LOG2420 – Analyse et conception des interfaces utilisateurs

Travail dirigé N°3

Développement d'une application web de clavardage
en temps réel (Partie cliente)
Version 1.0

Professeur : Michel Desmarais
Chargé de laboratoire : Tanga Mathieu Kaboré

École Polytechnique de Montréal
Date : 12 / 03 / 2018

Table des matières

1) Objectif du travail et notions technologiques 3

2) Description du travail 3

a) Barre de menu et entête 3

b) Section de clavardage 4

c) Section de la liste des groupes de discussion 5

3) Exigences détaillées et barème 7

4) Informations sur le serveur 7

a) Évènements supportés 8

b) Format d'un message 8

c) Fichiers et classes de l'application cliente 8

5) Documentation utile 8

6) Icônes 8

7) Concepts utiles 8

8) Questions pertinentes à se poser 8

9) Annexes et quelques séquences 9

1) Objectif du travail et notions technologiques

Le TD3 consiste à implémenter une application cliente de clavardage qui permet de créer un groupe de discussion, de joindre un groupe, d'échanger des messages exclusifs avec les membres d'un même groupe, ou de quitter un groupe de discussion. Un serveur développé avec la technologie *NodeJs* pour ce laboratoire est mis à votre disposition pour vous permettre de vous concentrer sur les aspects du côté client.

Ce dernier travail intègre une grande partie des notions apprises au cours. Il s'agit de mettre en application le patron « Observé-Observateur » qui représente un élément clé du concept MVC (Modèle-Vue-Contrôleur). Il permet d'aborder également les concepts de communication en temps réel en utilisant les « Web Sockets » natifs dans le langage Javascript, en plus de la gestion des évènements avec les éléments du DOM.

Les technologies et concepts utilisés dans ce laboratoire sont :

- HTML/CSS – Javascript / jQuery
- MVC / Observer-Observable
- Interaction d'un WebSocket client et un serveur
- Évènements en javascript
- Sérialisation/Désérialisation – Object Map Javascript

2) Description du travail

a) Barre de menu et entête

La barre de menu dans l'entête contient le logo de la plateforme à gauche, il s'agit d'un lien hypertexte qui permet de rafraichir la page sans faire une redirection, donc aucun traitement spécial n'est requis.

Cependant, la section à droite contient quatre (4) liens contenant des icônes comme dans l'image ci-dessous. Seul le lien encerclé en rouge doit être implémenté afin d'afficher un badge contenant le nombre de messages nouvellement reçus.

Points Bonus pour des fonctionnalités additionnelles :

- Se connecter au serveur avec un autre nom d'utilisateur ;
- Ajouter une option de changement de la langue de l'interface ;

b) Section de clavardage

Comme vous pouvez le constater dans la capture d'écran à l'annexe A, la section est divisée en trois (3) blocs.

L'entête de la vue des messages

Il affiche le groupe actif lorsqu'on clique sur un groupe de discussion dans le panneau latéral droit. **Attention ! L'utilisateur doit joindre un groupe avant de pouvoir y envoyer un message. Consulter les instructions pour joindre un groupe dans la section « Liste des groupes de discussion ».**

La zone de messagerie

Elle contient les messages des autres utilisateurs à gauche, et ceux de l'utilisateur courant à droite. Notez que le nom de l'auteur n'est pas affiché lorsqu'il s'agit de l'utilisateur courant, et la liste des messages est mise à jour et synchronisée en temps réel avec celle des autres utilisateurs d'un même groupe de discussion.

Il se peut que vous ayez besoin de reformater la date et l'heure du message retourné par le serveur pour avoir le même rendu.

