Conception d’une application client-serveur

Exemple d’une application simple de chatte.

Commençons par le cahier des charges: Il comporte les interfaces en wireframe avec le descriptif des fonctionnalités.

1. Cahier des charges :

Application de chatte

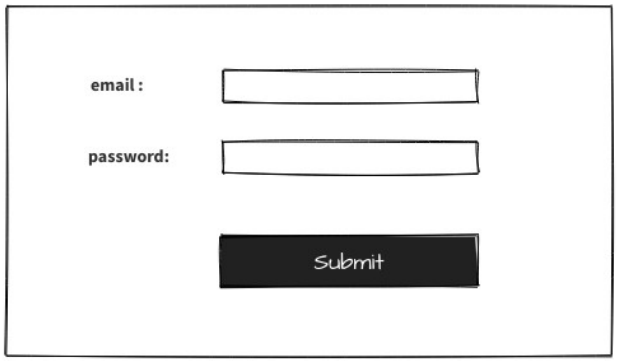
L’application comporte 3 interfaces. Elle est minimaliste, les utilisateurs sont ajoutés par un administrateur et les chattes se font qu’entre 2 utilisateurs, il n’y a pas de discussion de groupe.

1) **Connexion** : avec l’email et un mot de passe.

2) **Gestion des utilisateurs** : l’administrateur peut supprimer , ajouter et modifier des utilisateurs. Il n’y a pas de gestion de droit, le compte admin n’est pas en base de donnée, il n’en existe qu’un. Il n’est pas prévu non plus d’interface d’inscription. Les utilisateurs doivent être ajouter par cette interface.

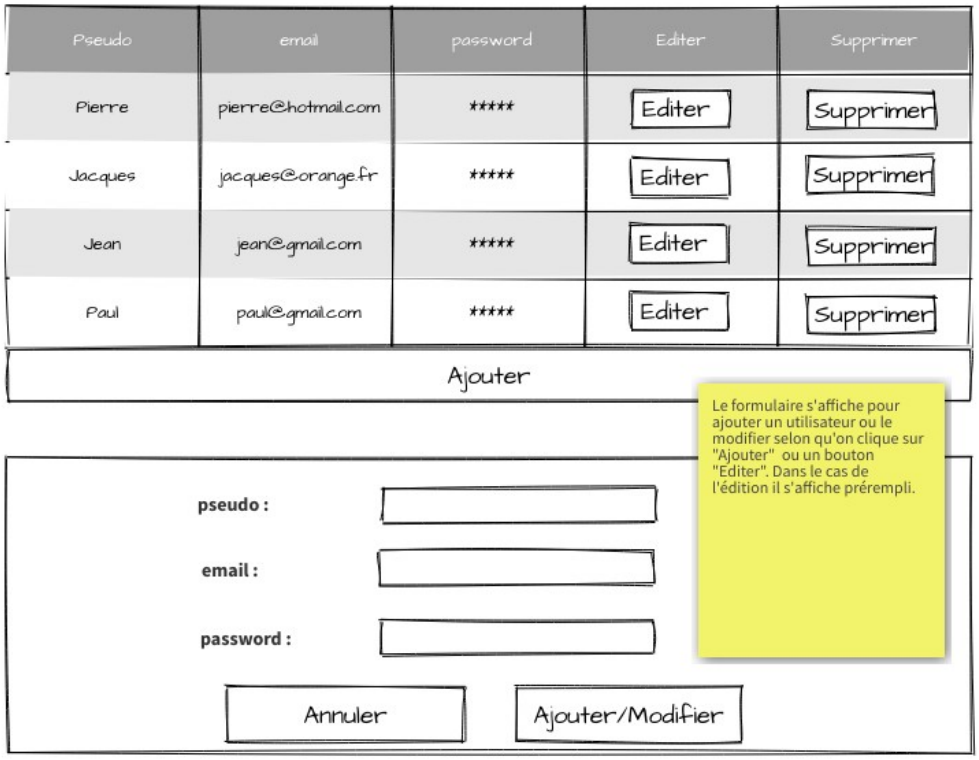
3) **fenêtre de chatte** : Elle affiche la liste des utilisateurs ainsi qu’un discussion avec l’un d’eux.

