

# RAPPORT DE PROJET

APPLICATION D'AFFECTION  
DE SALLES DE TP



**Préparé par,**

Rayen Akriche

Mohamed Amine Hajji

Mootaz Hamam

Hend Jacob

**Encadré par:**

-Houda Alaya

-Saloua Zammali

-Hajer dammak

-Marwen sakli

# Sommaire

## Introduction

PAGES 03-06

## Analyse des besoins

PAGES 07-13

## Analyse de domaine et conception

PAGES 14-21

## Realisation

PAGES 22-48

## Les interfaces

PAGES 49-63

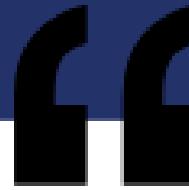


01

# INTRODUCTION

# 01

## INTRODUCTION



### Le contexte

La gestion des salles de travaux pratiques (TP) dans un établissement d'enseignement est un processus complexe, impliquant l'affectation dynamique des ressources en fonction des besoins pédagogiques, des contraintes techniques et des disponibilités. Actuellement, les méthodes manuelles ou semi-automatisées entraînent des problèmes récurrents : conflits de réservation, difficultés à respecter les plages de maintenance, suivi imprécis des affectations, et temps perdu dans la recherche d'informations. Ces défis impactent directement l'efficacité des enseignants et l'optimisation des infrastructures.

Notre projet vise à résoudre ces problèmes grâce à une application dédiée à l'affectation intelligente des salles de TP. Cette solution centralisera toutes les données clés (salles, enseignants, créneaux horaires) et automatisera les processus critiques, tout en garantissant le respect des contraintes telles que :

Le nombre maximal d'affectations hebdomadaires par salle.

Les jours de maintenance ou de nettoyage (indisponibilité fixe).

La capacité d'accueil des salles en fonction du nombre d'étudiants.

L'application proposera deux niveaux d'accès :

Un espace administrateur pour gérer le référentiel (ajout/modification/suppression des salles et enseignants) et superviser les statistiques (ex : salle la plus utilisée).

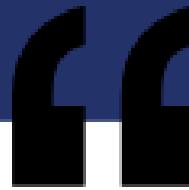
Un espace utilisateur pour les enseignants ou planificateurs, permettant de consulter les disponibilités, réserver des créneaux, et visualiser les emplois du temps en temps réel.

Grâce à des fonctionnalités avancées comme une procédure PL/SQL de vérification de disponibilité, l'outil garantira une allocation optimale des salles, évitera les chevauchements, et réduira les erreurs humaines. Il s'adaptera aux exigences académiques (ex : pics de réservation en période de projets) et améliorera la transparence pour tous les acteurs.

En remplaçant les systèmes obsolètes par une solution numérique robuste, cette application deviendra un pilier de l'organisation pédagogique, favorisant une utilisation équilibrée des ressources et une expérience simplifiée pour les enseignants et les gestionnaires.

# 01

## INTRODUCTION



### Besoins fonctionnels

#### 1/ Authentification :

Accès sécurisé via login/mot de passe.

#### 2/ Rôles : Administrateur (gestion complète) / Enseignant (consultation).

#### 3/ Recherche :

Recherche de salles, enseignants ou réservations par mot-clé.

#### 3/ Gestion des Utilisateurs :

Administrateur : Ajout/modification/suppression de comptes (enseignants, planificateurs).

#### 4/ Gestion des Enseignants :

Ajout/modification/suppression des enseignants (spécialité, disponibilités).

#### 5/ Gestion des Contraintes :

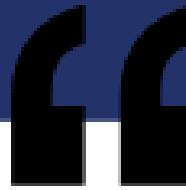
Définir jours de maintenance, nombre max d'affectations/semaine, capacité des salles.

#### 6/ Gestion des Salles :

Ajout/modification/suppression des salles (nom, capacité, équipements).

# 01

## INTRODUCTION



### Besoins non fonctionnels

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l'utilisateur, ce qui fait qu'ils ne doivent pas être négligés. Pour cela, il faut répondre aux exigences suivantes :

#### Fiabilité :

L'application doit fonctionner de façon cohérente sans erreurs et doit être satisfaisante. Elle doit garantir la disponibilité des salles selon les contraintes techniques (maintenance, capacité) et éviter les conflits d'affectation.

#### Gestion des Erreurs :

Les ambiguïtés (ex: chevauchement de réservations, dépassement de capacité) doivent être signalées par des messages d'erreurs bien organisés pour guider l'utilisateur et le familiariser avec l'application.

#### Ergonomie et Interface Utilisateur :

L'application doit être adaptée à l'utilisateur sans qu'il ne fournisse aucun effort. Une navigation intuitive entre les pages, un design clair (couleurs sobres, textes lisibles), et une accessibilité rapide aux fonctionnalités (ex: calendrier des salles) sont essentiels.

#### Sécurité :

La solution doit respecter la confidentialité des données sensibles (ex: informations des enseignants, historiques de réservation). L'accès aux fonctionnalités critiques (ex: suppression de salles) est réservé aux administrateurs authentifiés.

#### Aptitude à la Maintenance et Réutilisation :

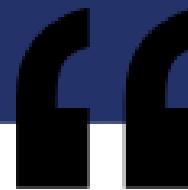
Le système doit être conforme à une architecture standard et claire (ex: modularité du code, documentation technique) pour faciliter sa maintenance (ex: mise à jour des contraintes) et sa réutilisation dans d'autres contextes académiques.

#### Compatibilité et Portabilité :

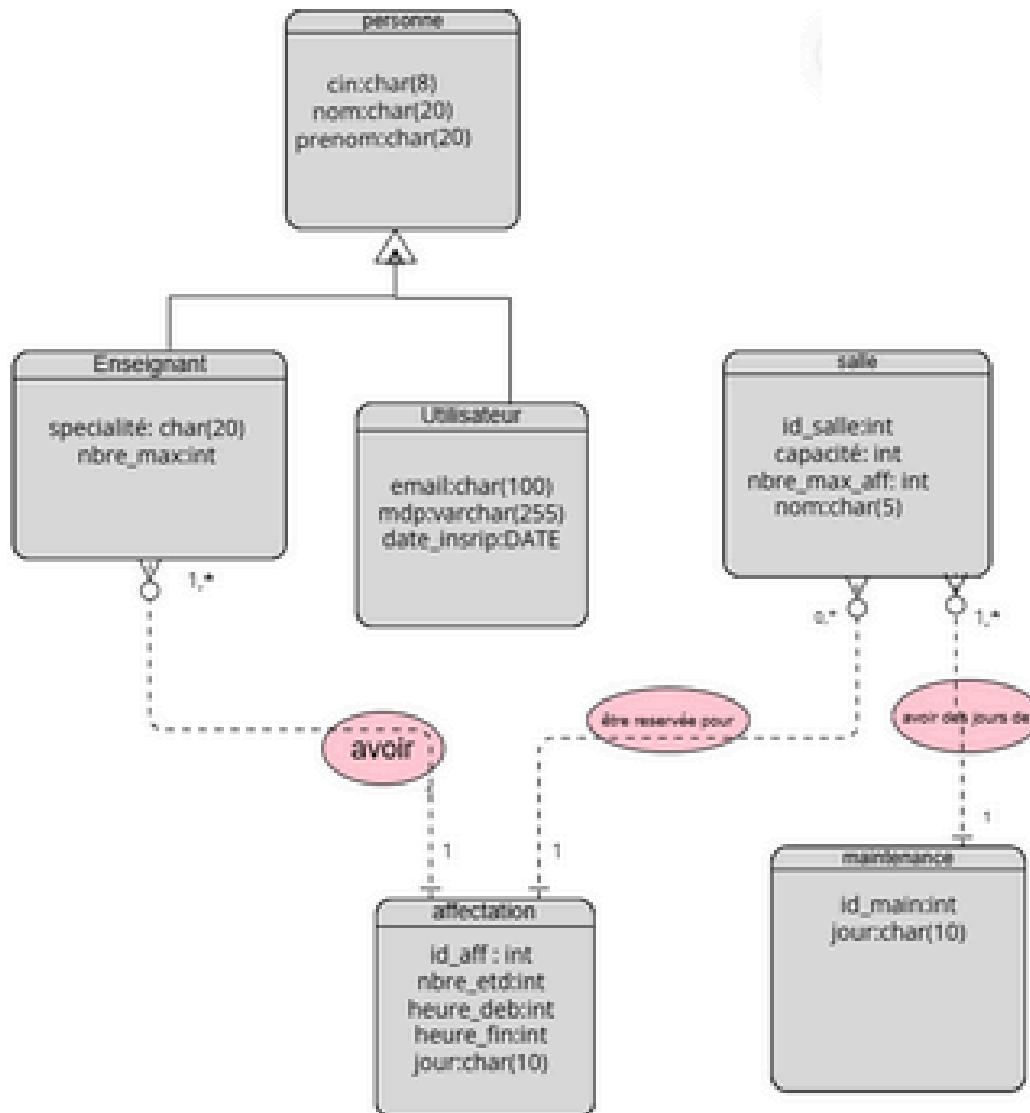
L'application doit être compatible avec les systèmes d'exploitation courants (Windows, Linux, macOS) et les navigateurs web modernes. Elle doit aussi s'intégrer facilement aux bases de données existantes (ex: Oracle, MySQL).

