Master 1 MIAGE

Réseaux - Projet STOMP

24 mai 2022

Nathan Courtin - Alexandra Repetti

Théo Gonzalez

Table des matières :

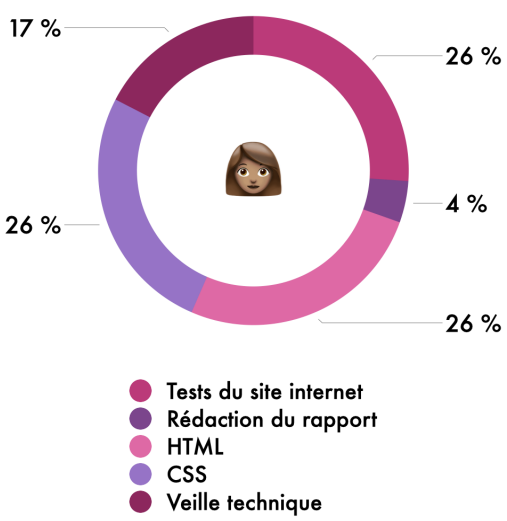
Présentation du projet

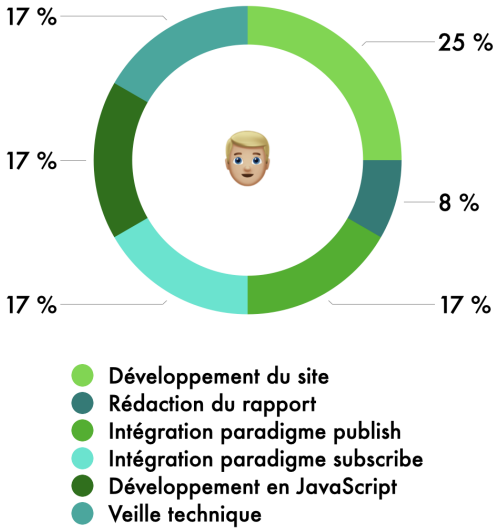
Le projet consiste à développer, en groupe de trois développeurs, un serveur implémentant le protocole STOMP.

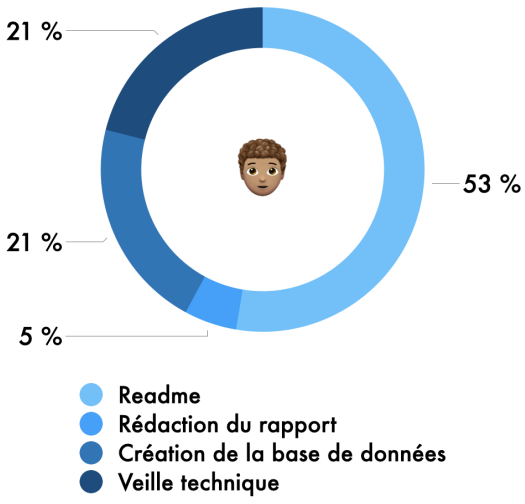
Protocole STOMP

STOMP définit pour les clients et les serveurs un protocole pour communiquer. Il est développé pour fonctionner sur du texte. Son format permet aux clients de communiquer avec le serveur pour permettre une interopérabilité de messagerie simple et étendue. Cette communication est indépendante des langages, des plates-formes et des serveurs utilisés dans l'architecture. Il définit un ensemble de trames qui sont mappées sur les WebSockets. Le protocole STOMP fournit une en-tête de message avec des propriétés et un corps de trame comme AMQP.

Répartition du travail







Architecture du projet :

Le projet se découpe en plusieurs fichiers :

Dans le répertoire principale, on peut retrouver server.js et 3 autres répertoires tels que :

- public regroupant index.html, style.css et client.js ;
- Modele contenant Chat.js qui est un script permettant de créer et paramétrer une base de données ;
- node_module qui regroupe tous les modules installés.

Choix techniques

NODE.JS

Nous avons choisi d'implémenter notre serveur avec la plateforme logicielle libre NODE.JS. Ayant une bonne connaissance de ce dernier, il nous a permis d'obtenir une répartition côté client et une côté serveur plus facilement. De plus, ce langage nous permet d'utiliser des librairies open sources notamment socket.io.

Socket.io est une librairie qui met en relation un client et un serveur tout en assurant une communication rapide et bidirectionnelle. Cette bibliothèque nous a été utile pour implémenter plus facilement le serveur côté client.

Séparation des éléments

Nous avons aussi veillé à bien séparer les différents éléments à savoir : le client et la partie HTML/CSS. Cela permet d'avoir une vision claire de l'affichage et d'une charte graphique du site.

