Rapport du Projet de la Messagerie Instantanée

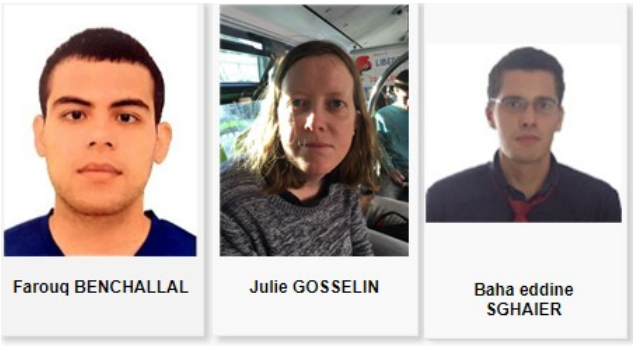
Année Universitaire 2018/2019

Réalisé par :

- GOSSELIN Julie
- BENCHALLAL Farouq
- SGHAIER Baha eddine

Encadré par :

- Mr Karim NOUIOUA
- Mr Bertrand ESTELLON



Introduction :

Dans nos jours, il est devenu presque indispensable de pouvoir communiquer de façon aisée avec des outils facile et accessible.

Parmi ces outils, l y a la messagerie qui fait en sorte de pouvoir échanger des textes, des photos, des documents et autres, en instantané ou en différé.

But du Projet : L'objectif du projet vise à mettre en pratique les connaissances acquises dans le cadre du module de développement WEB en Master 2 CCI.

Sujet : Il s'agit de concevoir une application de messagerie instantanée dans laquelle il va falloir développer un serveur et un client. Le rôle principal du serveur consiste à vérifier l'identité d'un client qui se connecte à l'aide d'un login et d'un mot de passe et de mettre les personnes du service de messagerie en relation.

Le client se charge de transmettre les informations nécessaires au serveur et les messages écrits destinés aux autres clients.

Organisation du travail

Le travail de conception de l'application a comporté trois étapes principales :

- L'élaboration du cahier des charges qui comporte les spécifications de l'application: description de chacun des onglets et les fonctionnalités qui leurs sont associées.
- La conception de la base de données SQLite sous-jacente à l'utilisation de l'application.
- Le développement des fonctionnalités.

Les deux premiers points font l'objet des parties 2 et 3 du mémoire.

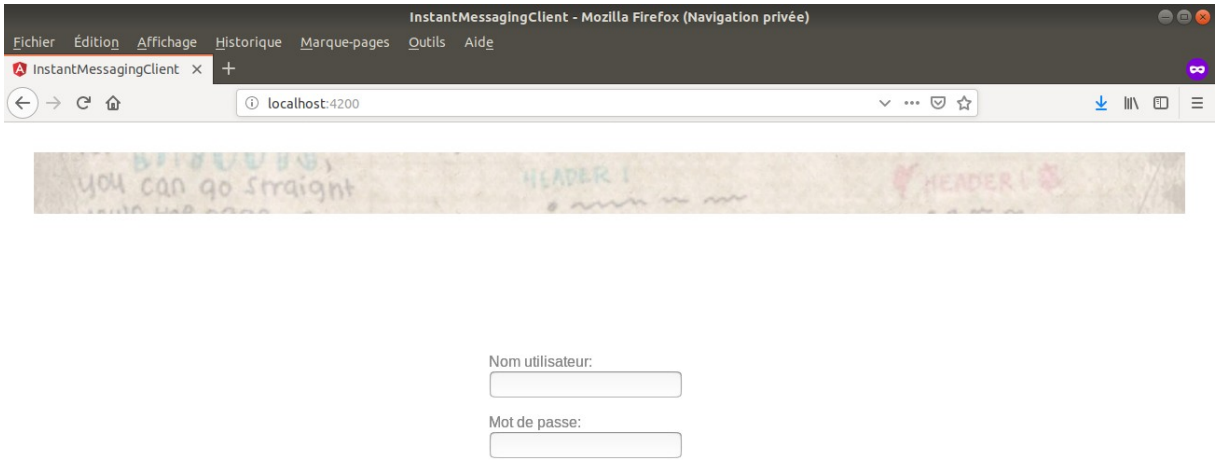
Pour le troisième, nous avons utilisé angular, nodejs, javascript…

Nous nous sommes ensuite répartis le travail, où chacun d'entre nous s'est occupé d'une partie dont les vues, la création de la base de données ainsi que les actions et les fonctions nécessaires à son interrogation et de sa gestion.

