Projet Gorille

Messagerie instantanée websocket - MVC

Html – CSS – Bootstrap – Javascript - PHP – MySQL - ws

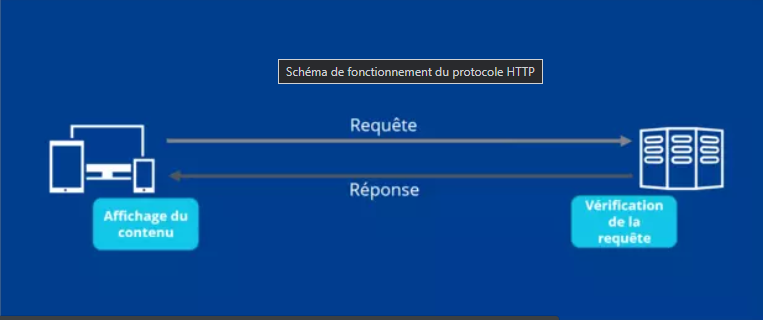
Le but de ce projet est de réaliser une messagerie instantanée intégrée à un site web dans lequel les utilisateurs pourront s’inscrire et dialoguer dans différentes « room » à la façon de Discord.

1. **La messagerie instantanée**

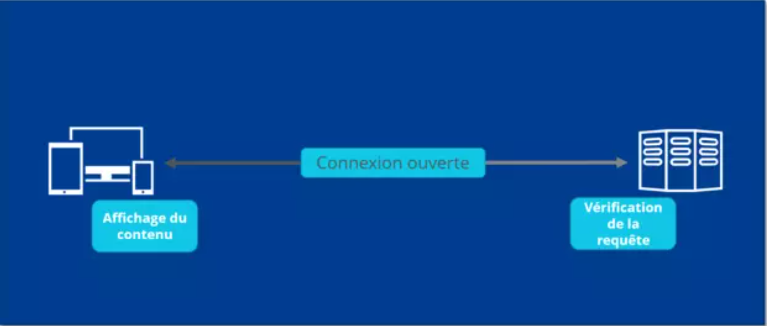
Le protocole WebSocket est un protocole réseau basé sur le protocole TCP. Ce protocole établit une connexion entre deux **points finaux de communication** qu’on appelle des **sockets**. Ainsi, une communication bidirectionnelle s’établit entre les données.

**HTTP versus WS**

En HTTP, pour consulter un site Web, le client doit d’abord envoyer une **requête au serveur**. Celui-ci peut ensuite répondre et transmettre le contenu désiré.



En WebSocket, une connexion active est ouverte après l’établissement d’une négociation (handshake) entre le client et le serveur. Le serveur peut alors envoyer de nouvelles informations au client sans requête préalable.



Le nouveau schéma URL WebSocket pour les pages Web consultées commencera par ws au lieu de http.

Le WebSocket est généralement utilisé lorsqu’une **connexion rapide** est nécessaire.

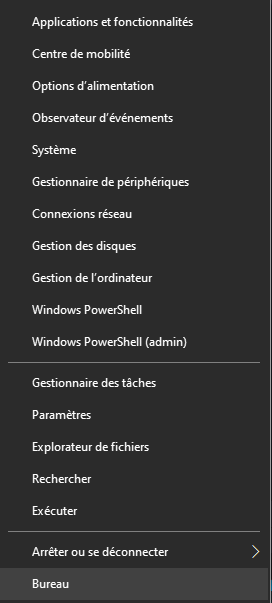
* Discussion instantanée
* Affichage de données en bourse
* Jeux en temps réel

Afin de se familiariser avec les websockets, nous allons installer la messagerie instantanée. La messagerie se compose de deux fichiers index.php (le client) et server.php (le serveur).

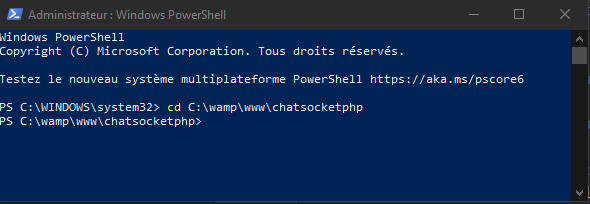
Le serveur écoute en permanence la connexion de nouveaux sockets, à l’adresse [ws://localhost:9000](http://localhost:9000/).

Lorsqu’un socket envoie un message, il est lu par le serveur et distribué à tous les sockets présents.

