CAHIER DES CHARGES

Projet d'Atelier de Professionnalisation

Application de gestion d'images artistiques — Systeme Gestion Image (Gravitart)

Client	Gravitart — Magasin de fournitures artistiques
Prestataire	Individu libéral.
Applications	Back-end desktop Java (ce dépôt) + front web PHP (autre dépôt)
Base de données	MariaDB — `GestionSystemeImage`

1. Cadre du projet

1.1 Description du projet

Gravitart souhaite proposer une plateforme communautaire de publication d'images artistiques, inspirée des galeries visuelles en ligne. Ce dépôt constitue le back-end d'administration : une application desktop Java permettant la gestion complète des utilisateurs, des images, des tags et des signalements, connectée à la même base MariaDB que le front web PHP.

1.2 Contexte de l'entreprise

L'entreprise veut attirer de nouveaux artistes et valoriser son image de marque, tout en gardant un contrôle éditorial sur les contenus publiés.

Enjeux principaux : - Gestion centralisée des comptes utilisateurs (création, modification, suppression) - Modération des images et des signalements - Gestion du catalogue de tags - Interface sécurisée réservée aux opérateurs internes (connexion par email + mot de passe BCrypt)

1.3 Objectifs

Référence	Description
Obj. 1	Permettre la connexion sécurisée à la base MariaDB via une interface de configuration et d'authentification
Obj. 2	Autoriser la création, la modification et la suppression de comptes utilisateurs avec

Référence	Description
	attribution de rôle
Obj. 3	Permettre la gestion complète des images (téléversement BLOB, modification de visibilité, suppression) et de leurs tags associés
Obj. 4	Permettre la consultation et la suppression des signalements
Obj. 5	Restreindre les fonctions d'administration aux seuls utilisateurs de rôle Admin (id_role = 1)
Obj. 6	S'appuyer sur la base MariaDB `GestionSystemeImage` partagée avec le front web PHP

1.4 Livrables

Livrable	Contenu
L1	Application desktop Java (architecture MVC + Swing)
L2	Schéma relationnel MariaDB `GestionSystemeImage`
L3	Script SQL d'initialisation/export de la base
L4	Documentation technique consolidée (ce document)
L5	Référentiel des rôles, règles d'accès et mesures de sécurité

---

2. Conception de l'interface

2.1 Principes visuels

- Interface desktop orientée gestion administrative (fenêtres Swing modales)
- Fenêtre principale unique (`VuePrincipale`) à barre de menus dynamique
- Navigation par boîtes de dialogue dédiées à chaque fonctionnalité
- Menus affichés ou masqués selon le rôle de l'utilisateur connecté

2.2 Fenêtres principales

- `VuePrincipale` : fenêtre-racine avec barre de menus dynamique (Session, Galerie, Administration)
- `VueConnexion` : dialogue d'authentification email + mot de passe
- `VueConfiguration` : dialogue de configuration de la connexion à la base (SGBDR, hôte, port, base, utilisateur, mot de passe)
- `VueCreerUtilisateur` : formulaire de création d'un compte (admin uniquement)
- `VueListerUtilisateur` : tableau de liste et gestion des comptes (admin uniquement)
- `VueGestionImages` : gestion CRUD des images avec prévisualisation, filtrage par tag et gestion des tags associés
- `VueGestionTags` : gestion du catalogue de tags (admin uniquement)
- `VueGestionSignalements` : consultation et suppression des signalements (admin uniquement)
- `VueMessage` : boîte de dialogue d'information/erreur générique

---

3. Spécifications fonctionnelles

3.1 Rôles applicatifs

Rôle	id_role	Description
Admin	1	Accès complet : gestion utilisateurs, images, tags, signalements
Artiste	2	Gestion de ses propres images et tags ; lecture de la galerie
Invité	3	Lecture seule (images publiques uniquement)

Remarque : la vérification du rôle est effectuée à l'ouverture de chaque vue ; toute tentative d'accès admin par un non-admin est bloquée avec un message d'erreur.

3.2 Fonctionnalités par rôle

OBJECTIF	ACTEUR	IMPACT	FONCTIONNALITÉS
Gérer l'accès à l'application	Tous	Session	Configuration DB, connexion, déconnexion

OBJECTIF	ACTEUR	IMPACT	FONCTIONNALITÉS
Gérer les comptes	Admin	Utilisateurs	Création, modification (avec ou sans changement de mot de passe), suppression, listing avec rôle
Gérer les images	Admin / Artiste	Images	Téléversement BLOB via `JFileChooser`, affichage, modification de visibilité (publique/privée), suppression, tri et filtrage par tag
Gérer les tags	Admin	Classification	Ajout et suppression de tags du catalogue et des associations image–tag
Consulter et modérer les signalements	Admin	Gouvernance	Liste complète des signalements avec auteur et motif ; suppression d'un signalement
Explorer la galerie	Artiste / Invité	Consultation	Affichage des images avec informations auteur ; filtre par tag

4. Spécifications techniques

4.a Présentation de la solution

La solution est composée de : - Une application desktop Java (ce dépôt) organisée en couches `Controler/`, `Model/`, `View/` - Une base MariaDB `GestionSystemeImage` partagée avec le front web PHP - Un front web PHP complémentaire (autre dépôt) assurant l'accès public à la galerie

L'application desktop couvre les fonctions d'administration étendues : gestion des comptes, des images (BLOB), des tags et des signalements, depuis un poste interne.

4.b Choix technologiques

Technologie	Usage	Justification
Java SE	Logique applicative	Portabilité, typage fort, écosystème riche
Swing (javax.swing)	Interface graphique	Bibliothèque GUI intégrée au JDK ; fenêtres modales adaptées à un outil d'administration
JDBC + PreparedStatement	Accès base de données	Protection native contre les injections SQL
MariaDB / MySQL Connector	Pilote JDBC	Compatibilité avec la base `GestionSystemeImage` (mariadb-java-client-3.0.6.jar)
Architecture MVC	Organisation du code	Séparation contrôleur / modèle / vues ; couplage faible via interfaces
Pattern Observer	Synchronisation modèle–vue	Mise à jour automatique des vues lors des changements de session
BCrypt (jbcrypt-0.4.jar)	Sécurité des mots de passe	Hachage compatible avec `password_hash` / `password_verify` PHP ; résistant aux attaques par force

Technologie	Usage	Justification
		brute

4.c Architecture applicative

i. Contrôleur

- `AbstractControler` : classe abstraite détenant la référence au `Modele` ; implémente `InterfCRUD`
- `Controleur` : implémentation concrète ; implémente `InterfOperationsVuePrincipale` et `InterfOperationsAdmin` ; délègue toutes les opérations au modèle

ii. Couche modèle / DAO

Classes métier : `Utilisateur`, `Image`, `Tags`, `Signalement`, `Signale`, `Associe`, `Role`, `Options`, `Employe`, `Type_Employe`

DAO : `UtilisateurDAO`, `ImageDAO`, `TagsDAO`, `SignalementDAO`, `SignaleDAO`, `AssocieDAO`, `RoleDAO`, `OptionsDAO`, `SystemeDAO`

Classes système : `Session`, `ConfigurationDB`, `ConnectionDB`, `Menu`, `Option`

Utilitaires : `PasswordUtil` (BCrypt), `BCrypt`, `PasswordMigrationUtil`, `Outils`, `DateChecker`, `XML_POJO`

iii. Flux fonctionnels clés

- **Connexion :** `VueConnexion` → `Controleur::exist(user)` → `SystemeDAO::exist` → vérification email + `PasswordUtil.verifyPassword` (BCrypt) → `setUserSession` → `notifyObserver` → mise à jour de `VuePrincipale`
- **Configuration DB :** `VueConfiguration` → `Controleur::connect(ConfigurationDB)` → `Session::connect` → `ConnectionDB::connectDB`
- **Création d'utilisateur :** `VueCreerUtilisateur` → `Controleur::insertinto(Utilisateur)` → `UtilisateurDAO::insertinto` → hachage BCrypt du mot de passe + `INSERT INTO UTILISATEUR`
- **Gestion des images :** `VueGestionImages` → `Controleur::selectAllInfo(Image)` → `ImageDAO::selectAllInfo` → affichage dans `JTable` ; téléversement via `JFileChooser` → `ImageDAO::insertinto` (BLOB)
- **Gestion des tags :** `VueGestionTags` → `Controleur::insertinto/deletefrom(Tags)` → `TagsDAO` ; associations via `AssocieDAO`
- **Gestion des signalements :** `VueGestionSignalements` → `Controleur::selectAllInfo(Signalement)` → `SignalementDAO::selectAllInfo` ; suppression via `deletefrom`

4.d Base de données

i. Dictionnaire de données (synthèse)

