ENSEIGNEMENT SUPÉRIEUR ET UNIVERSITAIRE

Université Nouveaux Horizons

Faculté des sciences informatiques

Système de chat

Programmation orientée Objet
UML

Par

CANSA KAYEMBE AMAURY
KINYANTA NKONKOSHA DANIEL
KAYOMBO KAKANGA RUSADE
MWENDA MUKUNTU MIKE
TSHABU NGANDU BERNARD

Licence 2 Informatique Réseaux et Infrastructure

Année académique 2022-2023

Introduction

L'ère de la communication en ligne est en constante évolution et les applications de chat en groupe sont devenues un outil essentiel pour connecter les utilisateurs du monde entier. Dans ce contexte, nous sommes ravis de présenter notre application de chat en groupe utilisant la technologie WebSocket, développée en utilisant le langage de programmation Java.

Notre application de chat en groupe offre une plateforme sécurisée et interactive permettant aux utilisateurs de créer des salons de discussion, d'inviter leurs amis et de partager des messages en temps réel. Grâce à la technologie WebSocket, cette application permet une communication bidirectionnelle en temps réel entre le serveur et les clients, offrant ainsi une expérience utilisateur fluide et instantanée.

Fonctionnalités principales :

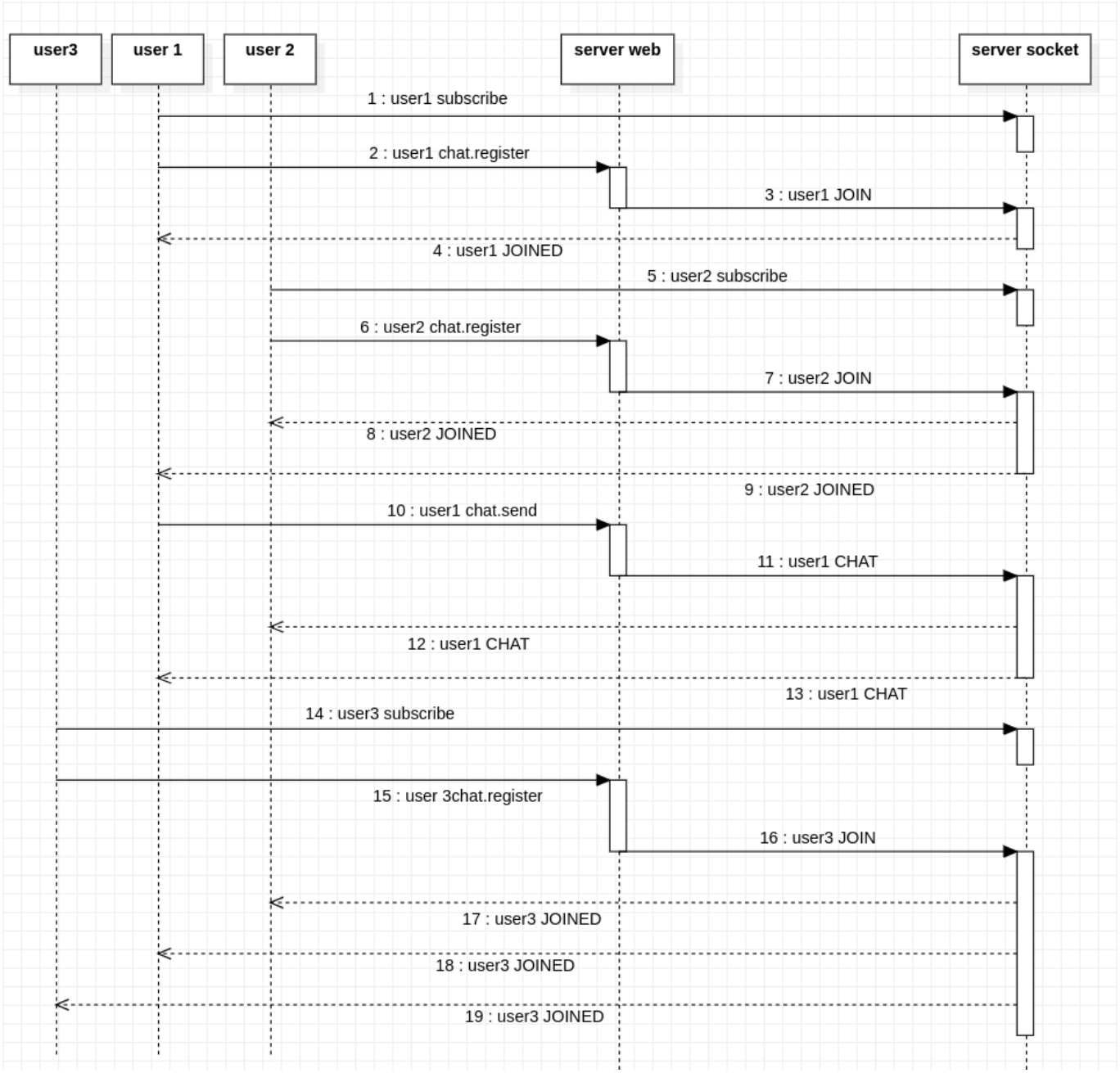
- **Connexion au réseau local** : L'application utilise une technologie de communication WebSocket pour faciliter la connexion des utilisateurs au réseau local. Il leur suffit de se connecter à leur réseau Wi-Fi et d'entrer les informations de connexion fournies.

