Rapport du site de chat

Anthony Ferrand
Lucas Nélaupe

10 décembre 2014

1 Introduction

Le but de ce projet est de développer une application web de chat. L'utilisateur doit avoir le choix entre 3 technologies de requêtes asynchrones pour la communication client-serveur : Le "Polling", "Long Polling" et le "Push"

2 Les différents modes de communication

2.1 "Polling"

Le "polling" est la solution la plus naïve (et utilisée) des trois, elle consiste à demander à un serveur à intervalles régulier de nous envoyer les nouveaux messages. Elle ne permet pas une conversation en temps réel. En cas de grand nombre de connexion, le serveur peut être submergé par ces sollicitations. De plus elles sont inutiles dans le cas où aucun nouveau message n'est arrivé entre deux requêtes.

2.2 "Long Polling"

La seconde solution est le "long Polling". Dans ce cas, le client indique au serveur qu'il souhaite recevoir les nouveaux messages dès qu'il y en a. On peut apparenter ça à une inscription sur l'événement. Le problème de cette solution, est que l'inscription ne fonctionne que pour une requête. C'est à dire que une fois que le serveur aura envoyé les nouveaux messages, le client doit se réinscrire à l'événement pour recevoir les prochains. Elle permet de limiter les requêtes inutiles contrairement au "Polling".

2.3 "Push"

Depuis la nouvelle génération du web, nous pouvons désormais utiliser une méthode de "push" grâce aux "websockets". Dans ce cas, on établi une connexion entre le client et le serveur. A chaque nouveau message, le serveur les envoie au client. Dans ce cas il n y a plus de requêtes inutiles. Cependant, cette technologie encore récente n'est pas disponible sur tous les navigateurs.

3 Choix des technologies

Nous avons donc choisi de développer notre application web en se basant sur un serveur "NodeJS". Cette technologie nous permet de mettre en place le système de websocket. Les autres méthodes utilisent des protocoles "Ajax" donc ne nécessitent pas de serveur particulier. Par contre, le serveur NodeJS donne accès aux requêtes du client, ainsi nous pouvons les garder en mémoire pour y répondre au besoin dans le cas du long polling.

4 Fonctionnement

Le serveur enregistre la liste de tous les messages qui lui sont envoyés. Cela permet de gérer les 3 modes de manière asynchrone. C'est à dire qu'un client utilisant la méthode "push" de *websocket* peut communiquer avec un client utilisant une autre méthode comme celle du "Polling" par exemple. Dans ce cas, celui en méthode "Push" aura les messages de son interlocuteur en temps réel, mais la réciproque n'est pas vrai.

5 Cas d'utilisation

Quand le client arrive sur l'application il choisit un pseudo et une méthode de communication. Il est ensuite ajouté au salon de discussion principal. Il peut ensuite créer un salon et ajouter les utilisateurs qu'il souhaite.

6 Conclusion

Notre application web permet une interopérabilité entre les différents modes. Cela évite de refuser un utilisateur sous prétexte que son navigateur ne gère pas la technologie utilisée. Cette méthode est parfois nécessaire en développement web pour permettre une rétrocompatibilité.

A Notice

Notre projet est disponible sur GitHub à l'adresse : https ://github.com/lucas34/WebsoCool. Bien entendu les modules de nodes ne sont pas installés. Il faut donc récupérer le projet et taper les commandes

```
#!/bin/bash
 $ npm install # Installe les modules de node
 $ nodejs app  # Lance le serveur
```

Le site se trouve à l'adresse : http ://localhost :7070