push(_courseId);
18
19          emit CourseEnrollment(_courseId, msg.sender);
20      }
```

3.2 CourseFactory.sol

Le contrat `CourseFactory.sol` facilite la création de cours avec des modèles prédéfinis :

```
1  function createCourse(
2          string memory _title,
3          string memory _description,
4          uint256 _price,
5          string memory _contentUrl,
6          uint256 _moduleCount
7      ) public returns (uint256) {
8          require(_moduleCount > 0, "Course must have at least one module");
9
10         _courseIds.increment();
11         uint256 newCourseId = _courseIds.current();
12
13         Course storage newCourse = courses[newCourseId];
14         newCourse.id = newCourseId;
15         newCourse.teacher = msg.sender;
16         newCourse.title = _title;
17         newCourse.description = _description;
18         newCourse.price = _price;
19         newCourse.contentUrl = _contentUrl;
20         newCourse.moduleCount = _moduleCount;
21
22         teacherCourses[msg.sender].push(newCourseId);
23
24         emit CourseCreated(newCourseId, msg.sender, _title, _price);
25         return newCourseId;
26     }
```

3.3 CertificateManager.sol

Le contrat `CertificateManager.sol` gère les certificats avec vérification cryptographique :

```
1  function issueCertificate(uint256 _courseId) public onlyEnrolledStudent(_courseId)
       {
2          StudentProgress storage progress = studentProgress[_courseId][msg.sender];
3          require(progress.coursePassed, "Course not completed yet");
4
5          // Check if student already has a certificate for this course
6          for (uint256 i = 0; i < studentCertificates[msg.sender].length; i++) {
7              uint256 certId = studentCertificates[msg.sender][i];
8              if (certificates[certId].courseId == _courseId) {
```

```
                revert("Certificate already issued");
            }
        }

        _certificateIds.increment();
        uint256 newCertificateId = _certificateIds.current();

        certificates[newCertificateId] = Certificate({
            id: newCertificateId,
            student: msg.sender,
            courseId: _courseId,
            timestamp: block.timestamp,
            issued: true
        });

        studentCertificates[msg.sender].push(newCertificateId);

        emit CertificateIssued(newCertificateId, msg.sender, _courseId, block.
            timestamp);
    }
```

Chapitre 4

Interface utilisateur et fonctionnalités

4.1 Pour les enseignants

4.1.1 Création d'un compte enseignant

Pour devenir enseignant sur la plateforme EduChain, les utilisateurs doivent suivre un processus simple mais complet :

1. Accéder à l'application via l'interface web
2. Se connecter avec MetaMask
3. Naviguer vers "Profil" -> "Devenir enseignant"
4. Remplir le formulaire de profil enseignant comprenant :
   - Nom complet
   - Biographie
   - Domaines d'expertise
   - Liens professionnels (LinkedIn, GitHub, etc.)
   - Photo de profil (stockée sur IPFS)
5. Soumettre le formulaire (nécessite une transaction blockchain)
6. Attendre la confirmation de la transaction

4.1.2 Création et gestion des cours

Une fois leur compte enseignant activé, les utilisateurs peuvent créer des cours via l'interface graphique ou via des scripts utilitaires.

Via l'interface graphique :

1. Accéder à la section "Enseigner" -> "Nouveau cours"
2. Remplir le formulaire détaillé :
   - Titre du cours
   - Description courte et détaillée
   - Catégorie et sous-catégorie
   - Niveau de difficulté
   - Prix en ETH
   - Image de couverture

   - Prérequis et public cible
3. Configurer la structure du cours (modules, unités, etc.)
4. Ajouter le contenu du cours (texte, vidéos, documents, quizz)
5. Prévisualiser et publier le cours

Les enseignants disposent également d'un tableau de bord complet leur permettant de :

- Visualiser l'ensemble de leurs cours publiés
- Consulter les statistiques d'inscription et de complétion
- Suivre les revenus générés par cours et au total
- Accéder aux évaluations et commentaires des étudiants

4.2 Pour les étudiants

4.2.1 Exploration et inscription aux cours

Les étudiants peuvent facilement découvrir et s'inscrire à des cours qui correspondent à leurs besoins :

1. Explorer le catalogue de cours avec filtres avancés (catégorie, niveau, prix, évaluation)
2. Consulter les détails d'un cours (description, structure, prérequis, évaluations)
3. S'inscrire au cours en effectuant un paiement en ETH via MetaMask
4. Accéder immédiatement au contenu du cours après confirmation de la transaction

4.2.2 Suivi de la progression

La plateforme offre aux étudiants des outils de suivi de progression efficaces :

- Tableau de bord personnel affichant l'ensemble des cours suivis
- Vue détaillée de la progression pour chaque cours :
  - Pourcentage de complétion
  - Modules terminés vs restants
  - Temps passé sur le cours
  - Dernière activité
- Navigation intuitive entre les modules du cours
- Suivi automatique des modules complétés

4.2.3 Certification

Un des avantages majeurs d'EduChain est son système de certification basé sur la blockchain :

1. L'étudiant complète tous les modules du cours
2. Il demande son certificat via l'interface
3. La transaction est confirmée sur la blockchain
4. Le certificat numérique est généré avec :
   - Un identifiant unique sur la blockchain
   - Les informations du cours et de l'étudiant
   - Une signature cryptographique garantissant son authenticité
   - Un QR code de vérification
5. Le certificat peut être partagé et vérifié par n'importe qui

Chapitre 5

API et interfaces techniques

5.1 API Web3

Notre plateforme utilise Web3.js ou Ethers.js pour interagir avec la blockchain Ethereum.

5.1.1 Connexion au provider Ethereum

```
const initializeEthereum = useCallback(async () => {
  try {
    const detectedProvider = await detectEthereumProvider();

    if (detectedProvider) {
      console.log("MetaMask detected, initializing provider");
      const ethersProvider = new ethers.BrowserProvider(window.ethereum);
      setProvider(ethersProvider);

      // Get all accounts from MetaMask
      const allAccounts = await window.ethereum.request({
        method: 'eth_accounts',
      });

      // ...
    }
  } catch (err) {
    console.error("Ethereum initialization error:", err);
    setError('Failed to connect to Ethereum network.');
    setLoading(false);
  }
}, []);
```

5.1.2 Interaction avec les contrats

```
// Initialisation du contrat
const eduContract = new ethers.Contract(
  CONTRACT_ADDRESS,
  EduChainABI.abi,
  ethersProvider
);
```

```
7
8  // Cr ation d'un cours
9  export const createCourse = async (contract, signer, title, description, price,
       contentUrl, moduleCount) => {
10   try {
11     const contractWithSigner = contract.connect(signer);
12
13     // Fix price conversion - ensure it's properly formatted
14     let priceValue = price.toString().trim();
15
16     // Handle price conversions correctly
17     if (parseFloat(priceValue) > 10) {
18       priceValue = (parseFloat(priceValue) / 1000).toString();
19     }
20
21     const priceInWei = ethers.parseEther(priceValue);
22
23     const tx = await contractWithSigner.createCourse(
24       title,
25       description,
26       priceInWei,
27       contentUrl,
28       moduleCount
29     );
30
31     console.log('Transaction submitted:', tx.hash);
32
33     const receipt = await tx.wait();
34     console.log('Transaction receipt:', receipt);
35
36     // Handle various receipt formats to get the course ID
37     let courseId = '1';
38
39     if (receipt.events && Array.isArray(receipt.events)) {
40       const event = receipt.events.find(event => event.event === 'CourseCreated');
41       if (event && event.args) {
42         courseId = event.args.courseId.toString();
43       }
44     }
45
46     return courseId;
47   } catch (error) {
48     console.error("Error creating course:", error);
49     throw error;
50   }
51 };
52
53 // Inscription    un cours
54 const handleEnroll = async (courseId, price) => {
55   try {
56     setEnrolling(true);
57     setError('');
58
59     await enrollInCourse(contract, signer, courseId, price);
60
61     // Update the UI after enrollment
62     const enrolledCourse = availableCourses.find(course => course.id === courseId);
63     setEnrolledCourses([...enrolledCourses, enrolledCourse]);
```

```
64      setEnrolledCourseIds(new Set([...enrolledCourseIds, courseId]));
65      setAvailableCourses(availableCourses.filter(course => course.id !== courseId));
66
67      setEnrolling(false);
68    } catch (err) {
69      console.error('Error enrolling in course:', err);
70      setError('Failed to enroll in the course. Please try again.');
71      setEnrolling(false);
72    }
73  };
```

5.1.3 Événements et abonnements

```
1  //  couter  les nouveaux cours
2  const receipt = await tx.wait();
3  if (receipt.events && Array.isArray(receipt.events)) {
4      const event = receipt.events.find(event => event.event === 'CourseCreated');
5      if (event && event.args) {
6          courseId = event.args.courseId.toString();
7      }
8  }
9
10 //  couter  les inscriptions
11 const handleEnroll = async (courseId, price) => {
12   try {
13     setEnrolling(true);
14     setError('');
15
16     await enrollInCourse(contract, signer, courseId, price);
17
18     // Update the UI after enrollment
19     const enrolledCourse = availableCourses.find(course => course.id === courseId);
20     setEnrolledCourses([...enrolledCourses, enrolledCourse]);
21     // ...
22   } catch (err) {
23     console.error('Error enrolling in course:', err);
24     setError('Failed to enroll in the course. Please try again.');
25     setEnrolling(false);
26   }
27 };
28
29 //  couter  les certificats  mis
30 const handleGetCertificate = async () => {
31   try {
32     setProcessing(true);
33     setError('');
34     setSuccessMessage('');
35
36     const certificateId = await issueCertificate(contract, signer, id);
37     console.log('Certificate issued with ID:', certificateId);
38
39     setSuccessMessage('Certificate issued successfully! View it in your
          certificates page.');
40     // ...
41   } catch (err) {
```