- **Nom d'utilisateur personnalisé** : Chaque utilisateur peut spécifier un nom d'utilisateur unique, ce qui lui permet d'être identifiable par les autres membres du groupe. Cela facilite la communication et crée un environnement convivial.

- **Discussions en groupe** : Une fois connecté, l'utilisateur peut accéder à une liste des différents groupes de discussion disponibles dans le réseau local. Il peut choisir de rejoindre un groupe spécifique ou créer son propre groupe s'il le souhaite. Les messages envoyés par un utilisateur sont visibles par tous les autres membres du groupe, permettant ainsi les conversations en temps réel.

- **Fonctionnalités de base de discussion** : L'application de chat en groupe offre des fonctionnalités de base telles que l'envoi de messages texte, l'envoi de fichiers, l'ajout de réactions aux messages et la suppression de messages inappropriés. Ces fonctionnalités facilitent la collaboration et favorisent une expérience de discussion fluide.

Diagramme des séquences

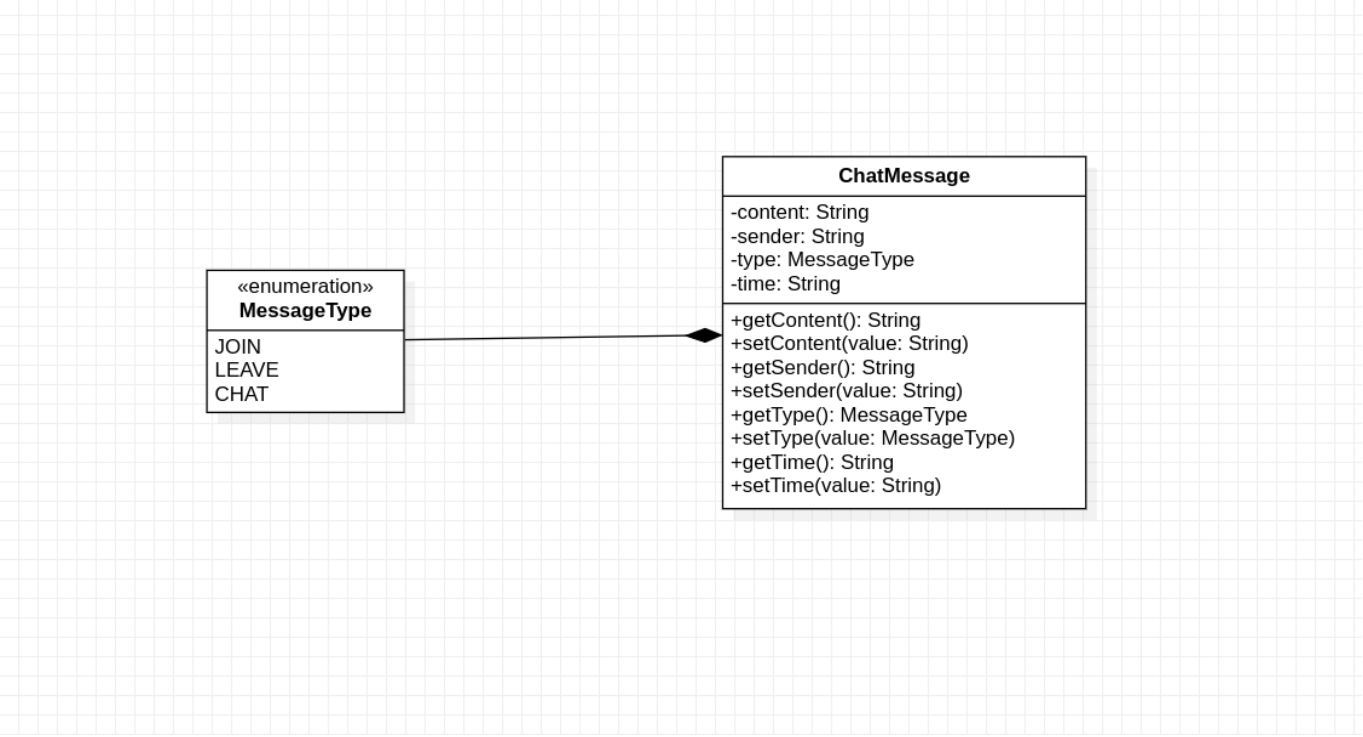


Dans ce diagramme de séquence, nous pouvons observer les interactions entre les utilisateurs et le serveur dans le cadre d'une plateforme de chat en temps réel.

- L'utilisateur se connecte à la plateforme en envoyant une requête au websocket pour s'abonner. Cela permet à l'utilisateur de recevoir les messages du chat.
- L'utilisateur envoie un message pour s'enregistrer en utilisant la fonctionnalité chat.register. Le serveur reçoit ce message et enregistre l'utilisateur avec son nom d'utilisateur.
- Le serveur envoie un message au websocket pour notifier la connexion d'un nouvel utilisateur. Cette notification est envoyée à tous les utilisateurs connectés.
- Le websocket envoie un message pour informer tous les utilisateurs connectés qu'un nouvel utilisateur s'est connecté. Cela permet aux autres utilisateurs de prendre connaissance de la présence de ce nouvel utilisateur.
- Lorsqu'un utilisateur envoie un message via la fonctionnalité chat.send, le serveur reçoit ce message et le renvoie à tous les utilisateurs connectés via le topic/public. Cela permet à tous les utilisateurs de recevoir le message en temps réel.

En résumé, ce diagramme de séquence illustre le flux d'interaction entre les utilisateurs et le serveur dans un système de chat en temps réel. Il montre comment les utilisateurs s'abonnent, s'enregistrent, envoient des messages et reçoivent des messages via le serveur et le websocket

