

**Sujet:**

## **Gestion des Risques de Contrepartie basé sur la Blockchain**



**Réalisé PAR :**

- Siham Yamlahi Alami

**Sous l'encadrement de :**

- Mr. Mohammad AHNOUCH

**Année Universitaire : 2024/2025**

# Table des matières

Introduction Générale.....	3
Chapitre1 : Contexte et Outils Utilisés.....	4
1. Introduction .....	5
2. Contexte du Projet.....	5
3. Outils et Technologies.....	5
3.1. Blockchain et Smart Contracts.....	5
3.2. Hardhat.....	5
3.3. MetaMask .....	6
4. Raison du Choix de Hardhat .....	7
Chapitre2 : Implémentation du Smart Contract .....	8
1. Introduction .....	9
2. Structure du Smart Contract.....	9
2.1. Structures de Données.....	9
2.2. Variables d'État .....	10
2.3. Événements .....	10
3. Fonctionnalités du Smart Contract.....	11
3.1. Ajout et Réactivation de Contreparties .....	11
3.2. Mise à jour des Expositions et Garanties .....	12
3.3. Gestion des Transactions .....	12
3.4. Calcul des Risques et Vérifications .....	13
4. Consultation des Données .....	14
5. Conclusion.....	15
Chapitre3 : Déploiement, Interface et Sécurité .....	16
1. Introduction .....	17
2. Déploiement et Configuration.....	17
2.1. Configuration de l'Environnement .....	17
3. Interface Utilisateur.....	18
3.1. Index.....	18
3.1. Create Counterparty.....	19
3.2. Admin .....	22
3.3. Transactions .....	25
3.4. Dashboard .....	28
4. Conclusion.....	29

Chapitre 4 : Questions d'Évaluation .....	30
1. Compréhension Technique.....	31
2. Compréhension de la Gestion des Risques .....	32
3. Implémentation et Innovation .....	33
Conclusion Générale .....	34

# Introduction Générale

La gestion des risques de contrepartie est un élément fondamental pour assurer la stabilité des marchés financiers. Elle consiste à évaluer et à suivre les risques associés à une partie avec laquelle une autre partie contracte une obligation financière. Lorsqu'une contrepartie ne respecte pas ses engagements, cela peut entraîner des pertes considérables, affectant ainsi l'ensemble du système économique. Cependant, les méthodes traditionnelles de gestion des risques reposent souvent sur des systèmes centralisés et manuels, ce qui les rend vulnérables aux erreurs humaines, aux fraudes et à la lenteur dans les processus de décision.

**Problématique :** Malgré les avancées technologiques dans d'autres domaines, la gestion des risques de contrepartie continue de souffrir d'un manque de transparence et d'automatisation. Les systèmes actuels, souvent rigides et manuels, ne sont pas suffisamment réactifs face à la complexité des transactions financières modernes. De plus, ces systèmes sont sujets à des erreurs humaines et à des vulnérabilités en matière de sécurité. Face à ces défis, l'adoption de la **blockchain** apparaît comme une solution prometteuse. Cette technologie permet de créer des systèmes décentralisés, sécurisés et transparents, où les données sont stockées de manière immuable, garantissant ainsi une traçabilité complète et une gestion plus fiable des risques.

**Objectif du projet :** Ce projet vise à développer un **système de gestion des risques de contrepartie basé sur la blockchain**, en utilisant des **smart contracts** pour automatiser la gestion des expositions, le calcul des risques et la gestion des limites de manière sécurisée. En remplaçant les processus manuels et centralisés par des solutions décentralisées et automatisées, ce système cherche à améliorer la transparence, la sécurité et l'efficacité des opérations financières, tout en réduisant le risque d'erreurs humaines et de fraudes.

Dans ce rapport, nous aborderons les différentes étapes de ce projet, depuis la conception et l'implémentation du **smart contract** jusqu'au déploiement de l'**interface utilisateur** pour la gestion des contreparties et des risques, en passant par les tests et les optimisations du système.

## **Chapitre1 : Contexte et Outils Utilisés**

### 1. Introduction

La gestion des risques de contrepartie est cruciale pour assurer la stabilité des transactions financières. Cependant, les systèmes traditionnels sont souvent vulnérables aux erreurs humaines et manquent de transparence. Ce projet vise à développer un système basé sur la **blockchain** pour automatiser la gestion des expositions et des risques. Grâce aux **smart contracts**, le système garantit la sécurité, la transparence et l'efficacité. L'objectif est d'améliorer la gestion des risques tout en réduisant les fraudes et erreurs.

### 2. Contexte du Projet

La gestion des risques de contrepartie est essentielle dans les transactions financières, car elle permet d'assurer que les parties respectent leurs obligations contractuelles. Les risques de contrepartie se produisent lorsque l'une des parties ne respecte pas ses engagements, ce qui peut entraîner des pertes financières importantes. Cependant, les systèmes traditionnels de gestion des risques de contrepartie sont souvent centralisés, manuels et vulnérables aux erreurs humaines. De plus, ces systèmes manquent souvent de transparence et de traçabilité, rendant difficile la gestion efficace de ces risques.

La **blockchain** offre une alternative puissante aux méthodes traditionnelles de gestion des risques en fournissant un registre décentralisé, transparent et immuable. Grâce à cette technologie, il est possible d'automatiser et de sécuriser les processus de gestion des risques de contrepartie, garantissant ainsi une meilleure transparence, une plus grande sécurité et une réduction des erreurs humaines. L'utilisation de **smart contracts** permet d'automatiser les calculs de risques, la gestion des expositions et de mettre à jour les informations de manière autonome.

Ce projet a pour but de créer un système de gestion des risques de contrepartie basé sur la blockchain. Le système repose sur l'utilisation de **smart contracts** pour automatiser les processus de gestion des risques, offrant ainsi une solution plus fiable et plus efficace.

### 3. Outils et Technologies

#### 3.1. Blockchain et Smart Contracts


- ❖ **Blockchain** : La blockchain est utilisée pour garantir la sécurité et la transparence des données. Toutes les transactions liées aux contreparties sont stockées de manière décentralisée, ce qui permet d'assurer l'intégrité et la traçabilité des informations sans risque de falsification.
- ❖ **Smart Contracts** : Les smart contracts sont des programmes qui s'exécutent automatiquement lorsqu'une condition spécifique est remplie. Dans ce projet, les smart contracts sont utilisés pour gérer l'ajout de contreparties, mettre à jour les expositions et calculer les risques en temps réel. Ces contrats sont conçus pour être autonomes et garantir que les processus sont exécutés de manière transparente et sécurisée, sans nécessiter d'intervention humaine.

#### 3.2. Hardhat

**Hardhat** est un framework de développement Ethereum qui permet d'écrire, tester et déployer des smart contracts de manière efficace. Il fournit un environnement local pour tester les contrats avant leur déploiement sur un réseau en production, comme **Polygon**. Hardhat offre également des outils pour optimiser les transactions et améliorer l'efficacité en termes de gas, ce qui permet de réduire les coûts de déploiement.

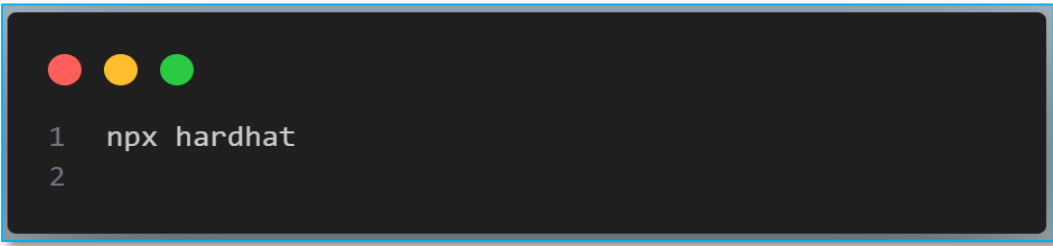
### Commandes principales

#### ❖ Installation de Hardhat :



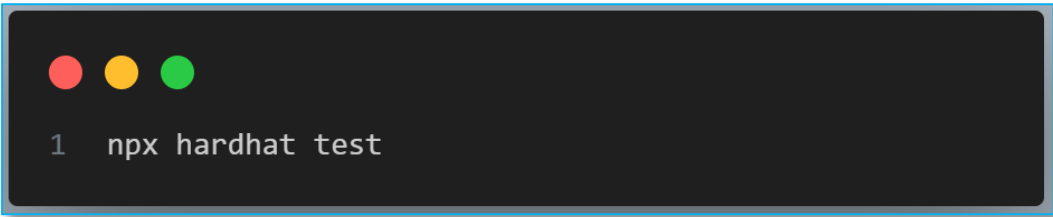
```
1  
2 npm install --save-dev hardhat  
3
```

#### ❖ Création d'un projet Hardhat



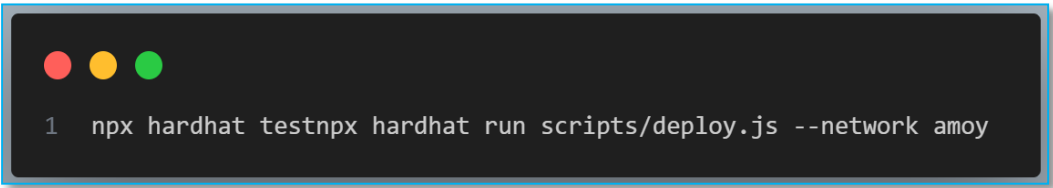
```
1 npx hardhat  
2
```

#### ❖ Lancer les tests :



```
1 npx hardhat test
```

#### ❖ Déployer le contrat sur un réseau



```
1 npx hardhat testnpx hardhat run scripts/deploy.js --network amoy
```

### 3.3. MetaMask

**MetaMask** est une extension de navigateur qui permet aux utilisateurs de gérer leur portefeuille Ethereum. Elle est utilisée pour se connecter à la blockchain, effectuer des transactions, et interagir avec la DApp via des smart contracts. MetaMask permet de signer des transactions de manière sécurisée et d'effectuer des actions, telles que la mise à jour des expositions ou l'ajout de contreparties, tout en garantissant que l'administrateur a un contrôle total sur le système.

