Programme principale :

Programme principale :

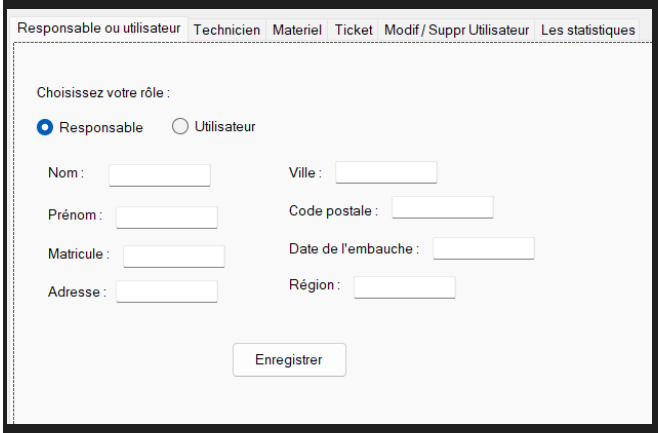
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Projet_final_C_
{
    public partial class Form1 : Form
    {
        private Technicien technicienAModifier;
        private Utilisateur utilisateurAModifier;

        public Form1()
        {
            InitializeComponent();
            this.Load += Form1_Load; // au cas où l'event n'était pas
relié
        }

        internal Form1(Technicien unTechnicien)
        {
            InitializeComponent();
            technicienAModifier = unTechnicien;  // si tu veux aussi
préremplir l'onglet technicien
            RemplirInformationsTechnicien();
        }

        private void RemplirInformationsTechnicien()
        {
            //vérifie s'il y a bien un technicien selectionner
            if (technicienAModifier != null)
            {
                // Sélectionner l'onglet Technicien (index 1 d'après
Designer.cs : tabPage2 est ajouté en 2eme)
```

```
                tabControl1.SelectedTab = tabPage2;

                nomT.Text = technicienAModifier.getNomt();
                prenomT.Text = technicienAModifier.getPrenomT();
                matriculeT.Text = technicienAModifier.getMatriculeT();
                adresseT.Text = technicienAModifier.getAdresseT();
                villeT.Text = technicienAModifier.getVilleT();
                cpT.Text = technicienAModifier.getCodePT();
                date_embaucheT.Text =
technicienAModifier.getDate_embaucheT().ToString("dd-MM-yyyy");
                formationT.Text = technicienAModifier.getFormation();
                regionT.Text = technicienAModifier.getRegionT();

                //coche automatiquement le bon boutton radio
                string niveau =
technicienAModifier.getNiv_intervention();
                if (niveau == "Niveau 1")
                {
                    niveau_1.Checked = true;
                }
                else if (niveau == "Niveau 2")
                {
                    niveau_2.Checked = true;
                }
                else if (niveau == "Niveau 3")
                {
                    niveau_3.Checked = true;
                }

                enregistrer_technicien.Text = "Modifier";
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            selectID.DataSource =
Base_de_donnee.GetTousLesIdsUtilisateurs();
            ChargerTicketsNonAttribuesDansCombo();
            selectID.DataSource =
Base_de_donnee.GetTousLesIdsUtilisateurs();
            ChargerTicketsNonAttribuesDansCombo();
            ChargerTechniciensDansCombo();
            //je rempli la combo en cours de traitement
```

```
            ChargerTicketsEnCoursDansComboResolu();
            ChargerTicketsResolusDansCombo();
            selectID.DataSource =
Base_de_donnee.GetTousLesIdsUtilisateurs();
            ChargerTicketsNonAttribuesDansCombo();
            selectID.DataSource =
Base_de_donnee.GetTousLesIdsUtilisateurs();
            ChargerTicketsNonAttribuesDansCombo();
            ChargerTechniciensDansCombo();
            ChargerUtilisateursDansComboTickets();
            // AJOUT :
            ChargerIdsMaterielsDansCombo();

            choix_responsable.DataSource =
Base_de_donnee.GetTousLesIdsResponsables();

            //relier la combo du technicien à l'évènement
            comboTechnicien.SelectedIndexChanged -=
comboTechnicien_SelectedIndexChanged; // évite les doublons
            comboTechnicien.SelectedIndexChanged +=
comboTechnicien_SelectedIndexChanged;

            var ids = Base_de_donnee.GetTousLesIdsUtilisateurs();
            id_utilisateur.DataSource = null;
            id_utilisateur.DataSource = ids;
            id_utilisateur.SelectedIndex = -1;

        }

        private void enregistrer_Click(object sender, EventArgs e)
        {

            // vérifie que l'un des deux boutons radio est coché
            if (!responsable.Checked && !utilisateur.Checked)
            {
                MessageBox.Show("Veuillez choisir un rôle.");
                return;
            }

            // Récupération des champs (noms EXACTS de ton Designer)
            string nomSaisi = nom.Text;
            string prenomSaisi = prenom.Text;
            string matriculeSaisi = matricule.Text;
```

```
            string adresseSaisie = adresse.Text;
            string villeSaisie = ville.Text;
            string cpSaisi = cp.Text;
            string dateEmbaucheSaisie = date_embauche.Text;
            string regionSaisie = textBox8.Text;

            // On s'assure que les champs essentiels ne sont pas vides
            if (nomSaisi == "" || prenomSaisi == "" || matriculeSaisi
== "")
            {
                MessageBox.Show("Nom, prénom et matricule
obligatoires.");
                return;
            }

            try
            {
                // ===== RESPONSABLE =====
                if (responsable.Checked)
                {
                    int idR = Base_de_donnee.GetNextIdResponsable();

                    Responsable r = new Responsable(
                        idR,
                        nomSaisi,
                        prenomSaisi,
                        matriculeSaisi,
                        adresseSaisie,
                        villeSaisie,
                        cpSaisi,
                        dateEmbaucheSaisie,
                        regionSaisie
                    );

                    Base_de_donnee.AjouterResponsable(r);
                    MessageBox.Show("Responsable enregistré.");
                }
                // ===== UTILISATEUR =====
                else
                {
                    int idU = Base_de_donnee.GetNextIdUtilisateur();

                    // Pour l'instant, responsable par défaut
```

```
                int idResponsable = 1;

                Utilisateur u = new Utilisateur(
                    idU,
                    nomSaisi,
                    prenomSaisi,
                    matriculeSaisi,
                    adresseSaisie,
                    villeSaisie,
                    cpSaisi,
                    dateEmbaucheSaisie,
                    regionSaisie,
                    idResponsable
                );

                Base_de_donnee.AjouterUtilisateur(u);
                MessageBox.Show("Utilisateur enregistré.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Erreur : " + ex.Message);
        }
    }
```



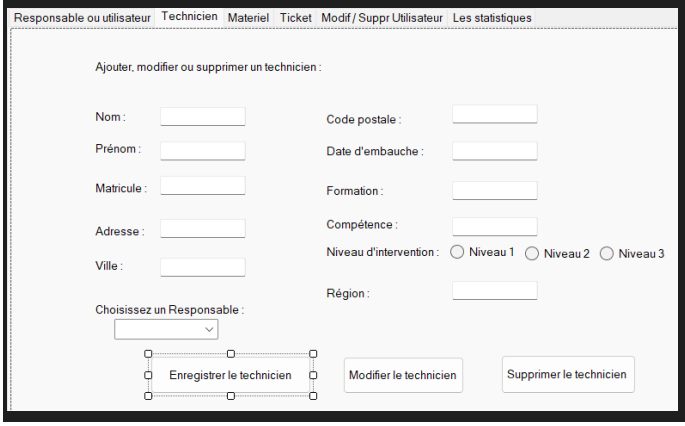

```
    // retourne le niveau en fonction du bouton radio
    private string GetNiveauTechnicien()
    {
        if (niveau_1.Checked) return "Niveau 1";
        if (niveau_2.Checked) return "Niveau 2";
        if (niveau_3.Checked) return "Niveau 3";
```

```
            return null;
        }

        private bool ChampsTechnicienValides()
        {
            if (string.IsNullOrWhiteSpace(nomT.Text) ||
                string.IsNullOrWhiteSpace(prenomT.Text) ||
                string.IsNullOrWhiteSpace(matriculeT.Text))
            {
                MessageBox.Show("Nom, prénom et matricule sont
obligatoires.");
                return false;
            }

            string niv = GetNiveauTechnicien();
            if (niv == null)
            {
                MessageBox.Show("Veuillez choisir un niveau
d'intervention (Niveau 1 / 2 / 3).");
                return false;
            }

            return true;
        }
        // Pour l'ajout : pas de baseTech, on construit uniquement
depuis les champs
        private Technicien ConstruireTechnicienDepuisChamps(int idT)
        {
            // on appelle la version complète en passant null
            return ConstruireTechnicienDepuisChamps(idT, null);
        }

        // Construit un objet Technicien à partir des champs Interface
Utilisateur
        // idT est passé en paramètre (car pour ajouter/modifier on n'a
pas le même)
        private Technicien ConstruireTechnicienDepuisChamps(int idT,
Technicien baseTech)
        {
            string niv = GetNiveauTechnicien();
            if (niv == null && baseTech != null)
                niv = baseTech.getNiv_intervention();
```

```
            DateTime dateEmbauche;
            if (!DateTime.TryParse(date_embaucheT.Text, out
dateEmbauche))
            {
                if (baseTech != null)
                    dateEmbauche = baseTech.getDate_embaucheT();
                else
                    throw new Exception("Date d'embauche invalide");
            }

            string nomFinal = string.IsNullOrWhiteSpace(nomT.Text)
                ? (baseTech != null ? baseTech.getNomt() : "")
                : nomT.Text.Trim();

            string prenomFinal =
string.IsNullOrWhiteSpace(prenomT.Text)
                ? (baseTech != null ? baseTech.getPrenomT() : "")
                : prenomT.Text.Trim();

            string formationFinal =
string.IsNullOrWhiteSpace(formationT.Text)
                ? (baseTech != null ? baseTech.getFormation() : "")
                : formationT.Text.Trim();

            string competenceFinal =
string.IsNullOrWhiteSpace(competences.Text)
                ? (baseTech != null ? baseTech.getCompetences() : "")
                : competences.Text.Trim();

            string matriculeFinal =
string.IsNullOrWhiteSpace(matriculeT.Text)
                ? (baseTech != null ? baseTech.getMatriculeT() : "")
                : matriculeT.Text.Trim();

            string adresseFinal =
string.IsNullOrWhiteSpace(adresseT.Text)
                ? (baseTech != null ? baseTech.getAdresseT() : "")
                : adresseT.Text.Trim();

            string villeFinal = string.IsNullOrWhiteSpace(villeT.Text)
                ? (baseTech != null ? baseTech.getVilleT() : "")
                : villeT.Text.Trim();
```

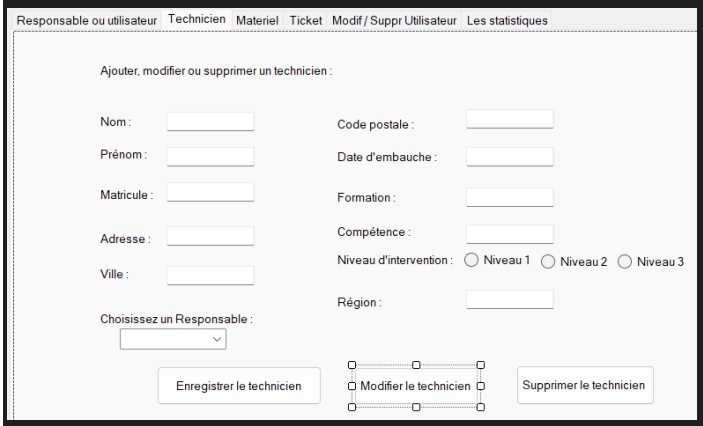
```
            string cpFinal = string.IsNullOrWhiteSpace(cpT.Text)
                ? (baseTech != null ? baseTech.getCodePT() : "")
                : cpT.Text.Trim();

            string regionFinal =
string.IsNullOrWhiteSpace(regionT.Text)
                ? (baseTech != null ? baseTech.getRegionT() : "")
                : regionT.Text.Trim();