Diagramme des classes



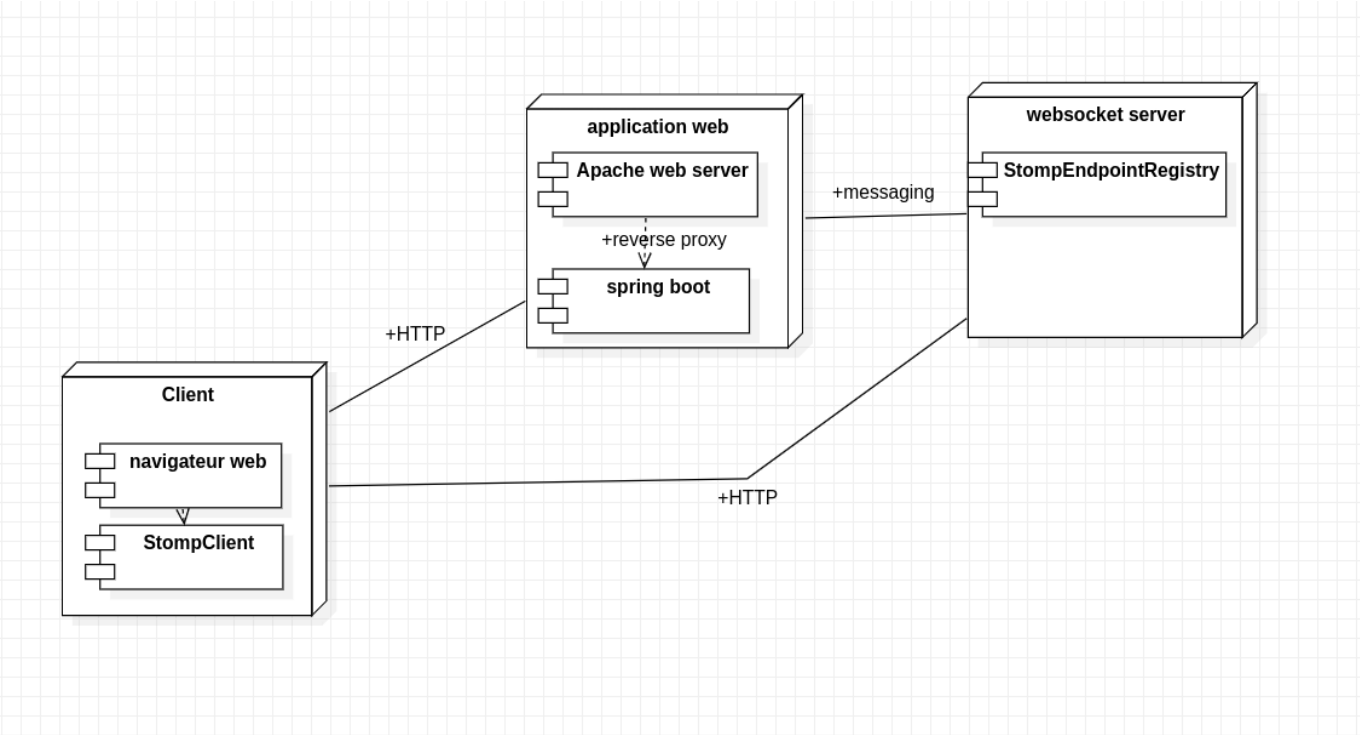
Nous avons d'une part une classe **ChatMessage** et **MessageType** de l'autre. La classe ChatMessage comporte les attributs qui suivent :

- Le content, qui va contenir le message
- Le sender, qui défini le nom d'utilisateur de celui qui envoie le message
- Le type, qui défini le genre de message en présence
- time, qui nous donne l'heure en temps réel

La classe **MessageType** comporte elle une énumération composé de JOIN, LEAVE et CHAT qui constitue le type de message dont il est question dans l'attribut type de la classe **ChatMessage**

Entre la classe **ChatMessage** et la classe **MessageType** nous avons une relation de « composition » parce qu'au regard de la dépendance entre les deux classes, nous constatons que la classe ChatMessage a besoin des services fournis par la classe MessageType pour accomplir sa fonction autrement dit ceci permet aux deux classes d'interagir ensemble.