On arrive sur la messagerie via l'URL qui redirige automatiquement à la page d'Accueil.

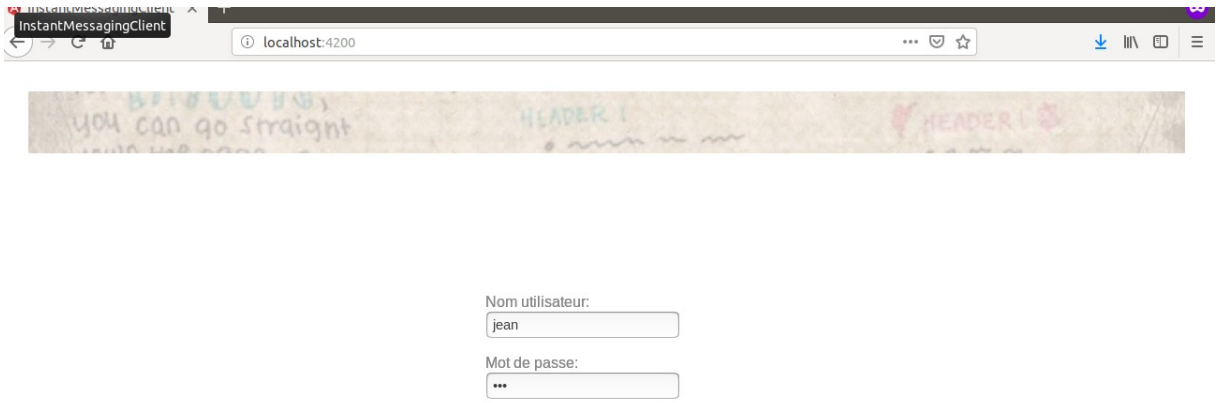
La page Accueil :

La page d'accueil est la page principale de la messagerie sur laquelle se trouve l'utilisateur.



Connexion :

Pour se connecter à la messagerie, il faut un mot de passe.



Si l’utilisateur a commis une faute dans les champs de l’identifiant et/ou mot de passe, rien ne se passe, et donc pas d’accès à la messagerie.

Après connexion, on arrive sur la page suivante :

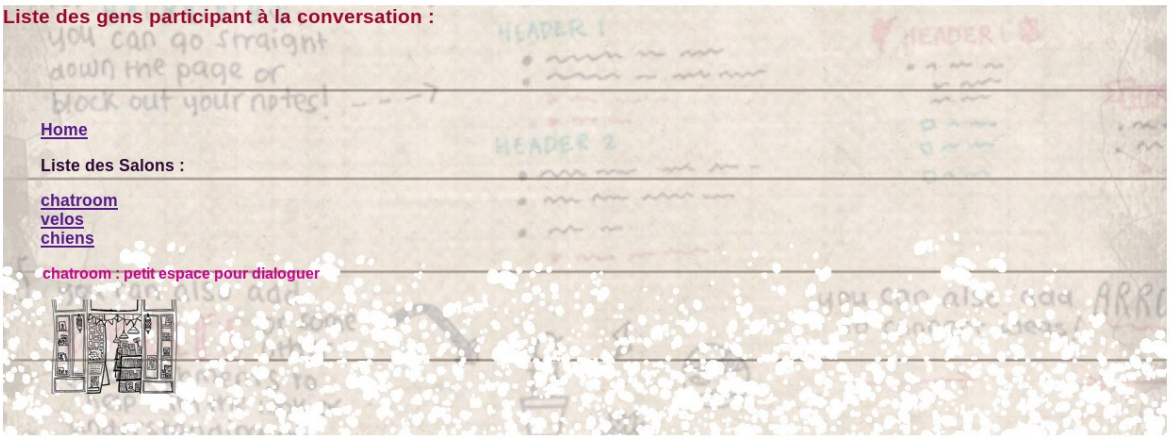
Salons de discussion :

Après la connexion, chaque utilisateur peut voir la liste des salons de discussion ainsi que la liste des utilisateurs inscrits.