```
    console.error('Error issuing certificate:', err);
    setError(`Failed to issue certificate: ${err.message || 'Unknown error'}`);
    setProcessing(false);
  }
};
```

5.2 Interface REST (Serveur Express)

En complément de l'interaction directe avec la blockchain, nous proposons une API REST pour certaines fonctionnalités :

```
GET    /api/courses                  # Liste des cours
GET    /api/courses/:id              # D tails d'un cours
GET    /api/teachers                 # Liste des enseignants
GET    /api/teachers/:address        # Profil d'un enseignant
GET    /api/students/:address/courses # Cours d'un  tudiant
GET    /api/certificates/:id         # V rification d'un certificat
POST   /api/ipfs/upload              # Upload vers IPFS
```

Chapitre 6

Tests et validation

6.1 Tests unitaires

Nos smart contracts sont rigoureusement testés à l'aide du framework Truffle et des outils OpenZeppelin pour les tests :

```
const EduChain = artifacts.require("EduChain");
const { expectRevert, expectEvent } = require('@openzeppelin/test-helpers');

contract("EduChain", accounts => {
  const [owner, teacher, student] = accounts;
  let eduChain;

  beforeEach(async () => {
    eduChain = await EduChain.new({ from: owner });
  });

  describe("Cr ation de cours", () => {
    it("permet    un utilisateur de cr er un cours", async () => {
      const tx = await eduChain.createCourse(
        "Titre test",
        "Description test",
        web3.utils.toWei("0.01", "ether"),
        "ipfs://test",
        3,
        { from: teacher }
      );

      expectEvent(tx, 'CourseCreated', {
        teacher: teacher,
        title: "Titre test"
      });

      const courseCount = await eduChain.getCourseCount();
      assert.equal(courseCount.toString(), "1", "Le compteur de cours devrait  tre
          incr ment ");

      const course = await eduChain.courses(1);
      assert.equal(course.teacher, teacher, "L'enseignant devrait  tre
         correctement enregistr ");
      assert.equal(course.title, "Titre test", "Le titre devrait  tre  correctement
          enregistr ");
    });
```

```
  });
});
```

6.2 Tests d'intégration

Nous effectuons également des tests d'intégration qui simulent le parcours complet d'un utilisateur, de la création de cours à l'obtention d'un certificat :

```
contract("Cycle de vie d'un cours", accounts => {
  const [owner, teacher, student1, student2] = accounts;
  let eduChain, courseFactory, certificateManager;
  let courseId;

  before(async () => {
    // D ploiement des contrats
    eduChain = await EduChain.new({ from: owner });
    courseFactory = await CourseFactory.new(eduChain.address, { from: owner });
    certificateManager = await CertificateManager.new(eduChain.address, { from:
        owner });

    // Configuration
    await eduChain.setCourseFactory(courseFactory.address, { from: owner });
    await eduChain.setCertificateManager(certificateManager.address, { from: owner
        });
  });

  it("1. Cr ation d'un cours", async () => {
    const tx = await eduChain.createCourse(
      "Cours complet",
      "Description d taill e du cours",
      web3.utils.toWei("0.05", "ether"),
      "ipfs://QmTestHash",
      5,
      { from: teacher }
    );

    const event = tx.logs.find(log => log.event === 'CourseCreated');
    courseId = event.args.courseId.toNumber();
    assert.isTrue(courseId > 0, "Le cours devrait avoir un ID valide");
  });

  it("2. Inscription des  tudiants ", async () => {
    // Test d'inscription...
  });