02

## ANALYSE DES BESOINS



## Modèle entité association

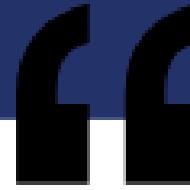


## Modèle Relationnel :

- > Enseignant (CIN, nom, prenom, specialite, nbre\_max\_heure)
- > Salle (id\_salle, nom, capacite, nbr\_max\_aff)
- > Maintenance (id\_maint, jour, #num\_salle)
- > Affectation (id\_affect, jour, heure\_deb, heure\_fin, nbre\_etud, #id\_ens, #id\_salle)
- > Users (CIN, nom, prenom, email, mot\_de\_passe, role, date\_inscription)

# 02

## ANALYSE DES BESOINS



Requêtes SQL:

Création de la base de données :

-- Création de la base de données

```
CREATE DATABASE IF NOT EXISTS salle_tp;
```

```
USE salle_tp;
```

-- Supprime la table enseignant si elle existe déjà

```
DROP TABLE IF EXISTS enseignant;
```

-- Crée une nouvelle table enseignant

```
CREATE TABLE enseignant (
```

-- Colonne pour le CIN de l'enseignant (8 caractères)

```
cin VARCHAR(8) NOT NULL,
```

-- Colonne pour le nom de l'enseignant, ne peut pas être  
vide

```
nom VARCHAR(10) NOT NULL,
```

-- Colonne pour le prénom de l'enseignant, ne peut pas être  
vide

```
prenom VARCHAR(10) NOT NULL,
```

-- Colonne pour la spécialité de l'enseignant

```
specialite VARCHAR(20),
```

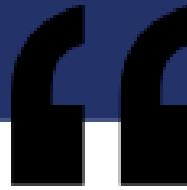
-- Colonne pour le nombre maximum d'heures  
d'enseignement par semaine

```
nbre_max_heure INTEGER,
```

-- Définit la clé primaire de la table sur la colonne cin

```
PRIMARY KEY (cin)
```

```
);
```

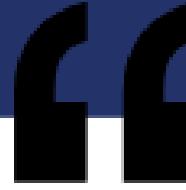


### Requêtes SQL:

Création de la base de données :

```
-- Supprime la table Sallè si elle existe déjà
DROP TABLE IF EXISTS Sallè;
-- Crée une nouvelle table Sallè
CREATE TABLE Sallè (
    -- Colonne pour l'ID de la salle, auto-incrémentée
    Id_sallè INTEGER NOT NULL AUTO_INCREMENT,
    -- Colonne pour le nom de la salle (5 caractères max), doit
    être unique
    nom VARCHAR(5) UNIQUE,
    -- Colonne pour la capacité d'accueil de la salle
    Capacité INTEGER,
    -- Colonne pour le nombre maximum d'affectations par
    semaine
    nbr_max_aff INTEGER,
    -- Définit la clé primaire de la table sur la colonne Id_sallè
    PRIMARY KEY (Id_sallè)
);
```

## ANALYSE DES BESOINS



## Requêtes SQL:

Création de la base de données :

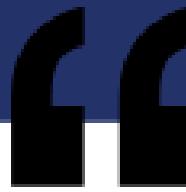
```
-- Supprime la table maintenance si elle existe déjà
DROP TABLE IF EXISTS maintenance;
-- Crée une nouvelle table maintenance
CREATE TABLE maintenance (
    -- Colonne pour l'ID de maintenance, auto-incrémentée
    id_maint INTEGER NOT NULL AUTO_INCREMENT,
    -- Colonne pour le jour de maintenance (ex: "Lundi", "Mardi")
    jour VARCHAR(10),
    -- Colonne pour le numéro de salle en maintenance
    num_salle INTEGER,
    -- Définit la clé primaire de la table sur la colonne id_maint
    PRIMARY KEY (id_maint),
    -- Crée un index sur la colonne num_salle
    KEY num_salle_idx(num_salle),
    -- Définit une contrainte de clé étrangère vers la table salle
    CONSTRAINT num_salle FOREIGN KEY (num_salle)
    REFERENCES salle (id_salle)
);
```

# ANALYSE DES BESOINS

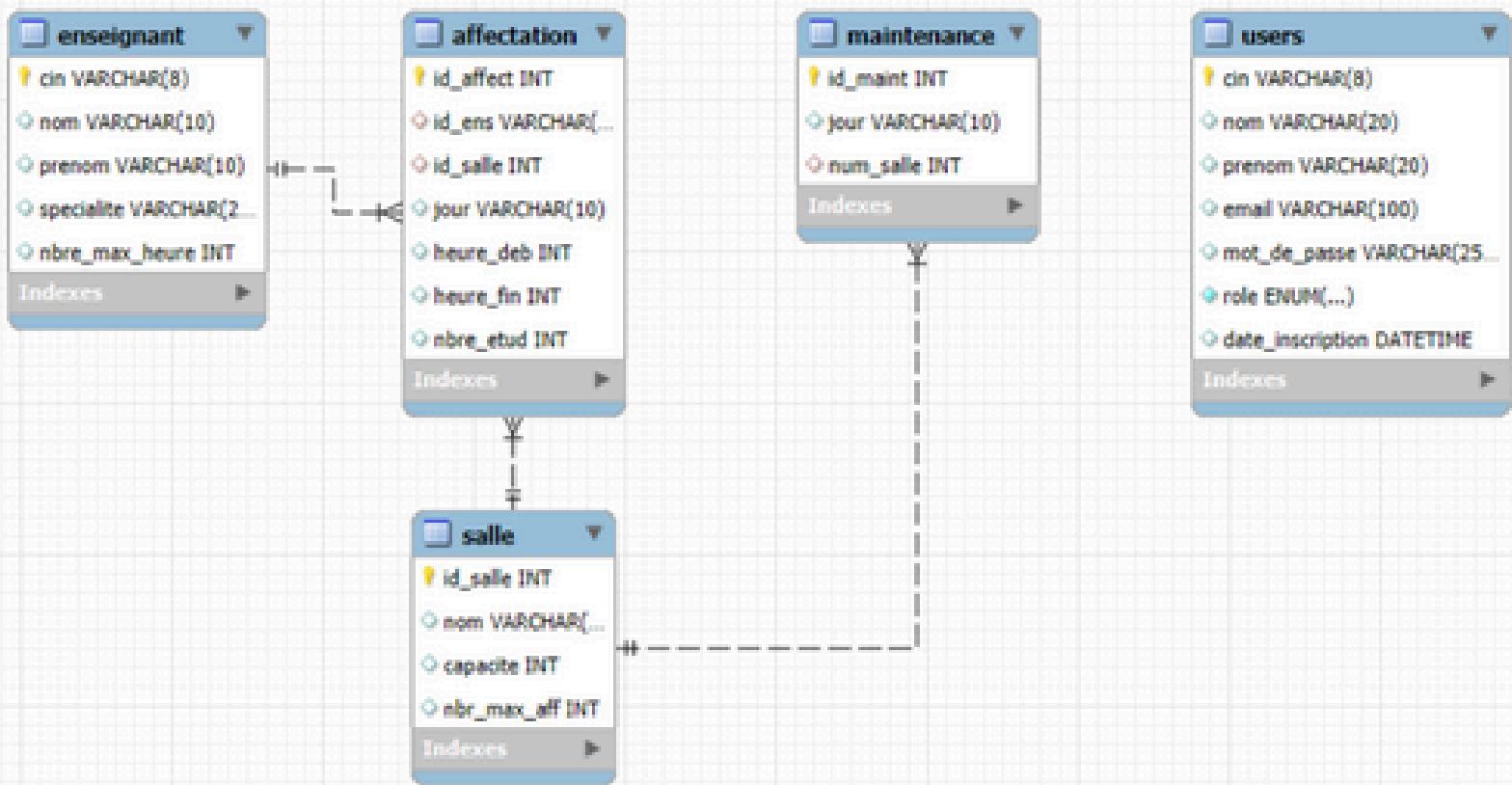
Requêtes SQL:

Création de la base de données :

```
-- Supprime la table affectation si elle existe déjà DROP TABLE IF EXISTS affectation;
-- Crée une nouvelle table affectation
CREATE TABLE affectation (
    -- Colonne pour l'ID d'affectation, auto-incrémentée
    id_affect INTEGER NOT NULL AUTO_INCREMENT,
    -- Colonne pour le CIN de l'enseignant affecté
    id_ens VARCHAR(8),
    -- Colonne pour l'ID de la salle affectée
    id_salle INTEGER,
    -- Colonne pour le jour de l'affectation (ex: "Lundi")
    jour VARCHAR(10),
    -- Colonne pour l'heure de début de la séance
    heure_deb TIME,
    -- Colonne pour l'heure de fin de la séance
    heure_fin TIME,
    -- Colonne pour le nombre d'étudiants prévus
    nbre_etud INTEGER,
    -- Définit la clé primaire de la table sur la colonne id_affect
    PRIMARY KEY (id_affect),
    -- Crée un index sur la colonne id_ens
    KEY id_ens_idx(id_ens),
    -- Crée un index sur la colonne id_salle
    KEY id_salle_idx(id_salle),
    -- Contrainte de clé étrangère vers la table enseignant
    CONSTRAINT id_ens FOREIGN KEY (id_ens)
        REFERENCES enseignant (cin),
    -- Contrainte de clé étrangère vers la table salle
    CONSTRAINT id_salle FOREIGN KEY (id_salle)
        REFERENCES salle (id_salle)
);
```



