|  |
| --- |
| C:\Users\Lwaxana\Desktop\bannnieremons_hainaut2.jpg |
| Projet Analyse et conception d’applications |
| Analyse d’une application de type ‘Chat’ |
|  |
| **Jankowski Philippe** |

Année 2013-2014

|  |
| --- |
| Ce rapport est réalisé dans le cadre du cours d’analyse et de conception d’applications du Bac Informatique de Gestion (2ème année). |

# Enoncé du projet

Le projet est une application de type ‘chat’( IRC ) basée sur une architecture client/serveur fonctionnant en réseau local.

# Les besoins du client

L’application doit permettre à l’utilisateur de :

* S’identifier
* Créer un nouveau compte
* Modifier son compte (informations personnelles et mot de passe)
* Rejoindre un ou des canaux de communications
* Modifier son pseudo
* Envoyer des messages publics sur un canal
* Envoyer des messages privés à un autre utilisateur
* Créer un salon privé
* Consulter ses logs de conversation

L’application doit permettre à un modérateur de :

* Réaliser les opérations d’un utilisateur classique
* Gérer un ou plusieurs salons
* Consulter les informations des utilisateurs des salons modérés
* Consulter les logs des salons modérés

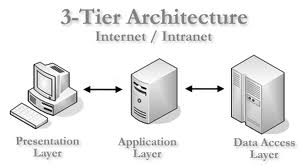
L’application doit permettre à un administrateur  de :

* Réaliser les opérations d’un utilisateur et d’un modérateur
* Gérer un ou des serveurs
  + Supprimer, ajouter, modifier, redémarrer un serveur
  + Consulter les logs de l’activité du/des serveurs
  + Ajouter un nouvel administrateur
* Gérer l’ensemble des salons
  + Créer un ou des nouveaux salons publics
  + Fermer un ou des salons publics
  + Modifier un ou des salons publics
* Gérer les modérateurs
  + Ajouter nouveau modérateur
  + Nommer nouveau modérateur
  + Attribuer un ou des salons à un modérateur
* Gérer les comptes des utilisateurs
  + Bannir un utilisateur
  + Réinitialiser le compte d’un utilisateur

# Les contraintes techniques du projet

Le chat est une application qui se connecte à un serveur de base de données distant via le réseau local afin de récupérer et d’écrire les données. Ce qui nécessite l’intégration d’un serveur client entre l’application cliente et le serveur de base de données.

L’architecture de l’application est à 3 niveaux (architecture 3-tiers).



La communication entre l’application client et le serveur client se fera par l’envoi de message codé via la sérialisation par le protocole TCP.

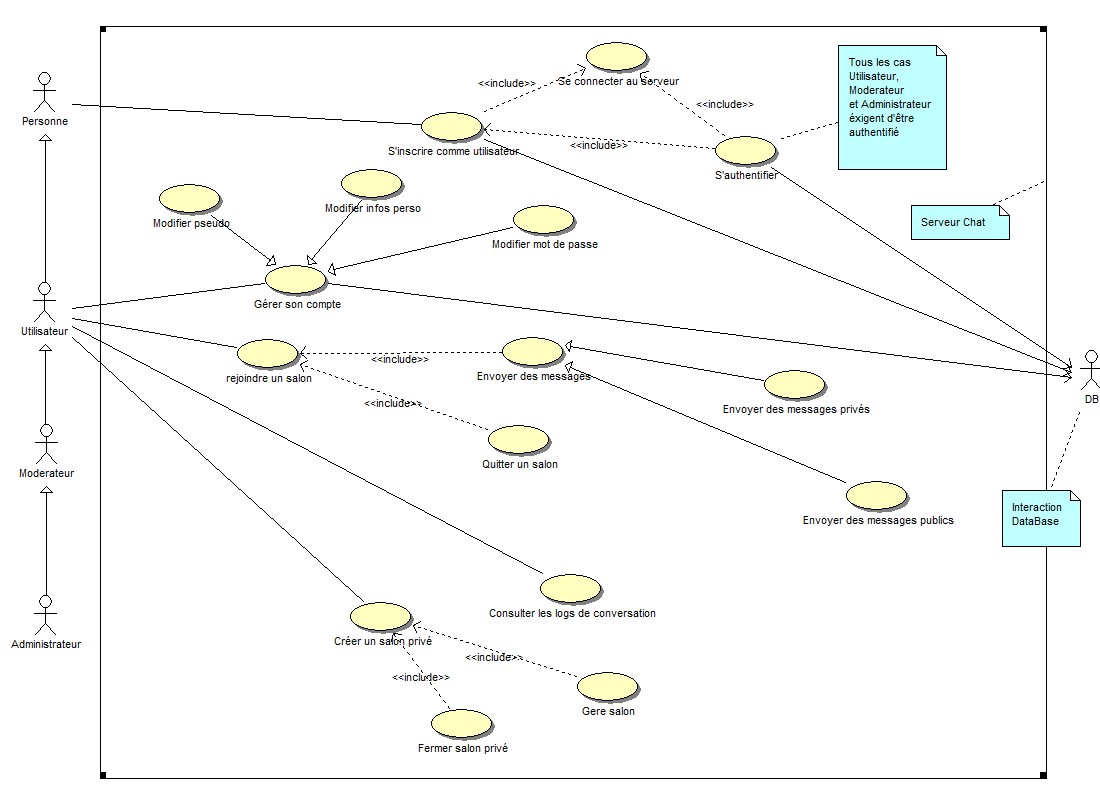
Les données ‘sensibles’ à savoir les informations de l’utilisateur, identifiants et mot de passe seront cryptées par une session SSL/TLS afin garantir la sécurité des informations.