  // Tests suppl mentaires...
});
```

Chapitre 7

Captures d'écran

Les figures 1 à 15 illustrent la démonstration réalisée en local à l'aide de Ganache, tandis que les figures 16 à 27 présentent la visualisation de notre plateforme déployée sur le réseau de test Sepolia. Ces captures d'écran montrent les différentes étapes d'interaction avec le contrat intelligent via l'interface utilisateur. Enfin, les figures 28, 29 et 30 illustrent une démonstration effectuée sur Etherscan.

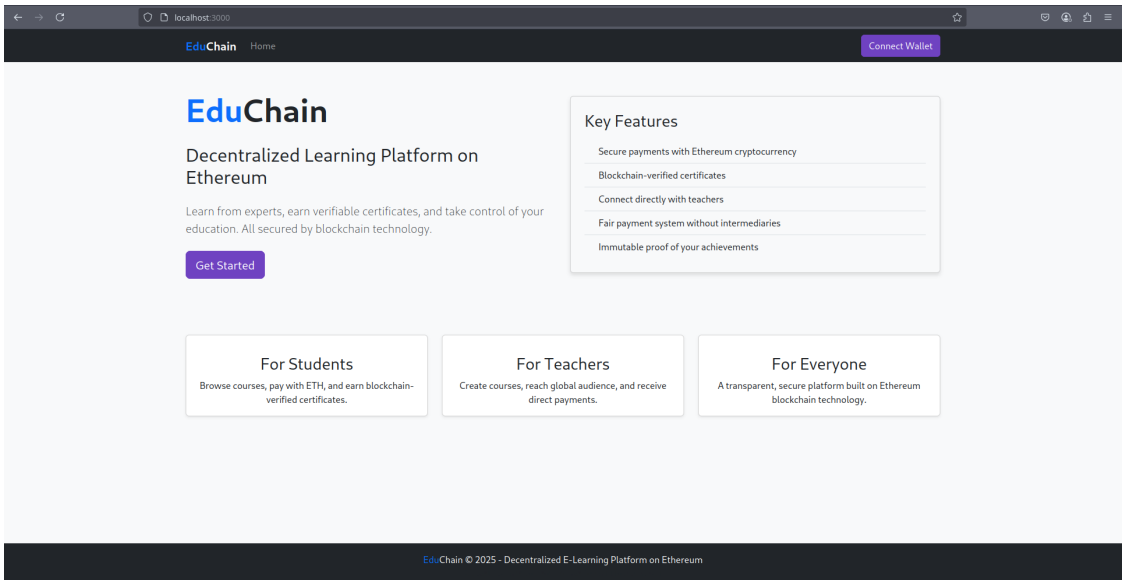

FIGURE 7.1 – Page d'accueil EduChain - Présentation de la plateforme

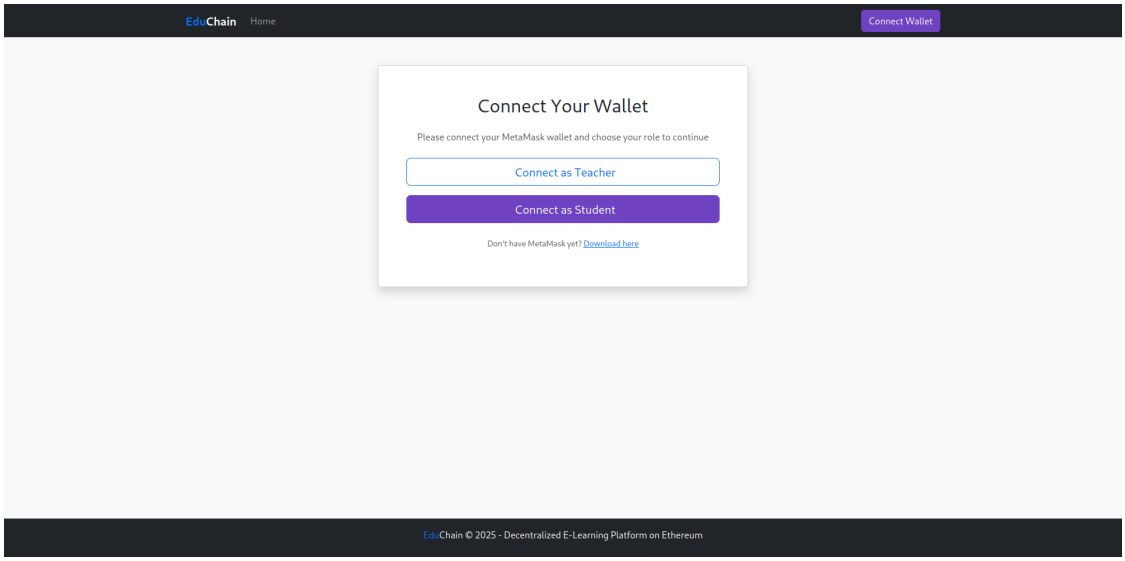

FIGURE 7.2 – Connexion au ptefeuille MetaMask - Sélection du rôle

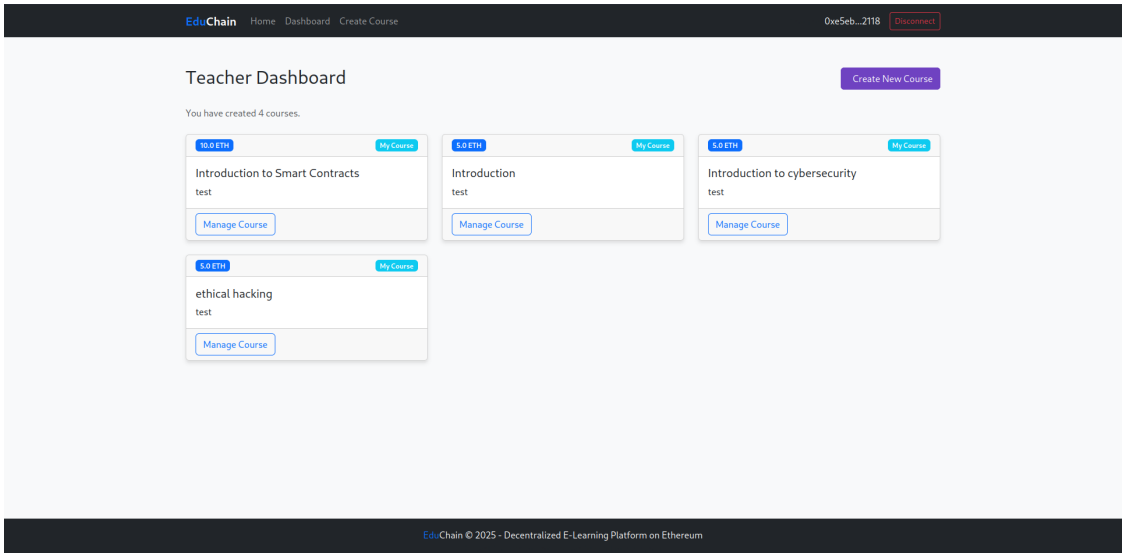

FIGURE 7.3 – Tableau de bord de l'enseignant - Affichage des cours créés avec options de gestion

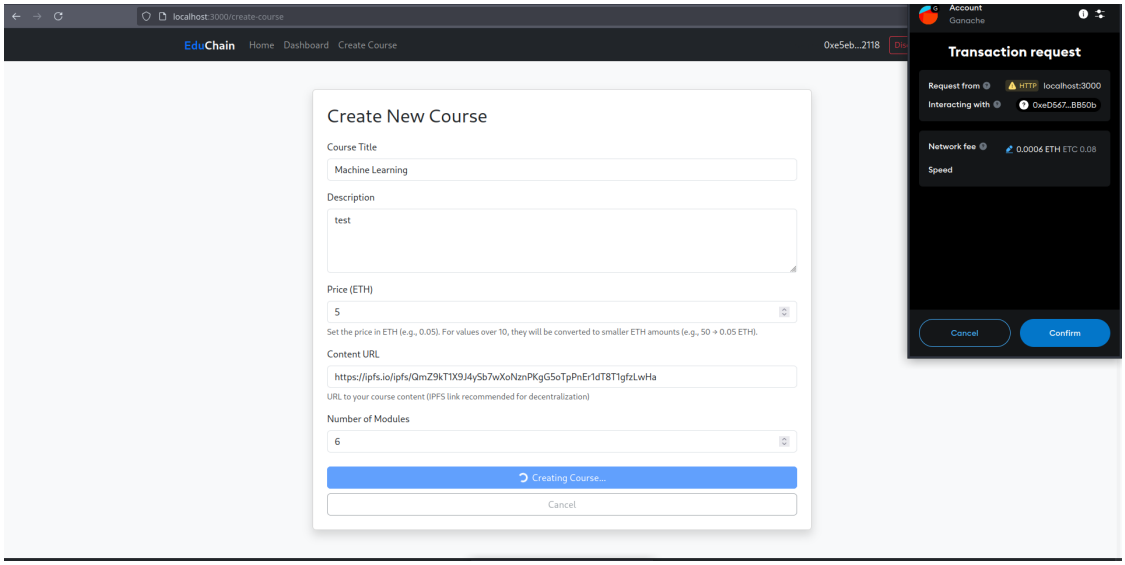

FIGURE 7.4 – Interface de création d'un nouveau cours