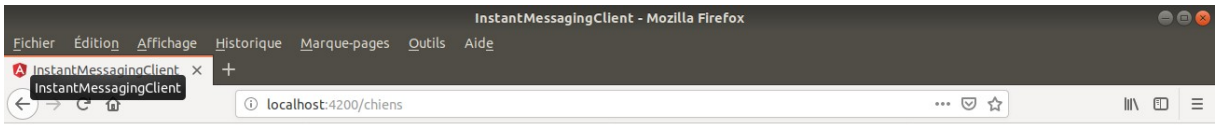
Liste des salons :

Une fois connecté, l’utilisateur se retrouve face à une liste de salons auxquels il peut avoir accès, ces salons sont de thématiques différentes (sujets de discussion). Il peut donc accéder à n’importe quel salon de la liste affichée. Dans notre messagerie, nous avons les salons chatroom, chiens et vélos qui sont les suivant :

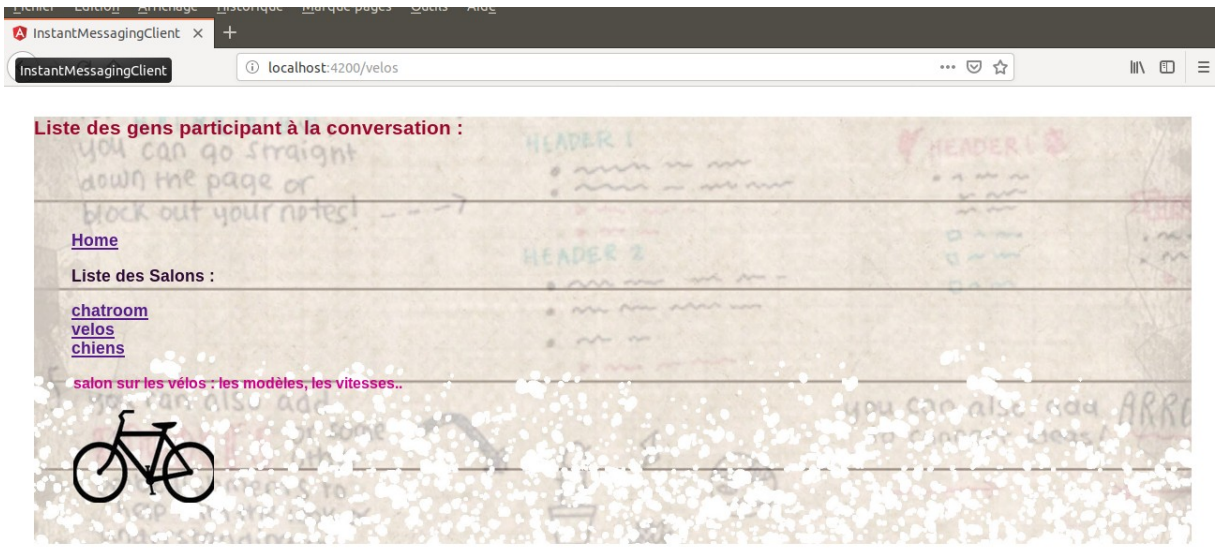
Salon chatroom :



Salon chiens :



Salon Vélos :



Conception de la base de données SQLite:

L'objectif ici est de réaliser une base de données pour le client permettant de stocker les informations relatives aux utilisateurs et leurs comptes.

La base de données SQLite fonctionne sans serveur et est facile à mettre en place. Du coup, c'est une base de données locale.

Élaboration du dictionnaire des données :

Le dictionnaire de données représente les différentes propriétés ainsi que leurs types et les contraintes sur chacune d'elles.
Il est représenté comme suit :

Propriétés	Type	Définition	Contraintes
id_user	INT	identifiant d'un user	NOT NULL
Nom_user	TEXT	Nom d'un user	NOT NULL
Tel_user	INT	téléphone d'un user	NOT NULL
Mail_user	TEXT	Mail d'un user	NOT NULL
id_salon	INT	identifiant d'un salon de discussion	NOT NULL
Nom_salon	TEXT	Nom d'un salon de discussion	NOT NULL
date_msg	NUMERIC	date d'un message	NOT NULL
content	TEXT	contenu d'un message	NOT NULL