Connexion



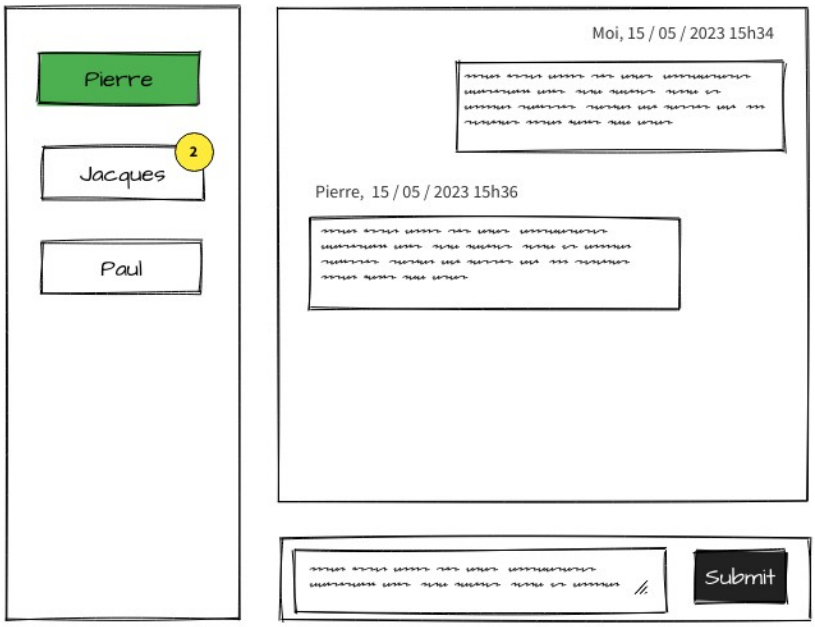
Fonctionnalité	description	action
Connexion	Permet de s’identifier sur l’application et d’être amener à l’interface «fenêtre de chatte»	Validation du formulaire

Gestion des utilisateurs



Fonctionnalité	Description	Action
Liste des utilisateurs	Affiche les pseudo, email et une colonne password remplie de «*» qui ne sert à rien d'autre que d'indiquer qu'à l'édition de l'utilisateur on pourra renseigner un nouveau mot de passe.	Par défaut sur la page
Suppression d'un utilisateur	La suppression d'un utilisateur le supprime de la liste, il ne sera plus possible de chatter avec lui et il ne pourra plus se connecter.	Clique sur le bouton «supprimer» de l'utilisateur.
Modification d'un utilisateur	On peut modifier les données d'un utilisateur à travers un formulaire accessible sur un clique du bouton «Editer» de l'utilisateur. Le formulaire apparaît alors prérempli par les données actuelles. Sa validation engendre les modifications saisies.	Clique sur le bouton «Editer» de l'utilisateur puis usage du formulaire.
Ajout d'un utilisateur	Le formulaire d'ajout apparaît vide quand on clique sur le bouton «ajouter» sous la liste. Il faut alors renseigner les 3 informations demandées : pseudo, email et password.	Clique sur le bouton «Ajouter» puis usage du formulaire.

fenêtre de chatte



Fonctionnalité	Description	Action
Liste des utilisateurs	Une liste des utilisateurs s'affiche dans le cadre de gauche. Ils sont affichés selon l'ordre anti-chronologique du dernier messages de leur discussion. Celui qui est vert correspond à l discussion affichée dans le cadre à droite. Une puce ronde sur le nom de l'utilisateur indique le nombre des messages qu'il a envoyés et qui sont non lus.	Par défaut sur la page
Fil de discussion	Dans le cadre haut droit s'affiche les messages échangés avec l'utilisateur choisi. La date et l'heure sont précisées juste au-dessus de chaque message.	Choix de fil de discussion en cliquant sur l'utilisateur voulu dans la liste des utlisateurs.
Envoi d'un message	Le cadre en bas à droite est un formulaire qui envoie un message à l'utilisateur choisi (donc en vert dans la liste des utilisateurs et dont le fil de discussion est affiché) .	Saisie d'un texte et clic sur le bouton submit.