### Diagramme de classe:



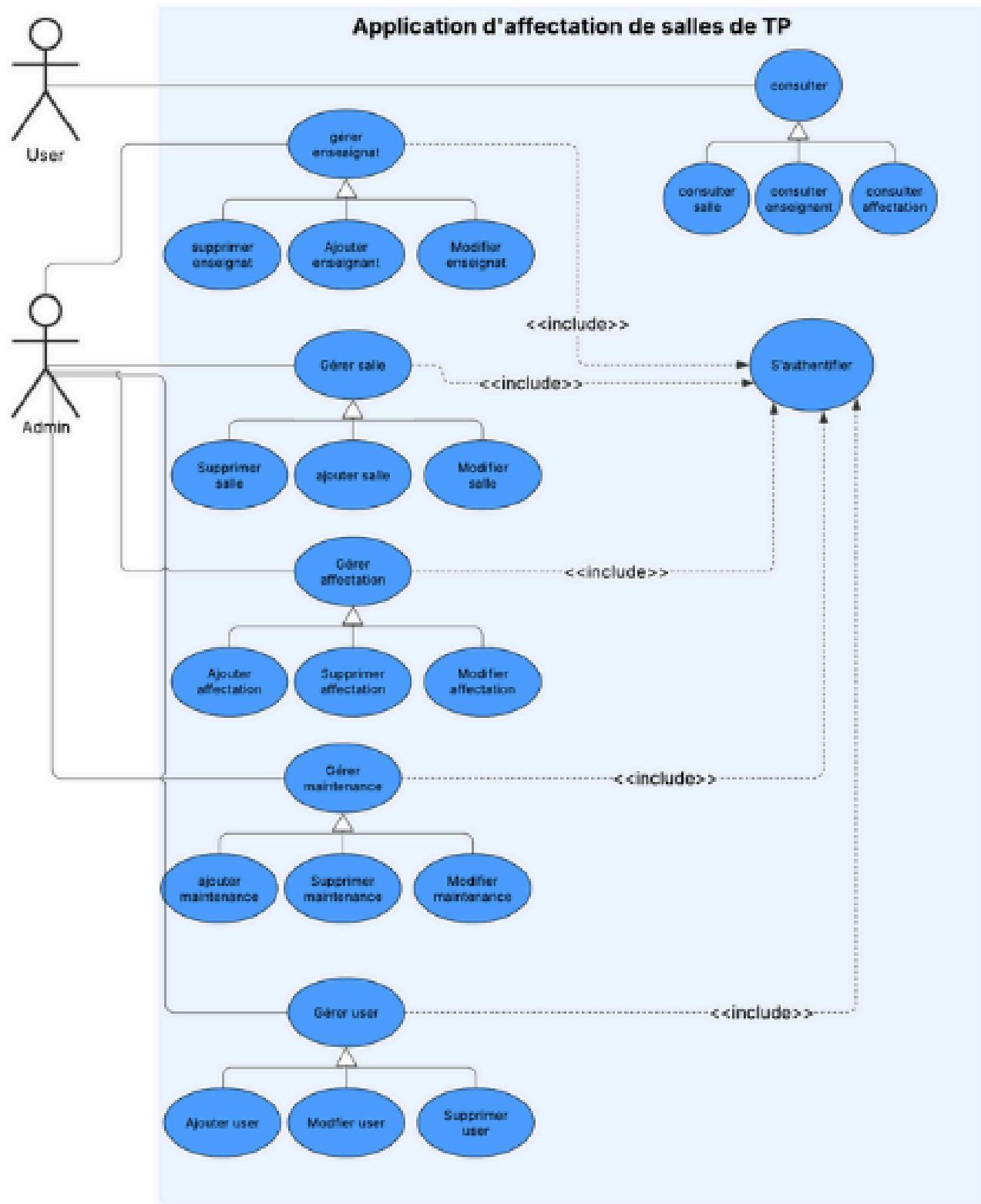
**03**

# **ANALYSE DE DOMAINE ET CONCEPTION**

# 03

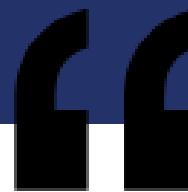
## ANALYSE DE DOMAINE ET CONCEPTION

Diagramme de cas d'utilisation:



# 03

## ANALYSE DE DOMAINE ET CONCEPTION



### Analyse de diagramme de cas d'utilisation:

Les cas d'utilisations sont les outils formels qui permettent de consigner et d'examiner les interactions et les dialogues des utilisateurs (acteurs) avec le système. Un cas d'utilisation est une narration qui décrit un scénario appliqué à une utilisation particulière, dans lequel un acteur fournit des entrées et pour lequel le système produit une sortie observable. Un cas d'utilisation doit exprimer ce que doit faire un acteur, sans préjuger de la façon dont cela sera fait, il décrit le comportement du système vu de l'extérieur.

# Gerer Enseignant

Cas d'utilisation	Gérer Enseignant
Acteurs	Administrateur
Pré-condition	L'administrateur est authentifié
Scénario nominal	<p>[début]</p> <ol style="list-style-type: none"><li>1. L'administrateur accède à l'espace de gestion</li><li>2. Le système affiche la liste des enseignants</li><li>3. L'administrateur choisit une action : Ajouter, Modifier ou Supprimer un enseignant ;<ul style="list-style-type: none"><li>- Si "Ajouter" :<ol style="list-style-type: none"><li>a. Le système affiche un formulaire de saisie ;</li><li>b. L'administrateur saisit les informations de l'enseignant ;</li><li>c. Le système enregistre les données ;</li></ol></li><li>- Si "Modifier" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne un enseignant ;</li><li>b. Le système affiche les informations de l'enseignant ;</li><li>c. L'administrateur modifie les données et les enregistre ;</li></ol></li><li>- Si "Supprimer" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne un enseignant ;</li><li>b. Le système demande confirmation ;</li><li>c. L'administrateur confirme la suppression ;</li><li>d. Le système supprime l'enseignant ;</li></ol></li></ul><p>4. Le système met à jour la liste des enseignants.</p><p>[fin]</p></li></ol>
Alternative Exception	Si les informations saisies sont incomplètes ou incorrectes, le système affiche un message d'erreur et demande à l'administrateur de corriger les données.

# Gerer Salle

Cas d'utilisation	Les
Acteurs	Gérer salle
Pré-condition	L'administrateur est authentifié
Scénario nominal	<p>[début]</p> <ol style="list-style-type: none"><li>1. L'administrateur accède à l'espace de gestion des salles</li><li>2. Le système affiche la liste des salles existantes</li><li>3. L'administrateur choisit une action : Ajouter, Modifier ou Supprimer une salle ;<ul style="list-style-type: none"><li>- Si "Ajouter" :<ol style="list-style-type: none"><li>a. Le système affiche un formulaire de saisie ;</li><li>b. L'administrateur saisit les informations de la salle ;</li><li>c. Le système enregistre les données ;</li></ol></li><li>- Si "Modifier" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne une salle ;</li><li>b. Le système affiche les informations de la salle ;</li><li>c. L'administrateur modifie les données et les enregistre ;</li></ol></li><li>- Si "Supprimer" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne une salle ;</li><li>b. Le système demande confirmation ;</li><li>c. L'administrateur confirme la suppression ;</li><li>d. Le système supprime la salle ;</li></ol></li></ul></li><li>4. Le système met à jour la liste des salles.</li></ol> <p>[fin]</p>
Alternative Exception	Si les informations saisies sont incomplètes ou incorrectes, le système affiche un message d'erreur et demande à l'administrateur de corriger les données.

# Gerer affectation

Cas d'utilisation	Les	Gérer salle
Acteurs		Administrateur
Pré-condition		L'administrateur est authentifié
Scénario nominal		<p>[début]</p> <ol style="list-style-type: none"><li>1. L'administrateur accède à l'interface de gestion des affectations ;</li><li>2. Le système affiche les listes des enseignants, matières, salles et horaires ;</li><li>3. L'administrateur sélectionne les éléments à affecter : un enseignant, une matière, une salle et un horaire ;</li><li>4. Le système vérifie la disponibilité des ressources sélectionnées ;</li><li>5. Si toutes les ressources sont disponibles, le système enregistre l'affectation</li><li>; 6. Le système affiche la nouvelle affectation dans la liste ;</li></ol> <p>[fin]</p>
Alternative Exception		Si une des ressources est déjà affectée à ce créneau horaire, le système affiche un message d'erreur et invite l'administrateur à modifier sa sélection.

# Gerer maintenance

Cas d'utilisation	Les	Gérer salle
Acteurs		Administrateur
Pré-condition		L'administrateur est authentifié
Scénario nominal		<p>[début]</p> <ol style="list-style-type: none"><li>1. L'administrateur accède à l'interface de gestion de maintenance ;</li><li>2. Le système affiche la liste des maintenances enregistrées ;</li><li>3. L'administrateur choisit une action : Ajouter, Modifier ou Supprimer une fiche de maintenance ;<ul style="list-style-type: none"><li>- Si "Ajouter" :<ol style="list-style-type: none"><li>a. Le système affiche un formulaire ;</li><li>b. L'administrateur saisit les informations (objet, description, date, etc.) ;</li><li>c. Le système enregistre la maintenance ;</li></ol></li><li>- Si "Modifier" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne une maintenance ;</li><li>b. Modifie les informations et enregistre ;</li></ol></li><li>- Si "Supprimer" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne une fiche ;</li><li>b. Confirme la suppression ;</li><li>c. Le système supprime la fiche ;</li></ol></li></ul></li><li>4. Le système met à jour l'interface.</li></ol> <p>[fin]</p>
Alternative Exception		Si les données sont incomplètes ou incorrectes, le système affiche une erreur et demande une correction.