Dictionnaire de Données

Les tables de la base de Données :

Les tables principales contenues dans la base de données seront les suivantes :
users (id_user, Nom_user, Tel_user, Mail_user)
salons (id_salon, id_user)
messages (date_msg, content)

avec :
id_user : Clé primaire de la tables users
(id_user, date_msg) : Couple formant la clé primaire de la tables messages
Id_salon : Clé primaire de la table salons

Définition des règles de gestion :

Règle users-users :

- un user peut discuter avec plusieurs users

Règle users-salons :

- un user peut créer plusieurs salons de discussion
- un salon peut être créé par un seul user
- un user peut être membre de plusieurs salons de discussion
- un salon de discussion peut contenir un ou plusieurs users

Règle users-messages :

- un user peut écrire plusieurs messages
- un message peut être écrit par un seul user

Règle messages-salon:

- un salon peut contenir plusieurs messages
- un message peut être contenu dans plusieurs salons

Les relations entres les différentes tables seront alors comme suit :

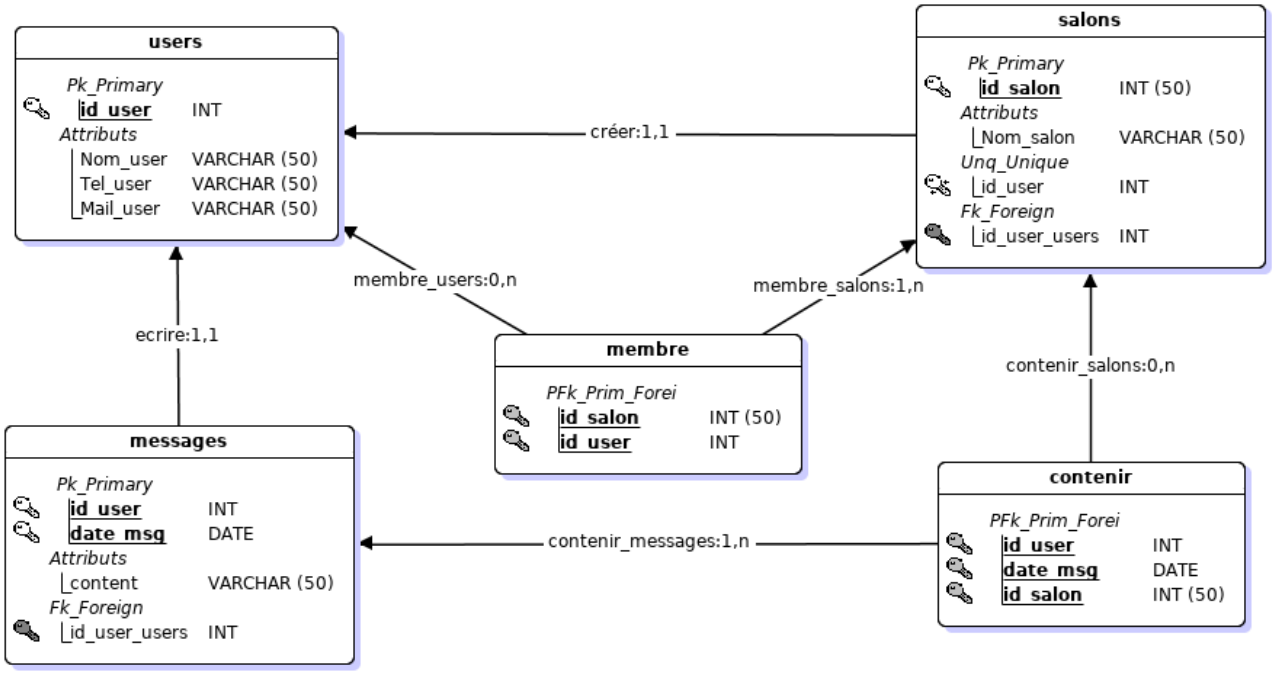
- user (0,n) discuter (0,n) user
- user (0,n) créer (1,1) salons
- user (0,n) membre (1,n) salons
- user (0,n) écrire (1,1) messages
- messages (1,1) contenir (0,n) salons

Modèle Conceptuel de Données :

Le modèle conceptuel est réalisé suite à un ensemble de règles de gestion donnant sens aux relations entre chaque table de la base de données. Ces règles sont celle définis plus haut.