Mnémonique	Type	Sensibilité	Observation

Mnémonique	Type	Sensibilité	Observation
id_utilisateur	INT AI	Personnelle	PK UTILISATEUR
nom_utilisateur / prenom_utilisateur	VARCHAR(50)	Personnelle	Identité
mail_utilisateur	VARCHAR(50)	Personnelle	UNIQUE ; normalisé en minuscules
cell_utilisateur	VARCHAR(15)	Personnelle sensible	UNIQUE
mdp	VARCHAR(200)	Sensible	Hash BCrypt (compatible PHP)
id_role	INT	-	FK vers ROLE
id_image	INT AI	-	PK IMAGE
image	LONGBLOB	Potentiellement sensible	Données binaires de l'image
switchpriv	TINYINT(1)	-	Visibilité : 0 = publique, 1 = privée
tag	VARCHAR(100)	-	PK TAGS
id_signalement	INT AI	-	PK SIGNALEMENT
libelle_signalement	ENUM	-	Motif normalisé
libelle_signalement_custom	VARCHAR(250)	-	Complément libre

ii. Dépendances fonctionnelles (principales)

- id_utilisateur → nom_utilisateur, prenom_utilisateur, mail_utilisateur, cell_utilisateur, mdp, id_role
- id_role → libelle_role
- id_image → image, switchpriv, id_utilisateur
- tag → (valeur du tag)
- id_signalement → id_utilisateur, id_image, libelle_signalement, libelle_signalement_custom

iii. MCD (entités et associations)

Entités : UTILISATEUR, ROLE, IMAGE, TAGS, SIGNALEMENT

Associations : - `ASSOCIE(id_image, tag)` : liaison n,n entre images et tags - `SIGNALE(id_image, id_signalement, id_utilisateur)` : liaison entre image signalée, motif et auteur du signalement

iv. MLD (vue relationnelle)

- `ROLE(id_role, libelle_role)`
- `UTILISATEUR(id_utilisateur, nom_utilisateur, prenom_utilisateur, mail_utilisateur, cell_utilisateur, mdp, id_role#)`
- `IMAGE(id_image, image, switchpriv, id_utilisateur#)`
- `TAGS(tag)`
- `SIGNALEMENT(id_signalement, id_utilisateur#, id_image#, libelle_signalement, libelle_signalement_custom)`
- `ASSOCIE(id_image#, tag#)`
- `SIGNALE(id_image#, id_signalement#, id_utilisateur#)`

v. Script SQL et ordre logique de création

1. Tables de référence : `ROLE`, `TAGS`
2. Utilisateurs : `UTILISATEUR`
3. Médias : `IMAGE`
4. Signalements : `SIGNALEMENT`
5. Tables de liaison : `ASSOCIE`, `SIGNALE`

4.e Cybersécurité

Mesure	Portée	Détail
PreparedStatement JDBC	DAO	Protection contre les injections SQL sur toutes les requêtes paramétrées
Hash BCrypt (jbcrypt)	Authentification	`PasswordUtil.hashPassword` (coût 10) à la création/modification ; `PasswordUtil.verifyPassword` à la connexion
Compatibilité PHP `password_hash`	Interopérabilité	Le hash produit par Java (`$2a$`) est vérifié côté PHP via `password_verify`, et vice-versa
Contrôle d'accès par rôle	Vues	Vérification de `id_role == 1` avant ouverture des vues

Mesure	Portée	Détail
		d'administration ; message d'accès refusé sinon
Normalisation de l'email	Authentification	Conversion en minuscules à la saisie (`setMail`) et comparaison insensible à la casse en base
Configuration DB modifiable	Connexion	Les paramètres de connexion (hôte, port, base, utilisateur, mot de passe) sont modifiables à l'exécution via `VueConfiguration` sans recompilation