# Gérer user

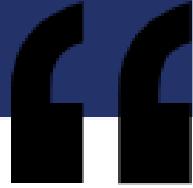
Cas d'utilisation	Les
Acteurs	Gérer salle Administrateur
Pré-condition	L'administrateur est authentifié
Scénario nominal	<p>[début]</p> <ol style="list-style-type: none"><li>1. L'administrateur accède à l'interface de gestion des utilisateurs</li><li>2. Le système affiche la liste des utilisateurs existants</li><li>3. L'administrateur choisit une action : Ajouter, Modifier ou Supprimer un utilisateur<ul style="list-style-type: none"><li>- Si "Ajouter" :<ol style="list-style-type: none"><li>a. Le système affiche un formulaire ;</li><li>b. L'administrateur saisit les informations (nom, CIN, rôle, mot de passe...);</li><li>c. Le système enregistre le nouvel utilisateur ;</li></ol></li><li>- Si "Modifier" :<ol style="list-style-type: none"><li>a. L'administrateur sélectionne un utilisateur ;</li><li>b. Il modifie les informations et enregistre ;</li></ol></li><li>- Si "Supprimer"<ul style="list-style-type: none"><li>a. L'administrateur sélectionne un utilisateur ;</li><li>b. Confirme la suppression ;</li><li>c. Le système supprime l'utilisateur ;</li></ul></li></ul></li><li>4. Le système met à jour la liste des utilisateurs.</li></ol> <p>[fin]</p>
Alternative Exception	Si les informations saisies sont incomplètes ou le CIN est déjà utilisé, le système affiche un message d'erreur.

04

# REALISATION

# 04

## REALISATION

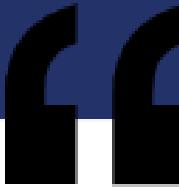


### Les outils de développement

L'étape de réalisation est la dernière de notre projet, elle se présente comme étant l'étape la plus cruciale vu qu'elle traite l'onglet pratique du projet. Nous commençons d'abord par une brève illustration de l'environnement de travail ainsi que l'ensemble des logiciels qu'on a utilisé dans la réalisation de notre application d'affectation des salles de TP et l'implémentation de base de données, puis nous passons à un aperçu des interfaces les plus importantes de notre application.

# 04

## REALISATION



### Les outils de développement

#### **MySQL:**

MySQL est le système de gestion de base de données relationnelle (SGBDR) que nous avons utilisé pour stocker et organiser toutes les données de l'application (salles, enseignants, affectations). Son modèle client-serveur et son support du langage SQL nous ont permis de :

Créer la structure relationnelle via des tables (salle, enseignant, affectation, etc.)

Implémenter des contraintes d'intégrité et la procédure PL/SQL

Gérer les transactions et requêtes complexes

#### **MySQL Workbench:**

Outil visuel complémentaire à MySQL, nous l'avons utilisé pour :

Concevoir le modèle physique de la base via des diagrammes ER

Générer et exécuter les scripts SQL de création de tables

Tester les requêtes d'affectation et de maintenance

Administrer la base salle\_tp (droits d'accès, sauvegardes)

#### **Java Development Kit (JDK):**

Le JDK 17 (version LTS) a été essentiel pour :

Compiler et exécuter l'application Java

Utiliser les API JDBC pour la connexion à la base MySQL

Bénéficier des dernières fonctionnalités du langage Java

#### **NetBeans:**

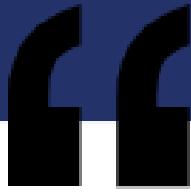
Notre IDE principal pour le développement offre :

Un environnement intégré pour le code Java et l'interface graphique (Swing)

Une intégration native avec MySQL via le connecteur JDBC

Des outils de débogage et de gestion de dépendances

La génération automatique de code pour les classes DAO



Les langages de programmation utilisés:

### **Le langage SQL:**

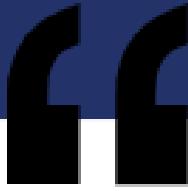
Le langage de requête structuré (SQL) est un langage de programmation permettant de stocker et de traiter des informations dans une base de données relationnelle. Une base de données relationnelle stocke les informations sous forme de tableaux, avec des lignes et des colonnes représentant les différents attributs de données et les relations entre les valeurs. Vous pouvez utiliser des instructions SQL pour stocker, mettre à jour, supprimer, rechercher et récupérer des informations de la base de données. Vous pouvez également utiliser SQL pour maintenir et optimiser les performances de la base de données.

### **Définition de Java:**

Java C'est un langage de programmation orienté objet, développé par Sun Microsystems. Dont le squelette principale et constitué du langage C++. En revanche, ces deux langages sont très différents dans leurs structures (organisation du code et gestion des variables). Ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web (applet) ou encore comme langage serveur (jsp) et la réalisation de jeux fortement basé sur le graphisme. On effet, il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris) et la création de graphismes élaborés, animés, ainsi que la présentation de texte évolué (hypertexte). L'avantage principal de Java par rapport aux autres langages c'est sa PORTABILITÉ, le fait qu'un programme Java puisse théoriquement être exécuté sur n'importe quelle plate-forme (type de processeur et système d'exploitation)

# 04

## REALISATION



### La distribution des tâches

#### **Rayen Akriche :**

- Design des interfaces graphique en Canva
- Page login et inscription
- Page de modification des comptes
- Les interfaces de l'admin et d'utilisateur
- Affichage des enseignants et des affectations
- Une partie du rapport

#### **Mohamed Amine Hajji:**

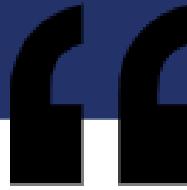
- Gestion des salles
- Affichage des salles
- Une partie du rapport

#### **Mootaz Hamam:**

- Gestion des enseignants
- Gestion des utilisateurs
- Analyse du projet
- Une partie du rapport

#### **Hend Jacob :**

- Gestion des affectations
- Gestion des maintenances
- Une partie du rapport
- La creation des tables en SQL
- Les diagrammes CU et de classe



Les requêtes SQL Utilisé en Java:

Connexion:

```
connect();
String query = "SELECT role, cin FROM users WHERE email = ? AND mot_de_passe = ?";
try {
    PreparedStatement pst = conn.prepareStatement(query);
    pst.setString(1, email);
    pst.setString(2, password);
    ResultSet rs = pst.executeQuery();
```

Inscription:

```
try {
    // Check if a user with the same CIN or email already exists
    String checkQuery = "SELECT * FROM users WHERE cin = ? OR email = ?";
    PreparedStatement checkStmt = conn.prepareStatement(checkQuery);
    checkStmt.setString(1, cinStr);
    checkStmt.setString(2, email);
    ResultSet rs = checkStmt.executeQuery();
```

```
// Insert the new user; role is set to 'user' by default, and date_inscription is auto-set.
String insertQuery = "INSERT INTO users (cin, nom, prenom, email, mot_de_passe, role) VALUES (?, ?, ?, ?, ?, 'user')";
PreparedStatement insertStmt = conn.prepareStatement(insertQuery);
insertStmt.setString(1, cinStr);
insertStmt.setString(2, nom);
insertStmt.setString(3, prenom);
insertStmt.setString(4, email);
insertStmt.setString(5, password);
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Modification du compte:

```
String query = "SELECT cin, nom, prenom, email FROM users WHERE cin = ?";
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setString(1, currentCin);
ResultSet rs = stmt.executeQuery();

// 1. Check for duplicate email
String emailCheckQuery = "SELECT * FROM users WHERE email = ? AND cin != ?";
try (PreparedStatement emailCheckStmt = conn.prepareStatement(emailCheckQuery)) {
    emailCheckStmt.setString(1, newEmail);
    emailCheckStmt.setString(2, currentCin);
    try (ResultSet emailRs = emailCheckStmt.executeQuery()) {

// 2. Update user data
String updateQuery = "UPDATE users SET nom = ?, prenom = ?, email = ? WHERE cin = ?";
try (PreparedStatement updateStmt = conn.prepareStatement(updateQuery)) {
    updateStmt.setString(1, newNom);
    updateStmt.setString(2, newPrenom);
    updateStmt.setString(3, newEmail);
    updateStmt.setString(4, currentCin);

    int rowsUpdated = updateStmt.executeUpdate();

// Vérification du rôle avant redirection
String roleQuery = "SELECT role FROM users WHERE cin = ?";
try (PreparedStatement roleStmt = conn.prepareStatement(roleQuery)) {
    roleStmt.setString(1, currentCin);
    try (ResultSet roleRs = roleStmt.executeQuery()) {

String roleQuery = "SELECT role FROM users WHERE cin = ?";
try (PreparedStatement pst = conn.prepareStatement(roleQuery)) {
    pst.setString(1, currentCin);
    try (ResultSet rs = pst.executeQuery()) {
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Modification mot de passe:

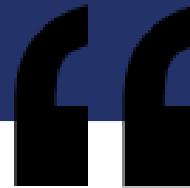
```
String query = "SELECT mot_de_passe FROM users WHERE cin = ?";  
try (PreparedStatement pst = con.prepareStatement(query)) {  
    pst.setString(1, currentCin);  
    try (ResultSet rs = pst.executeQuery()) {
```

```
// Update password  
String updateQuery = "UPDATE users SET mot_de_passe = ? WHERE cin = ?";  
try (PreparedStatement updatePst = con.prepareStatement(updateQuery)) {  
    updatePst.setString(1, newMdp);  
    updatePst.setString(2, currentCin);  
    updatePst.executeUpdate();  
}
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:



Interface de l'admin:

Gestion des enseignants:

```
private void loadEnseignants() {
    DefaultTableModel model = (DefaultTableModel) EnsTable.getModel();
    model.setRowCount(0); // Clear table first

    try {
        connect();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM enseignant");

        // Vérifie si un enseignant avec ce CIN existe déjà
        String checkQuery = "SELECT * FROM enseignant WHERE cin = ?";
        PreparedStatement checkStmt = conn.prepareStatement(checkQuery);
        checkStmt.setString(1, cinStr);
        ResultSet rs = checkStmt.executeQuery();