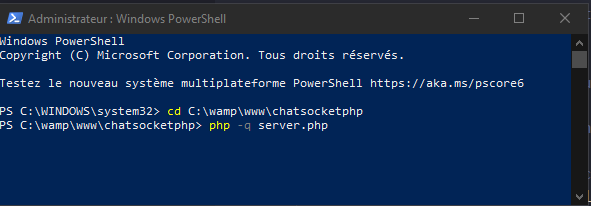
1. Dezippez l’archive ‘chatsocket.zip’ sous le répertoire ‘www’ de wampserver.
2. Ouvrir la console Windows PowerShell (admin) – Ca fonctionne aussi sous GitBash



1. Aller au répertoire ‘chatsocket’



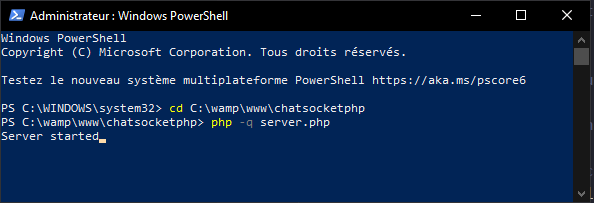
1. Saisir la commande ‘**php -q server.php’**



1. Puis <Entrée>

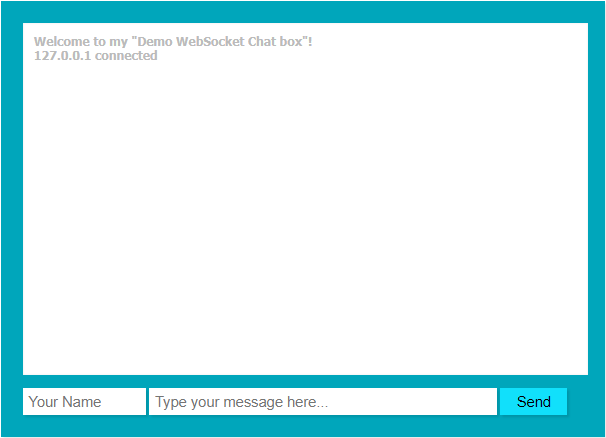
Le serveur websocket est alors démarré.

Dans la console tapez **CTRL + C** pour l’arrêter



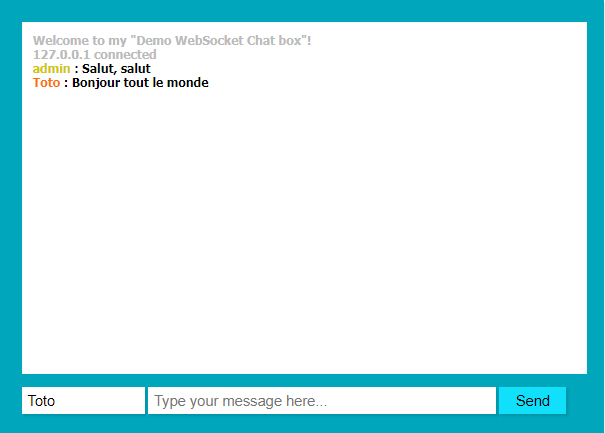
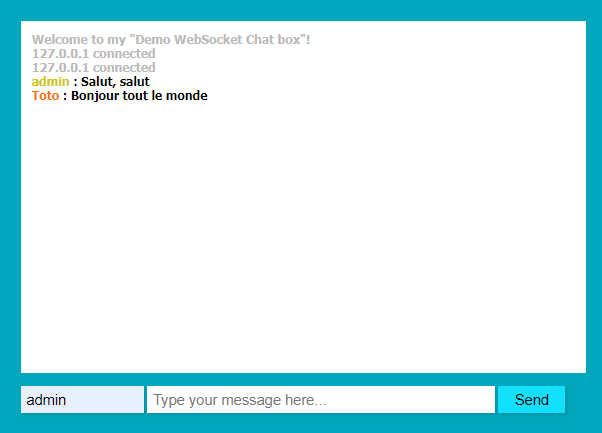
Ne pas fermer cette fenêtre, sinon le serveur sera stoppé, la messagerie ne fonctionnera pas.

1. Dans un navigateur, saisir l’url ‘**http://localhost/chatsocket/index.php**’. On obtient :



Dans un deuxième onglet, saisir la même url.

Saisir un nom et un message. Il apparaîtra simultanément dans les deux onglets.



