IRC – DOCUMENTATION

Table des matières

1. Présentation du projet

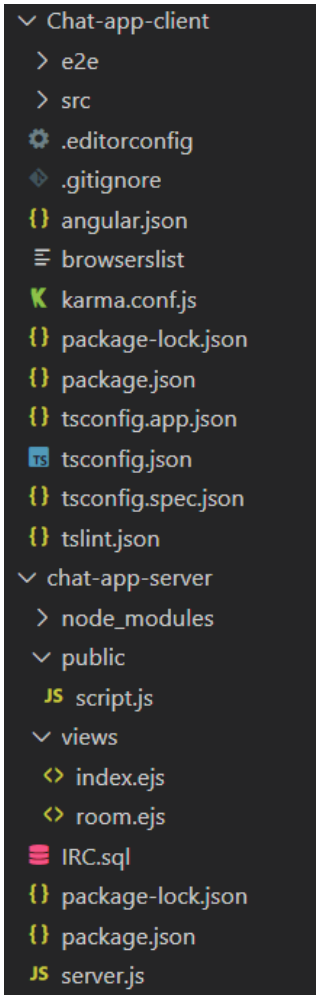
Ce projet consiste à réaliser en NodeJS et ExpressJS un chat textuel dans lequel les utilisateurs peuvent rejoindre plusieurs channels simultanément, ainsi que créer, renommer et supprimer des channels.

Le client, lui, doit être codé en Angular.

2. Architecture

Notre projet est divisé en deux parties, la partie back et la partie front.

Voici une arborescence de nos documents :



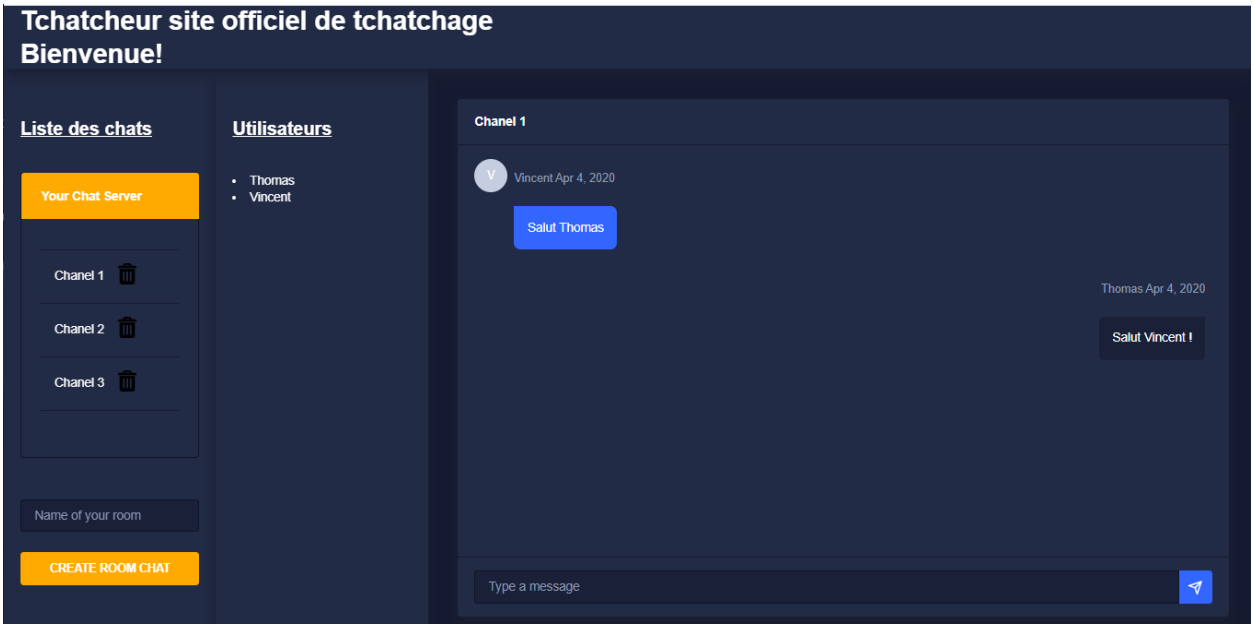
3. Base de données

Pour des raisons de simplicité, nous n'avons créé qu'une seule table. Cette table 'messages' contient un id, le contenu du message, la date, le nom de l'utilisateur ayant écrit ce message, et la room dans laquelle il a écrit ce message :

Messages
id int(11) NOT NULL, **message** varchar(2048) NOT NULL, **date** datetime NOT NULL DEFAULT current_timestamp(), **user** varchar(24) NOT NULL, **room** varchar(64) NOT NULL

4. Mock-up

Voici à quoi ressemble notre solution finale :



De gauche à droite :

- La liste des channels ainsi que les boutons pour les supprimer ou en créer de nouveaux
- La liste des utilisateurs dans le channel qu'on utilise
- Le chat, c'est-à-dire la liste des messages envoyés avec le nom des utilisateurs et la date à laquelle le message a été envoyé

5. Technologies utilisées

Comme demandé, notre serveur a été développé en NodeJS et notre client en Angular.

6. Routes API

Routes du serveur :

* "/home" :

  Method : "GET"

  Récupère les rooms actuellement actives

* "/room" :

  Method : "POST"

  Body-param : room

  Crée une nouvelle room avec le nom précisé dans le Body

* "/delete/${roomName}" :

  Method : "DELETE"

  Supprime le channel portant le nom en parametre d'url

7. Messages socket

PORT 3000

Messages entrants :

* "new-user", (room, name) :

  Messages à envoyer quand l'utilisateur se connecte à une room

* "send-chat-message", (room, message) :

  Envoyer un message dans une room

Messages sortants :

* "chat-message-log", Object : {users: {key, nom}, messages: {user, message, date}}

  Quand un "new-user" est reçu, le serveur envoie tous les messages et les utilisateurs actuellement connectés à la room

* "chat-message", (room, message) :

  Nouveau message Y dans la room X

* "room-created", roomName :

  Nouvelle room

* "room-deleted", roomName

  Room suprimée

* "user-connected", (room, user) :

  Nouvel utilisateur connecté à une room

* "user-disconnected" (user, room) :

  User déconnecté de la room