        // Insertion
        String sql = "INSERT INTO enseignant (cin, nom, prenom, specialite, nbre_max_heure) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, cinStr);
        pstmt.setString(2, nom);
        pstmt.setString(3, prenom);
        pstmt.setString(4, specialite);
        pstmt.setInt(5, nbreMax);

        pstmt.executeUpdate();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error : " + e.getMessage());
    }
}
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des enseignants:

```
try {
    connect();
    String sql = "UPDATE enseignant SET nom = ?, prenom = ?, specialite = ?, nbre_max_heure = ? WHERE cin = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, nom);
    pstmt.setString(2, prenom);
    pstmt.setString(3, specialite);
    pstmt.setInt(4, nbreMax);
    pstmt.setString(5, cinStr);

    int rowsAffected = pstmt.executeUpdate();
```

```
// 1. Supprime les affectations liées
String deleteAffectationQuery = "DELETE FROM Affectation WHERE id_ens = ?";
try (PreparedStatement pstAffectation = conn.prepareStatement(deleteAffectationQuery)) {
    pstAffectation.setString(1, cin);
    pstAffectation.executeUpdate();
}

// 2. Supprime l'enseignant
String deleteEnseignantQuery = "DELETE FROM Enseignant WHERE cin = ?";
try (PreparedStatement pstEnseignant = conn.prepareStatement(deleteEnseignantQuery)) {
    pstEnseignant.setString(1, cin);
    int rowsDeleted = pstEnseignant.executeUpdate();
```

Gestion des maintenances:

```
private void updateMaintenance(String idMaintenance, int column, String newValue) {
    try (Connection con = connectToDatabase()) {
        // 1. Récupérer les valeurs actuelles
        String currentSql = "SELECT * FROM maintenance WHERE id_maint = ?";
        PreparedStatement currentPst = con.prepareStatement(currentSql);
        currentPst.setString(1, idMaintenance);
        ResultSet rs = currentPst.executeQuery();
```

# 04

## REALISATION



Les requêtes SQL Utilisé en Java:

Gestion des maintenances:

```
// 2. Vérifier qu'il n'y a pas d'affectation ce jour-là dans cette salle
String checkAffectation = "SELECT COUNT(*) FROM affectation WHERE id_salle = ? AND jour = ?";
try (PreparedStatement pst = con.prepareStatement(checkAffectation)) {
    pst.setString(1, numSalle);
    pst.setString(2, jour);
    ResultSet rsAffect = pst.executeQuery();
```

```
// 3. Vérifier unicité dans maintenance (sauf la ligne actuelle)
String checkUnique = "SELECT COUNT(*) FROM maintenance WHERE num_salle = ? AND jour = ? AND id_maint != ?";
try (PreparedStatement pst = con.prepareStatement(checkUnique)) {
    pst.setString(1, numSalle);
    pst.setString(2, jour);
    pst.setString(3, idMaintenance);
    ResultSet rsUnique = pst.executeQuery();
```

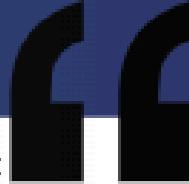
```
        //vérifier salle dans la base
        String checkSalle = "SELECT COUNT(*) FROM salle WHERE id_salle = ?";
        try (PreparedStatement pstSalle = con.prepareStatement(checkSalle)) {
            pstSalle.setString(1, numSalle);
            try (ResultSet rsSalle = pstSalle.executeQuery()) {
```

```
// 4. Effectuer la mise à jour
String updateSql = "";
switch (column) {
    case 1: updateSql = "UPDATE maintenance SET jour = ? WHERE id_maint = ?"; break;
    case 2: updateSql = "UPDATE maintenance SET num_salle = ? WHERE id_maint = ?"; break;
}

try (PreparedStatement pst = con.prepareStatement(updateSql)) {
    pst.setString(1, newValue);
    pst.setString(2, idMaintenance);
    pst.executeUpdate();
    loadMaintenance();
```

# 04

## REALISATION



Les requêtes SQL Utilisé en Java:

Gestion des maintenances:

```
private void loadMaintenance() {
    try (Connection con = connectToDatabase()) {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM maintenance");
```

```
try (Connection con = connectToDatabase()) {
    PreparedStatement pst = con.prepareStatement("DELETE FROM maintenance WHERE id_maint = ?");
    pst.setString(1, idMaintenance);
    pst.executeUpdate();
    loadMaintenance(); // Recharger les affectations après la suppression
```

```
try (Connection con = connectToDatabase()) {

    // 1. Vérifier que la salle existe
    String checkSalle = "SELECT COUNT(*) FROM salle WHERE id_salle = ?";
    try (PreparedStatement pstSalle = con.prepareStatement(checkSalle)) {
        pstSalle.setInt(1, numSalle);
        try (ResultSet rsSalle = pstSalle.executeQuery()) {
```

```
// 2. Vérifier qu'il n'y a pas d'affectation ce jour-là dans cette salle
String checkAffectation = "SELECT COUNT(*) FROM affectation WHERE id_salle = ? AND jour = ?";
try (PreparedStatement pst = con.prepareStatement(checkAffectation)) {
    pst.setInt(1, numSalle);
    pst.setString(2, jour);
    ResultSet rsAffect = pst.executeQuery();
```

```
// 3. Vérifier unicité dans maintenance
String checkUnique = "SELECT COUNT(*) FROM maintenance WHERE num_salle = ? AND jour = ?";
try (PreparedStatement pst = con.prepareStatement(checkUnique)) {
    pst.setInt(1, numSalle);
    pst.setString(2, jour);
    ResultSet rsUnique = pst.executeQuery();
```

# REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des maintenances:

```
// 4. Insertion dans la table maintenance

String insertSql = "INSERT INTO maintenance (num_salle, jour) VALUES (?, ?)";

try (PreparedStatement pst = con.prepareStatement(insertSql)) {
    pst.setInt(1, numSalle);
    pst.setString(2, jour);
    pst.executeUpdate();
}
```

Gestion des utilisateurs:

```
try {
    connect();
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM users");
```

```
try {
    connect();

    String sql = "UPDATE users SET nom = ?, prenom = ?, email = ?, mot_de_passe = ?, role = ? WHERE cin = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, nom);
    pstmt.setString(2, prenom);
    pstmt.setString(3, email);
    pstmt.setString(4, motDePasse);
    pstmt.setString(5, role);
    pstmt.setString(6, cinStr);

    int rowsAffected = pstmt.executeUpdate();
```

```
try {
    connect();
    String sql = "DELETE FROM users WHERE cin = ?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, cinStr);

    int rowsDeleted = pstmt.executeUpdate();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des utilisateurs:

```
try {
    connect();
    String checkQuery = "SELECT * FROM users WHERE cin = ? OR email = ?";
    PreparedStatement checkStmt = conn.prepareStatement(checkQuery);
    checkStmt.setString(1, cinStr);
    checkStmt.setString(2, email);
    ResultSet rs = checkStmt.executeQuery();
```

```
try {
    connect();
    String sql = "INSERT INTO users (cin, nom, prenom, email, mot_de_passe, role) VALUES (?, ?, ?, ?, ?, ?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, cinStr);
    pstmt.setString(2, nom);
    pstmt.setString(3, prenom);
    pstmt.setString(4, email);
    pstmt.setString(5, motDePasse);
    pstmt.setString(6, role);

    int rowsInserted = pstmt.executeUpdate();
```

Gestion des salles:

```
try{
    connect();
    query = "SELECT * FROM salle";
    String[] columns = {"ID Salle", "Nom", "Capacité","Nbr aff max"};
    DefaultTableModel model = new DefaultTableModel(null, columns);