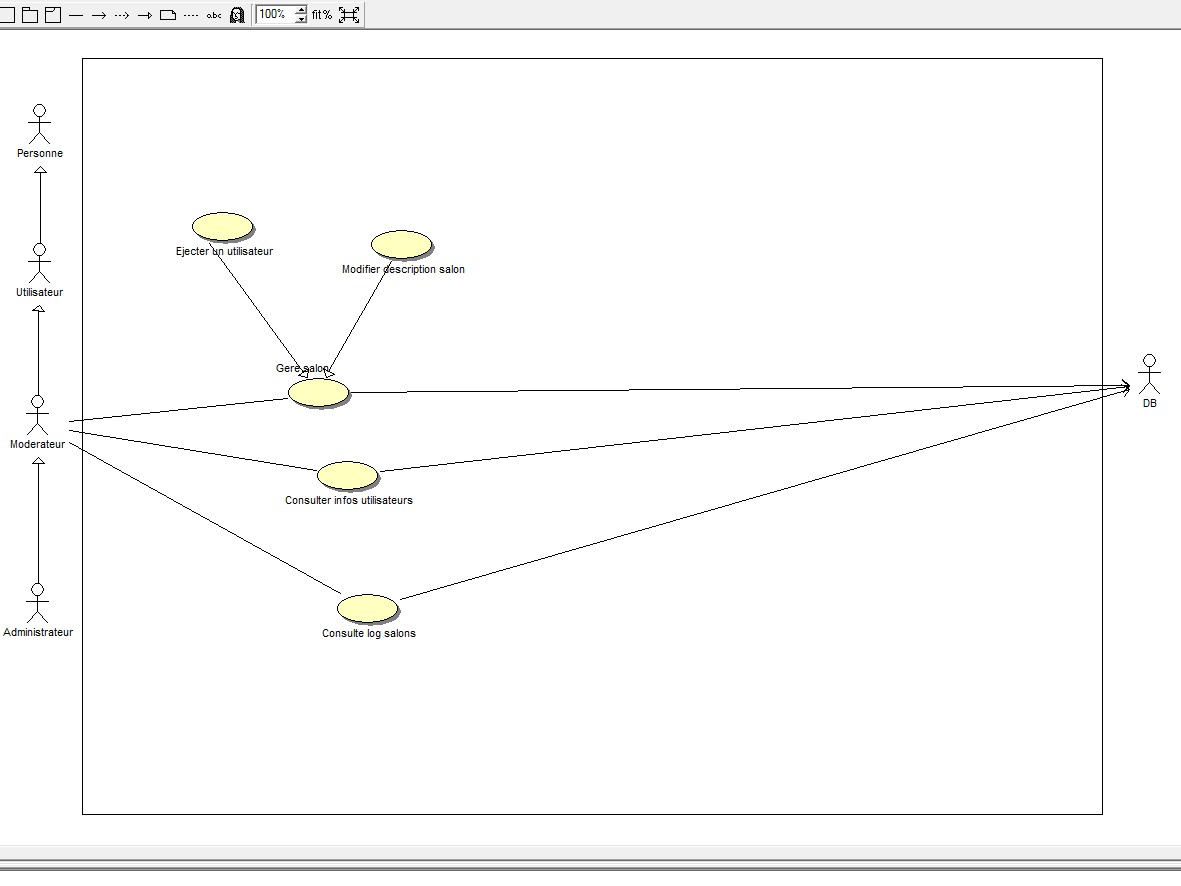
Le serveur client devra gérer la communication entre les différents clients connectés ainsi que la communication entre le serveur de base de données et le client.

