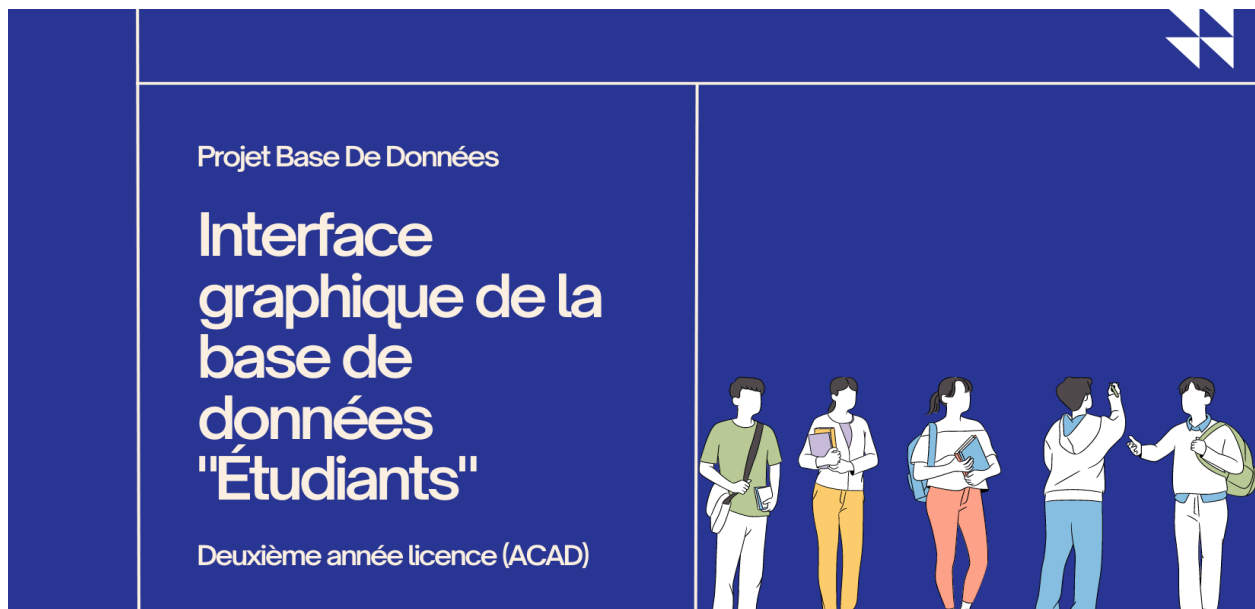


Rapport de projet Base De Données (BDD)

Fait par:

Matricule: 212132032981 | Nom: BENSAFIA | Prenom: Younes | Section: A | Groupe: 04.

Matricule: 202032021716 | Nom: BARAGH | Prenom: Yasser Abdeldjalil | Section: A | Groupe: 04.



Année universitaire 2022/2023

1. Introduction:

Dans ce Projet on va créer une interface [Java Swing](#) en utilisant l'IDE [NetBeans](#) pour la gestion des étudiants et enseignants. Cette interface permet de gérer les informations des étudiants et enseignants telles que les noms, les prénoms,...etc. Avec cette interface, il est possible d'ajouter, de supprimer, de modifier et de rechercher des informations sur les étudiants et enseignants de manière conviviale et intuitive.



L'interface [Java Swing](#) est une bibliothèque graphique pour le développement d'applications de bureau en Java. Elle permet de créer des interfaces graphiques utilisateur ([GUI](#)) avec des composants tels que des boutons, des champs de texte, des listes, des tableaux, des menus, etc. Elle est facile à utiliser et à personnaliser, ce qui facilite la création d'interfaces utilisateur conviviales et esthétiques.

En utilisant cette interface [Java Swing](#) pour la gestion des étudiants et enseignants, on peut améliorer l'efficacité et la précision de la gestion des données, ainsi que la convivialité de l'interface utilisateur. Cette interface peut être utilisée dans les établissements d'enseignement pour la gestion des données des étudiants et enseignants.

2. Développement:

a) Partie Base De Données:

Pour débiter, la première étape a été la création de notre base de données.

- i) **Les Tables:** On a quatre tables dans notre Base De Données (Etudiant, Enseignant, Unité, EtudiantUnité). (Voir les figure ci-dessous).

```
create table Etudiant
(
    matricule_etu varchar2(20),
    nom_etu       varchar2(10),
    prenom_etu    varchar2(10),
    date_naissance varchar2(20),
    constraint PK_Etudiant primary key(matricule_etu)
);
```

```
create table Enseignant
(
    matricule_ens varchar2(20),
    nom_ens       varchar2(20),
    prenom_ens    varchar2(20),
    age           varchar2(20),
    constraint PK_Enseignant primary key(matricule_ens)
);
```

```
create table Unite
(
    code_unite varchar2(20),
    libelle varchar2(20),
    nbr_heures varchar2(20),
    matricule_ens varchar2(20),
    constraint PK_Unite primary key(code_unite),
    constraint FK_Unite foreign key(matricule_ens)
    references Enseignant (matricule_ens)
);
```

```
create table EtudiantUnite
(
    matricule_etu number(8),
    code_unite varchar2(20),
    note_CC INTEGER,
    note_TP INTEGER,
    note_ex INTEGER,
    constraint PK_ETU primary key(code_unite, matricule_etu),
    constraint FK_ETU foreign key(matricule_etu)
    references Etudiant (matricule_etu),
    constraint FK_ETU2 foreign key(code_unite)
    references Unite (code_unite)
);
```

```
/*UNE TABLE "USERS" CONTIENT TOUS LES UTILISATEURS UTILISEES DANS NOTRE BDD*/  
CREATE TABLE Users (  
    id Varchar2(50) PRIMARY KEY,  
    pswd Varchar2(255),  
    typeDeCompte Varchar2(2)  
);
```

ii) **Les Utilisateurs**: On a trois Utilisateurs (Etudiant, Enseignant, BDDAdmin)

1. **Pour l'utilisateur "Etudiant" (étudiant) :**

- Un utilisateur nommé "Etudiant" est créé avec le mot de passe "TPEtudiant".
- Le privilège "create session" est accordé à l'utilisateur "Etudiant", ce qui lui permet de se connecter à la base de données.
- Le privilège "SELECT" est accordé à l'utilisateur "Etudiant" sur la table "Etudiant" dans la base de données "BDDAdmin". Cela lui permet de récupérer des données depuis cette table.

