**TP JPA 2-4 | Branche :**

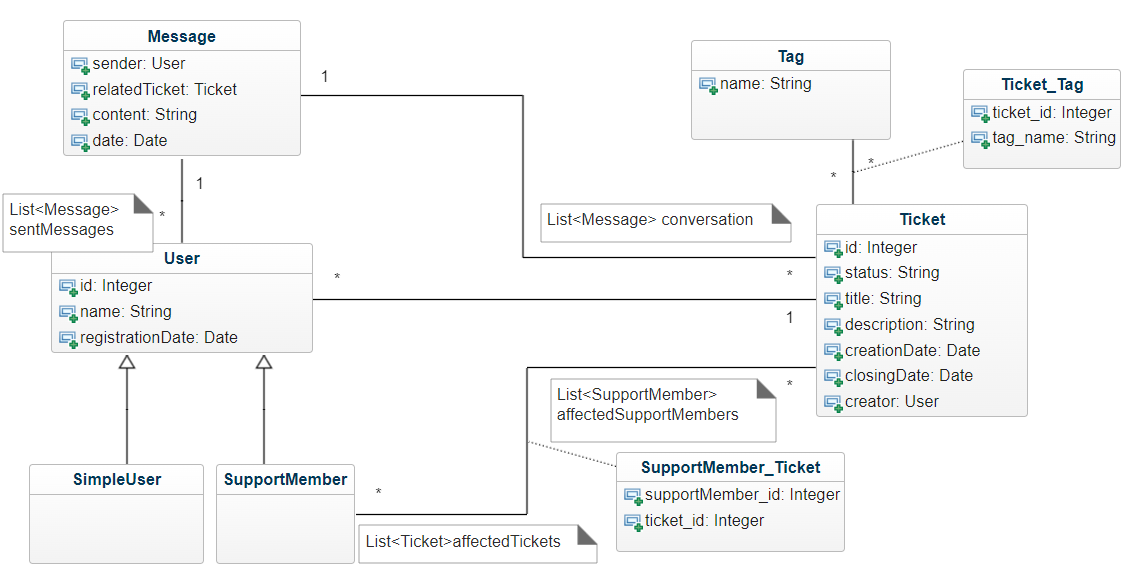
Nos tests sont effectués en local. En utilisant Docker, après les commandes :

docker run --name mydatabase -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:latest

docker run --name myadmin -d --link mydatabase:db -p 8082:80 phpmyadmin

On a alors accès à une base de données phpmyadmin via http://localhost:8082 , avec pour username « root » et en mot de passe « my-secret-pw ». Ensuite, il faut créer une nouvelle base de données nommée « mydatabase ».

**Questions 1 et 2 :**



Le diagramme ci-dessus représente les classes et les entités. Les attributs de chaque classe correspondent à la fois à un attribut de classe et à une colonne de l’entité dans la base de données.

Les associations marquées « \* » d’un côté correspond à l’existence d’une liste, en attribut de la classe métier. Le nom de cette liste est marqué en commentaire, à côté de l’entité, sur le diagramme.

On distingue 2 types de User : les « SimpleUser » et les « SupportMember ». Ce sont deux classes qui héritent de la classe « User ». D’un point de vue base de données, nous avons fait le choix d’avoir une seule table, avec une colonne « Dtype » qui prend la valeur « Simple User » ou « Support Member ». Cette représentation est très rudimentaire, le but de ce TP n’étant pas de créer un système de tickets complexe, mais plutôt d’appréhender les différentes possibilités proposées par JPA, ainsi que le lien effectué entre les classes métiers, et les tables de la base de données.

Les « User » peuvent créer des « Ticket ». Ainsi, chaque ticket a un unique créateur.

Un ticket peut être affecté à plusieurs « Support Member » qui sont chargés de traiter le ticket.

Un ticket peut avoir plusieurs « Tag », qui permettent de mieux catégoriser les tickets.

A propos d’un ticket, les « User » peuvent échanger des messages. Ainsi, un message a un envoyeur, et est lié à un ticket.

**Question 3 :**

Nous avons créé des classes DAO, une par classe métier, qui contiennent les requêtes.