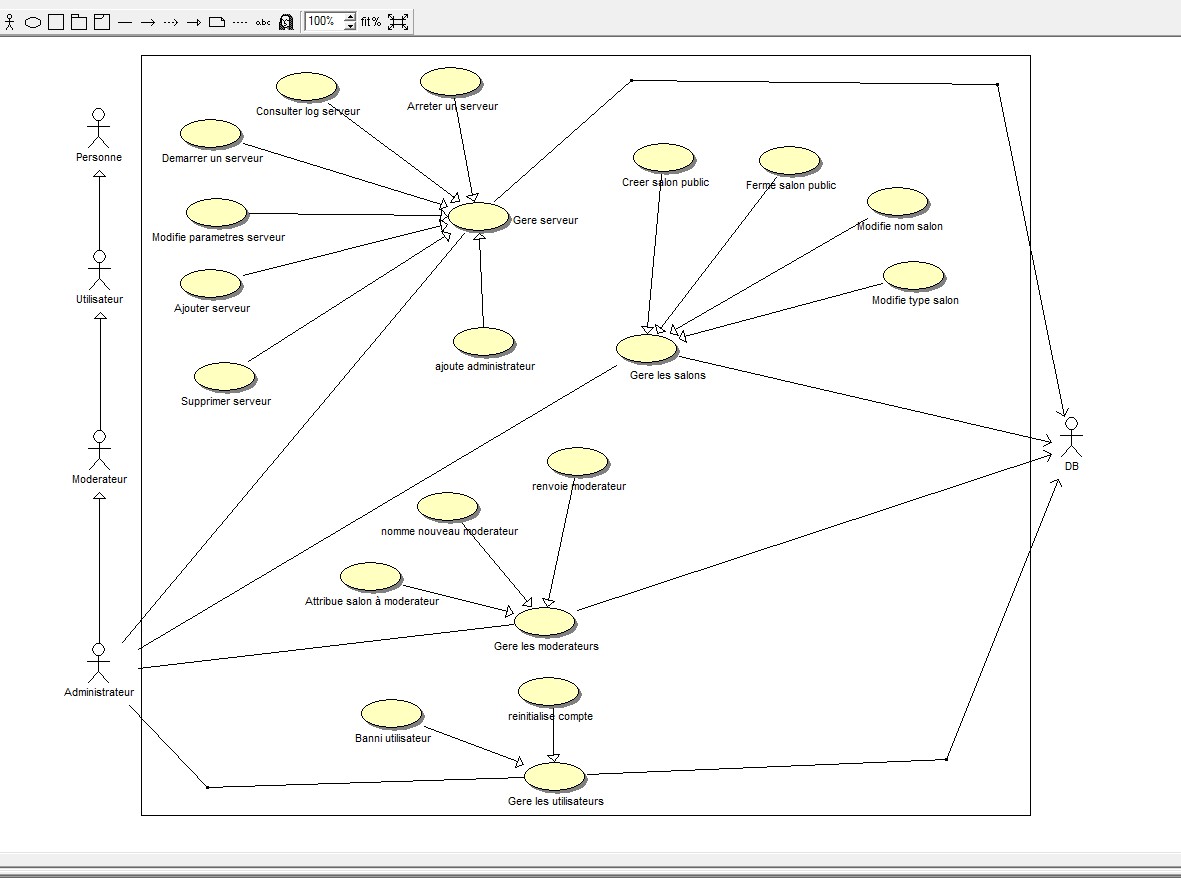
Le serveur de base de données inscrit et fournit les données au serveur client ainsi qu’aux clients.

L’interface doit être simple et compréhensible, l’utilisateur du système doit avoir une information claire et simple.