#### 4. Raison du Choix de Hardhat

**Hardhat** a été choisi plutôt que **Remix** pour plusieurs raisons techniques. Remix est un IDE en ligne idéal pour de petites expérimentations ou des projets simples, mais pour un projet de cette envergure, Hardhat offre une plus grande flexibilité et des outils plus adaptés au développement de smart contracts professionnels. Hardhat permet de simuler un réseau blockchain local, d'effectuer des tests unitaires, et d'optimiser le gas, ce qui est essentiel pour garantir l'efficacité du système. De plus, Hardhat offre un contrôle total sur le processus de déploiement, avec la possibilité de connecter facilement des réseaux de test ou de production comme **Polygon**, ce qui est crucial pour ce projet.




## **Chapitre2 : Implémentation du Smart Contract**

## 1. Introduction

Dans ce chapitre, nous détaillons l'implémentation du smart contract **GestionnaireRisques**, conçu pour automatiser la gestion des risques de contrepartie. Ce contrat utilise des smart contracts sur la blockchain pour gérer les contreparties, suivre leurs expositions, et calculer les risques associés. Nous décrivons les différentes structures de données utilisées, ainsi que les fonctions qui permettent de gérer les expositions, les garanties, les transactions, et de calculer le risque en temps réel. Grâce à des mécanismes de sécurité et des vérifications des limites, ce contrat garantit une gestion transparente et sécurisée des risques financiers.

## 2. Structure du Smart Contract

Le smart contract **GestionnaireRisques** a été conçu pour gérer les risques de contrepartie en automatisant la gestion des expositions, des garanties et des risques.



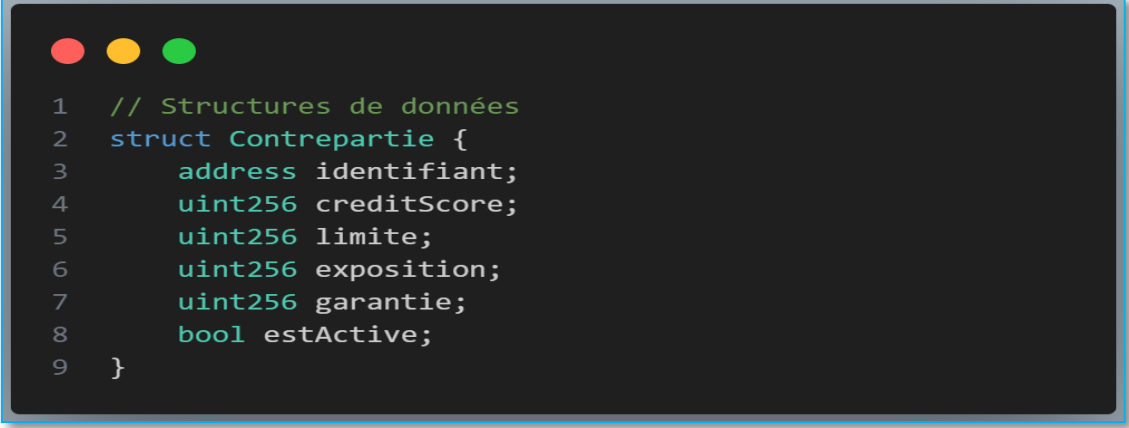
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract GestionnaireRisques {
```

Le contrat repose sur deux structures de données principales : **Contrepartie** et **EnregistrementTransaction**.

### 2.1. Structures de Données

#### ❖ Contrepartie :

Cette structure stocke les informations relatives à chaque contrepartie, telles que l'adresse, le score de crédit, la limite d'exposition, l'exposition actuelle, la garantie et l'état de la contrepartie.



```
1 // Structures de données
2 struct Contrepartie {
3     address identifiant;
4     uint256 creditScore;
5     uint256 limite;
6     uint256 exposition;
7     uint256 garantie;
8     bool estActive;
9 }
```

### ❖ EnregistrementTransaction :

Cette structure permet d'enregistrer les transactions entre expéditeur et récepteur, avec la valeur de la transaction et un horodatage.

```

1  struct EnregistrementTransaction {
2      address expéditeur;
3      address recepateur;
4      int256 valeur;
5      uint256 horodatage;
6  }
7

```

## 2.2. Variables d'État

Le contrat utilise plusieurs variables d'état pour stocker les informations relatives aux contreparties, aux transactions et aux relations d'exposition :

- **administrateur** : L'adresse Ethereum de l'administrateur du contrat.
- **contreparties** : Mapping associant l'adresse d'une contrepartie à ses données.
- **relationsExposition** : Mapping pour gérer l'exposition entre contreparties.
- **listeContreparties** : Tableau contenant toutes les contreparties enregistrées.
- **historique** : Tableau privé contenant l'historique des transactions effectuées.

```

1  // Variables d'état
2  address public administrateur;
3
4  mapping(address => Contrepartie) public contreparties;
5  mapping(address => mapping(address => int256)) public relationsExposition;
6
7  address[] public listeContreparties;
8  EnregistrementTransaction[] private historique;
9

```

## 2.3. Événements

Des événements sont utilisés pour notifier les actions importantes réalisées par le contrat, telles que l'ajout d'une contrepartie, la mise à jour de l'exposition ou la détection de risques élevés.

```

1  // Événements
2  event NouvelleContrepartie(address indexed identifiant, uint256 limite, uint256 creditScore);
3  event MiseAJourExposition(address indexed expéditeur, address indexed contrepartie, int256 nouvelleExposition);
4  event AlerteLimiteDepassee(address indexed contrepartie, uint256 expositionActuelle, uint256 limite);
5  event AlerteRisqueEleve(address indexed contrepartie, string typeAlerte, uint256 valeur);
6  event MiseAJourGarantie(address indexed contrepartie, uint256 nouvelleGarantie);
7  event NouvelleOperation(address indexed de, address indexed vers, int256 valeur, uint256 horodatage);
8  event ReactivationContrepartie(address indexed identifiant);
9

```

### 3. Fonctionnalités du Smart Contract

Le contrat **GestionnaireRisques** inclut plusieurs fonctionnalités permettant de gérer les contreparties, de suivre leurs expositions et de calculer les risques associés.

#### 3.1. Ajout et Réactivation de Contreparties

##### ❖ Ajouter une contrepartie :

Seul l'administrateur peut ajouter une nouvelle contrepartie en définissant son score de crédit, sa limite d'exposition et en initialisant son état à actif.

```

1 // Fonctions d'administration
2 function ajouterContrepartie(
3     address _identifiant,
4     uint256 _creditScore,
5     uint256 _limite
6 ) public seulementAdministrateur {
7     require(_identifiant != address(0), "Adresse invalide");
8     require(contreparties[_identifiant].identifiant == address(0), "Contrepartie deja existante");
9     require(_creditScore > 0, "Score de credit invalide");
10    require(_limite > 0, "Limite invalide");
11
12    contreparties[_identifiant] = Contrepartie({
13        identifiant: _identifiant,
14        creditScore: _creditScore,
15        limite: _limite,
16        exposition: 0,
17        garantie: 0,
18        estActive: true
19    });
20
21    listeContreparties.push(_identifiant);
22    emit NouvelleContrepartie(_identifiant, _limite, _creditScore);
23 }

```

##### ❖ Réactiver une contrepartie :

Cette fonction permet de réactiver une contrepartie qui a été désactivée, en vérifiant que son exposition ne dépasse pas la limite et que la garantie est suffisante.

```

1 function reactiverContrepartie(address _identifiant) public seulementAdministrat
eur {
2     Contrepartie storage c = contreparties[_identifiant];
3     require(!c.estActive, "Contrepartie deja active");
4     require(c.exposition <= c.limite, "Exposition depasse la limite");
5     require(calculerRatioGarantie(_identifiant) >= 50, "Garantie insuffisante");
6
7     c.estActive = true;
8     emit ReactivationContrepartie(_identifiant);
9 }
10

```

### 3.2. Mise à jour des Expositions et Garanties

#### ❖ Mise à jour de l'exposition :

Cette fonction permet de mettre à jour l'exposition d'une contrepartie, et déclenche un événement pour notifier le changement.