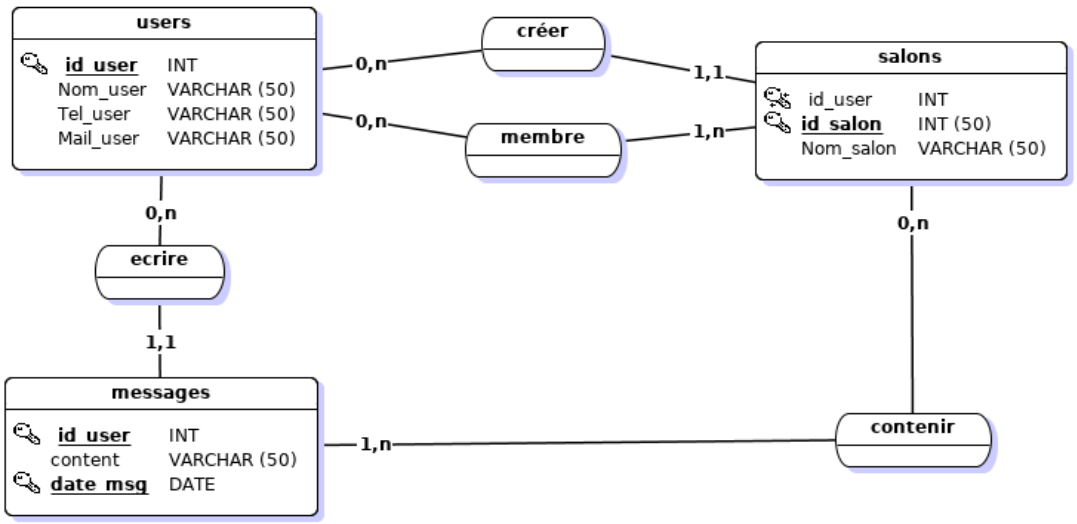
Passage du Modèle conceptuel au Modèle Logique de Données :

Le passage du MCD au MLD se fait par la prise en compte des types de relations entre chaque table et une autre. Les relations de type (n,n) donnent naissance à d'autres tables supplémentaires ayant comme clés primaires les clés primaires des tables qu'elles lient.



Modèle Logique de Données :

Le Modèle Logique de Données est représenté avec la figure suivante :



Technologies Utilisées :

Angular

Angular est un Framework coté client open source basé sur TypeScript dirigée par l'équipe du projet Angular à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de AngularJS, cadriciel construit par la même équipe.

POURQUOI DÉVELOPPER AVEC ANGULAR ?

Angular est Framework open source écrit en TypeScript. Il apporte un cadre de travail qui structure les développements. Il facilite le travail des développeurs en apportant ce qui se faisait de mieux dans l'univers de J2EE. Ce Framework est utilisé pour développer des applications web et mobile.

Avec cette technologie, on réalise des interfaces de type monopage ou "one page" qui fonctionnent sans rechargement de la page web.

Angular propose un ensemble de conventions et d'outils pour délimiter les bases d'une solution. Les développements sont ainsi optimisés, s'effectuent plus rapidement et de manière plus sûre.

Etant un langage open source, Angular est régulièrement alimenté et enrichi par une communauté importante de contributeurs. Ces derniers aident à l'amélioration du Framework dans le temps.

Les avantages de développer avec Angula

Son principal intérêt est de rendre l'expérience utilisateur sur les applications web et mobiles beaucoup plus agréables. La navigation est fluidifiée grâce notamment à la synchronisation bidirectionnelle spécifique à ce Framework.

Rapidité d'exécution conséquente : les allers-retours entre le serveur et le navigateur sont considérablement réduits. Tous les calculs se font sur la partie client. Par conséquent, quand l'utilisateur final utilise l'application il a un sentiment de réelle efficacité dans l'exécution des requêtes et dans le temps. Les réponses arrivent très rapidement. Sachant qu'aujourd'hui un utilisateur estime qu'une seconde est le temps d'attente admissible pour qu'une action s'exécute sur son site.

