

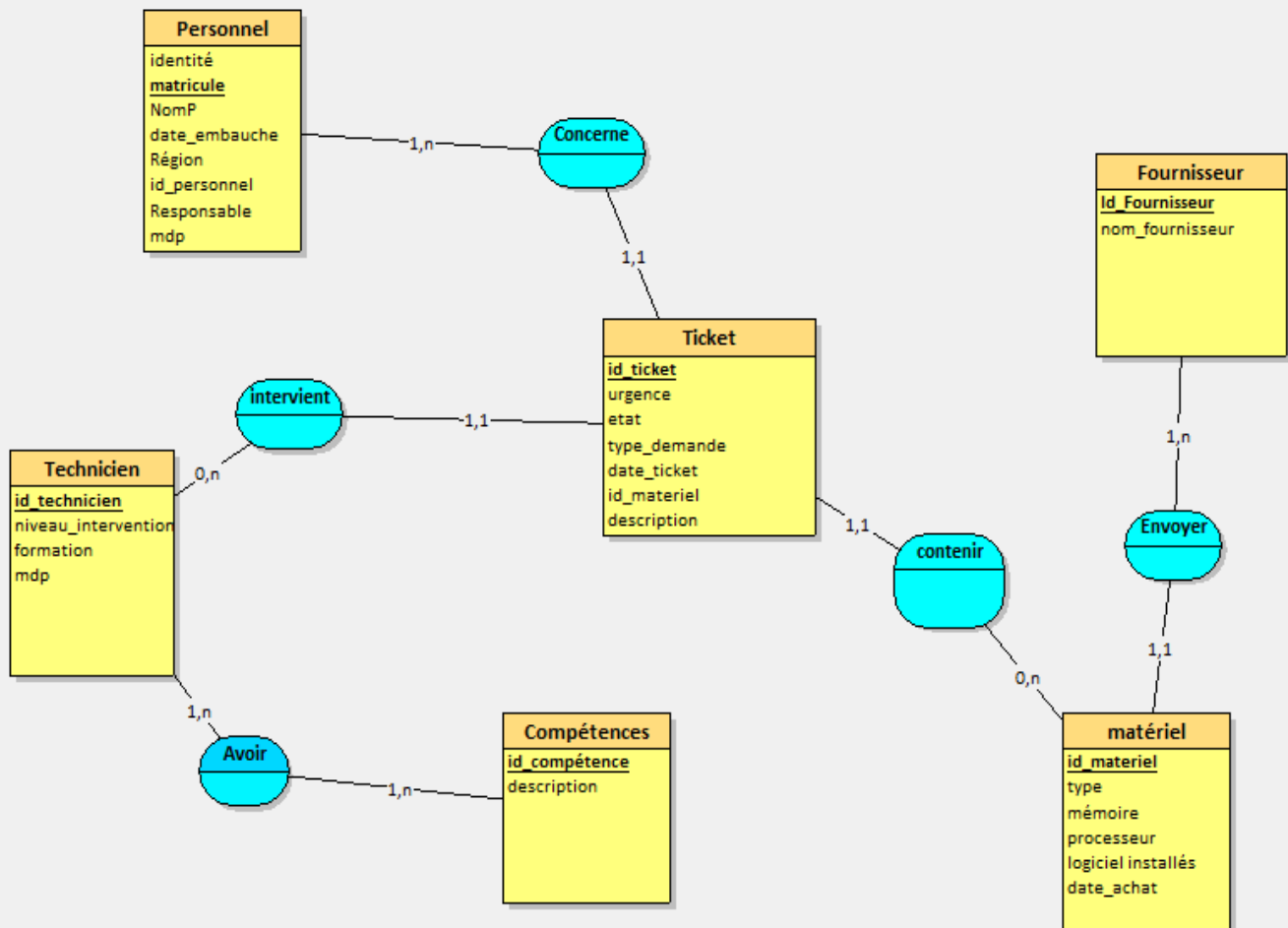
# PROJET



# **LISTE DES TÂCHES**

| Liste des tâches: |   |                        |                     |
|-------------------|---|------------------------|---------------------|
| <u>Séance</u>     | <u>Tâches</u>   | <u>Personne Chargé</u> | <u>Durée(Heure)</u> |
| Séance 1          | Structure BD  | Walid, Warren          | 1                   |
|                   | Gantt prévisionnel<br>(planification des tâches et ressources)          | Walid                  | 1                   |
|                   | Prototype   | Warren                 | 1                   |
| Séance 2          | Création BD (mise en place des tables et relations)                     | Warren                 | 2                   |
|                   | Réalisation des classes métiers (modélisation des entités BD)           | Walid, Warren          | 3                   |
| Séance 3          | Réalisation de la classe BD   | Walid, Warren          | 3                   |
|                   | Réalisation du programme principal (intégration des classes et logique) | Walid, Warren          | 2                   |
| Séance 4          | Débug du programme  | Walid, Warren          | 1                   |
| Séance 5          | Rédaction du rapport de projet  | Walid, Warren          | 2                   |
| Séance 5          | Génération du set-up (fichier exécutable ou installateur)               | Walid, Warren          | 2                   |
| Séance 5          | Mise à jour du Gantt réel (ajustement par rapport au planning initial)  | Walid, Warren          | 2                   |

# MCD



# MLD

Personnel = (matricule VARCHAR(50), identité VARCHAR(50), NomP VARCHAR(50), date\_embauche VARCHAR(50), Région VARCHAR(50), id\_personnel INT, Responsable LOGICAL, mdp VARCHAR(50));

Technicien = (id\_technicien INT, niveau\_intervention VARCHAR(50), formation VARCHAR(50), mdp VARCHAR(50));

Compétences = (id\_compétence VARCHAR(50), description VARCHAR(50));

Fournisseur = (Id\_Fournisseur COUNTER, nom\_fournisseur VARCHAR(50));

matériel = (id\_materiel VARCHAR(50), type VARCHAR(50), mémoire VARCHAR(50), processeur VARCHAR(50), logiciel\_installés VARCHAR(50), date\_achat VARCHAR(50), #Id\_Fournisseur);

Avoir = (#id\_technicien, #id\_compétence);  
Nous avons mis les Dates en format Date au début mais nous avons changé et mis en Varchar car plus c'est simple.

## GANTT PRÉVISIONNEL




| Nom   | Date de début | Date de fin |
|---|---------------|-------------|
| Structure BD  | 07/11/2024    | 12/11/2024  |
| Gantt prévisionnel (planification des tâches et ressourc... | 07/11/2024    | 07/11/2024  |
| Prototype   | 12/11/2024    | 15/11/2024  |
| Création BD (mise en place des tables et relations)         | 07/11/2024    | 11/11/2024  |
| Réalisation des classes métiers (modélisation des entit...  | 12/11/2024    | 12/11/2024  |
| Réalisation de la classe BD                                 | 12/11/2024    | 12/11/2024  |
| Réalisation du programme principal (intégration des cl...   | 12/11/2024    | 13/11/2024  |
| Débug du programme  | 14/11/2024    | 14/11/2024  |
| Rédaction du rapport de projet                              | 14/11/2024    | 14/11/2024  |
| Génération du set-up (fichier exécutable ou installateur)   | 14/11/2024    | 14/11/2024  |
| Mise à jour du Gantt réel (ajustement par rapport au pl...  | 14/11/2024    | 14/11/2024  |

## **GANNT RÉEL**

| Nom   | Date de début | Date de fin |
|---|---------------|-------------|
| Structure BD  | 07/11/2024    | 07/11/2024  |
| Gantt prévisionnel (planification des tâches et ressourc... | 07/11/2024    | 07/11/2024  |
| Prototype   | 12/11/2024    | 15/11/2024  |
| Création BD (mise en place des tables et relations)         | 07/11/2024    | 13/11/2024  |
| Réalisation des classes métiers (modélisation des entit...  | 12/11/2024    | 14/11/2024  |
| Réalisation de la classe BD                                 | 12/11/2024    | 14/11/2024  |
| Réalisation du programme principal (intégration des cl...   | 12/11/2024    | 15/11/2024  |
| Débug du programme  | 14/11/2024    | 15/11/2024  |
| Rédaction du rapport de projet                              | 14/11/2024    | 15/11/2024  |
| Génération du set-up (fichier exécutable ou installateur)   | 14/11/2024    | 15/11/2024  |
| Mise à jour du Gantt réel (ajustement par rapport au pl...  | 14/11/2024    | 15/11/2024  |

# Générer Setup

▼ Today

|   |                     |                      |
|---|---------------------|----------------------|
|  Laboratoire       | 11/19/2024 11:05 AM | Application Manifest |
|  setup             | 11/19/2024 11:05 AM | Application          |
|  Application Files | 11/19/2024 11:05 AM | File folder          |

Grâce à la génération du setup, nous avons déployé notre application

## *Class BD*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Configuration;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Laboratoire
{
    11 references
    public class BD
    {
        /// <summary>
        /// Classe BD pour gérer les interactions avec la base de données 'laboratoire'.
        /// </summary>
        private static string connStr = "Server=127.0.0.1; Database=laboratoire; Uid=root; Password=; ";
        MySqlConnection conn;
        /// <summary>
        /// Ouvre une connexion à la base de données et la retourne.
        /// </summary>
        /// <returns>Une connexion ouverte à la base de données.</returns>
        21 references
        public static MySqlConnection Connexion()...
        /// <summary>
        /// Ferme la connexion à la base de données.
        /// </summary>
        0 references
        public static void Deconnexion()...
        /// <summary>
        /// Ajoute un technicien à la base de données.
        /// </summary>
        /// <param name="unTechnicien">Objet Technicien à ajouter.</param>
        1 reference
        public static void AddTechnicien(Technicien unTechnicien)
        {
            using (MySqlConnection connection = Connexion())...
            Connexion().Close();
        }
    }
}
```

```
/// <summary>
/// Modifie les informations d'un technicien.
/// </summary>
/// <param name="id">ID du technicien à modifier.</param>
/// <param name="formationT">Nouvelle formation du technicien.</param>
```

1 reference

```
public static void ModifTechnicien(int id, string formationT)...
```

```
/// <summary>
/// Supprime un technicien de la base de données.
/// </summary>
/// <param name="id">ID du technicien à supprimer.</param>
```

1 reference

```
public static void SuppTechnicien(int id)...
```

```
/// <summary>
/// Sélectionne tous les techniciens dans la base de données.
/// </summary>
/// <returns>Liste de techniciens.</returns>
```

0 references

```
public static List<Technicien> SelectTechnicien()...
```

```
/// <summary>
/// Ajoute un ticket à la base de données.
/// </summary>
/// <param name="unTicket">Objet Ticket à ajouter.</param>
```

0 references