```
1 // Mise à jour des contreparties
2 function actualiserExposition(address _identifiant, uint256 ajoutExposition) public {
3     Contrepartie storage contrepartie = contreparties[_identifiant];
4     require(contrepartie.estActive, "Contrepartie inactive");
5
6     contrepartie.exposition += ajoutExposition;
7
8     emit MiseAJourExposition(msg.sender, _identifiant, int256(contrepartie.exposition));
9     verifierLimites(_identifiant);
10 }
```

#### ❖ Mise à jour de la garantie :

Cette fonction permet à l'administrateur de modifier la garantie associée à une contrepartie.

```
1 function actualiserGarantie(address _identifiant, uint256 nouvelleGarantie) public
  c seulementAdministrateur {
2     Contrepartie storage contrepartie = contreparties[_identifiant];
3     require(contrepartie.identifiant != address(0), "Contrepartie inexistante");
4
5     contrepartie.garantie = nouvelleGarantie;
6
7     emit MiseAJourGarantie(_identifiant, nouvelleGarantie);
8     verifierLimites(_identifiant);
9 }
```

### 3.3. Gestion des Transactions

#### ❖ Enregistrement des transactions :

Lors de chaque transaction entre contreparties, l'exposition est mise à jour et l'historique des transactions est conservé.

```

1 // Gestion des transactions
2 function enregistrerTransaction(
3     address recepateur,
4     int256 valeur
5 ) public contrepartieActive(msg.sender) contrepartieActive(recepateur) {
6     require(valeur != 0, "Valeur invalide");
7
8     int256 expositionActuelle = relationsExposition[msg.sender][recepateur];
9     expositionActuelle += valeur;
10    relationsExposition[msg.sender][recepateur] = expositionActuelle;
11
12    uint256 absValeur = uint256(valeur < 0 ? -valeur : valeur);
13    contreparties[msg.sender].exposition += absValeur;
14    contreparties[recepateur].exposition += absValeur;
15
16    historique.push(EnregistrementTransaction({
17        expéditeur: msg.sender,
18        recepateur: recepateur,
19        valeur: valeur,
20        horodatage: block.timestamp
21    }));
22
23    emit NouvelleOperation(msg.sender, recepateur, valeur, block.timestamp);
24
25    verifierLimites(msg.sender);
26    verifierLimites(recepateur);
27 }
28

```

### 3.4. Calcul des Risques et Vérifications

#### ❖ Calcul du risque :

Le risque est calculé en fonction de l'exposition actuelle de la contrepartie, de sa limite et de son score de crédit. Le calcul du risque est effectué selon la formule suivante:

$$\text{Score de Risque} = \frac{\text{Exposition Courante}}{\text{Limite d'Exposition}} \times \frac{100}{\text{Score de Crédit}}$$

Cette fonction retourne un indice de risque calculé.

```

1 // Calculs du risque et vérifications
2 function calculerRisque(address _identifiant) public view returns (uint256) {
3     Contrepartie storage c = contreparties[_identifiant];
4     require(c.limite > 0 && c.creditScore > 0, "Parametres invalides");
5     return (c.exposition * 1e4) / (c.limite * c.creditScore);
6 }

```

### ❖ Calcul du ratio de garantie :

Le ratio de garantie est calculé comme suit :

$$\text{Ratio de Couverture} = \frac{\text{Collatéral}}{\text{Exposition Totale}}$$

Si le ratio de garantie est inférieur à 50%, une alerte est émise.

```
1 function calculerRatioGarantie(address _identifiant) public view returns (uint256) {
2     Contrepartie storage c = contreparties[_identifiant];
3     return c.exposition == 0 ? 100 : (c.garantie * 100) / c.exposition;
4 }
5
```

C

### ❖ Vérification des limites :

Cette fonction vérifie si l'exposition dépasse la limite, si le ratio de garantie est suffisant et si le risque est trop élevé. Elle déclenche des alertes si nécessaire.

```
1 function verifierLimites(address _identifiant) internal {
2     Contrepartie storage c = contreparties[_identifiant];
3     uint256 ratioGarantie = calculerRatioGarantie(_identifiant);
4     uint256 risque = calculerRisque(_identifiant);
5
6     if (c.exposition > c.limite) {
7         c.estActive = false;
8         emit AlerteLimiteDepassee(_identifiant, c.exposition, c.limite);
9     }
10
11     if (ratioGarantie < 50) {
12         emit AlerteRisqueEleve(_identifiant, "Garantie insuffisante", ratioGarantie);
13     }
14
15     if (risque > 200) {
16         emit AlerteRisqueEleve(_identifiant, "Risque eleve", risque);
17     }
18 }
19
```

## 4. Consultation des Données

Les fonctions **obtenirListeContreparties** et **obtenirHistoriqueTransactions** permettent la consultation de deux types de données importantes dans le contrat :

- ❖ **obtenirListeContreparties:** Cette fonction renvoie un tableau d'adresses Ethereum, représentant les contreparties actives enregistrées dans le système.
- ❖ **obtenirHistoriqueTransactions:** Cette fonction retourne un tableau d'objets EnregistrementTransaction, contenant les détails des transactions effectuées, incluant l'expéditeur, le récepteur, la valeur de la transaction et un horodatage.

Ces fonctions sont publiques et permettent à tout utilisateur d'accéder facilement à ces informations essentielles de manière transparente.



```
1 // Consultation
2 function obtenirListeContreparties() public view returns (address[] memory) {
3     return listeContreparties;
4 }
5
6 function obtenirHistoriqueTransactions() public view returns (EnregistrementTransactio
n[] memory) {
7     return historique;
8 }
```

## 5. Conclusion

Le contrat **GestionnaireRisques** offre une gestion complète et automatisée des risques de contrepartie, garantissant une meilleure sécurité, transparence et efficacité. Grâce à l'utilisation de **smart contracts**, ce système assure l'automatisation des processus tout en optimisant le calcul du risque et en assurant le respect des limites d'exposition et de garantie. Les alertes en cas de risque élevé ou de dépassement de limite permettent de prendre des mesures immédiates, renforçant ainsi la gestion des risques financiers.



## **Chapitre3 : Déploiement, Interface et Sécurité**

## 1. Introduction

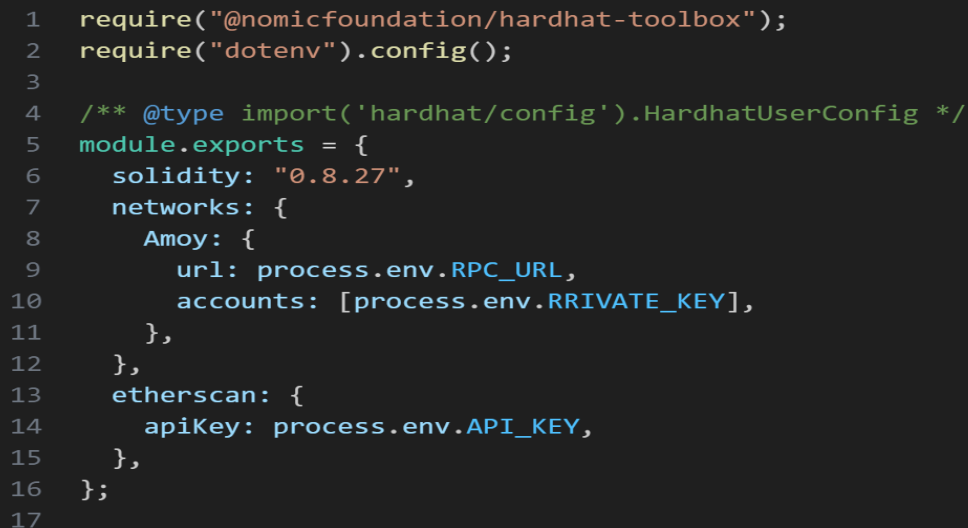
Le **Chapitre 3** se concentre sur le déploiement du smart contract **GestionnaireRisques**, ainsi que sur la mise en place de l'interface utilisateur et des mesures de sécurité. Nous détaillerons la configuration de l'environnement de développement avec **Hardhat**, le déploiement du contrat sur le réseau **Polygon**, et l'intégration de l'interface via **MetaMask**. Enfin, nous aborderons les aspects de sécurité garantissant la protection des transactions et des données sensibles dans ce système de gestion des risques.

## 2. Déploiement et Configuration

Le **déploiement** du contrat **GestionnaireRisques** repose sur plusieurs étapes essentielles pour assurer que le smart contract soit bien configuré, testé et prêt à être utilisé sur la blockchain. Nous avons utilisé **Hardhat** comme framework de développement pour faciliter ce processus. Ce choix est basé sur ses nombreuses fonctionnalités, notamment la possibilité de tester localement le contrat avant de le déployer sur des réseaux comme **Polygon**.

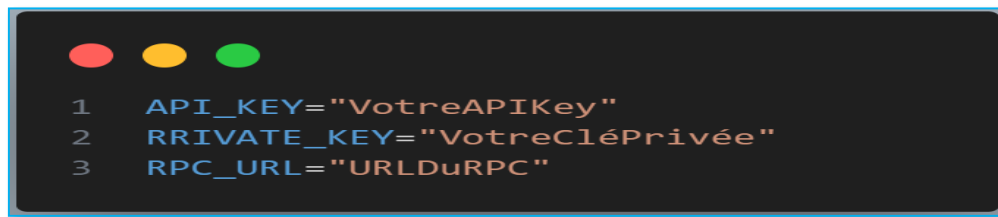
### 2.1. Configuration de l'Environnement

Le processus commence par l'installation de **Node.js** et **npm**, suivie de l'installation de **Hardhat**. Une fois installé, un fichier de configuration **hardhat.config.js** permet de définir les paramètres nécessaires pour connecter Hardhat à des réseaux de test ou de production. Dans ce projet, la configuration de Polygon a été utilisée pour déployer le contrat pour ce projet on a utilisé Amoy.



