# **Documentation Technique**

# **Introduction**

Cette documentation présente les classes DAOImpl, qui sont responsables de la gestion des opérations CRUD (Create, Read, Update, Delete) pour les objets de notre système.

# **Classes et Leur Fonctionnalité**

## **Classe Cours**

La classe **Cours** représente un cours offert par l'établissement scolaire. Elle permet de gérer les informations sur les cours, telles que le code et le libellé, et offre des méthodes pour créer, mettre à jour, rechercher et supprimer des cours dans la base de données.

## **Classe Etudiant**

La classe **Etudiant** représente un étudiant inscrit à l'établissement scolaire. Elle permet de gérer les informations personnelles des étudiants, telles que leur nom, prénom, date de naissance et classe. Cette classe offre des méthodes pour créer, rechercher, mettre à jour et supprimer des étudiants dans la base de données.

## **Classe Inscription**

La classe **Inscription** gère les inscriptions des étudiants à des cours spécifiques. Elle permet de gérer les informations sur les inscriptions, telles que l'identifiant de l'inscription, l'identifiant de l'étudiant, le code du cours et la date d'inscription. Cette classe offre des méthodes pour créer, rechercher, mettre à jour et supprimer des inscriptions, ainsi que pour récupérer les inscriptions par code de cours ou par identifiant d'étudiant.

## **Classe User**

La classe **User** est utilisée pour gérer les utilisateurs du système. Elle permet de gérer les informations sur les utilisateurs, telles que leur nom, mot de passe haché et rôle. Cette classe offre des méthodes pour ajouter un utilisateur, rechercher un utilisateur par nom et vérifier l'authentification en comparant les mots de passe hachés.

## **Classe DAOImpl**

### **Méthode insert**

* Description : Cette méthode permet d'ajouter un nouvel objet à notre base de données. Les détails de l'objet sont spécifiés dans l'objet correspondant fourni en tant que paramètre. Si l'opération réussit, elle renvoie **void**.
* Signature : **public void insert(Objet objet)**

### **Méthode findById**

* Description : Utilisée pour rechercher un objet dans notre base de données en se basant sur son identifiant unique (ID). Si l'objet est trouvé, un objet correspondant est renvoyé. Dans le cas contraire, **null** est renvoyé.
* Signature : **public Objet findById(Long id)**

### **Méthode findAll**

* Description : Cette méthode permet de récupérer tous les objets enregistrés dans notre base de données. Elle renvoie une liste contenant tous les objets.
* Signature : **public List<Objet> findAll()**

### **Méthode update**

* Description : Utilisée pour mettre à jour les informations d'un objet existant dans la base de données. Les nouvelles informations sont spécifiées dans l'objet correspondant fourni en tant que paramètre.
* Signature : **public void update(Objet objet)**

### **Méthode delete**

* Description : Cette méthode permet de supprimer un objet de notre base de données en se basant sur son identifiant unique (ID).
* Signature : **public void delete(Long id)**

## **Gestion des Exceptions**

Toutes ces classes gèrent les exceptions en affichant les traces d'erreur en cas d'échec des opérations de base de données.

## **Sécurité des Mots de Passe**

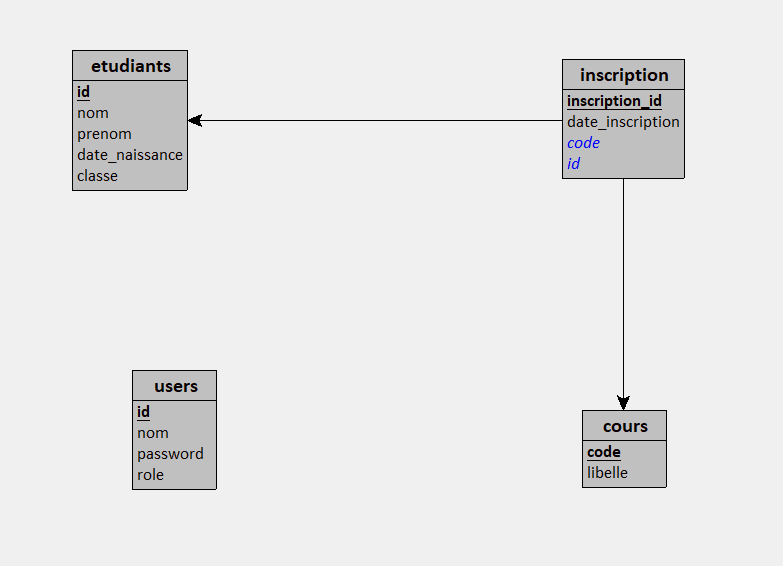
Pour la classe **UserDAOImpl**, les mots de passe sont stockés de manière sécurisée en utilisant la bibliothèque BCrypt.

# **Relations entre les Tables**

La base de données est conçue de manière à ce que les différentes tables interagissent entre elles pour représenter les liens essentiels dans notre système. Voici un aperçu des relations clés entre ces tables :

* **Table etudiants et Table cours :** Ces deux tables sont connectées par l'intermédiaire de la table **inscription**. La table **inscription** joue un rôle crucial en enregistrant les inscriptions des étudiants à des cours spécifiques. Chaque enregistrement dans la table **inscription** pointe vers un étudiant dans la table **etudiants** et vers un cours dans la table **cours**, grâce aux colonnes **etudiant\_id** et **cours\_code**.
* **Table users :** Bien qu'elle ne soit pas directement liée aux autres tables, la table **users** est essentielle pour gérer les utilisateurs de notre système. Elle conserve des informations sur les utilisateurs, telles que leurs noms, mots de passe hachés et rôles. Cette table assure la sécurité et l'authentification des utilisateurs lors de leur interaction avec les fonctionnalités du système.

MLD



UML