            //Gestion du responsable
            int idR;

            if (choix_responsable != null &&
choix_responsable.SelectedItem != null)
            {
                // responsable choisi par l'utilisateur
                idR = Convert.ToInt32(choix_responsable.SelectedItem);
            }
            else if (baseTech != null)
            {
                // modification sans changer le responsable
                idR = baseTech.getIdr();
            }
            else
            {
                // ajout sans sélection --> un responsable par défaut
                idR = 1;
            }


            return new Technicien(
                idT,
                nomFinal,
                prenomFinal,
                formationFinal,
                competenceFinal,
                niv,
                matriculeFinal,
                adresseFinal,
                villeFinal,
                cpFinal,
                dateEmbauche,
                regionFinal,
```

```
                idR
            );
        }




        //Enregistrer un technicien
        private void enregistrer_technicien_Click(object sender,
EventArgs e)
        {
            if (!ChampsTechnicienValides()) return;

            try
            {
                int newIdT = Base_de_donnee.GetNextIdTechnicien();
                Technicien t =
ConstruireTechnicienDepuisChamps(newIdT);

                Base_de_donnee.AjouterTechnicien(t);
                MessageBox.Show("Technicien enregistré.");
            }

            catch (MySql.Data.MySqlClient.MySqlException ex)
            {
                MessageBox.Show($"Erreur MySQL #{ex.Number} :
{ex.Message}");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur : " + ex.Message);
            }
        }
```



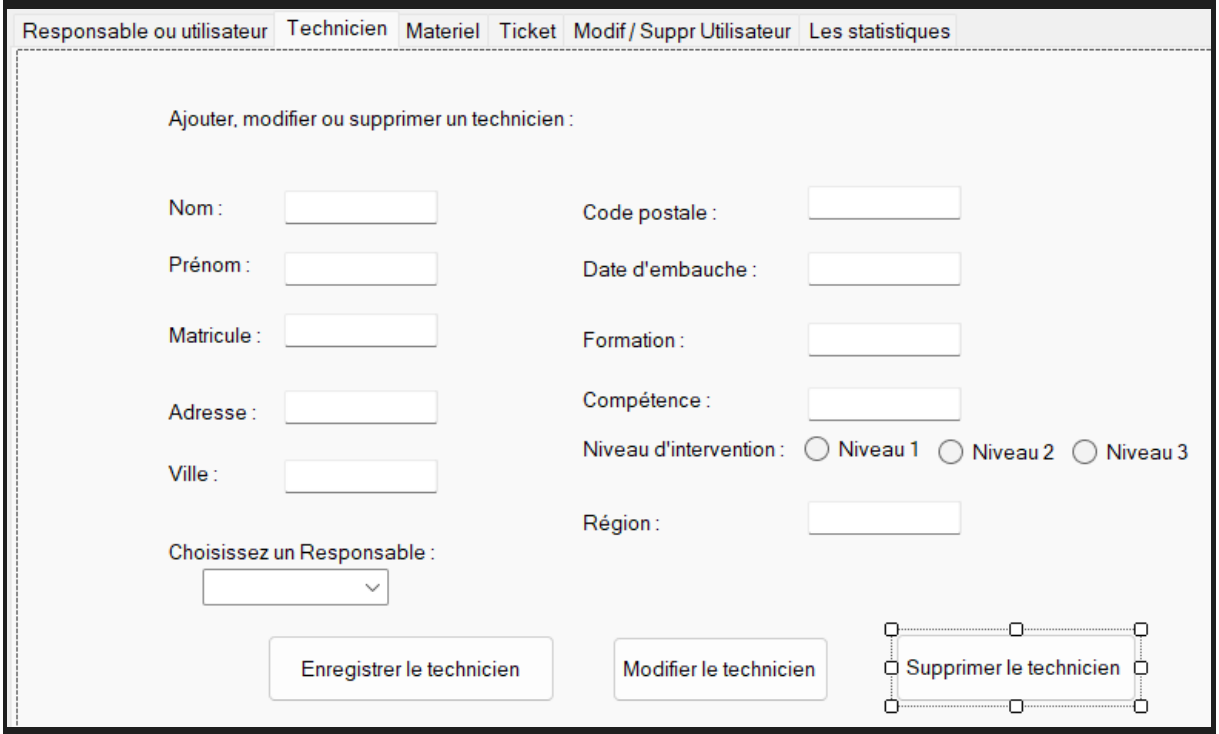
```
        //Modifier un technicien
        private void Modifier_technicien_Click(object sender, EventArgs
e)
        {
            if (!ChampsTechnicienValides()) return;

            try
            {
                int? idT = null;

                if (technicienAModifier != null)
                    idT = technicienAModifier.getIDTech();
                else
                    idT =
Base_de_donnee.GetIdTechnicienParMatricule(matriculeT.Text);

                if (idT == null)
                {
                    MessageBox.Show("Impossible de modifier :
technicien introuvable.");
                    return;
                }

                // TOUJOURS récupérer une base complète
                Technicien baseTech =
                    technicienAModifier ??
                    Base_de_donnee.GetTechnicienParId(idT.Value);

                if (baseTech == null)
                {
                    MessageBox.Show("Impossible de charger les données
actuelles du technicien.");
                    return;
                }

                // Construire le technicien en combinant les anciens
champs avec les nouveaux
                Technicien t =
ConstruireTechnicienDepuisChamps(idT.Value, baseTech);

                Base_de_donnee.ModifierTechnicien(t);
                MessageBox.Show("Technicien modifié.");
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show("Erreur modification :\n" +
ex.Message);
            }
        }
```




```
        //Supprimer un technicien
        private void Supprimer_technicien_Click(object sender,
EventArgs e)
        {
            try
            {
                // On supprime soit celui chargé (technicienAModifier),
soit via matricule
                int? idT = null;

                if (technicienAModifier != null)
                    idT = technicienAModifier.getIDTech();
                else
                    idT =
Base_de_donnee.GetIdTechnicienParMatricule(matriculeT.Text);

                if (idT == null)
                {
                    MessageBox.Show("Impossible de supprimer :
technicien introuvable (vérifie le matricule).");
                    return;
                }

                //message de confirmation pour ne pas supprimer sans
faire exprès
                var confirm = MessageBox.Show(
```

```
                    "Voulez-vous vraiment supprimer ce technicien ?",
                    "Confirmation",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Warning
                );

                if (confirm != DialogResult.Yes) return;

                Base_de_donnee.SupprimerTechnicien(idT.Value);
                MessageBox.Show("Technicien supprimé.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur (suppression technicien) : " +
ex.Message);
            }
        }
```



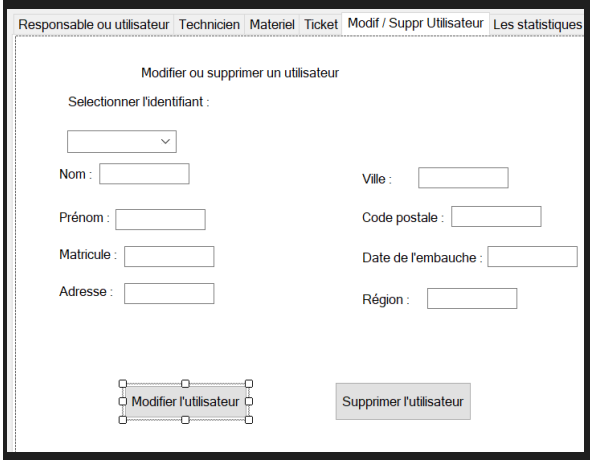

```
//Modifier ou supprimer un utilisateur



    // UTILISATEUR : retrouver l'id via matricule
```

```
        public static int? GetIdUtilisateurParMatricule(string
matricule)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idU FROM utilisateur
WHERE matriculeU = @matricule LIMIT 1;";
                    cmd.Parameters.AddWithValue("@matricule",
matricule);

                    object result = cmd.ExecuteScalar();
                    if (result == null || result == DBNull.Value)
return null;

                    return Convert.ToInt32(result);
                }
            }
        }


        private void comboBoxIdentifiantU_SelectedIndexChanged(object
sender, EventArgs e)
        {
            if (selectID.SelectedItem == null) return;

            int idU = Convert.ToInt32(selectID.SelectedItem);
            Utilisateur u = Base_de_donnee.GetUtilisateurParId(idU);

            if (u == null) return;

            utilisateurAModifier = u;

            // Remplace les noms des TextBox par de nouveaux
            nomS.Text = u.getNomu();
            prenomS.Text = u.getPrenomu();
            matriculeS.Text = u.getMatriculeu();
            adresseS.Text = u.getAdresseu();
            villeS.Text = u.getVilleu();
```

```
            cpS.Text = u.getCodepu();
            date_embaucheS.Text = u.getDate_embaucheu();
            regionS.Text = u.getRegionu();
        }

        private Utilisateur ConstruireUtilisateurDepuisChamps(int idU,
Utilisateur baseU)
        {
            // Remplace les noms des TextBox par les nouveaux
            string nomFinal = string.IsNullOrWhiteSpace(nomS.Text) ?
baseU.getNomu() : nomS.Text.Trim();
            string prenomFinal =
string.IsNullOrWhiteSpace(prenomS.Text) ? baseU.getPrenomu() :
prenomS.Text.Trim();
            string matriculeFinal =
string.IsNullOrWhiteSpace(matriculeS.Text) ? baseU.getMatriculeu() :
matriculeS.Text.Trim();
            string adresseFinal =
string.IsNullOrWhiteSpace(adresseS.Text) ? baseU.getAdresseu() :
adresseS.Text.Trim();
            string villeFinal = string.IsNullOrWhiteSpace(villeS.Text)
? baseU.getVilleu() : villeS.Text.Trim();
            string cpFinal = string.IsNullOrWhiteSpace(cpS.Text) ?
baseU.getCodepu() : cpS.Text.Trim();
            string dateFinal =
string.IsNullOrWhiteSpace(date_embaucheS.Text) ?
baseU.getDate_embaucheu() : date_embaucheS.Text.Trim();
            string regionFinal =
string.IsNullOrWhiteSpace(regionS.Text) ? baseU.getRegionu() :
regionS.Text.Trim();

            int idR = baseU.getIdr(); // on garde le responsable actuel

            return new Utilisateur(
                idU,
                nomFinal,
                prenomFinal,
                matriculeFinal,
                adresseFinal,
                villeFinal,
                cpFinal,
                dateFinal,
                regionFinal,
```

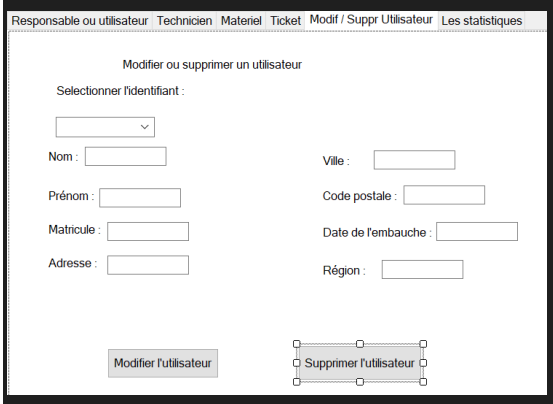
```
                idR
            );
        }

        //Modifier un utilisateur

        private void Modifier_utilisateur_Click(object sender,
EventArgs e)
        {
            try
            {
                int? idU = null;

                // si on a sélectionné dans la combo
                if (selectID.SelectedItem != null)
                    idU = Convert.ToInt32(selectID.SelectedItem);

                // sinon, on tente via matricule (si tu veux permettre
ça)
                if (idU == null &&
!string.IsNullOrWhiteSpace(matriculeS.Text))
                    idU =
Base_de_donnee.GetIdUtilisateurParMatricule(matriculeS.Text.Trim());

                if (idU == null)
                {
                    MessageBox.Show("Impossible de modifier :
utilisateur introuvable.");
                    return;
                }

                // Toujours charger une base complète
                Utilisateur baseU = utilisateurAModifier ??
Base_de_donnee.GetUtilisateurParId(idU.Value);
                if (baseU == null)
                {
                    MessageBox.Show("Impossible de charger les données
actuelles de l'utilisateur.");
                    return;
                }

                // crée la fusion entre les nouvelles données et celle
qui n'ont pas été changer
```

```
                Utilisateur u =
ConstruireUtilisateurDepuisChamps(idU.Value, baseU);

                Base_de_donnee.ModifierUtilisateur(u);
                MessageBox.Show("Utilisateur modifié.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur modification utilisateur :\n" +
ex.Message);
            }
        }
```




