Implémentation du Chat en Temps Réel et Sécurisation des Rooms

1. Introduction aux Technologies Utilisées

Dans le cadre de notre plateforme sécurisée de collaboration scientifique, nous avons mis en place un système de chat en temps réel permettant aux utilisateurs de communiquer efficacement. Pour ce faire, nous utilisons plusieurs technologies adaptées aux exigences de performance et de sécurité :

- Django Channels : Extension de Django pour gérer les WebSockets.

- Redis : Base de données en mémoire utilisée comme broker pour Django Channels.

- Daphne : Serveur ASGI permettant de gérer les connexions WebSocket.

- WebSockets : Protocole permettant la communication en temps réel entre le serveur et les clients.

- HTMX/React : Technologies utilisées pour l'interface utilisateur du chat.

2. Le Fonctionnement des WebSockets

Contrairement à HTTP qui fonctionne sur un modèle requête-réponse, WebSockets établit une connexion bidirectionnelle persistante entre le client et le serveur. Cela permet :

- Une communication en temps réel sans rafraîchissement de la page.

- Une réduction de la latence dans l'envoi et la réception des messages.

- Une utilisation efficace des ressources réseau.

Dans notre projet, Django Channels interagit avec Daphne et Redis pour gérer les connexions WebSocket et distribuer les messages aux utilisateurs connectés.

3. Modélisation des Rooms et Messages

Dans notre implémentation, une Room représente un espace de discussion entre plusieurs utilisateurs. Nous avons défini les modèles suivants dans Django :

- Room : Contient le nom de la room et les utilisateurs participants.

- Message : Contient le texte du message, l'expéditeur et la room associée.

Exemple de modélisation Django :

```
class Room(models.Model):
    name = models.CharField(max_length=255, unique=True)
    users = models.ManyToManyField(User, related_name="rooms")
```

```
class Message(models.Model):
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    sender = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
```

4. Création et Gestion des Rooms

Un utilisateur peut créer une room et y inviter d'autres participants. Les étapes de création et de gestion des rooms sont les suivantes :

- Création d'une room avec un nom unique.

- Attribution de permissions (publique ou privée).

- Gestion des invitations et ajout de membres.

- Stockage des messages et récupération des historiques de discussion.

L'intégration avec Django Channels permet d'envoyer les messages en temps réel et de les stocker dans la base de données.

5. Chiffrement des Messages et Sécurité des Rooms

Pour garantir la confidentialité des échanges, nous envisageons d'implémenter un chiffrement de bout en bout. Notre approche inclura :

- Utilisation d'un algorithme de chiffrement adapté aux systèmes embarqués.

- Échange de clés sécurisées entre les participants d'une room.

- Vérification de l'intégrité des messages échangés.

De plus, nous mettons en place des contrôles d'accès stricts pour éviter toute intrusion non autorisée.

6. Améliorations Futures et Mesures Complémentaires

Nous prévoyons d'améliorer la sécurité et les fonctionnalités du chat en ajoutant :

- Un système de détection d'activités suspectes et de prévention des attaques.

- Un mécanisme de suppression automatique des messages après un certain temps.

- Une meilleure gestion des permissions pour les administrateurs de rooms.

- Un renforcement du chiffrement et de la gestion des clés.

Ces améliorations permettront d’assurer une communication sécurisée et conforme aux exigences des utilisateurs.