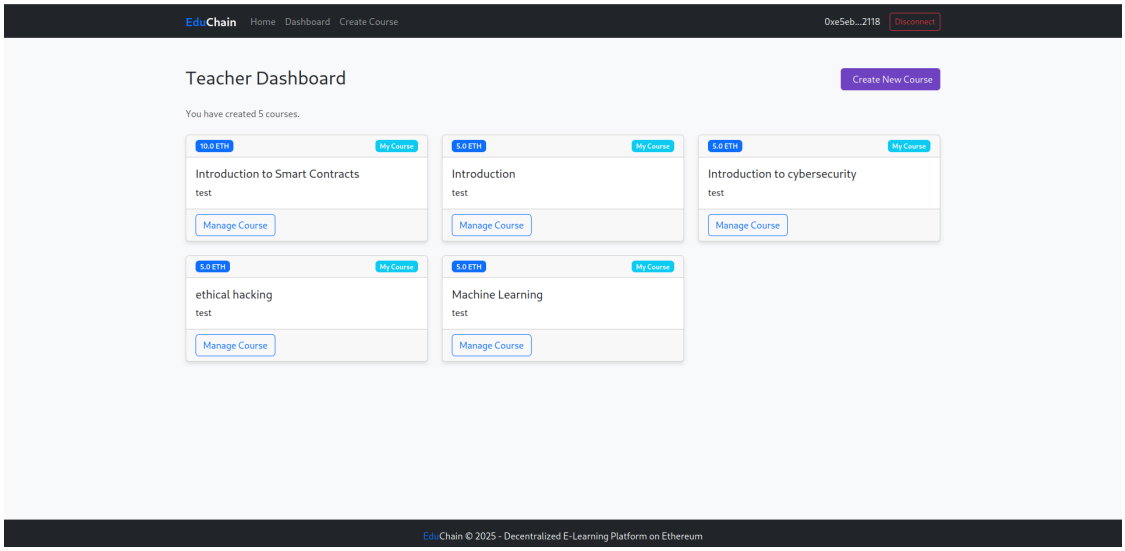

FIGURE 7.5 – Tableau de bord enseignant - Liste des cours créés

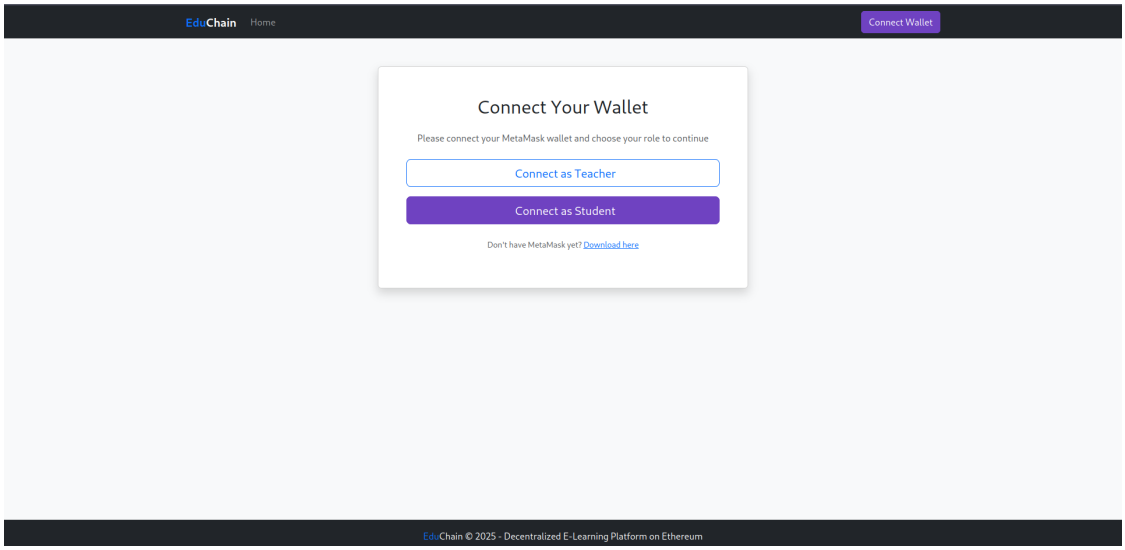

FIGURE 7.6 – Page de connexion principale - Choix du rôle

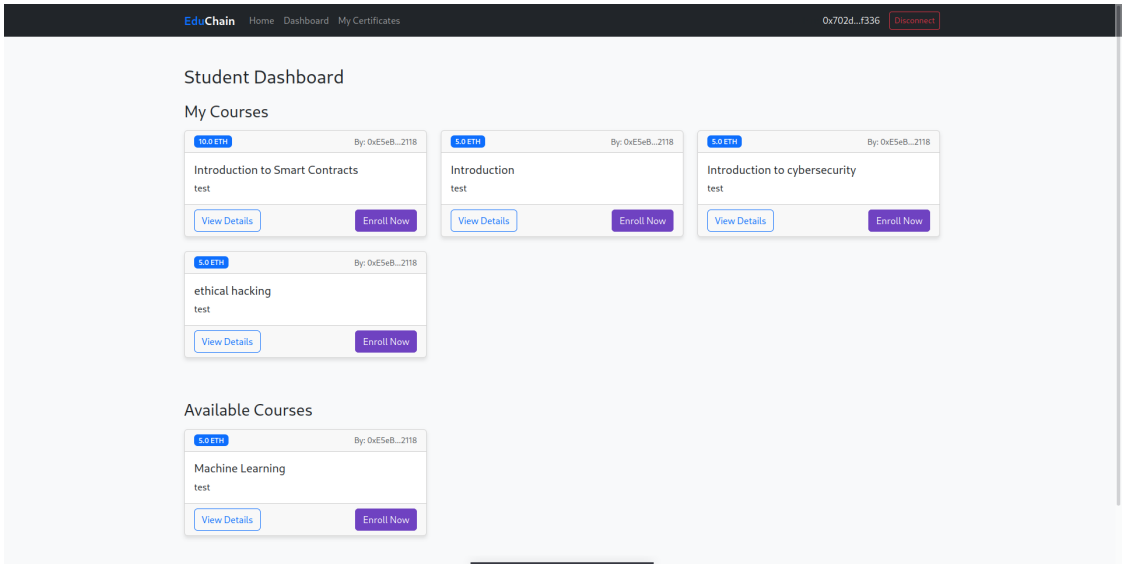

FIGURE 7.7 – Tableau de bord de l'étudiant - Présentation des cours disponibles et inscriptions

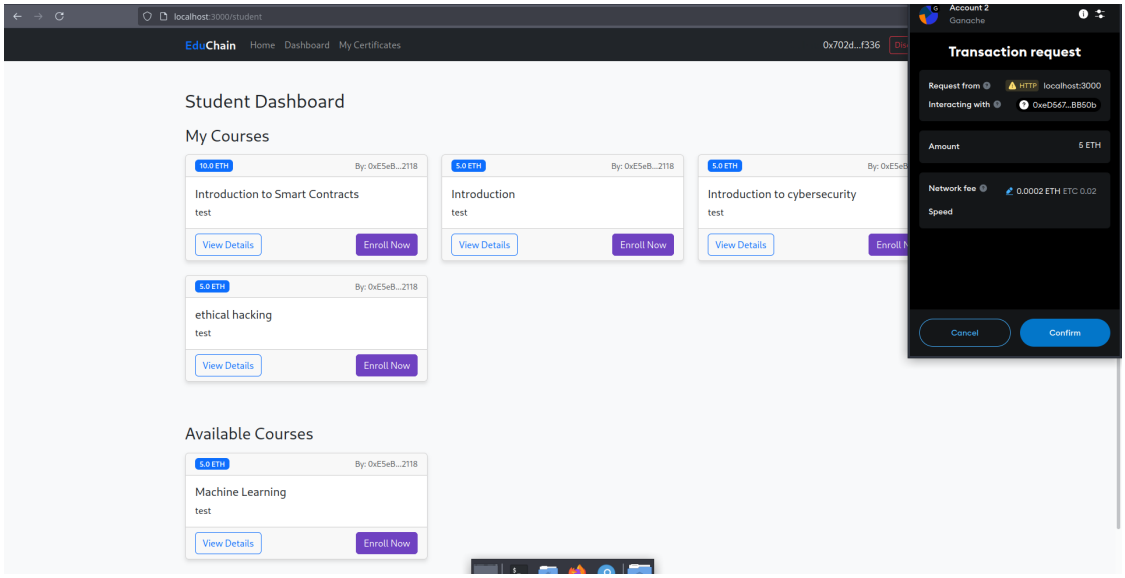

FIGURE 7.8 – Tableau de bord étudiant - Cours suivis et disponibles

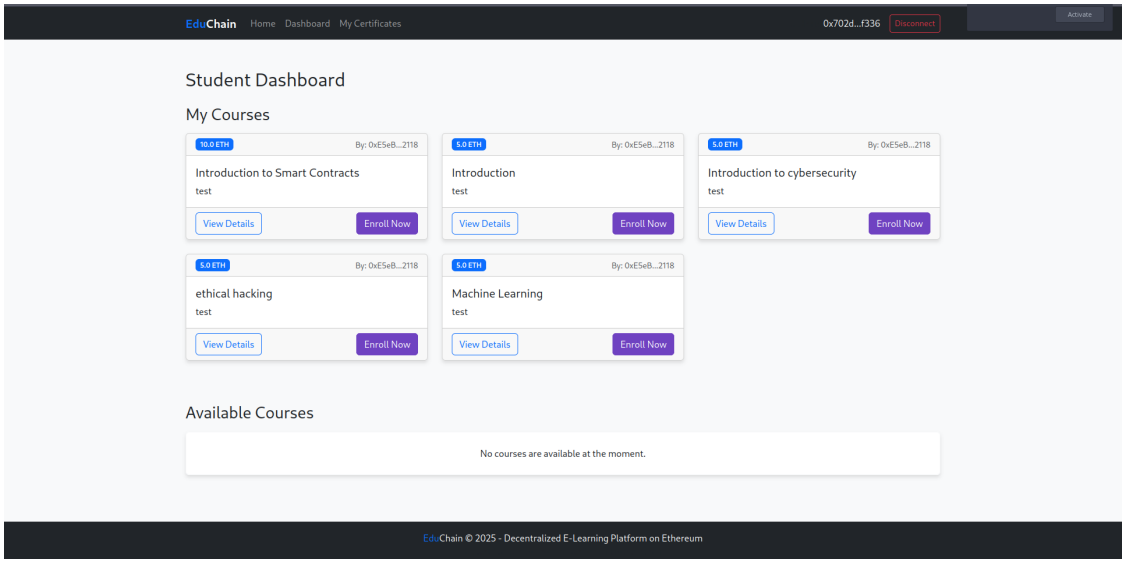

FIGURE 7.9 – Tableau de bord étudiant - Aperçu des cours et erreurs

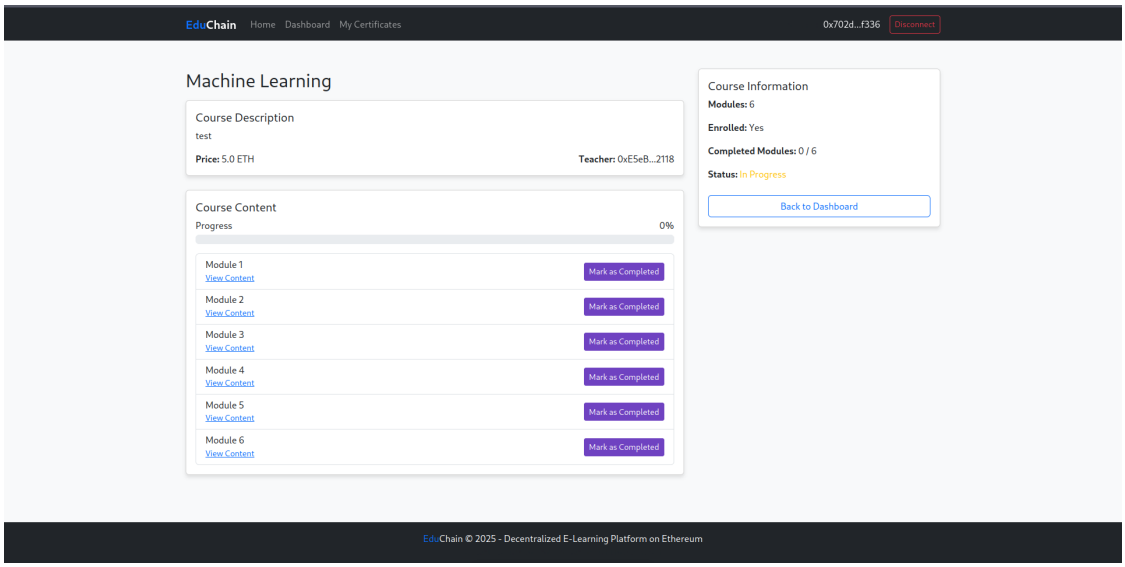

FIGURE 7.10 – Détails du cours 'Machine Learning' - Modules et progression

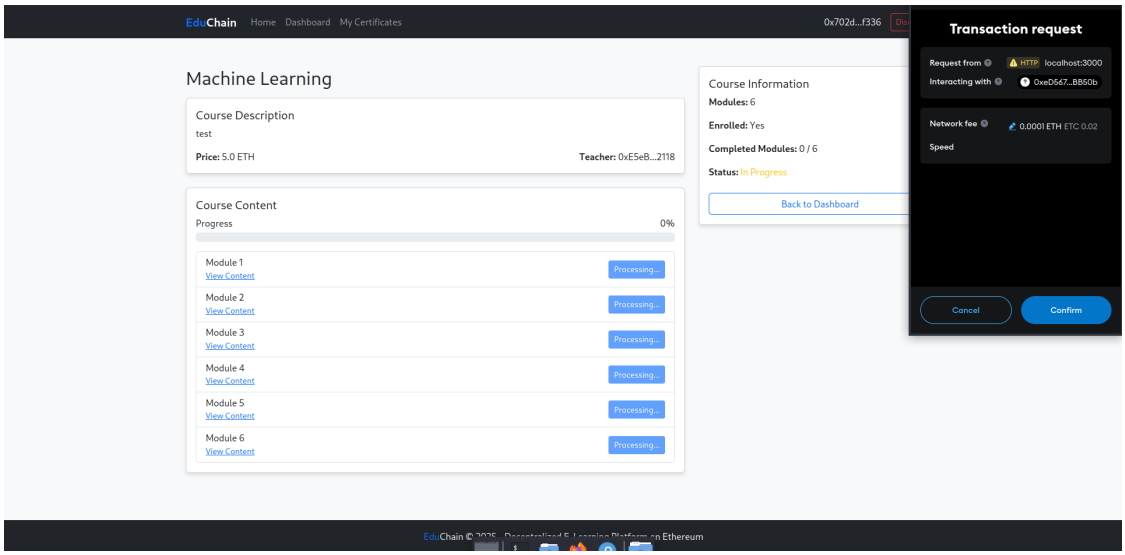

FIGURE 7.11 – Vue détaillée d'un cours avec processus d'inscription

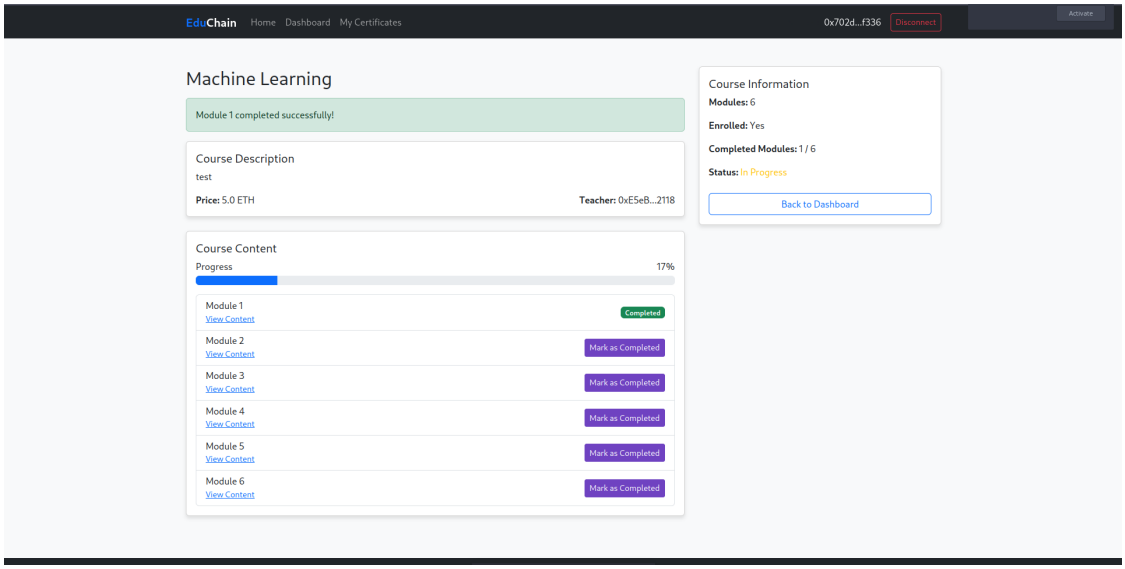

FIGURE 7.12 – Détails du cours 'Machine Learning' - Modules partiellement terminés et progression

