INTERNET OF THINGS

Projet de création d'une application de chat en invite de commande sous le paradigme client-server

Sous la supervision de Pr David Bromberg

Sommaire

I. Use cases implémentés 2
II. Les fonctionnalités implémentées 2
1. TD3 2
2. TD4 2
III. Explication conceptuelle de l'API 2
IV. Documentation de l'API 3
V. Fonctionnalités manquantes 5

I. Use cases implémentés

Les uses cases que nous avons implémentés sont :

- Client (nodeJs) / serveur (nodeJs) : JSON over TCP/IP
- Client (nodeJs) / serveur (nodeJs) : websocket

II. Les fonctionnalités implémentées

1. TD3

- Connexion du client et réponse du serveur par un « *Hello* »
- Envoie des messages privés **send**
- Envoie des messages à toutes les personnes connectées **broadcast**
- Avoir la liste des utilisateurs connectés au serveur **list**
- L'utilisateur se déconnecte du serveur **quit**
- L'utilisateur est automatiquement notifié lorsqu'une personne se connecte

2. TD4

- Créer un groupe
- Joindre un groupe
- Envoyer des messages dans un groupe
- Avoir la liste des membres d'un groupe
- Avoir la liste des messages envoyés dans le groupe
- Avoir la liste des groupes existant
- Quitter un groupe
- Invité un utilisateur dans un groupe
- Ejecté un membre du groupe
- Exclure définitivement un membre du groupe
- Ne plus exclure définitivement un membre du groupe
- Liste de tous les évènements survenus dans le groupe
- Implémentation des groupes publiques et privés

III. Explication conceptuelle de l'API

La conception de notre API suit les séquences :

- Le client se connecte au serveur via un numéro de port et une adresse ip indiquée par le serveur lorsque celui-ci est démarré
- Une fois connecté le serveur lui envoie un feedback pour attester de sa connexion
- Lorsque le client souhaite effectuer une requête, il formalise sa requête et l'envoie au serveur sur son socket.
- Le serveur reçoit la requête, effectue l'action associée et écrit sur le socket pour envoyer sa réponse au client.
- Le client peut ainsi lire la réponse sur son socket et afficher le résultat de celle-ci

IV. Documentation de l'API

Les formats de messages du client ayant déjà été spécifiés nous n'allons donner que les formats de messages du serveur. Il est à noter que ces messages seront passés en paramètres de la fonction « stringify » de l'objet JSON.

- Connexion au serveur « client-hello »
- Reponse du serveur au client **{action : "server-hello", description : "welcome "+msg.from}**
- Notification du serveur aux utilisateurs **{description: "server> "+msg.from + " is connected",action : "hello-notifier"}**
- Envoie des messages privés « client-send » **:** **{description :`Received from ${msg.from} : ${msg.msg}`,action : "foward"}**
- Envoie des messages à toutes les personnes connectées « client-broadcast » **:** **{description : `Received from ${msg.from} : ${msg.msg}`, action : "broadcast",}**
- Avoir la liste des utilisateurs connectés au serveur « client-list-clients » : **{msg : list, description : "server> list of all connected users : ",action : "list"}**
- L'utilisateur se déconnecte du serveur « client-quit» :
- Reponse du serveur au client **{description : `server> you have been disconnected`,action : "quit-attestation"}**
- Notification du serveur aux utilisateurs **{description :`server> ${msg.from} is disconnected`,action : "quit" }**
- Créer un groupe « cgroup » :

  On dispose d'un tableau de groupes composés d'objets littéraux de la forme

  « groupName » :{group_members: members, group_messages : messages, kick_members : kick, ban_members : ban, group_events : events, }

  Chaque élément de l'objet littéral est un tableau.

  Le format de la reponse du serveur est **{description : `server> group ${msg.group} created successfully`,action: 'create-group'}**

- Joindre un groupe « join » :

{msg : groups.get(msg.group).group_members, description : "server> you joined the group successfully",action: 'join-group' }

- Envoyer des messages dans un groupe « bgroup » :
  {description : `${msg.group}> ${msg.from} : ${msg.msg}`,group : msg.group, action : "group-broadcast"}
- Avoir la liste des membres d'un groupe
  {msg : groups.get(msg.group).group_members, description : `server> list of all the members of the group ${msg.group}`, action: 'group-members' }
- Avoir la liste des messages envoyés dans le groupe
  {msg : groups.get(msg.group).group_messages, description : `server> list of all the messages of the group ${msg.group}`,action: 'group-messages' }
- Avoir la liste des groupes existant :
  C'est la variable cle qui contient la liste des groupes
  {msg : cle, description : "server> list of all existing groups",action: 'groups-list'}
- Quitter un groupe
  {group : msg.group, description : `server> vous avez quitté le groupe ${msg.group}`,action: 'group-leave'}
- Invité un utilisateur dans un groupe
  {description : `server> you successfully added ${msg.to} to the group :${msg.group} `,action: 'group-invitation'}
- Ejecté un membre du groupe

- Reponse du serveur au client **{group : msg.group, description : `server> you have dismissed ${msg.to} from ${msg.group} `,action: 'group-kick'}**
- Notification du serveur aux membres du groupe
  {description : `server> you have been dismissed by ${msg.from} to the group : ${msg.group} because${msg.reason}`,action: 'kick-feedback'}

- Exclure définitivement un membre du groupe

- Reponse du serveur au client
  {description : `server> you have been dismissed by ${msg.from} to the group : ${msg.group} because${msg.reason}`,action: 'kick-feedback'}
- Notification du serveur aux membres du groupe
  {description : `server> you have been dismissed by ${msg.from} to the group : ${msg.group} because${msg.reason}`, action: 'ban-feedback'}

- Ne plus exclure définitivement un membre du groupe

- Reponse du serveur au client
  {group : msg.group, description : `server> you have dismissed ${msg.to} from ${msg.group} `,action: 'group-ban'}
- Notification du serveur aux membres du groupe
  {description : `server> you have been admitted by ${msg.from} to the group : ${msg.group}`,action: 'unban-feedback'}

- Liste de tous les évènements survenus dans le groupe

{msg : groups.get(msg.group).group_events, description : `server> list of all the events of the group ${msg.group}`,action: 'group-events'}

V. Fonctionnalités manquantes

- Interruption Ctrl-C
- Le code n'a pas été modulé
- La base de données
- La gestion des sessions
- Chiffrement des messages