```
public static void AddTicket(Ticket unTicket)...
```

```
/// <summary>
/// Sélectionne tous les tickets dans la base de données.
/// </summary>
/// <returns>Liste de tickets.</returns>
```

1 reference

```
public static List<Ticket> SelectTicket()...
```

```
/// <summary>
/// Ajoute un matériel à la base de données.
```

```
/// </summary>
/// <param name="unMateriel">Objet Materiel à ajouter.</param>
```

1 reference

```
public static void AddMateriel(Materiel unMateriel)...
```

```
/// <summary>
/// Sélectionne tous les matériels dans la base de données.
/// </summary>
/// <returns>Liste de matériels.</returns>
```

1 reference

```
public static List<Materiel> SelectMateriel()...
```

```
/// <summary>
/// Supprime un matériel de la base de données en fonction de son ID.
/// </summary>
/// <param name="id">ID du matériel à supprimer.</param>
```

1 reference

```
public static void SuppMateriel(int id)...
```

```
/// <summary>
/// Ajoute une compétence à la base de données.
/// </summary>
/// <param name="uneCompetence">Objet Competence à ajouter.</param>
```

0 references

```
public static void AddCompetence(Competence uneCompetence)
{
    using (SqlConnection connection = Connexion())...
```

```
/// <summary>
/// Sélectionne toutes les compétences dans la base de données.
/// </summary>
/// <returns>Liste de compétences.</returns>
```

0 references

```
public static List<Competence> SelectCompetence()...
```

```
/// <summary>
/// Ajoute un fournisseur à la base de données.
/// </summary>
/// <param name="unFournisseur">Objet Fournisseur à ajouter.</param>
```

0 references

```
public static void AddFournisseur(Fournisseur unFournisseur)...
```

```

/// <summary>
/// Sélectionne tous les fournisseurs dans la base de données.
/// </summary>
/// <returns>Liste de fournisseurs.</returns>
0 references
public static List<Fournisseur> SelectFournisseur()...

/// <summary>
/// Ajoute un personnel à la base de données.
/// </summary>
/// <param name="unPersonnel">Objet Personnel à ajouter.</param>
1 reference
public static void AddPersonnel(Personnel unPersonnel)...

/// <summary>
/// Modifie les informations d'un personnel.
/// </summary>
/// <param name="id">ID du personnel à modifier.</param>
/// <param name="identiteP">Nouvelle identité du personnel.</param>
1 reference
public static void ModifPersonnel(int id, string identiteP)...

/// <summary>
/// Supprime un personnel de la base de données en fonction de son ID.
/// </summary>
/// <param name="id">ID du personnel à supprimer.</param>
1 reference
public static void SuppPersonnel(int id)...

/// <summary>
/// Sélectionne tous les personnels dans la base de données.
/// </summary>
/// <returns>Liste de personnels.</returns>
0 references
public static List<Personnel> SelectPersonnel()...

/// <summary>
/// Modifie l'état et la description d'un ticket.
/// </summary>
/// <param name="idT">ID du ticket à modifier.</param>
/// <param name="etat">Nouvel état du ticket.</param>
/// <param name="Description">Nouvelle description du ticket.</param>
1 reference
public static void ModifTicket(int idT, string etat, string Description)
{
    using (MySQLConnection connection = Connexion())...
}

```

namespace Laboratoire

```

{
    public class BD
    {
        /// <summary>

```



```

    /// Classe BD pour gérer les interactions avec la base de données `laboratoire`.
    /// </summary>
    private static string connStr = "Server=127.0.0.1; Database=laboratoire; Uid=root;
Password=";
    MySqlConnection conn;
    /// <summary>
    /// Ouvre une connexion à la base de données et la retourne.
    /// </summary>
    /// <returns>Une connexion ouverte à la base de données.</returns>
    public static MySqlConnection Connexion()
    {
        MySqlConnection connection = new MySqlConnection(connStr);

        try
        {
            connection.Open();
            Console.WriteLine("Connexion à la base de données réussie.");
            return connection;
        }
        catch (MySqlException ex)
        {
            Console.WriteLine($"Erreur de connexion : {ex.Message}");
            throw new Exception("Impossible de se connecter à la base de données.", ex);
        }
    }
    /// <summary>
    /// Ferme la connexion à la base de données.
    /// </summary>
    public static void Deconnexion()
    {
        Connexion().Close();
    }
    /// <summary>
    /// Ajoute un technicien à la base de données.
    /// </summary>
    /// <param name="unTechnicien">Objet Technicien à ajouter.</param>

    public static void AddTechnicien(Technicien unTechnicien)
    {
        using (MySqlConnection connection = Connexion())
        {

```

```

        string requete = "INSERT INTO technicien (niveau_intervention, formation , mdp)
VALUES (@niveau_technicien, @formationT , @mdp)";

```

```

        using (MySqlCommand commande = new MySqlCommand(requete, connection))
        {

            commande.Parameters.AddWithValue("@niveau_technicien",
unTechnicien.GetNiveauIntervention());
            commande.Parameters.AddWithValue("@formationT",
unTechnicien.GetFormation());
            commande.Parameters.AddWithValue("@mdp", unTechnicien.GetMdp());

            commande.ExecuteNonQuery();

```

```

        }
    }
    Connexion().Close();
}

/// <summary>
/// Modifie les informations d'un technicien.
/// </summary>
/// <param name="id">ID du technicien à modifier.</param>
/// <param name="formationT">Nouvelle formation du technicien.</param>
public static void ModifTechnicien(int id, string formationT)
{
    using (MySqlConnection connection = Connexion())
    {
        string requete = "UPDATE technicien SET formation = @formation WHERE
id_technicien = @id ";
        using (MySqlCommand commande = new MySqlCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id", id);
            commande.Parameters.AddWithValue("@formation", formationT);
            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>
/// Supprime un technicien de la base de données.
/// </summary>
/// <param name="id">ID du technicien à supprimer.</param>
public static void SuppTechnicien(int id)
{

```

```

        using (MySqlConnection connection = Connexion())
        {
            string requete = "DELETE FROM technicien WHERE id_technicien =
@id_technicien";
            using (MySqlCommand commande = new MySqlCommand(requete, connection))
            {
                commande.Parameters.AddWithValue("@id_technicien", id);

                commande.ExecuteNonQuery();
            }
        }
        Connexion().Close();
    }
    /// <summary>
    /// Sélectionne tous les techniciens dans la base de données.
    /// </summary>
    /// <returns>Liste de techniciens.</returns>

```

```

public static List<Technicien> SelectTechnicien()
{
    List<Technicien> lesTechniciens = new List<Technicien>();

    using (MySqlConnection connection = Connexion())
    {

        string requete = "SELECT * FROM technicien";

        using (MySqlCommand commande = new MySqlCommand(requete, connection))
        {

            using (MySqlDataReader reader = commande.ExecuteReader())
            {

                while (reader.Read())
                {

                    Technicien unTechnicien = new Technicien(
                        reader["niveau_intervention"].ToString(),
                        reader["formation"].ToString()
                    );

```

```

        lesTechniciens.Add(unTechnicien);
    }
}
}

return lesTechniciens;
}
/// <summary>
/// Ajoute un ticket à la base de données.
/// </summary>
/// <param name="unTicket">Objet Ticket à ajouter.</param>

```

```

public static void AddTicket(Ticket unTicket)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "INSERT INTO ticket
(urgence,etat,type_demande,date_ticket,id_technicien,id_materiel,matricule,description)
VALUES
(@urgence,@etat,@type_demande,@date_ticket,@id_technicien,@id_materiel,@matricule,
@description)";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@urgence", unTicket.GetUrgence());
            commande.Parameters.AddWithValue("@etat", unTicket.GetEtat());
            commande.Parameters.AddWithValue("@type_demande", unTicket.GetType());
            commande.Parameters.AddWithValue("@date_ticket",
unTicket.GetDateTicket());
            commande.Parameters.AddWithValue("@id_technicien",
unTicket.GetIdTechnicien());
            commande.Parameters.AddWithValue("@id_materiel",
unTicket.GetIdMateriel());
            commande.Parameters.AddWithValue("@matricule", unTicket.GetMatricule());
            commande.Parameters.AddWithValue("@descripton",
unTicket.GetDescription());
            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>

```

```

/// Sélectionne tous les tickets dans la base de données.
/// </summary>
/// <returns>Liste de tickets.</returns>

```

```

public static List<Ticket> SelectTicket()
{
    List<Ticket> lesTickets = new List<Ticket>();

    using (MySQLConnection connection = Connexion())
    {
        string requete = "SELECT * FROM ticket";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            using (MySQLDataReader reader = commande.ExecuteReader())
            {
                while (reader.Read())
                {
                    Ticket unTicket = new Ticket(Convert.ToInt32((reader["id_ticket"])),
reader["type_demande"].ToString(), reader["etat"].ToString(), reader["urgence"].ToString(),
reader["date_ticket"].ToString(), Convert.ToInt32((reader["id_technicien"])),
reader["id_materiel"].ToString(), reader["matricule"].ToString(),
reader["description"].ToString());
                    lesTickets.Add(unTicket);
                }
            }
        }
    }

    return lesTickets;
}
/// <summary>
/// Ajoute un matériel à la base de données.
/// </summary>
/// <param name="unMateriel">Objet Materiel à ajouter.</param>

```

```

public static void AddMateriel(Materiel unMateriel)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "INSERT INTO materiel
(type,memoire,processeur,logiciels_installes,date_achat,id_fournisseur) VALUES
(@type,@memoire,@processeur,@logiciels_installes,@date_achat,@id_fournisseur)";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@type", unMateriel.getleType());
            commande.Parameters.AddWithValue("@memoire", unMateriel.getMemoire());

```

```

        commande.Parameters.AddWithValue("@processeur",
unMateriel.getProcesseur());
        commande.Parameters.AddWithValue("@logiciels_installes",
unMateriel.getLogiciel());
        commande.Parameters.AddWithValue("@date_achat",
unMateriel.getDate_achat());
        commande.Parameters.AddWithValue("@id_fournisseur",
unMateriel.getId_Fournisseur());
        commande.ExecuteNonQuery();
    }
}
}
/// <summary>
/// Sélectionne tous les matériels dans la base de données.
/// </summary>
/// <returns>Liste de matériels.</returns>
public static List<Materiel> SelectMateriel()
{
    List<Materiel> lesMateriels = new List<Materiel>();

    using (MySqlConnection connection = Connexion())
    {
        string requete = "SELECT * FROM materiel";
        using (MySqlCommand commande = new MySqlCommand(requete, connection))
        {
            using (MySqlDataReader reader = commande.ExecuteReader())
            {
                while (reader.Read())
                {
                    Materiel unMateriel = new
Materiel(Convert.ToInt32(reader["id_materiel"]), reader["type"].ToString(),
reader["memoire"].ToString(), reader["processeur"].ToString(),
reader["logiciels_installes"].ToString(), reader["date_achat"].ToString(),
Convert.ToInt32(reader["id_fournisseur"]));

                    lesMateriels.Add(unMateriel);
                }
            }
        }
    }

    return lesMateriels;
}
/// <summary>
/// Supprime un matériel de la base de données en fonction de son ID.
/// </summary>
/// <param name="id">ID du matériel à supprimer.</param>
public static void SuppMateriel(int id)

```