```
        //Supprimer un utilisateur
        private void Supprimer_utilisateur_Click(object sender,
EventArgs e)
        {
            try
            {
                int? idU = null;

                if (selectID.SelectedItem != null)
                    idU = Convert.ToInt32(selectID.SelectedItem);

                if (idU == null &&
!string.IsNullOrWhiteSpace(matriculeS.Text))
                    idU =
Base_de_donnee.GetIdUtilisateurParMatricule(matriculeS.Text.Trim());

                if (idU == null)
                {
                    MessageBox.Show("Impossible de supprimer :
utilisateur introuvable.");
```

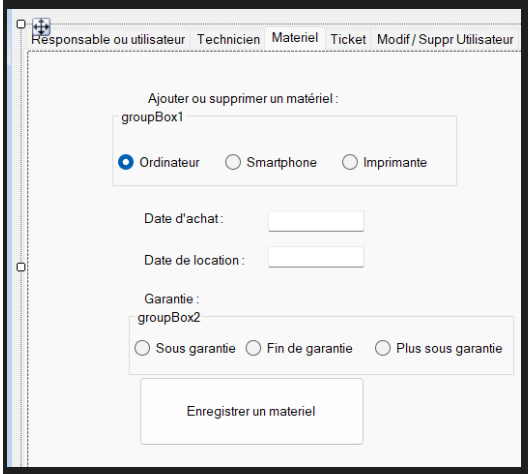
```
                    return;
                }

                var confirm = MessageBox.Show(
                    "Voulez-vous vraiment supprimer cet utilisateur ?",
                    "Confirmation",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Warning
                );

                if (confirm != DialogResult.Yes) return;

                Base_de_donnee.SupprimerUtilisateur(idU.Value);
                MessageBox.Show("Utilisateur supprimé.");

                // rafraichie la combo
                selectID.DataSource =
Base_de_donnee.GetTousLesIdsUtilisateurs();

                // vide les champs
                nomS.Clear(); prenomS.Clear(); matriculeS.Clear();
adresseS.Clear();
                villeS.Clear(); cpS.Clear(); date_embaucheS.Clear();
regionS.Clear();

                utilisateurAModifier = null;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur suppression utilisateur :\n" +
ex.Message);
            }
        }
```



```
        //retourne le matériel selectionner
        private string GetTypeMateriel()
        {
            if (ordinateur.Checked) return "Ordinateur";
            if (smartphone.Checked) return "Smartphone";
            if (imprimante.Checked) return "Imprimante";
            return null;
        }

        //Pareil mais pour les ganrantie
        private string GetGarantie()
        {
            if (sous_garantie.Checked) return "Sous garantie";
            if (fin_garantie.Checked) return "Fin de garantie";
            if (plus_sous_garantie.Checked) return "Plus sous
garantie";
            return null;
        }

        //vérifier que les champs du formulaire sont corrects
        private bool ChampsMaterielValides()
        {
            if (GetTypeMateriel() == null)
            {
                MessageBox.Show("Veuillez choisir un type de
matériel.");
                return false;
            }

            if (GetGarantie() == null)
            {
                MessageBox.Show("Veuillez choisir un état de
garantie.");
                return false;
            }

            // Dates : autorisées vides, mais si remplies elles doivent
être valides
            if (!string.IsNullOrWhiteSpace(date_achat.Text) &&
!DateTime.TryParse(date_achat.Text, out _))
            {
                MessageBox.Show("Date d'achat invalide.");
                return false;
```

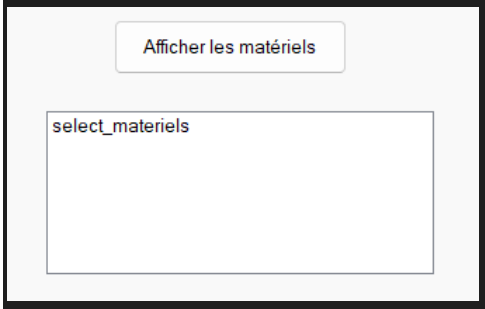
```
            }

            if (!string.IsNullOrWhiteSpace(date_location.Text) &&
!DateTime.TryParse(date_location.Text, out _))
            {
                MessageBox.Show("Date de location invalide.");
                return false;
            }

            return true;
        }

        //bouton enregister un matériel
        private void btnEnregistrerMateriel_Click(object sender,
EventArgs e)
        {
            if (!ChampsMaterielValides()) return;

            try
            {
                string idM = Base_de_donnee.GetNextIdMateriel(); //
méthode ajoutée
                string type = GetTypeMateriel();
                string garantie = GetGarantie();

                DateTime? dateAchat = null;
                if (!string.IsNullOrWhiteSpace(date_achat.Text))
                    dateAchat = DateTime.Parse(date_achat.Text);

                DateTime? dateLocation = null;
                if (!string.IsNullOrWhiteSpace(date_location.Text))
                    dateLocation = DateTime.Parse(date_location.Text);

                //si l'utilisateurs existe et qu'il est bien
selectionnée
                int idU = 1;
                if (selectID != null && selectID.SelectedItem != null)
                    idU = Convert.ToInt32(selectID.SelectedItem);

                Materiel m = new Materiel(idM, type, dateAchat,
dateLocation, garantie, idU);

                Base_de_donnee.AjouterMateriel(m);
```

```
                MessageBox.Show("Matériel enregistré !");


                MessageBox.Show("Matériel enregistré !");
            }

            catch (MySql.Data.MySqlClient.MySqlException ex)
            {
                MessageBox.Show($"Erreur MySQL #{ex.Number} :
{ex.Message}");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur : " + ex.Message);
            }
        }
```




```
        //Consulter un matériel

        private void btnAfficherMateriels_Click(object sender,
EventArgs e)
        {
            try
            {
                select_materiels.Items.Clear();

                List<Materiel> materiels =
Base_de_donnee.GetTousLesMateriels();

                foreach (Materiel m in materiels)
                {
```

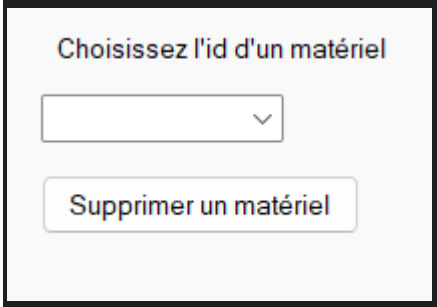
```
                    select_materiels.Items.Add(m);
                }

                if (materiels.Count == 0)
                {
                    MessageBox.Show("Aucun matériel enregistré.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur lors du chargement des
matériels : " + ex.Message);
            }
        }
```




```
        //afficher le détail des matériels lors d'un clique sur un
matériel
        private void listBoxMateriels_SelectedIndexChanged(object
sender, EventArgs e)
        {
            if (select_materiels.SelectedItem == null) return;

            Materiel m = (Materiel)select_materiels.SelectedItem;

            MessageBox.Show(
                $"ID : {m.getIDM()}\n" +
                $"Type : {m.getType_materiel()}\n" +
                $"Garantie : {m.getGaranti()}\n" +
                $"Date achat : {m.getDate_achat()}\n" +
                $"Date location : {m.getDate_location()}",
                "Détail du matériel"
            );
        }

        //Ajouter une méthode pour remplir la comboBox avec les id des
matériels
```

```
        private void ChargerIdsMaterielsDansCombo()
        {
            var ids = Base_de_donnee.GetTousLesIdsMateriels();

            id_materiel.DataSource = null;
            id_materiel.Items.Clear();

            foreach (var id in ids)
                id_materiel.Items.Add(id);

            id_materiel.SelectedIndex = (id_materiel.Items.Count > 0) ?
0 : -1;
        }


        //Bouton supprimer le matériel
        private void btnSupprimerMateriel_Click(object sender,
EventArgs e)
        {
            if (id_materiel.SelectedItem == null)
            {
                MessageBox.Show("Veuillez choisir un id matériel.");
                return;
            }

            string idM = id_materiel.SelectedItem.ToString();

            var confirm = MessageBox.Show(
                $"Voulez-vous supprimer le matériel {idM} ?",
                "Confirmation",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Warning
            );

            if (confirm != DialogResult.Yes) return;

            try
            {
                bool ok = Base_de_donnee.SupprimerMateriel(idM);

                if (ok)
                {
```

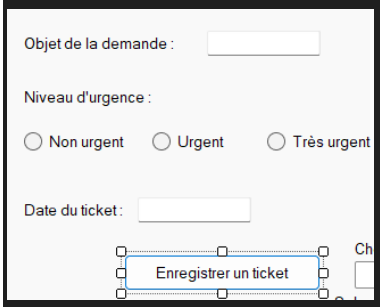
```
                    MessageBox.Show("Matériel supprimé.");

                    //  mettre a jour la ComboBox
                    ChargerIdsMaterielsDansCombo();
                }
                else
                {
                    MessageBox.Show("La suppression a échoué (id
inexistant ?).");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur suppression : " + ex.Message);
            }
        }
```



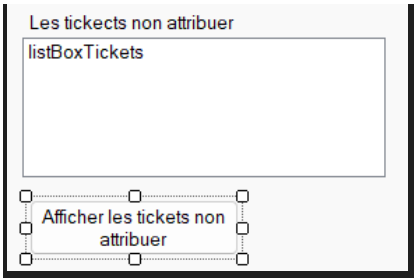

```
        //Enregistrer un ticket
        private string GetNiveauUrgenceTicket()
        {
            //retourne le radio checke
            if (Non_urgent.Checked) return "Non urgent";
            if (urgent.Checked) return "Urgent";
            if (tres_urgent.Checked) return "Très urgent";
            return null;
        }

        private bool ChampsTicketValides()
        {
            //sassure que les champs soit bien remplie
            if (string.IsNullOrWhiteSpace(objet_demande.Text))
            {
                MessageBox.Show("Veuillez saisir l'objet de la
demande.");
                return false;
            }
```

```
            string urg = GetNiveauUrgenceTicket();
            if (urg == null)
            {
                MessageBox.Show("Veuillez choisir un niveau
d'urgence.");
                return false;
            }

            DateTime dt;
            if (!DateTime.TryParse(date_ticket.Text, out dt))
            {
                MessageBox.Show("Date du ticket invalide.");
                return false;
            }

            return true;
        }

        //Le bouton enregistrer Tickett
        private void enregistrer_ticket_Click(object sender, EventArgs
e)
        {
            if (!ChampsTicketValides()) return;

            try
            {
                int idTicket = Base_de_donnee.GetNextIdTicket();

                string objet = objet_demande.Text.Trim();
                string nivUrgence = GetNiveauUrgenceTicket();

                DateTime dt = DateTime.Parse(date_ticket.Text);
                string dateSql = dt.ToString("yyyy-MM-dd");

                int idU = utilisateurAModifier != null ?
utilisateurAModifier.getIDU() : 1;
                int idM = 1;

                // Etat initial + phase initiale
                string etat = "Déclarée";
                int idPhase = 1;
```

```
                Ticket t = new Ticket(idTicket, objet, nivUrgence,
etat, dateSql, idM, null, idU, idPhase);

                Base_de_donnee.AjouterTicket(t);

                MessageBox.Show("Ticket enregistré.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur (ticket) : " + ex.Message);
            }
        }
```



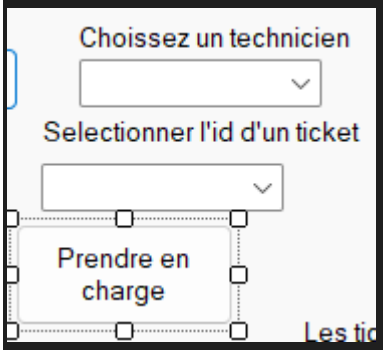

```
        //Afficher les ticket non attribuée
        private void btnAfficherTickets_Click(object sender, EventArgs
e)
        {
            try
            {
                var tickets = Base_de_donnee.GetTicketsNonAttribues();

                listBoxTickets.Items.Clear();

                foreach (var t in tickets)
                    listBoxTickets.Items.Add(t);

                if (tickets.Count == 0)
                    MessageBox.Show("Aucun ticket non attribué.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur (affichage tickets) : " +
ex.Message);
            }
        }
```