Les différents types de messages supportés par le serveur que vous devez afficher dans la boîte des messages ainsi que les évènements associés sont :

- ***onMessage*** : Envoyer / Recevoir des messages dans un scénario normal.
- ***onJoinChannel*** : Lorsqu'un nouveau membre rejoint un groupe, un message provenant du serveur sera envoyé à tous les membres, vous devez afficher ce message également en temps réel.
- ***onLeaveChannel*** : Lorsqu'un membre quitte un groupe, un message mentionnant qu'il a quitté le groupe, en plus de son nom sera reçu et doit être affiché.
- ***onError*** : Lorsqu'un message n'a pas pu être transmis par le serveur, ou si l'application n'arrive pas à envoyer le message au serveur, l'auteur du message devra en être informé. (Erreur de connexion par exemple).

Hors mis le scénario des messages échangés entre utilisateurs de façon normale, les autres messages sont assez spéciaux, et doivent être mis en évidence avec un autre type de couleur. ***(Cette couleur est laissée à votre appréciation et sera évalué, tout comme les autres requis fonctionnels et non-fonctionnels).***

La boîte de saisie

Comme les boîtes de messagerie courantes, le curseur de la souris doit être placé dans la zone de saisie par défaut. Après avoir écrit un message, un utilisateur peut simplement taper sur la

touche « entrez » du clavier ou le bouton « Envoyer » pour partager son message avec les autres membres du groupe.

N. B. La boîte de saisie doit être vidée après l'envoi d'un message. **Le pouce à gauche** de la boîte ne sera pas implémenté dans ce travail.

Vous pouvez consulter les captures d'écran pour plus de détails sur d'autres exigences visuelles non mentionnées.

c) Section de la liste des groupes de discussion

L'entête de la vue contenant la liste des groupes disponibles
Cette section est composée du titre du panneau comme vous pouvez le constater dans la capture et de deux (2) icônes.

- **Icône d'ajout d'un nouveau groupe de discussion**

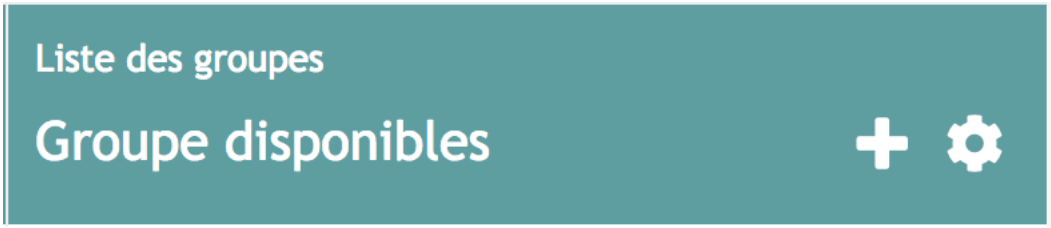
Cette icône affiche une boîte de dialogue simple qui permet à l'utilisateur de saisir le nom du groupe qu'il souhaite créer. Une fois le nom saisi et validé, un message avec un évènement de type « ***onCreateChannel*** » doit être crée et envoyer au serveur, ce dernier à son tour s'occupera de traiter la demande et de retourner les informations du nouveau groupe, le serveur retourne une erreur si la demande ne peut pas être traitée. Vous devez donc écouter le canal des erreurs « ***onError*** » également afin de détecter un tel évènement, et informer l'utilisateur dans une partie de la vue si nécessaire.

Vous ne devriez ajouter le nouveau groupe dans la liste que si le serveur vous retourne le groupe crée, et sans erreur via le même type d'évènement (***onCreateChannel***), Il se peut que le serveur vous demande juste de mettre à jour la liste des groupes. Cela permettra aux autres utilisateurs d'être informés aussi de l'événement.