2. Règles de conception

Nous allons imposer plusieurs règles déjà connues est utilisées dans le monde de la programmation informatique. Leur cumul n'est pas systématique. Il s'agit donc là d'une volonté particulière de vouloir cumuler les notions MVC (avec une centralisation vers un contrôleur) , Objet (POO) et RESTful (communication par message, sans mémoire des précédents états pour leur traitement).

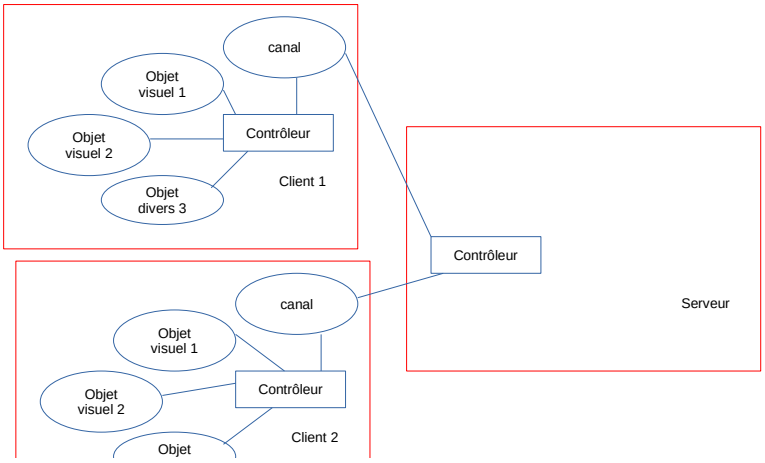
Liste des règles de développement que nous suivrons :

a. L'application sera conçu orientée objet.
b. Il y a une partie client qui s'occupe du visuel et une partie serveur qui s'occupe des données.
c. Les objets communiqueront par message. Ils émettent des messages et traite des messages.
d. Les messages seront tous centralisés et gérés par un contrôleur.
e. Les contrôleurs de la partie client et le contrôleur de la partie serveur converseront en se contentant de retransmettre les messages.
f. La partie client a plusieurs exécutions, qui correspondent aux différents clients connectés au serveur, elles ne pourront pas converser directement entre elles, elles devront passer par la partie serveur.

Ces règles vont induire plusieurs contraintes. L'application sera scindée en 2 parties «client et serveur» [règle b)]et chacune d'elle sera découpée en plusieurs objets [règle a)]. En ajoutant le fait que ces parties conversent via des contrôleurs [règle e)] on retrouve un peu le concept MVC qui s'étend aussi côté client.
Les objets ne pourront pas s'appeler directement [règle d)], cela signifie qu'ils ne connaissent pas les instances des autres objets.
Un contrôleur fait l'aiguillage des messages[règle d)], il reçoit un message émis par un objet et le transmet à un autre ou à un contrôleur [règle e)].
Chaque objet connaît donc l'instance du contrôleur afin de lui donner des messages [règle c) et d)], le contrôleur connaît chaque instance des objets et le nom de leur méthode qui réceptionne les messages [règle c) et d)]. chaque contrôleur connaît le canal de transmission des messages vers les autres contrôleurs [règle e)], pour le contrôleur de la partie client, il ne connaît qu'un seul canal de transmission vers un seul contrôleur [règle f)]. Enfin, il faut que le contrôleur sache pour chaque message possible quels sont les objets qui peuvent être concernés pour le traiter [règle d) e)].