```
        //Prise en charge d'un ticket
        private void ChargerTicketsNonAttribuesDansCombo()
        {
            var tickets = Base_de_donnee.GetTicketsNonAttribues();

            id_ticket.DataSource = null;
            id_ticket.Items.Clear();

            foreach (var t in tickets)
                id_ticket.Items.Add(t);

            id_ticket.SelectedIndex = (id_ticket.Items.Count > 0) ? 0 :
-1;
        }

       //Charger la liste des techniciens
        private void ChargerTechniciensDansCombo()
        {
            var techs = Base_de_donnee.GetTousLesTechniciens();

            comboTechnicien.DataSource = null;
            comboTechnicien.Items.Clear();

            foreach (var t in techs)
                comboTechnicien.Items.Add(t);

            comboTechnicien.SelectedIndex =
(comboTechnicien.Items.Count > 0) ? 0 : -1;
        }

        //Prendre en charge un ticket par un technicien
        private void btnPrendreEnCharge_Click(object sender, EventArgs
e)
        {
            try
            {
```

```
                // Ticket qui doit être sélectionné
                if (id_ticket.SelectedItem == null)
                {
                    MessageBox.Show("Veuillez sélectionner un
ticket.");
                    return;
                }

                // Technicien qui doit être sélectionné
                if (comboTechnicien.SelectedItem == null)
                {
                    MessageBox.Show("Veuillez sélectionner un
technicien.");
                    return;
                }

                // Récupérer les objets depuis les ComboBox
                Ticket ticket = (Ticket)id_ticket.SelectedItem;
                Technicien tech =
(Technicien)comboTechnicien.SelectedItem;

                int idTicket = ticket.getIDTicket();
                int idTech = tech.getIDTech();

                // Appel DB
                bool ok =
Base_de_donnee.PrendreEnChargeIncident(idTicket, idTech);

                if (ok)
                {
                    MessageBox.Show("Ticket pris en charge !");
                    MessageBox.Show("Etat enregistré en BD : " +
Base_de_donnee.DebugEtatTicket(idTicket));

                    ChargerTicketsNonAttribuesDansCombo(); // mis a
jour des tickets restants
                }
                else
                {
                    MessageBox.Show("Impossible : ticket déjà
attribué.");
                }
            }
```

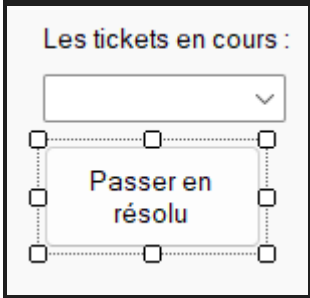
```
            catch (Exception ex)
            {
                MessageBox.Show("Erreur (prise en charge) : " +
ex.Message);
            }
        }
```




```
        //Charger les ticket en cours de traitement dans la combobox
        private void ChargerTicketsEnCoursDansCombo()
        {
            if (comboTechnicien.SelectedItem == null)
            {
                id_ticket.DataSource = null;
                id_ticket.Items.Clear();
                id_ticket.SelectedIndex = -1;
                return;
            }

            Technicien tech = (Technicien)comboTechnicien.SelectedItem;

            var tickets =
Base_de_donnee.GetTicketsEnCoursPourTechnicien(tech.getIDTech());
            MessageBox.Show($"Tickets en cours trouvés: {tickets.Count}
pour le technicien {tech.getIDTech()}");


            id_ticket.DataSource = null;
            id_ticket.Items.Clear();

            foreach (var t in tickets)
                id_ticket.Items.Add(t);

            id_ticket.SelectedIndex = (id_ticket.Items.Count > 0) ? 0 :
-1;
        }
```

```
        //Clique bouton résolut avec seul les tickets en cours de
traitement
        private void btnPasserEnResolu_Click(object sender, EventArgs
e)
        {
            try
            {
                if (comboTechnicien.SelectedItem == null)
                {
                    MessageBox.Show("Veuillez sélectionner un
technicien.");
                    return;
                }

                if (ticket_en_cours.SelectedItem == null)
                {
                    MessageBox.Show("Veuillez sélectionner un ticket en
cours.");
                    return;
                }

                Technicien tech =
(Technicien)comboTechnicien.SelectedItem;
                Ticket ticket = (Ticket)ticket_en_cours.SelectedItem;

                bool ok =
Base_de_donnee.PasserTicketEnResolu(ticket.getIDTicket(),
tech.getIDTech());

                if (ok)
                {
                    MessageBox.Show("Ticket passé en résolu !");
                    ChargerTicketsEnCoursDansComboResolu(); //
rafraîchir la liste
                }
                else
                {
                    MessageBox.Show("Impossible : le ticket n'est pas
en cours ou n'est pas affecté à ce technicien.");
                }
            }
            catch (Exception ex)
```

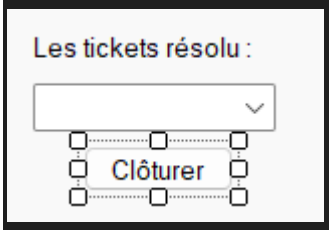
```
            {
                MessageBox.Show("Erreur : " + ex.Message);
            }
        }
```




```
        private void comboTechnicien_SelectedIndexChanged(object
sender, EventArgs e)
        {
            ChargerTicketsEnCoursDansComboResolu();
        }



        //charge la list box pour pouvoir selectionner les tieckets en
cours de traitement
        private void btnFiltrerTicketsEnCours_Click(object sender,
EventArgs e)
        {
            ChargerTicketsEnCoursDansCombo();
        }

        //Prendre la ligne selectionner dans la listbox pour la mettre
dans la combo pour le bouton prise en charge
        private void listBoxTickets_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (listBoxTickets.SelectedItem == null) return;

            Ticket t = (Ticket)listBoxTickets.SelectedItem;

            // Synchronisation avec la ComboBox utilisée par le bouton
            id_ticket.SelectedItem = t;
        }

        //remplir la combo avec les ticket en cours de traitement
        private void ChargerTicketsEnCoursDansComboResolu()
```

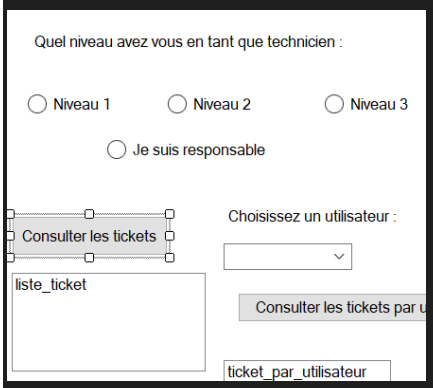
```
        {
            if (comboTechnicien.SelectedItem == null)
            {
                ticket_en_cours.DataSource = null;
                ticket_en_cours.Items.Clear();
                ticket_en_cours.SelectedIndex = -1;
                return;
            }

            Technicien tech = (Technicien)comboTechnicien.SelectedItem;

            var tickets =
Base_de_donnee.GetTicketsEnCoursPourTechnicien(tech.getIDTech());

            MessageBox.Show($"Tickets en cours trouvés :
{tickets.Count}");

            ticket_en_cours.DataSource = null;
            ticket_en_cours.Items.Clear();

            foreach (var t in tickets)
                ticket_en_cours.Items.Add(t);

            ticket_en_cours.SelectedIndex =
(ticket_en_cours.Items.Count > 0) ? 0 : -1;
        }


        //Remplie la combo box seulement des tickets résolu
        private void ChargerTicketsResolusDansCombo()
        {
            var tickets = Base_de_donnee.GetTicketsResolus();

            ticket_resolu.DataSource = null;
            ticket_resolu.Items.Clear();

            foreach (var t in tickets)
                ticket_resolu.Items.Add(t);

            ticket_resolu.SelectedIndex =
                ticket_resolu.Items.Count > 0 ? 0 : -1;
        }
```

```
        //Le bouton clôturer
        private void btnCloturer_Click(object sender, EventArgs e)
        {
            if (ticket_resolu.SelectedItem == null)
            {
                MessageBox.Show("Veuillez sélectionner un ticket
résolu.");
                return;
            }

            Ticket t = (Ticket)ticket_resolu.SelectedItem;

            bool ok = Base_de_donnee.CloturerTicket(t.getIDTicket());

            if (ok)
            {
                MessageBox.Show("Ticket clôturé !");
                ChargerTicketsResolusDansCombo();
                AfficherStatistiques();
            }
            else
            {
                MessageBox.Show("Impossible : le ticket n'est pas
résolu.");
            }
        }
```

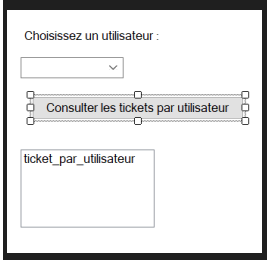



```
        //le bouton consulter les ticket
        private void btnConsulterTickets_Click(object sender, EventArgs
e)
        {
            liste_ticket.Items.Clear();

            bool estResponsable = etre_responsable.Checked;
            string niveau = GetNiveauTechnicien();

            var tickets =
Base_de_donnee.GetTicketsVisiblesParNiveau(niveau, estResponsable);
```

```
            foreach (var t in tickets)
                liste_ticket.Items.Add(t);

            if (tickets.Count == 0)
                MessageBox.Show("Aucun ticket à afficher.");
        }
```




```
        //Consulter les tickets d'un utilisateur selon celui qui est selectionner

        private void btnConsulterTicketsParUtilisateur_Click(object sender, EventArgs e)
        {
            if (id_utilisateur.SelectedItem == null)
            {
                MessageBox.Show("Veuillez sélectionner un utilisateur.");
                return;
            }

            int idU = Convert.ToInt32(id_utilisateur.SelectedItem);
            var tickets = Base_de_donnee.GetIncidentsPourUtilisateur(idU);
```

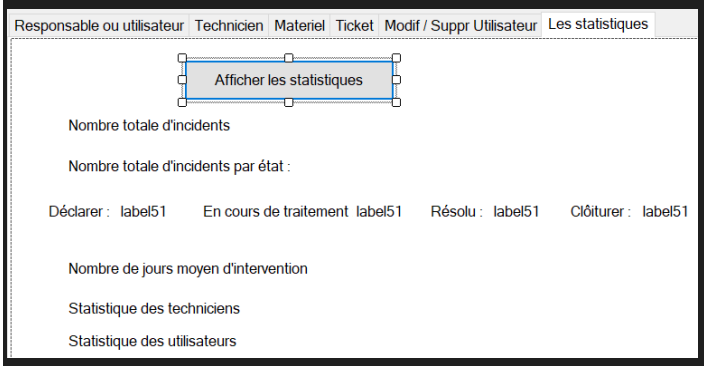



```
            // Afficher dans la LISTBOX
            ticket_par_utilisateur.Items.Clear();
            foreach (var t in tickets)
```

```
                ticket_par_utilisateur.Items.Add(t);

            if (tickets.Count == 0)
                MessageBox.Show("Aucun ticket pour cet utilisateur.");
        }

        private void ChargerUtilisateursDansComboTickets()
        {
            var ids = Base_de_donnee.GetTousLesIdsUtilisateurs();

            id_utilisateur.DataSource = null;
            id_utilisateur.DataSource = ids;
            id_utilisateur.SelectedIndex = -1;
        }




        //
        //
        //Les statistiques
        //
        //
        private void AfficherStatistiques()
        {
            try
            {
                // ombre total d'incidents
                int total = Base_de_donnee.GetNombreTotalIncidents();
                nb_incident.Text = total.ToString();


                // Nombre d'incidents par état
                int nbDeclaree =
Base_de_donnee.GetNombreIncidentsParEtat("Déclarée");
                int nbEnCours =
Base_de_donnee.GetNombreIncidentsParEtat("En cours de traitement");
                int nbResolue =
Base_de_donnee.GetNombreIncidentsParEtat("Résolue");
                int nbCloturee =
Base_de_donnee.GetNombreIncidentsParEtat("Clôturée");

                lblDeclaree.Text = nbDeclaree.ToString();
```

```
                lblEnCours.Text = nbEnCours.ToString();
                lblResolue.Text = nbResolue.ToString();
                lblCloturee.Text = nbCloturee.ToString();


                // Durée moyenne d'intervention
                double moyMinutes =
Base_de_donnee.GetDureeMoyenneInterventionEnMinutes();
                double moyJours = moyMinutes / 60.0 / 24.0;

                nb_moyen_intervention.Text = $"{moyJours:0.00}
jour(s)";