- **Icône des réglages**

Cette fonctionnalité ne sera pas implémentée dans ce laboratoire, mais fera l'objet d'un bonus si vous y ajoutez des options comme :

  - Activer / Désactiver un court « son » **adapté** (bip par exemple) lors de la réception d'un message
  - Limiter le nombre de groupes à afficher
  - Autres améliorations considérables



Capture 1 : Entête de la vue des groupes de discussion

La liste des groupes de discussion

Lors de la connexion au serveur, le serveur identifie l'utilisateur et l'ajoute par défaut au groupe de discussion « Général », puis retourne la liste de tous les groupes disponibles par un évènement « ***updateChannelsList*** ». Vous devez écouter cet évènement du **web socket** pour mettre à jour la liste des groupes retournée par le serveur. Un ***observateur*** pourrait avoir la responsabilité de mettre à jour cette vue. **Attention !** Les groupes dont l'utilisateur courant y est associé sont affichés en premier dans la liste, et le groupe « *Général* » en tête de liste.

N. B. Le groupe de discussion « *Général* » est créé par défaut dans le serveur, et tous les utilisateurs y sont ajoutés par défaut lors de la connexion. Ajouter un indicateur pourrait être une bonne idée pour mettre en évidence cette particularité du groupe.

Chaque ligne contient à gauche deux (2) icônes que l'on alternera selon le principe suivant :

- **Icône plus (+)**
    - **Avant :** Afficher l'icône si l'utilisateur n'est pas dans le groupe, lui permet de rejoindre le groupe en cliquant sur cette icône. (NB : on détecte l'évènement « click » (DOM) sur l'icône, non pas sur la ligne).

    - **Après :** Envoyer un message avec l'événement « ***onJoinChannel*** » (WebSocket) au serveur avec l'identifiant du groupe, le serveur traitera la demande et retournera une liste à jour. Enfin, afficher l'icône avec le signe **moins (-)** pour l'action **ci-dessous** et mettre à jour la liste ordonnée.

- **Icône moins (-) :**
    - **Avant :** Afficher l'icône si l'utilisateur a été auparavant ajouté au groupe, cela lui permettra de quitter le groupe en cliquant sur cette icône. (NB : on détecte l'évènement « click » (DOM) sur l'icône, non pas sur la ligne).

    - **Après :** Envoyer un message avec l'évènement « ***onLeaveChannel*** » (WebSocket) au serveur avec l'identifiant du groupe, le serveur traitera la demande et retournera une liste à jour. Enfin, afficher l'icône avec le signe **plus (+)** pour l'action **ci-dessus** et mettre à jour la liste ordonnée.

En résumé, on alterne les actions précédentes, de même que les icônes de la liste.

Un dernier évènement dans cette section est la **sélection d'un groupe de discussion de la liste**. Deux (2) opérations doivent être exécutées :

- **Afficher le nom du groupe** sélectionné dans l'entête de la vue de clavardage comme mentionné précédemment.

- **Charger la liste des messages** de ce groupe dans la vue des messages (on n'appelle plus le serveur dans ce cas même si on aurait pu le faire). **NB :** Vous devez être à

mesure de retrouver les informations relatives au groupe courant pour avoir ses messages.

Voir la section annexe pour plus de détails.

Capture 2 : Liste des groupes de discussion

Bonus : Vous pouvez proposer une meilleure présentation de la liste des groupes de discussion.

3) Exigences détaillées et barème

Consultez la section annexe qui fournit des captures d'écrans de l'application. Servez-vous de cette référence, en plus des contraintes mentionnées pour implémenter l'application.

Le travail sera évalué selon les critères suivants :

- L'application est conforme aux exigences
- L'architecture est modulaire (**important**)
- L'application démarre sans problème avec les navigateurs Firefox et Google Chrome
- Le code est adéquatement commenté et suit les recommandations de JSDoc

4) Informations sur le serveur

URL de base : http://log2420-nginx.info.polymtl.ca

Socket serveur : URL de base + chatService?username={ nom }

a) Évènements supportés

- ***onMessage*** : Envoi / Recevoir d'un message
- ***onCreateChannel*** : Création d'un groupe de discussion
- ***onJoinChannel*** : Ajout d'un utilisateur dans un groupe de discussion
- ***onLeaveChannel*** : Retrait d'un membre d'un groupe de discussion
- ***updateChannelsList*** : Mise à jour de la liste des groupes
- ***onError*** : En cas d'erreur

b) Format d'un message

Voir la classe *« message.js »* (tous les messages ont le même format).