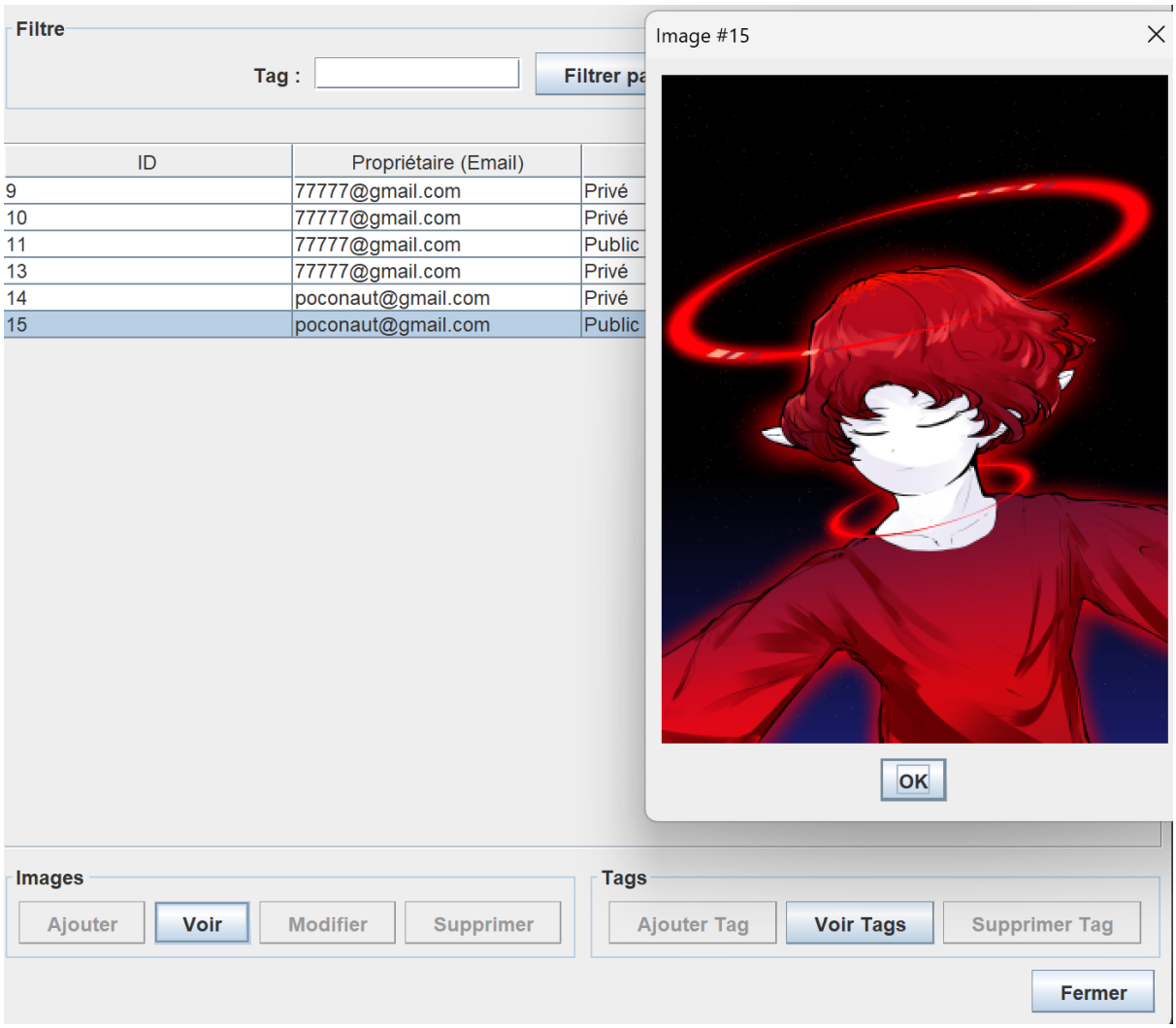
4.f Écarts et cohérence avec l'existant

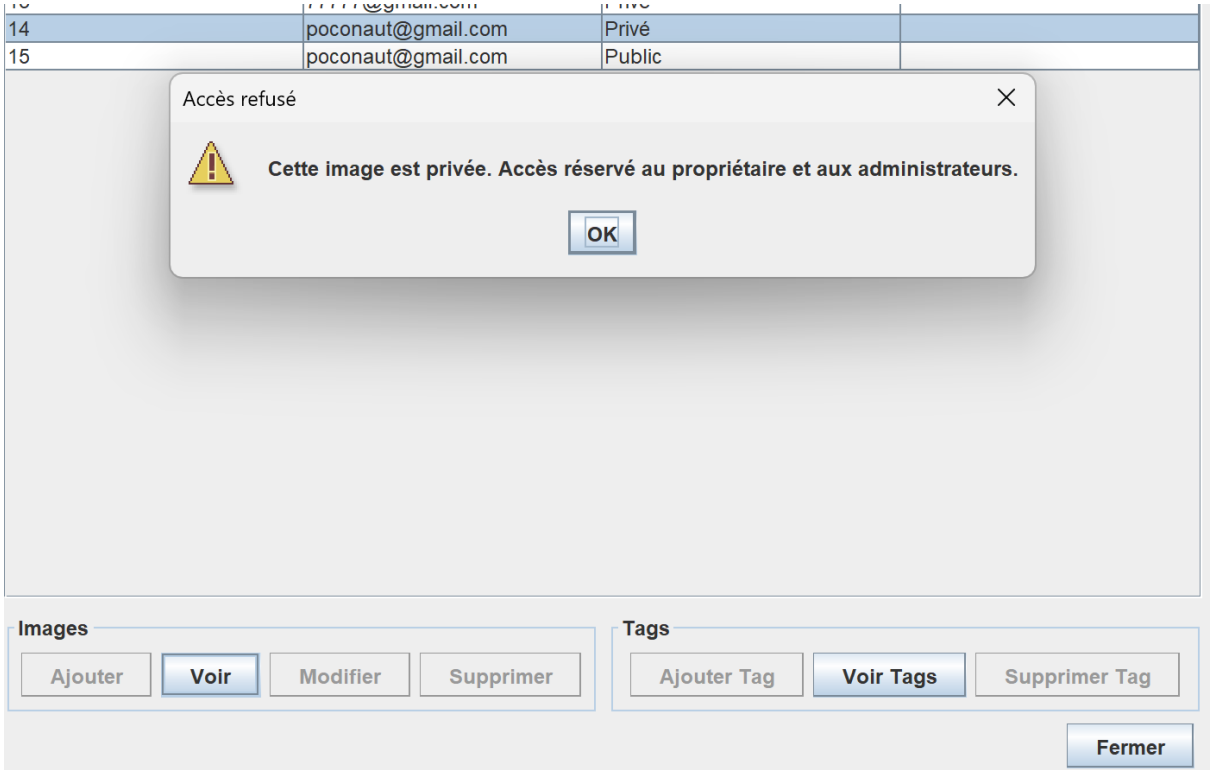
- Ce dépôt constitue le back-end desktop Java mentionné comme application complémentaire dans la documentation du front PHP.
- Le schéma de base de données est partagé et identique entre les deux dépôts.
- Les tables `avoir_acces`, `OPTION` et `comporte` (gestion de menus dynamiques par rôle en base) ne sont pas présentes dans le schéma actuel ; les menus sont construits directement dans `VuePrincipale` selon le `id_role` de l'utilisateur connecté.
- Cette documentation est alignée sur l'état réel du code et du schéma SQL présents dans ce projet.

CAPTURES D'ECRAN

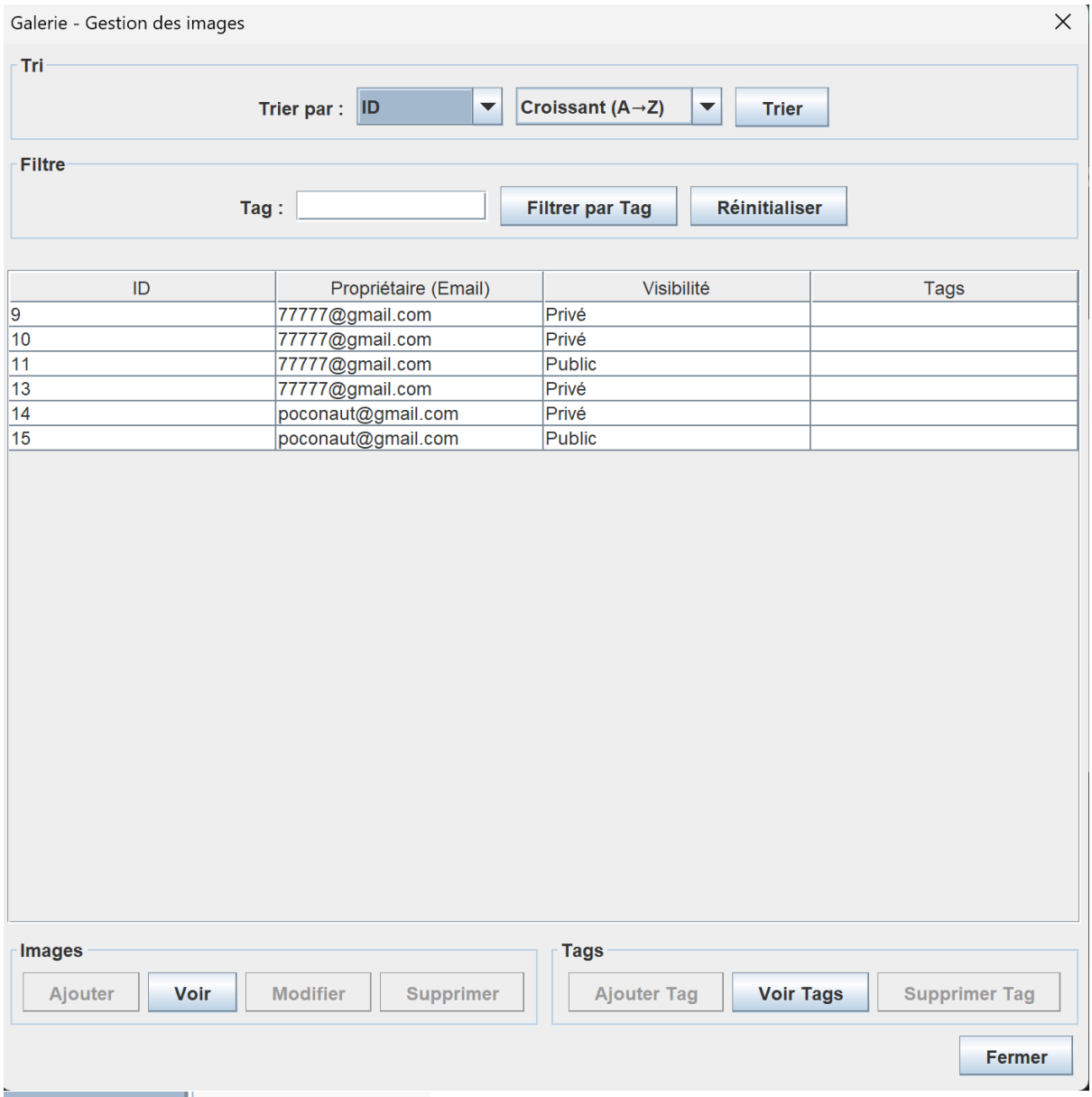
Vue du backend JAVA:

- administration:

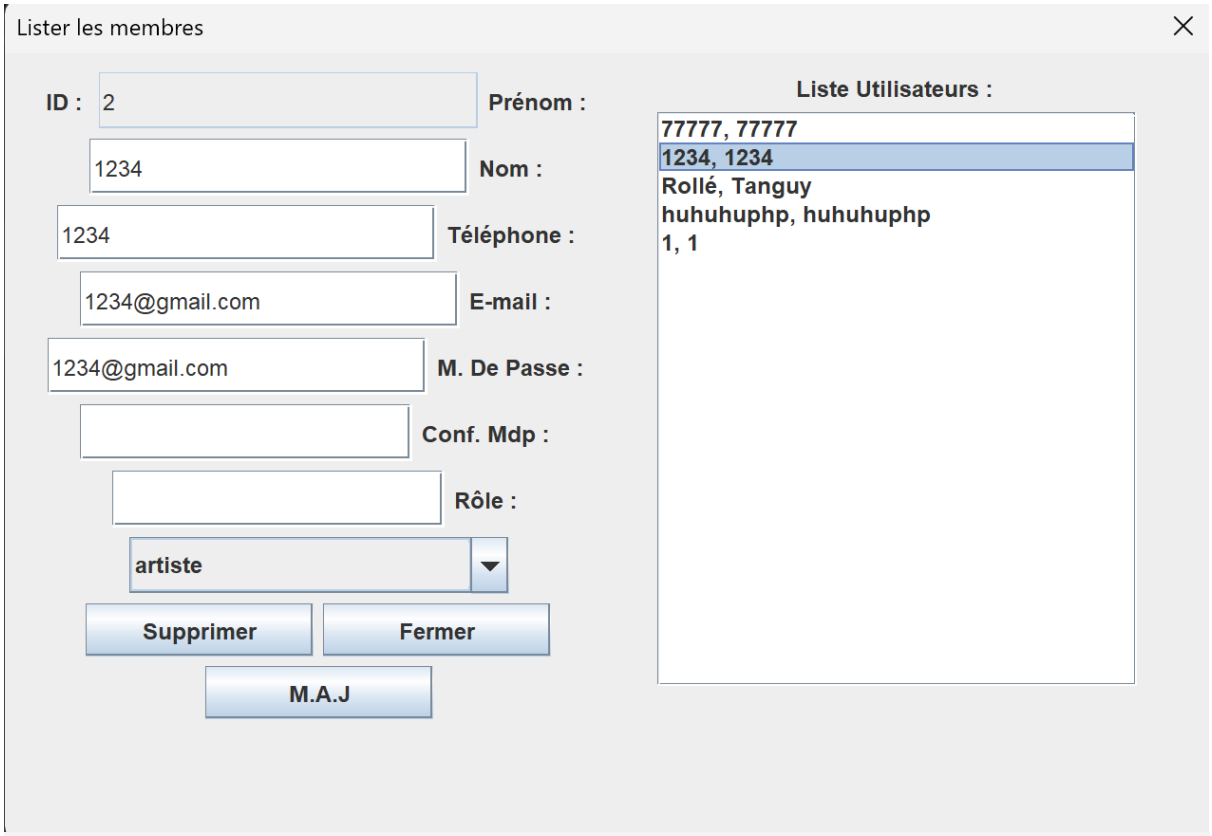


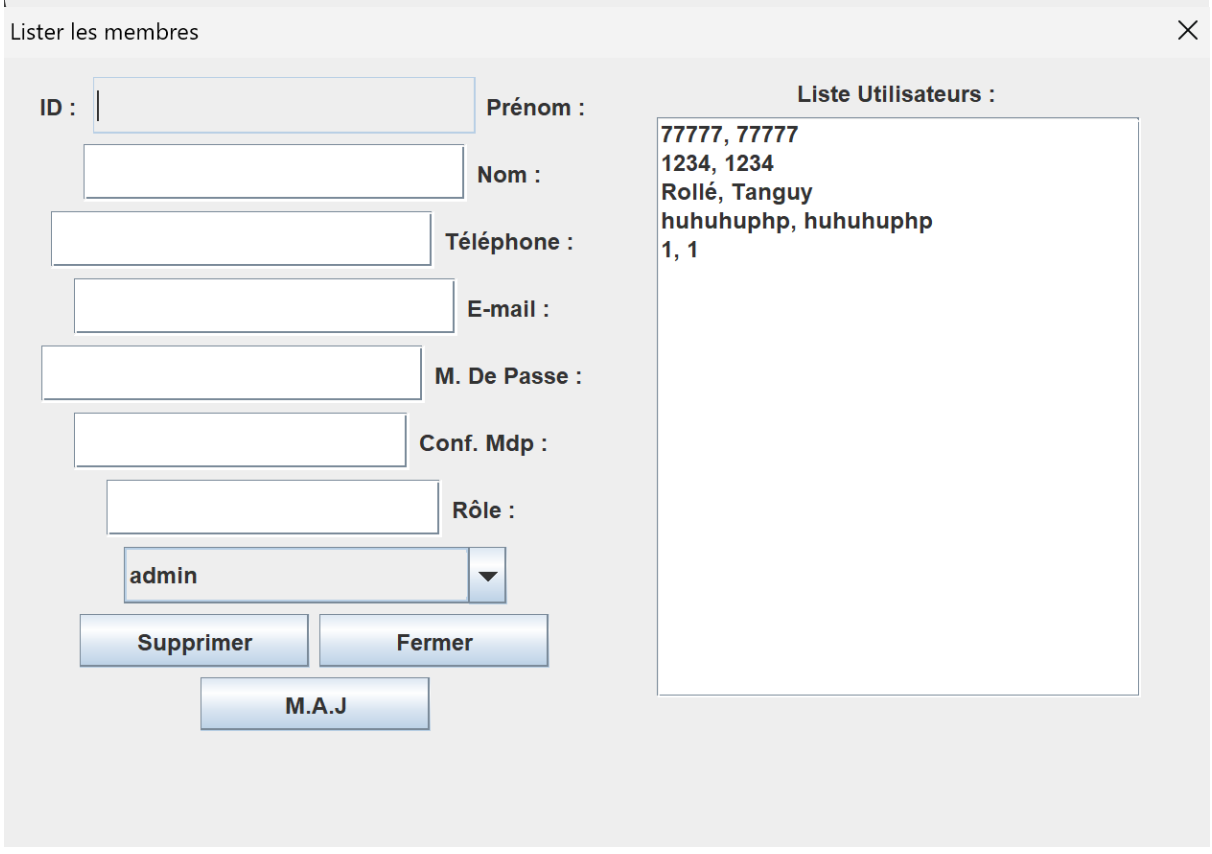

14
poconaut@gmail.com
Privé
15
poconaut@gmail.com
Public
Accès refusé
Cette image est privée. Accès réservé au propriétaire et aux administrateurs.
OK

Images
Ajouter
Voir
Modifier
Supprimer
Tags
Ajouter Tag
Voir Tags
Supprimer Tag
Fermer

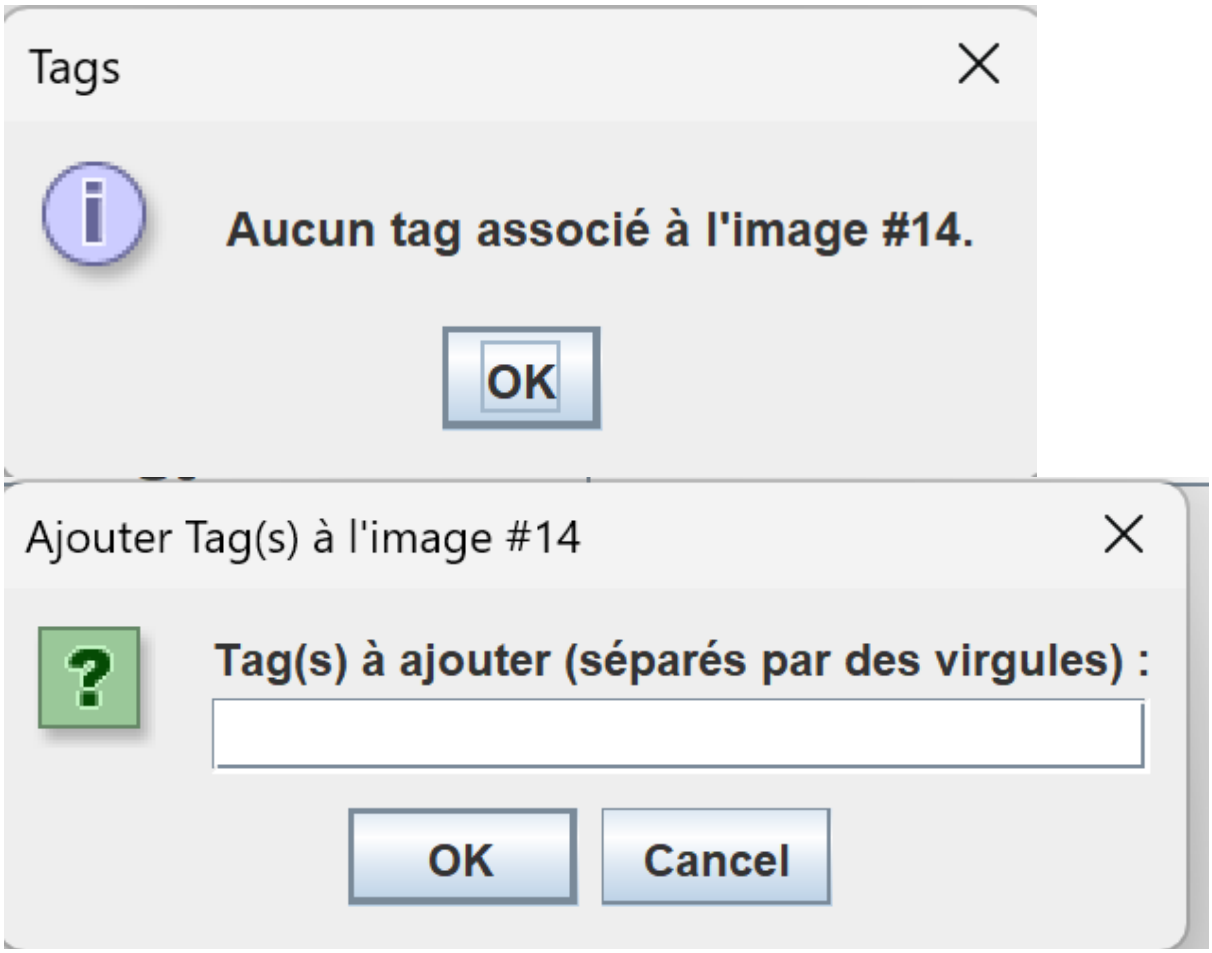

Galerie - Gestion des images
Tri
Trier par :
ID
Croissant (A→Z)
Trier
Filtre
Tag :
Filtrer par Tag
Réinitialiser
ID
Propriétaire (Email)
Visibilité
Tags
9 77777@gmail.com Privé
10 77777@gmail.com Privé
11 77777@gmail.com Public
13 77777@gmail.com Privé
14 poconaut@gmail.com Privé
15 poconaut@gmail.com Public
Images
Ajouter
Voir
Modifier
Supprimer
Tags
Ajouter Tag
Voir Tags
Supprimer Tag
Fermer

