Création d'une API de chat avec FastAPI

Pour créer une API FastAPI utilisant le protocole de communication WebSocket pour un chat en Python, suivez ces étapes :

Configuration de l'environnement : Assurez-vous d'avoir `fastapi` et `uvicorn` installés. Si ce n'est pas le cas, installez-les avec :

```
pip install fastapi uvicorn
```

Création de l'API avec FastAPI :

- Créez un fichier Python, par exemple `main.py`.
- Développez un serveur de chat utilisant FastAPI et WebSocket.

Implémentation du serveur de chat :

Voici le code pour créer un serveur de chat avec FastAPI utilisant les WebSockets (il est à noter qu'il s'agit d'une ébauche destinée à rapidement évoluer) :

```python
# main.py

from fastapi import FastAPI, WebSocket, WebSocketDisconnect
from typing import List

app = FastAPI()

class ConnectionManager:
    def __init__(self):
        self.active_connections: List[WebSocket] = []

    async def connect(self, websocket: WebSocket):
        await websocket.accept()
        self.active_connections.append(websocket)

    def disconnect(self, websocket: WebSocket):
        self.active_connections.remove(websocket)

    async def send_message(self, message: str, websocket: WebSocket):
        await websocket.send_text(message)

    async def broadcast(self, message: str):
        for connection in self.active_connections:
            await connection.send_text(message)

manager = ConnectionManager()
```

PROF

```python
@app.websocket("/ws/chat")
async def websocket_endpoint(websocket: WebSocket):
    await manager.connect(websocket)
    try:
        while True:
            data = await websocket.receive_text()
            await manager.broadcast(f"Message reçu : {data}")
    except WebSocketDisconnect:
        manager.disconnect(websocket)
        await manager.broadcast("Un utilisateur s'est déconnecté.")
```

Explication du code :

- **ConnectionManager** : Gère les connexions WebSocket actives. Il permet de connecter, déconnecter et envoyer des messages aux clients connectés.
- **websocket_endpoint** : Gère la connexion WebSocket et diffuse les messages reçus à tous les utilisateurs connectés.

Exécution du serveur :

Pour exécuter le serveur, utilisez la commande suivante :

```
uvicorn main:app --reload
```

Test du WebSocket :

- Utilisez un client WebSocket (comme l'extension Chrome "Simple WebSocket Client" ou un script Python utilisant `websockets`) pour tester le chat.

Suggestions :

PROF

- Ajouter des tests unitaires pour vérifier le comportement du serveur.
- Ajouter une fonctionnalité pour gérer les salles de chat (channels) dans le serveur WebSocket.