Dans le projet, exécuter « JpaTest » appellera alors 3 fonctions qui permettent de peupler la base de données :

fillDatabaseWithNormalUsers() ;

fillDatabaseWithSupportMembers();

fillDatabaseWithTicketsMessagesTags();

Voici l’état de la base de données après l’exécution du Main :



Figure : Tables de la Base de Données



Figure : Table "Ticket"

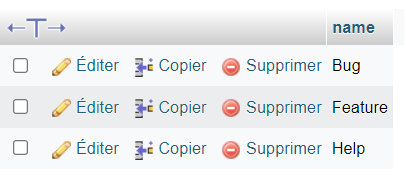


Figure : Table "Tag"

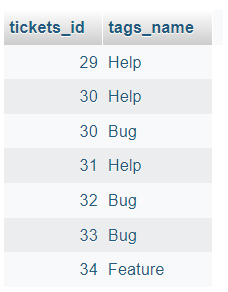


Figure : Table d'association entre "Ticket" et "Tag" (« Tickets\_Tags »).

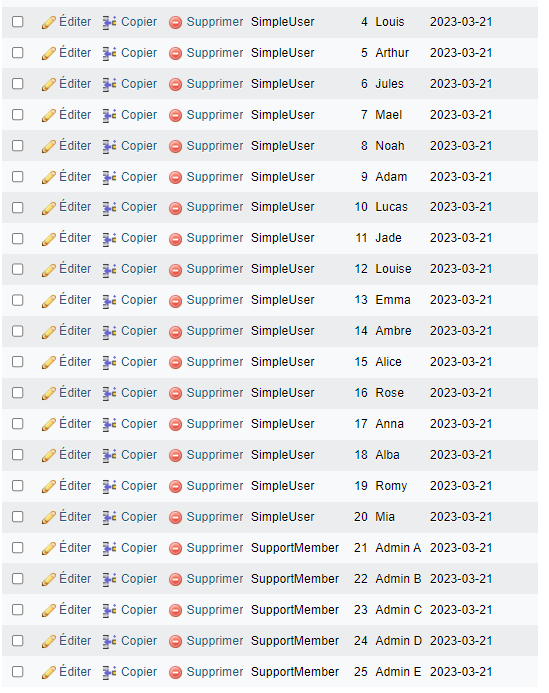


Figure : Table "User"

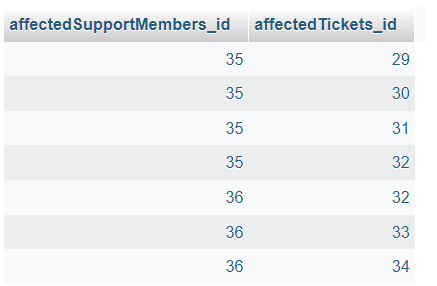


Figure : Table d'association entre "Ticket" et "Support Member" : « Support\_Tickets »

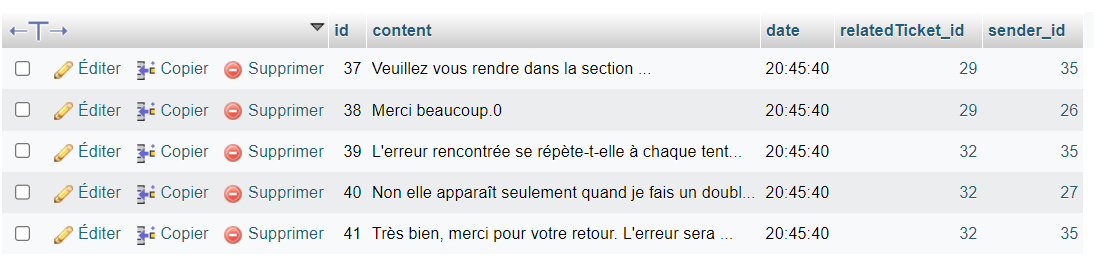


Figure : Table "Message"

La fonction « requestTest », appelée dans le Main, donne dans la console les résultats de différentes requêtes.

**Question 5 :**

La relation d’héritage entre « User », « SimpleUser » et « SupportMember » a été décrite plus tôt. Nous avons choisi de représenter cette relation sous une seule table, car dans notre modèle très simplifié, il n’y a pas d’attributs supplémentaires dans les classes enfants, et nous n’avons alors même pas l’inconvénient d’avoir de nombreuses colonnes ayant la valeur « null ».

Étant donné que nous utilisons le pattern DAO, nous n’avons pas vraiment besoin d’utiliser des requêtes nommées. Un exemple de requête nommée très simple se situe dans la classe/entité « User ».