                // Statistiques des techniciens (Label multi-lignes)
                var statsTech = Base_de_donnee.GetStatsTechniciens();
                StringBuilder sbTech = new StringBuilder();

                foreach (var t in statsTech)
                {
                    sbTech.AppendLine(
                        $"Technicien #{t.getIdT()} - {t.getNomT()}
{t.getPrenomT()}\n" +
                        $"  Assignés : {t.getNbTotalAssignes()}\n" +
                        $"  En charge : {t.getNbEnCharge()}\n" +
                        $"  Résolus : {t.getNbResolus()}\n" +
                        $"  Clôturés : {t.getNbClotures()}\n"
                    );
                }

                statistique_techniciens.Text =
                    sbTech.Length == 0 ? "Aucun technicien." :
sbTech.ToString();


                // Statistiques des utilisateurs
                var statsUsers = Base_de_donnee.GetStatsUtilisateurs();
                StringBuilder sbUsers = new StringBuilder();

                foreach (var u in statsUsers)
                {
                    sbUsers.AppendLine(
```

```
                        $"Utilisateur #{u.getIDU()} - {u.getNomU()}
{u.getPrenomU()}\n" +
                        $"  Incidents déclarés :
{u.getNbIncidentsDeclares()}\n" +
                        $"  En cours : {u.getNbEnCours()}\n" +
                        $"  Résolus : {u.getNbResolus()}\n"
                    );
                }

                statistique_utilisateur.Text =
                    sbUsers.Length == 0 ? "Aucun utilisateur." :
sbUsers.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erreur affichage statistiques : " +
ex.Message);
            }
        }



        //Bouton pour afficher les statistiques
        private void btnAfficherStatistiques_Click(object sender,
EventArgs e)
        {
            AfficherStatistiques();
        }
```