c) Fichiers et classes de l'application cliente

- message.js (*fourni*)
- channel.js (*fourni*)
- channelsObserver.js
- messagesObserver.js
- connectionHandler.js

- index.html
- main.js
- main.css

5) Modalité de la remise

Le travail se fait par des équipes de deux (2) étudiants, et le dépôt de votre code doit être fait sous la forme d'une archive .zip ou .tar nommée **TD3-Groupe X** (***Voir dans votre boite de remise sur Moodle***) contenant les fichiers nécessaires au bon fonctionnement de l'application.

6) Documentation utile

a) Icônes

- FontAwesome & Bootstrap glyphicon

b) Concepts utiles

- La programmation orientée objet en Javascript lien
- Le patron observateur lien
- Sérialisation/Désérialisation lien
- Manipulation des objets (Object) de type Map en Javascript lien
- Socket de communication : WebSocket lien
- JSDoc

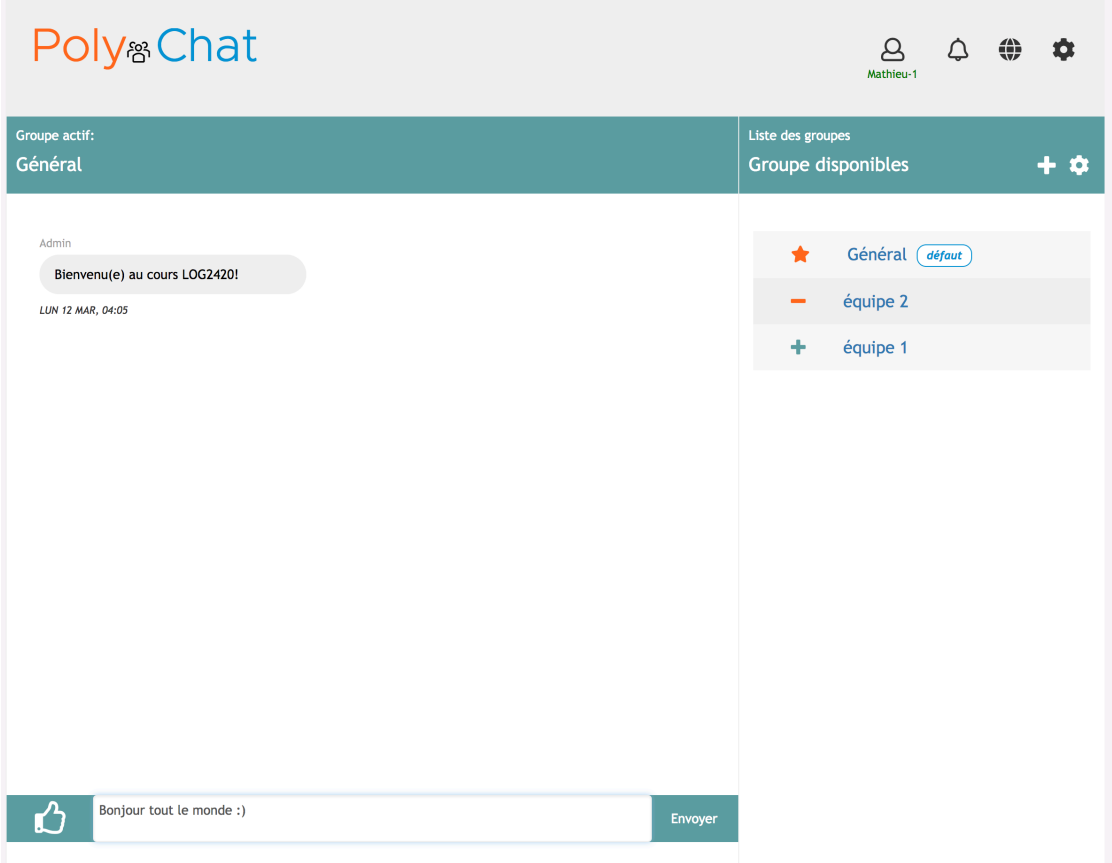
7) Questions pertinentes à se poser

1- Comment fonctionne les web sockets, comment les utiliser dans une application cliente (Les évènements standards disponibles)