JQUERY

La bibliothèque jQuery permet, entre autres, de gagner en rapidité dans l'interaction avec le code HTML d'une page Web.

Elle propose comme principales fonctionnalités :

- la manipulation du Document Object Model ;
- la gestion des événements (mouvements de souris, clics, etc.).

Nous avons alors fait recours à elle pour récupérer des éléments du code HTML dans notre javascript. Elle nous a également permis de faire la traversée et la manipulation de documents HTML.

Sequelize et Sqlite

Sequelize est un ORM qui nous a permis de gérer la base de données. Ce framework à l'avantage d'être simple à utiliser et permet de nous connecter à notre base SQLite.

Difficultés et solutions :

Durant l'implémentation nous avons rencontré diverses difficultés. Nous avons eu du mal à comprendre les protocoles Websocket et STOMP. Le Websocket est différent des protocoles que l'on a pu utiliser habituellement notamment sur la notion du nombre de requêtes envoyées. Pour pallier cette difficulté nous nous sommes formés à l'aide de formations en ligne sur socket.io.

La seconde difficulté à été de lier le client et le serveur : lors de l'utilisation d'Express. Nous avons également créé un fichier public et visionner des vidéos à son sujet.

Nous avons pris part de créer un serveur à plusieurs channels. Cependant, il a fallu que les messages ne s'écrivent que sur le channel concerné. Pour cela nous avons intégré une base de données afin de stocker les messages dedans. Nous avons également utilisé un attribut room afin de spécialiser le bon channel.

D'autre part, l'implémentation de listener permet de savoir quand on rentre dans un channel et lorsque l'on en sort.

Comme évoqué précédemment nous avons utilisé la bibliothèque jQuery. Cette dernière ne fonctionnait pas pour diverses raisons. Nous avons donc utilisé *document.querySelector* pour récupérer certains éléments de notre HTML.

Une autre difficulté à été de modifier l'html en fonction du statut de l'utilisateur. En effet, quand l'utilisateur est sub, il a la possibilité de voir et d'écrire dans le channel mais pas possible s'il n'est pas inscrit. La difficulté résidait dans la liaison des "room" et de savoir si l'utilisateur était abonné ou non. Nous avons donc dû faire quelques conditions "à la main" afin de pouvoir vérifier ceci.

Implémentation

Ce qui est implémenté :

- un serveur en JavaScript fonctionnant avec Node.JS
    - déclaration du serveur ;
    - instanciation d'express et de Socket.io ;
    - déclaration de la BDD ;
    - fonction permettant l'envoie de messages et la gestion des entrées dans le chat;
    - fonctions permettant de rentrer et sortir d'un channel ;
    - fonctions permettant la mise à jour de la BDD.

- Un client en JS
    - fonction permettant l'envoie de messages au clic sur "envoyer ";
    - fonctions permettant de se balader entre les channels quand on clique dessus;
    - fonction permettant l'initialisation des messages ;
    - fonction permettant de modifier l'interface en fonction de son statut (sub ou non) ;
    - fonction permettant l'affichage des messages envoyés.

- Un html et un css
    - liste des channels à gauche ;
    - chat au milieu ;
    - champs, pour entrer son nom et message, ainsi que le bouton "envoyer" en bas.

Ce qui ne l'est pas :

- Certains éléments de STOMP notamment quelques Client Frames;
- La capacité de se connecter avec un mot de passe;
- La capacité de se déconnecter;
- La possibilité de créer plus de channels;
- L'utilisation de WebSocket car cela ne fonctionnait pas au début du projet puis ensuite ;
- cela était trop tard pour changer;

Comment on a testé le projet :

Pour s'assurer de la bonne implémentation et du bon fonctionnement du serveur nous avons testé grâce à deux navigateurs différents. Nous avons ouvert deux navigateurs puis nous avons envoyé des messages.

Chaque personnes recevaient les messages qu'elles devaient avoir et non des messages destinés à d'autres personnes.

Afin de lancer le serveur, nous exécutons la commande node server dans le terminal en nous situant dans le fichier principal du projet.

Utilisation

Les utilisations du serveurs sont nombreuses. On peut imaginer un channel pour parler avec des personnes d'un de nos chanteurs favoris ou de recettes de cuisine.

On va prendre l'exemple d'une famille qui utilise le serveur pour communiquer entre elles. Elle peut dans le serveur créer des channels par thème. On peut alors imaginer un canal commun pour parler des banalités. Un channel dédié pour l'organisation du cadeau d'anniversaire du petit cousin ou encore un channel pour les courses à faire pour le repas de famille du dimanche.