


<p style="text-align: center; font-size: 2em; font-weight: bold;">NSI</p>	<p style="text-align: center; font-size: 2em; font-weight: bold;">INTERACTIONS ENTRE L'HOMME ET LA MACHINE SUR LE WEB</p>	
---	---	--

Activité machine : Un serveur de chat avec flask-socketio

Compétences visées :

- ✓ **Interaction avec l'utilisateur dans une page Web :**
 - Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page Web.
- ✓ **Interaction client-serveur. Requêtes HTTP, réponses du serveur :**
 - Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre.
- ✓ **Formulaire d'une page Web :**
 - Analyser le fonctionnement d'un formulaire simple.
 - Distinguer les transmissions de paramètres par les requêtes POST ou GET.

I- Tentative de chat avec des requêtes POST

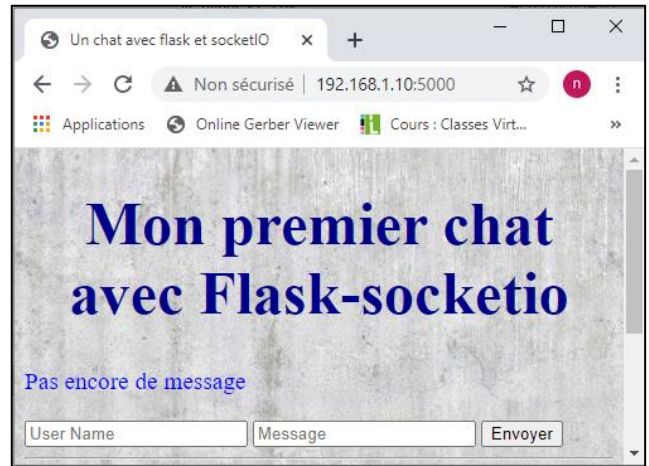
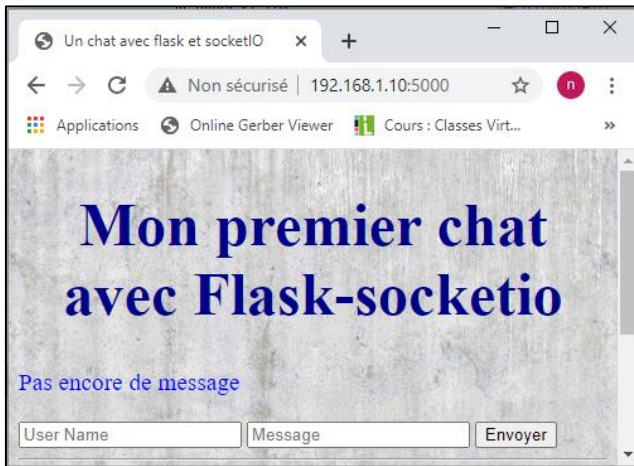
On vous fournit un premier exemple dans le dossier flaskSocketioChatV1_0, il contient principalement :

- un fichier python main.py contenant le programme d'un serveur utilisant le framework Flask ;
- un fichier index.html (dans le dossier templates)
- quelques fichiers supplémentaires (dans le dossier static) css, image.

1. Découverte

- 🖥 Exécuter le fichier main.py
- 🖥 Relever dans la console l'adresse IP et le port sur lequel le serveur attend une requête.
- 🖥 Ouvrir deux fenêtres de navigateur et taper l'adresse IP et le port dans les deux fenêtres.

Vous devriez obtenir quelques choses comme ça :



🖥️ Essayer de lancer une conversation entre les deux fenêtres.

Q1 Listez les dysfonctionnements constatés.

2. Plus en détails

🖥️ Ouvrir le fichier index.html

Q2 Repérez les lignes du fichier qui contiennent le formulaire.

Q3 Indiquez la méthode utilisée par ce formulaire.