```
1  require("@nomicfoundation/hardhat-toolbox");
2  require("dotenv").config();
3
4  /** @type import('hardhat/config').HardhatUserConfig */
5  module.exports = {
6    solidity: "0.8.27",
7    networks: {
8      Amoy: {
9        url: process.env.RPC_URL,
10       accounts: [process.env.PRIVATE_KEY],
11     },
12   },
13   etherscan: {
14     apiKey: process.env.API_KEY,
15   },
16 };
17
```

Les variables sensibles telles que **PRIVATE\_KEY**, **API\_KEY**, et **RPC\_URL** sont définies dans un fichier **.env** et peuvent être facilement modifiées sans toucher au code source. Cela permet de configurer le projet de manière flexible et sécurisée, notamment pour le déploiement sur différents réseaux.



### 3. Interface Utilisateur

L'interface utilisateur a été développée en utilisant **HTML**, **CSS**, **JavaScript**, et **Bootstrap** pour garantir une conception réactive et moderne.

#### 3.1. Index

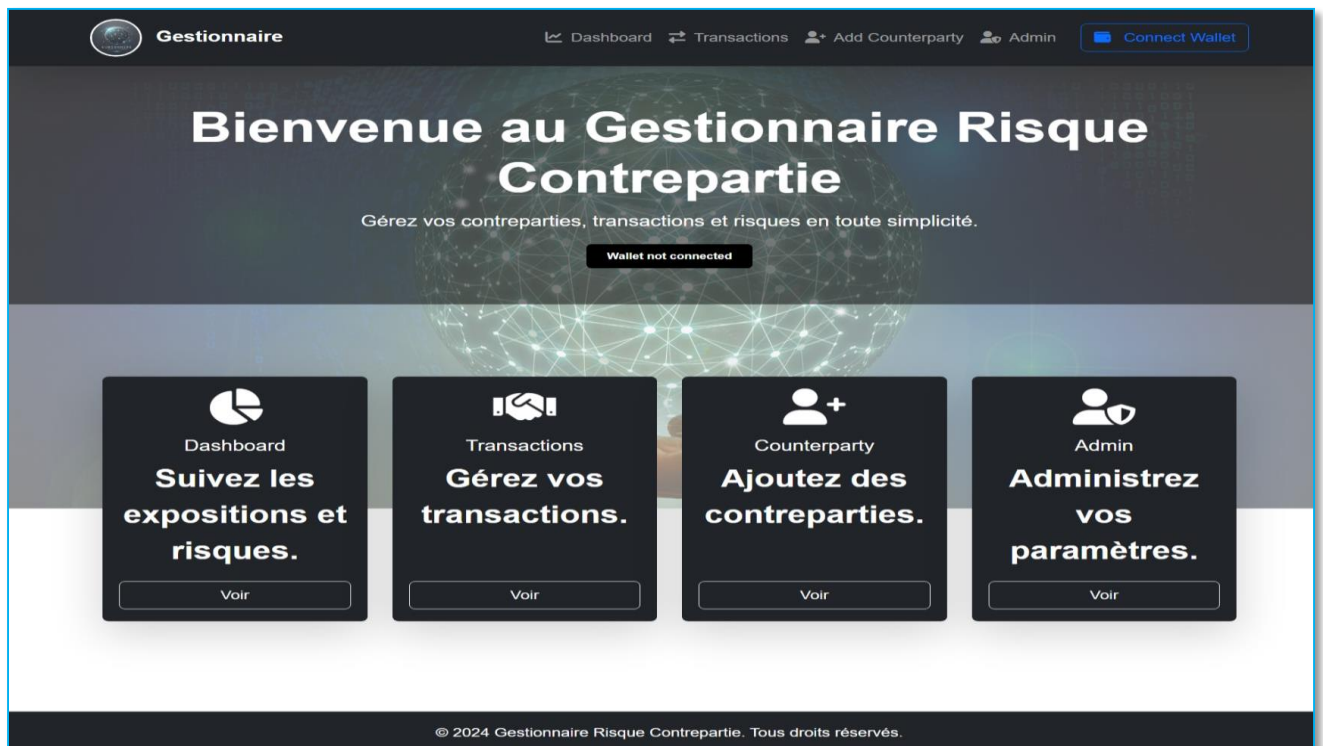
##### ❖ Objectif :

La page **Index** sert de point d'entrée pour l'application. Elle est conçue pour accueillir l'utilisateur et le guider vers les sections principales du système. Cette page contient des liens vers les différentes fonctionnalités du système, comme la gestion des contreparties, l'enregistrement des transactions et l'accès au tableau de bord.

##### ❖ Fonctionnalités :

- **Page d'accueil :**

L'interface est simple et intuitive, avec une navigation claire vers les différentes sections de l'application



- **Connexion au portefeuille :**

Avant d'accéder à toute fonctionnalité, l'utilisateur doit se connecter à **MetaMask** via un bouton dédié. Cela permet de garantir que les transactions et les actions sont effectuées par le bon utilisateur.



- **Navigation :**

Des liens dans le menu permettent de naviguer vers :

- **Ajouter une Contrepartie**
- **Admin**
- **Historique des Transactions**
- **Tableau de Bord**

### 3.1. Create Counterparty

❖ **Objectif :**

Cette page permet à l'administrateur d'ajouter une nouvelle contrepartie au système en renseignant l'adresse Ethereum, le score de crédit et la limite d'exposition. Cette information est ensuite enregistrée sur la blockchain via un appel à la fonction « **ajouterContrepartie** » du contrat **GestionnaireRisques**.

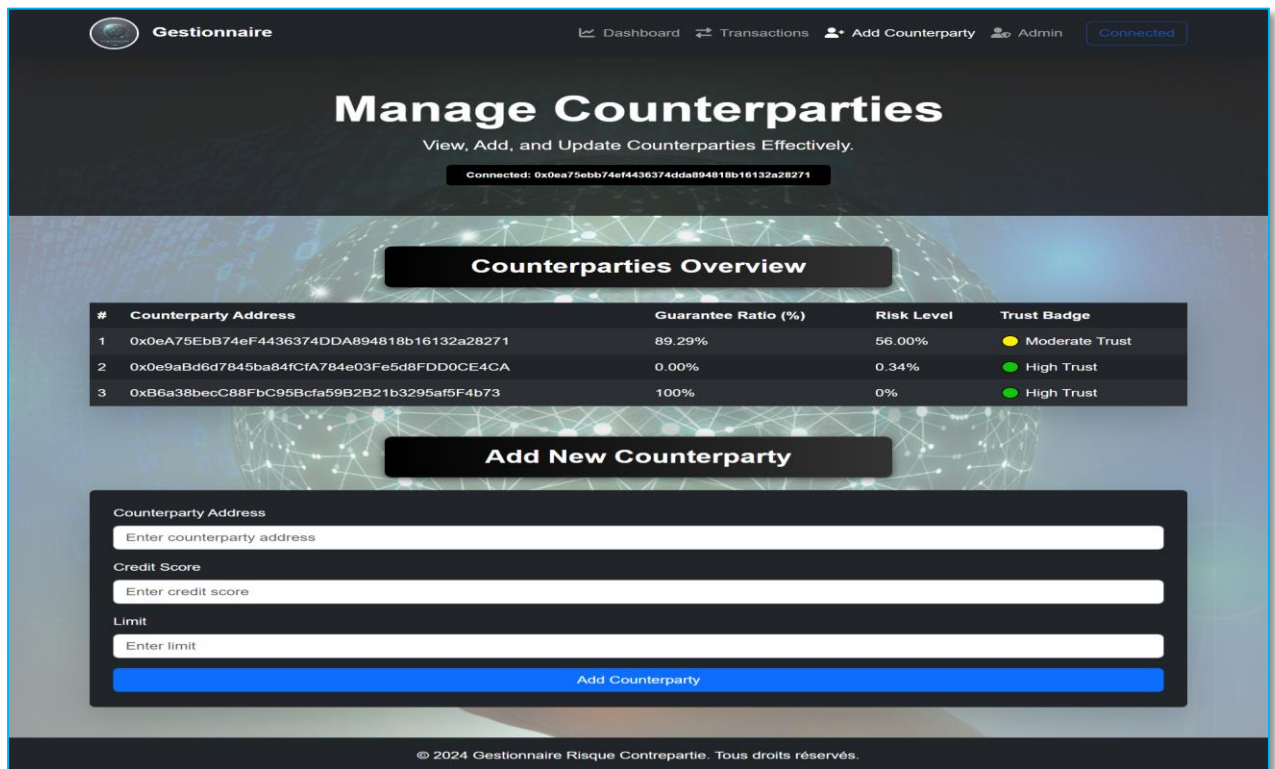
❖ **Fonctionnalités :**

- **Formulaire d'ajout de contrepartie :** Permet à l'administrateur d'entrer l'adresse Ethereum de la contrepartie, son score de crédit et la limite d'exposition pour l'ajouter au système.

- **Affichage des contreparties existantes** : Le tableau présente une liste des contreparties déjà enregistrées avec des informations importantes telles que le **ratio de garantie**, le **niveau de risque**, et le **badge de confiance**.
- **Affichage des contreparties existantes** : Le tableau présente une liste des contreparties déjà enregistrées avec des informations importantes telles que le ratio de garantie, le niveau de risque, et le badge de confiance.
  - Le **badge de confiance** est défini en fonction du **niveau de risque** calculé. Voici comment il est attribué :
    - **High Trust** : Pour les contreparties avec un risque faible.
    - **Moderate Trust** : Pour les contreparties avec un risque modéré (risque supérieur à 50).
    - **Low Trust** : Pour les contreparties avec un risque élevé (risque supérieur à 100).
- **Validation du formulaire** : Avant d'ajouter une contrepartie, le formulaire effectue une validation pour s'assurer que les champs sont correctement remplis.

❖ *Lien avec le Contrat :*

Lors de la soumission du formulaire, la fonction `ajouterContrepatrie` est appelée dans le contrat. Elle prend l'adresse, le score de crédit et la limite d'exposition comme paramètres, puis ajoute la contrepartie à la blockchain.



**Manage Counterparties**  
View, Add, and Update Counterparties Effectively.

Connected: 0x0ea75ebb74ef4436374dda894818b16132a28271

**Counterparties Overview**

#	Counterparty Address	Guarantee Ratio (%)	Risk Level	Trust Badge
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	89.29%	56.00%	🟡 Moderate Trust
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	0.00%	0.34%	🟢 High Trust
3	0xB6a38becC88FbC95Bcfa59B2B21b3295af5F4b73	100%	0%	🟢 High Trust

**Add New Counterparty**

Counterparty Address  
Enter counterparty address

Credit Score  
Enter credit score

Limit  
Enter limit

Add Counterparty

© 2024 Gestionnaire Risque Contrepatrie. Tous droits réservés.

❖ **Test:**

- **Ajouter une Contrepartie**

#	Counterparty Address	Guarantee Ratio (%)	Risk Level	Trust Badge
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	89.29%	56.00%	Moderate Trust
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	0.00%	0.34%	High Trust
3	0xB6a38becC88FbC95Bcfa59B2B21b3295af5F4b73	100%	0%	High Trust
4	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	100%	0%	High Trust

**Add New Counterparty**

Counterparty Address: 0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB

Credit Score: 56

