ISEN OUEST

TP N°4 - Communications Web
WebSocket

Thibault Napoléon
thibault.napoleon@isen-ouest.yncrea.fr

Notions abordées :

- ✓ Création du canal de communication
- ✓ Communication client → serveur
- ✓ Communication serveur → client

Préambule

Le but de ce TP est de comprendre la communication à travers les *WebSockets* grâce à un exemple simple et concret en *JavaScript*. Afin de ne pas perdre de temps sur l'aspect visuel, mais aussi pour respecter les standards actuels d'affichage sur téléphone et tablette (*Reponsive Design*), nous utiliserons le framework *CSS* nommé « `Bootstrap` » : https://getbootstrap.com/docs/
Nous utiliserons aussi par moment la bibliothèque d'icônes « `Font Awesome` » pour plus de lisibilité : https://fontawesome.com/

Le but du TP est d'écrire le code *JavaScript* permettant de mettre en place un chat entre utilisateurs. Le serveur, écrit en *C++* avec la bibliothèque *Qt*, s'exécute sur une machine distante dont l'`IP` et le port vous seront fournis. Son rôle est de retransmettre chaque message reçu à l'ensemble des utilisateurs qui lui sont connectés. L'aspect visuel du chat pourra ressembler à la figure 1.

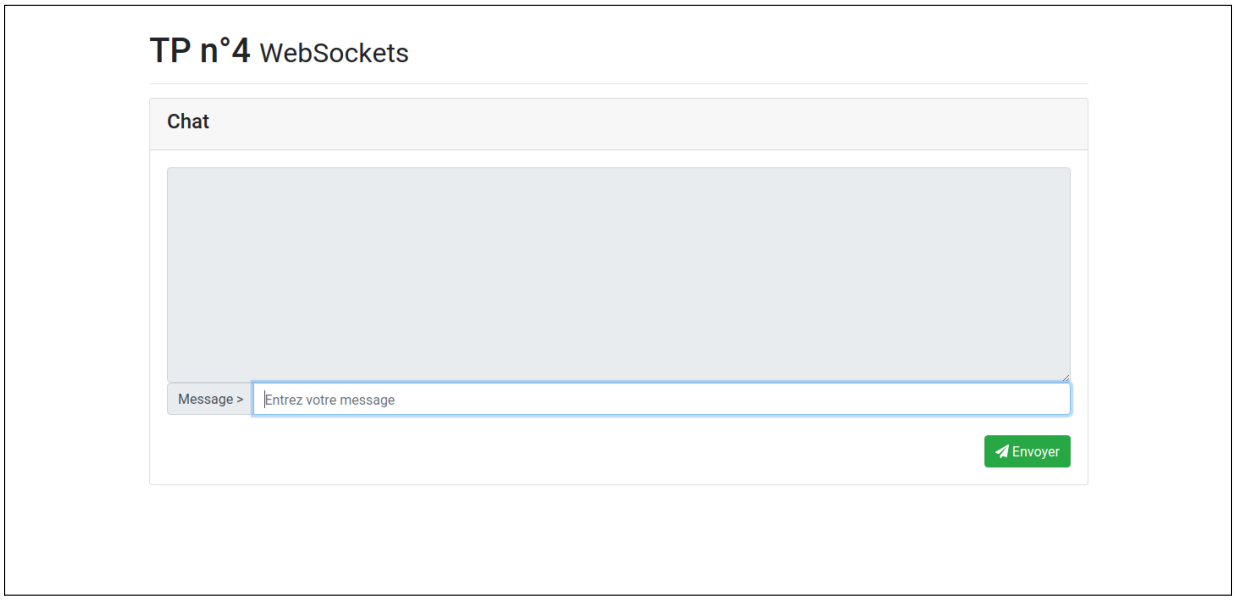

FIGURE 1 – *Capture d'écran du chat utilisant des WebSockets.*

1 Structure *HTML* du chat

Dans cette partie on souhaite écrire le code *HTML* permettant de donner la structure du chat comprenant :

- Un espace d'affichage des messages ;
- Un champ de texte pour entrer un nouveau message ;
- Un bouton d'envoi.

Pour écrire cette structure vous devrez écrire un fichier « index.html » contenant le code de base d'une page *HTML* (`DOCTYPE`, en-tête, corps, titre...), que vous pourrez retrouver dans les TP précédents, ainsi que :

- Un `Card` *BootStrap* (cliquez ici pour la documentation) contenant un en-tête ;
- Le corps (*body*) de ce `Card` sera composé d'un formulaire d'id `chat-send` contenant :
  - Un `textarea` d'id `chat-room` en `readonly` et `disabled` ;
  - Un `input-group` *BootStrap* (cliquez ici pour la documentation) dont l'`input` sera d'id `chat-message` ;
  - Un `button` « Envoyer » (cliquez ici pour la documentation) de type `submit`.

2 Création de la *WebSocket* avec le serveur en *JavaScript*

Dans cette partie on souhaite écrire le code permettant de dialoguer avec le serveur de chat au travers d'une *WebSocket*. Pour cela, créez un script « chat.js » dans le dossier « js ». Pensez à inclure ce script dans votre page *HTML*.

Création du canal de communications

Avant toute chose ajoutez une variable globale `websocket` (ce sera le canal de communications entre le client et le serveur) ainsi qu'une variable `login` contenant votre « pseudo ». Écrivez et appelez ensuite la fonction sans paramètre `createWebSocket`.

Dans celle-ci, initialisez votre `socket` avec le constructeur de `WebSocket`. Vous utiliserez l'adresse `IP` et le port du serveur donné au tableau. Ajoutez un `callback` à l'évènement `onmessage` qui permettra d'ajouter, dans notre fenêtre de chat, les messages reçus du serveur. Pour cela, vous pourrez utiliser le code suivant :

```
textArea.value +=  ...;
```

Où `textarea` est l'élément *HTML* correspondant à votre zone de texte. Afin de faire défiler automatiquement la fenêtre de chat vous pourrez utiliser :

```
textArea.scrollTop = textArea.scrollHeight;
```

Gestion de l'envoi d'un message

Afin de pouvoir envoyer des messages sur le chat, il va être nécessaire d'ajouter un gestionnaire d'événement au formulaire. Pour cela, associez à la validation du formulaire (événement `submit`) la fonction `callback` suivante :

```
function sendMessage(event)
```

Celle-ci aura pour rôle :

- De stopper la propagation de l'évènement `submit` ;
- De récupérer le message à envoyer ;
- D'envoyer le message au serveur de chat en le précédent du `login` (défini en variable globale du script).