• Responsives : les applications développées avec Angular sont responsives et s'adaptent à tous les écrans : desktop, mobile et tablette.

• Des compétences provenant de Google : les développeurs qui maintiennent le code source d'Angular sont formés par Google, ce qui assure un certain niveau de qualité, de performance et de sécurité.

• Interfaces haut de gamme : il est un excellent choix pour créer des solutions aux interfaces graphiques complexes. Il est également choisi pour créer d'excellentes animations graphiques.

• Les directives et la déclarative building : elles améliorent considérablement le travail en équipe et les développements effectués en parallèle sur des gros projets. Grâce à approche de composants réutilisables et re-testables, les développeurs gagnent un temps considérable.

• Facilité de prise en main des projets : la structure du code est facile à lire et permet de bien se retrouver dans le projet.

• Une forte maintenabilité : le Framework facilite la maintenance des applications web ou mobile. Il est plus facile avec cette technologie de se projeter dans des évolutions et dans le temps.

Node.js

Node.js est une plateforme logicielle libre et événementielle orientée vers les applications réseau qui doivent pouvoir monter en charge.

Node.js utilise le langage JavaScript du serveur et permet de faire du JavaScript en dehors du navigateur. En se basant sur la puissance du JavaScript, Node.js propose une nouvelle façon de développer des applications web dynamiques.

Node.js bénéficie de la force de JavaScript au profit du développement de sites web dynamiques. Ce langage propose de nombreux modules pour effectuer de la conception de logiciel via une architecture logicielle simplifiée.

POURQUOI DÉVELOPPER AVEC NODE.JS ?

La rapidité et sa puissance grâce à son :

•Moteur Google V8 (moteur d'exécution de Google Chrome).

• Le système non bloquant.

Les avantages de Node.js

•Multi systèmes

Node.js fonctionne sous Linux, Mac et Windows.

•La force du langage unique

Pour fonctionner sur tous les systèmes : les librairies peuvent être partagées à la fois côté serveur et à la fois côté client (navigateur).

•Gestionnaire de paquets : package npm

La plateforme possède par défaut un système de paquets qui permet de télécharger et d'installer rapidement n'importe quoi. On y trouve des modules à destination du back-end et du front-end.

•Importante communauté

Websocket

WebSocket est un standard du Web désignant un protocole réseau de la couche application et une interface de programmation du World Wide Web visant à créer des canaux de communication full-duplex par-dessus une connexion TCP.

Pourquoi utiliser le protocole Websocket ?

WebSocket nous permets d'établir une connexion persistante, afin d'obtenir une communication bidirectionnelle entre le client et le serveur.

L'utilisation de WebSocket nous permet d'avoir :

•Une communication en temps réel.

•Moins de frais généraux du http.

• La possibilité de diffuser des données binaires de taille arbitraire entre le client et le serveur.

Base de Données SQLite:

La partie nodejs :

La base de données SQLite créée nous permet de stocker les données relatives à l'application (les noms d'utilisateurs, les mots de passe, etc…). Elle sera également accessible depuis les scripts nodejs, cette accessibilité est garantie avec l'installation du module nodejs SQLite.

La partie JavaScript :

Un fichier d'extension *.js* est créé et destiné à intégrer le code **javascript** se connectant à la base de données **mysql**.

Ces connexions étant établies, l'extraction de données à partir de la base de données mysql sera donc possible.

Conclusion :

Ce cahier des charges joue aussi le rôle du manuel de l'application de messagerie instantanée.

Durant ce projet, nous apprenons à travailler en groupe et à améliorer nos compétences de développement Web, car parmi les objectifs ici est de finir avec un travail qui fonctionne.

A travers ce travail, nous apprenons également à mener à bien des projets en mettant en place des méthodologies de gestion, d'organisation, de tests et de validation.

