Rapport du projet Chat

Application de discussion en ligne avec Django

Réalisé par :

Ben Issa Ranim
DIOUF Aminta
FRANCOIS-BATTAGLIA Enzo

21 janvier 2026

Table des matières

1 Introduction

Dans le cadre de ce projet de développement web, nous avons réalisé une application de discussion en ligne à l'aide du framework **Django**. L'objectif principal est de permettre à des utilisateurs authentifiés d'échanger des messages au sein de **salons** et de **canaux (channels)**, tout en respectant une architecture web structurée et sécurisée.

Le système propose plusieurs fonctionnalités clés, notamment :

— la création de comptes utilisateurs et l'authentification ;

— la gestion des salons et des canaux ;

— l'envoi, la modification et la suppression de messages ;

— des actions de modération (promotion, bannissement, etc.).

Ce projet vise également à appliquer concrètement les connaissances vues en cours : conception de modèles de données, routage, vues, authentification et communication entre frontend et backend.

2 Choix techniques et organisation du projet

2.1 Framework et architecture

Le framework **Django** a été choisi pour sa structure claire et ses fonctionnalités intégrées : gestion de l'authentification, routage des URL, ORM (Object-Relational Mapping) et bonnes pratiques de sécurité.

L'architecture repose sur le modèle **MVT** :

— **Model** : structure des données et relations (ORM) ;

— **View** : logique serveur, traitement des requêtes, réponses JSON ou HTML ;

— **Template** : rendu HTML côté client.

2.2 Base de données

La base de données est gérée via l'ORM de Django, ce qui évite l'écriture manuelle de requêtes SQL et réduit les risques d'erreurs. Les modèles ont été conçus pour représenter les entités principales : utilisateurs, salons, canaux et messages.

2.3 Interface utilisateur

L'interface repose sur des templates HTML, avec **Bootstrap** pour obtenir une interface responsive et moderne. Les formulaires Django permettent une validation robuste des entrées (sécurité, contraintes, feedback utilisateur).

3 Modélisation de la base de données

La modélisation a été conçue pour représenter simplement et efficacement le fonctionnement d'une application de discussion en ligne.

3.1 Salon

Le **Salon** correspond à un espace de discussion principal. Il possède :

— un nom (unique) ;

— un *slug* (identifiant lisible pour les URL) ;

— une description optionnelle ;

— un créateur (administrateur principal).

3.2 Channel (Canal)

Le **Channel** est associé à un salon et permet d'organiser les discussions par thèmes. Chaque canal appartient à un seul salon, et une contrainte d'unicité empêche la duplication de canaux portant le même nom dans un salon.

3.3 Message

Le **Message** représente les messages envoyés par les utilisateurs :

— un auteur ;

— un contenu textuel ;

— une date d'envoi ;

— éventuellement un fichier (upload) ;

— rattachement à un salon **ou** à un channel.

3.4 Rôles et bannissements (modération)

Pour la modération, le projet gère :

— des rôles (ex. modérateur) via un modèle de liaison (`SalonRole`) ;

— des bannissements via un modèle dédié (`Ban`) indiquant l'utilisateur, le salon, l'état actif et la raison.

4 Gestion de l'authentification et des utilisateurs

L'authentification utilise le système intégré de Django :

— Inscription (création de compte) ;

— Connexion / Déconnexion ;

— Accès protégé aux salons et canaux via le décorateur `@login_required`.

4.1 Redirection `login_required`

Pour éviter les erreurs de redirection vers `/accounts/login/`, une configuration `LOGIN_URL` permet de rediriger vers la route de connexion du projet (ex. `/connexion/`).

5 Interaction frontend / backend

Le backend est géré par Django et le frontend par des templates HTML + Bootstrap.

5.1 API et requêtes asynchrones

Pour offrir une expérience fluide, l'application utilise des requêtes AJAX :

— affichage des messages (salon / channel) sans rechargement complet ;

— envoi de messages ;

— modification et suppression ;

— actions de modération (ban / unban / promote / demote).

Cette séparation permet de garder une logique serveur claire tout en améliorant l'ergonomie côté client.