3. Schéma de l’architecture du point de vue du contrôleur.

Du point de vue du réseau il y a plusieurs clients et un serveur. Chaque client est composés des même objets communiquant avec son contrôleur. Le serveur est aussi composé de plusieurs objets communiquant avec leur contrôleur. Le contrôleur du serveur est relié aux autres contrôleur par un canal de transmission. L’architecture est la suivante :



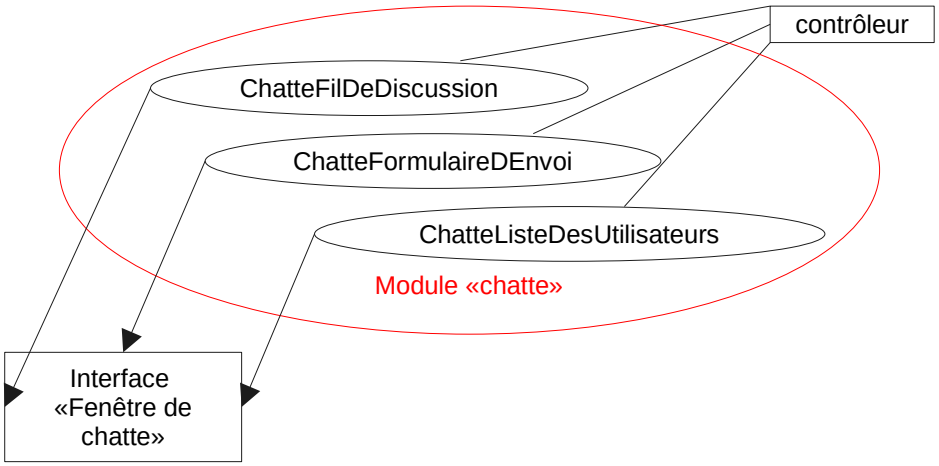
4. Découpage de la partie client en module, et des modules en objets visuels.

Nous séparerons la partie client en modules. Un module est un ensemble d’objets visuels qui s’occupent de données de même nature. Ici, Nous avons deux modules : «utilisateurs» et «chatte». Le module «utilisateur» comprends des objets visuels qui sont présents dans les interfaces «connexion» et «gestion des utilisateurs», le module «chatte» comprend des objets visuels qui sont tous regroupés dans l’interface «fenêtre de chatte».
Les interfaces telles qu’elles sont visibles dans le cahiers des charge ne sont donc qu’un choix de regroupement d’objets visuels. Une fois les modules bien définis avec leurs différents objets visuels nous pourrons construire ces interfaces en agençant ces objets. Nous pourrons même modifier l’agencement pour construire d’autres interfaces si des modifications de design sont demandées.
Par convention : Les objets qui communiquent avec le contrôleur auront leur nom qui commence par le nom du module.
Par convention : Les noms seront très explicites et nous utiliserons la nomenclature habituelle qui consiste à coller les mots en mettant des majuscules à leur première lettre. (par exemple, UtilisateurFormulaireDeConnexion)

Ainsi nous pouvons distinguer deux ou trois objets visuels dans le module «chatte». L’objet «ChatteListeDesUtilisateurs» et l’objet «ChatteConversation» qui pourrait être séparé en deux objets : «ChatteFilDeDiscussion» et «ChatteFormulaireDEnvoi». Nous choisirons la version 3 objets visuels pour le module «chatte», ainsi :

Nous aurons les objets suivants : «ChatteListeDesUtilisateurs», «ChatteFilDeDiscussion» et «ChatteFormulaireDEnvoi».
Il faudra faire de même avec le module «utilisateur».

Pour le module «chatte» nous avons donc la structure suivante qui vient se connecter au contrôleur :



Les lignes qui connectent les objets au contrôleur représentent les échanges de messages. Les flèches qui les relient à l'interface signifient que l'interface utilise ces objets visuels pour son affichage.

