# Introduction :

Dans notre projet, j'ai développé une application de chat en utilisant l'architecture MVC (Modèle-Vue-Contrôleur). J'ai utilisé la base de données Oracle HR pour gérer les informations des utilisateurs. L'application propose des fonctionnalités telles que la création de comptes, la recherche d'utilisateurs, la messagerie instantanée et la suppression de comptes.

# Description du projet :

Le projet de chat est une application de messagerie permettant aux utilisateurs de communiquer entre eux. L'objectif principal du projet est de fournir une plateforme conviviale et interactive pour les conversations en ligne.

Le projet repose sur une architecture MVC (Modèle-Vue-Contrôleur) qui permet une séparation claire des responsabilités. Le modèle gère les données, la vue affiche l'interface utilisateur et le contrôleur gère les interactions entre les utilisateurs et les données.

La base de données utilisée est Oracle HR, qui stocke les informations des utilisateurs, les conversations et les messages. Les données sont accessibles via des Data Access Objects (DAOs), qui fournissent une interface pour interagir avec la base de données et effectuer des opérations CRUD (Création, Lecture, Suppression).

L'application offre des fonctionnalités telles que la recherche d'utilisateurs, la sélection de conversations, la création des utilisateurs, l'affichage des messages et l'envoi de nouveaux messages. Les utilisateurs peuvent interagir avec l'interface à l'aide de boutons et de zones de texte.

# Spécification des besoins

# Besoins fonctionnels :

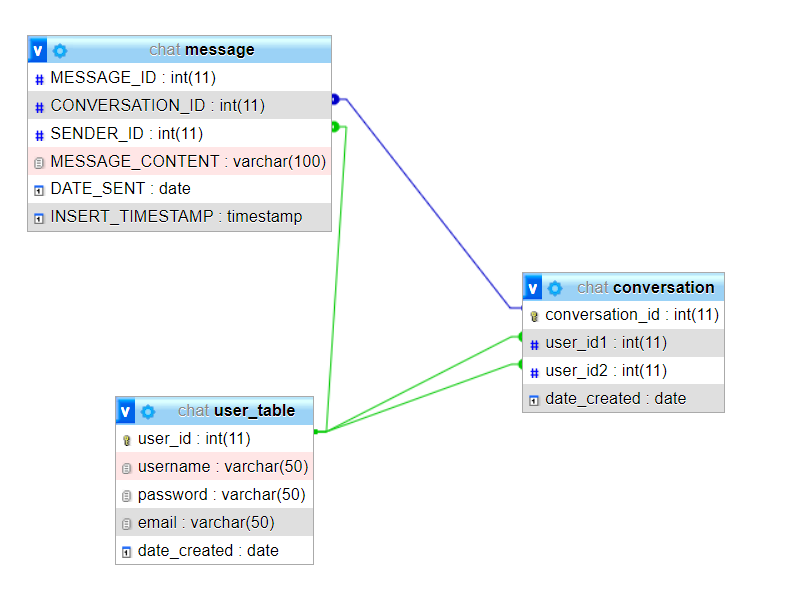
* Création d'un compte utilisateur : Les utilisateurs doivent pouvoir créer un compte pour accéder au chat.
* Connexion au chat : Les utilisateurs doivent pouvoir se connecter au chat avec leurs identifiants.
* Affichage de la liste des utilisateurs : Les utilisateurs connectés doivent pouvoir voir la liste des autres utilisateurs disponibles pour discuter.
* Création d'une conversation : Les utilisateurs doivent pouvoir initier une nouvelle conversation avec un autre utilisateur.
* Envoi de messages : Les utilisateurs doivent pouvoir envoyer des messages dans une conversation en cours.
* Réception de messages : Les utilisateurs doivent pouvoir recevoir et afficher les messages reçus dans une conversation en cours.
* Suppression de conversations : Les utilisateurs doivent pouvoir supprimer des conversations existantes.