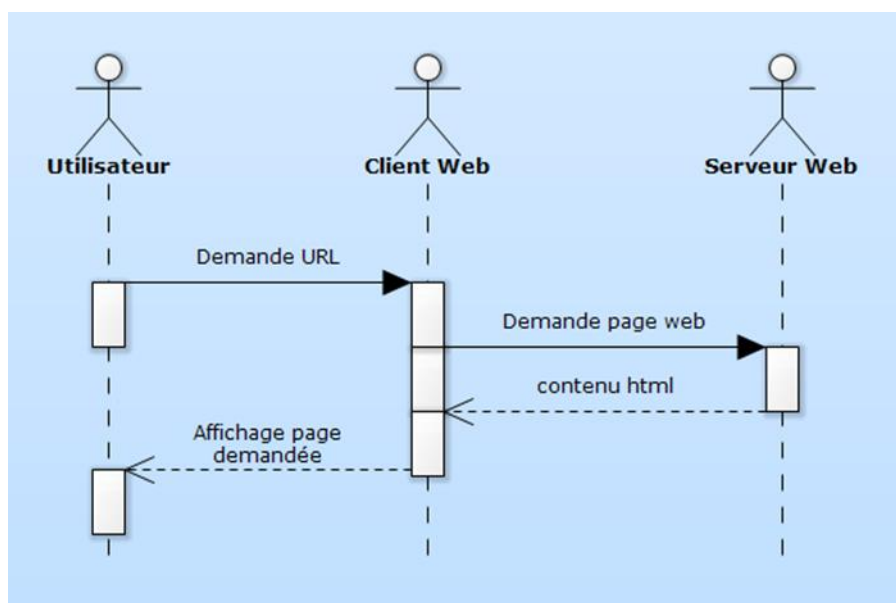
🖥️ En lisant le code python main.py

Q4 Repérez les lignes du fichier qui traitent la réponse au formulaire

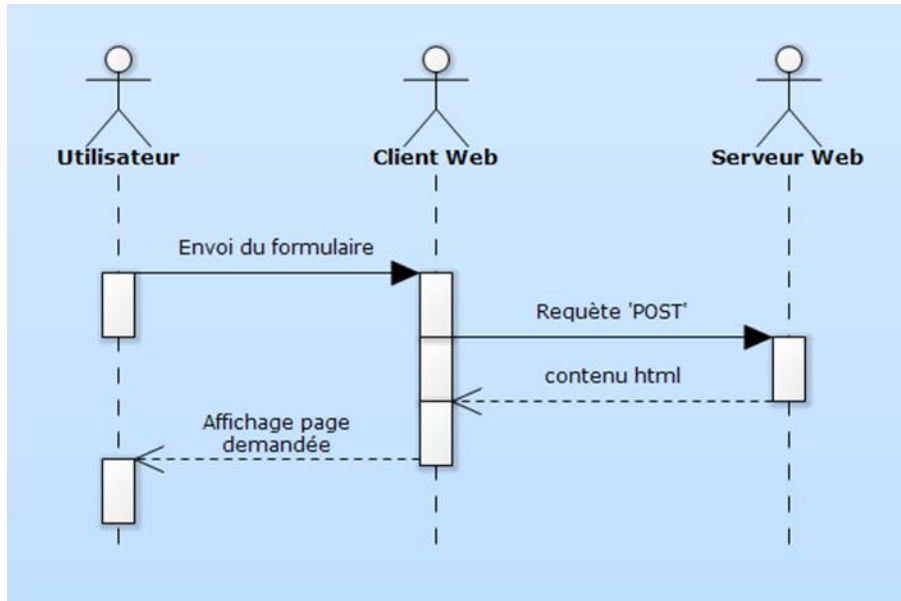
Q5 Expliquez ce que fait le serveur lorsqu'il reçoit un message.

3. Explication

Dans une communication client-serveur, c'est le client qui prend l'initiative et le serveur ne réponds que s'il reçoit une requête. Une demande de contenu par exemple :



Ou l'envoi d'un formulaire :



Ici, nous avons deux clients : vos deux fenêtres de navigateur. Le serveur renvoie la page mise à jour au client qui lui a envoyé le formulaire et rien qu'à lui. Notre chat ne peut donc pas fonctionner ainsi ...

II- Utilisation d'un socket-io

Un socket io est un tunnel de communication, mis en place à l'initiative du client, mais qui, une fois installé permet la communication dans les deux sens. On parle de communication asynchrone car le serveur n'a pas besoin d'attendre une requête du client pour lui envoyer quelque chose.

Mise en place du socket-io :

Le socket est déjà mis en place dans les fichiers fournis, nous allons voir ici comment cela est fait.

📁 Ouvrir le fichier index.html

📁 Repérer les lignes suivantes :

```
<!-- Le navigateur doit télécharger le script suivant pour faire fonctionner la socketio. -->
<!-- Il est disponible en ligne avec le lien suivant. -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.min.js"></script>
<!-- Charge le script js pour se connecter et dialoguer avec la socket -->
<script src = "/static/main.js" > </script>
```

Le navigateur charge sur internet, un fichier javascript qui contient le programme du socket-io et l'installe. Ensuite, il demande au serveur le fichier javascript main.js et l'exécute, c'est ce script qui va nous permettre d'envoyer et de recevoir des informations à travers le socket.

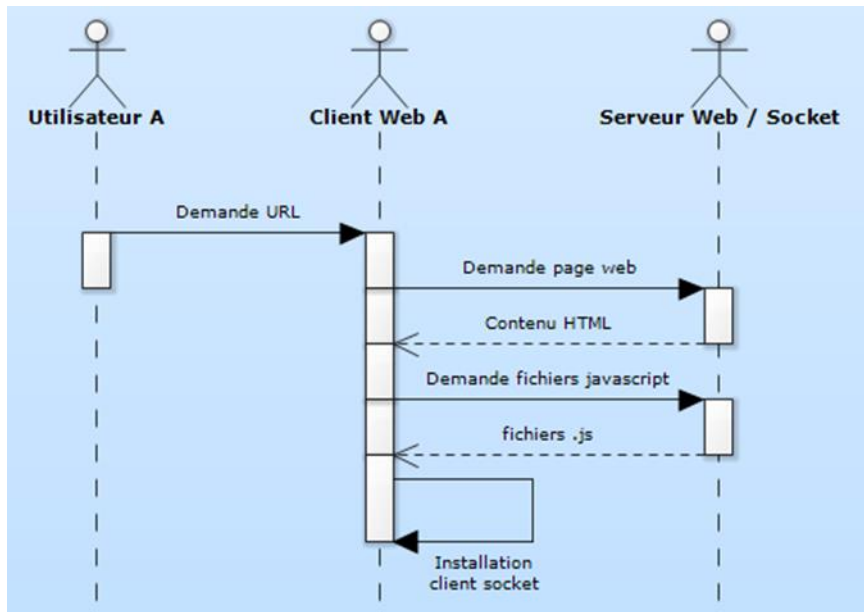
📁 Ouvrir main.js dans le dossier static.

