UNIVERSITÉ PARIS 8

Développement web avancé

Chat Node.js

Suivi par Nicolas Prieur

Nom : TACHOUR
Prénom : Omar
Num d'étudiant : 16706265

Introduction :

Mon projet consiste à faire un chat Node.js tels que node.js est une plateforme construite sur le moteur JavaScript V8 de chrome ,dont l'avantage est de développer des applications en utilisant une approche non bloquante permettant d'effectuer des entrées/sorties de manière asynchrone.

Contenu du projet :

MVC:

J'ai utilisé le Model View Contrôler(MVC) pour architecturer mon projet, pour cela j'ai créé 3 dossiers:

models:

il comporte 3 fichiers (user.js , chat.js et room.js) qui contiennent du code permettant d'effectuer une connexion a une base de données mongoDB et mongoose pour faire un schéma global a mon application.
Chaque fichier permet d'effectuer une connexion a cette base de données. En tout j'ai 3 schémas :

1. userSchema : pour faire un schema à la liste d'utilisateurs inscrit sur l'application.

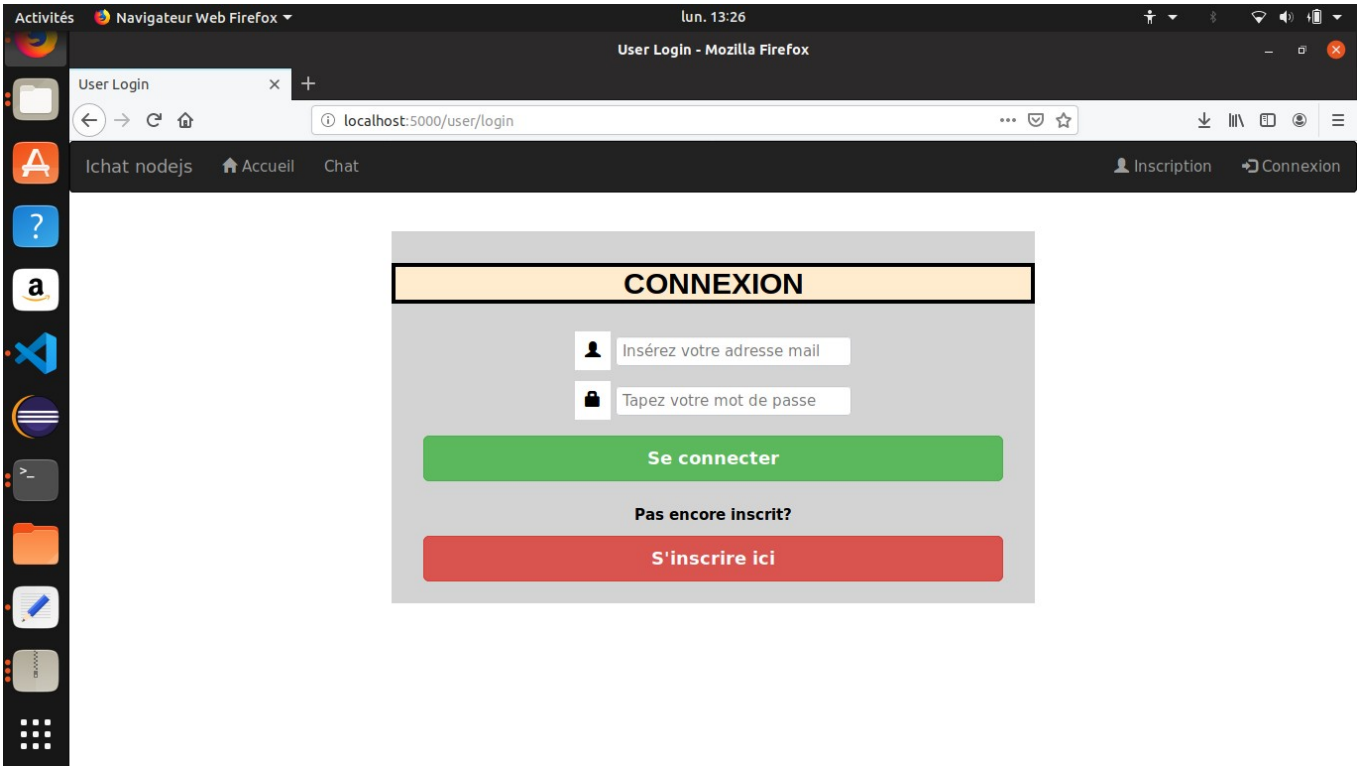
2. chatSchema : elle contient toutes les informations sur le chat(date de creation, destinataire, le contenu de message, la forme, et la room).