2. **Pour l'utilisateur "Enseignant" (enseignant) :**

- Un utilisateur nommé "Enseignant" est créé avec le mot de passe "TPEnseignant".
- Le privilège "create session" est accordé à l'utilisateur "Enseignant", ce qui lui permet de se connecter à la base de données.
- Le privilège "INSERT" est accordé à l'utilisateur "Enseignant" sur la table "Enseignant". Cela lui permet d'insérer des données dans cette table.
- Le privilège "SELECT" est accordé à l'utilisateur "Enseignant" sur la table "Enseignant". Cela lui permet de récupérer des données depuis cette table.

3. **Pour l'utilisateur "BDDAdmin" (admin) :**

- Il a tous les privilèges (INSERT, SELECT, DELETE,...etc) sur toutes les tables qui existent dans la base de données.

iii) **Les Insertions:** Voir la figure ci-dessous.

```
INSERT INTO Users values('BDDAdmin','TPAdmin','AD');
INSERT INTO Users values('Etudiant','TPEtudiant','ET');
INSERT INTO Users values('Enseignant','TPEnseignant','EN');

alter table Etudiant add(Adresse Varchar2(100));
alter table Enseignant drop(age);
alter table Etudiant add constraint check_MatE check(matricule_etu>'20190000' and matricule_etu <'20199999');
alter table Etudiant modify(prenom_etu varchar2(25));

INSERT INTO Etudiant VALUES (20190001, 'BOUSSAI', 'MOHAMED', TO_DATE('12-01-2000', 'DD-MM-YYYY'), 'Alger');
INSERT INTO Etudiant VALUES (20190002, 'CHAID', 'LAMIA', TO_DATE('01-10-1999', 'DD-MM-YYYY'), 'Batna');
INSERT INTO Etudiant VALUES (20190003, 'BRAHIMI', 'SOUAD', TO_DATE('18-11-2000', 'DD-MM-YYYY'), 'Setif');
INSERT INTO Etudiant VALUES (20190004, 'LAMA', 'SAID', TO_DATE('23-05-1999', 'DD-MM-YYYY'), 'Oran');

INSERT INTO Enseignant values(20000001,'HAROUNI', 'AMINE');
INSERT INTO Enseignant values(19990011,'FATHI','OMAR');
INSERT INTO Enseignant values(19980078,'BOUZIDANE', 'FARAH');
INSERT INTO Enseignant values(20170015,'ARABI','ZOUBIDA');

INSERT INTO Unite values('FEI0001', 'POO', 6, 20000001);
INSERT INTO Unite values('FEI0002','BDD',6,19990011);
INSERT INTO Unite values('FEI0003','RESEAU', 3 ,20170015);
INSERT INTO Unite values('FEI0004','SYSTEME', 6 ,19980078);

INSERT INTO EtudiantUnite values(20190001,'FEI0001', 10, 15, 9);
INSERT INTO EtudiantUnite values(20190002,'FEI0001',20,13,10);
INSERT INTO EtudiantUnite values(20190004,'FEI0001',13, 17, 16);
INSERT INTO EtudiantUnite values(20190002,'FEI0002',10, 16, 17);
INSERT INTO EtudiantUnite values(20190003,'FEI0002',9, 8, 15);
INSERT INTO EtudiantUnite values(20190004,'FEI0002',15, 9, 20);
INSERT INTO EtudiantUnite values(20190002,'FEI0004',12, 18, 14);
INSERT INTO EtudiantUnite values(20190003,'FEI0004',17, 12, 15);
INSERT INTO EtudiantUnite values(20190004,'FEI0004',12, 13, 20);
```

Après toutes les insertions on a ajouté l'instruction "commit" pour confirmer les modifications.

```
commit;
```

iv) **Les "UPDATE":**

- Augmenter la note_CC de 2 pour tous les étudiants dont le nom commence par 'B'.
- Remettre toutes les notes d'examen de l'unité "SYSTEME" à 0 pour tous les étudiants.

```
UPDATE EtudiantUnite SET note_ex = 0 where code_unite in ( SELECT code_unite from unite where libelle='SYSTEME');
UPDATE etudiantunite SET note_TP = note_TP + 2 where matricule_etu IN (SELECT matricule_etu FROM etudiant WHERE nom_etu LIKE 'B%');
```

v) Les Requêtes SQL:

```
/*Afficher les noms et prénoms des étudiants ayant obtenus des notes d'examens égales à 20.*/
SELECT
    nom_etu,
    prenom_etu
FROM
    etudiant
where
    matricule_etu in (
        SELECT
            matricule_etu
        FROM
            EtudiantUnite
        WHERE
            note_ex = 20
    )
```

```
/*Afficher les noms et prénoms des étudiants qui ne sont pas inscrits dans l'unité « POO ».*
SELECT
    nom_etu,
    prenom_etu
FROM
    Etudiant MINUS
SELECT
    nom_etu,
    prenom_etu
FROM
    Etudiant
where
    matricule_etu in (
        SELECT
            matricule_etu
        FROM
            EtudiantUnite
        where
            code_unite in (
                SELECT

```

```
/*Afficher les libellés des unités d'enseignement dont aucun étudiant n'est inscrit.*/
SELECT
    libelle
FROM
    Unite
WHERE
    code_Unite NOT IN (
        SELECT
            code_Unite
        FROM
            EtudiantUnite
    )
```

```

/*Afficher pour chaque étudiant, son nom, son prénom sa moyenne par unité d'enseignement ainsi que le libellé de l'unité.*/
SELECT
  e.nom_etu,
  e.prenom_etu,
  u.libelle,
  AVG((eu.note_CC + eu.note_TP + eu.note_ex) / 3)
FROM
  Etudiant e,
  EtudiantUnite eu,
  Unite u
WHERE
  e.matricule_etu = eu.matricule_etu
  AND eu.code_unite = u.code_unite
GROUP BY
  e.matricule_etu,
  e.nom_etu,
  e.prenom_etu,
  u.libelle;