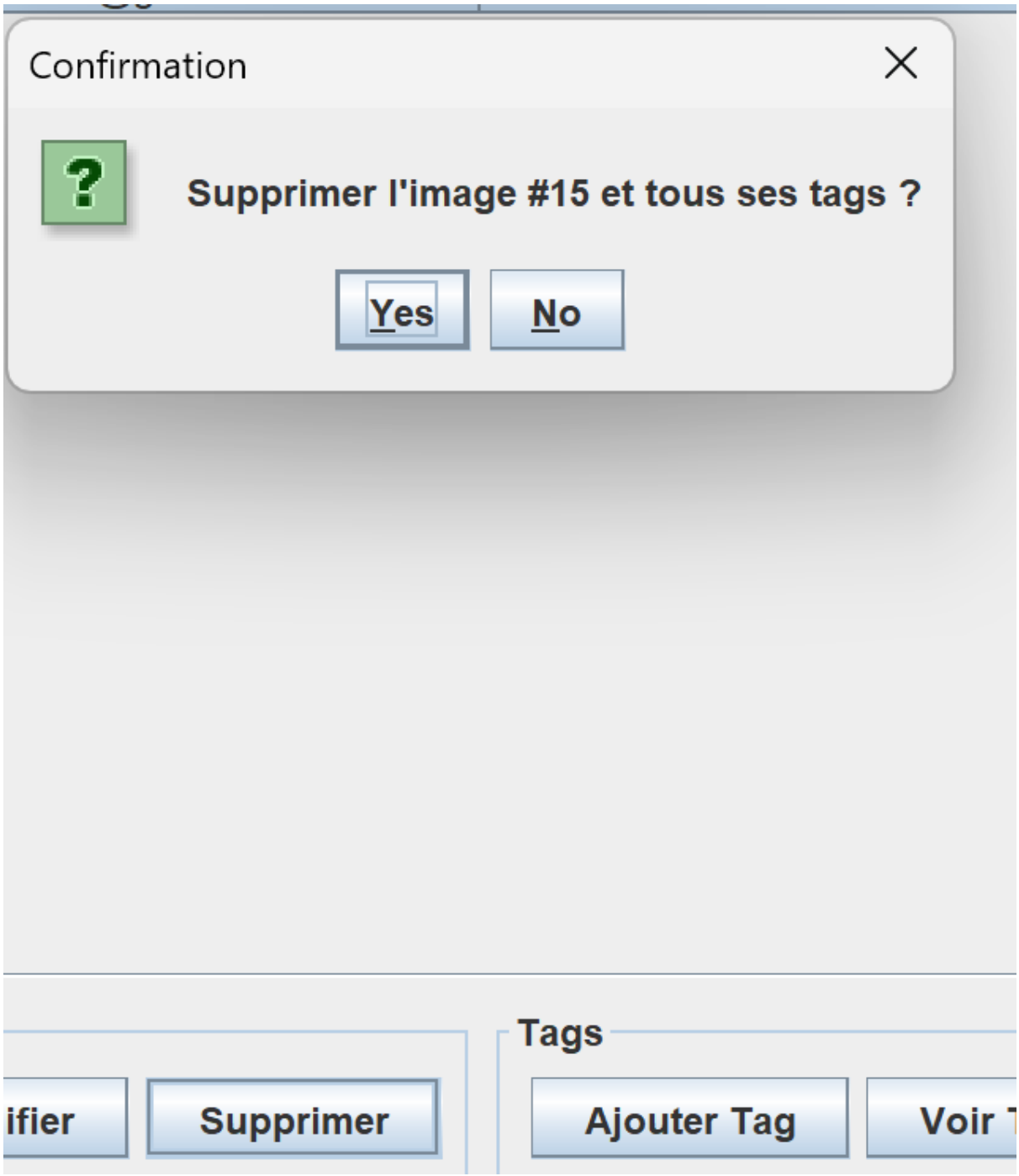

Galerie
Gérer les images

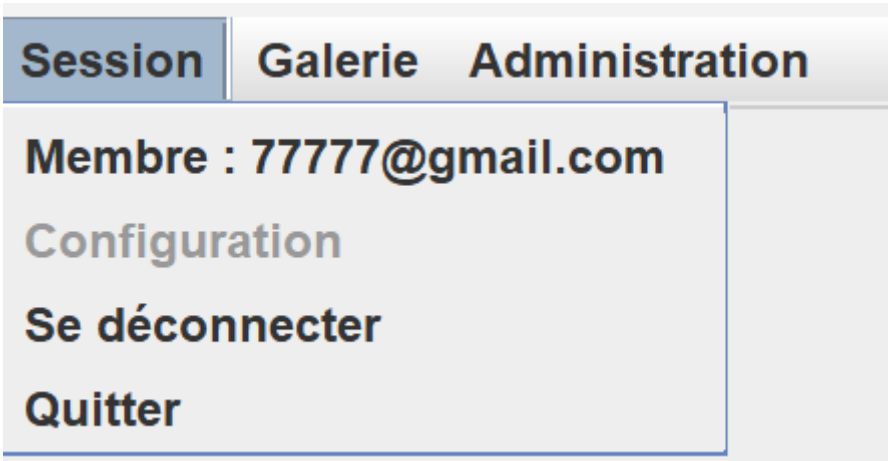

Lister les membres
ID :
2
Prénom :
1234
Nom :
1234
Téléphone :
1234@gmail.com
E-mail :
1234@gmail.com
M. De Passe :
Conf. Mdp :
Rôle :
artiste
Supprimer
Fermer
M.A.J
Liste Utilisateurs :
77777, 77777
1234, 1234
Rollé, Tanguy
huhuhuphp, huhuhuphp
1, 1

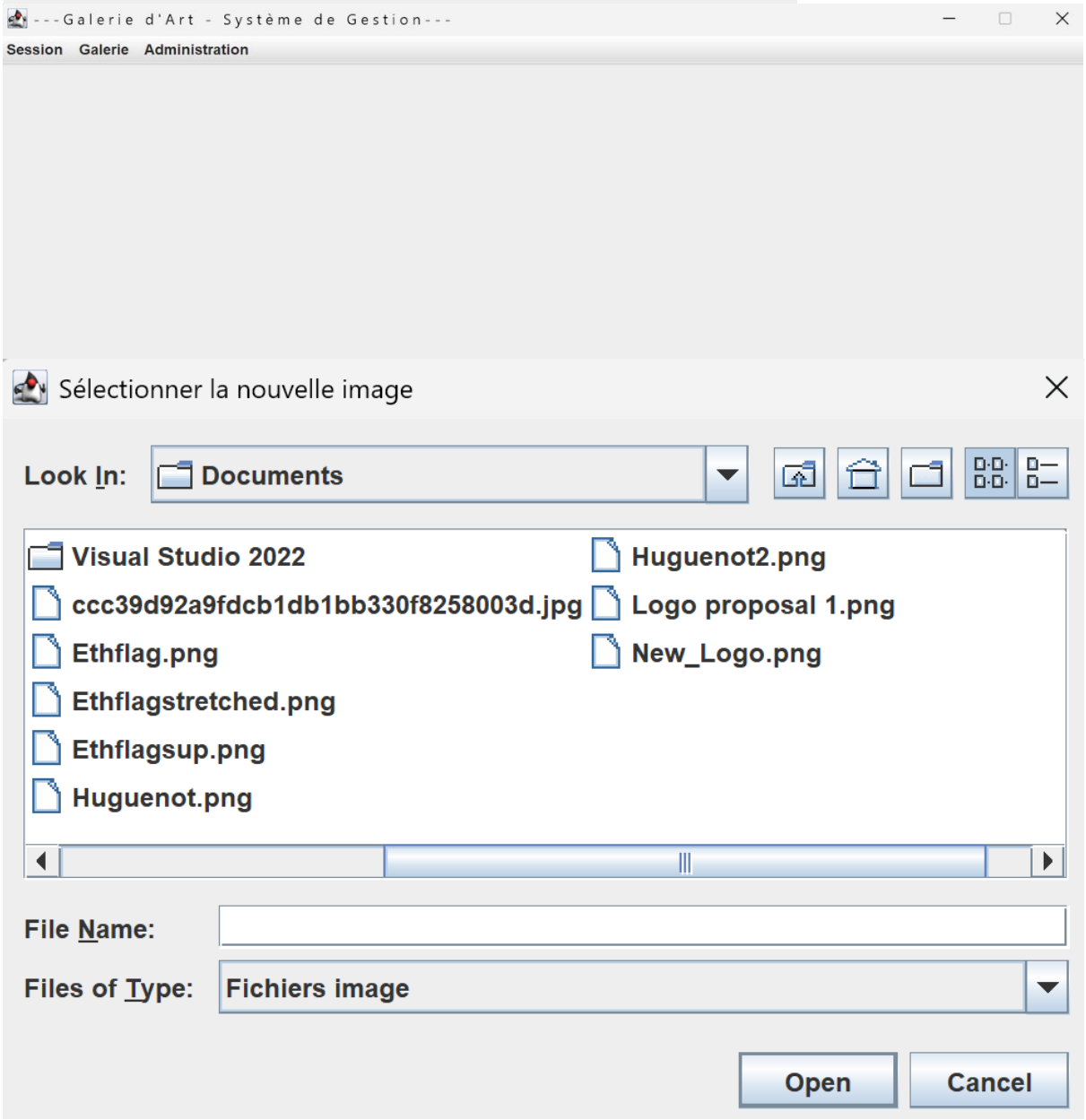

Lister les membres
ID :
Prénom :
Nom :
Téléphone :
E-mail :
M. De Passe :
Conf. Mdp :
Rôle :
admin
Supprimer
Fermer
M.A.J
Liste Utilisateurs :
77777, 77777
1234, 1234
Rollé, Tanguy
huhuhuphp, huhuhuphp
1, 1