1. **Le site internet**

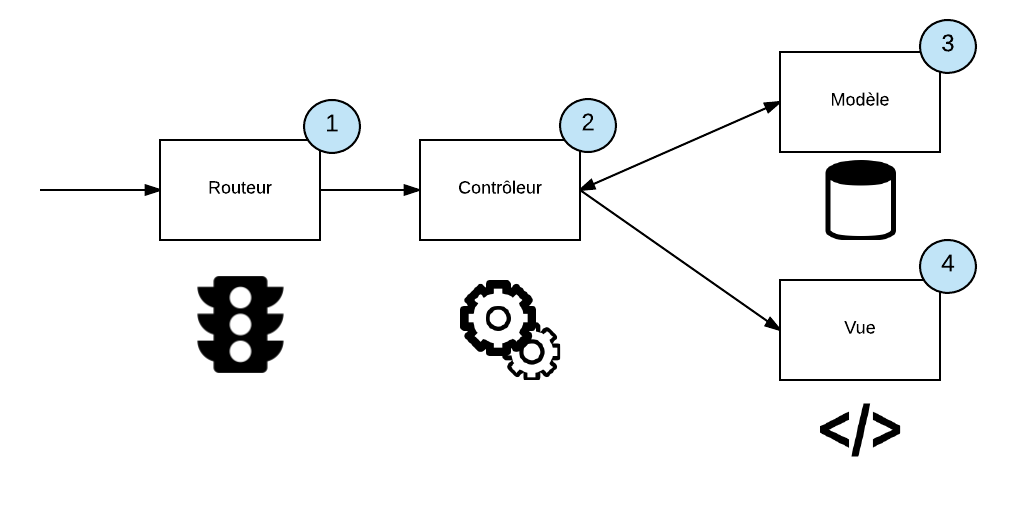
La structure du projet respecte les règles du Design Pattern « MVC » (Modèle – Vue – Contrôleur).

**Modèle** : cette partie gère les données de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.

**Vue** : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

**Contrôleur** : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du langage serveur. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

Le **routeur** est le point d’entrée d’un site. Nous allons le coder dans le fichier **index.php**. Il va se charger d'appeler le bon contrôleur.



Méthodologie

### Dézipper l’archive chatmvc.zip sous le répertoire www

### Séparer le code CSS du fichier index.php de la messagerie instantanée et le mettre dans un fichier style.css à part.

1. S’il n’existe pas, créer un fichier .htaccess à la racine du site pour configurer la redirection d’URL. Ce fichier contient ces lignes :

RewriteEngine on  
RewriteCond %{REQUEST\_FILENAME} !-f  
RewriteRule ^([a-zA-Z0-9\-\\_\/]\*)$ index.php?action=$1

Il s’agit d’un fichier de configuration du serveur Apache. Dorénavant, lorsqu’on saisit l’url :

<http://locahost/chatmvc/login/login>

le serveur Apache va comprendre :

<http://localhost/chatmvc/index.php?action=login/login>

la valeur du paramètre ‘action’ (login/login) est accessible dans PHP par $\_GET[‘action’]

1. Créer la base de données et les tables selon la description ci-dessous.
2. Définir les routes nécessaires et leurs contrôleurs associés.

Il y aura un contrôleur pour l’identification de l’utilisateur (loginController) et un contrôleur pour la gestion de la messagerie (chatController) :

|  |  |
| --- | --- |
| Route | Contrôleur |
| /login | LoginController.php |
| /chat | ChatController.php |

La base de données

La base « messagerie » contient 2 tables :

|  |  |
| --- | --- |
| users | contient les informations de login des utilisateurs |
| rooms | contient les noms des salons |

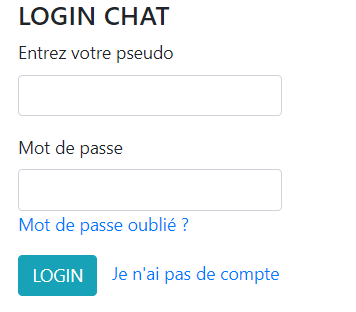
Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Layouts :

**La vue ‘login’ présente un formulaire comprenant :**

* Un champ Nom dans lequel l’utilisateur renseigne son pseudo
* Un champ Mot de passe
* Un bouton de soumission du formulaire
* Un lien ‘Créer un compte’
* Un lien ‘Mot de passe oublié’



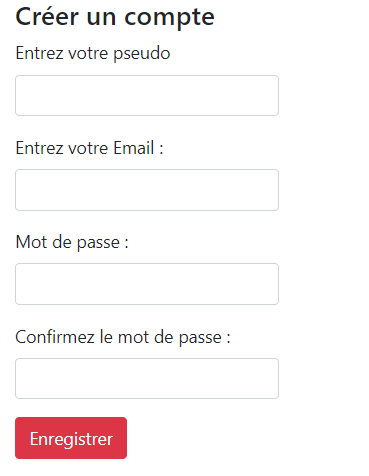
Si tout se passe bien lors de son identification, l’utilisateur est redirigé vers le chat : une page qui affiche la messagerie instantanée.