```

b) Partie Code:


i) Partie “Login”:

LOGIN

User

Password

Login



Dans cette partie le code contient 3 fonctions nécessaires et un constructeur:

- jToggleButton1ActionPerformed
- BDDFrameTous
- Main

1) jToggleButton1ActionPerformed:

```
private void jToggleButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    DBConnection conn = new DBConnection();  
    Users userlogin=new Users( id: User.getText(),pswd: jPasswordField1.getText());  
    conn.connexion();  
    String query = "SELECT * FROM bddadmin.users WHERE id = ? and pswd = ? ";  
    try {  
        PreparedStatement statement = conn.getNewConnection().prepareStatement( string: query)) {  
            statement.setString(1, string: userlogin.getId());  
            statement.setString(2, string: userlogin.getPswd());  
  
            ResultSet resultSet = statement.executeQuery();  
  
            if (resultSet.next()){  
                if (userlogin.getId().equals( anObject: resultSet.getString(1)) && userlogin.getPswd().equals( anObject: resultSet.getString(2))) {  
                    userlogin.setType( type: resultSet.getString(3));  
                    conn.deconnexion();  
                    if (userlogin.getType().equals( anObject: "AD")){  
                        this.setVisible( b:false);  
                        BDDAdminCeQuiVeut2.BDDFirstFrame();  
                        System.out.println( "Login Successful");  
                    }  
                    else  
                    {  
                        if(userlogin.getType().equals( anObject: "ET")){  
                            this.setVisible( b:false);  
                            PourConnDeEtudiant.frameEtudiant1();  
                        }  
                        else  
                        {  
                            if(userlogin.getType().equals( anObject: "EN"))  
                            {  
                                this.setVisible( b:false);  
                                ConsultationInsertionEns.EnseignantFrame();  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
    }else{  
        erreurLabel.setText( text: "Mot de Passe ou Utilisateur incorrecte.");  
        System.out.println( "Error");  
    }  
}
```



```

    }
    }else{
        erreurLabel.setText(text: "Mot de Passe ou Utilisateur incorrecte.");
        System.out.println(x: "Error");
    }
}
catch (SQLException e) {
    e.printStackTrace();
    System.out.println(x: "Login failed");
    erreurLabel.setText(text: "Mot de Passe ou Utilisateur incorrecte.");
    System.out.println(x: "Error");
}
}
}

```

c'est cette méthode (fonction) qui est responsable du **login** des utilisateurs (**BDDAdmin**, **Enseignant**, **Étudiant**). Une instance de la classe **DBConnection** est créée pour gérer la connexion à la base de données. Un nouvel objet **Users** est créé en utilisant les valeurs fournies par l'utilisateur pour le **nom d'utilisateur** et le **mot de passe**.

La connexion à la base de données est établie en appelant la méthode **connexion()** de l'objet **conn**. Une requête **SQL** est préparée pour sélectionner toutes les lignes de la table "**users**" où l'**ID** correspond à celui fourni par l'utilisateur et le **mot de passe** correspond à celui fourni par l'utilisateur. Les valeurs des paramètres de la requête sont définies à l'aide des méthodes **setString()** de l'objet **statement**. La requête est exécutée en appelant **executeQuery()** sur l'objet **statement**, et le résultat est stocké dans un objet **ResultSet**. Si le résultat contient une ligne, cela signifie que l'authentification a réussi. Les valeurs de l'**ID** utilisateur, du mot de passe et du type d'utilisateur sont extraites du **ResultSet**. La connexion à la base de données est fermée en appelant la méthode **deconnexion()** de l'objet "**conn**".

En fonction du type d'utilisateur, une action appropriée est effectuée :

Si le type d'utilisateur est "**AD**" (administrateur), la fenêtre actuelle est masquée et une nouvelle fenêtre **BDDFirstFrame()** est affichée.

Si le type d'utilisateur est "**ET**" (étudiant), la fenêtre actuelle est masquée et une nouvelle fenêtre **frameEtudiant1()** est affichée.

Si le type d'utilisateur est "EN" (enseignant), la fenêtre actuelle est masquée et une nouvelle fenêtre **EnseignantFrame()** est affichée.

Si le résultat ne contient aucune ligne, cela signifie que l'authentification a échoué. Un message d'erreur est affiché à l'utilisateur

2) **BDDFrameTous:**

```
public static void BDDFrameTous() {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            new BDDLoginTousLEMonde().setVisible(true);  
        }  
    });  
}
```

La méthode **BDDFrameTous()** est une méthode statique qui lance une nouvelle instance de la classe **BDDLoginTousLEMonde** et l'affiche à l'écran. La méthode **BDDFrameTous()** est appelée pour créer et afficher une nouvelle fenêtre de connexion à la base de données.

L'appel à **java.awt.EventQueue.invokeLater()** garantit que la création et l'affichage de la fenêtre sont effectués dans le thread de l'interface utilisateur pour éviter les problèmes de concurrence.

À l'intérieur de la méthode **run()** de l'objet **Runnable**, une nouvelle instance de **BDDLoginTousLEMonde** est créée.

Enfin, la méthode **setVisible(true)** est appelée pour rendre la fenêtre visible à l'utilisateur.

3) **le Main:** C'est le Main de la classe **BDDLoginTousLeMonde**.

```
public static void main(String[] args) {  
    BDDFrameTous();  
}
```

4) Le Constructeur:

```
public BDDLoginTousLEMonde() {  
    initComponents();  
}
```