2- Quels sont les acteurs et l'interaction entre ces derniers dans le concept MVC et le patron Observer/Observable.
3- Comment associer dynamiquement des évènements sur des nœuds du DOM en javascript.
4- Manipulation des Object Map en Javascript
5- Qu'est-ce qu'une sérialisation vs désérialisation

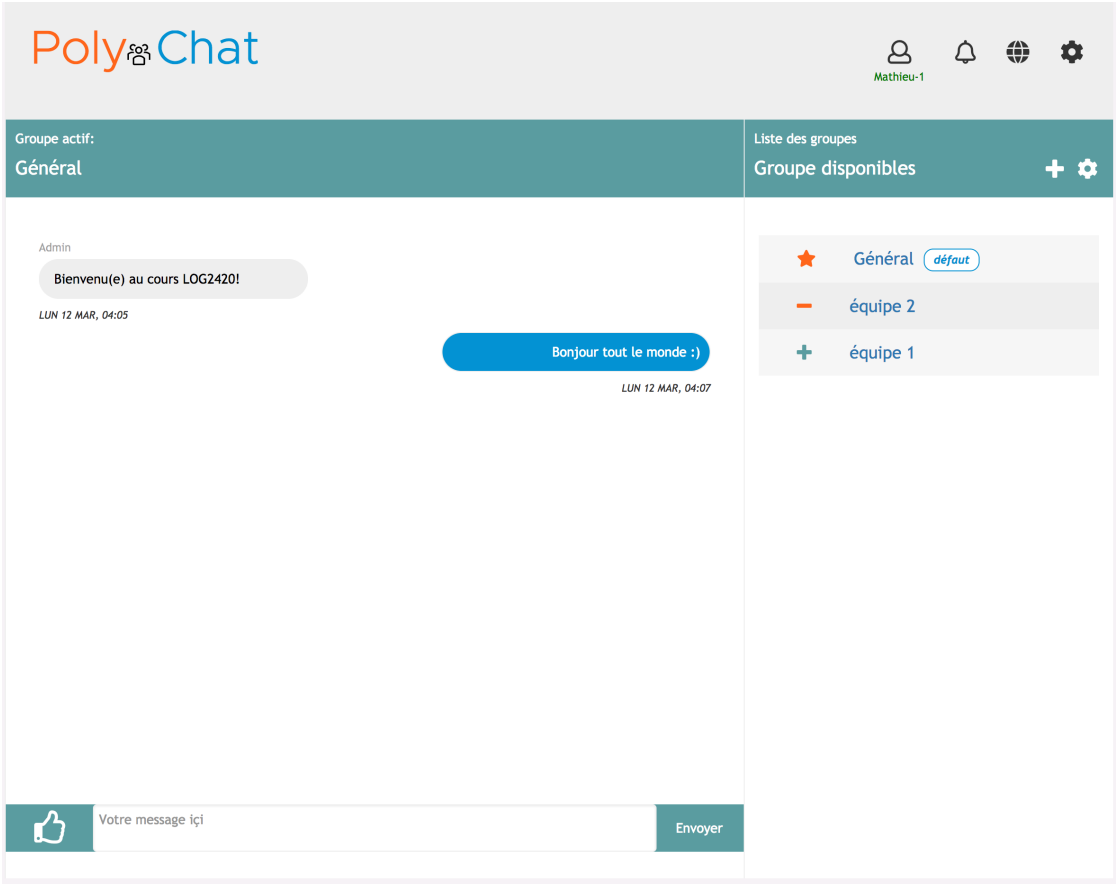
8) Annexes et quelques séquences

Annexe A : L'utilisateur « Mathieu-1 » est abonné au groupe « Général » par défaut en plus du groupe « équipe 2 », il peut quitter le groupe « équipe 2 », mais pas « Général », ou rejoindre le groupe « équipe 1 ». En plus, la zone de saisie a un contour en surbrillance pour mettre en évidence la boîte.