```

{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "DELETE FROM materiel WHERE id_materiel = @id_materiel";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id_materiel", id);

            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>
/// Ajoute une compétence à la base de données.
/// </summary>
/// <param name="uneCompetence">Objet Competence à ajouter.</param>

public static void AddCompetence(Competence uneCompetence)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "INSERT INTO competence (id_competence,description)
VALUES (@id_competence,@description)";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id_competence",
uneCompetence.getId_competence());
            commande.Parameters.AddWithValue("@description",
uneCompetence.getDescription());
            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>
/// Sélectionne toutes les compétences dans la base de données.
/// </summary>
/// <returns>Liste de compétences.</returns>

public static List<Competence> SelectCompetence()
{
    List<Competence> lesCompetences = new List<Competence>();

```

```

using (MySqlConnection connection = Connexion())
{
    string requete = "SELECT description FROM competence";
    using (MySqlCommand commande = new MySqlCommand(requete, connection))
    {
        using (MySqlDataReader reader = commande.ExecuteReader())
        {
            while (reader.Read())
            {
                Competence uneCompetence = new
Competence(reader["description"].ToString());
                lesCompetences.Add(uneCompetence);
            }
        }
    }
}

return lesCompetences;
}

```

```

/// <summary>
/// Ajoute un fournisseur à la base de données.
/// </summary>
/// <param name="unFournisseur">Objet Fournisseur à ajouter.</param>
public static void AddFournisseur(Fournisseur unFournisseur)
{
    using (MySqlConnection connection = Connexion())
    {
        string requete = "INSERT INTO fournisseur (id_fournisseur,nom_fournisseur)
VALUES (@id_fournisseur,@nom_fournisseur)";
        using (MySqlCommand commande = new MySqlCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id_fournisseur",
unFournisseur.getId_fournisseur());
            commande.Parameters.AddWithValue("@nom_fournisseur",
unFournisseur.getNomFournisseur());
            commande.ExecuteNonQuery();
        }
    }
}

```

```

/// <summary>
/// Sélectionne tous les fournisseurs dans la base de données.
/// </summary>
/// <returns>Liste de fournisseurs.</returns>
public static List<Fournisseur> SelectFournisseur()
{

```



```

List<Fournisseur> lesFournisseurs = new List<Fournisseur>();

using (MySQLConnection connection = Connexion())
{
    string requete = "SELECT nom_fournisseur FROM fournisseur";
    using (MySQLCommand commande = new MySQLCommand(requete, connection))
    {
        using (MySQLDataReader reader = commande.ExecuteReader())
        {
            while (reader.Read())
            {
                Fournisseur unFournisseur = new
Fournisseur(reader["nom_fournisseur"].ToString());
                lesFournisseurs.Add(unFournisseur);
            }
        }
    }
}

return lesFournisseurs;
}

```

```

/// <summary>
/// Ajoute un personnel à la base de données.
/// </summary>
/// <param name="unPersonnel">Objet Personnel à ajouter.</param>
public static void AddPersonnel(Personnel unPersonnel)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "INSERT INTO personnel
(identite,matricule,nomP,date_embauche,region, id_personnel,responsable) VALUES
(@identite,@matricule,@nomP,@date_embauche,@region,@id_personnel,@responsable)";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@identite", unPersonnel.getIdentiteP());
            commande.Parameters.AddWithValue("@matricule",
unPersonnel.getMatricule());
            commande.Parameters.AddWithValue("@nomP", unPersonnel.getNomP());
            commande.Parameters.AddWithValue("@date_embauche",
unPersonnel.getDte_embauche());
            commande.Parameters.AddWithValue("@region", unPersonnel.getRegion());
            commande.Parameters.AddWithValue("@id_personnel",
unPersonnel.getId_personnel());
            commande.Parameters.AddWithValue("@responsable",
unPersonnel.getResponsable());
            commande.ExecuteNonQuery();
        }
    }
}

```

```

    }
}

/// <summary>
/// Modifie les informations d'un personnel.
/// </summary>
/// <param name="id">ID du personnel à modifier.</param>
/// <param name="identiteP">Nouvelle identité du personnel.</param>

public static void ModifPersonnel(int id, string identiteP)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "UPDATE personnel SET identite= @identite WHERE
id_personnel = @id_personnel";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id_personnel", id);
            commande.Parameters.AddWithValue("@identite", identiteP);

            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>
/// Supprime un personnel de la base de données en fonction de son ID.
/// </summary>
/// <param name="id">ID du personnel à supprimer.</param>

public static void SuppPersonnel(int id)
{
    using (MySQLConnection connection = Connexion())
    {
        string requete = "DELETE FROM personnel WHERE id_personnel =
@id_personnel";
        using (MySQLCommand commande = new MySQLCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@id_personnel", id);

            commande.ExecuteNonQuery();
        }
    }
}

/// <summary>
/// Sélectionne tous les personnels dans la base de données.
/// </summary>

```

```

/// <returns>Liste de personnels.</returns>

public static List<Personnel> SelectPersonnel()
{
    List<Personnel> lesPersonnels = new List<Personnel>();

    using (SqlConnection connection = Connexion())
    {
        string requete = "SELECT nomP FROM personnel";
        using (SqlCommand commande = new SqlCommand(requete, connection))
        {
            using (MySqlDataReader reader = commande.ExecuteReader())
            {
                while (reader.Read())
                {
                    Personnel unPersonnel = new Personnel(reader["identiteP"].ToString(),
reader["matricule"].ToString(), reader["nomP"].ToString(),
Convert.ToDateTime(reader["date_embauche"]), reader["region"].ToString(),
Convert.ToBoolean(reader["responsable"]), reader["mdp"].ToString());
                    lesPersonnels.Add(unPersonnel);
                }
            }
        }
    }

    return lesPersonnels;
}

/// <summary>
/// Modifie l'état et la description d'un ticket.
/// </summary>
/// <param name="idT">ID du ticket à modifier.</param>
/// <param name="etat">Nouvel état du ticket.</param>
/// <param name="Description">Nouvelle description du ticket.</param>
public static void ModifTicket(int idT, string etat, string Description)
{
    using (SqlConnection connection = Connexion())
    {
        string requete = "UPDATE ticket SET etat = @etat, Description = @Description
WHERE id_ticket = @id_ticket ";
        using (SqlCommand commande = new SqlCommand(requete, connection))
        {
            commande.Parameters.AddWithValue("@etat", etat);
            commande.Parameters.AddWithValue("@Description", Description);
            commande.Parameters.AddWithValue("@id_ticket", idT);
            commande.ExecuteNonQuery();
        }
    }
}

```

```
}  
}
```

# Class Matériel

```
namespace Laboratoire  
{  
    /// <summary>  
    /// Représente un matériel dans le système.  
    /// </summary>  
    11 references  
    public class Matériel  
    {  
        // Variables d'instance privées pour stocker les attributs du matériel  
        private int id_materiel;           // Identifiant unique du matériel  
        private string letype;             // Type de matériel  
        private string memoire;            // Mémoire du matériel  
        private string processeur;         // Processeur du matériel  
        private string logiciel;           // Logiciel installé sur le matériel  
        private string date_achat;         // Date d'achat du matériel  
        private int id_fournisseur;        // Identifiant du fournisseur du matériel  
  
        /// <summary>  
        /// Constructeur de la classe Matériel.  
        /// Initialise un nouveau matériel avec les valeurs fournies.  
        /// </summary>  
        /// <param name="id_materiel">L'identifiant du matériel.</param>  
        /// <param name="letype">Le type du matériel.</param>  
        /// <param name="memoire">La mémoire du matériel.</param>  
        /// <param name="processeur">Le processeur du matériel.</param>  
        /// <param name="logiciel">Le logiciel installé sur le matériel.</param>  
        /// <param name="date_achat">La date d'achat du matériel.</param>  
        /// <param name="id_fournisseur">L'identifiant du fournisseur.</param>  
        1 reference  
        public Matériel(int id_materiel, string letype, string memoire, string processeur, string logiciel, string date_achat, int id_fournisseur) {...}  
  
        /// <summary>  
        /// Constructeur de la classe Matériel.  
        /// Initialise un matériel avec les valeurs nécessaires sauf l'identifiant.  
        /// </summary>  
        /// <param name="letype">Le type du matériel.</param>  
        /// <param name="memoire">La mémoire du matériel.</param>  
        /// <param name="processeur">Le processeur du matériel.</param>  
        /// <param name="logiciel">Le logiciel installé sur le matériel.</param>  
        /// <param name="date_achat">La date d'achat du matériel.</param>  
        /// <param name="id_fournisseur">L'identifiant du fournisseur.</param>  
        1 reference  
        public Matériel(string letype, string memoire, string processeur, string logiciel, string date_achat, int id_fourni {...}  
  
        /// <summary>  
        /// Obtient l'identifiant du matériel.  
        /// </summary>  
        /// <returns>L'identifiant du matériel.</returns>  
        1 reference  
        public int getId_materiel() {...}  
  
        /// <summary>  
        /// Obtient le type du matériel.  
        /// </summary>  
        /// <returns>Le type du matériel.</returns>  
        2 references  
        public string getletype() {...}  
  
        /// <summary>  
        /// Obtient la mémoire du matériel.  
        /// </summary>  
        /// <returns>La mémoire du matériel.</returns>  
        1 reference  
        public string getMemoire() {...}  
  
        /// <summary>  
        /// Obtient le processeur du matériel.  
        /// </summary>  
        /// <returns>Le processeur du matériel.</returns>  
        1 reference  
        public string getProcesseur() {...}  
  
        /// <summary>  
        /// Obtient le logiciel installé sur le matériel.  
        /// </summary>  
        /// <returns>Le logiciel installé sur le matériel.</returns>  
        1 reference  
        public string getLogiciel() {...}
```

```

/// <summary>
/// Obtient le logiciel installé sur le matériel.
/// </summary>
/// <returns>Le logiciel installé sur le matériel.</returns>
1 reference
public string getLogiciel()...