```
    }
}
```

Class Base de Donnée :

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

namespace Projet_final_C_
{
    internal class Base_de_donnee
    //création de tous les inserts
    {
        //insert table technicien
        public static void AjouterTechnicien(Technicien UnTechnicien)
        {
```

```
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                    INSERT INTO technicien
                    (idT, niv_intervention, formation, competences,
nomT, prenomT, matriculeT,
                     adresseT, villeT, codePT, date_embaucheT, regionT,
idR)
                    VALUES
                    (@idT, @niv_intervention, @formation, @competences,
@nomT, @prenomT, @matriculeT,
                     @adresseT, @villeT, @codePT, @date_embaucheT,
@regionT, @idR);
                    ";
                    cmd.Parameters.AddWithValue("@idT",
UnTechnicien.getIDTech());
                    cmd.Parameters.AddWithValue("@niv_intervention",
UnTechnicien.getNiv_intervention());
                    cmd.Parameters.AddWithValue("@formation",
UnTechnicien.getFormation());
                    cmd.Parameters.AddWithValue("@competences",
UnTechnicien.getCompetences());
                    cmd.Parameters.AddWithValue("@nomT",
UnTechnicien.getNomt());
                    cmd.Parameters.AddWithValue("@prenomT",
UnTechnicien.getPrenomT());
                    cmd.Parameters.AddWithValue("@matriculeT",
UnTechnicien.getMatriculeT());
                    cmd.Parameters.AddWithValue("@adresseT",
UnTechnicien.getAdresseT());
                    cmd.Parameters.AddWithValue("@villeT",
UnTechnicien.getVilleT());
                    cmd.Parameters.AddWithValue("@codePT",
UnTechnicien.getCodePT());
                    cmd.Parameters.Add("@date_embaucheT",
MySqlDbType.Date).Value = UnTechnicien.getDate_embaucheT().Date;
```

```
                    cmd.Parameters.AddWithValue("@regionT",
UnTechnicien.getRegionT());
                    cmd.Parameters.AddWithValue("@idR",
UnTechnicien.getIdr());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTechnicien.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table responsable
        public static void AjouterResponsable(Responsable
UnResponsable)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "INSERT INTO responsable (idR,
nomR, prenomR, matriculeR, adresseR, villeR, codePR, date_embaucheR,
regionR) VALUES (@idR, @nomR, @prenomR, @matriculeR, @adresseR,
@villeR, @codePR, @date_embaucheR, @regionR)";
                    cmd.Parameters.AddWithValue("@idR",
UnResponsable.getIDR());
                    cmd.Parameters.AddWithValue("@nomR",
UnResponsable.getNomr());
                    cmd.Parameters.AddWithValue("@prenomR",
UnResponsable.getPrenomr());
                    cmd.Parameters.AddWithValue("@matriculeR",
UnResponsable.getMatriculer());
                    cmd.Parameters.AddWithValue("@adresseR",
UnResponsable.getAdresser());
                    cmd.Parameters.AddWithValue("@villeR",
UnResponsable.getViller());
                    cmd.Parameters.AddWithValue("@codePR",
UnResponsable.getCodepr());
```

```
                    cmd.Parameters.AddWithValue("@date_embaucheR",
UnResponsable.getDate_embaucher());
                    cmd.Parameters.AddWithValue("@regionR",
UnResponsable.getRegionr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table utilisateur
        public static void AjouterUtilisateur(Utilisateur
UnUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "INSERT INTO utilisateur (idU,
nomU, prenomU, matriculeU, adresseU, villeU, codePU, date_embaucheU,
regionU, idR) VALUES (@idU, @nomU, @prenomU, @matriculeU, @adresseU,
@villeU, @codePU, @date_embaucheU, @regionU, @idR)";
                    cmd.Parameters.AddWithValue("@idU",
UnUtilisateur.getIDU());
                    cmd.Parameters.AddWithValue("@nomU",
UnUtilisateur.getNomu());
                    cmd.Parameters.AddWithValue("@prenomU",
UnUtilisateur.getPrenomu());
                    cmd.Parameters.AddWithValue("@matriculeU",
UnUtilisateur.getMatriculeu());
                    cmd.Parameters.AddWithValue("@adresseU",
UnUtilisateur.getAdresseu());
                    cmd.Parameters.AddWithValue("@villeU",
UnUtilisateur.getVilleu());
                    cmd.Parameters.AddWithValue("@codePU",
UnUtilisateur.getCodepu());
                    cmd.Parameters.AddWithValue("@date_embaucheU",
UnUtilisateur.getDate_embaucheu());
```

```
                    cmd.Parameters.AddWithValue("@regionU",
UnUtilisateur.getRegionu());
                    cmd.Parameters.AddWithValue("@idR",
UnUtilisateur.getIdr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table matériels
        public static void AjouterMateriel(Materiel UnMateriel)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "Insert Into materiel (idM,
type_materiel, date_Achat, date_location, garanti, idU) Values (@idM,
@type_materiel, @date_Achat, @date_location, @garanti, @idU)";
                    cmd.Parameters.AddWithValue("@idM",
UnMateriel.getIDM());
                    cmd.Parameters.AddWithValue("@type_materiel",
UnMateriel.getType_materiel());
                    //Gestion de la nullité des dates achats et
locations
                    if (UnMateriel.getDate_achat() == null)
                        cmd.Parameters.AddWithValue("@date_Achat",
DBNull.Value);
                    else
                        cmd.Parameters.AddWithValue("@date_Achat",
UnMateriel.getDate_achat());

                    if (UnMateriel.getDate_location() == null)
                        cmd.Parameters.AddWithValue("@date_location",
DBNull.Value);
                    else
                        cmd.Parameters.AddWithValue("@date_location",
UnMateriel.getDate_location());
```

```
                    //
                    cmd.Parameters.AddWithValue("@garanti",
UnMateriel.getGaranti());
                    cmd.Parameters.AddWithValue("@idU",
UnMateriel.getIDU());

                    cmd.ExecuteNonQuery();
                }
            }

        }
        //insert table ticket
        public static void AjouterTicket(Ticket UnTicket)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "INSERT INTO ticket (id_ticket,
objet_demande, niv_urgence, etat_demande, date_ticket, idM, idT, idU)
VALUES (@id_ticket, @objet_demande, @niv_urgence, @etat_demande,
@date_ticket, @idM, @idT, @idU)";
                    cmd.Parameters.AddWithValue("@id_ticket",
UnTicket.getIDTicket());
                    cmd.Parameters.AddWithValue("@objet_demande",
UnTicket.getObjet_demande());
                    cmd.Parameters.AddWithValue("@niv_urgence",
UnTicket.getNiv_urgence());
                    cmd.Parameters.AddWithValue("@etat_demande",
UnTicket.getEtat_demande());
                    cmd.Parameters.AddWithValue("@date_ticket",
UnTicket.getDate_ticket());
                    cmd.Parameters.AddWithValue("@idM",
UnTicket.getIDM());
                    if (UnTicket.getIDTech() == null)
                        cmd.Parameters.AddWithValue("@idT",
DBNull.Value);
                    else
```

```
                        cmd.Parameters.AddWithValue("@idT",
UnTicket.getIDTech());
                    cmd.Parameters.AddWithValue("@idU",
UnTicket.getIDU());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTicket.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //insert des phases d'un ticket
        public static void AjouterPhaseTicket(Phase_ticket phase)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                INSERT INTO phase_ticket
                (id_ticket, idT, date_phase, heure_debut, heure_fin,
description_travail)
                VALUES
                (@idTicket, @idT, @datePhase, @heureDebut, @heureFin,
@description)
            ";

                    cmd.Parameters.AddWithValue("@idTicket",
phase.getIdTicket());
                    cmd.Parameters.AddWithValue("@idT",
phase.getIdTechnicien());
                    cmd.Parameters.AddWithValue("@datePhase",
phase.getDatePhase());
                    cmd.Parameters.AddWithValue("@heureDebut",
phase.getHeureDebut());
                    cmd.Parameters.AddWithValue("@heureFin",
phase.getHeureFin());
```

```
                    cmd.Parameters.AddWithValue("@description",
phase.getDescriptionTravail());

                    cmd.ExecuteNonQuery();
                }
            }
        }


        //Création des modifications

        //modification table technicien

        public static void ModifierTechnicien(Technicien UnTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "UPDATE technicien SET
niv_intervention = @niv_intervention, formation = @formation,
competences = @competences, nomT = @nomT, prenomT = @prenomT,
matriculeT = @matriculeT, adresseT = @adresseT, villeT = @villeT,
codePT = @codePT, date_embaucheT = @date_embaucheT, regionT = @regionT,
idR = @idR WHERE idT = @idT;";
                    cmd.Parameters.AddWithValue("@idT",
UnTechnicien.getIDTech());
                    cmd.Parameters.AddWithValue("@niv_intervention",
UnTechnicien.getNiv_intervention());
                    cmd.Parameters.AddWithValue("@formation",
UnTechnicien.getFormation());
                    cmd.Parameters.AddWithValue("@competences",
UnTechnicien.getCompetences());
                    cmd.Parameters.AddWithValue("@nomT",
UnTechnicien.getNomt());
                    cmd.Parameters.AddWithValue("@prenomT",
UnTechnicien.getPrenomT());
```

```
                    cmd.Parameters.AddWithValue("@matriculeT",
UnTechnicien.getMatriculeT());
                    cmd.Parameters.AddWithValue("@adresseT",
UnTechnicien.getAdresseT());
                    cmd.Parameters.AddWithValue("@villeT",
UnTechnicien.getVilleT());
                    cmd.Parameters.AddWithValue("@codePT",
UnTechnicien.getCodePT());
                    cmd.Parameters.AddWithValue("@date_embaucheT",
UnTechnicien.getDate_embaucheT());
                    cmd.Parameters.AddWithValue("@regionT",
UnTechnicien.getRegionT());
                    cmd.Parameters.AddWithValue("@idR",
UnTechnicien.getIdr());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTechnicien.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //Modification d'un utilisateur

        public static void ModifierUtilisateur(Utilisateur
UnUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "UPDATE utilisateur SET nomU =
@nomU, prenomU = @prenomU, matriculeU = @matriculeU, adresseU =
@adresseU, villeU = @villeU, codePU = @codePU, date_embaucheU =
@date_embaucheU, regionU = @regionU, idR = @idR WHERE idU = @idU;";
                    cmd.Parameters.AddWithValue("@idU",
UnUtilisateur.getIDU());
```

```
                    cmd.Parameters.AddWithValue("@nomU",
UnUtilisateur.getNomu());
                    cmd.Parameters.AddWithValue("@prenomU",
UnUtilisateur.getPrenomu());
                    cmd.Parameters.AddWithValue("@matriculeU",
UnUtilisateur.getMatriculeu());
                    cmd.Parameters.AddWithValue("@adresseU",
UnUtilisateur.getAdresseu());
                    cmd.Parameters.AddWithValue("@villeU",
UnUtilisateur.getVilleu());
                    cmd.Parameters.AddWithValue("@codePU",
UnUtilisateur.getCodepu());
                    cmd.Parameters.AddWithValue("@date_embaucheU",
UnUtilisateur.getDate_embaucheu());
                    cmd.Parameters.AddWithValue("@regionU",
UnUtilisateur.getRegionu());
                    cmd.Parameters.AddWithValue("@idR",
UnUtilisateur.getIdr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //Création des suppressions
        //Suppression d'un technicien
        public static void SupprimerTechnicien(int idTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM technicien WHERE idT
= @idT;";
                    cmd.Parameters.AddWithValue("@idT", idTechnicien);

                    cmd.ExecuteNonQuery();
                }
            }
```

```
        }

        //Suppression d'un utilisateur
        public static void SupprimerUtilisateur(int idUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM utilisateur WHERE
idU = @idU;";
                    cmd.Parameters.AddWithValue("@idU", idUtilisateur);

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //Suppression d'un matériel
        // Suppression d'un matériel (CORRIGÉ)
        public static bool SupprimerMateriel(string idM)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM materiel WHERE idM =
@idM;";
                    cmd.Parameters.AddWithValue("@idM", idM);

                    return cmd.ExecuteNonQuery() > 0;
                }
            }
```

```
        }


        //Consultation des matériels

        public static List<Materiel> GetTousLesMateriels()
        {
            List<Materiel> lesMateriels = new List<Materiel>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idM, type_materiel, date_Achat, date_location,
garanti, idU
                FROM materiel
                ORDER BY idM;";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string idM = reader.GetString("idM");
// string
                            string type =
reader.GetString("type_materiel");      // string

                            DateTime? dateAchat =
reader.IsDBNull(reader.GetOrdinal("date_Achat"))
                                ? (DateTime?)null
                                : reader.GetDateTime("date_Achat");
// DateTime?

                            DateTime? dateLoc =
reader.IsDBNull(reader.GetOrdinal("date_location"))
                                ? (DateTime?)null
```

```
                                : reader.GetDateTime("date_location");
// DateTime?

                            string garanti =
reader.GetString("garanti");          // string
                            int idU = reader.GetInt32("idU");
// int

                            Materiel m = new Materiel(idM, type,
dateAchat, dateLoc, garanti, idU);
                            lesMateriels.Add(m);
                        }
                    }
                }
            }

            return lesMateriels;
        }

        //Consultation des incidents / tickets
        public static List<Ticket>
GetIncidentsPourTechnicien(Technicien tech)
        {
            List<Ticket> lesTickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // Tech niveau 1 : tickets niveau 1 + même région
(de l'utilisateur)
                    if (tech.getNiv_intervention() == "Niveau 1")
                    {
                        cmd.CommandText = @"
                    SELECT t.id_ticket, t.objet_demande, t.niv_urgence
                    FROM ticket t, utilisateur u
                    WHERE t.idU = u.idU
                      AND u.regionU = @region
                      AND t.niv_urgence = 'Niveau 1'
```

```
                    ORDER BY t.date_ticket DESC; ";

                        cmd.Parameters.AddWithValue("@region",
tech.getRegionT());
                    }
                    else
                    {
                        // Tech niveau > 1 : tous les tickets
                        cmd.CommandText = @"
                    SELECT id_ticket, objet_demande, niv_urgence,
etat_demande, date_ticket, idM, idT, idU,  id_phase
                    FROM ticket
                    ORDER BY date_ticket DESC;";
                    }

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string nivUrgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string dateTicket =
reader.GetString("date_ticket");
                            int idM = reader.GetInt32("idM");

                            // Si dans ta table idT peut être NULL, on
sécurise :
                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                            ? (int?)null
                            : reader.GetInt32("idT");

                            int idU = reader.GetInt32("idU");
                            int id_phase = reader.GetInt32("id_phase");
```

```
                            Ticket t = new Ticket(idTicket, objet,
nivUrgence, etat, dateTicket, idM, idT, idU, id_phase);
                            lesTickets.Add(t);
                        }
                    }
                }
            }

            return lesTickets;
        }

        //suivi des incidents déclaré par les utilisateurs
        public static List<Ticket> GetIncidentsPourUtilisateur(int
idUtilisateur)
        {
            List<Ticket> lesTickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                        SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                               date_ticket, idM, idT, idU,
                               0 AS id_phase
                        FROM ticket
                        WHERE idU = @idU
                        ORDER BY date_ticket DESC;
                    ";


                    cmd.Parameters.AddWithValue("@idU", idUtilisateur);

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
```

```
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string nivUrgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string dateTicket =
reader.GetString("date_ticket");
                            int idM = 0; // valeur par défaut si NULL /
vide / mauvais format
                            object rawIdM = reader["idM"];

                            if (rawIdM != DBNull.Value &&
int.TryParse(rawIdM.ToString(), out int parsedIdM))
                                idM = parsedIdM;

                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                                ? (int?)null
                                : reader.GetInt32("idT");

                            int idU = reader.GetInt32("idU");
                            int id_phase = reader.GetInt32("id_phase");

                            Ticket t = new Ticket(
                                idTicket, objet, nivUrgence, etat,
                                dateTicket, idM, idT, idU, id_phase);

                            lesTickets.Add(t);
                        }
                    }
                }
            }

            return lesTickets;
        }

        //prise en charge d'un incident par un technicien

        public static bool PrendreEnChargeIncident(int idTicket, int
idTechnicien)
```

```
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET idT = @idT,
                    etat_demande = 'En cours de traitement'
                WHERE id_ticket = @idTicket
                  AND idT IS NULL;
            ";

                    cmd.Parameters.AddWithValue("@idT", idTechnicien);
                    cmd.Parameters.AddWithValue("@idTicket", idTicket);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }


        //Passer le ticket en version résolu et enregistrement du
travail

        public static bool EnregistrerTravailEtResoudre(int idTicket,
Technicien tech, Phase_ticket phaseFinale)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (var tx = conn.BeginTransaction())
                {
                    try
                    {
                        // 1) Insert phase
```

```
                        using (MySqlCommand cmdPhase =
conn.CreateCommand())
                        {
                            cmdPhase.Transaction = tx;
                            cmdPhase.CommandText = @"
                        INSERT INTO phase_ticket
                        (id_ticket, idT, date_phase, heure_debut,
heure_fin, description_travail)
                        VALUES
                        (@idTicket, @idT, @datePhase, @heureDebut,
@heureFin, @description);
                    ";


cmdPhase.Parameters.AddWithValue("@idTicket",
phaseFinale.getIdTicket());
                            cmdPhase.Parameters.AddWithValue("@idT",
phaseFinale.getIdTechnicien());

cmdPhase.Parameters.AddWithValue("@datePhase",
phaseFinale.getDatePhase());

cmdPhase.Parameters.AddWithValue("@heureDebut",
phaseFinale.getHeureDebut());

cmdPhase.Parameters.AddWithValue("@heureFin",
phaseFinale.getHeureFin());

cmdPhase.Parameters.AddWithValue("@description",
phaseFinale.getDescriptionTravail());

                            cmdPhase.ExecuteNonQuery();
                        }

                        // 2) Update ticket => Résolue
                        using (MySqlCommand cmdTicket =
conn.CreateCommand())
                        {
                            cmdTicket.Transaction = tx;
                            cmdTicket.CommandText = @"
                        UPDATE ticket
                        SET etat_demande = 'Résolue'
                        WHERE id_ticket = @idTicket
```

```
                        AND idT = @idT
                        AND etat_demande = 'En cours de traitement';
                    ";


cmdTicket.Parameters.AddWithValue("@idTicket", idTicket);
                        cmdTicket.Parameters.AddWithValue("@idT",
tech.getIDTech());

                        int rows = cmdTicket.ExecuteNonQuery();
                        if (rows != 1)
                        {
                            tx.Rollback();
                            return false;
                        }
                    }

                    tx.Commit();
                    return true;
                }
                catch
                {
                    tx.Rollback();
                    throw;
                }
            }
        }

        //Nombre total d'incidents
        public static int GetNombreTotalIncidents()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT COUNT(*) FROM ticket;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
```

```
        }

        //Nombre d'incidents par état
        public static int GetNombreIncidentsParEtat(string etat)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT COUNT(*)
                FROM ticket
                WHERE etat_demande = @etat;";
                    cmd.Parameters.AddWithValue("@etat", etat);
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        //Nombre de jours moyen d'intervention