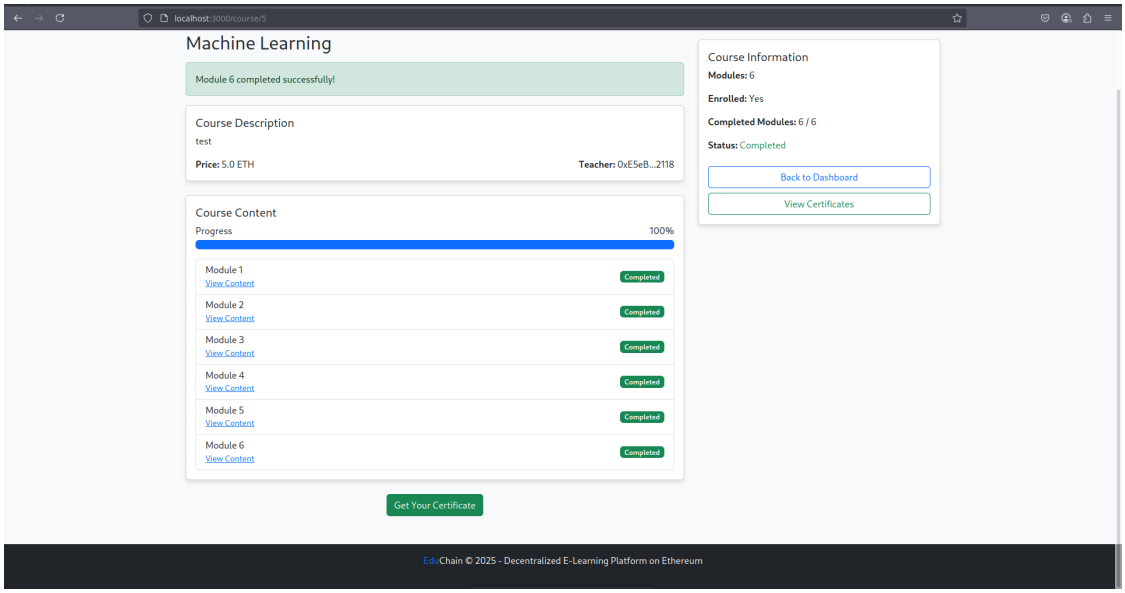

FIGURE 7.13 – Cours 'Machine Learning' terminé - Certification et tableau de bord final

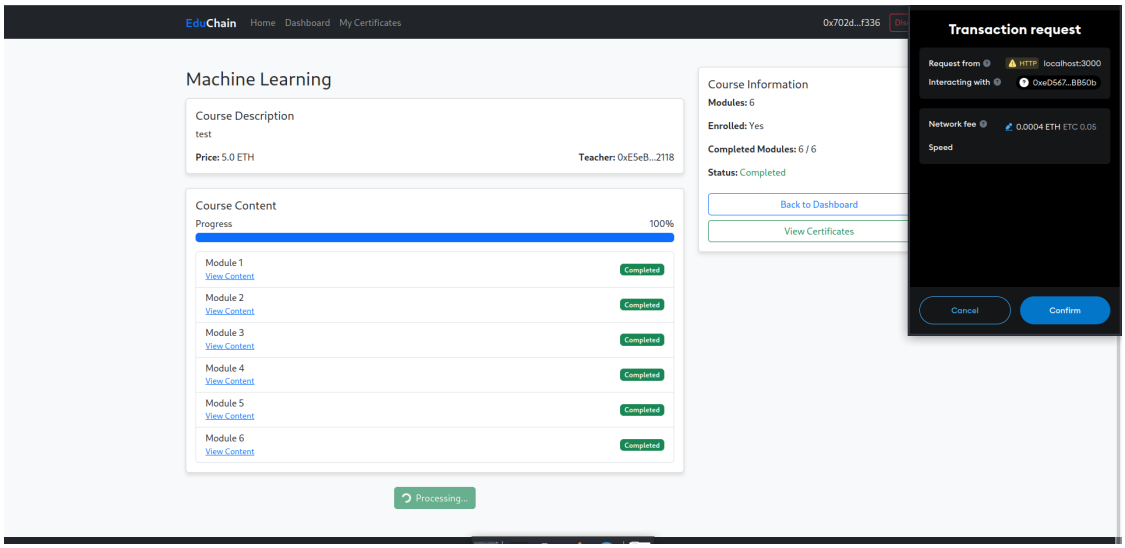

FIGURE 7.14 – Cours « Machine Learning » terminé – Obtention du certificat

FIGURE 7.15 – Portefeuille de certificats blockchain - Preuves de réussite sur Ethereum

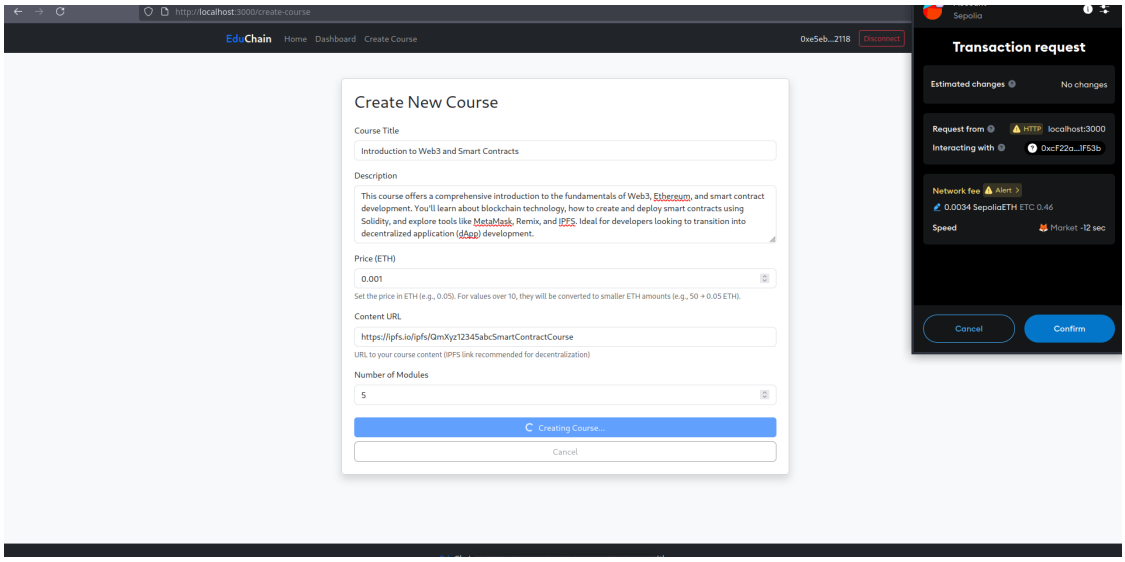

FIGURE 7.16 – Création d'un nouveau cours (Sepolia)

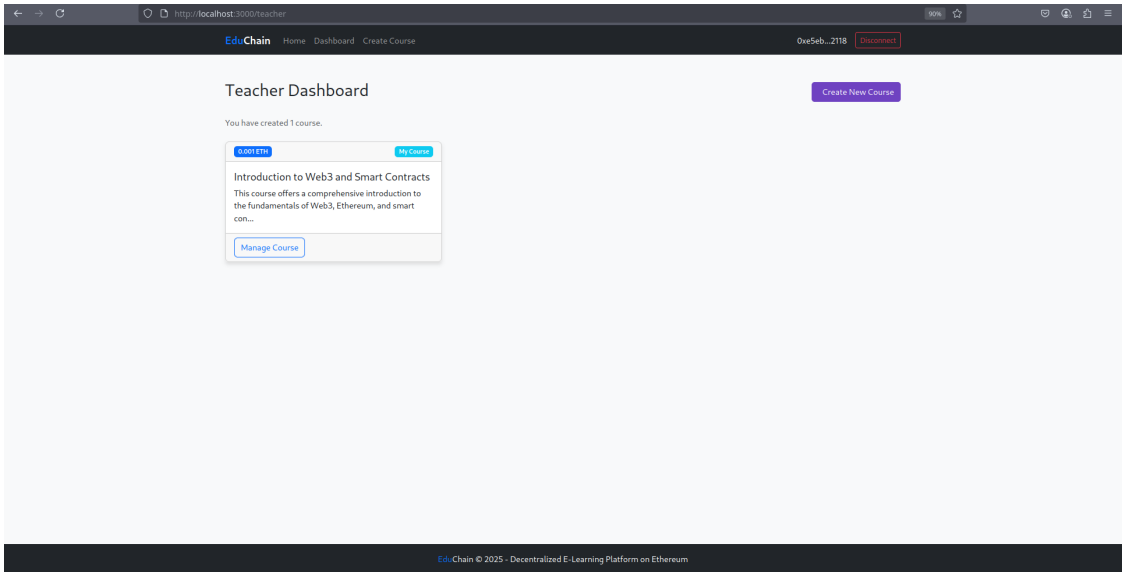

FIGURE 7.17 – Tableau de bord enseignant : gestion de cours

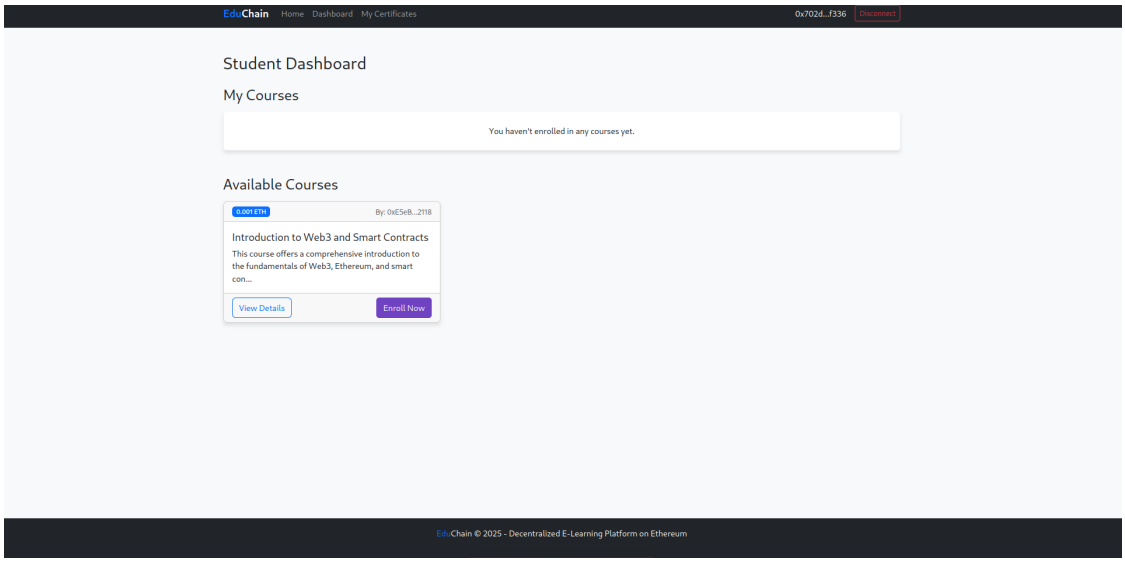

FIGURE 7.18 – Tableau de bord étudiant : cours disponibles

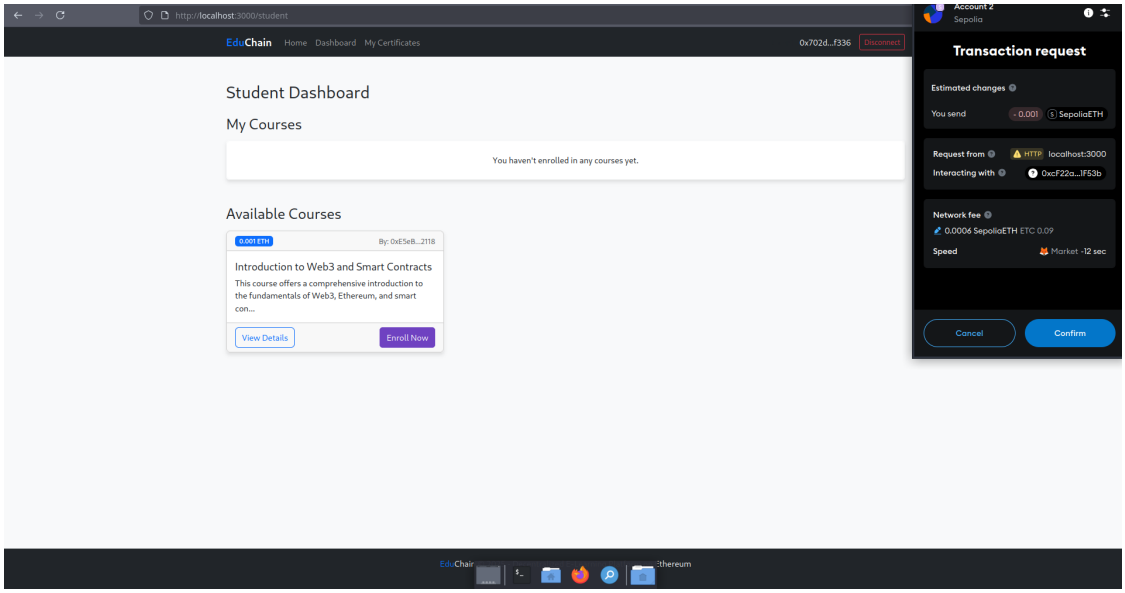

FIGURE 7.19 – Tableau de bord étudiant : demande de transaction

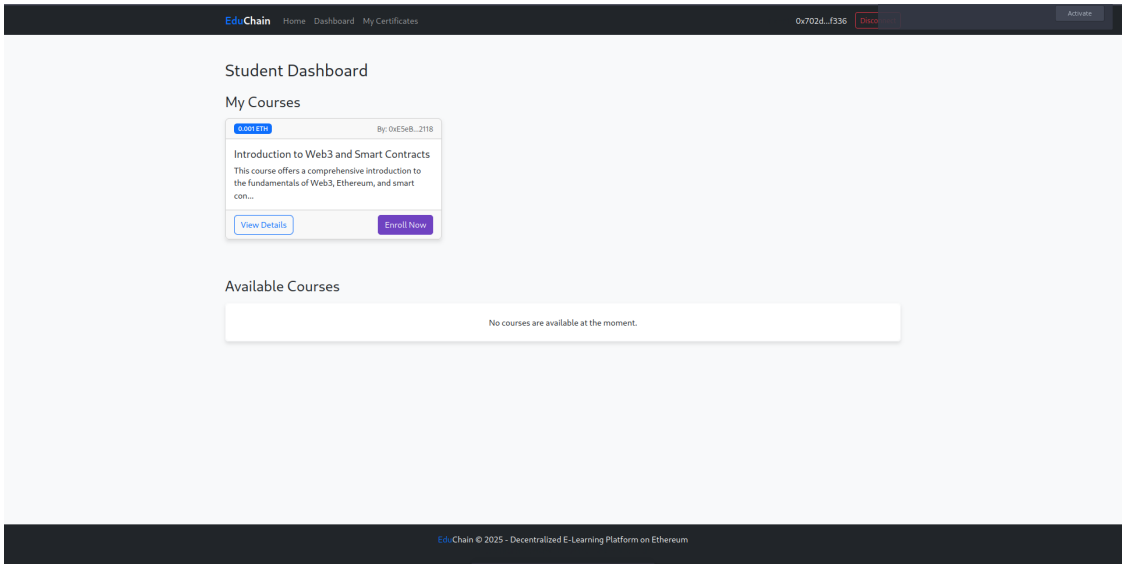

FIGURE 7.20 – Page de cours : détails et inscription

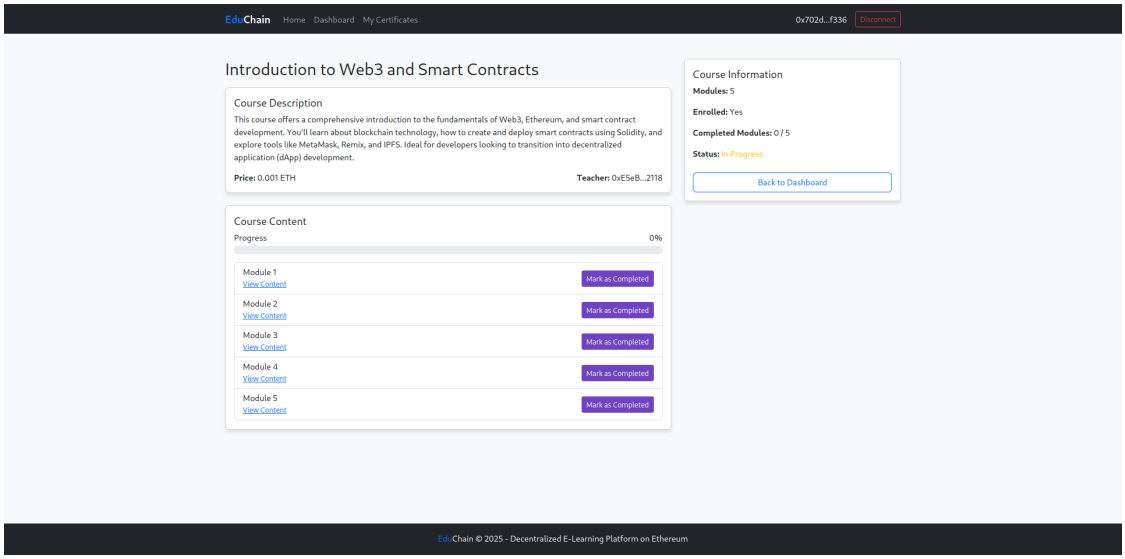

FIGURE 7.21 – Détails du cours : progression et modules