Limit: 1200

**Add Counterparty**

- **Après l'ajout d'une Contrepartie**

**Gestionnaire** | Dashboard | Transactions | Add Counterparty | Admin | Connected

## Manage Counterparties

View, Add, and Update Counterparties Effectively.

Connected: 0x0ea75ebb74ef4436374dda894818b16132a28271

### Counterparties Overview

#	Counterparty Address	Guarantee Ratio (%)	Risk Level	Trust Badge
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	89.29%	56.00%	Moderate Trust
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	0.00%	0.34%	High Trust
3	0xB6a38becC88FbC95Bcfa59B2B21b3295af5F4b73	100%	0%	High Trust
4	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	100%	0%	High Trust

### Add New Counterparty

Counterparty Address: Enter counterparty address

Credit Score: Enter credit score

Limit: Enter limit

**Add Counterparty**

© 2024 Gestionnaire Risque Contrepartie. Tous droits réservés.



### 3.2. Admin

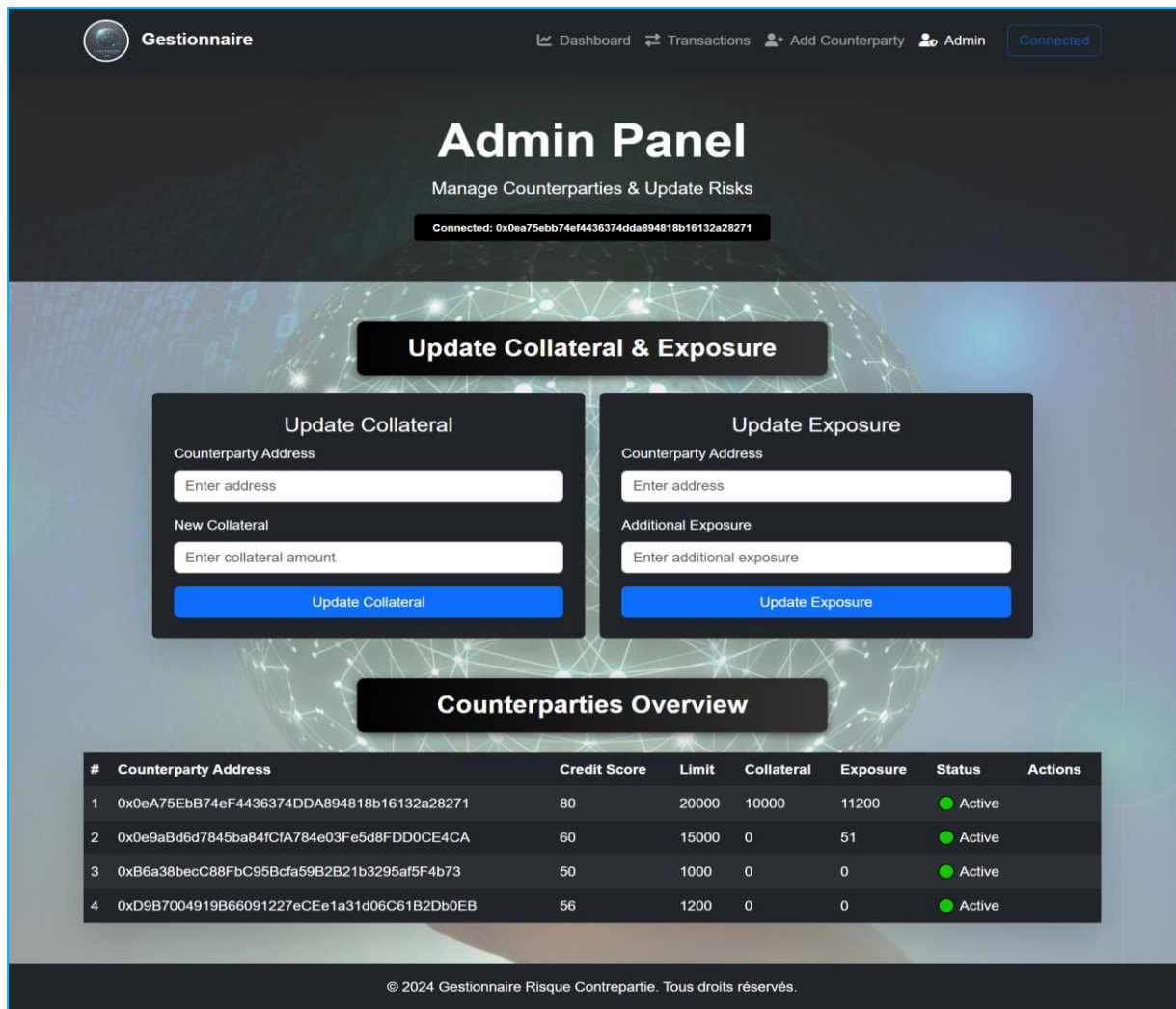
#### ❖ Objectif :

La page **Admin** permet à l'administrateur de gérer les contreparties, de mettre à jour leurs

garanties (collatéraux) et expositions, et de surveiller leur statut. Cette interface est essentielle pour la gestion en temps réel des risques associés aux contreparties et pour effectuer des actions administratives sur les comptes des contreparties. En outre, chaque transaction effectuée entraîne automatiquement la mise à jour de l'exposition des contreparties concernées.

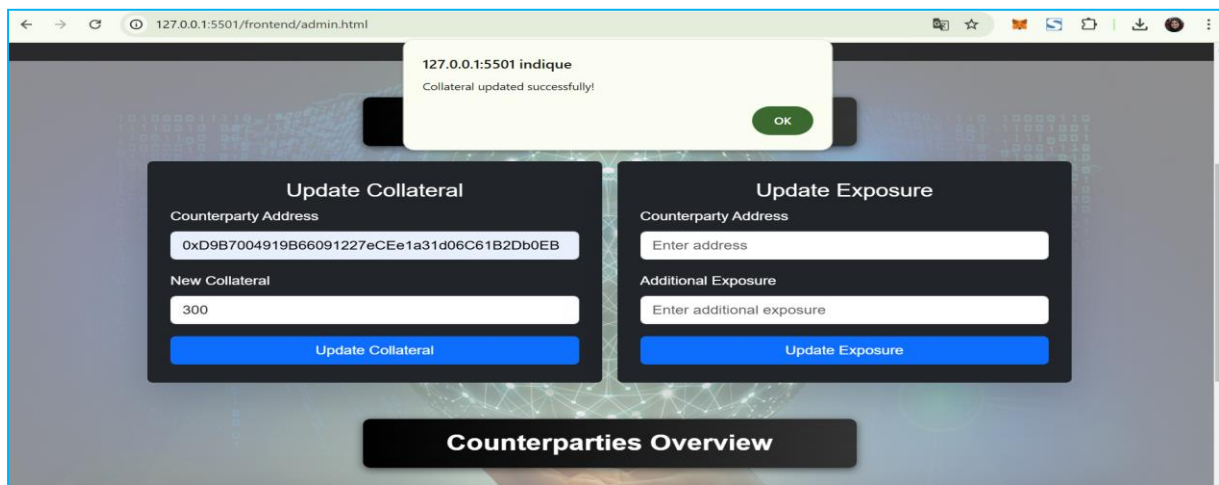
#### ❖ Fonctionnalités :

- **Mise à jour du collatéral (garantie) :**
  - L'administrateur peut saisir l'adresse d'une contrepartie et mettre à jour son collatéral (montant de la garantie). Cette action est cruciale pour ajuster la couverture des contreparties en fonction des risques et des expositions.
  - **Interaction avec le contrat :** La mise à jour du collatéral appelle la fonction `actualiserGarantie` du contrat, mettant à jour la garantie d'une contrepartie spécifique.
- **Mise à jour de l'exposition :**
  - Cette fonctionnalité permet de mettre à jour l'exposition d'une contrepartie en entrant l'adresse de la contrepartie et le montant d'exposition supplémentaire. L'exposition d'une contrepartie représente le montant total auquel elle est exposée dans le système.
  - **Interaction avec le contrat :** La fonction `actualiserExposition` est appelée pour ajuster l'exposition d'une contrepartie en fonction des nouvelles données fournies.
- **Modification de l'exposition lors des transactions :**
  - Lorsqu'une transaction est enregistrée entre deux contreparties, l'exposition de chaque contrepartie impliquée dans la transaction est automatiquement mise à jour. Ce processus garantit que les risques sont recalculés en temps réel et que les expositions sont toujours à jour, suivant les règles définies par le contrat.
  - **Interaction avec le contrat :** La fonction `enregistrerTransaction` du contrat met à jour les expositions des contreparties chaque fois qu'une transaction est effectuée.
- **Vue d'ensemble des contreparties :**
  - Un tableau affiche toutes les contreparties avec des informations essentielles comme l'adresse de la contrepartie, le score de crédit, la limite d'exposition, le collatéral actuel, l'exposition actuelle, le statut de la contrepartie (active ou inactive), et les actions possibles.
  - L'administrateur peut consulter le statut et la santé financière des contreparties.



❖ **Test:**

- Collatéral :
  - Mise à jour du collateral:





- Après Mise à jour du collatéral:

Counterparties Overview							
#	Counterparty Address	Credit Score	Limit	Collateral	Exposure	Status	Actions
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	80	20000	10000	11200	Active	
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	60	15000	0	51	Active	
3	0xB6a38becC88FbC95Bcfa59B2B21b3295af5F4b73	50	1000	0	0	Active	
4	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	56	1200	300	0	Active	

Collatéral mis à jour

- Exposition :
  - Mise à jour de l'exposition

127.0.0.1:5501 indique  
Exposure updated successfully!

OK

### Update Collateral

Counterparty Address

New Collateral

Update Collateral

### Update Exposure

Counterparty Address

Additional Exposure

Update Exposure

- Après Mise à jour de l'exposition

Counterparties Overview							
#	Counterparty Address	Credit Score	Limit	Collateral	Exposure	Status	Actions
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	80	20000	10000	11200	Active	
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	60	15000	0	51	Active	
3	0xB6a38becC88FbC95Bcfa59B2B21b3295af5F4b73	50	1000	0	100	Active	
4	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	56	1200	300	0	Active	

Exposition mis à jour

### 3.3. Transactions

#### ❖ Objectif :

La page **Transactions** permet aux utilisateurs d'enregistrer et de consulter les transactions entre contreparties. Chaque fois qu'une transaction est effectuée, l'exposition des contreparties impliquées est mise à jour automatiquement sur la page **Admin Panel**. Cette page joue un rôle clé dans le suivi des activités des contreparties et assure une gestion dynamique des expositions, car chaque modification de l'exposition due à une transaction est directement liée aux données affichées dans l'interface d'administration.