6 Fonctionnalités principales

6.1 Salons et canaux

— Création d'un salon ;

— Suppression d'un salon ;

— Création d'un canal dans un salon ;

— Accès à un canal et affichage des messages associés.

6.2 Messagerie

— Envoi d'un message (texte et/ou fichier) ;

— Affichage chronologique des messages ;

— Édition d'un message (auteur) ;

— Suppression d'un message (auteur et/ou modérateur).

6.3 Modération

— Liste des utilisateurs présents dans un salon ;

— Bannir / débannir un utilisateur avec raison optionnelle ;

— Promouvoir / rétrograder un utilisateur en modérateur ;

— Contrôles côté serveur (permissions) pour sécuriser les actions.

7 Résultats (captures d'écran)

Cette section présente quelques captures illustrant le fonctionnement de l'application.

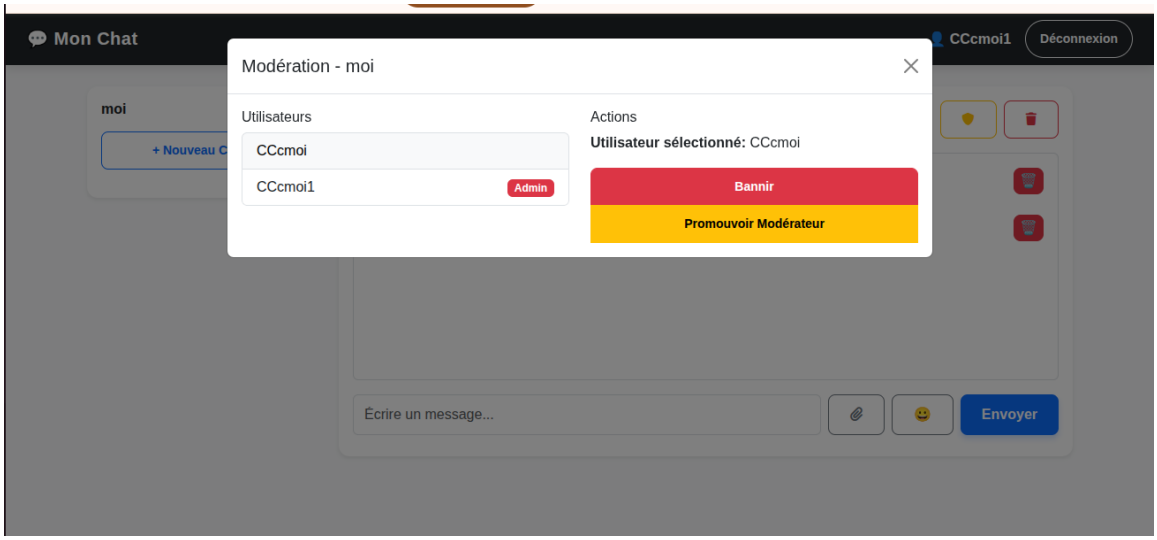

FIGURE 1 – Interface de discussion et actions de modération (ex. suppression/édition).