/// <summary>
/// Obtient la date d'achat du matériel.
/// </summary>
/// <returns>La date d'achat du matériel.</returns>
1 reference
public string getDate_achat()...

/// <summary>
/// Obtient l'identifiant du fournisseur du matériel.
/// </summary>
/// <returns>L'identifiant du fournisseur.</returns>
1 reference
public int getId_Fournisseur()...

/// <summary>
/// Modifie le type du matériel en fonction de certaines conditions.
/// </summary>
/// <param name="letype">Le nouveau type du matériel.</param>
0 references
public void setType(string letype)...

/// <summary>
/// Modifie le logiciel installé sur le matériel.
/// </summary>
/// <param name="lelogiciel">Le nouveau logiciel à installer sur le matériel.</param>
0 references
public void setLogiciel(string lelogiciel)...
```

public class Materiel

```

{
    // Variables d'instance privées pour stocker les attributs du matériel
    private int id_materiel;    // Identifiant unique du matériel
    private string letype;     // Type de matériel
    private string memoire;    // Mémoire du matériel
    private string processeur; // Processeur du matériel
    private string logiciel;   // Logiciel installé sur le matériel
    private string date_achat; // Date d'achat du matériel
    private int id_fournisseur; // Identifiant du fournisseur du matériel

    /// <summary>
    /// Constructeur de la classe Materiel.
    /// Initialise un nouveau matériel avec les valeurs fournies.
    /// </summary>
    /// <param name="id_materiel">L'identifiant du matériel.</param>
    /// <param name="letype">Le type du matériel.</param>
    /// <param name="memoire">La mémoire du matériel.</param>
    /// <param name="processeur">Le processeur du matériel.</param>
    /// <param name="logiciel">Le logiciel installé sur le matériel.</param>
    /// <param name="date_achat">La date d'achat du matériel.</param>
```

```

    /// <param name="id_fournisseur">L'identifiant du fournisseur.</param>
    public Materiel(int id_materiel, string letype, string memoire, string processeur, string
logiciel, string date_achat, int id_fournisseur)
    {
        this.id_materiel = id_materiel;        // Assigner l'ID du matériel
        this.letype = letype;                  // Assigner le type du matériel
        this.memoire = memoire;                // Assigner la mémoire
        this.processeur = processeur;          // Assigner le processeur
        this.logiciel = logiciel;              // Assigner le logiciel
        this.date_achat = date_achat;          // Assigner la date d'achat
        this.id_fournisseur = id_fournisseur;  // Assigner l'ID du fournisseur
    }

    /// <summary>
    /// Constructeur de la classe Materiel.
    /// Initialise un matériel avec les valeurs nécessaires sauf l'identifiant.
    /// </summary>
    /// <param name="letype">Le type du matériel.</param>
    /// <param name="memoire">La mémoire du matériel.</param>
    /// <param name="processeur">Le processeur du matériel.</param>
    /// <param name="logiciel">Le logiciel installé sur le matériel.</param>
    /// <param name="date_achat">La date d'achat du matériel.</param>
    /// <param name="id_fournisseur">L'identifiant du fournisseur.</param>
    public Materiel(string letype, string memoire, string processeur, string logiciel, string
date_achat, int id_fournisseur)
    {
        this.letype = letype;                  // Assigner le type du matériel
        this.memoire = memoire;                // Assigner la mémoire
        this.processeur = processeur;          // Assigner le processeur
        this.logiciel = logiciel;              // Assigner le logiciel
        this.date_achat = date_achat;          // Assigner la date d'achat
        this.id_fournisseur = id_fournisseur;  // Assigner l'ID du fournisseur
    }

    /// <summary>
    /// Obtient l'identifiant du matériel.
    /// </summary>
    /// <returns>L'identifiant du matériel.</returns>
    public int getId_materiel()
    {
        return id_materiel;
    }

    /// <summary>
    /// Obtient le type du matériel.
    /// </summary>
    /// <returns>Le type du matériel.</returns>
    public string getletype()

```

```

{
    return letype;
}

/// <summary>
/// Obtient la mémoire du matériel.
/// </summary>
/// <returns>La mémoire du matériel.</returns>
public string getMemoire()
{
    return memoire;
}

/// <summary>
/// Obtient le processeur du matériel.
/// </summary>
/// <returns>Le processeur du matériel.</returns>
public string getProcesseur()
{
    return processeur;
}

/// <summary>
/// Obtient le logiciel installé sur le matériel.
/// </summary>
/// <returns>Le logiciel installé sur le matériel.</returns>
public string getLogiciel()
{
    return logiciel;
}

/// <summary>
/// Obtient la date d'achat du matériel.
/// </summary>
/// <returns>La date d'achat du matériel.</returns>
public string getDate_achat()
{
    return date_achat;
}

/// <summary>
/// Obtient l'identifiant du fournisseur du matériel.
/// </summary>
/// <returns>L'identifiant du fournisseur.</returns>
public int getId_Fournisseur()
{
    return id_fournisseur;
}

```

```

    /// <summary>
    /// Modifie le type du matériel en fonction de certaines conditions.
    /// </summary>
    /// <param name="letype">Le nouveau type du matériel.</param>
    public void setType(string letype)
    {
        // Vérifie si le type fourni correspond à l'une des valeurs spécifiques et le modifie en
conséquence
        if (letype == "Prise en charge par telephone")
        {
            this.letype = letype;
        }
        if (letype == "Prise en charge en télémaintenance")
        {
            this.letype = letype;
        }
        if (letype == "Prise en charge en télémaintenance")
        {
            this.letype = letype;
        }
    }

    /// <summary>
    /// Modifie le logiciel installé sur le matériel.
    /// </summary>
    /// <param name="lelogiciel">Le nouveau logiciel à installer sur le matériel.</param>
    public void setLogiciel(string lelogiciel)
    {
        lelogiciel = logiciel; // Note: Cette ligne semble incorrecte, elle affecte la variable
locale `lelogiciel` et non l'attribut `logiciel`
    }
}
}

```



# Class Technicien

```
11 references
public class Technicien
{
    private string id_technicien;
    private string niveau_intervention;
    private string formationT;
    private string mdp;

    /// <summary>
    /// Constructeur pour initialiser un technicien avec un identifiant, un niveau d'intervention, une formation et un mot de pas
    /// </summary>
    /// <param name="id_technicien">L'identifiant unique du technicien.</param>
    /// <param name="niveau_intervention">Le niveau d'intervention du technicien.</param>
    /// <param name="formationT">La formation du technicien.</param>
    /// <param name="mdp">Le mot de passe du technicien.</param>
    0 references
    public Technicien(string id_technicien, string niveau_intervention, string formationT, string mdp)
    {
        this.id_technicien = id_technicien;
        this.niveau_intervention = niveau_intervention;
        this.formationT = formationT;
        this.mdp = mdp;
    }

    /// <summary>
    /// Constructeur pour initialiser un technicien avec un niveau d'intervention, une formation et un mot de passe.
    /// </summary>
    /// <param name="niveau_intervention">Le niveau d'intervention du technicien.</param>
    /// <param name="formationT">La formation du technicien.</param>
    /// <param name="mdp">Le mot de passe du technicien.</param>
    1 reference
    public Technicien(string niveau_intervention, string formationT, string mdp)
    {
        this.niveau_intervention = niveau_intervention;
        this.formationT = formationT;
        this.mdp = mdp;
    }

    /// <summary>
    /// Constructeur pour initialiser un technicien avec une formation et un mot de passe.
}
```

```

/// </summary>
/// <param name="formationT">La formation du technicien.</param>
/// <param name="mdp">Le mot de passe du technicien.</param>
1 reference
public Technicien(string formationT, string mdp)
{
    this.formationT = formationT;
    this.mdp = mdp;
}

/// <summary>
/// Obtient l'identifiant du technicien.
/// </summary>
/// <returns>L'identifiant du technicien.</returns>
0 references
public string GetIdTechnicien()
{
    return id_technicien;
}

/// <summary>
/// Modifie l'identifiant du technicien.
/// </summary>
/// <param name="unid_technicien">Le nouvel identifiant du technicien.</param>
0 references
public void SetIdTechnicien(string unid_technicien)
{
    id_technicien = unid_technicien;
}

/// <summary>
/// Obtient le niveau d'intervention du technicien.
/// </summary>
/// <returns>Le niveau d'intervention du technicien.</returns>
1 reference
public string GetNiveauIntervention()
{
    return niveau_intervention;
}

```

```

    /// <summary>
    /// Modifie le niveau d'intervention du technicien.
    /// </summary>
    /// <param name="unNiveauIntervention">Le nouveau niveau d'intervention.</param>
    0 references
    public void SetNiveauIntervention(string unNiveauIntervention)
    {
        niveau_intervention = unNiveauIntervention;
    }

    /// <summary>
    /// Obtient la formation du technicien.
    /// </summary>
    /// <returns>La formation du technicien.</returns>
    1 reference
    public string GetFormation()
    {
        return formationT;
    }

    /// <summary>
    /// Modifie la formation du technicien.
    /// </summary>
    /// <param name="Uneformation">La nouvelle formation du technicien.</param>
    0 references
    public void SetFormation(string Uneformation)
    {
        formationT = Uneformation;
    }

    /// <summary>
    /// Obtient le mot de passe du technicien.
    /// </summary>
    /// <returns>Le mot de passe du technicien.</returns>
    1 reference
    public string GetMdp()
    {
        return mdp;
    }

```

```

        return mdp;
    }

    /// <summary>
    /// Modifie le mot de passe du technicien.
    /// </summary>
    /// <param name="UnMdp">Le nouveau mot de passe du technicien.</param>
    0 references
    public void SetMdp(string UnMdp)
    {
        mdp = UnMdp;
    }
}

```

```

public class Technicien
{
    private string id_technicien;
    private string niveau_intervention;
    private string formationT;
    private string mdp;

    /// <summary>
    /// Constructeur pour initialiser un technicien avec un identifiant, un niveau
d'intervention, une formation et un mot de passe.
    /// </summary>
    /// <param name="id_technicien">L'identifiant unique du technicien.</param>
    /// <param name="niveau_intervention">Le niveau d'intervention du
technicien.</param>
    /// <param name="formationT">La formation du technicien.</param>
    /// <param name="mdp">Le mot de passe du technicien.</param>
    public Technicien(string id_technicien, string niveau_intervention, string formationT,
string mdp)
    {
        this.id_technicien = id_technicien;
        this.niveau_intervention = niveau_intervention;
        this.formationT = formationT;
        this.mdp = mdp;
    }