    Statement sta = con.createStatement();
    rs = sta.executeQuery(query);
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des salles:

```
// Supprimer d'abord les dépendances dans maintenance
PreparedStatement pst1 = con.prepareStatement("DELETE FROM affectation WHERE id_salle = ?");
pst1.setString(1, salle); // id est l'id_salle que tu veux supprimer
pst1.executeUpdate();
    salle elle-même

PreparedStatement pst2 = con.prepareStatement("DELETE FROM salle WHERE id_salle = ?");
pst2.setString(1, salle);
int rowsDeleted = pst2.executeUpdate();
```

```
try{
    connect();
    String sql = "UPDATE salle SET nom = ?, capacite = ?, nbr_max_aff = ? WHERE id_salle = ?";
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString(1, nom);
    pstmt.setInt(2, capacite);
    pstmt.setInt(3, specialite);
    pstmt.setInt(4, id);
```

```
try{
    connect();
    String sql = "INSERT INTO salle VALUES (?, ?, ?, ?)";
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setInt(1, Integer.parseInt(id1.getText())); // id_salle
    pstmt.setString(2, nom1.getText()); // nom
    pstmt.setInt(3, Integer.parseInt(capacite1.getText())); // capacite
    pstmt.setInt(4, Integer.parseInt(nb_max_aff.getText())); // nbr_max_aff

    pstmt.executeUpdate();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
String currentSql = "SELECT * FROM affectation WHERE id_affect = ?";  
PreparedStatement currentPst = con.prepareStatement(currentSql);  
currentPst.setString(1, idAffectation);  
ResultSet rs = currentPst.executeQuery();  
  
        String checkEnseignantOccupation = "";  
        SELECT COUNT(*) FROM affectation  
        WHERE id_ens = ? AND jour = ? AND id_affect != ?  
        AND (  
            (? BETWEEN heure_deb AND heure_fin)  
            OR (? BETWEEN heure_deb AND heure_fin)  
            OR (heure_deb BETWEEN ? AND ?))  
        )"";  
try (PreparedStatement pstEnsOcc = con.prepareStatement(checkEnseignantOccupation)) {  
    pstEnsOcc.setString(1, idEns);  
    pstEnsOcc.setString(2, jour);  
    pstEnsOcc.setString(3, idAffectation);  
    pstEnsOcc.setString(4, heureDeb);  
    pstEnsOcc.setString(5, heureFin);  
    pstEnsOcc.setString(6, heureDeb);  
    pstEnsOcc.setString(7, heureFin);  
    ResultSet rsEnsOcc = pstEnsOcc.executeQuery();
```

```
// 3. Vérifier maintenance  
String checkMaintenance = "SELECT COUNT(*) FROM maintenance WHERE num_salle = ? AND jour = ?";  
try (PreparedStatement pst = con.prepareStatement(checkMaintenance)) {  
    pst.setString(1, idSalle);  
    pst.setString(2, jour);  
    ResultSet rsMaint = pst.executeQuery();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
String queryQuotaSalle = "SELECT nbr_max_aff FROM salle WHERE id_salle = ?";  
try (PreparedStatement pstQuota = con.prepareStatement(queryQuotaSalle)) {  
    pstQuota.setString(1, idSalle);  
    ResultSet rsQuota = pstQuota.executeQuery();
```

```
// 6.2 Calculer la somme des affectations existantes  
//(en minutes) pour cette salle et cette semaine  
String queryTotalAffectation = """  
    SELECT count(*) as total  
    FROM affectation  
    WHERE id_salle = ?  
""";  
try (PreparedStatement pstTotal = con.prepareStatement(queryTotalAffectation)) {  
    pstTotal.setString(1, idSalle);  
    ResultSet rsTotal = pstTotal.executeQuery();
```

```
// 3. Vérifier maintenance  
String checkMaintenance = "SELECT COUNT(*) FROM maintenance WHERE num_salle = ? AND jour = ?";  
try (PreparedStatement pst = con.prepareStatement(checkMaintenance)) {  
    pst.setString(1, idSalle);  
    pst.setString(2, jour);  
    ResultSet rsMaint = pst.executeQuery();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
// 6.2 Calculer la somme des affectations existantes
//(en minutes) pour cette salle et cette semaine
String queryTotalAffectation = """
    SELECT count(*) as total
    FROM affectation
    WHERE id_salle = ?
""";
try (PreparedStatement pstTotal = con.prepareStatement(queryTotalAffectation)) {
    pstTotal.setString(1, idSalle);
    ResultSet rsTotal = pstTotal.executeQuery();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
// 4. Vérifier occupation salle (exclure l'affectation actuelle)
String checkOccupation = """
    SELECT COUNT(*) FROM affectation
    WHERE id_salle = ? AND jour = ? AND id_affect != ?
    AND (
        (? BETWEEN heure_deb AND heure_fin)
        OR (? BETWEEN heure_deb AND heure_fin)
        OR (heure_deb BETWEEN ? AND ?)
    )""";
try (PreparedStatement pst = con.prepareStatement(checkOccupation)) {
    pst.setString(1, idSalle);
    pst.setString(2, jour);
    pst.setString(3, idAffectation);
    pst.setString(4, heureDeb);
    pst.setString(5, heureFin);
    pst.setString(6, heureDeb);
    pst.setString(7, heureFin);
    ResultSet rsOcc = pst.executeQuery();
```

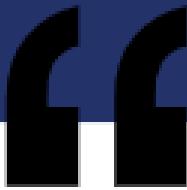
```
// 5. Vérification de la capacité de la salle
String checkCapaciteSalle = "SELECT capacite FROM salle WHERE id_salle = ?";
try (PreparedStatement pst = con.prepareStatement(checkCapaciteSalle)) {
    pst.setString(1, idSalle);
    ResultSet rsCapacite = pst.executeQuery();
```

delete from affectation

```
try (Connection con = connectToDatabase()) {
    PreparedStatement pst = con.prepareStatement("DELETE FROM affectation WHERE id_affect = ?");
    pst.setString(1, idAffectation);
    pst.executeUpdate();
    loadAffectations(); // Recharger les affectations après la suppression
```

# 04

## REALISATION



Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
// 6. Effectuer la mise à jour
String updateSql = "";
switch (column) {
    case 1: updateSql = "UPDATE affectation SET id_ens = ? WHERE id_affect = ?"; break;
    case 2: updateSql = "UPDATE affectation SET id_salle = ? WHERE id_affect = ?"; break;
    case 3: updateSql = "UPDATE affectation SET jour = ? WHERE id_affect = ?"; break;
    case 4: updateSql = "UPDATE affectation SET heure_deb = ? WHERE id_affect = ?"; break;
    case 5: updateSql = "UPDATE affectation SET heure_fin = ? WHERE id_affect = ?"; break;
    case 6: updateSql = "UPDATE affectation SET nbre_etud = ? WHERE id_affect = ?"; break;
}

try (PreparedStatement pst = con.prepareStatement(updateSql)) {
    pst.setString(1, newValue);
    pst.setString(2, idAffectation);
    pst.executeUpdate();
```

nombre total de minutes par enseignant:

```
String query = """
    SELECT
        SUM(TIMESTAMPDIFF(MINUTE, heure_deb, heure_fin)) AS diff_minutes
    FROM affectation
    WHERE id_ens = ?
""";
```

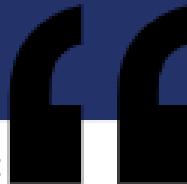
```
try (Connection conn = connectToDatabase();
      PreparedStatement stmt = conn.prepareStatement(query)) {

    // Remplacer le ? par le CIN de l'enseignant
    stmt.setString(1, cin);

    // Exécuter la requête
    ResultSet rs = stmt.executeQuery();
```

# 04

## REALISATION



Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
private void loadAffectations() {
    try (Connection con = connectToDatabase()) {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM affectation");
```

```
// Vérifier la maintenance
String checkMaintenance = "SELECT COUNT(*) FROM maintenance WHERE num_salle = ? AND jour = ?";
PreparedStatement pstMaint = con.prepareStatement(checkMaintenance);
pstMaint.setString(1, idSalle);
pstMaint.setString(2, jour);
ResultSet rsMaint = pstMaint.executeQuery();
```

```
// verifier que l enseignant existe dans la base
String checkEnseignant = "SELECT COUNT(*) FROM enseignant WHERE cin = ?";
try (PreparedStatement pstEns = con.prepareStatement(checkEnseignant)) {
    pstEns.setString(1, idEns);
    try (ResultSet rsEns = pstEns.executeQuery()) {
```

```
        // verifier salle existe dans la base:
        String checkSalle = "SELECT COUNT(*) FROM salle WHERE id_salle = ?";
        try (PreparedStatement pstSalle = con.prepareStatement(checkSalle)) {
            pstSalle.setString(1, idSalle);
            try (ResultSet rsSalle = pstSalle.executeQuery()) {
```

# REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
// Vérifier l'occupation
String checkOccupation = """
    SELECT COUNT(*) FROM affectation
    WHERE id_salle = ? AND jour = ?
    AND (
        (? BETWEEN heure_deb AND heure_fin)
        OR (? BETWEEN heure_deb AND heure_fin)
        OR (heure_deb BETWEEN ? AND ?)
    )""";
PreparedStatement pstOcc = con.prepareStatement(checkOccupation);
pstOcc.setString(1, idSalle);
pstOcc.setString(2, jour);
pstOcc.setString(3, heureDeb);
pstOcc.setString(4, heureFin);
pstOcc.setString(5, heureDeb);
pstOcc.setString(6, heureFin);
ResultSet rsOcc = pstOcc.executeQuery();
```

```
// 6. Vérifier quota d'affectation de la salle
int quotaSalle = 0;
int totalaff=0;

String queryQuotaSalle = "SELECT nbr_max_aff FROM salle WHERE id_salle = ?";
try (PreparedStatement pstQuota = con.prepareStatement(queryQuotaSalle)) {
    pstQuota.setString(1, idSalle);
    ResultSet rsQuota = pstQuota.executeQuery();

    String queryTotalAffectation = """
        SELECT count(*) as total
        FROM affectation
        WHERE id_salle = ?
    """;
    try (PreparedStatement pstTotal = con.prepareStatement(queryTotalAffectation)) {
        pstTotal.setString(1, idSalle);
        ResultSet rsTotal = pstTotal.executeQuery();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Gestion des affectations:

```
// 3. Vérifier capacité salle
String checkCapaciteSalle = "SELECT capacite FROM salle WHERE id_salle = ?";
PreparedStatement pstCapacite = con.prepareStatement(checkCapaciteSalle);
pstCapacite.setString(1, idSalle);
ResultSet rsCapacite = pstCapacite.executeQuery();
```

```
String query = """
    SELECT
        nbre_max_heure as nbre_max
    FROM enseignant
    WHERE cin = ?
""";
try (Connection conn = connectToDatabase()) {
    PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1,idEns);
        ResultSet rs = stmt.executeQuery();
```

```
// 5. Insertion
String sql = "INSERT INTO affectation (id_ens, id_salle, jour, heure_deb, heure_fin, nbre_etud)"
        +"VALUES (?, ?, ?, ?, ?, ?)";
PreparedStatement pst = con.prepareStatement(sql);
pst.setString(1, idEns);
pst.setString(2, idSalle);
pst.setString(3, jour);
pst.setString(4, heureDeb);
pst.setString(5, heureFin);
pst.setInt(6, nbreEtud);
pst.executeUpdate();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Interface du user:

Afficher Enseignant:

```
private void loadEnseignantsData() {  
    connect();  
    try {  
        String query = "SELECT cin, nom, prenom, specialite, nbr_max_heure FROM enseignant";  
        PreparedStatement pst = conn.prepareStatement(query);  
        ResultSet rs = pst.executeQuery();
```

Afficher Salles:

Verifier disponibilité salle

