Base de données

# Introduction

Afin de stocker tous les informations persistantes relatives à notre serveur de chat, nous avons utilisé une base de données de type SQL.

Nous avons décidé d'utilisé une de donnée de type SQLite. Ceci permet d'avoir une solution légère et portable. En effet, la base SQLite est directement lié au programme et permet d'être exportée facilement car elle se compose uniquement de deux fichiers :

* init.sql : ce fichier contient le schéma de la base de données, c'est-à-dire la structure de tables de la base de données. Il permet d'initialiser un fichier db.sqlite.
* db.sqlite : ce fichier contient à nouveau la structure de la DB ainsi que les données à proprement parler stockées en partie sous forme binaire.

Lorsque le serveur ne contient aucune donnée, le fichier db.sqlite est généré automatiquement à l'aide de init.sql.

# Schéma

Voici le schéma de notre base de données :

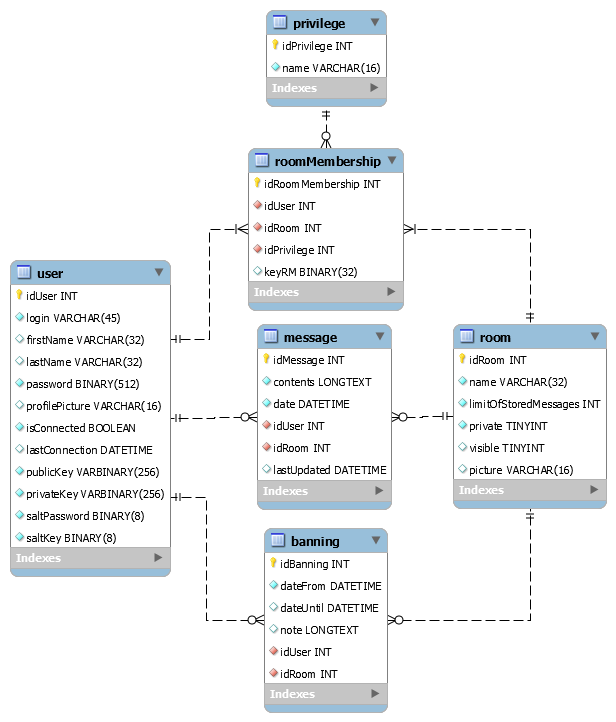


Figure Schéma de la base de donnée

Il s’agit d’un schéma plutôt simple doté de 6 tables. Il n’y a rien de particulier à noter à l’exception des différents champs propres à la sécurité de l'application tels que les sels ainsi que les paires de clés. Avec ces élément, on est capable de chiffrer des messages afin de les envoyer de manière sure mais également à ne pas stocker des informations en clair dans notre base de données.

### Table user

Cette table contient toutes les informations relatives à un utilisateur. Le stockage des clés est primordial car il permet à l'utilisateur, une fois ces dernières récupérées, de déchiffrer les messages des salles dont il est membre.

## Table message

Cette table permet de stocker les informations concernant un message : son contenu, son auteur, la salle dans laquelle il a été envoyé ainsi que la date de dernière édition dans le cas où le message aurait été modifié ultérieurement.

## Table room

Cette table contient les informations décrivant les salles telles que le type de salle (privée ou publique) et sa visibilité ou encore son image.

## Table membership

Lorsqu'un utilisateur a rejoint une salle, une nouvelle entrée est rajoutée dans cette table. L'entrée contient également une référence sur le privilège de l'utilisateur. Ainsi qu'une clé propre à chaque adhésion à une salle.

## Table privilege

Dans le stade actuel de l'application, il y a trois privilèges possibles : membre, administrateur ou en demande d'adhésion.

Le privilège membre est le plus basique et permet uniquement d'envoyer des messages. Un administrateur peut supprimer un membre d'une salle, inviter un membre dans une salle et accepter un membre dans le cas où il s'agit d'une salle privée. Le dernier privilège correspond à un utilisateur désireux de rejoindre une salle privée visible. Une fois accepté, son privilège changera en membre.

## Table banning

Nous nous ne sommes pas parvenu à développer une fonction de bannissement. Néanmoins, dans une extension future, cette table aurait permis de gérer ces bannissements.

Par exemple, un administrateur peut décider d'exclure un membre d'une salle pour une durée déterminée ou illimitées pour une raison X ou Y.

# Consultation des données

Pour des raisons de développement, nous avons été amené à vouloir consulter les informations stockées dans la base SQLite. Pour ce faire nous avons utilisés des programmes tels que SQLiteBrowser.

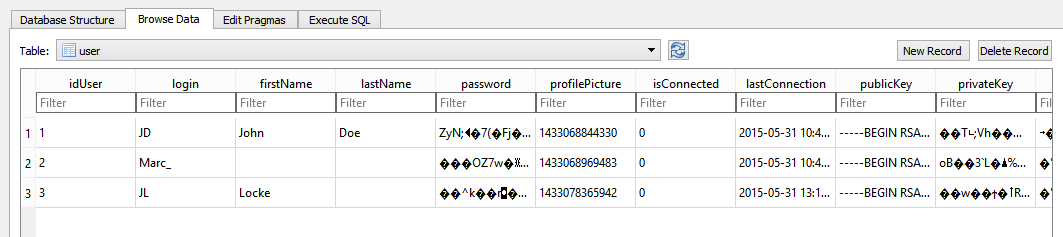


Figure Aperçu de la base de données depuis SQLiteBrowser

On voit bien que certaines informations, telles que le mot de passe, ne sont pas stockées en claires.

# Interaction avec la base de données

Les requêtes formulées pour interroger la base de données sont des requêtes SQL standards. Une fois le programme lié au fichier SQLite, on peut exécuter des requêtes à notre guise. Ces requêtes sont envoyées depuis les fichiers controllerDB.h du côté du serveur.

Exemple :

query.prepare("UPDATE message SET contents = :content, lastUpdated = datetime('NOW') WHERE idMessage = :idMessage");

query.bindValue(":content", message.getContent());

query.bindValue(":idMessage", message.getIdMessage());

query.exec();