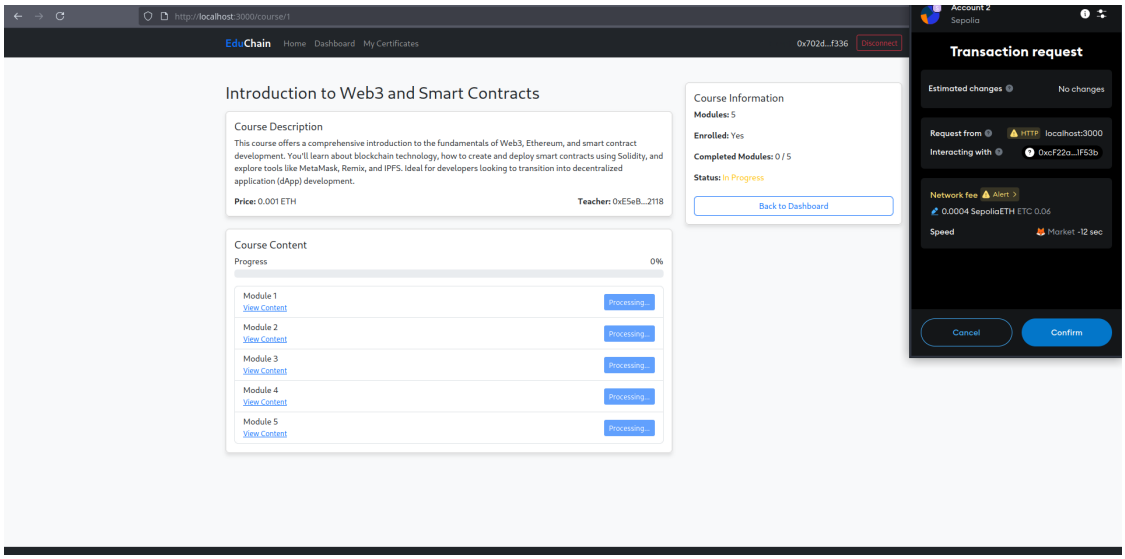

FIGURE 7.22 – Début de formation avec modules en cours de traitement (0%)

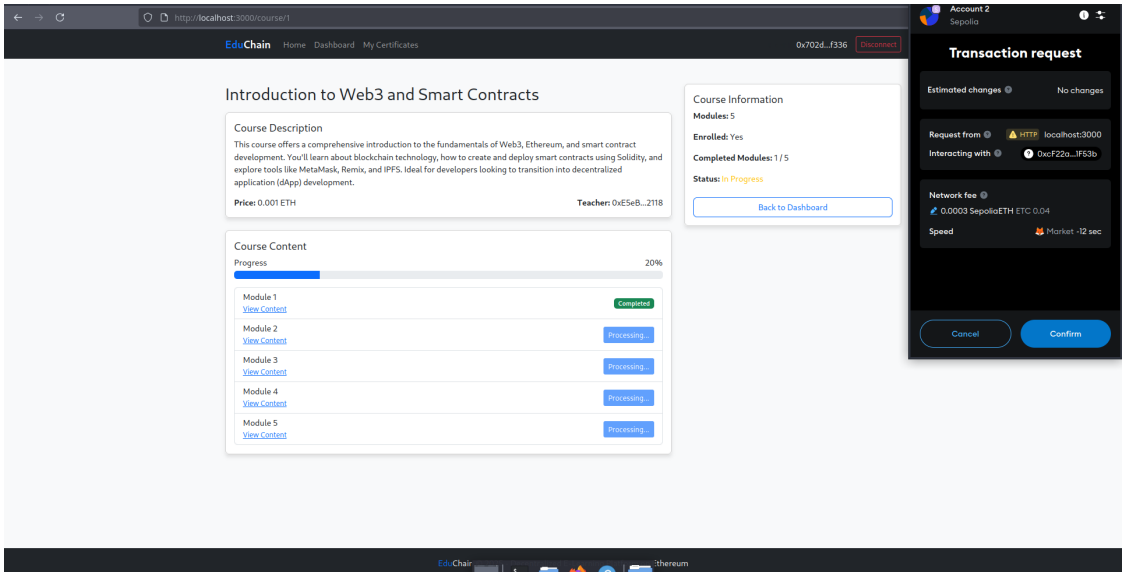

FIGURE 7.23 – Formation en cours avec un module complété (20%)

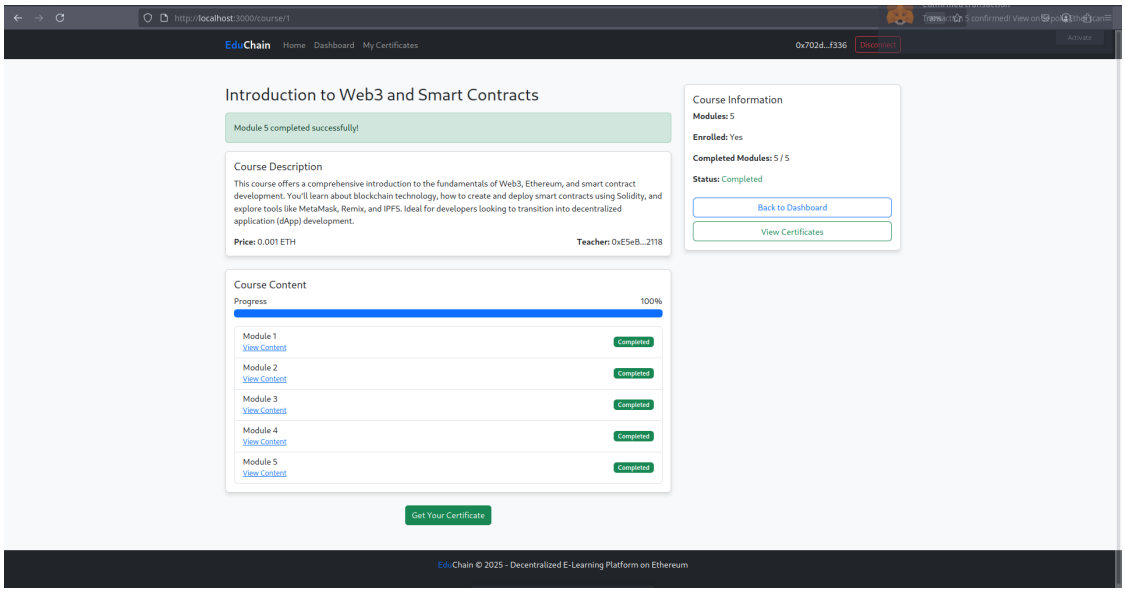

FIGURE 7.24 – Module 5 réussi : certificat disponible

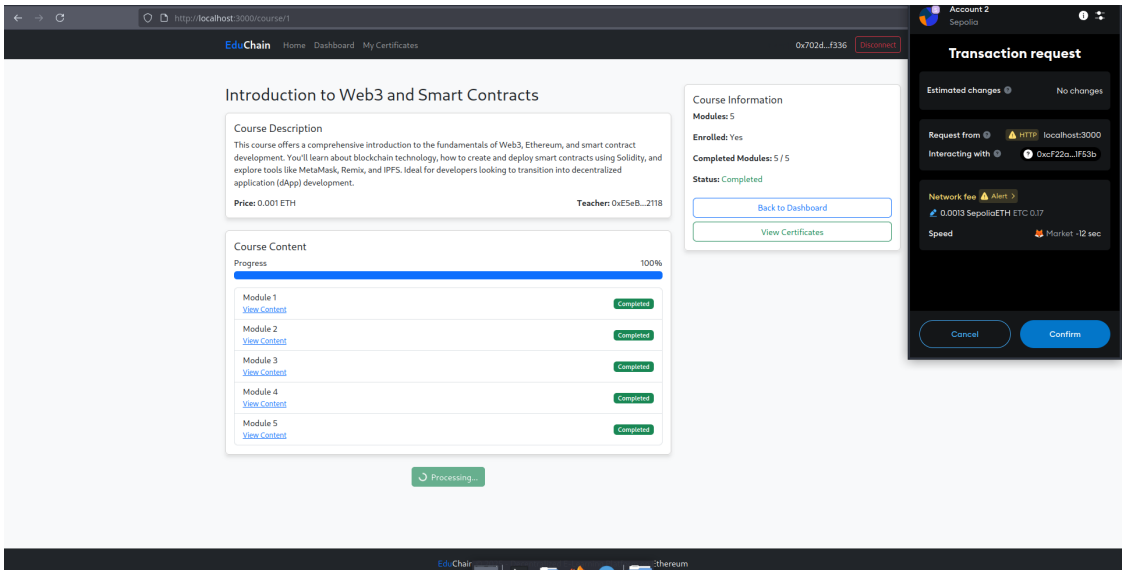

FIGURE 7.25 – Formation complétée avec tous les modules terminés (100%)

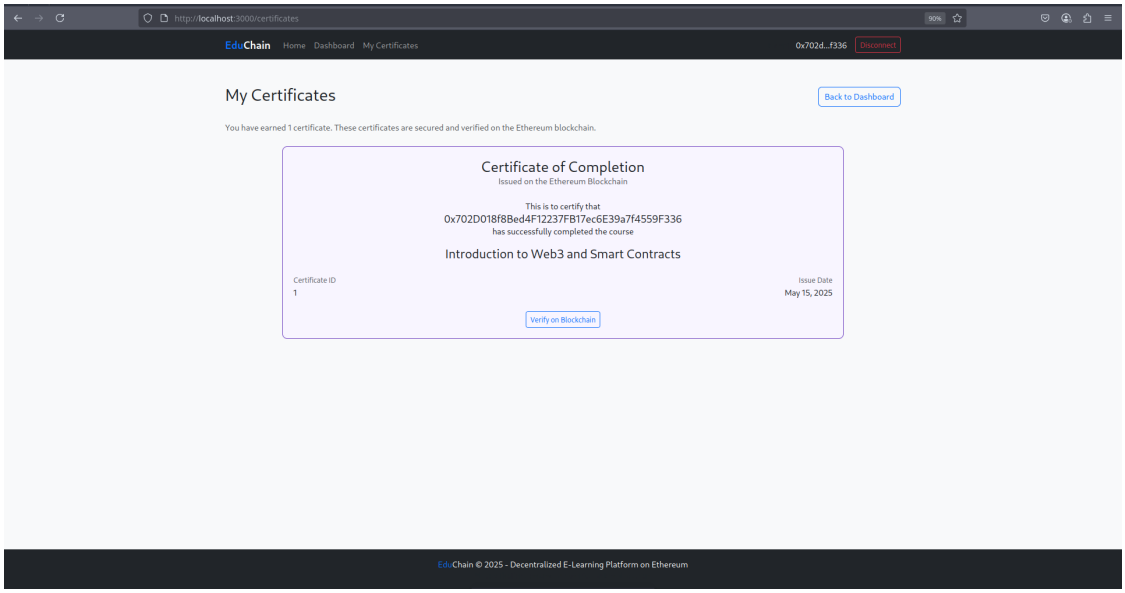

FIGURE 7.26 – Certificats obtenus - Vérification sur blockchain

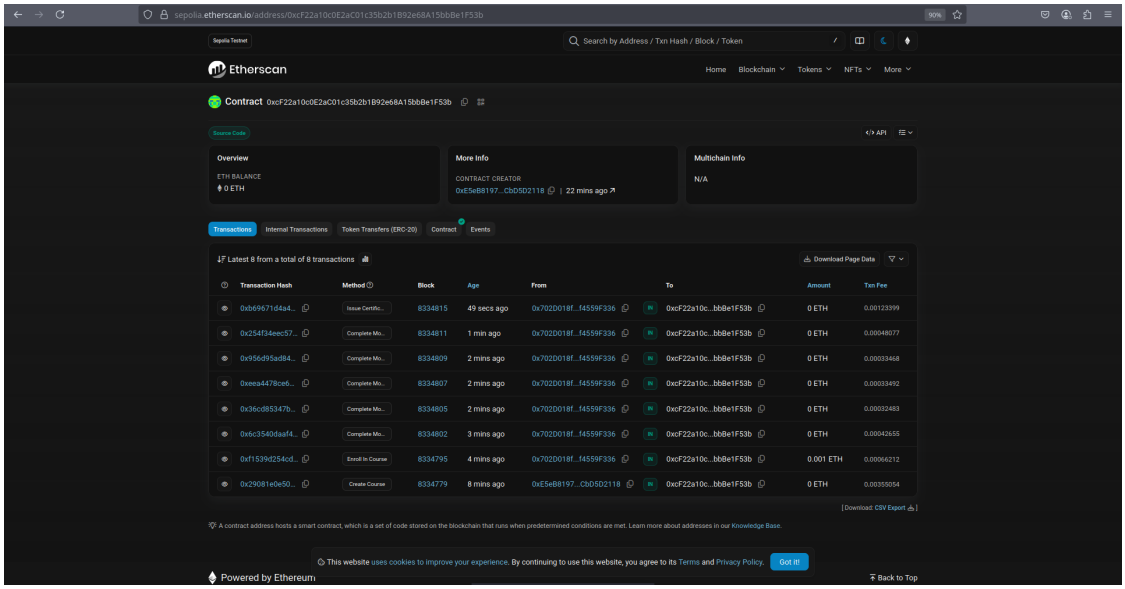

FIGURE 7.27 – Contrat intelligent : transactions sur Ethereum

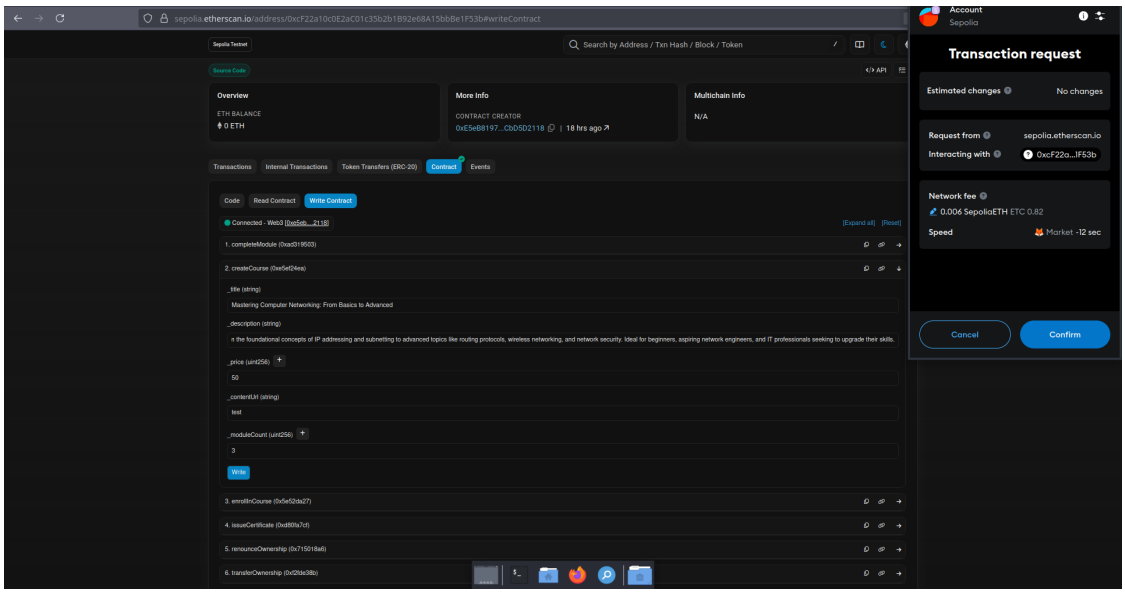

FIGURE 7.28 – Visualisation sur Etherscan — capture 1

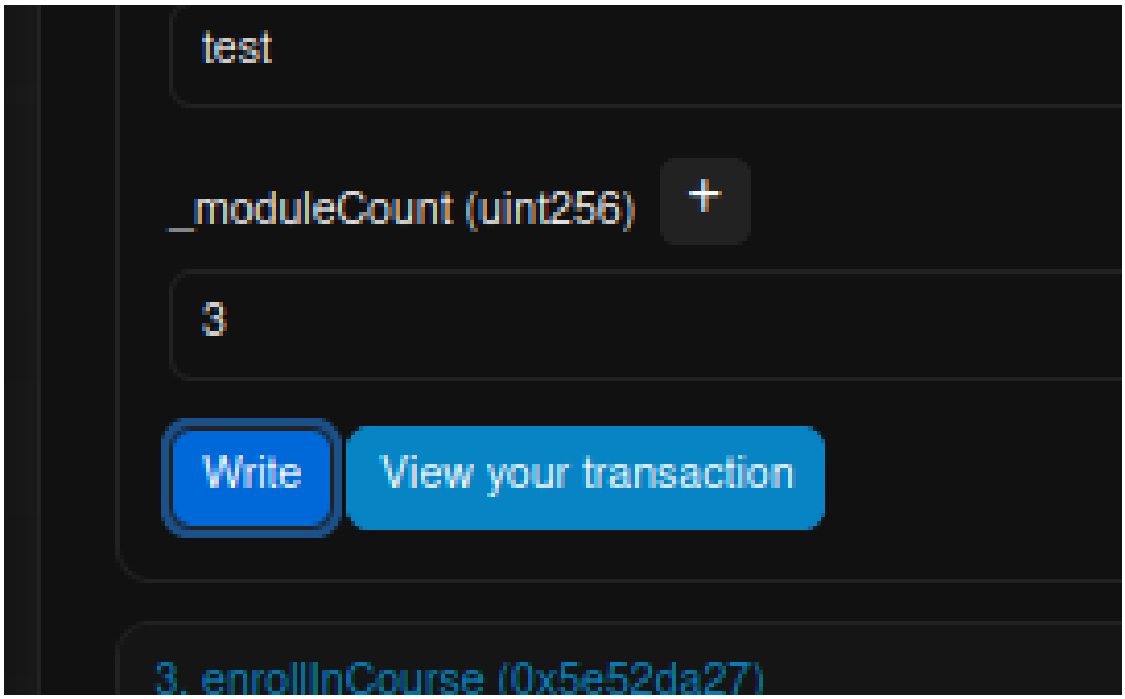

FIGURE 7.29 – Visualisation sur Etherscan — capture 2

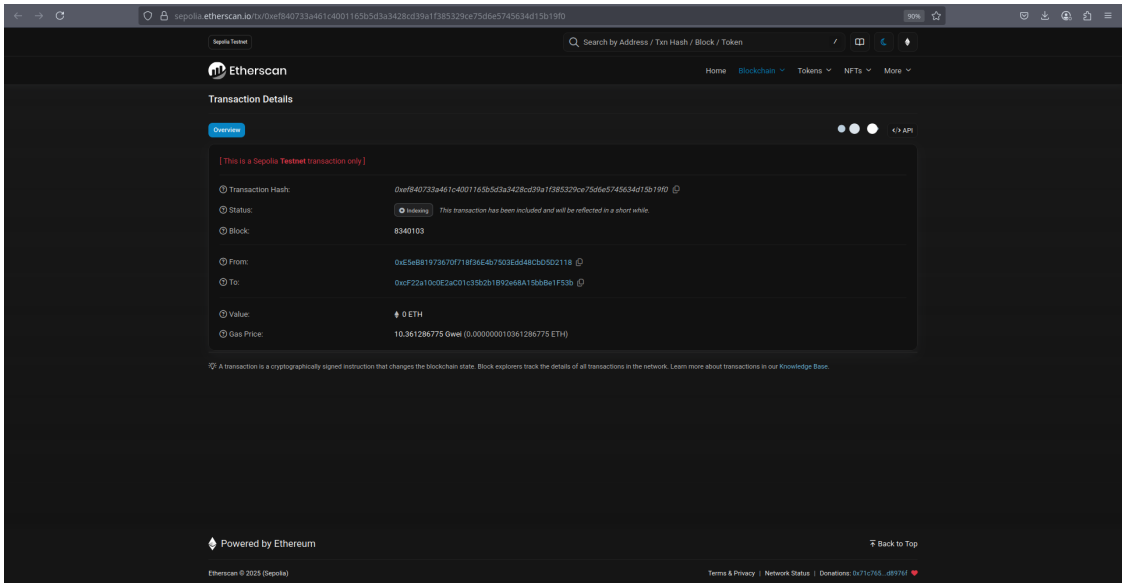