Annexes :

Scripte SQLite :

```
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS messages;
DROP TABLE IF EXISTS salons;
DROP TABLE IF EXISTS contenir;
DROP TABLE IF EXISTS membre;


------------------------------------------------------------
--        Script SQLite
------------------------------------------------------------


------------------------------------------------------------
-- Table: users
------------------------------------------------------------
CREATE TABLE users(
      id_user      INTEGER NOT NULL ,
      Nom_user     TEXT NOT NULL ,
      Tel_user     TEXT NOT NULL ,
      Mail_user    TEXT NOT NULL,
      CONSTRAINT users_PK PRIMARY KEY (id_user)
);


------------------------------------------------------------
-- Table: messages
------------------------------------------------------------
CREATE TABLE messages(
      id_user          INTEGER NOT NULL ,
      date_msg         NUMERIC NOT NULL ,
      content          TEXT NOT NULL ,
      id_user_users    INTEGER NOT NULL,
      CONSTRAINT messages_PK PRIMARY KEY (id_user,date_msg),
      CONSTRAINT messages_users_FK FOREIGN KEY (id_user_users)
REFERENCES users(id_user)
);

------------------------------------------------------------
-- Table: salons
------------------------------------------------------------
CREATE TABLE salons(
      id_salon         INTEGER NOT NULL ,
      id_user          INTEGER NOT NULL ,
      Nom_salon   TEXT NOT NULL,
      id_user_users    INTEGER NOT NULL,
      CONSTRAINT salons_PK PRIMARY KEY (id_salon),
      CONSTRAINT salons_users_FK FOREIGN KEY (id_user_users) REFERENCES
users(id_user)
);


------------------------------------------------------------
-- Table: contenir
------------------------------------------------------------
CREATE TABLE contenir(
      id_user     INTEGER NOT NULL ,
      date_msg    NUMERIC NOT NULL ,
      id_salon    INTEGER NOT NULL,
      CONSTRAINT contenir_PK PRIMARY KEY (id_user,date_msg,id_salon),
```

```
     CONSTRAINT contenir_messages_FK FOREIGN KEY (id_user,date_msg)
REFERENCES messages(id_user,date_msg),
     CONSTRAINT contenir_salons0_FK FOREIGN KEY (id_salon) REFERENCES
salons(id_salon)
);


------------------------------------------------------------
-- Table: membre
------------------------------------------------------------
CREATE TABLE membre(
     id_salon    INTEGER NOT NULL ,
     id_user     INTEGER NOT NULL,
     CONSTRAINT membre_PK PRIMARY KEY (id_salon,id_user),
     CONSTRAINT membre_salons_FK FOREIGN KEY (id_salon) REFERENCES
salons(id_salon),
     CONSTRAINT membre_users0_FK FOREIGN KEY (id_user) REFERENCES
users(id_user)
);


// Remplissage des tables:


------------------------------------------------------------
-- Table: users
------------------------------------------------------------

INSERT INTO users
(id_user,Nom_user,Tel_user,Mail_user)
VALUES (01,'slim',0781830724,'amine.lim@outlook.fr');

INSERT INTO users
(id_user,Nom_user,Tel_user,Mail_user)
VALUES (01,'farouq',0725430724,'jinjin@outlook.fr');

INSERT INTO users
(id_user,Nom_user,Tel_user,Mail_user)
VALUES (01,'bahaa',0781836523,'bahaasgh@outlook.fr');

INSERT INTO users
(id_user,Nom_user,Tel_user,Mail_user)
VALUES (01,'julie',0784435721,'julie.go@outlook.fr');

INSERT INTO users
(id_user,Nom_user,Tel_user,Mail_user)
VALUES (01,'ayoub',078180522,'ayoub@outlook.fr');

------------------------------------------------------------
-- Table: messages
------------------------------------------------------------
INSERT INTO messages
(id_user,date_msg,content)
VALUES (01,02/03/2019,'Bonjour! il y a quelqu un ?');

INSERT INTO messages
(id_user,date_msg,content)
VALUES (01,02/03/2019,'je suis disponible si qulqu un veut parler ;)');

INSERT INTO messages
```

```
(id_user,date_msg,content)
VALUES (02,02/03/2019,'hello toi ! ça va ?');

INSERT INTO messages
(id_user,date_msg,content)
VALUES (01,02/03/2019,'ah t là, ça va je vien de finir les examens, et
toi ?');


-------------------------------------------------------------
-- Table: salons
-------------------------------------------------------------
INSERT INTO salons
(id_user,id_salon)
VALUES (01,001);
```