    /// <summary>
    /// Constructeur pour initialiser un technicien avec un niveau d'intervention, une
formation et un mot de passe.
    /// </summary>
    /// <param name="niveau_intervention">Le niveau d'intervention du
technicien.</param>
    /// <param name="formationT">La formation du technicien.</param>
    /// <param name="mdp">Le mot de passe du technicien.</param>
    public Technicien(string niveau_intervention, string formationT, string mdp)
    {
        this.niveau_intervention = niveau_intervention;
        this.formationT = formationT;
        this.mdp = mdp;
    }

    /// <summary>
    /// Constructeur pour initialiser un technicien avec une formation et un mot de passe.
    /// </summary>
    /// <param name="formationT">La formation du technicien.</param>
    /// <param name="mdp">Le mot de passe du technicien.</param>
    public Technicien(string formationT, string mdp)
    {
        this.formationT = formationT;

```

```

        this.mdp = mdp;
    }

    /// <summary>
    /// Obtient l'identifiant du technicien.
    /// </summary>
    /// <returns>L'identifiant du technicien.</returns>
    public string GetIdTechnicien()
    {
        return id_technicien;
    }

    /// <summary>
    /// Modifie l'identifiant du technicien.
    /// </summary>
    /// <param name="unid_technicien">Le nouvel identifiant du technicien.</param>
    public void SetIdTechnicien(string unid_technicien)
    {
        id_technicien = unid_technicien;
    }

    /// <summary>
    /// Obtient le niveau d'intervention du technicien.
    /// </summary>
    /// <returns>Le niveau d'intervention du technicien.</returns>
    public string GetNiveauIntervention()
    {
        return niveau_intervention;
    }

    /// <summary>
    /// Modifie le niveau d'intervention du technicien.
    /// </summary>
    /// <param name="unNiveauIntervention">Le nouveau niveau d'intervention.</param>
    public void SetNiveauIntervention(string unNiveauIntervention)
    {
        niveau_intervention = unNiveauIntervention;
    }

    /// <summary>
    /// Obtient la formation du technicien.
    /// </summary>
    /// <returns>La formation du technicien.</returns>
    public string GetFormation()
    {
        return formationT;
    }

```

```

    /// <summary>
    /// Modifie la formation du technicien.
    /// </summary>
    /// <param name="Uneformation">La nouvelle formation du technicien.</param>
    public void SetFormation(string Uneformation)
    {
        formationT = Uneformation;
    }

    /// <summary>
    /// Obtient le mot de passe du technicien.
    /// </summary>
    /// <returns>Le mot de passe du technicien.</returns>
    public string GetMdp()
    {
        return mdp;
    }

    /// <summary>
    /// Modifie le mot de passe du technicien.
    /// </summary>
    /// <param name="UnMdp">Le nouveau mot de passe du technicien.</param>
    public void SetMdp(string UnMdp)
    {
        mdp = UnMdp;
    }
}
}

```

# Class Personnel

```
/// <summary>
/// Représente un personnel dans le système.
/// </summary>
10 references
public class Personnel
{
    /// Variables d'instance privées pour stocker les informations du personnel
    private string nomP;           // Nom du personnel
    private string matricule;      // Matricule du personnel
    private string identiteP;      // Identité du personnel
    private DateTime date_embauche; // Date d'embauche du personnel
    private string region;        // Région du personnel
    private string id_personnel;    // Identifiant unique du personnel
    private Boolean responsable;    // Indicateur de responsabilité du personnel
    private string mdp;           // Mot de passe du personnel

    /// <summary>
    /// Constructeur pour initialiser un nouveau personnel avec des valeurs fournies.
    /// </summary>
    /// <param name="nomP">Nom du personnel.</param>
    /// <param name="matricule">Matricule du personnel.</param>
    /// <param name="identiteP">Identité du personnel (individuelle ou résidentielle).</param>
    /// <param name="date_embauche">Date d'embauche du personnel.</param>
    /// <param name="region">Région du personnel.</param>
    /// <param name="id_personnel">Identifiant unique du personnel.</param>
    /// <param name="responsable">Indicateur de responsabilité du personnel.</param>
    /// <param name="mdp">Mot de passe du personnel.</param>
    0 references
    public Personnel(string nomP, string matricule, string identiteP, DateTime date_embauche, string region, string id_personnel, Boolean responsable, string mdp)...

    /// <summary>
    /// Constructeur pour initialiser un nouveau personnel sans l'identifiant unique.
    /// </summary>
    /// <param name="nomP">Nom du personnel.</param>
    /// <param name="matricule">Matricule du personnel.</param>
    /// <param name="identiteP">Identité du personnel (individuelle ou résidentielle).</param>
    /// <param name="date_embauche">Date d'embauche du personnel.</param>
    /// <param name="region">Région du personnel.</param>
}
```

```

/// <param name="responsable">Indicateur de responsabilité du personnel.</param>
/// <param name="mdp">Mot de passe du personnel.</param>
2 references
public Personnel(string nomP, string matricule, string identiteP, DateTime date_embauche, string region, Boolean responsable, string mdp)

/// <summary>
/// Obtient le nom du personnel.
/// </summary>
/// <returns>Le nom du personnel.</returns>
1 reference
public string getNomPO()...

/// <summary>
/// Obtient le matricule du personnel.
/// </summary>
/// <returns>Le matricule du personnel.</returns>
1 reference
public string getMatricule()...

/// <summary>
/// Obtient l'identité du personnel.
/// </summary>
/// <returns>L'identité du personnel.</returns>
1 reference
public string getIdentitePO()...

/// <summary>
/// Obtient la date d'embauche du personnel.
/// </summary>
/// <returns>La date d'embauche du personnel.</returns>
1 reference
public DateTime getDte_embauche()...

/// <summary>
/// Obtient la région du personnel.
/// </summary>
/// <returns>La région du personnel.</returns>
1 reference
public string getRegion()...

```

```

/// <summary>
/// Obtient l'identifiant unique du personnel.
/// </summary>
/// <returns>L'identifiant du personnel.</returns>
1 reference
public string getId_personnel()...

/// <summary>
/// Obtient l'indicateur de responsabilité du personnel.
/// </summary>
/// <returns>True si le personnel est responsable, sinon false.</returns>
1 reference
public bool getResponsable()...

/// <summary>
/// Modifie l'identité du personnel.
/// </summary>
/// <param name="uneidentite">Nouvelle identité à affecter.</param>
0 references
public void setIdentite(string uneidentite)...

/// <summary>
/// Modifie la région du personnel.
/// </summary>
/// <param name="uneregion">Nouvelle région à affecter.</param>
0 references
public void setRegion(string uneregion)...

/// <summary>
/// Modifie la date d'embauche du personnel.
/// </summary>
/// <param name="unedate_embauche">Nouvelle date d'embauche à affecter.</param>
0 references
public void setDateEmbauche(DateTime unedate_embauche)...

```

```

public class Personnel
{
    // Variables d'instance privées pour stocker les informations du personnel
    private string nomP;           // Nom du personnel

```



```

private string matricule;    // Matricule du personnel
private string identiteP;    // Identité du personnel
private DateTime date_embauche; // Date d'embauche du personnel
private string region;      // Région du personnel
private string id_personnel; // Identifiant unique du personnel
private Boolean responsable; // Indicateur de responsabilité du personnel
private string mdp;         // Mot de passe du personnel

/// <summary>
/// Constructeur pour initialiser un nouveau personnel avec des valeurs fournies.
/// </summary>
/// <param name="nomP">Nom du personnel.</param>
/// <param name="matricule">Matricule du personnel.</param>
/// <param name="identiteP">Identité du personnel (individuelle ou
résidentielle).</param>
/// <param name="date_embauche">Date d'embauche du personnel.</param>
/// <param name="region">Région du personnel.</param>
/// <param name="id_personnel">Identifiant unique du personnel.</param>
/// <param name="responsable">Indicateur de responsabilité du personnel.</param>
/// <param name="mdp">Mot de passe du personnel.</param>
public Personnel(string nomP, string matricule, string identiteP, DateTime
date_embauche, string region, string id_personnel, Boolean responsable, string mdp)
{
    this.nomP = nomP;
    this.matricule = matricule;
    this.identiteP = identiteP;
    this.date_embauche = date_embauche;
    this.region = region;
    this.id_personnel = id_personnel;
    this.responsable = responsable;
    this.mdp = mdp;
}

/// <summary>
/// Constructeur pour initialiser un nouveau personnel sans l'identifiant unique.
/// </summary>
/// <param name="nomP">Nom du personnel.</param>
/// <param name="matricule">Matricule du personnel.</param>
/// <param name="identiteP">Identité du personnel (individuelle ou
résidentielle).</param>
/// <param name="date_embauche">Date d'embauche du personnel.</param>
/// <param name="region">Région du personnel.</param>
/// <param name="responsable">Indicateur de responsabilité du personnel.</param>
/// <param name="mdp">Mot de passe du personnel.</param>
public Personnel(string nomP, string matricule, string identiteP, DateTime
date_embauche, string region, Boolean responsable, string mdp)
{
    this.nomP = nomP;

```

```
this.matricule = matricule;
this.identiteP = identiteP;
this.date_embauche = date_embauche;
this.region = region;
this.responsable = responsable;
this.mdp = mdp;
}
```

```
/// <summary>
/// Obtient le nom du personnel.
/// </summary>
/// <returns>Le nom du personnel.</returns>
public string getNomP()
{
    return nomP;
}
```

```
/// <summary>
/// Obtient le matricule du personnel.
/// </summary>
/// <returns>Le matricule du personnel.</returns>
public string getMatricule()
{
    return matricule;
}
```

```
/// <summary>
/// Obtient l'identité du personnel.
/// </summary>
/// <returns>L'identité du personnel.</returns>
public string getIdentiteP()
{
    return identiteP;
}
```

```
/// <summary>
/// Obtient la date d'embauche du personnel.
/// </summary>
/// <returns>La date d'embauche du personnel.</returns>
public DateTime getDte_embauche()
{
    return date_embauche;
}
```

```
/// <summary>
/// Obtient la région du personnel.
/// </summary>
/// <returns>La région du personnel.</returns>
```

```

public string getRegion()
{
    return region;
}

/// <summary>
/// Obtient l'identifiant unique du personnel.
/// </summary>
/// <returns>L'identifiant du personnel.</returns>
public string getId_personnel()
{
    return id_personnel;
}

/// <summary>
/// Obtient l'indicateur de responsabilité du personnel.
/// </summary>
/// <returns>True si le personnel est responsable, sinon false.</returns>
public bool getResponsable()
{
    return responsable;
}

/// <summary>
/// Modifie l'identité du personnel.
/// </summary>
/// <param name="uneidentite">Nouvelle identité à affecter.</param>
public void setIdentite(string uneidentite)
{
    if (uneidentite == "individuelle")
    {
        identiteP = uneidentite;
    }
    if (uneidentite == "residentielle")
    {
        identiteP = uneidentite;
    }
}

/// <summary>
/// Modifie la région du personnel.
/// </summary>
/// <param name="uneregion">Nouvelle région à affecter.</param>
public void setRegion(string uneregion)
{
    region = uneregion; // Cette ligne doit modifier l'attribut `region`, pas la variable locale
}

```

```

    /// <summary>
    /// Modifie la date d'embauche du personnel.
    /// </summary>
    /// <param name="unedate_embauche">Nouvelle date d'embauche à affecter.</param>
    public void setdateEmbauche(DateTime unedate_embauche)
    {
        date_embauche = unedate_embauche;
    }
}
}

```

## *Class Fournisseurs*

```

public class Fournisseur
{
    private int id_fournisseur;    // Identifiant unique du fournisseur
    private string nom_fournisseur; // Nom du fournisseur

    /// <summary>
    /// Constructeur de la classe Fournisseur.
    /// Initialise un fournisseur avec un identifiant et un nom.
    /// </summary>
    /// <param name="id_fournisseur">L'identifiant unique du fournisseur.</param>
    /// <param name="nom_fournisseur">Le nom du fournisseur.</param>
    0 references
    public Fournisseur(int id_fournisseur, string nom_fournisseur) ...