FIGURE 7.30 – Visualisation sur Etherscan — capture 3

Chapitre 8

Conclusion et perspectives

8.1 Bilan du projet

EduChain représente une approche innovante de l'éducation en ligne qui aligne les incitations des enseignants et des étudiants tout en éliminant les intermédiaires coûteux. Notre plateforme offre des avantages significatifs pour tous les acteurs de l'écosystème éducatif :

— **Pour les enseignants :** Revenus plus élevés, protection de la propriété intellectuelle, système de réputation transparent

— **Pour les étudiants :** Accès à des cours de qualité à prix réduits, certification vérifiable, propriété réelle de leurs réalisations

— **Pour l'écosystème éducatif :** Démocratisation de l'accès au savoir, réduction des barrières à l'entrée, innovation pédagogique

8.2 Perspectives d'évolution

Plusieurs pistes d'amélioration et d'extension sont envisagées pour EduChain :

— **Gouvernance DAO :** Mise en place d'un système de gouvernance décentralisée permettant à la communauté de participer aux décisions concernant l'évolution de la plateforme

— **Micro-accréditations :** Développement d'un système de badges et de micro-certifications pour des compétences spécifiques

— **Marketplace de ressources éducatives :** Création d'un marché où les enseignants peuvent échanger des ressources pédagogiques

— **Intégration de l'IA :** Utilisation de l'intelligence artificielle pour personnaliser l'apprentissage et recommander des cours pertinents

— **Expansion multi-chaînes :** Déploiement sur d'autres blockchains pour réduire les coûts et améliorer l'évolutivité

8.3 Mot de la fin

Ce projet nous a permis d'explorer les possibilités offertes par la blockchain dans le domaine de l'éducation. Nous sommes convaincus que la technologie blockchain peut transformer profondément la manière dont les connaissances sont partagées, certifiées et valorisées.