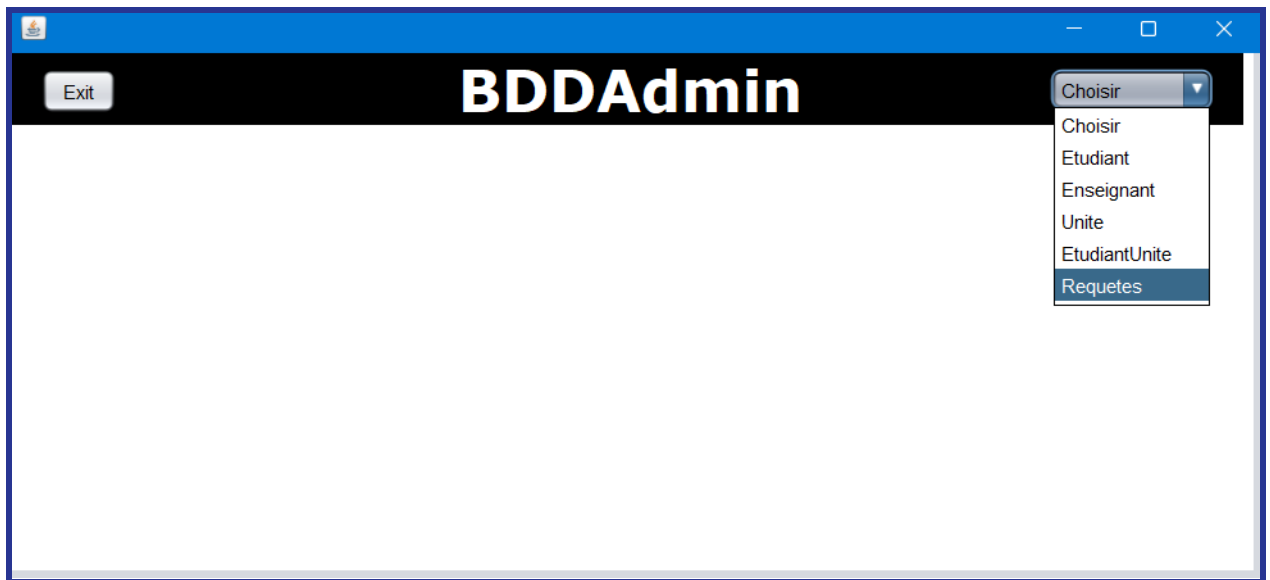
La méthode `initComponents()` est responsable de l'initialisation des composants de l'interface utilisateur de la fenêtre, tels que les boutons, les champs de texte, les étiquettes,...etc.

5) Les Bibliothèques importées:

```
package javaapplication1;  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
/**
```

ii) Les Utilisateur et leurs interface:

1) BDDAdmin:



- (a) **Choisir:** il ne va rien afficher.
- (b) **Etudiant:** Dans cette Interface le BDDAdmin pourra insérer, supprimer un étudiant ou consulter tous les étudiants.

The screenshot shows a window titled "BDDAdmin" with a blue header bar. On the left of the header is an "Exit" button, and on the right is a dropdown menu currently set to "Etudiant". Below the header is a form with five input fields: "Matricule", "Nom", "Prénom", "Date de Naissance" (with a calendar icon), and "Adresse". A green "Ajouter" button is centered below these fields. Below the form is a section with three buttons: "Annuler", "Supprimer Etudiant" (with a text input field next to it), and "reduire". At the bottom is a table with the following data:

Matricule	Nom	Prénom	Date De Naissance	Adresse
20190001	BOUSSAI	MOHAMED	2000-01-12	Alger
20190002	CHAID	LAMIA	1999-10-01	Batna
20190003	BRAHIMI	SOUAD	2000-11-18	Setif
20190004	LAMA	SAID	1999-05-23	Oran

- (i) Le Code: pour cela on a créé une classe supplémentaire "Etudiant" et contient les classes ci-dessous.

```
public class Etudiant {
    private int matricule;
    private String nom;
    private String prenom;
    private java.util.Date dateNaissance;
    private String adresse;
    private DBConnection connectionEtu;

    //getters & setters
    public String getAdresse() {
        return adresse;
    }

    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }

    public int getMatricule() {
        return matricule;
    }

    public Etudiant(int matricule) {
        this.matricule = matricule;
        connectionEtu = new DBConnection ();
    }

    public void setMatricule(int matricule) {
        this.matricule = matricule;
    }

    public String getNom () {
        return nom;
    }

    public void setNom (String nom) {
        this.nom = nom;
    }

    public String getPrenom () {
        return prenom;
    }

    public void setPrenom (String prenom) {
        this.prenom = prenom;
    }
}
```

```
//Constructeurs
public Etudiant(int matricule, String nom, String prenom, java.util.Date dateNaissance, String adresse) {

    this.matricule = matricule;
    this.nom = nom;
    this.prenom = prenom;
    this.dateNaissance = dateNaissance;
    this.adresse = adresse;
    connectionEtu = new DBConnection ();

}

public Etudiant (String nom, String prenom) {
    this.nom = nom;
    this.prenom = prenom;
    connectionEtu = new DBConnection ();
}
}
```

//Pour inserer un etudiant

```
public void insererEtudiant() {

    String query = "INSERT INTO BDDAdmin.Etudiant VALUES (?, ?, ?, ?, ?)";

    connectionEtu.connexion();
    try (PreparedStatement statement = connectionEtu.getNewConnection().prepareStatement(string: query)) {
        statement.setInt(1, this.getMatricule());
        statement.setString(2, string: this.getNom());
        statement.setString(3, string: this.getPrenom());
        statement.setDate(4, new java.sql.Date (date: this.getDateNaissance().getTime()));
        statement.setString(5, string: this.getAdresse());
        statement.executeUpdate();
        connectionEtu.deconnexion();
        System.out.println("ajoute succes ");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

// Supprime un étudiant de la table en utilisant le matricule comme clé

```
public void supprimerEtudiant() {
    String query = "DELETE FROM Etudiant WHERE matricule_etu = ?";
    connectionEtu.connexion();
    try (PreparedStatement statement = connectionEtu.getNewConnection().prepareStatement(string: query)) {
        statement.setInt(1, this.matricule);
        statement.executeUpdate();
        connectionEtu.deconnexion();
        System.out.println("supprime succes ");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