3 : roomSchema : pour faire un schema pour chaque room en cas de chat en groupe

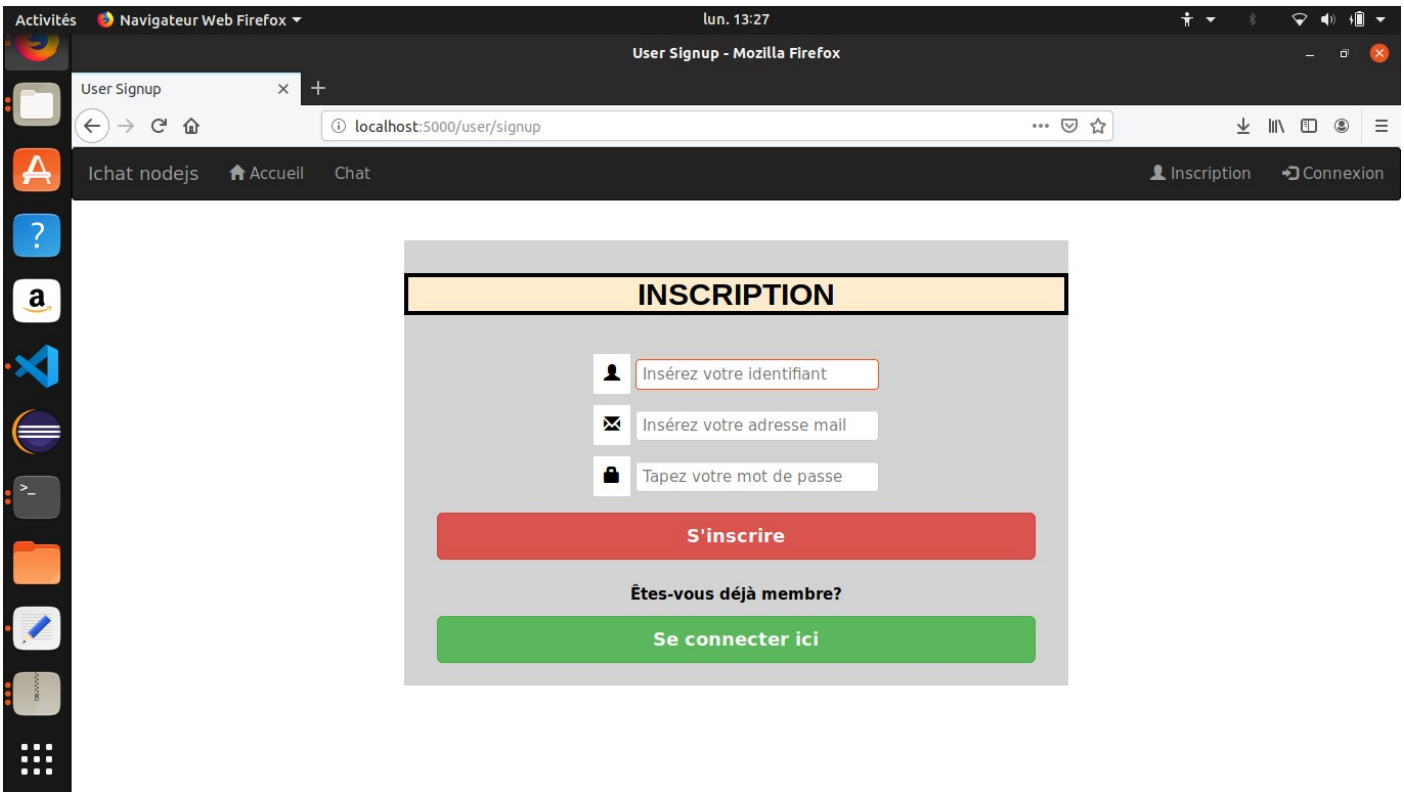
Views:

il contient 3 fichiers (login.ejs, signup.ejs, chat.ejs et message.ejs) dont l'extension est .ejs.
Sachant que ejs est un template proche à la syntaxe basique du html qui permet de générer des pages web dans un serveur.