# Besoins non fonctionnels :

* Performance : Le système doit être réactif et fournir des temps de réponse rapides lors de l'envoi et de la réception de messages.
* Sécurité : Les informations des utilisateurs et les échanges de messages doivent être sécurisés pour prévenir les accès non autorisés.
* Évolutivité : Le système doit être conçu de manière à pouvoir gérer un grand nombre d'utilisateurs et de conversations simultanément.
* Convivialité : L'interface utilisateur doit être conviviale et intuitive, facilitant la navigation et l'utilisation du chat.
* Disponibilité : Le chat doit être disponible et accessible aux utilisateurs de manière fiable, avec un temps d'arrêt minimal.

# Shema de base de données :

Le schéma de base de données comprend trois tables principales : "User\_table" (table des utilisateurs), "Conversation" (table des conversations) et "Message" (table des messages). Ces tables sont interconnectées par des clés étrangères pour établir des relations et maintenir l'intégrité des données.

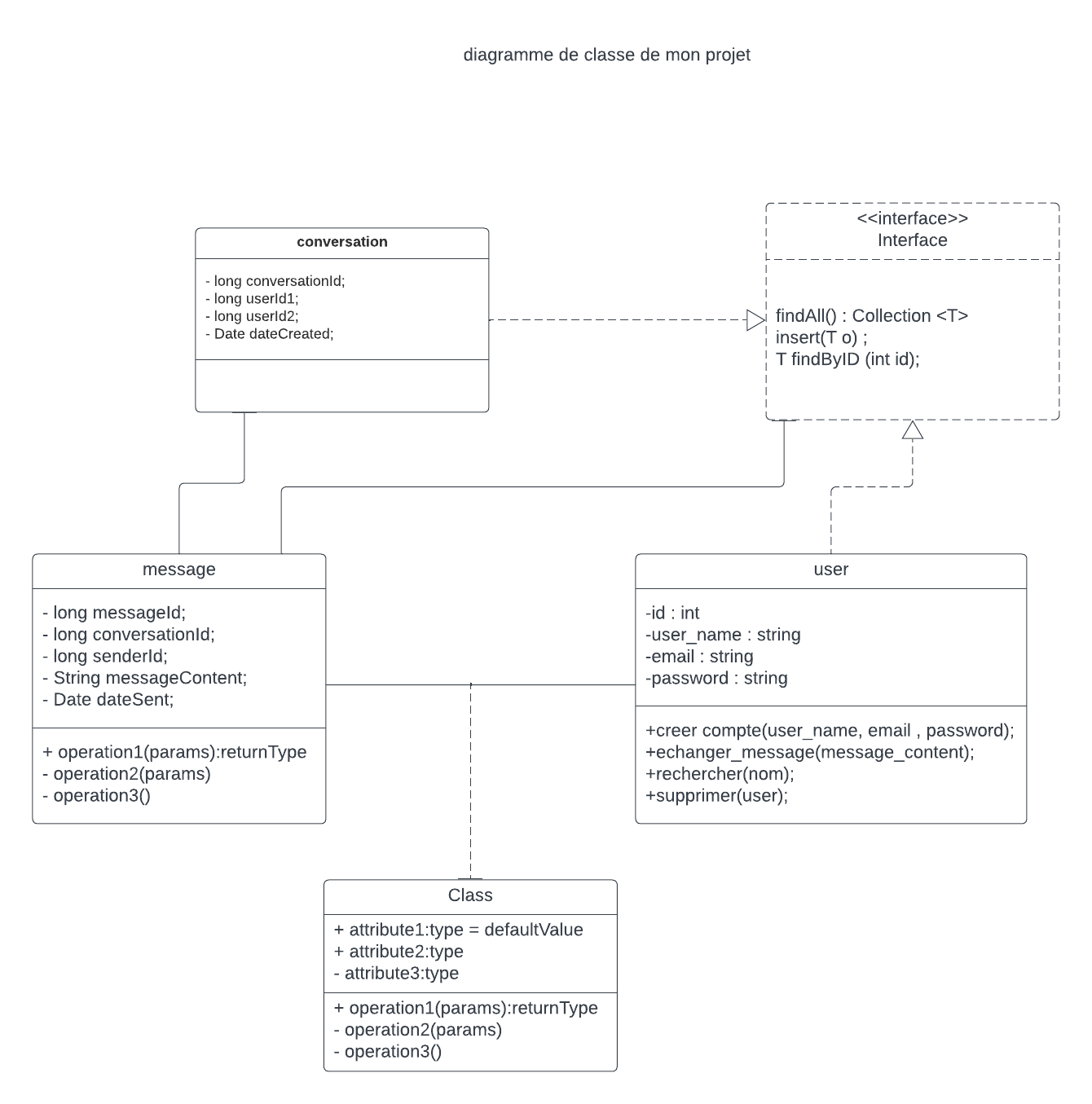


# Diagramme de classe :

j'ai utilisé un diagramme de classe pour représenter la structure et les relations entre les différentes entités de notre système. Le diagramme de classe offre une vue d'ensemble du projet, en identifiant les classes principales, leurs attributs et leurs relations.

