TP 5 – Application multiutilisateurs

But: créer une application Web simple (One Page App) avec gestion multiutilisateurs (un système de « chat » simple)

Démarche: Le backend est accessible via un Web Service(WS) en JSONP disponible en HTTP. Ce WS comporte quatre services, un fournissant tous les messages du chat pour un utilisateur, un autre permettant l'ajout d'un message au chat, un permettant de « connecter » un utilisateur au chat, et un dernier permettant de se déconnecter. Ces services sont respectivement disponibles via les url suivantes :

https://chabloz.eu/ws/chat/msg/get

https://chabloz.eu/ws/chat/msg/add?msg=XXX

https://chabloz.eu/ws/chat/user/login?user=XXX

https://chabloz.eu/ws/chat/user/logout

(Les XXX sont bien sûr à remplacer par des valeurs adéquates)

Comme le backend est déjà existant, il vous reste à développer le frontend. Celui-ci sera réalisé sous la forme d'une Web Application en utilisant les technologies HTML, CSS, JavaScript, et la méthodologie AJAX.

Etape 1 - Création d'une page de « Login »

Normalement, les utilisateurs du chat devraient se connecter avec un compte utilisateur avant de pouvoir accéder au chat. Pour des raisons de simplification de la gestion des comptes utilisateurs, l'application ne procédera pas à un véritable login, mais demandera un simple « nickname » à l'utilisateur. Réalisez une page HTML5 dont la balise <body> comprendra deux balises . Une balise ainsi qu'une balise . Puis réalisez le HTML et la CSS associée nécessaire à l'affichage d'un champ de saisie pour le « nickname » de l'utilisateur (20 caractères maximum) et d'un bouton de « login » (à l'intérieur de la balise). Libre à vous de choisir l'aspect graphique que vous voulez.

A l'aide de la CSS, faites que la ne soit pas affichée sur la page. Puis réalisez le JavaScript nécessaire à la gestion du click sur le bouton de « login ». Lors du click, vous devez vérifier que le champ ne contient que des caractères alphabétiques (entre a et z uniquement) et faire un appel au WS adéquat sur le serveur pour y connecter l'utilisateur. En cas d'erreur, vous devez afficher un message adéquat sur la page de login pour en informer l'utilisateur. Exemple de code pour tester qu'une chaine ne contient pas que des caractères entre 'a' et 'Z' :

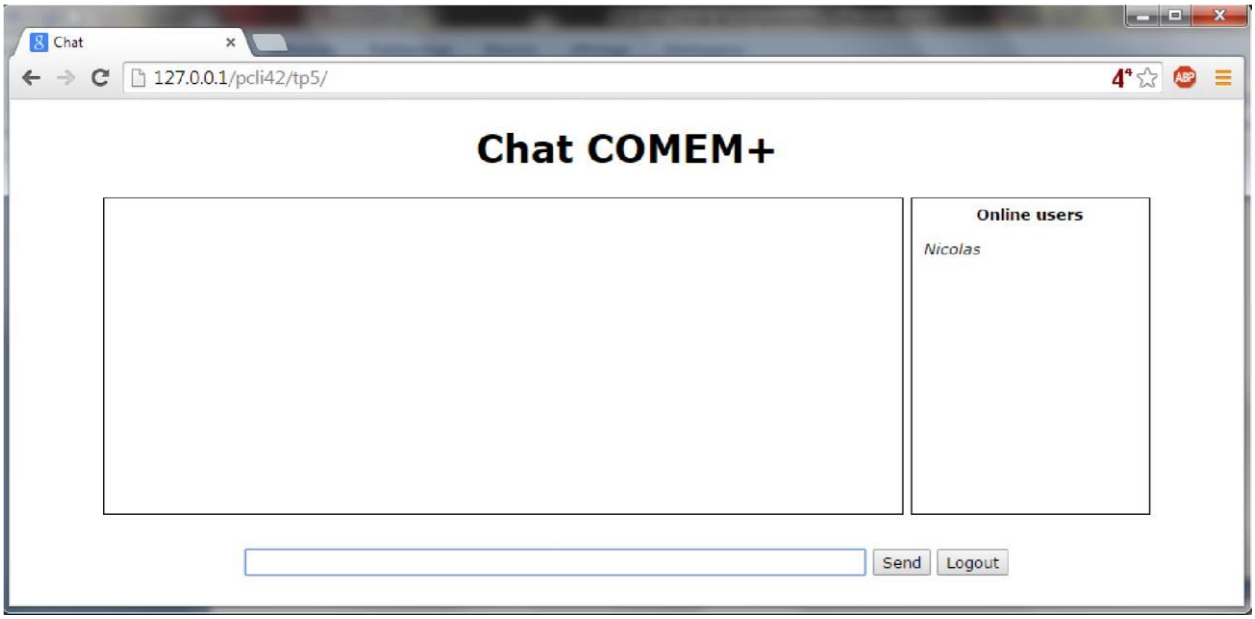
```
if (!user.match(/^[a-z]+$/i)) {
   // votre code ici
}
```

Si la phase de « login » c'est bien déroulée, cachez le et affichez le .

Etape 2 - Création d'une page de « chat »

La page de chat doit contenir cinq éléments. Un emplacement pour l'affichage des messages (un par exemple), un emplacement pour l'affichage de la liste des utilisateurs connectés au chat (un <div

id="online"> par exemple), un champ de saisie pour les messages du chat (un <input>), un bouton pour envoyer les messages, et finalement, un bouton de déconnexion. Voilà un exemple d'interface :



Il faut gérer les événements suivants :

- Lors d'un click sur le bouton pour envoyer un message, prendre la valeur du champ de saisie et l'envoyer au serveur via le bon WS. Puis videz le champ de saisie et y mettre le « focus ». Pour une ergonomie améliorée : faites que la touche « enter » dans le champ de saisie ait le même effet qu'un click sur le bouton d'envoi.
- Toutes les secondes (pour vos tests toutes les 5 secondes), exécutez un appel au bon WS pour recevoir les nouveaux messages du chat. Une fois les messages reçus, ajouter les à la fin de l'élément DOM adéquat. Pour exécuter une fonction toutes les secondes vous pouvez utiliser le code suivant :
  timerGetMsg = setInterval(callback, 1000);

  Lors de l'ajout des messages à l'élément DOM, il est possible que celui-ci soit déjà plein et qu'un ascenseur apparaisse. Afin d'éviter que l'utilisateur « scroll » vers le bas pour voir les nouveaux messages, voilà un code pour effectuer ce scroll automatique :
  $("#messages").scrollTop($("#messages").prop('scrollHeight')); // où #messages est à remplacer par le bon sélecteur CSS

- Toutes les 5 secondes, rafraichir la liste des utilisateurs connectés en utilisant les résultats de ce nouveau WS : https://chabloz.eu/ws/chat/user/online

- Finalement, lors d'un click sur le bouton de « logout », envoyer un message de déconnexion au serveur via le bon WS, et lors du retour de la réponse du serveur : cacher la page du chat, afficher la page de login et stopper les deux « timers » (les mises à jour des nouveaux messages et des utilisateurs connectés). Pour stopper un « timer », il faut utiliser la fonction suivante :
  clearInterval(timerGetMsg); // où timerGetMsg est le nom de la variable globale de référence au « timer »