# Diagrammes UseCase







## Analyse du cas « S’inscrire comme utilisateur »

Titre : S’inscrire comme utilisateur

But : Enregistrer une personne dans la base de données et permettre l’accès au système

Résumé : Le client rempli le formulaire et valide son inscription

Acteurs : Une personne

Pré condition : le serveur client doit être online

Scénario nominal :

1. La personne remplit le formulaire d’inscription et le valide
2. L’application client se connecte au serveur
3. Le serveur valide la connexion
4. L’application client envoie une demande d’inscription et démarre une session SSL
5. Le serveur démarre une session SSL
6. L’application client envoie les données du formulaire de la personne au serveur
7. La base de données enregistre les informations de la personne
8. Le système envoie un message « Ok pour login » au client

Scénarios alternatifs :

1. Champs obligatoires non valides
   1. L’application client affiche un message d’erreur
   2. Le scénario reprend de 1
2. Le login existe déjà dans la base de données

a. L’application client affiche un message d’erreur « Login existe déjà »

b. Le scénario reprend de 1 ensuite à partir de 6

## b) Analyse du cas « Modifier ses informations perso »

Titre : Modifier ses informations perso

But : Modifier les informations personnelles de son compte.

Résumé : L’utilisateur modifie les informations de son compte. Le système met à jour les données dans la base de données.

Acteurs : Utilisateur

Pré condition : L’utilisateur est authentifié

Post condition : Les informations de l’utilisateur sont modifiées

Scénario nominal :

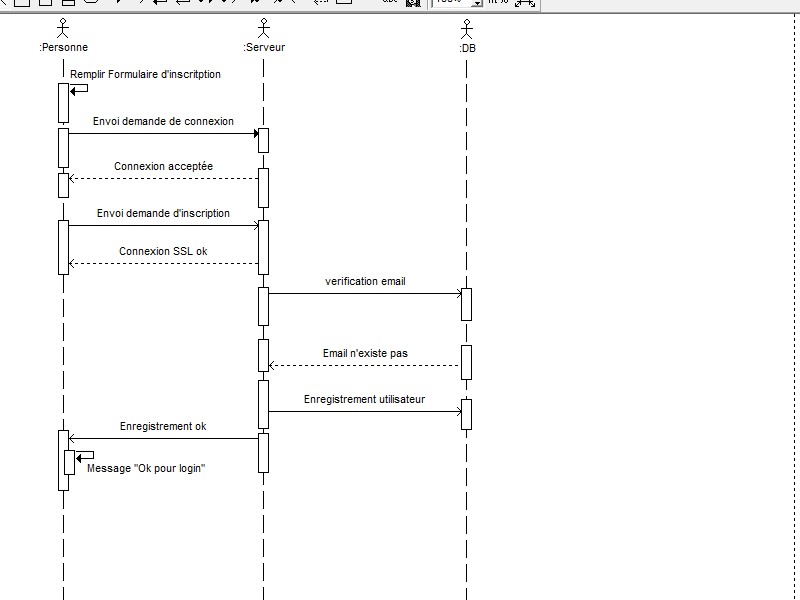
1. L’utilisateur consulte ses informations personnelles.
2. L’application affiche les informations de son compte.
3. L’utilisateur effectue les modifications nécessaires puis valide.
4. Le serveur sauvegarde les nouvelles informations.

Scénario alternatif :