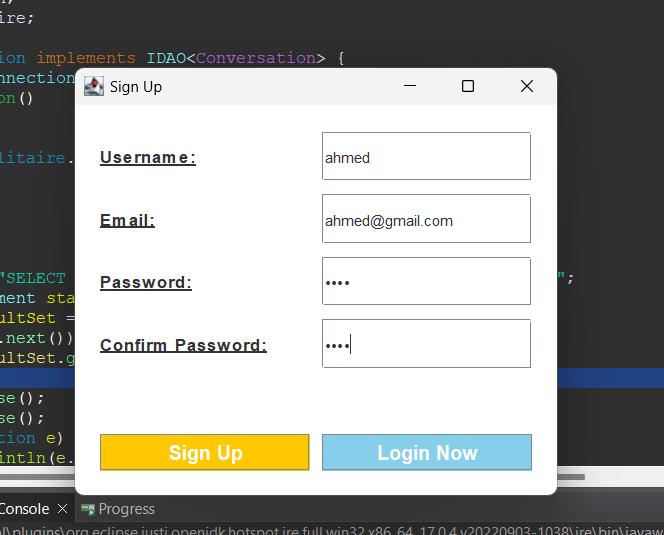
Le diagramme de classe est basé sur le modèle MVC (Modèle-Vue-Contrôleur) qui permet de séparer la logique métier, la présentation et la gestion des données. Cette approche favorise la modularité, la réutilisabilité et la maintenabilité du code.

Le diagramme de classe inclut trois classes principales: "User" (utilisateur), "Conversation" (conversation) et "Message" (message). La classe "User" représente les utilisateurs du système avec leurs attributs tels que l'identifiant, le nom d'utilisateur, le mot de passe, l'adresse e-mail, etc. La classe "Conversation" représente les conversations entre les utilisateurs et contient des références aux utilisateurs participants. Enfin, la classe "Message" représente les messages échangés dans une conversation avec des informations telles que l'identifiant du message, l'identifiant de la conversation, l'expéditeur, le contenu du message et les horodatages.

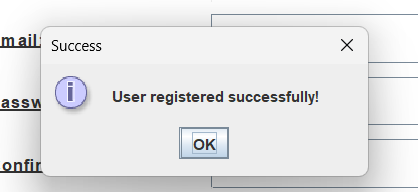


# Réalisations :

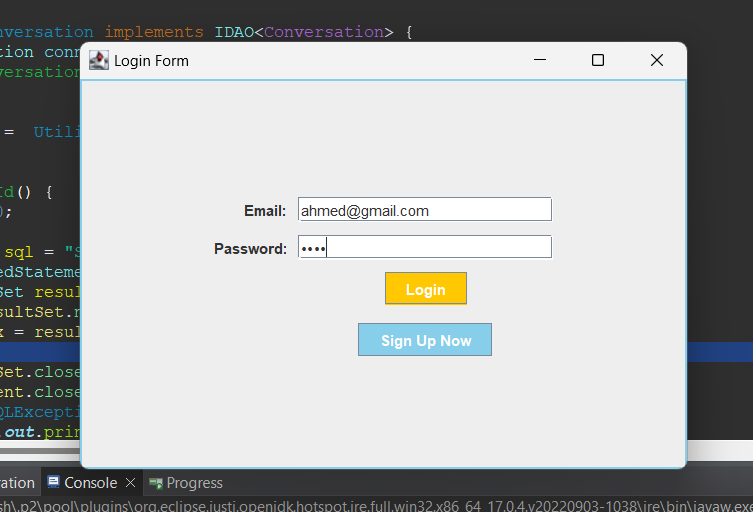
* Creation d’un nouvel utilisateur :



