**Rapport et lien :**

**Lien :** chatlite.alwaysdata.net

**Rapport :**

**Introduction**

Dans le cadre du cours *Technologies Web II / Programmation Web*, nous avons développé une application de chat centralisée répondant au cahier des charges fourni. Ce projet avait pour but d’implémenter une solution complète combinant les aspects client et serveur, avec un rendu dynamique des pages et une communication en temps réel.

**Technologies et outils utilisés**

Pour résoudre les différents aspects de ce projet, nous avons fait appel aux technologies et bibliothèques suivantes :

**1. Côté serveur : Django**

* **Pourquoi Django ?** Django, un framework Python, nous a permis de construire rapidement un serveur robuste et sécurisé. Sa gestion intégrée des modèles, des vues et des templates nous a simplifié l'implémentation des fonctionnalités suivantes :
  + Gestion des utilisateurs (inscription et connexion).
  + Création et gestion des salons de discussion.
  + Modération et gestion des droits des utilisateurs.
* **Principales fonctionnalités implémentées avec Django :**
  + Création des modèles pour les salons, utilisateurs et messages.
  + API pour les interactions dynamiques entre le client et le serveur.

**2. Websockets : Daphne et Django Channels**

Pour la gestion des messages en temps réel, nous avons utilisé :

* **Daphne :** Serveur ASGI permettant de supporter la communication asynchrone.
* **Django Channels :** Intégré à Django, il nous a permis d'ajouter une gestion des WebSockets afin de permettre une expérience de chat en direct sans rafraîchissement de la page.

**3. Côté client : HTML, CSS et JavaScript**

* **HTML et CSS :**
  + Utilisation pour structurer et styliser les pages web.
  + Les salons de discussion et interfaces utilisateur (connexion, salons) sont dynamiques et responsives.
  + Bootstrap a été utilisé pour assurer une mise en page harmonieuse.
* **JavaScript :**
  + Gestion des interactions dynamiques du côté client, comme l’envoi de messages via WebSockets.
  + Animation des salons et affichage en direct des messages.

**Architecture de l’application**

**1. Backend**

* **Gestion des modèles :** Les données sont structurées autour des modèles User, Salon et Message.
* **Gestion des WebSockets :** L’utilisation de Django Channels a permis de créer des routes asynchrones pour gérer en temps réel les messages des utilisateurs.

**2. Frontend**

* Les pages principales incluent :
  + Une page de connexion et d’inscription.
  + Une liste des salons accessibles par l’utilisateur connecté.
  + Une interface de salon avec messages dynamiques et gestion des emojis.

**3. Communication en temps réel**

Le mécanisme des WebSockets, via Django Channels et Daphne, permet d’assurer une mise à jour immédiate des messages envoyés dans un salon de discussion.

**Détails des choix techniques**

**Django Channels et Daphne**

* **Choix :** Nécessité d’une communication bidirectionnelle pour le chat en temps réel.
* **Avantage :** Permet de se conformer au cahier des charges (mise à jour dynamique) tout en s’intégrant parfaitement avec l’écosystème Django.

**Séparation Frontend et Backend**

* La gestion des messages, salons et utilisateurs est centralisée dans le backend. Le frontend se concentre sur l’expérience utilisateur.

**Utilisation de Bootstrap**

* **Pourquoi ?** Pour garantir une interface claire, responsive, et facile à naviguer sans développer un design complexe à partir de zéro.

**Défis rencontrés**

1. **Intégration des WebSockets dans Django :**  
   La configuration initiale des Channels et de Daphne a nécessité une phase d’apprentissage.
2. **Optimisation des performances :**
   * Gestion efficace des salons ayant un grand nombre de messages.
   * Minimisation de la charge serveur pour chaque salon actif.
3. **Coordination en équipe :**  
   Utilisation de Git pour le contrôle de version et la collaboration.

**Résultats obtenus**

L’application finale respecte les exigences suivantes :

* Communication en temps réel grâce aux WebSockets.
* Gestion multi-salons avec droits utilisateur.
* Interface utilisateur moderne et fonctionnelle.
* Fonctionnalités avancées comme la gestion des emojis et la modération.

L’application est hébergée à l’adresse suivante : chatlite.alwaysdata.net

**Conclusion**

Ce projet nous a permis de maîtriser des outils modernes pour le développement d’applications web dynamiques et en temps réel. La combinaison de Django, Daphne, Channels et des technologies frontend (HTML, CSS, JavaScript) a permis de produire une application répondant pleinement aux attentes.