#### ❖ Fonctionnalités :

- **Historique des transactions :**

- Un tableau présente l'historique des transactions entre contreparties, avec des détails sur l'expéditeur, le destinataire, le montant de la transaction et l'horodatage.
- Chaque transaction effectuée est enregistrée et affichée dans l'historique pour permettre aux utilisateurs de suivre les échanges passés.

- **Enregistrement d'une nouvelle transaction :**

- Un formulaire permet à l'utilisateur de saisir l'adresse du destinataire et le montant de la transaction. Cette transaction, une fois validée, est envoyée au contrat via la fonction `enregistrerTransaction`.
- Lorsque la transaction est effectuée, l'exposition de chaque contrepartie (expéditeur et destinataire) est mise à jour automatiquement, et les nouvelles données sont immédiatement visibles sur la page **Admin**.

#### ❖ Lien avec le Contrat :

Chaque fois qu'une transaction est ajoutée via le formulaire, la fonction `enregistrerTransaction` du contrat est appelée. Cela met à jour les expositions des deux contreparties concernées par la transaction. Le contrat enregistre ensuite la transaction dans l'historique et met à jour les expositions des contreparties en temps réel.

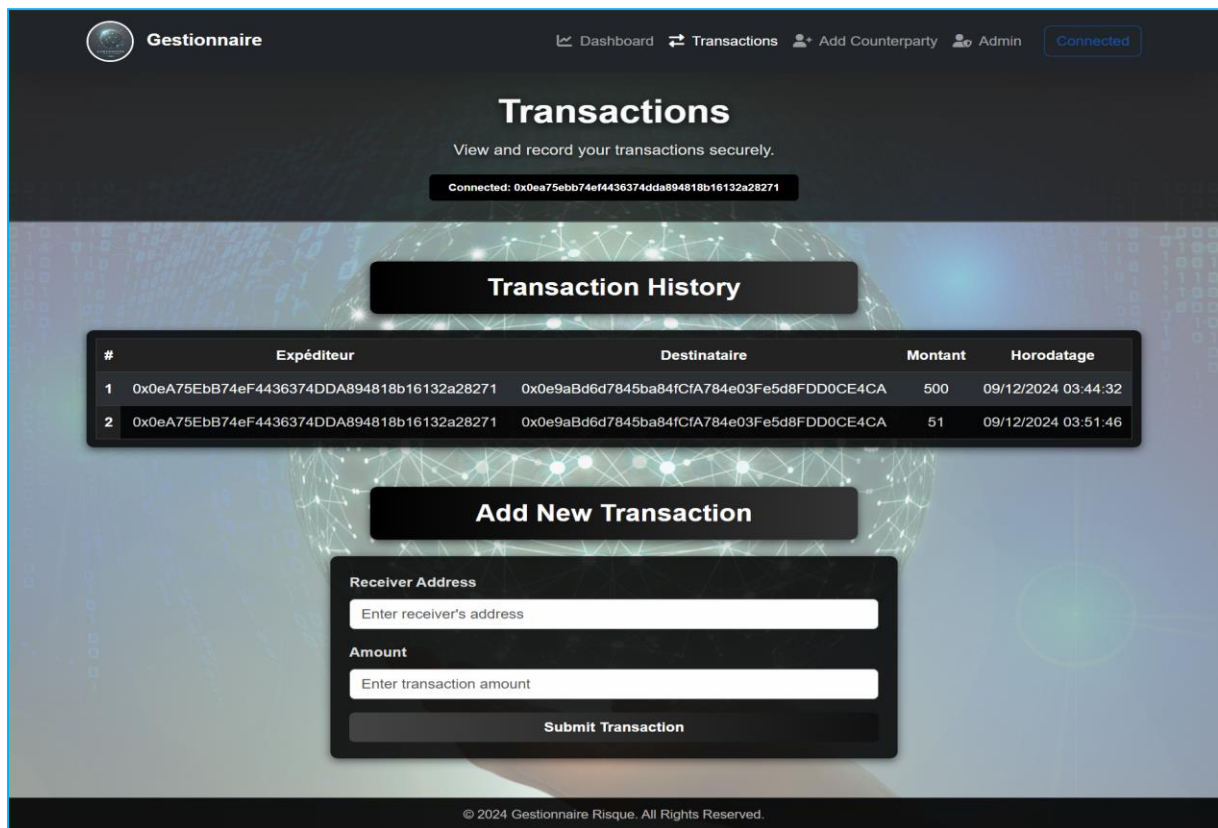
Le calcul de l'exposition est effectué à chaque transaction, et la mise à jour de ces informations est visible dans la page **Admin Panel** sous les colonnes **Exposition** et **Garantie**. Chaque fois que l'exposition change, les informations sont recalculées et mises à jour.

#### ❖ Impact sur l'Admin :

Lorsque l'utilisateur enregistre une nouvelle transaction sur la page **Transactions**, l'exposition des contreparties impliquées est immédiatement mise à jour sur la page **Admin Panel**. Cela se produit automatiquement, grâce à l'appel à la fonction `enregistrerTransaction` dans le contrat, qui ajuste les expositions et met à jour les données relatives à chaque contrepartie.

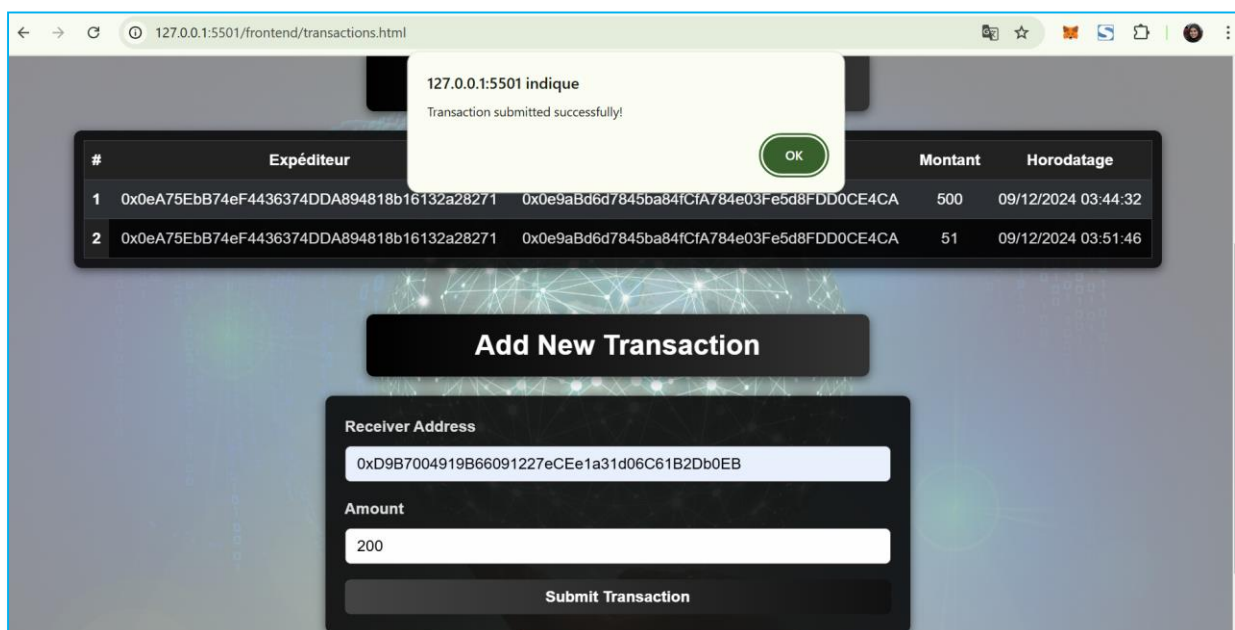
L'exposition de chaque contrepartie est calculée à chaque transaction et la page **Admin Panel** affiche ces nouvelles valeurs en temps réel. Cela permet à l'administrateur de surveiller de près

l'impact de chaque transaction sur le système et de s'assurer que les expositions et les garanties sont bien gérées.



❖ *Test:*

- **Effectuer transaction**



- Après la transaction

**Gestionnaire** Dashboard Transactions Add Counterparty Admin Connected

## Transactions

View and record your transactions securely.

Connected: 0x0ea75ebb74ef4436374dda894818b16132a28271

Exposition mis à jour

### Transaction History

#	Expéditeur	Destinataire	Montant	Horodage
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	500	09/12/2024 03:44:32
2	0x0eA75EbB74eF4436374DDA894818b16132a28271	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	51	09/12/2024 03:51:46
3	0x0eA75EbB74eF4436374DDA894818b16132a28271	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	200	09/12/2024 04:01:00

### Add New Transaction

Receiver Address

Amount

Submit Transaction

© 2024 Gestionnaire Risque. All Rights Reserved.

- Après la transaction : Exposition

## Counterparties Overview

#	Counterparty Address	Credit Score	Limit	Collateral	Exposure	Status	Actions
1	0x0eA75EbB74eF4436374DDA894818b16132a28271	80	20000	10000	11200	Active	
2	0x0e9aBd6d7845ba84fCfA784e03Fe5d8FDD0CE4CA	80	20000	0	551	Active	
3	0xB6a38becC88FbC95BcfA59B2B21b3295af5F4b73	50	1000	0	900	Active	
4	0xD9B7004919B66091227eCEe1a31d06C61B2Db0EB	56	1200	300	200	Active	

Exposition mis à jour

© 2024 Gestionnaire Risque Contrepartie. Tous droits réservés.

### 3.4. Dashboard

#### ❖ Objectif :

La page **Dashboard** permet de suivre les principales métriques du système, offrant une vue d'ensemble des contreparties, des transactions et de l'exposition totale. Cette interface est cruciale pour surveiller l'état général du système et analyser les risques financiers de manière efficace. Elle affiche également des graphiques et des statistiques qui aident à visualiser l'impact des expositions et des transactions.

#### ❖ Fonctionnalités :

- **Vue d'ensemble des métriques :**
  - Affiche les **total des contreparties**, **total des transactions** et **exposition totale** dans le système.
  - Un graphique présente la répartition des **contreparties**, **transactions**, et **exposition** sur une échelle logarithmique, offrant ainsi une vue claire de la taille et de l'impact des différentes catégories.
- **Vue d'ensemble des contreparties :**
  - Un tableau répertorie les contreparties enregistrées avec des informations telles que leur adresse Ethereum, leur ratio de garantie, le niveau de risque, et leur badge de confiance.