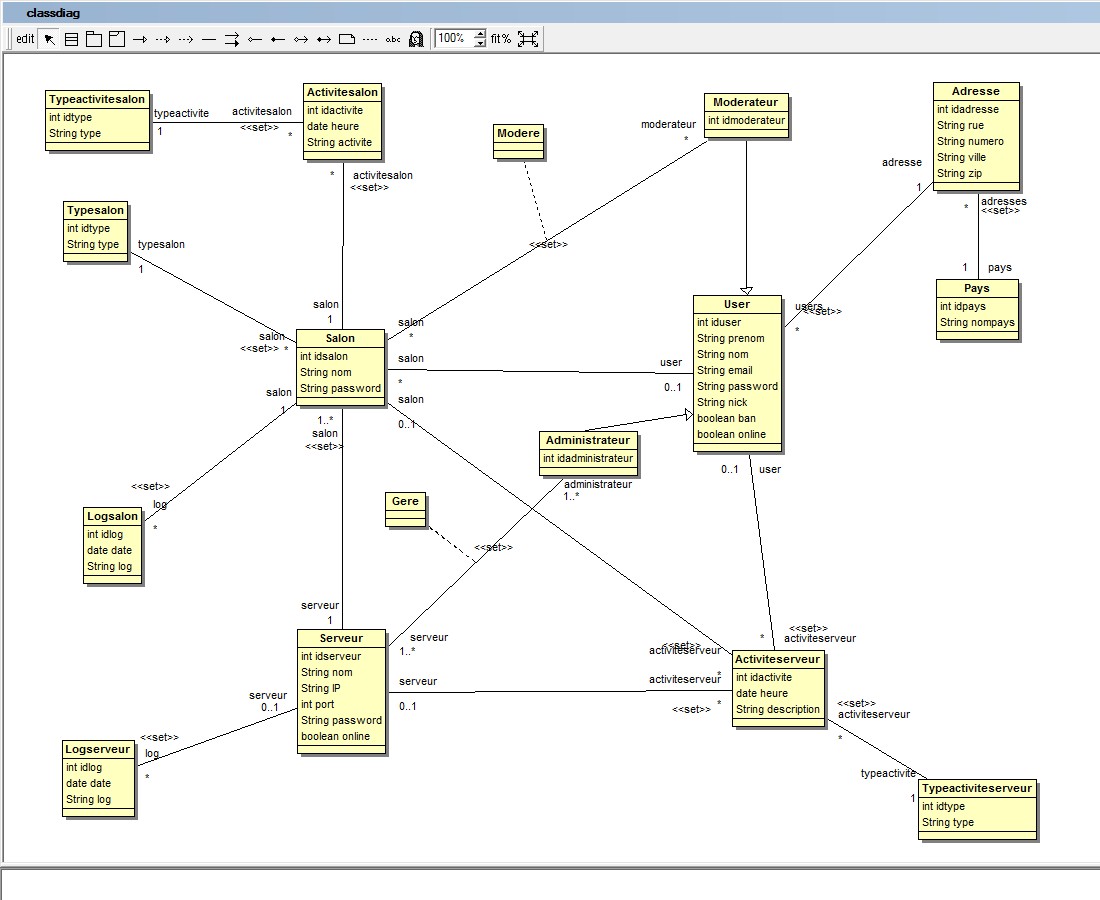
1. Champs obligatoires non valides et/ou vides
   1. L’application affiche un message d’erreur
   2. Le scénario reprend à 2