Une image contenant texte, capture d’écran, affichage, Rectangle

Le contenu généré par l’IA peut être incorrect.

**La vue ‘je n’ai pas de compte’ présente un formulaire comprenant :**

* Un champ où l’utilisateur saisit son pseudo,
* Un champ où l’utilisateur saisit son email.
* Un champ où l’utilisateur saisit un mot de passe,
* Un champ de vérification du mot de passe,
* Un bouton ‘Enregistrer’

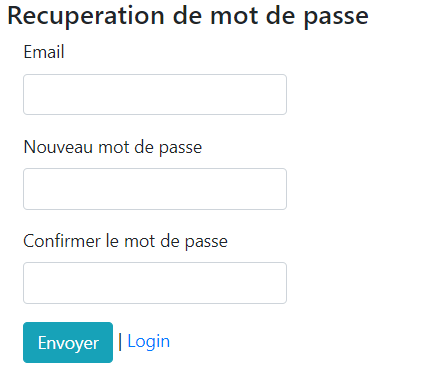


Une fonction Javascript « valid() » vérifie que les mots de passe saisis sont bien identiques.

Si tout se passe bien, l’utilisateur est redirigé vers le chat.

**La vue ‘Mot de passe oublié’ présente un formulaire comprenant :**

* Un champ email
* Un champ nouveau mot de passe
* Un champ de confirmation du mot de passe
* Un bouton d’envoi du formulaire



Une fonction javascript valid() vérifie que les mots de passe saisis sont bien identiques. S’ils sont identiques, le mot de passe de l’utilisateur est mis à jour en base de données.

Si tout se passe bien, l’utilisateur est redirigé vers la page login.

**La vue ‘chat’ présente**

* Le pseudo de l’utilisateur
* Le nom du salon (room) en cours
* Des liens vers les salons (colonne de gauche)
* Les messages du salon en cours (colonne de droite)
* Une zone de saisie des messages
* Un bouton d’envoi
* Un bouton rechercher qui pour l’instant est inactif



Méthodologie

Pour Intégrer la messagerie instantanée :

1. Créer un fichier ‘chat.js’ dans le répertoire js.
2. Déplacer la partie <script> du fichier ‘index.php’ de l’appli messagerie instantanée de la partie précédente dans le fichier ‘chat.js’.
3. Le fichier chat.js est placé dans le répertoire public/js

Le serveur de messagerie ‘server.php’ sera utilisé sans modification pour l’instant.

Le contrôleur LoginController.php inclue les méthodes :

|  |  |
| --- | --- |
| **Méthodes** | **Route** |
| loginIndex | login/loginIndex |
| signup | login/signup |
| forgotpassword | login/forgotpassword |

Il inclue également les méthodes « loadModel », qui permet de charger le modèle associé (LoginModel.php) et « render » qui affichera la vue correspondante.

Il est associé au modèle LoginModel.php

|  |  |
| --- | --- |
| **Méthodes** |  |
| \_\_construct | connexion à la base |
| createUser | création d’un nouvel utilisateur |
| existUser | Vérifie qu’un utilisateur existe en base |
| retrievePassword | mise à jour du mdp utilisateur |

Il est associé aux vues :

Views/login/LoginView.php  
Views/login/SignupView.php  
Views/login/ForgotPassword.php

On affichera les vues au moyen d’une méthode « render » qui prendra en paramètre le nom de la vue à afficher et les données à afficher sous la forma d’un tableau.

public function render(string $fichier, array $data = []): void

Cette méthode « render » fera appel à require\_once(‘nom\_de\_la\_vue.php’)

Le contrôleur ChatController.php inclue les méthodes :

|  |  |
| --- | --- |
| Méthodes | Route |
| chatIndex | chat/chatIndex |
| search | chat/search |
|  |  |

Il est associé au modèle ChatModel.php

|  |  |
| --- | --- |
| **Méthodes** |  |
| \_\_construct | connexion à la base |
| insertMessage | Insertion en BD d’un message utilisateur (voir paragraphe C) |
| getMessages | Récupère en BD les messages du salon (voir paragraphe C) |
| getRooms | Récupère en BD les noms des salon (voir paragraphe C) |