    /// <summary>
    /// Constructeur de la classe Fournisseur.
    /// Initialise un fournisseur avec seulement un nom.
    /// </summary>
    /// <param name="nom_fournisseur">Le nom du fournisseur.</param>
    1 reference
    public Fournisseur(string nom_fournisseur) ...

    /// <summary>
    /// Obtient l'identifiant du fournisseur.
    /// </summary>
    /// <returns>L'identifiant unique du fournisseur.</returns>
    1 reference
    public int getId_fournisseur() ...

    /// <summary>
    /// Obtient le nom du fournisseur.
    /// </summary>
    /// <returns>Le nom du fournisseur.</returns>
    1 reference
    public string getNomFournisseur() ...
}

```

```

public class Fournisseur
{
    private int id_fournisseur;    // Identifiant unique du fournisseur
    private string nom_fournisseur; // Nom du fournisseur

```

```

/// <summary>
/// Constructeur de la classe Fournisseur.
/// Initialise un fournisseur avec un identifiant et un nom.
/// </summary>
/// <param name="id_fournisseur">L'identifiant unique du fournisseur.</param>
/// <param name="nom_fournisseur">Le nom du fournisseur.</param>
public Fournisseur(int id_fournisseur, string nom_fournisseur)
{
    this.id_fournisseur = id_fournisseur; // Assigner l'identifiant du fournisseur
    this.nom_fournisseur = nom_fournisseur; // Assigner le nom du fournisseur
}

/// <summary>
/// Constructeur de la classe Fournisseur.
/// Initialise un fournisseur avec seulement un nom.
/// </summary>
/// <param name="nom_fournisseur">Le nom du fournisseur.</param>
public Fournisseur(string nom_fournisseur)
{
    this.nom_fournisseur = nom_fournisseur; // Assigner le nom du fournisseur
}

/// <summary>
/// Obtient l'identifiant du fournisseur.
/// </summary>
/// <returns>L'identifiant unique du fournisseur.</returns>
public int getId_fournisseur()
{
    return id_fournisseur; // Retourne l'identifiant du fournisseur
}

/// <summary>
/// Obtient le nom du fournisseur.
/// </summary>
/// <returns>Le nom du fournisseur.</returns>
public string getNomFournisseur()
{
    return nom_fournisseur; // Retourne le nom du fournisseur
}
}

```

## **Class Tickets**

```

public class Ticket
{
    // Attributs privés de la classe Ticket
    private int id_ticket;           // Identifiant unique du ticket
    private string urgence;          // Urgence du ticket (niveau de priorité)
    private string etat;             // État du ticket (ouvert, fermé, en cours, etc.)
    private string type_demande;     // Type de demande (catégorie du ticket)
    private string date_ticket;      // Date à laquelle le ticket a été créé
    private int idtechnicien;        // Identifiant du technicien assigné au ticket
    private string id_materiel;      // Identifiant du matériel lié au ticket
    private string matricule;        // Matricule de la personne ayant créé ou rapporté le ticket
    private string description;      // Description du problème ou de la demande


    /// <summary>
    /// Constructeur pour initialiser un nouveau ticket avec des valeurs fournies.
    /// </summary>
    /// <param name="id_ticket">Identifiant du ticket.</param>
    /// <param name="urgence">Urgence du ticket (niveau de priorité).</param>
    /// <param name="etat">État du ticket (ouvert, fermé, en cours, etc.).</param>
    /// <param name="type_demande">Type de demande (catégorie du ticket).</param>
    /// <param name="date_ticket">Date de création du ticket.</param>
    /// <param name="idtechnicien">Identifiant du technicien assigné.</param>
    /// <param name="id_materiel">Identifiant du matériel concerné par le ticket.</param>
    /// <param name="matricule">Matricule de la personne ayant rapporté le ticket.</param>
    /// <param name="description">Description détaillée du problème ou de la demande.</param>
    1 reference
    public Ticket(int id_ticket, string urgence, string etat, string type_demande, string date_ticket, int idtechnicien, string id_materiel, string matricule, string description)...


    /// <summary>
    /// Constructeur pour initialiser un ticket sans identifiant, avec les autres informations.
    /// </summary>
    /// <param name="urgence">Urgence du ticket (niveau de priorité).</param>
    /// <param name="etat">État du ticket (ouvert, fermé, en cours, etc.).</param>
    /// <param name="type_demande">Type de demande (catégorie du ticket).</param>
    /// <param name="date_ticket">Date de création du ticket.</param>
    /// <param name="idtechnicien">Identifiant du technicien assigné.</param>
    /// <param name="id_materiel">Identifiant du matériel concerné par le ticket.</param>
    /// <param name="matricule">Matricule de la personne ayant rapporté le ticket.</param>
    /// <param name="description">Description détaillée du problème ou de la demande.</param>


```


```
public Ticket(string urgence, string etat, string type_demande, string date_ticket, int idtechnicien, string id_materiel, string matricule, string description)


// Méthodes getter et setter pour chaque attribut

/// <summary>
/// Retourne l'identifiant du ticket.
/// </summary>
/// <returns>L'identifiant du ticket.</returns>
1 reference
public int GetIdTicket()

/// <summary>
/// Modifie l'identifiant du ticket.
/// </summary>
/// <param name="unidticket">Nouvel identifiant du ticket.</param>
0 references
public void SetIdTicket(int unidticket)

/// <summary>
/// Retourne l'urgence du ticket.
/// </summary>
/// <returns>Le niveau d'urgence du ticket.</returns>
2 references
public string GetUrgence()

/// <summary>
/// Modifie l'urgence du ticket.
/// </summary>
/// <param name="uneurgence">Nouveau niveau d'urgence pour le ticket.</param>
0 references
public void SetUrgence(string uneurgence)

/// <summary>
/// Retourne l'état du ticket.
/// </summary>
/// <returns>L'état actuel du ticket.</returns>
2 references
public string GetEtat()
```

```

/// Modifie l'état du ticket.
/// </summary>
/// <param name="unetat">Nouvel état pour le ticket.</param>
0 references
public void SetEtat(string unetat)...

/// <summary>
/// Retourne le type de demande du ticket.
/// </summary>
/// <returns>Le type de demande du ticket.</returns>
1 reference
public string GetTypeDemande()...

/// <summary>
/// Modifie le type de demande du ticket.
/// </summary>
/// <param name="untype">Nouveau type de demande pour le ticket.</param>
0 references
public void SetTypeDemande(string untype)...

/// <summary>
/// Retourne la date de création du ticket.
/// </summary>
/// <returns>La date de création du ticket.</returns>
1 reference
public string GetDateTicket()...

/// <summary>
/// Modifie la date de création du ticket.
/// </summary>
/// <param name="uneDate">Nouvelle date de création du ticket.</param>
0 references
public void SetDateTicket(string uneDate)...

/// <summary>
/// Retourne l'identifiant du technicien assigné au ticket.
/// </summary>
/// <returns>L'identifiant du technicien assigné au ticket.</returns>
1 reference
public int GetIdTechnicien()...

```



```

/// <summary>
/// Modifie l'identifiant du technicien assigné au ticket.
/// </summary>
/// <param name="unidtech">Nouvel identifiant pour le technicien assigné.</param>
0 references
public void SetIdTechnicien(int unidtech)...

/// <summary>
/// Retourne l'identifiant du matériel lié au ticket.
/// </summary>
/// <returns>L'identifiant du matériel concerné par le ticket.</returns>
1 reference
public string GetIdMateriel()...

/// <summary>
/// Modifie l'identifiant du matériel lié au ticket.
/// </summary>
/// <param name="unidmat">Nouvel identifiant pour le matériel lié au ticket.</param>
0 references
public void SetIdMateriel(string unidmat)...

/// <summary>
/// Retourne le matricule de la personne ayant rapporté le ticket.
/// </summary>
/// <returns>Le matricule de la personne ayant rapporté le ticket.</returns>
1 reference
public string GetMatricule()...

/// <summary>
/// Retourne la description du ticket.
/// </summary>
/// <returns>La description détaillée du problème ou de la demande.</returns>
1 reference
public string GetDescription()...

/// <summary>
/// Modifie le matricule de la personne ayant rapporté le ticket.
/// </summary>
/// <param name="unmatricule">Nouveau matricule pour la personne ayant rapporté le ticket.</param>
0 references
public void SetMatricule(string unmatricule)...