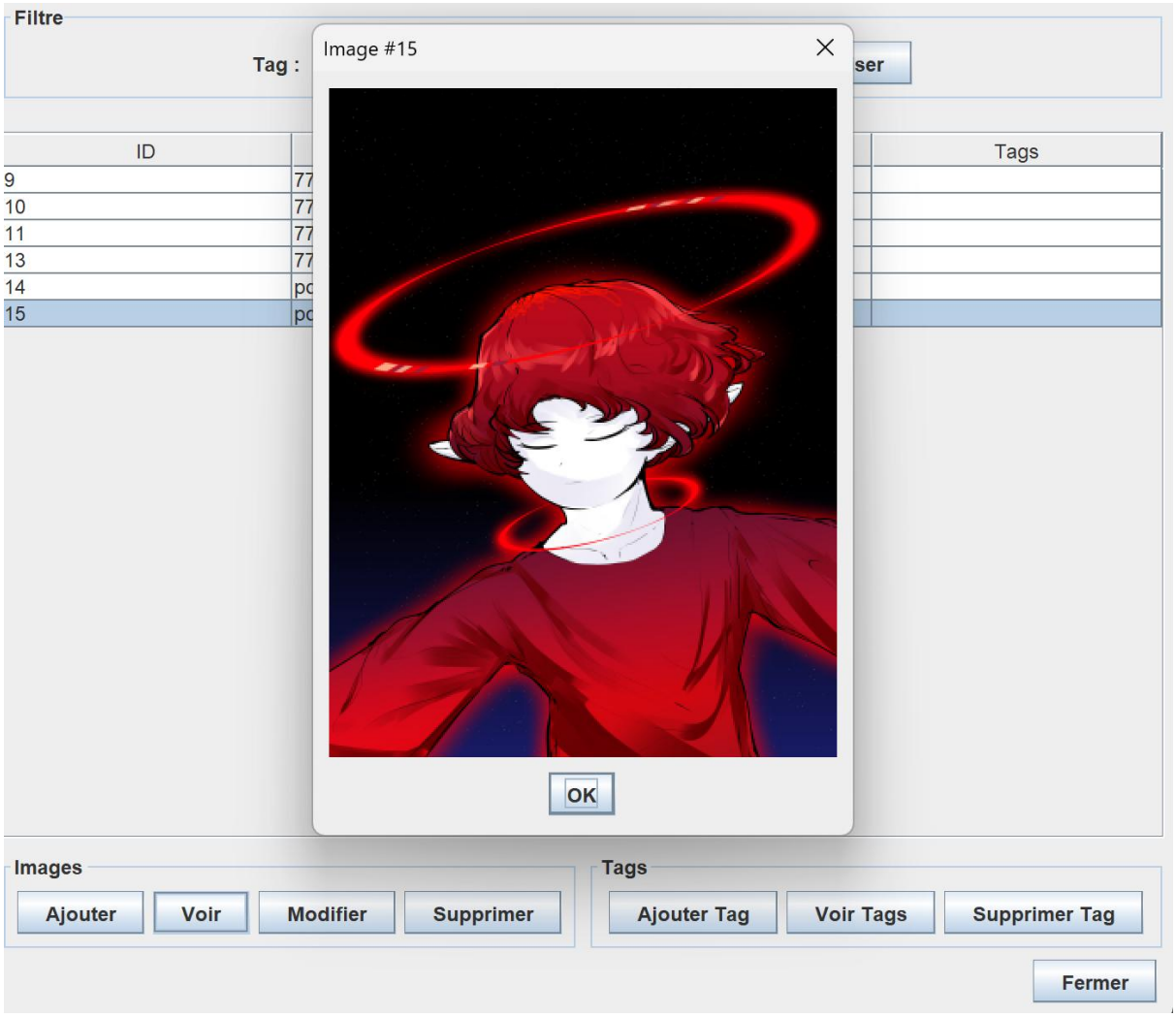

Créer un membre
Prénom :
Nom :
Téléphone :
E-mail :
Mot de passe :
Confirmer MdP :
Rôle :
admin
Valider
Fermer

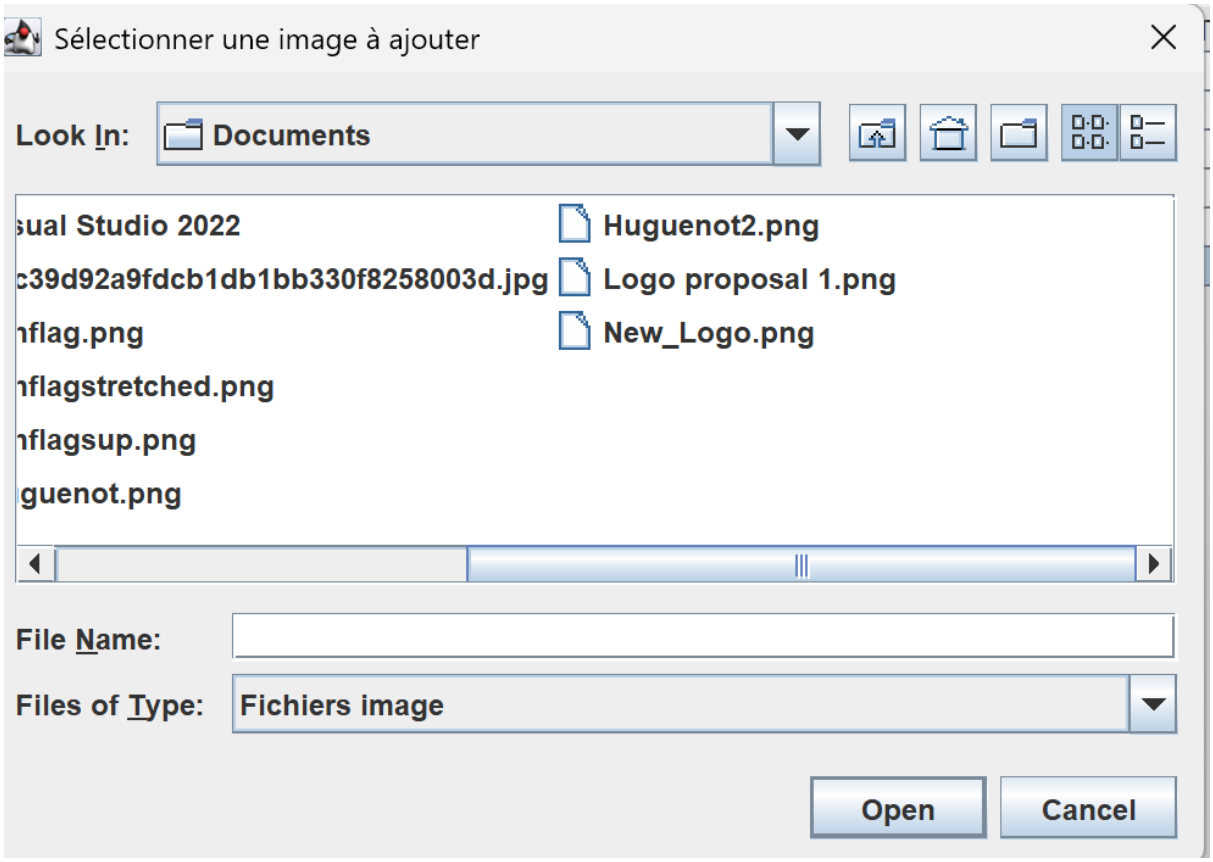

Tags
Aucun tag associé à l'image #14.
OK
Ajouter Tag(s) à l'image #14
Tag(s) à ajouter (séparés par des virgules) :
OK
Cancel