        public static double GetDureeMoyenneInterventionEnMinutes()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT AVG(TIMESTAMPDIFF(MINUTE, heure_debut,
heure_fin))
                FROM phase_ticket pt, ticket t
                WHERE pt.id_ticket = t.id_ticket
                AND t.etat_demande = 'Résolue';
            ";

                    object result = cmd.ExecuteScalar();
                    if (result == DBNull.Value)
                        return 0;
```

```
                    return Convert.ToDouble(result);
                }
            }
        }

        //statistique sur les techniciens
        public static List<StatTechnicien> GetStatsTechniciens()
        {
            List<StatTechnicien> stats = new List<StatTechnicien>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // IMPORTANT :
                    // - LEFT JOIN ticket : pour garder les techniciens
même s'ils n'ont aucun ticket
                    // - LEFT JOIN phase_ticket : pour calculer des
durées (si aucune phase => NULL => COALESCE)
                    // - Le DISTINCT dans les comptages évite de
compter plusieurs fois un ticket à cause des phases
                    cmd.CommandText = @"
                    SELECT
                        te.idT,
                        te.nomT,
                        te.prenomT,

                        COALESCE(COUNT(ti.id_ticket), 0) AS
nbTotalAssignes,

                        COALESCE(SUM(CASE WHEN ti.etat_demande = 'En
cours de traitement' THEN 1 ELSE 0 END), 0) AS nbEnCharge,
                        COALESCE(SUM(CASE WHEN ti.etat_demande =
'Résolue' THEN 1 ELSE 0 END), 0) AS nbResolus,
                        COALESCE(SUM(CASE WHEN ti.etat_demande =
'Clôturée' THEN 1 ELSE 0 END), 0) AS nbClotures

                    FROM technicien te
```

```
                    LEFT JOIN ticket ti ON ti.idT = te.idT
                    GROUP BY te.idT, te.nomT, te.prenomT
                    ORDER BY te.idT;";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idT = reader.GetInt32("idT");
                            string nom = reader.GetString("nomT");
                            string prenom =
reader.GetString("prenomT");

                            int nbEnCharge =
Convert.ToInt32(reader["nbEnCharge"]);
                            int nbResolus =
Convert.ToInt32(reader["nbResolus"]);
                            int nbClotures =
Convert.ToInt32(reader["nbClotures"]);
                            int nbTotalAssignes =
Convert.ToInt32(reader["nbTotalAssignes"]);

                            double dureeMoy = 0;
                            double dureeTotale = 0;

                            stats.Add(new StatTechnicien(
                                idT, nom, prenom,
                                nbEnCharge, nbResolus, nbClotures,
nbTotalAssignes,
                                dureeMoy, dureeTotale

                            ));
                        }
                    }
                }
            }
            return stats;
        }

        //Visualisation des statistique sur les utilisateurs

        public static List<StatUtilisateur> GetStatsUtilisateurs()
```

```
        {
            List<StatUtilisateur> stats = new List<StatUtilisateur>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT
                    u.idU,
                    u.nomU,
                    u.prenomU,

                    COUNT(t.id_ticket) AS nbDeclares,
                    COALESCE(SUM(CASE WHEN t.etat_demande = 'En cours
de traitement' THEN 1 ELSE 0 END), 0) AS nbEnCours,
                    COALESCE(SUM(CASE WHEN t.etat_demande = 'Résolue'
THEN 1 ELSE 0 END), 0) AS nbResolus

                FROM utilisateur u
                LEFT JOIN ticket t ON t.idU = u.idU
                GROUP BY u.idU, u.nomU, u.prenomU
                ORDER BY u.idU;
            ";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idU = reader.GetInt32("idU");
                            string nom = reader.GetString("nomU");
                            string prenom =
reader.GetString("prenomU");

                            int nbDeclares =
Convert.ToInt32(reader["nbDeclares"]);
                            int nbEnCours =
Convert.ToInt32(reader["nbEnCours"]);
```

```
                            int nbResolus =
Convert.ToInt32(reader["nbResolus"]);

                            stats.Add(new StatUtilisateur(
                                idU, nom, prenom,
                                nbDeclares, nbEnCours, nbResolus
                            ));
                        }
                    }
                }
            }
            return stats;
        }
        // ====== ID AUTOMATIQUES ======

        public static int GetNextIdResponsable()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(idR), 0) + 1
FROM responsable;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        public static int GetNextIdUtilisateur()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(idU), 0) + 1
FROM utilisateur;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
```

```
                }
            }
        }
        // ====== TECHNICIEN : ID AUTOMATIQUE ======
        public static int GetNextIdTechnicien()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(idT), 0) + 1
FROM technicien;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        // ========Matériel : Id automatique ======
        // enregistrer un matériel

        public static string GetNextIdMateriel()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // idM contient "1", "2", "3" ...
                    cmd.CommandText = "SELECT IFNULL(MAX(CAST(idM AS
UNSIGNED)), 0) + 1 FROM materiel;";
                    int next = Convert.ToInt32(cmd.ExecuteScalar());
                    return next.ToString();
                }
            }
        }
```

```
        // ====== TECHNICIEN : retrouver l'id via matricule ======
        public static int? GetIdTechnicienParMatricule(string
matricule)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idT FROM technicien WHERE
matriculeT = @matricule LIMIT 1;";
                    cmd.Parameters.AddWithValue("@matricule",
matricule);

                    object result = cmd.ExecuteScalar();
                    if (result == null || result == DBNull.Value)
return null;

                    return Convert.ToInt32(result);
                }
            }
        }
        public static Technicien GetTechnicienParId(int idT)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idT, nomT, prenomT, formation, competences,
niv_intervention,
                       matriculeT, adresseT, villeT, codePT,
date_embaucheT, regionT, idR
                FROM technicien
                WHERE idT = @idT;";
                    cmd.Parameters.AddWithValue("@idT", idT);
```

```
                    using (var r = cmd.ExecuteReader())
                    {
                        if (!r.Read()) return null;

                        return new Technicien(
                            r.GetInt32("idT"),
                            r.GetString("nomT"),
                            r.GetString("prenomT"),
                            r.GetString("formation"),
                            r.GetString("competences"),
                            r.GetString("niv_intervention"),
                            r.GetString("matriculeT"),
                            r.GetString("adresseT"),
                            r.GetString("villeT"),
                            r.GetString("codePT"),
                            r.GetDateTime("date_embaucheT"),
                            r.GetString("regionT"),
                            r.GetInt32("idR")
                        );
                    }
                }
            }
        }

        public static int? GetIdUtilisateurParMatricule(string
matricule)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idU FROM utilisateur
WHERE matriculeU = @matricule LIMIT 1;";
                    cmd.Parameters.AddWithValue("@matricule",
matricule);

                    object result = cmd.ExecuteScalar();
                    if (result == null || result == DBNull.Value)
return null;
```

```
                    return Convert.ToInt32(result);
                }
            }
        }

        //Modifier ou supprimer un utilisateur

        // ===== UTILISATEUR : charger un utilisateur par id =====
        public static Utilisateur GetUtilisateurParId(int idU)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idU, nomU, prenomU, matriculeU, adresseU,
villeU, codePU, date_embaucheU, regionU, idR
                FROM utilisateur
                WHERE idU = @idU;";
                    cmd.Parameters.AddWithValue("@idU", idU);

                    using (var r = cmd.ExecuteReader())
                    {
                        if (!r.Read()) return null;

                        return new Utilisateur(
                            r.GetInt32("idU"),
                            r.GetString("nomU"),
                            r.GetString("prenomU"),
                            r.GetString("matriculeU"),
                            r.GetString("adresseU"),
                            r.GetString("villeU"),
                            r.GetString("codePU"),
                            r.GetString("date_embaucheU"),
                            r.GetString("regionU"),
                            r.GetInt32("idR")
                        );
                    }
```

```
                }
            }
        }

        // ===== UTILISATEUR : liste simple pour remplir la ComboBox
=====
        public static List<int> GetTousLesIdsUtilisateurs()
        {
            List<int> ids = new List<int>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idU FROM utilisateur
ORDER BY idU;";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetInt32("idU"));
                    }
                }
            }

            return ids;
        }


        //Liste déroulante des ids des matériel pour les supprimer
        // ===== MATERIEL : liste des IDs pour ComboBox =====
        public static List<string> GetTousLesIdsMateriels()
        {
            List<string> ids = new List<string>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
```

```
                {
                    cmd.CommandText = "SELECT idM FROM materiel ORDER
BY CAST(idM AS UNSIGNED);";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetString("idM"));
                    }
                }
            }

            return ids;
        }



        ///
        ///
        /// selection d'un responsable pour en attribuer un à un
technicien
        ///
        ///
        public static List<int> GetTousLesIdsResponsables()
        {
            List<int> ids = new List<int>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idR FROM responsable
ORDER BY idR;";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetInt32("idR"));
                    }
                }
            }
            return ids;
```

```
        }



        //Enregistrer un ticket
        public static int GetNextIdTicket()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(id_ticket), 0)
+ 1 FROM ticket;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        //Consulter les ticket non attribuer
        public static List<Ticket> GetTicketsNonAttribues()
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                        SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                               date_ticket, idM, idT, idU
                        FROM ticket
                        WHERE idT IS NULL
                        ORDER BY date_ticket DESC;
                    ";
```

```
                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string urgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string date =
reader.GetString("date_ticket");
                            int idM = reader.GetInt32("idM");

                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                                ? (int?)null
                                : reader.GetInt32("idT");

                            int idU = reader.GetInt32("idU");

                            tickets.Add(new Ticket(idTicket, objet,
urgence, etat, date, idM, idT, idU, 0));
                        }
                    }
                }
            }

            return tickets;
        }
        //récupération de l'id d'un technicien
        public static List<Technicien> GetTousLesTechniciens()
        {
            List<Technicien> techs = new List<Technicien>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
```

```
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idT, nomT, prenomT, formation, competences,
niv_intervention,
                       matriculeT, adresseT, villeT, codePT,
date_embaucheT, regionT, idR
                FROM technicien
                ORDER BY idT;";

                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            techs.Add(new Technicien(
                                r.GetInt32("idT"),
                                r.GetString("nomT"),
                                r.GetString("prenomT"),
                                r.GetString("formation"),
                                r.GetString("competences"),
                                r.GetString("niv_intervention"),
                                r.GetString("matriculeT"),
                                r.GetString("adresseT"),
                                r.GetString("villeT"),
                                r.GetString("codePT"),
                                r.GetDateTime("date_embaucheT"),
                                r.GetString("regionT"),
                                r.GetInt32("idR")
                            ));
                        }
                    }
                }
            }

            return techs;
        }

        //Passer le ticke en résolu
        public static bool PasserTicketEnResolu(int idTicket, int
idTechnicien)
        {
```

```
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET etat_demande = 'Résolue'
                WHERE id_ticket = @idTicket
                  AND idT = @idT
                  AND etat_demande = 'En cours de traitement';
            ";

                    cmd.Parameters.AddWithValue("@idTicket", idTicket);
                    cmd.Parameters.AddWithValue("@idT", idTechnicien);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }

        //Ne charger (dans la liste des tickets) que les ticket en
cours de traitement

        public static List<Ticket> GetTicketsEnCoursPourTechnicien(int
idTechnicien)
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                       date_ticket, idM, idT, idU
```

```
                FROM ticket
                WHERE idT = @idT
                  AND TRIM(etat_demande) LIKE 'En cours%'
                ORDER BY date_ticket DESC;
            ";
                cmd.Parameters.AddWithValue("@idT", idTechnicien);

                using (MySqlDataReader reader =
cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        int idTicket =
reader.GetInt32("id_ticket");
                        string objet =
reader.GetString("objet_demande");
                        string urgence =
reader.GetString("niv_urgence");
                        string etat =
reader.GetString("etat_demande");
                        string date =
reader.GetString("date_ticket");
                        int idM = reader.GetInt32("idM");

                        int? idT =
reader.IsDBNull(reader.GetOrdinal("idT")) ? (int?)null :
reader.GetInt32("idT");
                        int idU = reader.GetInt32("idU");

                        tickets.Add(new Ticket(idTicket, objet,
urgence, etat, date, idM, idT, idU, 0));
                    }
                }
            }

            return tickets;
        }

        //débug le nom "En cours de traitement
        public static string DebugEtatTicket(int idTicket)
        {
```

```
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT CONCAT('[', etat_demande,
']') FROM ticket WHERE id_ticket = @id;";
                    cmd.Parameters.AddWithValue("@id", idTicket);
                    object res = cmd.ExecuteScalar();
                    return res == null ? "NULL" : res.ToString();
                }
            }
        }

        //Clôturer un ticket (UNIQUEMENT si Résolu)
        public static bool CloturerTicket(int idTicket)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET etat_demande = 'Clôturée'
                WHERE id_ticket = @idTicket
                  AND etat_demande = 'Résolue';
            ";

                    cmd.Parameters.AddWithValue("@idTicket", idTicket);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }
        //Charger UNIQUEMENT les tickets résolus (nouvelle ComboBox)
        // ===== Tickets résolus =====
        public static List<Ticket> GetTicketsResolus()
```

```
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                       date_ticket, idM, idT, idU
                FROM ticket
                WHERE etat_demande = 'Résolue'
                ORDER BY date_ticket DESC;
            ";

                    using (MySqlDataReader r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            tickets.Add(new Ticket(
                                r.GetInt32("id_ticket"),
                                r.GetString("objet_demande"),
                                r.GetString("niv_urgence"),
                                r.GetString("etat_demande"),
                                r.GetString("date_ticket"),
                                r.GetInt32("idM"),
                                r.IsDBNull(r.GetOrdinal("idT")) ?
(int?)null : r.GetInt32("idT"),
                                r.GetInt32("idU"),
                                0
                            ));
                        }
                    }
                }
            }
            return tickets;
        }

        //gestion du niveau du technicien (selon lequel est cocher
```

```
        public static List<Ticket> GetTicketsVisiblesParNiveau(string
niveauSelectionne, bool estResponsable)
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    if (estResponsable)
                    {
                        // Responsable : TOUS les tickets (attribués +
non attribués)
                        cmd.CommandText = @"
                    SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                           date_ticket, idM, idT, idU
                    FROM ticket
                    ORDER BY date_ticket DESC;";
                    }
                    else
                    {
                        // Détermine quels niveaux sont visibles
                        string niveauxVisibles;
                        if (niveauSelectionne == "Niveau 1")
niveauxVisibles = "('Niveau 1')";
                        else if (niveauSelectionne == "Niveau 2")
niveauxVisibles = "('Niveau 1','Niveau 2')";
                        else niveauxVisibles = "('Niveau 1','Niveau
2','Niveau 3')"; // Niveau 3

                        // Ici : on affiche les tickets attribués à un
technicien dont le niveau est dans la liste
                        cmd.CommandText = $@"
                    SELECT t.id_ticket, t.objet_demande, t.niv_urgence,
t.etat_demande,
                           t.date_ticket, t.idM, t.idT, t.idU
                    FROM ticket t
                    INNER JOIN technicien te ON te.idT = t.idT
```

```
                    WHERE te.niv_intervention IN {niveauxVisibles}
                    ORDER BY t.date_ticket DESC;";
                    }

                    using (MySqlDataReader r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            int idTicket =
Convert.ToInt32(r["id_ticket"]);
                            string objet =
r["objet_demande"]?.ToString();
                            string urgence =
r["niv_urgence"]?.ToString();
                            string etat =
r["etat_demande"]?.ToString();
                            string date = r["date_ticket"]?.ToString();

                            int idM = SafeToNullableInt(r["idM"]) ?? 0;
                            int? idT = SafeToNullableInt(r["idT"]);
                            int idU = SafeToNullableInt(r["idU"]) ?? 0;

                            // Si tu n'as pas id_phase dans la requête,
laisse 0
                            int idPhase = 0;

                            tickets.Add(new Ticket(
                                    idTicket,
                                    objet,
                                    urgence,
                                    etat,
                                    date,
                                    idM,
                                    idT,
                                    idU,
                                    idPhase
                                    ));
                        }
                    }
                }
            }

            return tickets;
```

```
        }

        private static int? SafeToNullableInt(object value)
        {
            if (value == null || value == DBNull.Value) return null;

            // si c'est déjà un int/long, on convertit proprement
            if (value is int i) return i;
            if (value is long l) return (int)l;

            string s = value.ToString().Trim();

            // gère chaînes vides ou "null"
            if (string.IsNullOrWhiteSpace(s) || s.Equals("null",
StringComparison.OrdinalIgnoreCase))
                return null;

            // tentative de conversion
            if (int.TryParse(s, out int result))
                return result;

            return null; // si c'est pas un nombre, on retourne null au
lieu de planter
        }




    }
}
```