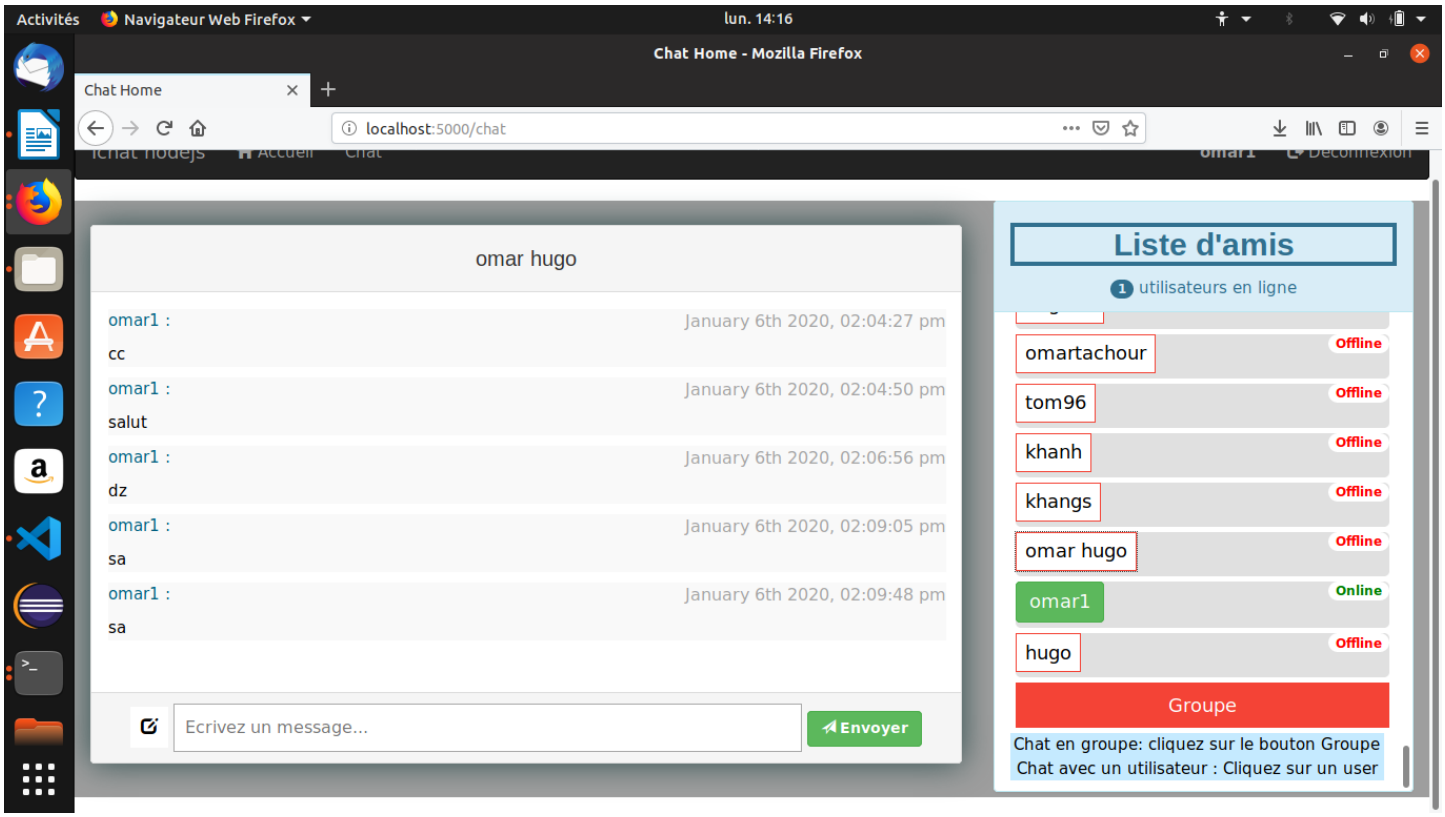
login.ejs : génère la page de connexion sur l'application



signup.ejs : génère la page de l'inscription sur l'application



chat.ejs : génère la page du chat qui est la page principale de l'application.



Controllers :

se compose de 4 fichiers qui permettent principalement de vérifier les chemins et fournir les accès au pages demandées par l'utilisateur.

Chat.js : contient l'itinéraire pour accéder à la page du chat.
home.js: contient le chemin pour accéder à la page d'accueil.
login.js : il permet de vérifier le chemin du connexion, il contient une fonction qui vérifie que (@mail et le mot de passe) sont correctes
signup.js : il permet de créer des nouveaux utilisateurs et vérifier les données insérées.

Intermdr: contient deux fichiers(authentifier.js et valider.js)

valider.js : il comporte une fonction qui permet de vérifier si l'utilisateur lors de l'inscription existe déjà, si oui il renvoie des messages d'erreur sinon il passe à la saisie du mot de passe.

authentifier.js : il contient une fonction qui sert a réorienter à la page de connexion/inscription ou à la page du chat.

Public : il contient 3 sous-dossiers

1:css : il comporte 3 fichiers .css (style, bootstrap, et icons.style) qui contiennent des informations indispensables pour styler l'interface graphique de l'application.

2 : js :

il contient deux fichiers (schriptForChat.js et scriptForSignup.js)

- schriptForChat.js : il gère l'affichage et tous les évènements qui se déroulent pendant le chat (afficher la liste d'utilisateurs, la surface du chat, écrire et envoyer les messages, cliquer sur les boutons ...)
- scriptForSignup.js :il contient des fonctions qui gère tout ce qui se passe pendant la création d'un compte utilisateur (vérification des @mail,utilisateur,mot de passe, affichage des erreurs …)

et 4 fichiers téléchargés pour faciliter le travail. Ce sont des modules nodejs, des bibliothèques et des framework sous licence libre pour faciliter les taches pendant la programmation.