  - Le **badge de confiance** est déterminé en fonction du **niveau de risque** de chaque contrepartie :
    - **High Trust** : Pour un faible risque.
    - **Moderate Trust** : Pour un risque modéré.
    - **Low Trust** : Pour un risque élevé.
  - Le tableau permet également de visualiser le statut de chaque contrepartie (active ou inactive).

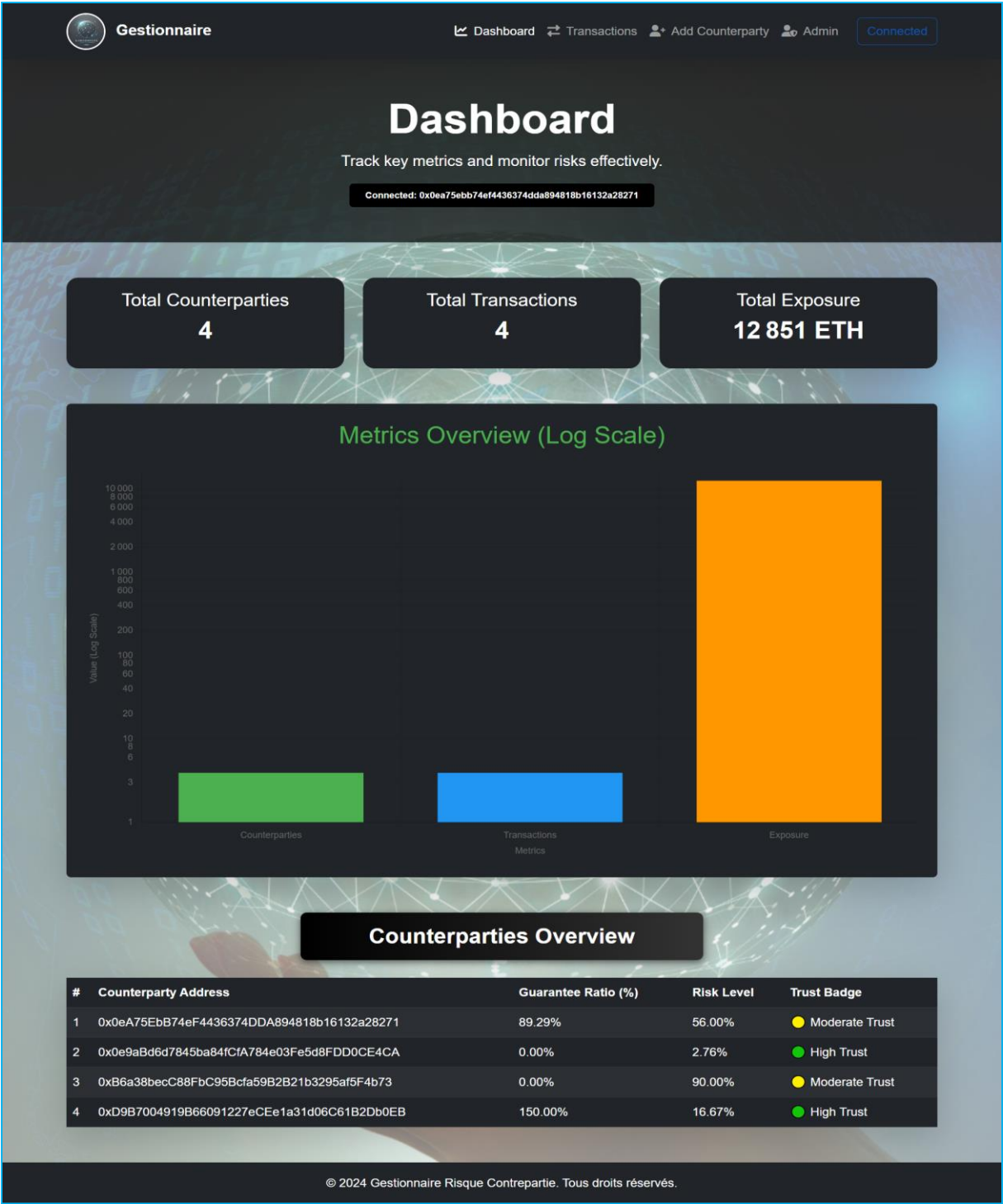
#### ❖ Lien avec le Contrat :

Les informations affichées sur le tableau de bord proviennent directement des données enregistrées dans le contrat **GestionnaireRisques** :

- **Total Counterparties** : Les informations relatives aux contreparties sont récupérées via la fonction **obtenirListeContreparties**.
- **Total Transactions** : Le nombre de transactions est calculé en fonction des données enregistrées dans l'historique du contrat.
- **Total Exposure** : L'exposition totale est mise à jour en temps réel grâce aux transactions effectuées et aux mises à jour d'exposition.

Le **badge de confiance** est attribué en fonction du calcul du risque, qui prend en compte le score de crédit et l'exposition de chaque contrepartie.





4. Conclusion

Le chapitre 3 a détaillé les interfaces **Dashboard**, **Admin**, **Transactions** et **Create Counterparty**, permettant une gestion complète des contreparties et des expositions en temps réel. Ces pages garantissent une mise à jour dynamique et transparente des risques via le contrat **GestionnaireRisques**.

## **Chapitre 4 : Questions d'Évaluation**

## 1. Compréhension Technique

### 1. Expliquez comment votre smart contract gère les limites d'exposition. Quelles mesures de sécurité empêchent les dépassements de limites ?

Le smart contract gère les limites d'exposition en associant à chaque contrepartie un montant d'exposition défini par sa limite d'exposition. Lorsqu'une contrepartie dépasse cette limite, le contrat la désactive automatiquement et déclenche une alerte de dépassement de limite. Des modificateurs sont utilisés pour vérifier l'état actif des contreparties avant d'effectuer des transactions ou des mises à jour. Les vérifications de limites sont effectuées à chaque fois qu'une exposition est mise à jour, garantissant ainsi que les règles de sécurité sont toujours respectées.

### 2. Décrivez votre implémentation des calculs de risque. Comment votre modèle prend-il en compte :

- ❖ **Les scores de crédit :** Le calcul du risque intègre le score de crédit d'une contrepartie en le divisant par la limite d'exposition, afin de déterminer le niveau de risque associé à une exposition donnée.
- ❖ **L'exposition courante :** Le risque est calculé en prenant en compte l'exposition actuelle d'une contrepartie par rapport à sa limite et à son score de crédit. Plus l'exposition est élevée par rapport à la limite, plus le risque sera élevé.
- ❖ **L'historique des transactions :** Bien que l'historique des transactions ne soit pas utilisé directement pour le calcul du risque dans cette implémentation, chaque nouvelle transaction modifie l'exposition et, par conséquent, l'indice de risque calculé pour chaque contrepartie.

### 3. Analysez l'efficacité en termes de gas de votre smart contract. Quelles optimisations avez-vous implémentées ?

L'efficacité en termes de gas a été optimisée par :

L'utilisation de mappings au lieu de tableaux pour les contreparties et les expositions, réduisant ainsi la consommation de gas liée à l'accès et à la mise à jour des informations.

La réduction des appels de fonctions internes coûteux et des événements, qui sont émis uniquement lorsqu'un changement significatif se produit, afin de ne pas consommer de gas inutilement.

### 4. Comment votre système gère-t-il les cas limites tels que ?

- ❖ **La congestion du réseau :** La congestion du réseau n'est pas directement gérée par le contrat. Cependant, des mécanismes de retry peuvent être utilisés dans le front-end pour garantir que les transactions sont envoyées plusieurs fois si nécessaire.
- ❖ **Les transactions échouées :** Les transactions échouées sont interceptées et un message d'erreur est renvoyé à l'utilisateur. Le système garantit également que les vérifications de conditions sont faites avant d'exécuter des actions susceptibles d'échouer.
- ❖ **Les entrées invalides :** Le contrat vérifie systématiquement les entrées avant de procéder à toute mise à jour (par exemple, validation de l'adresse, des montants et des limites), assurant ainsi la sécurité des transactions.

### 5. Expliquez votre stratégie de test et les résultats. Quels scénarios avez-vous testés ?



La stratégie de test repose sur l'utilisation de Hardhat pour simuler un environnement de test local avant de déployer le contrat sur un réseau de production. Les tests ont couvert :

- ❖ L'ajout et la mise à jour des contreparties.
- ❖ La gestion des limites d'exposition, des garanties et des risques.
- ❖ Le bon fonctionnement des événements lors de dépassements de limite et des mises à jour.
- ❖ Les transactions entre contreparties et l'impact sur les expositions.