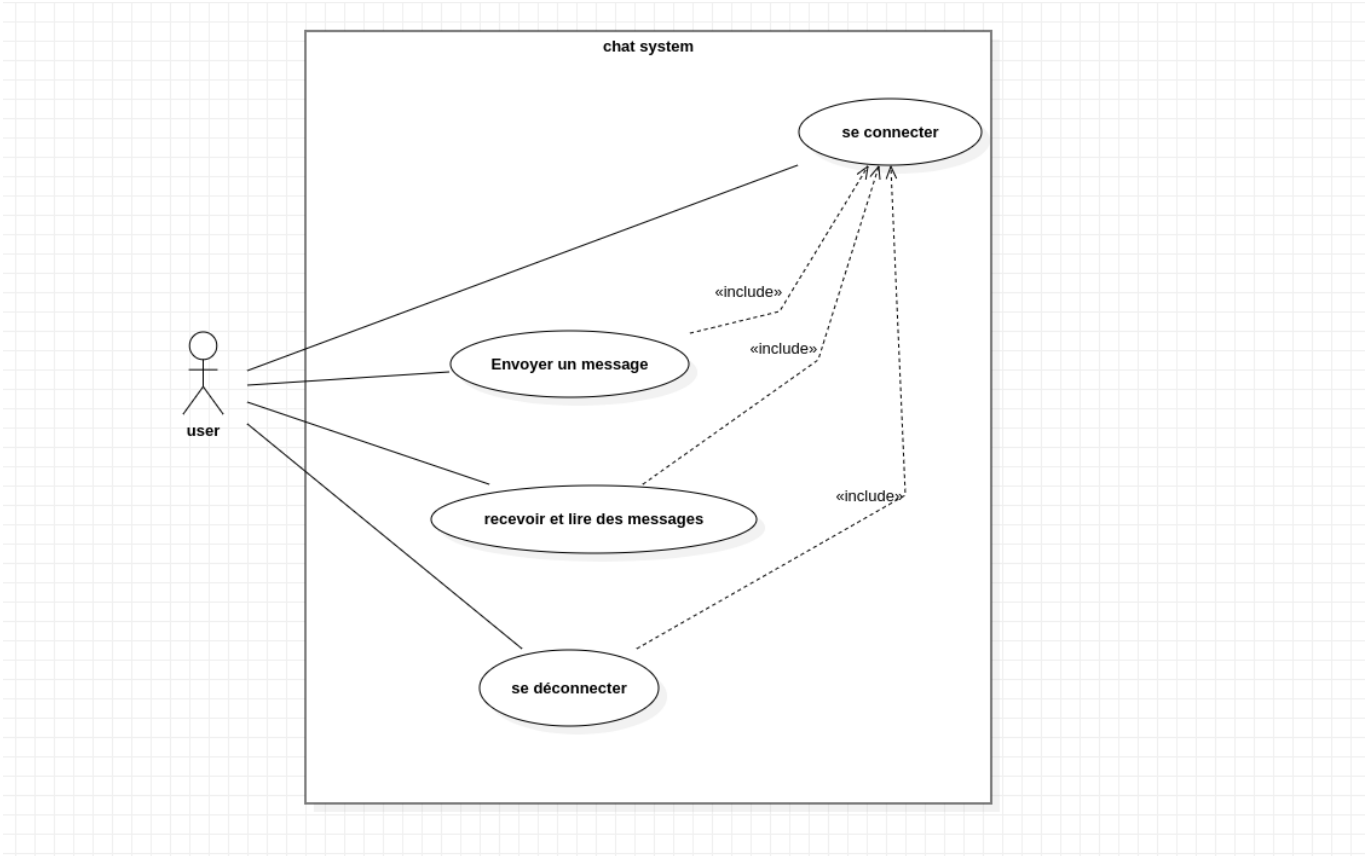
Diagramme de déploiement



Le modèle de déploiement décrit est un modèle de déploiement typique pour une application WebSocket utilisant Spring Boot. Le client, dans ce cas un navigateur web, se connecte au serveur WebSocket en utilisant le protocole WebSocket. Le serveur WebSocket est hébergé sur un serveur web Apache, qui exécute à son tour une application Spring Boot. L'application Spring Boot fournit le registre des points d'extrémité WebSocket, qui est responsable de la gestion des différents points d'extrémité WebSocket dans l'application.

Lorsqu'un client se connecte au serveur WebSocket, il envoie d'abord un message de prise de contact au serveur. Le serveur répond ensuite par un message de prise de contact, qui confirme que la connexion a été établie. Une fois la connexion établie, le client et le serveur peuvent échanger des messages à l'aide du protocole WebSocket.

Diagramme des cas d'utilisations



Le diagramme des cas d'utilisation montre les différentes interactions entre l'utilisateur et le serveur WebSocket. L'utilisateur peut se connecter et se déconnecter, envoyer des messages et les lire. Le serveur WebSocket est responsable de la validation des informations d'identification, de la diffusion des messages et de la livraison des messages aux utilisateurs.

Voici quelques détails supplémentaires sur les cas d'utilisation :

- Connexion : L'utilisateur saisit son nom d'utilisateur et son mot de passe et les envoie au serveur WebSocket. connecte l'utilisateur si elles sont valides.
- Déconnexion : L'utilisateur envoie une demande de déconnexion au serveur WebSocket. Le serveur WebSocket déconnecte l'utilisateur et le supprime de tous les canaux.
- Envoyer un message : L'utilisateur saisit un message et l'envoie au serveur WebSocket. Le serveur WebSocket diffuse le message à tous les utilisateurs abonnés au canal.
- Lire un message : L'utilisateur reçoit un message du serveur WebSocket. L'utilisateur affiche le message dans son interface utilisateur.