/// <summary>
/// Modifie la description du ticket.
/// </summary>
/// <param name="unedescription">Nouvelle description pour le ticket.</param>
0 references
public void SetDescription(string unedescription)...

```

```

public class Ticket
{
    // Attributs privés de la classe Ticket
    private int id_ticket;      // Identifiant unique du ticket
    private string urgence;     // Urgence du ticket (niveau de priorité)
    private string etat;        // État du ticket (ouvert, fermé, en cours, etc.)
    private string type_demande; // Type de demande (catégorie du ticket)
    private string date_ticket; // Date à laquelle le ticket a été créé
    private int idtechnicien;   // Identifiant du technicien assigné au ticket
    private string id_materiel; // Identifiant du matériel lié au ticket
    private string matricule;    // Matricule de la personne ayant créé ou rapporté le ticket
}

```

```

private string description;    // Description du problème ou de la demande

/// <summary>
/// Constructeur pour initialiser un nouveau ticket avec des valeurs fournies.
/// </summary>
/// <param name="id_ticket">Identifiant du ticket.</param>
/// <param name="urgence">Urgence du ticket (niveau de priorité).</param>
/// <param name="etat">État du ticket (ouvert, fermé, en cours, etc.).</param>
/// <param name="type_demande">Type de demande (catégorie du ticket).</param>
/// <param name="date_ticket">Date de création du ticket.</param>
/// <param name="idtechnicien">Identifiant du technicien assigné.</param>
/// <param name="id_materiel">Identifiant du matériel concerné par le ticket.</param>
/// <param name="matricule">Matricule de la personne ayant rapporté le
ticket.</param>
/// <param name="description">Description détaillée du problème ou de la
demande.</param>
public Ticket(int id_ticket, string urgence, string etat, string type_demande, string
date_ticket, int idtechnicien, string id_materiel, string matricule, string description)
{
    this.id_ticket = id_ticket;        // Assigne l'ID du ticket
    this.urgence = urgence;            // Assigne le niveau d'urgence
    this.etat = etat;                  // Assigne l'état du ticket
    this.type_demande = type_demande;  // Assigne le type de demande
    this.date_ticket = date_ticket;    // Assigne la date de création du ticket
    this.idtechnicien = idtechnicien;   // Assigne l'ID du technicien responsable
    this.id_materiel = id_materiel;     // Assigne l'ID du matériel concerné
    this.matricule = matricule;         // Assigne le matricule de la personne rapportant le
ticket
    this.description = description;    // Assigne la description du problème ou de la
demande
}

/// <summary>
/// Constructeur pour initialiser un ticket sans identifiant, avec les autres informations.
/// </summary>
/// <param name="urgence">Urgence du ticket (niveau de priorité).</param>
/// <param name="etat">État du ticket (ouvert, fermé, en cours, etc.).</param>
/// <param name="type_demande">Type de demande (catégorie du ticket).</param>
/// <param name="date_ticket">Date de création du ticket.</param>
/// <param name="idtechnicien">Identifiant du technicien assigné.</param>
/// <param name="id_materiel">Identifiant du matériel concerné par le ticket.</param>
/// <param name="matricule">Matricule de la personne ayant rapporté le
ticket.</param>
/// <param name="description">Description détaillée du problème ou de la
demande.</param>
public Ticket(string urgence, string etat, string type_demande, string date_ticket, int
idtechnicien, string id_materiel, string matricule, string description)
{

```

```

        this.urgence = urgence;          // Assigne le niveau d'urgence
        this.etat = etat;                // Assigne l'état du ticket
        this.type_demande = type_demande; // Assigne le type de demande
        this.date_ticket = date_ticket;  // Assigne la date de création du ticket
        this.idtechnicien = idtechnicien; // Assigne l'ID du technicien responsable
        this.id_materiel = id_materiel;   // Assigne l'ID du matériel concerné
        this.matricule = matricule;       // Assigne le matricule de la personne rapportant le
ticket
        this.description = description;   // Assigne la description du problème ou de la
demande
    }

```

// Méthodes getter et setter pour chaque attribut

```

/// <summary>
/// Retourne l'identifiant du ticket.
/// </summary>
/// <returns>L'identifiant du ticket.</returns>
public int GetIdTicket()
{
    return id_ticket;
}

```

```

/// <summary>
/// Modifie l'identifiant du ticket.
/// </summary>
/// <param name="unidticket">Nouvel identifiant du ticket.</param>
public void SetIdTicket(int unidticket)
{
    id_ticket = unidticket;
}

```

```

/// <summary>
/// Retourne l'urgence du ticket.
/// </summary>
/// <returns>Le niveau d'urgence du ticket.</returns>
public string GetUrgence()
{
    return urgence;
}

```

```

/// <summary>
/// Modifie l'urgence du ticket.
/// </summary>
/// <param name="uneurgence">Nouveau niveau d'urgence pour le ticket.</param>
public void SetUrgence(string uneurgence)
{
    urgence = uneurgence;
}

```

```

}

/// <summary>
/// Retourne l'état du ticket.
/// </summary>
/// <returns>L'état actuel du ticket.</returns>
public string GetEtat()
{
    return etat;
}

/// <summary>
/// Modifie l'état du ticket.
/// </summary>
/// <param name="unetat">Nouvel état pour le ticket.</param>
public void SetEtat(string unetat)
{
    etat = unetat;
}

/// <summary>
/// Retourne le type de demande du ticket.
/// </summary>
/// <returns>Le type de demande du ticket.</returns>
public string GetTypeDemande()
{
    return type_demande;
}

/// <summary>
/// Modifie le type de demande du ticket.
/// </summary>
/// <param name="untype">Nouveau type de demande pour le ticket.</param>
public void SetTypeDemande(string untype)
{
    type_demande = untype;
}

/// <summary>
/// Retourne la date de création du ticket.
/// </summary>
/// <returns>La date de création du ticket.</returns>
public string GetDateTicket()
{
    return date_ticket;
}

/// <summary>

```

```

/// Modifie la date de création du ticket.
/// </summary>
/// <param name="uneDate">Nouvelle date de création du ticket.</param>
public void SetDateTicket(string uneDate)
{
    date_ticket = uneDate;
}

/// <summary>
/// Retourne l'identifiant du technicien assigné au ticket.
/// </summary>
/// <returns>L'identifiant du technicien assigné au ticket.</returns>
public int GetIdTechnicien()
{
    return idtechnicien;
}

/// <summary>
/// Modifie l'identifiant du technicien assigné au ticket.
/// </summary>
/// <param name="unidtech">Nouvel identifiant pour le technicien assigné.</param>
public void SetIdTechnicien(int unidtech)
{
    idtechnicien = unidtech;
}

/// <summary>
/// Retourne l'identifiant du matériel lié au ticket.
/// </summary>
/// <returns>L'identifiant du matériel concerné par le ticket.</returns>
public string GetIdMateriel()
{
    return id_materiel;
}

/// <summary>
/// Modifie l'identifiant du matériel lié au ticket.
/// </summary>
/// <param name="unidmat">Nouvel identifiant pour le matériel lié au ticket.</param>
public void SetIdMateriel(string unidmat)
{
    id_materiel = unidmat;
}

/// <summary>
/// Retourne le matricule de la personne ayant rapporté le ticket.
/// </summary>
/// <returns>Le matricule de la personne ayant rapporté le ticket.</returns>

```

```

public string GetMatricule()
{
    return matricule;
}

/// <summary>
/// Retourne la description du ticket.
/// </summary>
/// <returns>La description détaillée du problème ou de la demande.</returns>
public string GetDescription()
{
    return description;
}

/// <summary>
/// Modifie le matricule de la personne ayant rapporté le ticket.
/// </summary>
/// <param name="unmatricule">Nouveau matricule pour la personne ayant rapporté le
ticket.</param>
public void SetMatricule(string unmatricule)
{
    matricule = unmatricule;
}

/// <summary>
/// Modifie la description du ticket.
/// </summary>
/// <param name="unedescription">Nouvelle description pour le ticket.</param>
public void SetDescription(string unedescription)
{
    description = unedescription;
}
}
}

```

# Class Compétences

```
public class Competence
{
    private int id_competence; // Identifiant unique de la compétence
    private string description; // Description de la compétence

    /// <summary>
    /// Constructeur de la classe Competence.
    /// Initialise une nouvelle compétence avec un identifiant et une description.
    /// </summary>
    /// <param name="id_competence">L'identifiant unique de la compétence.</param>
    /// <param name="description">La description de la compétence.</param>
    0 references
    public Competence(int id_competence, string description) ...

    /// <summary>
    /// Constructeur de la classe Competence.
    /// Initialise une nouvelle compétence avec seulement une description.
    /// </summary>
    /// <param name="description">La description de la compétence.</param>
    1 reference
    public Competence(string description) ...

    /// <summary>
    /// Obtient l'identifiant de la compétence.
    /// </summary>
    /// <returns>L'identifiant de la compétence.</returns>
    1 reference
    public int getId_competence() ...

    /// <summary>
    /// Obtient la description de la compétence.
    /// </summary>
    /// <returns>La description de la compétence.</returns>
    1 reference
    public string getDescription() ...

    /// <summary>
    /// Modifie la description de la compétence.
    /// </summary>
    /// <param name="LaDescription">La nouvelle description de la compétence.</param>
    0 references
    public void setDescription(string LaDescription) ...
}
```

```
public class Competence
{
    private int id_competence; // Identifiant unique de la compétence
    private string description; // Description de la compétence

    /// <summary>
    /// Constructeur de la classe Competence.
    /// Initialise une nouvelle compétence avec un identifiant et une description.
    /// </summary>
    /// <param name="id_competence">L'identifiant unique de la compétence.</param>
```

```

/// <param name="description">La description de la compétence.</param>
public Competence(int id_competence, string description)
{
    this.id_competence = id_competence; // Assigner l'identifiant de la compétence
    this.description = description;    // Assigner la description de la compétence
}

/// <summary>
/// Constructeur de la classe Competence.
/// Initialise une nouvelle compétence avec seulement une description.
/// </summary>
/// <param name="description">La description de la compétence.</param>
public Competence(string description)
{
    this.description = description; // Assigner la description de la compétence
}

/// <summary>
/// Obtient l'identifiant de la compétence.
/// </summary>
/// <returns>L'identifiant de la compétence.</returns>
public int getId_competence()
{
    return id_competence;
}

/// <summary>
/// Obtient la description de la compétence.
/// </summary>
/// <returns>La description de la compétence.</returns>
public string getDescription()
{
    return description;
}

/// <summary>
/// Modifie la description de la compétence.
/// </summary>
/// <param name="LaDescription">La nouvelle description de la compétence.</param>
public void setDescription(string LaDescription)
{
    description = LaDescription; // Correction: assigner la nouvelle description à l'attribut
'description'
}
}

```



# **Programme Principal**

4 references

```
public partial class Form1 : Form
{
```

```
    /// <summary>
    /// Initialise une nouvelle instance de la classe <see cref="Form1"/>.
    /// Configure les onglets visibles au démarrage de l'application.
    /// </summary>
```

1 reference

```
    public Form1()...
```

```
    /// <summary>
    /// Gère l'événement de clic sur le bouton de connexion.
    /// Active ou désactive les onglets en fonction du rôle saisi.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>
```

1 reference

```
    private void buttonConnexion_Click(object sender, EventArgs e)...
```

```
    /// <summary>
    /// Ajoute un nouveau matériel à la base de données.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>
```

1 reference

```
    private void buttonAjoutMateriel_Click(object sender, EventArgs e)...
```

