UNIVERSITÉ DE SHERBROOKE  
DÉPARTEMENT D’INFORMATIQUE

IFT 604 – Applications internet et mobilité

Rapport de projet

Travail présenté à

M. Armel Ayimdji Tekemetieu

Par

*Guillaume Cléroux - cleg1805*

*Marc-Étienne Pépin - pepm3004*

*Maxim-Olivier Gélinas - gelm1302*

*Camille Gauthier - gauc3305*

Le 7 décembre 2023

Pour le projet de session dans le cours Applications internet et mobilité, notre équipe a décidé d’implémenter un chat de proximité. Dans le présent rapport, nous allons décrire les fonctionnalités des différentes parties du projet (backend, application web et application mobile).

# Backend

Voici une liste des fonctionnalités de l’application backend :

* **Système distribué avec algorithme RAFT** : L'application utilise un algorithme RAFT pour les élections au sein d'un système distribué. Ce mécanisme garantit la cohérence et la fiabilité dans le processus de prise de décision.
* **Gestion des élections** :
  + **Leader et Followers** : Le premier nœud qui rejoint le cluster devient le leader. Les nœuds qui rejoignent ensuite sont considérés comme des followers.
  + **Log de Messages** : Le leader contrôle le log de tous les messages échangés depuis le lancement de l'application, assurant ainsi la cohérence des données. Seul le leader peut écrire dans ce log.
  + **Élection d'un Nouveau Leader** : En cas de défaillance du leader, les autres nœuds procèdent à une nouvelle élection pour désigner un remplaçant.
* **Traitement des demandes d'écriture** : Les nœuds reçoivent et traitent les demandes d'écriture en les transmettant au leader, qui les diffuse ensuite. Chaque nœud garde une trace de l'origine des demandes.
* **Communication Interne par gRPC** : L'utilisation de gRPC permet de maintenir des sessions persistantes et de s'abonner à un flux de messages. Les messages sont ainsi poussés dans le flux en temps réel.
* **Format de données Protobuf** : Au lieu du format JSON, l'application utilise Protobuf pour la sérialisation des données, offrant ainsi rapidité et compatibilité multiplateforme.
* **Reverse-Proxy gRPC avec API REST** : Chaque nœud dispose d'un reverse-proxy gRPC qui expose une API REST. Les requêtes REST envoyées au nœud sont traduites en gRPC, combinant la simplicité de REST avec l'efficacité de gRPC.
* **Serf pour le Service Discovery** : L'algorithme Serf est utilisé pour la découverte automatique des nœuds lors de l'ajout de nouveaux éléments au serveur, assurant une gestion efficace de la connectivité au sein du cluster.
* **Sécurité et encryption avec mTLS** : L'application assure la sécurité des communications et des données à travers une encryption mutuelle TLS (mTLS).
* **Listes d'Accès de Contrôle** : Seules les personnes détenant un certificat d’utilisateur valide peuvent accéder et communiquer sur le service, garantissant ainsi un haut niveau de sécurité et de contrôle d'accès.

# Application web

Voici une description des fonctionnalités et de l’architecture du client web de chat de poximité :

* **Utilisation du Framework React** : L'application client est construite avec React, un framework populaire pour le développement d'interfaces utilisateur dynamiques et réactives.
* **Gestion des Routes avec React Router** :
  + Une route mène à la liste des chats (ListeChat).
  + Une autre route permet d'accéder à une session de chat spécifique (Chat).
* **API Calls et Gestion des Données** :
  + **Fichier apiCall.tsx** : Contient des fonctions pour interagir avec l'API backend. Ces fonctions gèrent les appels API pour lire et écrire des messages, et pour récupérer des données.
* **Composant UI Chat** (chat.tsx) :
  + **Interaction en Temps Réel** : Gère l'affichage et l'envoi de messages en temps réel.
  + **État et Effets** : Utilise les hooks useState et useEffect de React pour gérer l'état local et les effets secondaires.
* **Composant UI ListeChat** (listeChat.tsx) :
  + **Affichage Dynamique des Serveurs** : Affiche une liste de serveurs disponibles, utilisant la géolocalisation pour personnaliser l'expérience utilisateur.
  + **Navigation Facile** : Permet aux utilisateurs de sélectionner un serveur et de naviguer vers la session de chat correspondante.
* **Bibliothèque Matériel UI** :
  + L'application utilise Material-UI pour les éléments d'interface.

# Application mobile

1. **Architecture et Navigation** :
   * L'application est construite avec React Native.
   * **Écrans Principaux** : Comprend deux écrans principaux - 'Home' pour la liste des chats (ListeChat) et 'Chat' pour la messagerie individuelle (Chat).
2. **Liste des Chats (ListeChat.js)** :
   * **Localisation et Serveurs Proximité** : Utilise expo-location pour obtenir la localisation actuelle de l'utilisateur et affiche une liste des serveurs de chat à proximité.
   * **Navigation Dynamique** : Permet aux utilisateurs de sélectionner un serveur de chat et de naviguer vers l'écran de chat correspondant avec des interactions tactiles.
3. **Chat (Chat.js)** :
   * **Gestion des Messages** : Permet aux utilisateurs d'envoyer et de recevoir des messages en temps réel.
   * **Stream de Données** : Utilise un appel fetch pour lire un stream de données depuis le backend, permettant une mise à jour dynamique des messages.
   * **Interface Utilisateur** : Champ de saisie de texte et un bouton d'envoi.
4. **Fonctionnalités Clés** :
   * **Gestion de l'État** : Utilise les hooks React useState et useEffect pour la gestion de l'état local, et de la mise à jour de l'interface utilisateur.
   * **Communication avec le Backend** : Les requêtes réseau sont gérées via fetch pour interagir avec le serveur backend, permettant des opérations telles que l'obtention des serveurs à proximité et l'envoi/réception de messages.