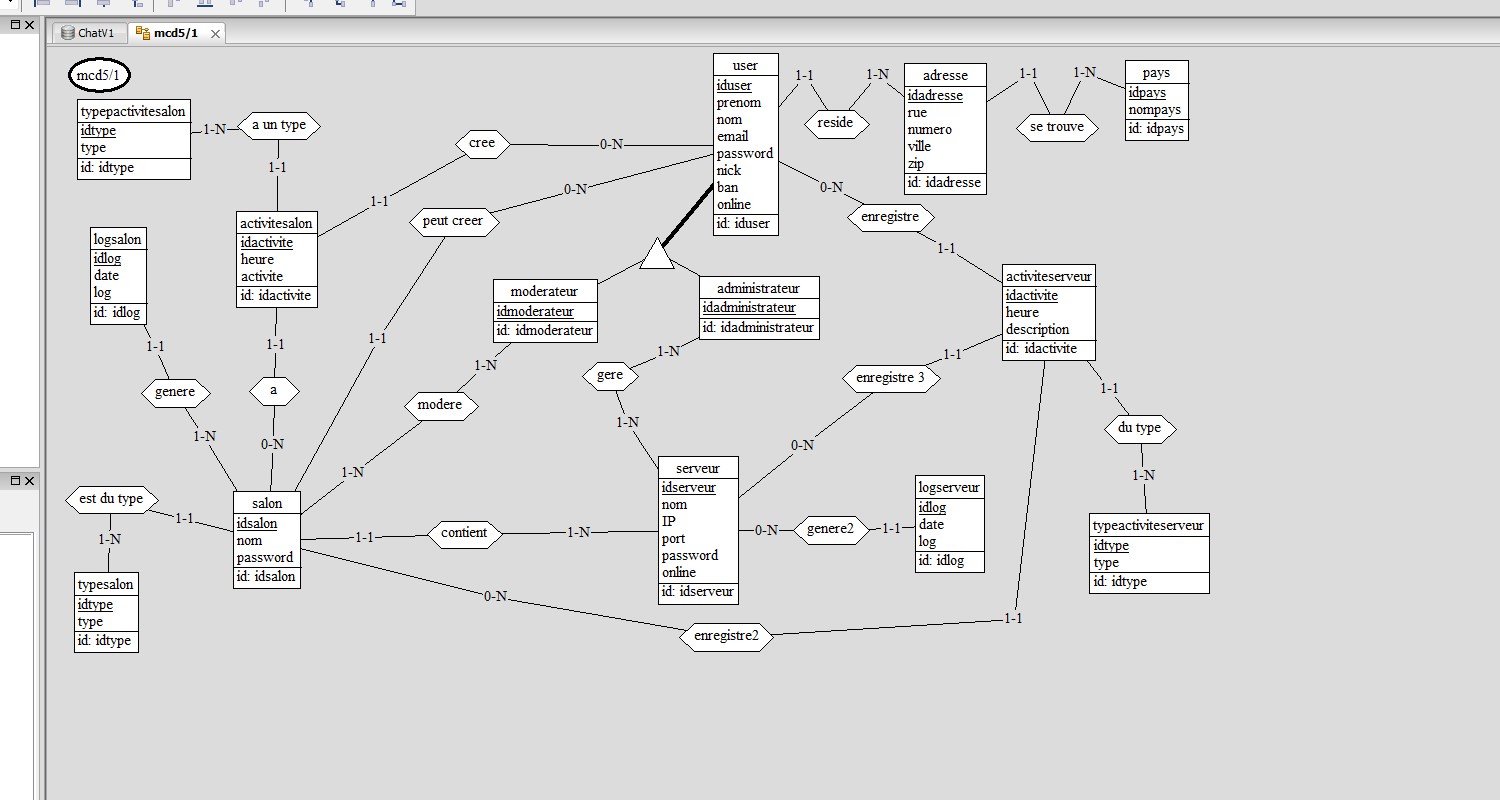
# Diagramme de séquence

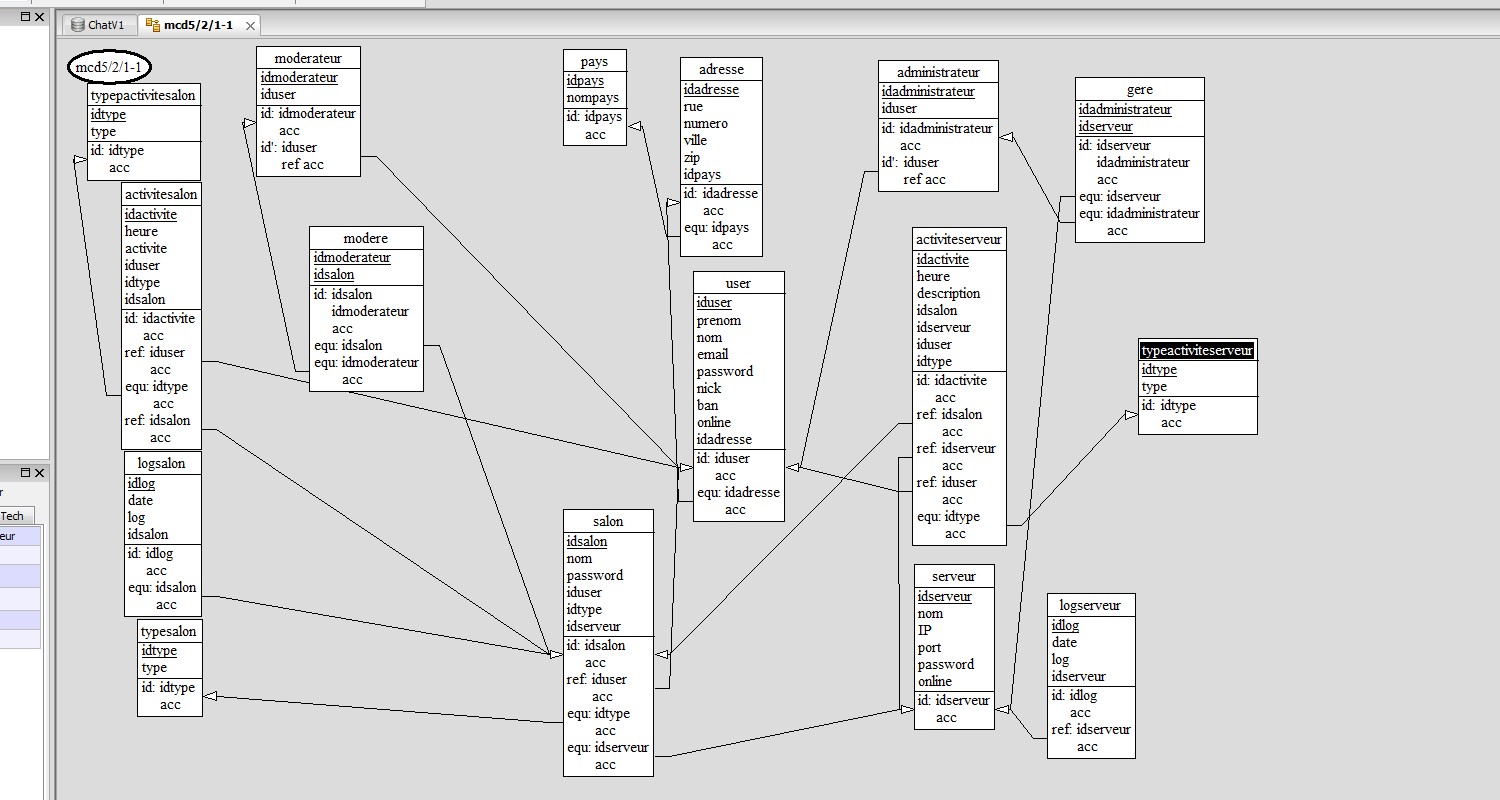
Le diagramme de séquence suivant correspond au UseCase « S’inscrire comme utilisateur »