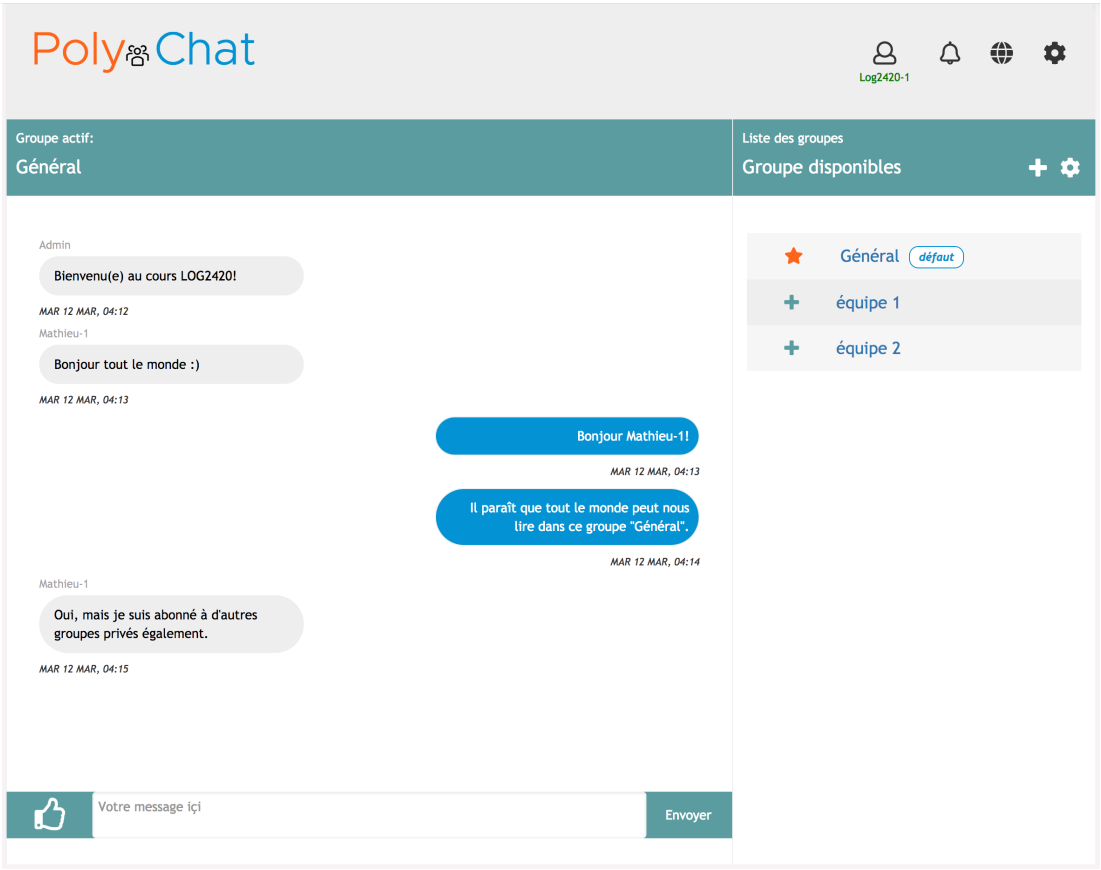
Annexe A : Page d'accueil avec le groupe « Général » actif par défaut.