```
String query = "SELECT id_salle, nom, capacite, nbr_max_aff " +  
    "FROM salle " +  
    "WHERE (? = 0 OR capacite >= ?) " + // Filtre capacité si renseigné  
    "AND id_salle NOT IN (" +  
    "    SELECT id_salle FROM affectation " +  
    "    WHERE jour = ? " +  
    "    AND ((? >= heure_deb ) " +  
    "    AND (? <= heure_fin) " +  
    "    OR (heure_deb BETWEEN ? AND ?)))";  
  
// 5. Exécution  
connect();  
PreparedStatement pst = con.prepareStatement(query);  
pst.setInt(1, nbEtudiants);  
pst.setInt(2, nbEtudiants);  
pst.setString(3, dateStr);  
pst.setTime(4, Time.valueOf(heureDeb));  
pst.setTime(5, Time.valueOf(heureFin));  
pst.setTime(6, Time.valueOf(heureDeb));  
pst.setTime(7, Time.valueOf(heureFin));  
  
ResultSet rs = pst.executeQuery();
```

# 04

## REALISATION

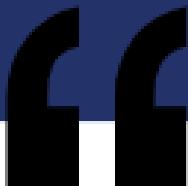
Les requêtes SQL Utilisé en Java:

Afficher Salles:

```
if (selection.equals("Toutes les salles")) {  
    query = "SELECT id_salle, nom, capacite, nbr_max_aff FROM salle";  
  
} else if (selection.equals("Salles par enseignant")) {  
    try {  
        query = "SELECT s.id_salle, s.nom, s.capacite, s.nbr_max_aff, e.nom AS Enseignant "  
        + "FROM salle s "  
        + "JOIN affectation a ON s.id_salle = a.id_salle "  
        + "JOIN enseignant e ON a.id_ens = cin AND e.nom LIKE '%" + champ + "%'";  
  
} else if (selection.equals("Jours libres par salle")) {  
    if(!champ.isEmpty()){  
        try {  
            int idSalle = Integer.parseInt(champ); // Vérifier si c'est un nombre  
            query = "SELECT s.id_salle, s.nom, s.capacite,s.nbr_max_aff, j.jour "  
            + "FROM salle s "  
            + "CROSS JOIN (SELECT 'lundi' AS jour UNION SELECT 'mardi' UNION SELECT 'mercredi' "  
            + "UNION SELECT 'jeudi' UNION SELECT 'vendredi' UNION SELECT 'samedi' UNION SELECT 'dimanche') j "  
            + "LEFT JOIN affectation a ON a.id_salle= s.id_salle "+" AND j.jour = a.jour "  
            + "WHERE a.id_salle IS NULL AND s.id_salle=" + idSalle;  
        } catch (NumberFormatException e) {  
            showError("Veuillez entrer un ID de salle valide !");  
            return;  
        }  
    }else{  
        query = "SELECT s.id_salle, s.nom, s.capacite,s.nbr_max_aff, j.jour "  
        + "FROM salle s "  
        + "CROSS JOIN (SELECT 'lundi' AS jour UNION SELECT 'mardi' UNION SELECT 'mercredi' "  
        + "UNION SELECT 'jeudi' UNION SELECT 'vendredi' UNION SELECT 'samedi' UNION SELECT 'dimanche') j ";  
    }  
} else if (selection.equals("Salle la plus affectée")) {  
    query = "SELECT s.id_salle, s.nom, s.capacite,s.nbr_max_aff, COUNT(a.id_affect) AS Nbr_Affectations "  
    + "FROM salle s "  
    + "LEFT JOIN affectation a ON s.id_salle = a.id_salle "  
    + "GROUP BY s.id_salle "  
    + "ORDER BY Nbr_Affectations DESC LIMIT 1";  
  
Statement sta = con.createStatement();  
rs = sta.executeQuery(query);
```

# 05

## LES INTERFACES



Afficher Salles:

```
// 4. Requête SQL adaptée
String query = "SELECT id_salle, nom, capacite, nbr_max_aff " +
    "FROM salle " +
    "WHERE (capacite >= ?) " + // Filtre capacité si renseigné
    "AND id_salle NOT IN (" +
        " SELECT id_salle FROM affectation " +
        " WHERE jour = ? " +
        " AND ((? >= heure_deb ) " +
        " AND (? <= heure_fin) " +
        " OR (heure_deb BETWEEN ? AND ?)))";
}

// 5. Exécution
connect();
PreparedStatement pst = con.prepareStatement(query);
pst.setInt(1, nbEtudiants);

pst.setString(2, dateStr);
pst.setTime(3, Time.valueOf(heureDeb));
pst.setTime(4, Time.valueOf(heureFin));
pst.setTime(5, Time.valueOf(heureDeb));
pst.setTime(6, Time.valueOf(heureFin));

ResultSet rs = pst.executeQuery();
```

# 04

## REALISATION

Les requêtes SQL Utilisé en Java:

Afficher Affectations:

```
private void loadAffectationData() {
    connect();
    try {
        String query = "SELECT id_affect, id_salle, jour, heure_deb, heure_fin, nbre_etud FROM affectation";
        PreparedStatement pst = conn.prepareStatement(query);
        ResultSet rs = pst.executeQuery();

private void showMostAffectedSalle() {
    connect();
    try {
        String query = "SELECT s.id_salle, s.nom, s.capacite, s.nbr_max_aff, COUNT(*) AS total_affectations " +
            "FROM affectation a " +
            "JOIN salle s ON a.id_salle = s.id_salle " +
            "GROUP BY s.id_salle " +
            "ORDER BY total_affectations DESC LIMIT 1";