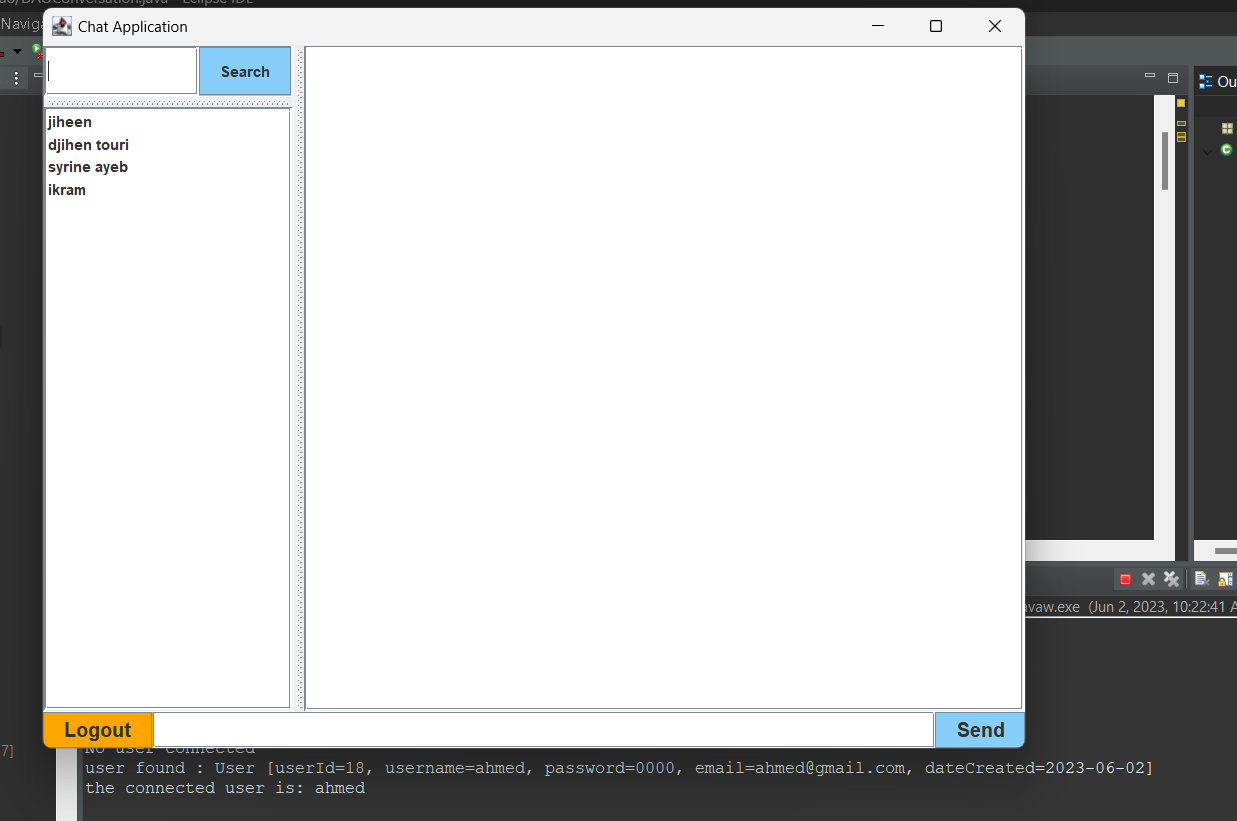
* La validation de la création :