Annexe B : *Un message envoyé par l'utilisateur « Mathieu-1 » dans le groupe « Général », tout le monde sans exception devra recevoir le message. Son nom n'est pas affiché dans son message même si le message revient du serveur****. Attention !*** *On ne range pas directement le message qu'il a écrit dans la liste, on range uniquement des messages provenant du serveur.*



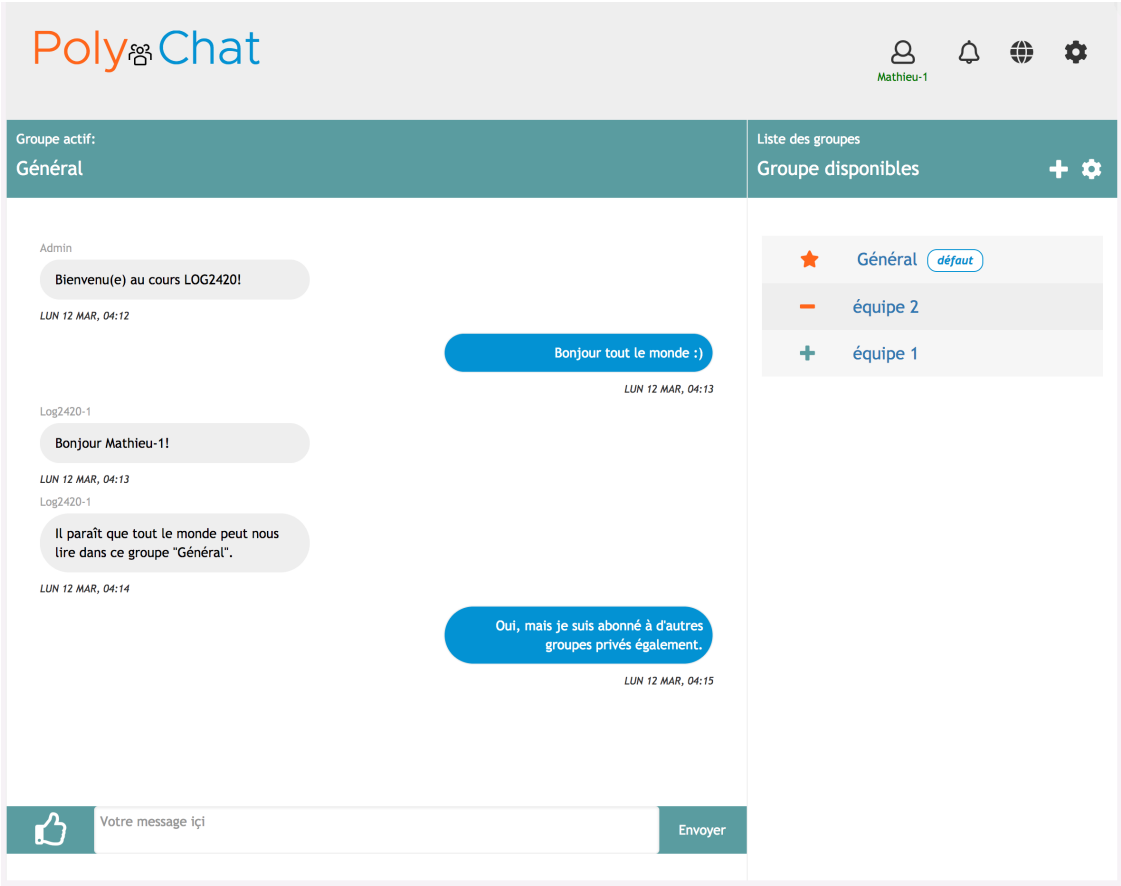
Annexe B : Un message envoyé par l'utilisateur « Mathieu-1 » dans le groupe « Général ».

Annexe C : L'utilisateur « Log2420-1 » échange des messages avec « Mathieu-1 » dans le groupe « Général ». Il peut joindre le groupe de discussion « équipe 1 et 2 », mais n'a pas une option pour quitter le groupe « Général ». C'est triste, mais on veut les obliger à être dans un groupe de façon par défaut.