## 2. Compréhension de la Gestion des Risques

### 1. Comparez la gestion traditionnelle des risques de contrepartie avec votre implémentation blockchain. Quels sont les avantages et les limitations de chaque approche ?

- ❖ **Gestion traditionnelle des risques** : Souvent centralisée, manuelle, et sujette à des erreurs humaines. Elle peut manquer de transparence et être lente dans l'exécution des contrôles de risques.
- ❖ **Implémentation blockchain** : Offrant une transparence et une sécurisation accrues, elle permet une gestion automatique via des smart contracts. Les transactions sont enregistrées de manière immuable, assurant une traçabilité complète. Toutefois, elle peut entraîner des frais de transaction sur certaines blockchains (comme Ethereum) et être limitée par la vitesse des blocs.

### 2. Comment votre système gère-t-il les scénarios de risque suivants :

- ❖ **Augmentation soudaine de l'exposition** : L'exposition est mise à jour en temps réel lors des transactions, et des alertes sont générées si l'exposition dépasse la limite définie.
- ❖ **Détérioration du score de crédit** : Le score de crédit est pris en compte dans le calcul du risque, et des alertes sont émises si le ratio d'exposition devient trop élevé par rapport à la limite.
- ❖ **Transactions multiples simultanées** : Les transactions sont gérées de manière séquentielle sur la blockchain, garantissant que les expositions sont toujours à jour, même en cas de transactions simultanées.

### 3. Proposez des améliorations potentielles à votre modèle de calcul des risques. Quels facteurs supplémentaires pourraient être incorporés ?

Des améliorations possibles incluent :

- ❖ L'intégration de l'historique des transactions pour mieux estimer la volatilité d'une contrepartie sur une période donnée.
- ❖ L'utilisation de modèles de risque externes, tels que des évaluations de marché, pour affiner le calcul du risque.

- ❖ L'inclusion de facteurs économiques comme les taux d'intérêt ou les indices de marché qui peuvent affecter les expositions et les scores de crédit.

### 3. Implémentation et Innovation

#### **1. Documentez les fonctionnalités supplémentaires que vous avez implémentées au-delà des exigences de base. Expliquez leur valeur métier.**

Alertes automatiques en cas de dépassement de limite d'exposition ou de faible garantie, permettant à l'administrateur de prendre des décisions rapides et éclairées.

Suivi en temps réel des expositions qui donne aux utilisateurs une visibilité claire sur les risques encourus, avec la possibilité de modifier rapidement les expositions si nécessaire.

#### **2. Analysez les applications potentielles de votre système dans le monde réel. Quelles modifications seraient nécessaires pour une utilisation en production ?**

Le système pourrait être utilisé dans des institutions financières pour gérer les risques de contrepartie dans des transactions de crédit, d'échange ou de dérivés.

Pour un usage en production, il serait nécessaire d'intégrer une solution de scalabilité, notamment pour gérer des volumes de transactions plus élevés, et de garantir une gestion des coûts en gas plus optimale.

#### **3. Réfléchissez sur le processus de développement :**

- ❖ Quels défis avez-vous rencontrés ?

La principale difficulté résidait dans la gestion de l'exécution des transactions en temps réel et l'optimisation des frais de gas.

- ❖ Comment les avez-vous surmontés ?

En utilisant Hardhat pour simuler des tests et optimiser le contrat pour réduire la consommation de gas.

- ❖ Que feriez-vous différemment ?

J'améliorerais l'intégration des outils d'analyse des risques en ajoutant davantage de paramètres pour affiner les calculs et mieux gérer les expositions dynamiques.

## **Conclusion Générale**

Ce projet a permis de développer un système de gestion des risques de contrepartie innovant, basé sur la technologie blockchain et l'utilisation de smart contracts. L'approche a été conçue pour automatiser la gestion des expositions et des risques, tout en garantissant une plus grande transparence et une réduction des erreurs humaines. Grâce à l'intégration des mécanismes de vérification des limites d'exposition et des alertes automatiques, le système assure une gestion sécurisée des transactions financières. L'utilisation de Hardhat a facilité le développement et les tests, tandis que l'interface utilisateur a permis d'offrir une expérience fluide pour l'administrateur. Cependant, pour une adoption en production à grande échelle, il serait nécessaire de mettre en place des solutions pour améliorer la scalabilité et réduire les coûts de transaction (gas). De plus, l'ajout de nouveaux paramètres d'évaluation de risque, comme des indicateurs économiques, pourrait renforcer l'efficacité du modèle. En somme, ce système présente un potentiel énorme pour la gestion des risques dans les transactions financières, mais il nécessite encore des améliorations pour mieux répondre aux exigences des grandes entreprises et des marchés financiers.