// Afficher le tableau des etudiant

```
public Etudiant() {connectionEtu = new DBConnection ();}
public List<Etudiant> consultationEtudiants() throws SQLException {
    List<Etudiant> etudiants = new ArrayList<>();
    String query = "SELECT * FROM BDDAdmin.Etudiant";
    connectionEtu.connexion();
    try (Statement statement = connectionEtu.getNewConnection().createStatement()) {
        ResultSet resultSet = statement.executeQuery(string: query);
        while (resultSet.next()) {
            int matricule = resultSet.getInt(string: "matricule_etu");
            String nom = resultSet.getString(string: "nom_etu");
            String prenom = resultSet.getString(string: "prenom_etu");
            Date dateNaissance = resultSet.getDate(string: "date_naissance");
            String adresse = resultSet.getString(string: "adresse");
            Etudiant etudiant = new Etudiant(matricule, nom, prenom, dateNaissance, adresse);
            etudiants.add(etudiant);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        connectionEtu.deconnexion(); // Close the connection outside the while loop
    }
    return etudiants;
}
```

//Affichage d'un etudiant

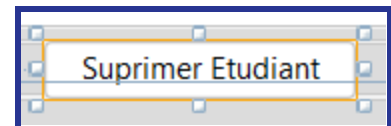
```
public void afficherEtudiant() {
    System.out.print("Matricule : " + matricule + " ");
    System.out.print("Nom : " + nom + " ");
    System.out.print("Prénom : " + prenom + " ");
    System.out.print("Date de naissance : " + dateNaissance + " ");
    System.out.println("Adresse : " + adresse + " ");
}
```

Après dans le code de notre interface on fait appelle a chaque fonction de cette classe.
L'insertion: lorsqu'il clique sur "Ajouter" → c'est le code ci-dessous qui va s'exécuter.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    Etudiant et1 = new Etudiant( matricule: Integer.parseInt(s:MatriculeEtudiant.getText()),nom: nomEtudiant.getText(),prenom: prenomEtudiant.getText(), dateNaissance:dateDeNaissanceEtudiant.getDa  
    et1.insererEtudiant();  
    try {  
        afficheEtudiant();  
    } catch (SQLException ex) {  
        Logger.getLogger( name: BDDAdminCeQuiVeut2.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);  
    }  
}
```

La suppression: lorsqu'il clique sur "Supprimer Etudiant" → c'est le code ci-dessous qui va s'exécuter.



```
private void supprimerEtudiantActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Etudiant Etu_supp = new Etudiant ( matricule: Integer.parseInt(s:MatriculeEtudiant1.getText()));  
    Etu_supp.supprimerEtudiant();  
    try {  
        afficheEtudiant();  
    } catch (SQLException ex) {  
        Logger.getLogger( name: BDDAdminCeQuiVeut2.class.getName()).log( level: Level.SEVERE, msg: null, thrown: ex);  
    }  
}
```

L'affichage dans le tableau ci-dessous:

Matricule	Nom	Prenom	Date De Naissance	Adresse

```

/* Afficher Etudiant */
public void afficheEtudiant() throws SQLException{

    DefaultTableModel tableModel = new DefaultTableModel();

    // Ajouter des colonnes
    tableModel.addColumn(columnName: "Matricule");
    tableModel.addColumn(columnName: "Nom");
    tableModel.addColumn(columnName: "Prénom");
    tableModel.addColumn(columnName: "Date De Naissance");
    tableModel.addColumn(columnName: "Adresse");

    jTable1.setModel(dataModel: tableModel);

    // Creation d'une liste de data
    Etudiant etudiants = new Etudiant();

    // Ajouter chaque élément de la liste en tant que nouvelle ligne dans le modèle de table
    for (Etudiant rowData : etudiants.consultationEtudiants()) {
        tableModel.addRow(rowData: rowData.toVector());
    }
    jTable1.setModel(dataModel: tableModel);

    jTable1.setModel(dataModel: tableModel);

    jTable1.setVisible(aFlag: true);
}

```

(c) Enseignant & EtudiantUnite & Unite:

Matricule	Nom	Prénom
20000001	HAROUNI	AMINE
19990011	FATHI	OMAR
19980078	BOUZIDANE	FARAH
20170015	ARABI	ZOUBIDA
20170002	BENSAFIA	Djilali

dans Enseignant on a raisonner de la même façon qu'Étudiant.
On a créer un autre classe (supplumainter) contient:

```
public class Enseignant {
    private int matricule;
    private String nom;
    private String prenom;
    private DBConnection connectionEns;
    //Constructeur
    public Enseignant(int matricule, String nom, String prenom) {...6 lines }

    public Enseignant() {...3 lines }

    public Enseignant(int matricule) {...4 lines }
    //Getters & Setters
    public int getMatricule() {...3 lines }

    public void setMatricule(int matricule) {...3 lines }

    public String getNom () {...3 lines }

    public void setNom (String nom) {...3 lines }

    public String getPrenom () {...3 lines }

    public void setPrenom (String prenom) {...3 lines }

    public DBConnection getConnectionEns() {...3 lines }

    public void setConnectionEns(DBConnection connectionEns) {...3 lines }
```

//Insérer Enseignant

```
public void insererEnseignant() {
    // Insère un nouvel enseignant dans la table
    String query = "INSERT INTO BDDAdmin.Enseignant VALUES (?, ?, ?)";
    connectionEns.connexion();
    try (PreparedStatement statement = connectionEns.getNewConnection().prepareStatement(string: query)) {
        statement.setInt(1, this.getMatricule());
        statement.setString(2, string: this.getNom());
        statement.setString(3, string: this.getPrenom());
        statement.executeUpdate();
        connectionEns.deconnexion();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

//Supprimer Enseignant

```
public void supprimerEnseignant() {
    // Supprime un enseignant de la table en utilisant le matricule comme clé
    String query = "DELETE FROM BDDAdmin.Enseignant WHERE matricule_ens = ?";
    connectionEns.connexion();
    try (PreparedStatement statement = connectionEns.getNewConnection().prepareStatement(string: query)) {
        statement.setInt(1, this.matricule);
        statement.executeUpdate();
        connectionEns.deconnexion();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

//Liste des Enseignants avec SELECT.