main.js contient déjà une ligne :

```
//Se connecte au serveur socket io  
var socket = io.connect('http://' + document.domain + ':' + location.port);
```

Elle permet de se connecter au socket du serveur web qui a fourni la page.

Voilà un diagramme qui résume les opérations réalisées à l'ouverture de la page.



1. Envoi d'un message à travers le socket

📁 Dans le fichier index.html

Q6 Trouver l'identifiant du bouton coucou, présent sur la page.

📁 On souhaite maintenant envoyer un message à travers le socket. Voilà un petit exemple de fonction qui réalise cela :

```
function envoiCoucou() {  
    socket.emit( 'messTest', "Bonjour tout le monde !!")  
}
```

Q7 Dans cette fonction, indiquer l'identifiant du message et son contenu.

Q8 Proposer un petit programme javascript (une ligne) qui permet d'exécuter cette fonction à chaque appui sur le bouton poussoir 'coucou'.

📁 Démarrer le serveur et afficher la page web dans un navigateur. Vérifier que chaque appui sur le bp 'coucou' déclenche bien un affichage en console.

2. Réception d'un message par le serveur

Pour recevoir un message envoyé avec l'identifiant 'messTest' par le socket, il faut ajouter ce code coté serveur (main.py) :

```
@socketio.on('messTest')
def handle_my_custom_event(mess):
    print(f"Le message '{mess}' est arrivé !")
```

- 📖 Saisir ce code et vérifier le bon fonctionnement.

3. Envoi d'un message du serveur vers le client

- 📖 Pour envoyer un message depuis le serveur a travers le socket, voici la syntaxe python à utiliser :

```
socketio.emit('repTest', "Le serveur vous salut !")
```

- 📖 Pour que le programme javaScript du navigateur reçoive ce message, il faut utiliser cette syntaxe dans le fichier main.js :

```
function reception( msg ) {
    console.log('Un message du serveur : ' + msg)
}

socket.on( 'repTest', reception);
```

- 📖 Modifier les programmes main.js et main.py en conséquence.
- 📖 Tester et valider le résultat obtenu après un clic sur le bouton :

Dans la console du navigateur :

Dans le shell python :

```
Le message a été envoyé au serveur      main.js:9
Un message du serveur : Le serveur vous  main.js:16
salut !
```

```
Le message 'Bonjour tout le monde !!' est arrivé !
```

- 📖 Ouvrir une seconde page de navigateur, taper l'adresse du serveur dans la barre de tâche.
- 📖 Cliquer sur le bouton 'coucou' d'une des deux pages.

Q9 Indiquer ce que vous constatez dans la console de l'autre page du navigateur. Qui reçoit les données envoyées par le serveur à travers le socket.

4. Envoi d'un dictionnaire

Il est possible d'envoyer un dictionnaire pour regrouper plusieurs données en un seul envoi. On peut écrire par exemple, en javaScript :

```
socket.emit( 'messTest', {nom : 'Tom', mess : 'coucou !'})
```

Le serveur recevra un dictionnaire python contenant deux clés : 'nom' et 'mess'

Q10 Modifiez votre programme main.py pour afficher dans le shell le message reçu sous cette forme :

```
Un message de Tom est arrivé : 'coucou !'
```

III- Finalisation du serveur de chat :

Vous avez maintenant toutes les connaissances pour terminer le serveur de chat. Il reste à faire :

- Envoyer le contenu des deux champs du formulaire lors de l'appui sur un BP à travers le socket.
- Effacer le champ message du formulaire et lui donner le focus pour rendre la saisie du message suivant plus agréable et rapide.
- Ajouter une ligne de chat supplémentaire à chaque fois que le serveur envoie un message aux clients.

📁 Finaliser les programmes main.py et main.js

Vérifier le fonctionnement : En ouvrant deux fenêtres sur le même PC ou sur deux PC différents. Vous devriez obtenir ceci :



Aide : Pour ajouter une nouvelle ligne de chat sans effacer les précédentes. On peut utiliser les méthodes `appendChild` et `innerHTML` déjà utilisée dans les activités javascript. Cela peut ressembler à ça :

```
//pointe la division où saisir le message
let divChat = document.getElementById( ... );
//ajoute une nouvelle ligne
let liSupp = divChat.appendChild(document.createElement('p'));
//y place le message .innerHTML
liSupp.innerHTML = 'Le message à ajouter sur la nouvelle ligne';
```