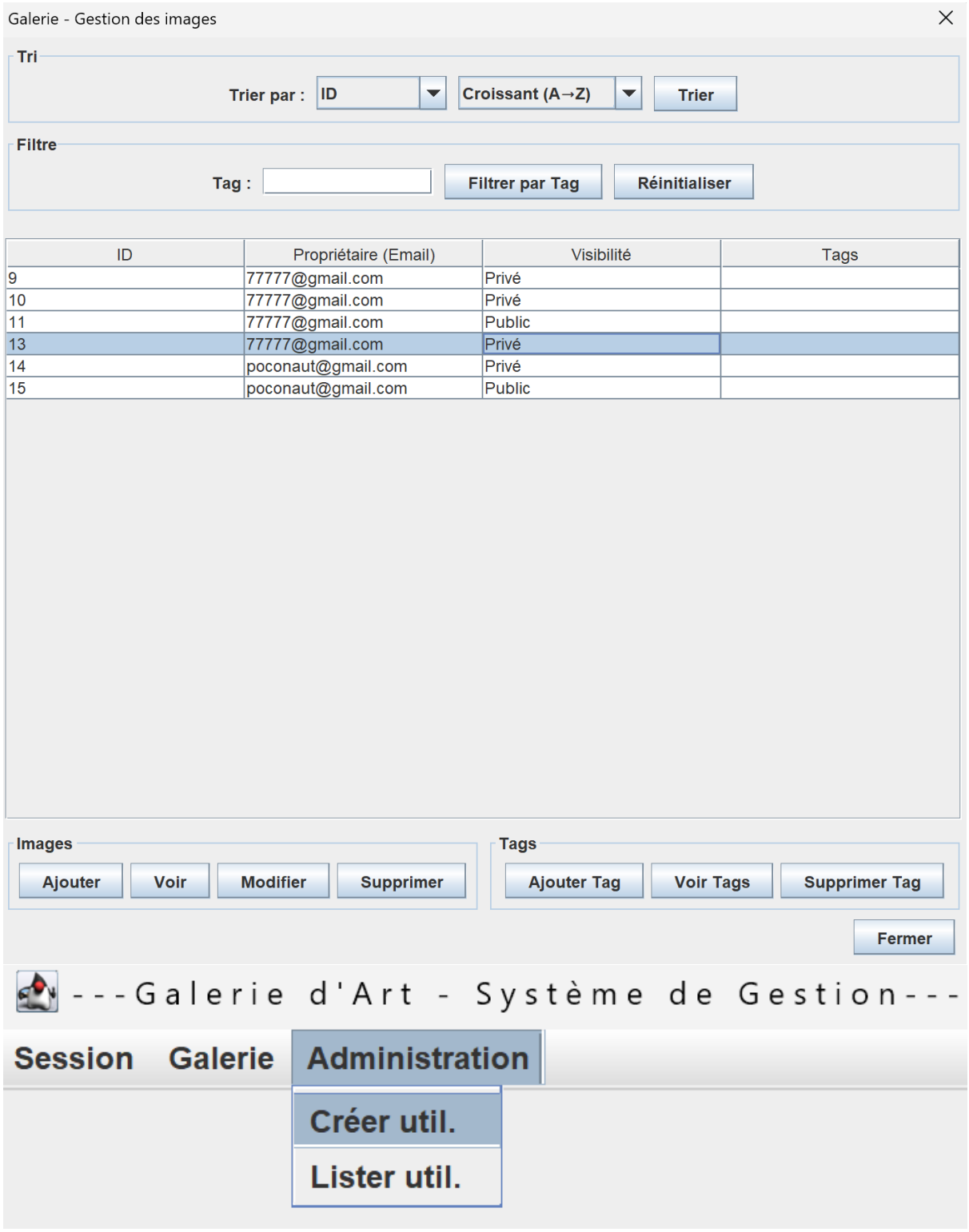

Confirmation
Supprimer l'image #15 et tous ses tags ?
Yes
No
Tags
ifier
Supprimer
Ajouter Tag
Voir

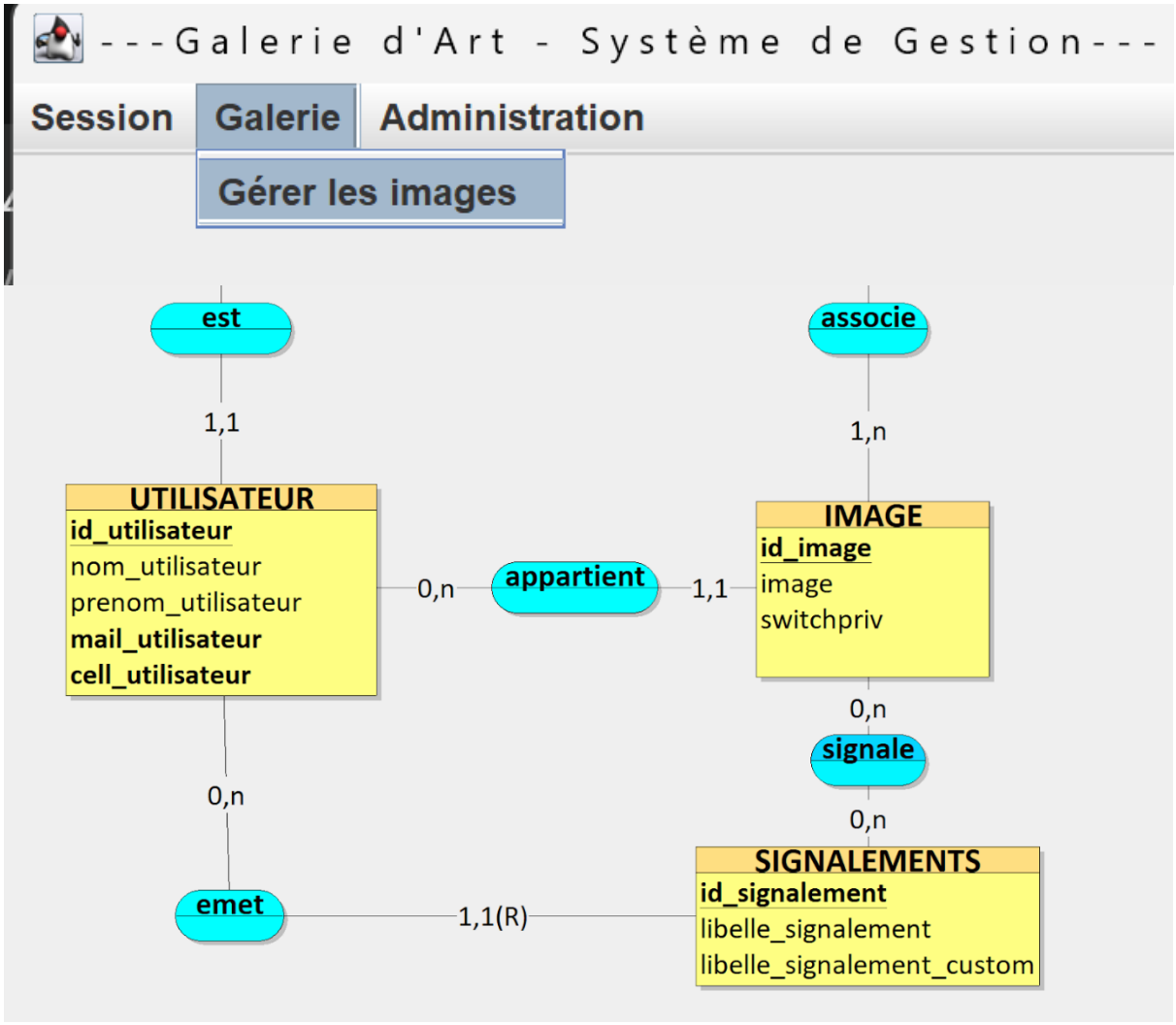

Session
Galerie
Administration
Membre : 77777@gmail.com
Configuration
Se déconnecter
Quitter


---Galerie d'Art - Système de Gestion---
Session
Galerie
Administration
Sélectionner la nouvelle image
Look In:
Documents
Visual Studio 2022
ccc39d92a9fdcb1db1bb330f8258003d.jpg
Ethflag.png
Ethflagstretched.png
Ethflagsup.png
Huguenot.png
Huguenot2.png
Logo proposal 1.png
New_Logo.png
File Name:
Files of Type:
Fichiers image
Open
Cancel


Filtre
Tag :
ser
Image #15
ID
Tags
9
10
11
13
14
15
OK
Images
Ajouter
Voir
Modifier
Supprimer
Tags
Ajouter Tag
Voir Tags
Supprimer Tag
Fermer


Sélectionner une image à ajouter
Look In:
Documents
ual Studio 2022
c39d92a9fdcb1db1bb330f8258003d.jpg
nflag.png
nflagstretched.png
nflagsup.png
guenot.png
Huguenot2.png
Logo proposal 1.png
New_Logo.png
File Name:
Files of Type:
Fichiers image
Open
Cancel


Galerie - Gestion des images
Tri
Trier par :
ID
Croissant (A→Z)
Trier
Filtre
Tag :
Filtrer par Tag
Réinitialiser
ID
Propriétaire (Email)
Visibilité
Tags
9 77777@gmail.com Privé
10 77777@gmail.com Privé
11 77777@gmail.com Public
13 77777@gmail.com Privé
14 poconaut@gmail.com Privé
15 poconaut@gmail.com Public
Images
Ajouter
Voir
Modifier
Supprimer
Tags
Ajouter Tag
Voir Tags
Supprimer Tag
Fermer
---Galerie d'Art - Système de Gestion---
Session
Galerie
Administration
Créer util.
Lister util.