# Diagrammes de classe







# Code SQL

create database chat;

use chat;

-- Tables Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_

create table activitesalon (

idactivite int not null AUTO\_INCREMENT,

heure date not null,

activite varchar(250) not null,

iduser int not null,

idtype int not null,

idsalon int not null,

constraint ID\_activitesalon\_ID primary key (idactivite));

create table activiteserveur (

idactivite int not null AUTO\_INCREMENT,

heure date not null,

description varchar(250) not null,

idsalon int null,

idserveur int not null,

iduser int null,

idtype int not null,

constraint ID\_activiteserveur\_ID primary key (idactivite));

create table administrateur (

idadministrateur int not null AUTO\_INCREMENT,

iduser int not null,

constraint ID\_administrateur\_ID primary key (idadministrateur),

constraint FKuse\_adm\_ID unique (iduser));

create table adresse (

idadresse int not null AUTO\_INCREMENT,

rue varchar(100) not null,

numero varchar(10) not null,

ville varchar(50) not null,

zip varchar(10) not null,

idpays int not null,

constraint ID\_adresse\_ID primary key (idadresse));

create table gere (

idadministrateur int not null ,

idserveur int not null,

constraint ID\_gere\_ID primary key (idserveur, idadministrateur));

create table logsalon (

idlog int not null AUTO\_INCREMENT,

date date not null,

log varchar(250) not null,

idsalon int not null,

constraint ID\_logsalon\_ID primary key (idlog));

create table logserveur (

idlog int not null AUTO\_INCREMENT,

date date not null,

log varchar(250) not null,

idserveur int not null,

constraint ID\_logserveur\_ID primary key (idlog));

create table moderateur (

idmoderateur int not null AUTO\_INCREMENT,

iduser int not null,

constraint ID\_moderateur\_ID primary key (idmoderateur),

constraint FKuse\_mod\_ID unique (iduser));

create table modere (

idmoderateur int not null,

idsalon int not null,

constraint ID\_modere\_ID primary key (idsalon, idmoderateur));

create table pays (

idpays int not null AUTO\_INCREMENT,

nompays varchar(100) not null,

constraint ID\_pays\_ID primary key (idpays));

create table salon (

idsalon int not null AUTO\_INCREMENT,

nom varchar(50) not null,

password varchar(20) null,

iduser int null,

idtype int not null,

idserveur int not null,

constraint ID\_salon\_ID primary key (idsalon));

create table serveur (

idserveur int not null AUTO\_INCREMENT,

nom varchar(50) not null,

IP varchar(15) not null,

port int not null,

password varchar(64) null,

online char not null,

constraint ID\_serveur\_ID primary key (idserveur));

create table typeactiviteserveur (

idtype int not null AUTO\_INCREMENT,

type varchar(50) not null,

constraint ID\_typeactiviteserveur\_ID primary key (idtype));

create table typepactivitesalon (

idtype int not null AUTO\_INCREMENT,

type varchar(100) not null,

constraint ID\_typepactivitesalon\_ID primary key (idtype));

create table typesalon (

idtype int not null AUTO\_INCREMENT,

type varchar(50) not null,

constraint ID\_typesalon\_ID primary key (idtype));

create table user (

iduser int not null AUTO\_INCREMENT,

prenom varchar(25) not null,

nom varchar(50) not null,

email varchar(50) not null UNIQUE,

password varchar(64) not null,

nick varchar(20) not null,

ban tinyint not null,

online tinyint not null,

idadresse int not null,

constraint ID\_user\_ID primary key (iduser));