5. Spécification de l'objet «ChatteFilDeDiscussion»

ChatteFilDeDiscussion		
Liste des propriétés		
support	Élément HTML dans lequel sera fait l'affichage	
familleObjet	Une chaine de caractères qui identifie une famille d'objet qui dériveraient d'une même classe	Cela est normalement une constante.
nomDInstance	Une chaîne de caractères qui permet à une instance de connaître le nom de sa variable	Renseigné à la création de l'instance.
contextVisuel	Un objet dont la structure correspond bien à ce qui doit être affiché.	
reactContextVisuel	L'objet reactif de contextVisuel nécessaire pour l'utilisation de Alpine.js	
verbose	Un booléen qui permet de	Par défaut à false. On change la

	débugger en signifiant si on veut que l'objet exécute des console.log	valeur directement dans la console lors du debuggage.
listeDesMessagesRecus	Un Array des messages que l'objet veut recevoir	
listeDesMessagesEmis	Un Array des messages que l'objet veut émettre	

listeDiscussions	Un Object dont les index sont les pseudos des utilisateurs (les correspondants de la discussion) les valeurs sont des tableaux d'object qui contiennent le pseudo de l'émetteur du message, son texte et sa date de réception par le serveur.	Exemple de valeur:
		{
		"pierre":
		[
		{
		"pseudo":"pierre",
		"texte":"hello",
		"date":"2023-05-25 12:25:03"
		},
		{
		"pseudo":"pierre",
		"texte":"coucou",
		"date":"2023-05-25 12:29:03"
		}
		],
		"alain":
		[
		{
		"pseudo":"alain",
		"texte":"Hola",
		"date":"2023-05-24 12:25:03"
		}
		]
		}

Liste des méthodes		
constructor(s,c,nom)	S:élément HTML qui servira à l'affichage à mémoriser dans this.support C : instance du contrôleur nom : nom de la variable de l'instance	
affiche()	Affiche du contenu dans this.support	
traitementMessage(mesg)	Mesg : Object dont l'index est le nom du message et la valeur le contenu du message	
data2ContextVisuel(donnees)	Donnees : Array dont la structure correspond à une valeur de l'Object listeDiscussion.	L'objectif de cette méthode est de modifier la variable reactContextVisuel en fonction des données passées en

		paramètre.
Liste des messages emis		
chatte_fil_de_discussion_demande_d_une_discussion		
Liste des messages reçus		
serveur_chatte_selection_d_un_utilisateur		
chatte_formulaire_d_envoi_nouveau_message		
serveur_chatte_nouveaux_messages		
serveur_chatte_donne_discussion		
serveur_chatte_valide_reception_message		

La structure des messages n'est pas précisée ici, seul le nom des messages apparaît. En effet les messages sont utilisés par plusieurs objets et les définir à plusieurs endroits est source d'erreur car il est difficile de vérifier que leur définition est la même dans tous les documents. Aussi il convient d'établir une liste des messages du module.

6. Liste des messages utilisés par le module «chatte»

À compléter en exercice.

Messages du module «chatte»			
Nom du message	Objets abonnés en émission	Objets abonnés en réception	structure
chatte_fil_de_discussion_demande_d_une_discussion	ChatteFilDeDiscussion	ServeurNodeJS	['pseudo']=le pseudo de l'interlocuteur ['date_debut']=date au format YY/MM/JJ hh:mm:ss à partir de laquelle on veut les messages ['date_fin']=date au format YY/MM/JJ hh:mm:ss jusqu'à laquelle on veut les messages
chatte_formulaire_d_envoi_nouveau_message	ChatteFormulaireDEnvoi	ChatteFilDeDiscussion ServeurNodeJS	['texte']= le contenu tapé dans le formulaire de chatte ['numero']= un nombre qui identifie ce post. Il sera nécessaire pour que le serveur puisse attester de sa réception au client.