MLD

UTILISATEUR(id_utilisateur, nom_utilisateur, prenom_utilisateur, mail_utilisateur, cell_utilisateur, id_role)
Clé primaire: id_utilisateur
#Clé étrangère id_role de référence ROLE

IMAGE(id_image, image, switchpriv)
Clé primaire: id_image
#Clé étrangère id_utilisateur de référence UTILISATEUR

SIGNALEMENT(id_signalement, id_utilisateur libelle_signalement, libelle_signalement_custom, id_utilisateur)
Clé primaire: id_signalement, id utilisateur
#Clé étrangère id_utilisateur de référence UTILISATEUR

TAGS(tag, id_image)
Clé primaire: tag
#Clé étrangère id_image de référence IMAGE

ROLE(id_role, libelle_role)
Clé primaire: id_role

—Relations n,n, clé étrangères exclusivement—

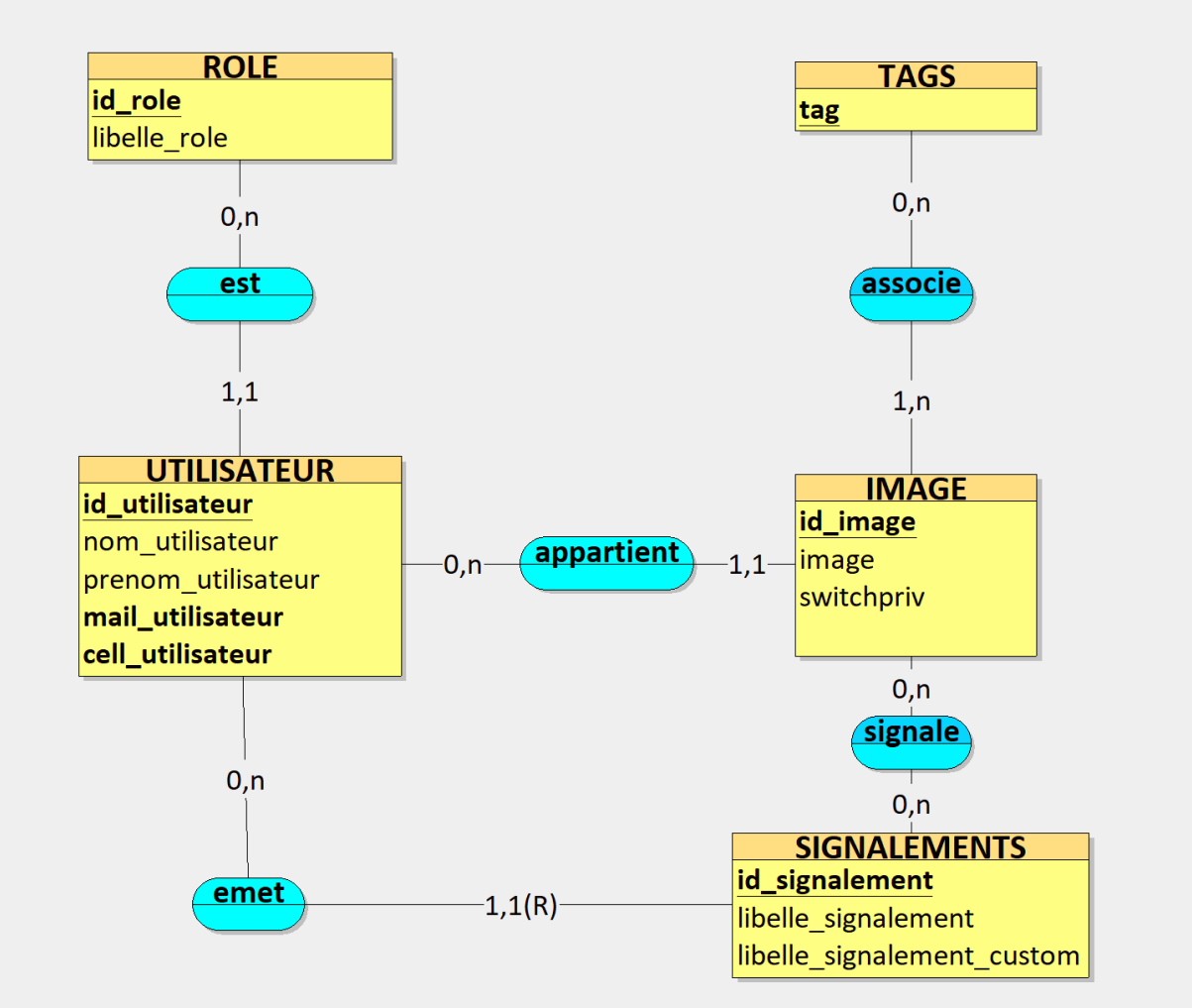
signale(id_image, id_signalement, id_utilisateur)
Clé primaire: id_image, id_signalement, id_utilisateur
#Clé étrangère id_image de référence IMAGE
#Clé étrangère id_signalement de référence SIGNALEMENT

```
CREATE TABLE ROLE (

id_role NOT NULL AUTO_INCREMENT,

libelle_role varchar(50) NOT NULL,

PRIMARY_KEY (id_role)

);


CREATE TABLE UTILISATEUR (

id_utilisateur NOT NULL AUTO_INCREMENT,

nom_utilisateur varchar(50) NOT NULL,

prenom_utilisateur varchar(50) NOT NULL,

mail_utilisateur varchar(50) NOT NULL UNIQUE,

cell_utilisateur varchar(15) UNIQUE,

id_role NOT NULL AUTO_INCREMENT,

PRIMARY KEY (id_utilisateur),
FOREIGN KEY (id_role) REFERENCES ROLE (id_role)

);


CREATE TABLE IMAGE (

id_image NOT NULL AUTO_INCREMENT,

image LONGBLOB,

switchpriv BOOLEAN,

id_utilisateur NOT NULL AUTO_INCREMENT,
```

```
PRIMARY KEY (id_image),
FOREIGN KEY (id_utilisateur) REFERENCES UTILISATEUR (id_utilisateur)

);


CREATE TABLE SIGNALEMENT (

id_signalement NOT NULL AUTO_INCREMENT,

id_utilisateur NOT NULL AUTO_INCREMENT,

libelle_signalement enum('violence', 'haine', 'contenu_sexuel', 'contenu_illegal', 'plagiat', 'autre')
NOT NULL,

libelle_signalement_custom varchar(250),

PRIMARY KEY (id_signalement, id_utilisateur),
FOREIGN KEY (id_utilisateur) REFERENCES UTILISATEUR (id_utilisateur)

);

CREATE TABLE TAGS (

tag varchar(100) NOT NULL UNIQUE,

id_image NOT NULL AUTO_INCREMENT,

PRIMARY KEY (tag),
FOREIGN KEY (id_image) REFERENCES IMAGE (id_image)

);


CREATE TABLE signale(

id_image NOT NULL AUTO_INCREMENT,

id_signalement NOT NULL AUTO_INCREMENT,

id_utilisateur NOT NULL AUTO_INCREMENT,

PRIMARY KEY (id_image, id_signalement, id_utilisateur),
FOREIGN KEY (id_image) REFERENCES IMAGE (id_image),
FOREIGN KEY (id_signalement) REFERENCES IMAGE (id_signalement),
FOREIGN KEY (id_utilisateur) REFERENCES IMAGE (id_utilisateur)
```

);

A noter qu'une erreur conceptuelle fut repérée vers la fin du projet, concernant la gestion des tags d'images. La cardinalité 1,1 de tags à image fut changée en une relation 0, n et une nouvelle association fut ajoutée dans le MCD, MLD et le script: soit:



//
TAGS(tag, id_image)
Clé primaire: tag

associe(id_image, tag)
Clé primaire: id_image, tag
#Clé étrangère id_image de référence IMAGE
#Clé étrangère tags de référence TAGS

```
//

CREATE TABLE TAGS (

tag varchar(100) UNIQUE,

PRIMARY KEY (tag)

);

CREATE TABLE ASSOCIE (

id_image INT,

tag varchar(100) UNIQUE,

FOREIGN KEY (id_image) REFERENCES IMAGE(id_image),
FOREIGN KEY (tag) REFERENCES TAGS(tag),
PRIMARY KEY (id_image, tag)

);
```