- socket.io1.7.3.js : c un module de nodejs qui permet de créer des connections bi-directionnelles entre clients et serveur qui permettent une communication en temps réel sur un http.

- moment.min.js : c'est une bibliothèque permet de gérer les dates sous javascript.

- jquery-3.1.1.min.js , bootstrap.min.js : sont des frameworks qui permettent de faciliter des fonctionnalités communes et gagner du temps lors de développement .

3:fonts :

il contient des polices utilisées lors du développement de l'application.

Libssl :

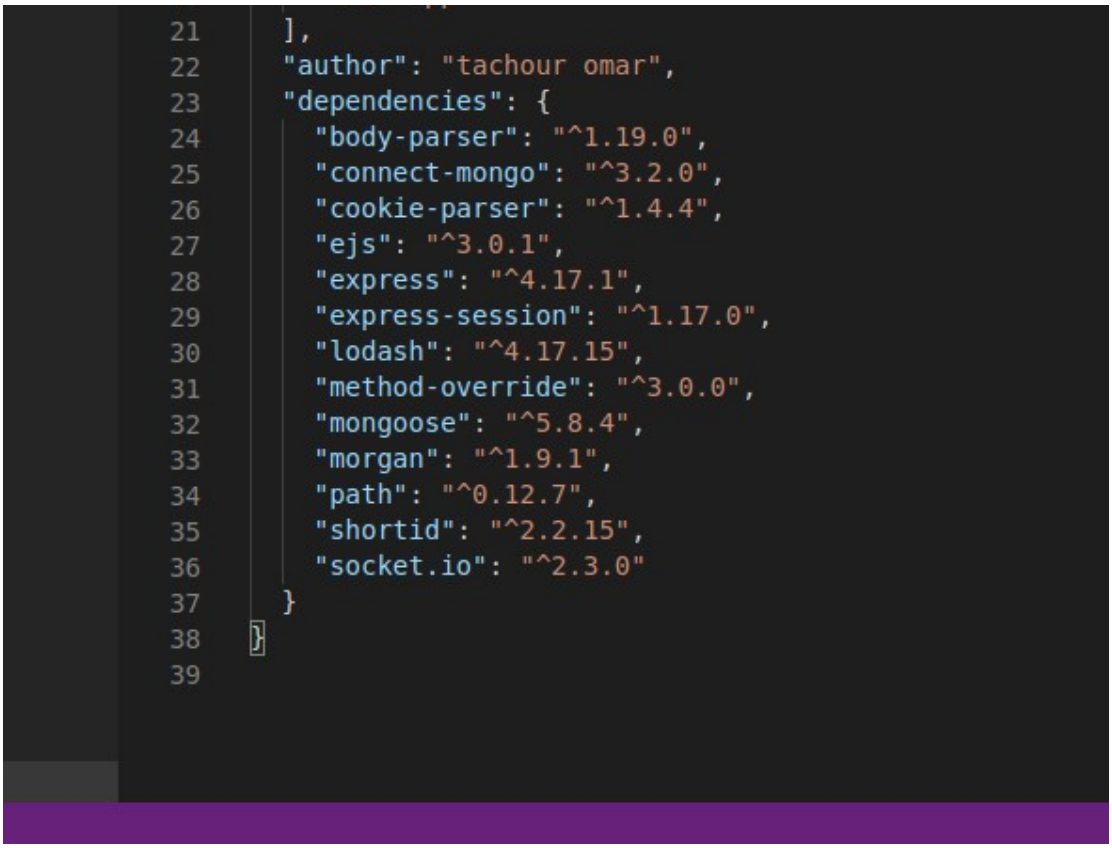
il comporte deux fichier .js :

chat.js : c'est un grand fichers qui contient plusieurs fonctions. Au début j'ai ajouté les model de database qui sont dans le MVC et utilisé les 3 schemas que qui sont dans le dossier models. Ensuite j'ai fait appel à la fonction responsable pour commencer le temps réel et on établie une connexion au socket.io, lister

tous les utilisateurs, réglage des room, lire la database,lire et enregistrer les chat dans la database,…)

et enfin le fichier crypto.js pour créer Hma

les dépendances :



```
  ],
  "author": "tachour omar",
  "dependencies": {
    "body-parser": "^1.19.0",
    "connect-mongo": "^3.2.0",
    "cookie-parser": "^1.4.4",
    "ejs": "^3.0.1",
    "express": "^4.17.1",
    "express-session": "^1.17.0",
    "lodash": "^4.17.15",
    "method-override": "^3.0.0",
    "mongoose": "^5.8.4",
    "morgan": "^1.9.1",
    "path": "^0.12.7",
    "shortid": "^2.2.15",
    "socket.io": "^2.3.0"
  }
}
```