```
public List<Enseignant> consultationEnseignants() throws SQLException {
    // Récupère tous les enseignants de la table
    List<Enseignant> enseignants = new ArrayList<>();
    String query = "SELECT * FROM BDDAdmin.Enseignant";
    connectionEns.connexion();
    try (Statement statement = connectionEns.getNewConnection().createStatement()) {
        ResultSet resultSet = statement.executeQuery(string: query);
        while (resultSet.next()) {
            int matricule = resultSet.getInt(string: "matricule_ens");
            String nom = resultSet.getString(string: "nom_ens");
            String prenom = resultSet.getString(string: "prenom_ens");
            Enseignant enseignant = new Enseignant(matricule, nom, prenom);
            enseignants.add(enseignant);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        connectionEns.deconnexion(); // Ferme la connexion à l'extérieur de la boucle while
    }
}
```

```
//Recherche d'un enseignant
public Enseignant recherche() throws SQLException, NullPointerException {
    String query = "SELECT * FROM BDDAdmin.Enseignant WHERE matricule_ens = ?";
    Enseignant enseignant = null;
    connectionEns.connexion();
```

```
try (PreparedStatement statement = connectionEns.getNewConnection().prepareStatement(string: query)) {
    statement.setInt(1, this.getMatricule());
    ResultSet resultSet = statement.executeQuery();

    if (resultSet.next()) {
        enseignant = new Enseignant(
            matricule: resultSet.getInt(string: "matricule_ens"),
            nom: resultSet.getString(string: "nom_ens"),
            prenom: resultSet.getString(string: "prenom_ens")
        );
    }
} catch (SQLException e) {
    if (e instanceof SQLIntegrityConstraintViolationException) {
        // Gérer l'exception de violation de contrainte d'intégrité
    } else if (e instanceof SQLDataException) {
        System.out.println("Enseignant n'existe pas");
    }
} catch (NullPointerException e) {
} finally {
    connectionEns.deconnexion();
    return enseignant;
}
```

//To Vector pour Enseignant

```
public Vector<String> toVector() {
    Vector<String> vector = new Vector<>();
    vector.add(String.valueOf(this.matricule));
    vector.add(this.nom);
    vector.add(this.prenom);
    return vector;
}
```


même chose pour Unité et EtudiantUnité les mêmes fonctions, même raisonnement. Avec les classe supplémentaire (Unité, EtudiantUnité)

Code Unité	Libelle	Nombre Heures	Matricule Enseignant
FEI0001	POO	6	20000001
FEI0002	BDD	6	19990011
FEI0003	RESEAU	3	20170015
FEI0004	SYSTEME	6	19980078
FEI0005	THL	3	20170015

Matricule Etudiant	Code Unité	Note CC	Note TP	Note EX
20190001	FEI0001	10	17	9
20190004	FEI0001	13	17	16
20190002	FEI0002	10	16	17
20190003	FEI0002	9	10	15
20190004	FEI0002	15	9	20

```
package javaapplication1;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;

public class Unite {
    private String codeUnite;
    private String libelle;
    private String nbrHeures;
    private int matriculeEns;
    private DBConnection connection;

    public DBConnection getConnection() {
        return connection;
    }

    public void setConnection(DBConnection connection) {
        this.connection = connection;
    }

    // Constructeur
    public Unite(String codeUnite, String libelle, String nbrHeures, int matriculeEns) {
        this.codeUnite = codeUnite;
        this.libelle = libelle;
        this.nbrHeures = nbrHeures;
        this.matriculeEns = matriculeEns;
        this.connection = new DBConnection();
    }

    public Unite() {
        this.connection = new DBConnection();
    }

    public Unite(String codeUnite) {
        this.codeUnite = codeUnite;
        this.connection = new DBConnection();
    }

    // Getters et setters
    public String getCodeUnite() {
        return codeUnite;
    }

    public void setCodeUnite(String codeUnite) {
        this.codeUnite = codeUnite;
    }
}
```

```
Unité.java x EtudiantUnité.java x
Source History

1 //
2
3
4
5 package javaapplication1;
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9 import java.sql.Date;
10 import com.toedter.calendar.JDateChooser;
11 import java.util.Calendar;
12 import com.sun.jdbc.connect.spi.Connection;
13 import java.util.Vector;
14
15 /**
16  *
17  * @author chems
18  */
19
20 public class EtudiantUnité {
21     private int matriculeEtu;
22     private String codeUnite;
23     private int noteCC;
24     private int noteTP;
25     private int noteEx;
26     private DBConnection connection;
27
28     EtudiantUnité() {
29         connection = new DBConnection ( );
30     }
31
32     EtudiantUnité(int matriculeEtu, String codeUnite) {
33         this.matriculeEtu = matriculeEtu;
34         this.codeUnite = codeUnite;
35         connection = new DBConnection ( );
36     }
37
38     public DBConnection getConnection() {
39         return connection;
40     }
41
42     public void setConnection(DBConnection connection) {
43         this.connection = connection;
44     }
45
46     // Constructeur
47     public EtudiantUnité(int matriculeEtu, String codeUnite, int noteCC, int noteTP, int noteEx) {
48         this.matriculeEtu = matriculeEtu;
49         this.codeUnite = codeUnite;
50         this.noteCC = noteCC;
51         this.noteTP = noteTP;
52     }
53 }
```

(d) Les Requêtes:

On a créé une classe Requête qui contient les deux fonctions pour les deux requêtes.

```
public class Requetes {
    //Connection
    private DBConnection connection = new DBConnection();
    //La premier requete retourne une Liste d'Etudiant
    public List<Etudiant> requete1() throws SQLException {
        List<Etudiant> etudiants = new ArrayList<>();
        String query = "SELECT nom_etu, prenom_etu FROM etudiant where matricule_etu in (SELECT matricule_etu FROM EtudiantUnité WHERE note_ex = 20)";
        connection.connexion();
        try (Statement statement = connection.getNewConnection().createStatement()) {
            ResultSet resultSet = statement.executeQuery("string: query");
            while (resultSet.next()) {

                String nom = resultSet.getString("nom_etu");
                String prenom = resultSet.getString("prenom_etu");