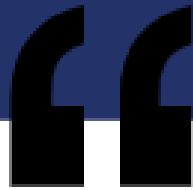
        PreparedStatement pst = conn.prepareStatement(query);
        ResultSet rs = pst.executeQuery();
```

05

# LES INTERFACES

# 05

## LES INTERFACES

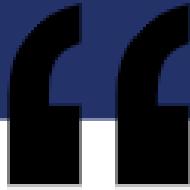


Page se connecter:

The screenshot shows a mobile application interface. On the left, there is a large background image of a modern office interior with desks and chairs. Overlaid on this image is a dark blue rounded rectangle containing the login form. At the top right of the form, it says "APPLICATION D'AFFECTATION DE SALLES TP" and has a small logo. Below this, the word "Crée par:" is followed by the names MOTAZ HEMMAM, MOHAMED AMINE HAJJI, HEND JACOB, and RAYEN AKRICHE. To the right of the names is a "SE CONNECTER" button. Below the button is a text input field labeled "Adresse Email:" with a placeholder box. Below that is another text input field labeled "Mot de passe:" with a placeholder box. At the bottom right of the form is another "SE CONNECTER" button. At the very bottom right of the entire screen is a "S'INSCRIRE" button. A small note at the bottom center of the screen says "Vous n'avez pas de compte ? Inscrivez-vous dès maintenant."

# 05

## LES INTERFACES



Page Inscription:

The image shows a smartphone screen displaying a mobile application interface. The app has a dark blue header with the text "APPLICATION D'AFFECTATION DE SALLES TP" and a logo consisting of a stylized "G" and "V". Below the header is a large white button labeled "S'INSCRIRE". The form consists of five input fields with placeholder text: "CIN:", "NOM:", "PRENOM:", "ADRESSE EMAIL:", and "MOT DE PASSE:". Each field is preceded by its respective label. At the bottom right of the form is a white "S'INSCRIRE" button. At the bottom left of the screen is a red button with the text "PASSEZ L'APPLICATION" and a person icon. At the bottom right is a white button with the text "SE CONNECTER". A small note at the bottom center says "Vous avez déjà un compte ?".

# 05

## LES INTERFACES

“

Page affichage enseignant:

**LISTE DES ENSEIGNANTS EN ACTIVITÉ**

CIN	Nom	Prénom	Spécialité	Heures max/se...
01234587	O'Connor	Sean	Mécanique	11
02345678	Leroy	Sophie	Biologie	10
07654321	Van-Der	Maarten	Optique	10
08765432	Dubois	Annie	Mathémati...	12
09876543	Dupont	Émilie	Physique	9

**FERMER L'APPLICATION** 

**RETOUR**

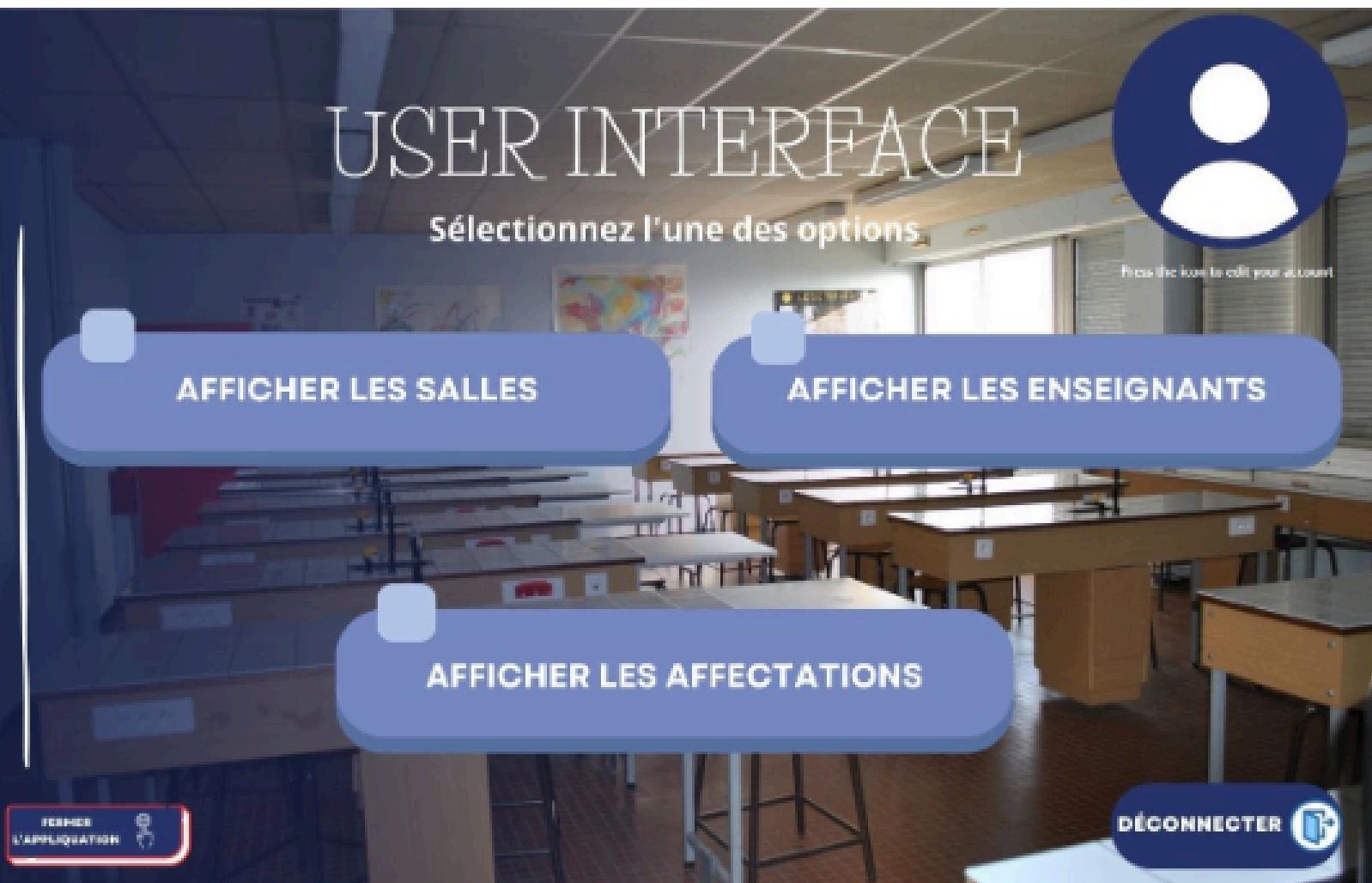
**DÉCONNEXTER** 

# 05

## LES INTERFACES

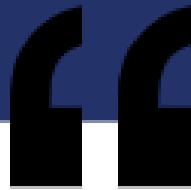
“

Page interface user



# 05

## LES INTERFACES



Page afficher Salle:

### AFFICHER LES SALLES

ID Salle	Nom	Capacité	Nbr aff max	Détails
5	S3	35	25	-
6	S4	40	30	-
7	SE5A	50	35	-
8	LAB1	20	15	-
9	dssdv	12	20	-

**CONFIRMER**

Toutes les salles

Entrer nbr Etudiant:

Entrez Date:

Heure deb:

Heure fin:

**RETOUR**

**DÉCONNECTER**

# 05

## LES INTERFACES

“

Page afficher affectations

### LISTE DES AFFECTATIONS

Num Affectat...	Num Salle	Jour	Heure Déb	Heure Fin	Nb Étudiants
1	5	lundi	08:00:00	10:00:00	30
2	6	mardi	10:00:00	12:00:00	25
3	7	mercredi	13:00:00	15:00:00	40
4	8	jeudi	09:00:00	11:00:00	12
5	9	vendredi	14:00:00	16:00:00	70

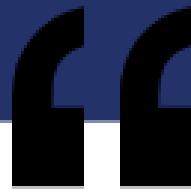
FERMER  
L'APPLICATION

RETOUR

DÉCONNECTER

# 05

## LES INTERFACES

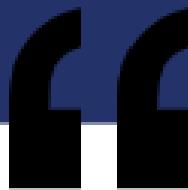


Page Interface Admin:

The screenshot shows a login interface for an 'ADMIN INTERFACE'. The background is a photograph of a classroom with desks and chairs. At the top, there are three gear icons (yellow, orange, blue) on the left and a large pink gear icon with a user profile icon inside on the right. The title 'ADMIN INTERFACE' is centered at the top in a large, serif font. Below it, the text 'Sélectionnez l'une des options.' (Select one of the options) is displayed. There are five blue, rounded rectangular buttons containing white text: 'GÉRER SALLE' (Manage Room), 'GÉRER LES ENSEIGNANTS' (Manage Teachers), 'GÉRER LES MAINTENANCES' (Manage Maintenance), 'GÉRER LES AFFECTATIONS' (Manage Assignments), and 'GÉRER LES UTILISATEURS' (Manage Users). In the bottom left corner, there is a red button labeled 'FERMER L'APPLICATION' (Close Application) with a small window icon. In the bottom right corner, there is a blue button labeled 'DÉCONNECTER' (Disconnect) with a small user icon.

# 05

## LES INTERFACES



Page gerer enseignant:

### GÉRER LES ENSEIGNANTS

CIN	Nom	Prénom	Spécialité	HeuresM...
01234567	O'Connor	Sean	Mécanique	11
02345678	Leroy	Sophie	Biologie	10
07654321	Van Der	Maarten	Optique	10
08705432	Dubois	Annie	Mathématiques	12

SUPPRIMER

RÉINITIALISER

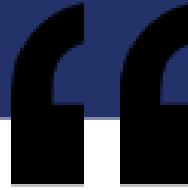
RETOUR

DÉCONNECTER

FERMER L'APPLICATION

# 05

## LES INTERFACES



Page gerer salle:

### GÉRER LES SALLES

ID Salle	Nom	Capacité	Nbr aff max
5	S3	35	25
6	S4	40	30
7	SESA	50	35
8	LAB1	20	15

**MODIFIER** **AJOUTER**

ID SALLE:

NOM:

CAPACITE:

AFFECTATIONS MAX:

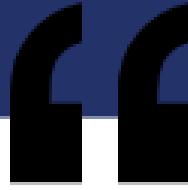
**SUPPRIMER** **RÉINITIALISER**

**RETOUR** **DÉCONNECTER**

**TERMINER L'APPLICATION**

# 05

## LES INTERFACES



Page gerer maintenance:

### GÉRER LES MAINTENANCES

ID Maintenance	Jour	ID Salle
1	lundi	5
2	mardi	6
4	jeudi	8
6	lundi	10

CLIQUEZ SUR UNE CASE PUIS "MODIFIER" POUR MODIFIER UNE CASE

**AJOUTER UNE MAINTENANCE**

ID SALLE:

JOUR:

**OK**

**SUPPRIMER**

**RETOUR**

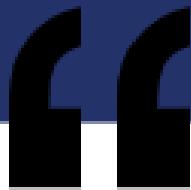
**DÉCONNECTER**

FERMER L'APPLICATION

9

# 05

## LES INTERFACES



Page se connecter:

### GÉRER LES AFFECTATIONS

id_aff	id_ens	id_salle	jour	heure_début	heure_fin	nbre_et...
1	09876543	5	lundi	08:00:00	10:00:00	30
2	11223344	6	mardi	10:00:00	12:00:00	25
3	02345678	7	mercredi	13:00:00	15:00:00	40
4	14567892	8	jeudi	09:00:00	11:00:00	12

**AJOUTER UNE AFFECTATION**

CIN:

ID SALLE:

JOUR:  lundi

HEURE DÉBUT:

HEURE FIN:

NB ÉTUDIANTS:

**AJOUTER**

**MODIFIER**

**SUPPRIMER**

CLIQUEZ SUR UNE CASE PUIS "MODIFIER" POUR MODIFIER UNE CASE

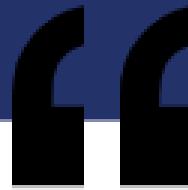
**RETOUR**

**DÉCONNECTER**

**FERMER L'APPLICATION**

# 05

## LES INTERFACES



Page gestion user:

**GÉRER LES UTILISATEURS**

Cin	Nom	Prénom	Email	MDP	Role	Date i...
012376...	Müller	Hans	h.mulle...	B3rlin2...	user	2025-0...
023498...	haji	moha...	amine...	Vietna...	admin	2025-0...
076543...	Dubois	Pierre	p.duboi...	Paris12...	user	2025-0...
087654...	Gómez	Ana	ana.go...	F10w3r...	user	2025-0...

**MODIFIER**   **AJOUTER**

**CIN:**

**NOM:**

**PRÉNOM:**

**EMAIL:**

**MDP:**

**ROLE:**  admin

**SUPPRIMER**   **RÉINITIALISER**

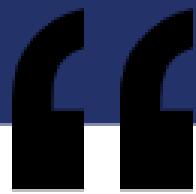
**RETOUR**

**DÉCONNECTER**

**FERMER L'APPLICATION**

# 05

## LES INTERFACES

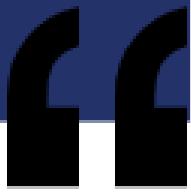


Page modifier le compte:



# 05

## LES INTERFACES



Changer le mot de pass:

### MODIFIER VOTRE MOT DE PASSE

ANCIEN MOT DE PASSE :

NOUVEAU MOT DE PASSE :

CONFIRMER

FERMER L'APPLICATION

REVENIR EN ARRIER

# Pour plus d'informations

Lien Github pour le code original du projet:

<https://github.com/RayenAkrich/Application-d-affectation-de-salles-de-TP>

Notre Contact :

akricherayen@gmail.com

amine.hajji@etudiant-fst.utm.tn

hend.yacoub@etudiant-fst.utm.tn

mootaz.hamam@gmail.com