Il est associé aux vues :

/Views/chat/ChatView.php  
/Views/chat/SearchView.php

On affichera les vues au moyen d’une méthode « render » qui prendra en paramètre le nom de la vue à afficher et les données à afficher sous la forma d’un tableau.

public function render(string $fichier, array $data = []): void

Cette méthode « render » fera appel à require\_once(‘nom\_de\_la\_vue.php’)

1. **Persistance des données**

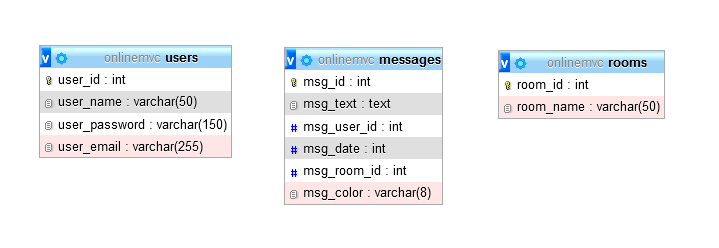
Pour l’instant, si on change de salon ou si on rafraîchit la page, les messages sont perdus. Nous allons donc stocker les messages en base de données.

La base de données

La base « messagerie » contient maintenant 3 tables :

|  |  |
| --- | --- |
| users | contient les informations de login des utilisateurs |
| messages | contient l’historique de tous les messages |
| rooms | contient le nom des salons |

Le Modèle Conceptuel de Données



Au clic sur le nom d’un salon, les 10 derniers messages de ce salon sont affichés.

**Modifier le client ‘chat.js’ de façon à ce que :**

* Au moment de l’envoi d’un message, celui-ci soit envoyé en base de données avec, l’id de l’auteur, la date sous forme de timestamp, l’id du salon

Pour cela créer une fonction store\_message dans chat.js afin de récupérer les paramètres du message (texte, nom, couleur, room). Un appel AJAX permettra de faire appel au chatController pour insérer les données en base. Vous pourrez utiliser l’API JS fetch ou bien utiliser jQuery installé avec Bootstrap :

Une image contenant texte, Police, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.

**Modifier le serveur ‘server.php’ de façon à ce que**

* Les messages émis depuis d’autres salons n’apparaissent pas dans le salon courant.



1. **Gestion de la couleur**

Une couleur est affectée au hasard à chaque utilisateur au moment de la connexion parmi une liste de couleurs prédéfinies à l’application et reste la même pendant toute la durée de la session.

**Modifier le loginController.php pour ajouter cette fonctionnalité.**

* La couleur est insérée en base de données pour chaque message de façon à ce que l’affichage de l’historique soit affiché avec les couleurs d’origine.

1. **Gestion des dates**

La date de soumission du message est insérée sous forme de timestamp (int) en base de données et affichée dans l’historique et dans l’écran de la messagerie instantanée.

1. **Rechercher dans les messages**

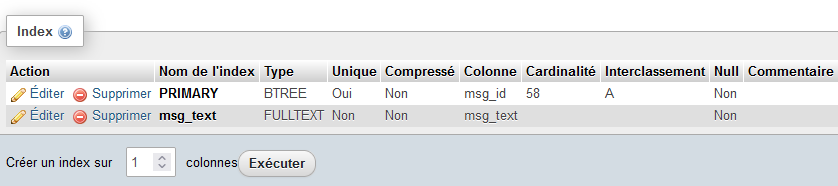
La page de la messagerie présente aussi un bouton ‘Recherche’ qui redirige l’utilisateur vers une page affichant le résultat de recherche.

**La vue ‘search’ présente**

* Le pseudo de l’utilisateur
* Une barre de recherche dans laquelle l’utilisateur saisi un mot clé
* La liste de tous les messages contenant ce mot clé, affichée au clic sur le bouton ‘Envoyer’

Dans la fenêtre du résultat de recherche, chaque message est précédé du nom de son auteur, du salon dans lequel il a été émis et de sa date.

La recherche des messages se fait en méthode Fulltext dans MySQL. Elle s’effectue sur la table complète au moyen des fonctions SQL **MATCH()** et **AGAINST()**. Sous phpMyAdmin, modifier la table ‘messages’, de façon à créer un index fulltext sur la colonne ‘msg\_text’. [[Plus d’infos ici]](http://samisd2003.free.fr/WinLAMP/MYSQL/fulltext-search.html)



Un bouton ‘Retour’ en haut à droite de l’écran permet de revenir à la page de la messagerie

.