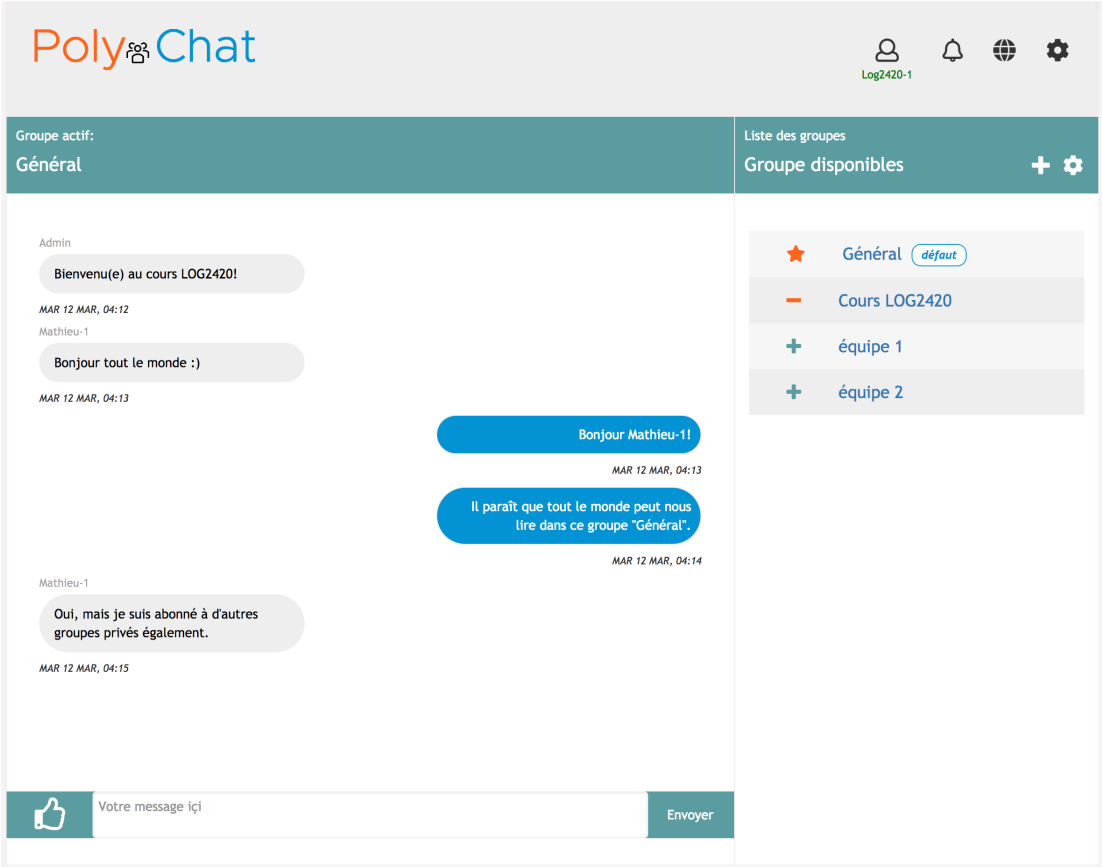
Annexe C : L'utilisateur « Log2420-1 » échange des messages avec « Mathieu-1 » dans le groupe « Général ».

Annexe D : *L'utilisateur « Mathieu-1 » a reçu deux (2) autres messages, et a envoyé un second message dans le groupe « Général ».*



Annexe D : L'utilisateur « Mathieu-1 » a reçu deux (2) autres messages de « log2420-1 » et envoie son second message dans le groupe « Général ».

***Annexe E** : L'utilisateur « Log2420-1 » a rejoint le groupe « Cours LOG2420 », mais le groupe « Général » est toujours actif. Il ne verra les messages du groupe « Cours LOG2420 » que lorsqu'il cliquera sur le nom de celui-ci.*



Annexe E : L'utilisateur « Log2420-1 » reçoit de nouveau un message de « Mathieu-1 ».