```
    /// <summary>
    /// Supprime un matériel en fonction de son ID.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>
```

1 reference

```
    private void buttonSupprimerMateriel_Click(object sender, EventArgs e)...
```

```
    /// <summary>
    /// Ajoute un nouveau technicien à la base de données.
```

```

/// Ajoute un nouveau technicien à la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
1 reference
private void boutonAjoutTechnicien_Click(object sender, EventArgs e)...
/// <summary>
/// Ajoute un nouvel utilisateur à la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
1 reference
private void boutonAjoutUtilisateur_Click(object sender, EventArgs e)...
/// <summary>
/// Supprime un technicien en fonction de son ID.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

1 reference
private void boutonSuppTech_Click(object sender, EventArgs e)...
/// <summary>
/// Supprime un utilisateur en fonction de son ID.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
1 reference
private void boutonSuppPersonnel_Click(object sender, EventArgs e)...
/// <summary>
/// Modifie les informations d'un technicien dans la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

1 reference
private void boutonModifierTech_Click(object sender, EventArgs e)...
/// <summary>
/// Modifie les informations d'un utilisateur dans la base de données.
/// </summary>

```

```

/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
1 reference
private void boutonModifierUtilisateur_Click(object sender, EventArgs e)
{
    /// <summary>
    /// Affiche tous les matériels dans la liste.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>

    1 reference
private void boutonAfficherMateriel_Click(object sender, EventArgs e)
{
    /// <summary>
    /// Affiche tous les incidents dans la liste.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>

    1 reference
private void boutonConsulterIncidents_Click(object sender, EventArgs e)
{
    /// <summary>
    /// Déclare un nouveau ticket dans la base de données.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>

    1 reference
private void boutonModifierEtat_Click(object sender, EventArgs e)
{
    1 reference
private void boutonDeclarer_Click(object sender, EventArgs e)
{

```

```

public partial class Form1 : Form
{
    /// <summary>
    /// Initialise une nouvelle instance de la classe <see cref="Form1"/>.
    /// Configure les onglets visibles au démarrage de l'application.
    /// </summary>
    public Form1()
    {
        InitializeComponent();
        tabControl1.TabPages.Remove(tabPage2);
        tabControl1.TabPages.Remove(tabPage3);
        tabControl1.TabPages.Remove(tabPage4);
    }

    /// <summary>
    /// Gère l'événement de clic sur le bouton de connexion.

```

```

/// Active ou désactive les onglets en fonction du rôle saisi.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

private void buttonConnexion_Click(object sender, EventArgs e)
{

    if (textBox1.Text == "Tech1")
    {

        tabControl1.TabPages.Add(tabPage2);
        tabControl1.TabPages.Add(tabPage3);
        tabControl1.TabPages.Add(tabPage4);
        tabControl1.TabPages.Remove(tabPage1);

    }
    if (textBox1.Text == "Perso2")
    {

        tabControl1.TabPages.Add(tabPage3);
        tabControl1.TabPages.Remove(tabPage1);

    }

}

/// <summary>
/// Ajoute un nouveau matériel à la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

private void buttonAjoutMateriel_Click(object sender, EventArgs e)
{
    Materiel unMateriel = new Materiel(Convert.ToString(textBoxType.Text),
Convert.ToString(textBoxMemoire.Text), Convert.ToString(textBoxProcesseur.Text),
Convert.ToString(textBoxLogiciel.Text), textBoxDateAchat.Text,
Convert.ToInt32(textBoxFournisseur.Text));

    BD.AddMateriel(unMateriel);

}

```

```

/// <summary>
/// Supprime un matériel en fonction de son ID.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
private void buttonSupprimerMateriel_Click(object sender, EventArgs e)
{
    int id = Convert.ToInt32(textBoxsuppMat.Text);
    BD.SuppMateriel(id);

}
/// <summary>
/// Ajoute un nouveau technicien à la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
private void buttonAjoutTechnicien_Click(object sender, EventArgs e)
{
    Technicien unTechnicien = new Technicien(Convert.ToString(textBoxNvInter.Text),
Convert.ToString(textBoxFormation.Text), Convert.ToString(textBoxMdpT.Text));
    BD.AddTechnicien(unTechnicien);
}
/// <summary>
/// Ajoute un nouvel utilisateur à la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
private void buttonAjoutUtilisateur_Click(object sender, EventArgs e)
{
    Personnel unPersonnel = new Personnel(Convert.ToString(textBoxNomP.Text),
Convert.ToString(textBoxMatricule.Text), Convert.ToString(textBoxidentite.Text),
Convert.ToDateTime(textBoxDateEmbauche.Text), textBoxRegion.Text,
checkBoxResponsable.Checked, Convert.ToString(textBoxMdpU.Text));
    BD.AddPersonnel(unPersonnel);

}
/// <summary>
/// Supprime un technicien en fonction de son ID.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

private void buttonSuppTech_Click(object sender, EventArgs e)
{
    int id = Convert.ToInt32(textBoxIDTsupp.Text);
    BD.SuppTechnicien(id);
}

```

```

    }
    /// <summary>
    /// Supprime un utilisateur en fonction de son ID.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>
    private void buttonSuppPersonnel_Click(object sender, EventArgs e)
    {
        int id = Convert.ToInt32(textBoxIDPSupp.Text);
        BD.SuppPersonnel(id);
    }
    /// <summary>
    /// Modifie les informations d'un technicien dans la base de données.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>

    private void buttonModifierTech_Click(object sender, EventArgs e)
    {
        int id = Convert.ToInt32(textBoxIdModifT.Text);
        string formation = textBoxModifFormT.Text;
        BD.ModifTechnicien(id, formation);
    }
    /// <summary>
    /// Modifie les informations d'un utilisateur dans la base de données.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>
    private void buttonModifierUtilisateur_Click(object sender, EventArgs e)
    {
        int id = Convert.ToInt32(textBoxIdmodifU.Text);
        string identite = textBoxModifUtili.Text;
        BD.ModifPersonnel(id, identite);
    }
    /// <summary>
    /// Affiche tous les matériels dans la liste.
    /// </summary>
    /// <param name="sender">Objet qui déclenche l'événement.</param>
    /// <param name="e">Données associées à l'événement.</param>

    private void buttonAfficherMateriel_Click(object sender, EventArgs e)
    {
        listBoxMateriel.Items.Clear();

        foreach (Materiel unMateriel in BD.SelectMateriel())

```

```

    {

        listBoxMateriel.Items.Add(Convert.ToString((unMateriel.getId_materiel() + " " +
unMateriel.getleType())));
    }

}
/// <summary>
/// Affiche tous les incidents dans la liste.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>
private void buttonConsulterIncidents_Click(object sender, EventArgs e)
{ listBoxIncidents.Items.Clear();
  foreach (Ticket unTicket in BD.SelectTicket())
  {

      listBoxIncidents.Items.Add(Convert.ToString((unTicket.GetIdTicket() + " " +
unTicket.GetEtat()+ " " + unTicket.GetTypeDemande() + " " + unTicket.GetUrgence())));
  }

}

/// <summary>
/// Déclare un nouveau ticket dans la base de données.
/// </summary>
/// <param name="sender">Objet qui déclenche l'événement.</param>
/// <param name="e">Données associées à l'événement.</param>

private void buttonModifierEtat_Click(object sender, EventArgs e)
{
    int idT = Convert.ToInt32(textBoxIdT.Text);
    string etat = Convert.ToString(textBoxEtat.Text);
    string Description = Convert.ToString(textBoxTravailE.Text);
    BD.ModifTicket (idT,etat, Description);

}

private void buttonDeclarer_Click(object sender, EventArgs e)
{
    Ticket unTicket = new Ticket(textBoxUrgence.Text, textBoxEtat1.Text,
textBoxTypeD.Text, textBoxDate1.Text);
    BD.AddTicket(unTicket);
}
}
}

```



# L'interface

Connexion Materiel Incidents Ajout/Modif

Utilisateurs

Login

Mdp

Connexion

Connexion Materiel Incidents Ajout/Modif

Processeur

Type

Mémoire

Id Fournisseur

id\_matériel

Logiciel Installé

Date Achat

listBoxMateriel

Ajouter Materiel

Supprimer Materiel

Afficher Materiel

Connexion
Materiel
Incidents
Ajout/Modif

listBoxIncidents

Consulter Incidents

Urgence

etat

Type\_demande

Date

Déclarer incident

ID

Etat

Travail effectué

Modifier Etat

Connexion
Materiel
Incidents
Ajout/Modif

Region

Niveau intervention

Formation

Mot de passe

Ajout Technicien

id

Formation

Modifier Technicien

Matricule

identite

Nom Prenom

Ajout Utilisateur

id

identite

Modifier Personnel

Date Embauche

Responsable

Mot de passe

Supprimer Technicien

Supprimer Personnel

**Prototype**

Afin de se connecter à l'application les utilisateurs arrivent sur cette page afin de se connecter :

Connexion

Utilisateurs

Login

Mdp

Connexion

Si on se connecte en tant que technicien, il y aura une redirection vers ces pages :

Materiel Incidents Ajout/Modif

Processeur

id\_matériel

Type

Logiciel Installé

Supprimer Materiel

Mémoire

Date Achat

Id Fournisseur

Ajout Materiel

Afficher Materiel

Materiel

Incidents

Ajout/Modif

ID

Etat

Travail effectué

Modifier Etat

Consulter Incidents

Urgence

etat

Type\_demande

Date

Déclarer incident

Materiel

Incidents

Ajout/Modif

Region

Niveau intervention

Matricule

Date Embauche

id

id

Formation

identite

☐ Responsable

Supprimer Technicien

Supprimer Personnel

Mot de passe

Nom Prenom

Mot de passe

Ajout Technicien

Ajout Utilisateur

id

id

Formation

identite

Modifier Technicien

Modifier Personnel

Sinon si on est un simple utilisateur, on aura uniquement accès à cette page :

Incidents

Consulter Incidents

Urgence

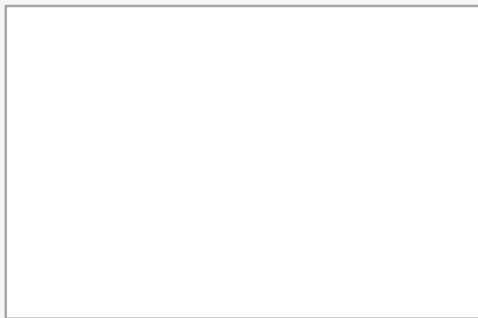
etat

Type\_demande

Date

Déclarer incident

## Incidents



ID

Etat

Travail effectué

Modifier Etat

Consulter Incidents

Urgence

etat

Type\_demande

Date

Déclarer incident