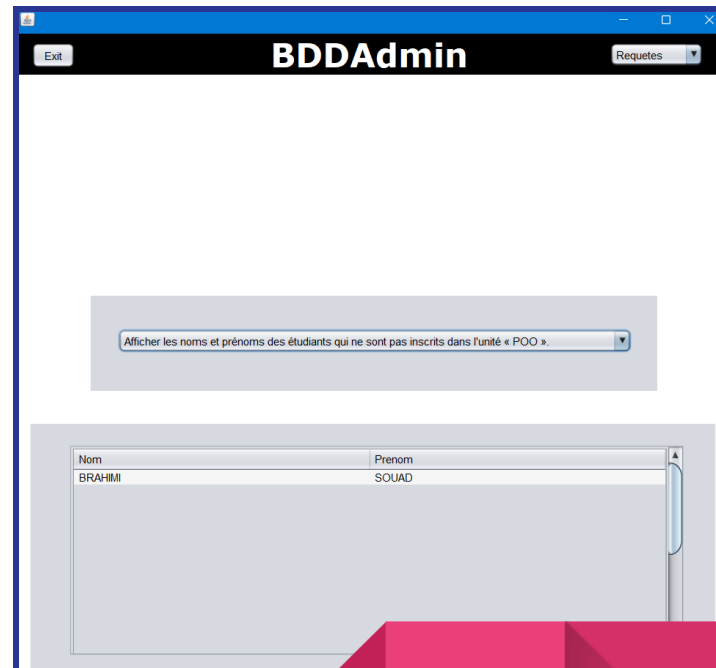
                Etudiant etudiant = new Etudiant(nom, prenom);
                etudiants.add(etudiant);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            connection.deconnexion(); // Close the connection outside the while loop
        }
        return etudiants;
    }

    //La deuxieme requete retourne une L.iste d'Etudiant
    public List<Etudiant> requete2() throws SQLException {
        List<Etudiant> etudiants = new ArrayList<>();
        String query = "SELECT nom_etu, prenom_etu FROM Etudiant MINUS SELECT nom_etu, prenom_etu FROM Etudiant where matricule_etu in (SELECT matricule_etu FROM EtudiantUnité where code_";
        connection.connexion();
        try (Statement statement = connection.getNewConnection().createStatement()) {
            ResultSet resultSet = statement.executeQuery("string: query");
            while (resultSet.next()) {

                String nom = resultSet.getString("nom_etu");
                String prenom = resultSet.getString("prenom_etu");

                Etudiant etudiant = new Etudiant(nom, prenom);
                etudiants.add(etudiant);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            connection.deconnexion(); // Close the connection outside the while loop
        }
        return etudiants;
    }
}
```

leur Interface:



- 2) **Etudiant:** l'étudiant peut seulement consulter dans la table Etudiant seulement.

The screenshot shows a window titled "Etudiant" with a blue header bar. In the top left corner of the window is an "Exit" button. Below the header, there is a light blue horizontal bar. The main content area has a grey background and contains a search bar with the label "Matricule" and a text input field. To the right of the input field is a "reduire" button. Below the search bar is a table with five columns: "Matricule", "Nom", "Prénom", "Date De Naissance", and "Adresse". The table is currently empty, showing only the header row. A vertical scrollbar is visible on the right side of the table.

This screenshot shows the same "Etudiant" application window, but now the table contains one data row. The search bar still has "20190001" entered, and the "reduire" button is present. The table data is as follows:

Matricule	Nom	Prénom	Date De Naissance	Adresse
20190001	BOUSSAI	MOHAMED	2000-01-12	Alger

Pour la consultation d'Étudiant se fait grâce à l'exécution de ce code:

```
private void matriculeEtudiantActionPerformed(java.awt.event.ActionEvent evt) {  
    Etudiant Etu_supp = new Etudiant (matricule: Integer.parseInt(s:matriculeEtudiant11.getText()));  
    try {  
        affiche3();  
    } catch (SQLException ex) {  
        Logger.getLogger(PourConnDeEtudiant.class.getName()).log(Level.SEVERE, msg: null, thrown: ex);  
    }  
}
```

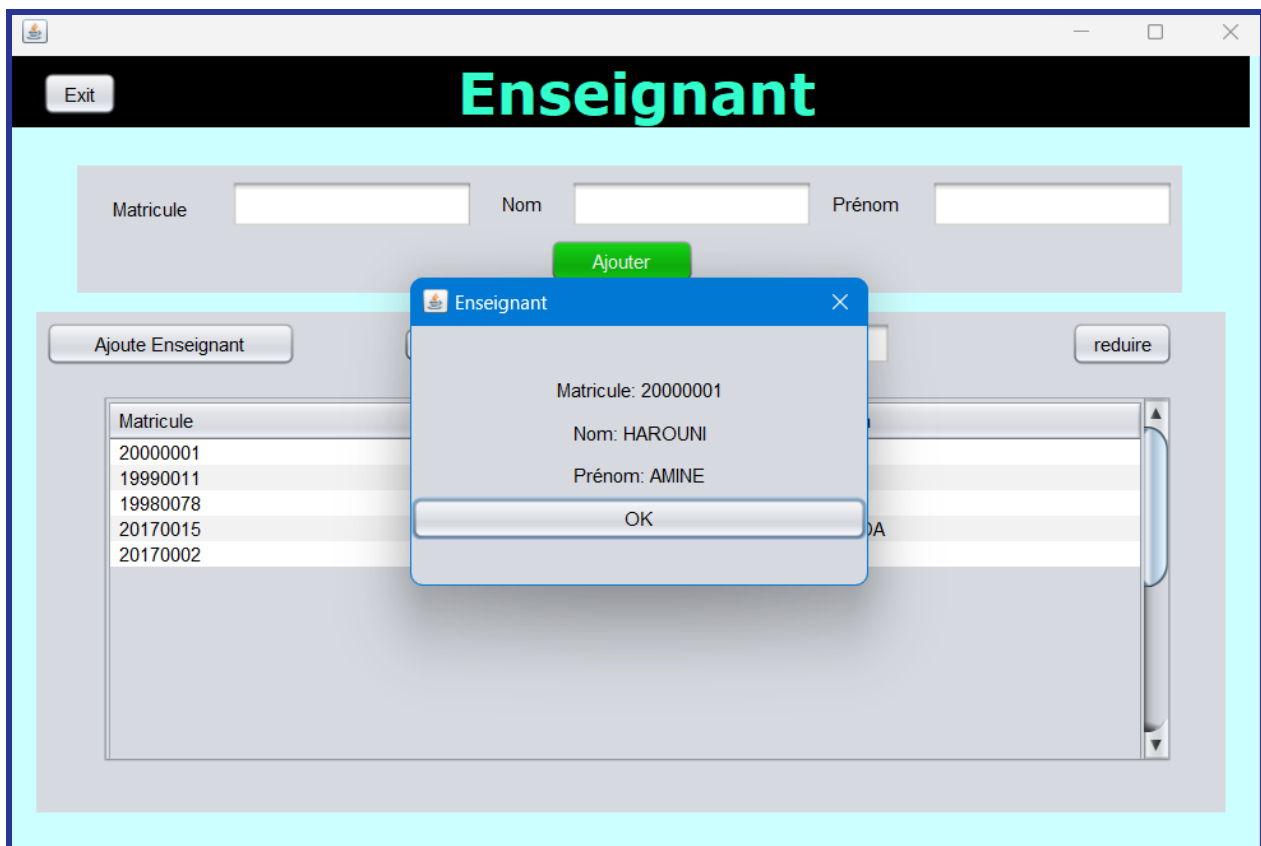
3) Enseignant:

Pour l'enseignant il pourra faire l'insertion et la consultation dans la table des Enseignants.

Matricule	Nom	Prénom
20000001	HAROUNI	AMINE
19990011	FATHI	OMAR
19980078	BOUZIDANE	FARAH
20170015	ARABI	ZOUBIDA
20170002	BENSAFIA	Djillali

1) La Consultation:

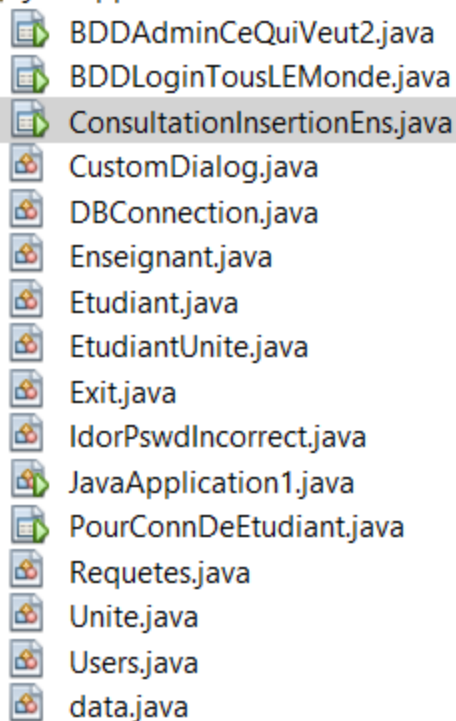
```
private void MatriculeEtudiant12ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Enseignant enseignant = new Enseignant(matricule: Integer.parseInt(s:MatriculeEnseignant13.getText()));  
    try {  
        Vector<String> enseignantRech= enseignant.recherche().toVector();  
        CustomDialog dialog = new CustomDialog( parent: this, titre: "Enseignant", text: "", tab: enseignantRech);  
        dialog.setVisible(b:true);  
    } catch (SQLException | NullPointerException ex) {  
        CustomDialog dialog = new CustomDialog( parent: this, titre: "Enseignant", text: "Enseignant n'existe pas", tab: null);  
        dialog.setVisible(b:true);  
    }  
}
```



2) L'Insertion:

```
private void ajouterEnseignant12ActionPerformed(java.awt.event.ActionEvent evt) {  
    Enseignant es1 = new Enseignant(matricule: Integer.parseInt(s:MatriculeEnseignant12.getText()),nom: nomEnseignant12.getText(),prenom: prenomEnseignant12.getText());  
    es1.insererEnseignant();  
    try {  
        affiche3();  
    } catch (SQLException ex1) {  
        Logger.getLogger(BDDAdminCeQuiVeut2.class.getName()).log(Level.SEVERE, msg: null, thrown: ex1);  
    }  
}
```

Dans ce Projet on a creer tous ces classe:



- BDDAdminCeQuiVeut2.java
- BDDLoginTousLEMonde.java
- ConsultationInsertionEns.java
- CustomDialog.java
- DBConnection.java
- Enseignant.java
- Etudiant.java
- EtudiantUnite.java
- Exit.java
- IdorPswdIncorrect.java
- JavaApplication1.java
- PourConnDeEtudiant.java
- Requetes.java
- Unite.java
- Users.java
- data.java

et on a fait la connexion à partir de la classe `DBConnection`.

```
package javaapplication1;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection{
    private Connection newConnection;
    private final String url = "jdbc:oracle:thin:@localhost:1521:XE";
    private String username ;
    private String password ;

    public DBConnection(String username, String password) {...4 lines }

    public Connection getNewConnection() {...3 lines }

    public void setNewConnection ( Connection newConnection) {...3 lines }

    public String getUsername() {...3 lines }

    public void setUsername(String username) {...3 lines }

    public String getPassword() {...3 lines }

    public void setPassword(String password) {...3 lines }

    public DBConnection() {...6 lines }

    public void connexion() {
        // Database connection parameters
        try {
            // Register the Oracle JDBC driver
            Class.forName(className:"oracle.jdbc.driver.OracleDriver");

            // Establish the connection
            newConnection = DriverManager.getConnection(url, user: username, password);
            // Connection successful, do your database operations here

        } catch (ClassNotFoundException | SQLException e) {
        }
    }

    public void deconnexion() throws SQLException{
        // Close the connection
        newConnection.close();
    }
}
```

3. Conclusion:

En conclusion, l'utilisation de l'interface Java Swing pour la gestion des étudiants et enseignants offre de nombreux avantages.

En utilisant cette interface, les établissements d'enseignement peuvent améliorer leur efficacité et leur précision dans la gestion des données des étudiants et enseignants. Les fonctionnalités telles que l'ajout et la modification des informations permettent de tenir à jour les données de manière rapide et précise. De plus, la fonction de recherche facilite la récupération des informations spécifiques nécessaires. L'interface Java Swing offre également la possibilité de personnaliser l'apparence de l'interface utilisateur, ce qui permet de créer une interface esthétiquement agréable et adaptée aux besoins de l'établissement. Les composants graphiques tels que les boutons, les champs de texte et les listes offrent une interaction conviviale avec les utilisateurs, rendant la gestion des données plus facile et plus intuitive.