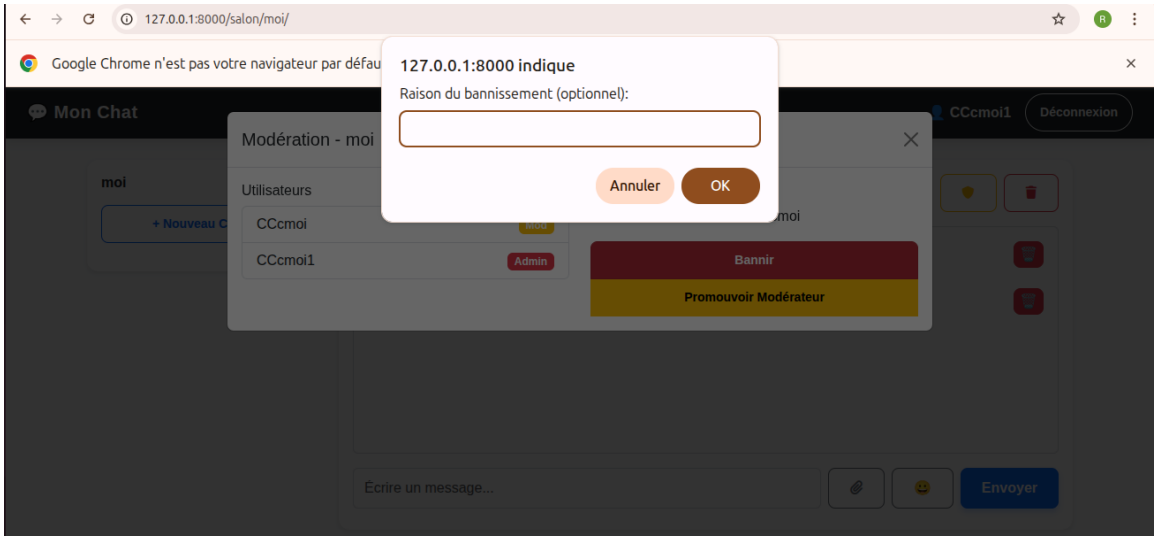

FIGURE 2 – Fenêtre de modération : liste des utilisateurs et actions (ban/promotion).

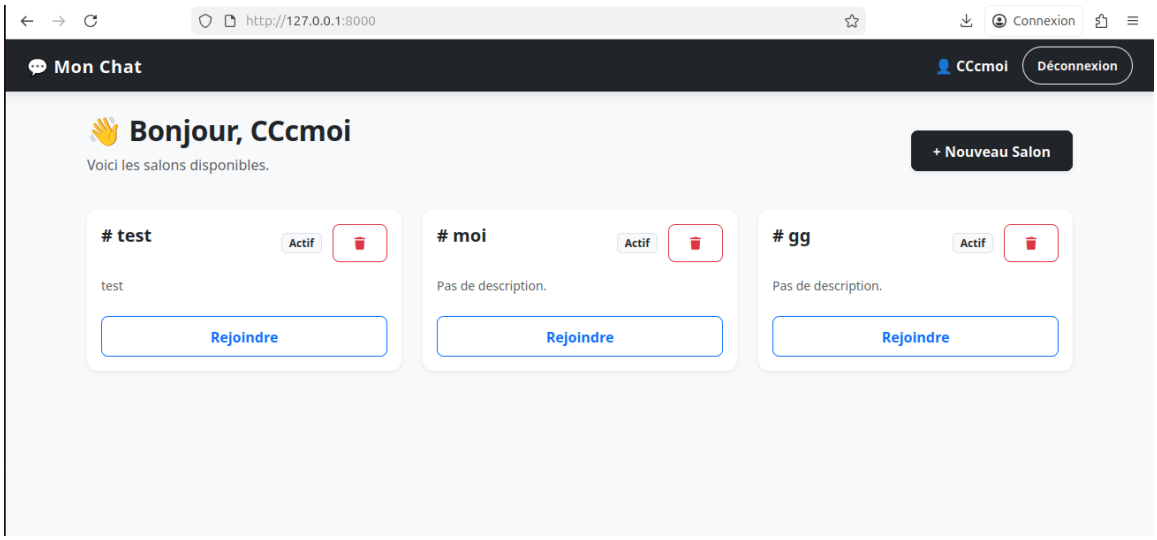

FIGURE 3 – Page d'accueil : liste des salons disponibles et accès rapide.

8 Difficultés rencontrées et solutions

8.1 Gestion des permissions de modération

Une difficulté fréquente concerne la cohérence des permissions côté client et côté serveur. La solution consiste à :

— centraliser les règles de permissions côté backend (admin/modérateur/auteur) ;

— retourner les informations nécessaires (rôle, ban, etc.) via l'API ;

— afficher/masquer les boutons côté frontend selon ces permissions, tout en gardant la sécurité côté serveur.

8.2 Redirections d'authentification

Les routes de connexion personnalisées (ex. `/connexion/`) nécessitent une configuration `LOGIN_URL` adaptée pour éviter les 404.

9 Conclusion

Ce projet a permis de concevoir et développer une application de discussion en ligne complète, en appliquant les notions vues en cours sur Django : modèles, vues, routage, authentification et interactions frontend/backend.

Les choix réalisés (architecture salons/canaux, AJAX pour la fluidité, gestion des rôles et du bannissement, Bootstrap pour l'interface) aboutissent à une application fonctionnelle, sécurisée et maintenable.