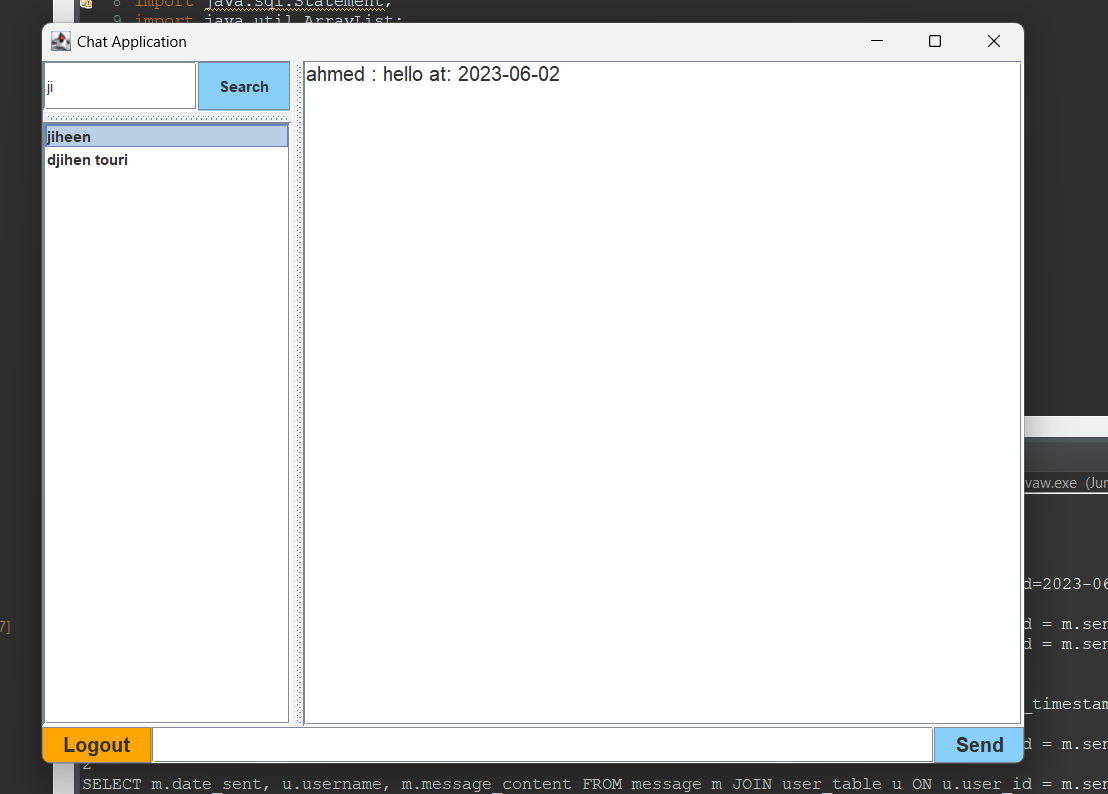
* La connection avec ce compte :



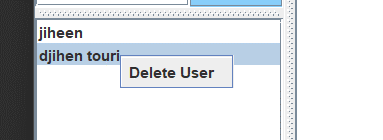
* L’interface d’où on peut consulter tous les utilisateurs et la liste des messages



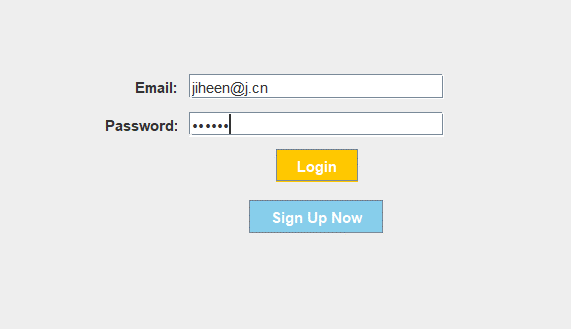
* Tester la recherche et l’envoie d’un message :



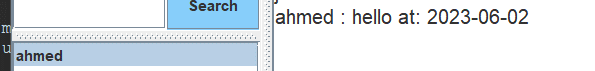
* Tester la supression d’un message :



* Ouvrir le compte destinataire :



* Verification de la reciption du message :



# Conclusion :

En conclusion, dans notre projet de chat, nous avons mis en œuvre l'architecture MVC (Modèle-Vue-Contrôleur) combinée avec le design pattern DAO (Data Access Object). Cette combinaison a permis d'organiser efficacement notre code et de séparer les différentes responsabilités de notre application.