-- Constraints Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

alter table activitesalon add constraint FKcree\_FK

foreign key (iduser)

references user (iduser);

alter table activitesalon add constraint FKa\_un\_type\_FK

foreign key (idtype)

references typepactivitesalon (idtype);

alter table activitesalon add constraint FKa\_FK

foreign key (idsalon)

references salon (idsalon);

alter table activiteserveur add constraint FKenregistre2\_FK

foreign key (idsalon)

references salon (idsalon);

alter table activiteserveur add constraint FKenregistre\_3\_FK

foreign key (idserveur)

references serveur (idserveur);

alter table activiteserveur add constraint FKenregistre\_FK

foreign key (iduser)

references user (iduser);

alter table activiteserveur add constraint FKdu\_type\_FK

foreign key (idtype)

references typeactiviteserveur (idtype);

alter table administrateur add constraint FKuse\_adm\_FK

foreign key (iduser)

references user (iduser);

alter table adresse add constraint FKse\_trouve\_FK

foreign key (idpays)

references pays (idpays);

alter table gere add constraint FKger\_ser

foreign key (idserveur)

references serveur (idserveur);

alter table gere add constraint FKger\_adm\_FK

foreign key (idadministrateur)

references administrateur (idadministrateur);

alter table logsalon add constraint FKgenere\_FK

foreign key (idsalon)

references salon (idsalon);

alter table logserveur add constraint FKgenere2\_FK

foreign key (idserveur)

references serveur (idserveur);

alter table moderateur add constraint FKuse\_mod\_FK

foreign key (iduser)

references user (iduser);

alter table modere add constraint FKmod\_sal

foreign key (idsalon)

references salon (idsalon);

alter table modere add constraint FKmod\_mod\_FK

foreign key (idmoderateur)

references moderateur (idmoderateur);

alter table salon add constraint FKpeut\_creer\_FK

foreign key (iduser)

references user (iduser);

alter table salon add constraint FKest\_du\_type\_FK

foreign key (idtype)

references typesalon (idtype);

alter table salon add constraint FKcontient\_FK

foreign key (idserveur)

references serveur (idserveur);

alter table user add constraint FKreside\_FK

foreign key (idadresse)

references adresse (idadresse);

-- Index Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_

create unique index ID\_activitesalon\_IND

on activitesalon (idactivite);

create index FKcree\_IND

on activitesalon (iduser);

create index FKa\_un\_type\_IND

on activitesalon (idtype);

create index FKa\_IND

on activitesalon (idsalon);

create unique index ID\_activiteserveur\_IND

on activiteserveur (idactivite);

create index FKenregistre2\_IND

on activiteserveur (idsalon);

create index FKenregistre\_3\_IND

on activiteserveur (idserveur);

create index FKenregistre\_IND

on activiteserveur (iduser);

create index FKdu\_type\_IND

on activiteserveur (idtype);

create unique index ID\_administrateur\_IND

on administrateur (idadministrateur);

create unique index FKuse\_adm\_IND

on administrateur (iduser);

create unique index ID\_adresse\_IND

on adresse (idadresse);

create index FKse\_trouve\_IND

on adresse (idpays);

create unique index ID\_gere\_IND

on gere (idserveur, idadministrateur);

create index FKger\_adm\_IND

on gere (idadministrateur);

create unique index ID\_logsalon\_IND

on logsalon (idlog);

create index FKgenere\_IND

on logsalon (idsalon);

create unique index ID\_logserveur\_IND

on logserveur (idlog);

create index FKgenere2\_IND

on logserveur (idserveur);

create unique index ID\_moderateur\_IND

on moderateur (idmoderateur);

create unique index FKuse\_mod\_IND

on moderateur (iduser);

create unique index ID\_modere\_IND

on modere (idsalon, idmoderateur);

create index FKmod\_mod\_IND

on modere (idmoderateur);

create unique index ID\_pays\_IND

on pays (idpays);

create unique index ID\_salon\_IND

on salon (idsalon);

create index FKpeut\_creer\_IND

on salon (iduser);

create index FKest\_du\_type\_IND

on salon (idtype);

create index FKcontient\_IND

on salon (idserveur);

create unique index ID\_serveur\_IND

on serveur (idserveur);

create unique index ID\_typeactiviteserveur\_IND

on typeactiviteserveur (idtype);

create unique index ID\_typepactivitesalon\_IND

on typepactivitesalon (idtype);

create unique index ID\_typesalon\_IND

on typesalon (idtype);

create unique index ID\_user\_IND

on user (iduser);

create index FKreside\_IND

on user (idadresse);

create unique index email\_user

on user (email);

# Maquettes

Voir feuilles suivantes