Class Utilisateur :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Utilisateur
    {
        private int idu;
        private string nomu;
        private string prenomu;
        private string matriculeu;
        private string adresseu;
        private string villeu;
        private string codepu;
```

```
        private string date_embaucheu;
        private string regionu;
        private int idr;


        public Utilisateur(int UnIdu, string UnNomu, string UnPrenomu,
string UnMatriculeu, string UneAdresseu, string UneVilleu, string
UnCodepu, string UneDate_embauche, string UneRegionu, int UnIdr)
        {
            idu = UnIdu;
            nomu = UnNomu;
            prenomu = UnPrenomu;
            matriculeu = UnMatriculeu;
            adresseu = UneAdresseu;
            villeu = UneVilleu;
            codepu = UnCodepu;
            date_embaucheu = UneDate_embauche;
            regionu = UneRegionu;
            idr = UnIdr;
        }
        public int getIDU()
        {
            return idu;
        }
        public string getNomu()
        {
            return nomu;
        }

        public string getPrenomu()
        {
            return prenomu;
        }

        public string getMatriculeu()
        {
            return matriculeu;
        }

        public string getAdresseu()
        {
            return adresseu;
        }
```

```
        public string getVilleu()
        {
            return villeu;
        }

        public string getCodepu()
        {
            return codepu;
        }

        public string getDate_embaucheu()
        {
            return date_embaucheu;
        }

        public string getRegionu()
        {
            return regionu;
        }

        public int getIdr()
        {
            return idr;
        }

        public void setRegionu(string newregion)
        {
            regionu = newregion;
        }
    }
}
```

Class Matériel :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Materiel
    {
        private string idm;
        private string type_materiel;
        private DateTime? date_achat;//datetime ? --> veut dire qu'il
peut être nullable
        private DateTime? date_location;//datetime ? --> veut dire
qu'il peut être nullable
        private string garanti;
```

```
        private int idu;



        public Materiel(string UnIdm, string UnType_materiel, DateTime?
UneDate_achat, DateTime? Unedate_location, string UneGaranti, int
UnIdu)
        {
            idm = UnIdm;
            type_materiel = UnType_materiel;
            date_achat = UneDate_achat;
            date_location = Unedate_location;
            garanti = UneGaranti;
            idu = UnIdu;
        }
        public string getIDM()
        {
            return idm;
        }
        public string getType_materiel()
        {
            return type_materiel;
        }
        public DateTime? getDate_achat()
        {
            return date_achat;
        }
        public DateTime? getDate_location()
        {
            return date_location;
        }
        public string getGaranti()
        {
            return garanti;
        }

        public int getIDU()
        {
            return idu;
        }
        public override string ToString()
        {
            return $"{idm} - {type_materiel} - {garanti}";
```

```
        }

    }
}
```

Phase ticket :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Phase_ticket
    {
        private int idPhase;
        private int idTicket;
        private int idTechnicien;

        private DateTime datePhase;
        private TimeSpan heureDebut;
        private TimeSpan heureFin;
```

```
        private string descriptionTravail;

        public Phase_ticket(int unIdTicket, int unIdTech, DateTime
uneDate,
                            TimeSpan debut, TimeSpan fin, string
description)
        {
            idTicket = unIdTicket;
            idTechnicien = unIdTech;
            datePhase = uneDate;
            heureDebut = debut;
            heureFin = fin;
            descriptionTravail = description;
        }

        public int getIdTicket()
        {
            return idTicket;
        }

        public int getIdTechnicien()
        {
            return idTechnicien;
        }

        public DateTime getDatePhase()
        {
            return datePhase;
        }

        public TimeSpan getHeureDebut()
        {
            return heureDebut;
        }
        public TimeSpan getHeureFin()
        {
            return heureFin;
        }

        public string getDescriptionTravail()
        {
            return descriptionTravail;
```

```
        }


    }
}
```

Class Responsable :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Responsable
    {
        private int idr;
        private string nomr;
        private string prenomr;
        private string matriculer;
        private string adresser;
        private string viller;
        private string codepr;
```

```
        private string date_embaucher;
        private string regionr;


        public Responsable(int UnIdr, string UnNomr, string UnPrenomr,
string UnMatriculer, string UneAdresser, string UneViller, string
UnCodepr, string UneDate_embaucher, string UneRegionr)
        {
            idr = UnIdr;
            nomr = UnNomr;
            prenomr = UnPrenomr;
            matriculer = UnMatriculer;
            adresser = UneAdresser;
            viller = UneViller;
            codepr = UnCodepr;
            date_embaucher = UneDate_embaucher;
            regionr = UneRegionr;
        }

        public int getIDR()
        {
            return idr;
        }
        public string getNomr()
        {
            return nomr;
        }
        public string getPrenomr()
        {
            return prenomr;
        }
        public string getDate_embaucher()
        {
            return date_embaucher;
        }
        public string getMatriculer()
        {
            return matriculer;
        }
        public string getAdresser()
        {
            return adresser;
        }
```

```
        public string getViller()
        {
            return viller;
        }
        public string getCodepr()
        {
            return codepr;
        }
        public string getRegionr()
        {
            return regionr;
        }
        public void setRegionr(string newregionr)
        {
            regionr = newregionr;
        }
    }
}
```

Class Technicien :

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projet_final_C_
{
    internal class Technicien
    {
        private int idT;
        private string nomT;
        private string prenomT;
        private string formation;
        private string competences;
```

```
        private string niv_intervention;
        private string matriculeT;
        private string adresseT;
        private string villeT;
        private string codePT;
        private DateTime date_embaucheT;
        private string regionT;
        private int idR;
        //private int id_phase;

        public Technicien(int UnIdt, string UnNomt, string UnPrenomT,
string UneFormation, string UneCompetence, string UnNiv_intervention,
string UnMatricule, string UneAdresse, string UneVille, string
UnCodePT, DateTime UneDate_embaucheT, string UneRegion, int UnIdR)
        {
            idT = UnIdt;
            nomT = UnNomt;
            prenomT = UnPrenomT;
            formation = UneFormation;
            competences = UneCompetence;
            niv_intervention = UnNiv_intervention;
            matriculeT = UnMatricule;
            adresseT = UneAdresse;
            villeT = UneVille;
            codePT = UnCodePT;
            date_embaucheT = UneDate_embaucheT;
            regionT = UneRegion;
            idR = UnIdR;
            //id_phase = UnId_phase;

        }
        public int getIDTech()
        {
            return idT;
        }
        public string getNomt()
        {
            return nomT;
        }

        public string getPrenomT()
        {
            return prenomT;
```

```
        }

        public string getMatriculeT()
        {
            return matriculeT;
        }

        public string getAdresseT()
        {
            return adresseT;
        }

        public string getVilleT()
        {
            return villeT;
        }

        public string getCodePT()
        {
            return codePT;
        }

        public DateTime getDate_embaucheT()
        {
            return date_embaucheT;
        }

        public string getRegionT()
        {
            return regionT;
        }

        public string getFormation()
        {
            return formation;
        }

        public string getCompetences()
        {
            return competences;
        }

        public string getNiv_intervention()
```

```
        {
            return niv_intervention;
        }
        public int getIdr()
        {
            return idR;
        }
        //public int getIDPhase()
        //{
        //    return id_phase;
        //}


        public void setRegionT (string newregion)
        {
            regionT = newregion;
        }
        public void setFormationT(string newformation)
        {
            formation = newformation;
        }
        public override string ToString()
        {
            return $"{idT} - {nomT} {prenomT} ({niv_intervention})";
        }

    }

}
```

Class Ticket

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Ticket
    {
        private int id_ticket;
        private string objet_demande;
        private string niv_urgence;
        private string etat_demande;
        private string date_ticket;
        private int idM;
        private int? idT;
```

```
        private int idU;
        private int id_phase;

        public Ticket(int UnId_ticket, string UnObjet_demande, string
UnNiveau_urgence, string UnEtat_demande, string UneDate_ticket, int
UnIdM, int? UnIdT, int UnIdU, int UnId_phase)
        {
            id_ticket = UnId_ticket;
            objet_demande = UnObjet_demande;
            niv_urgence = UnNiveau_urgence;
            etat_demande = UnEtat_demande;
            date_ticket = UneDate_ticket;
            idM = UnIdM;
            idT = UnIdT;
            idU = UnIdU;
            id_phase = UnId_phase;
        }
        public int getIDTicket()
        {
            return id_ticket;
        }
        public string getObjet_demande()
        {
            return objet_demande;
        }
        public string getNiv_urgence()
        {
            return niv_urgence;
        }

        public string getEtat_demande()
        {
            return etat_demande;
        }
        public string getDate_ticket()
        {
            return date_ticket;
        }

        public int getIDM()
        {
            return idM;
        }
```

```
        public int? getIDTech()
        {
            return idT;
        }

        public int getIDU()
        {
            return idU;
        }
        public int getIDPhase()
        {
            return id_phase;
        }
        public void setNouvelle_Objet_demande(string Nouvelle_demande)
        {
            objet_demande = Nouvelle_demande;
        }

        // (optionnel) setter pour affecter un technicien après
création
        public void setIDTech(int? nouvelIdT)
        {
            idT = nouvelIdT;
        }
        public override string ToString()
        {
            return $"#{id_ticket} - {objet_demande} ({niv_urgence}) -
{etat_demande}";
        }




    }
}
```

Classe Session :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Session
    {
        public static Technicien TechnicienConnecte { get; set; }
        public static Utilisateur UtilisateurConnecte { get; set; }
    }
}
```

Class Stats technicien :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class StatTechnicien
    {
        private int idT;
        private string nom;
        private string prenom;

        private int nbEnCharge;    // "En cours de traitement"
        private int nbResolus;     // "Résolue"
        private int nbClotures;    // "Clôturée" (si tu l'utilises)
        private int nbTotalAssignes;
```

```
        private double dureeMoyMinutes;   // moyenne des phases sur ses
tickets
        private double dureeTotaleMinutes; // somme des phases sur ses
tickets

        public StatTechnicien(int UnIdT, string UnNom, string UnPrenom,
                              int UnNbEnCharge, int UnNbResolus, int
UnNbClotures, int UnNbTotalAssignes,
                              double UneDureeMoyMinutes, double
UneDureeTotaleMinutes)
        {
            idT = UnIdT;
            nom = UnNom;
            prenom = UnPrenom;
            nbEnCharge = UnNbEnCharge;
            nbResolus = UnNbResolus;
            nbClotures = UnNbClotures;
            nbTotalAssignes = UnNbTotalAssignes;
            dureeMoyMinutes = UneDureeMoyMinutes;
            dureeTotaleMinutes = UneDureeTotaleMinutes;
        }

        // ===== GETTERS (même écriture que dans Phase_ticket) =====

        public int getIdT()
        {
            return idT;
        }

        public string getNomT()
        {
            return nom;
        }

        public string getPrenomT()
        {
            return prenom;
        }

        public int getNbEnCharge()
        {
            return nbEnCharge;
```

```
        }

        public int getNbResolus()
        {
            return nbResolus;
        }

        public int getNbClotures()
        {
            return nbClotures;
        }

        public int getNbTotalAssignes()
        {
            return nbTotalAssignes;
        }

        public double getDureeMoyMinutes()
        {
            return dureeMoyMinutes;
        }

        public double getDureeTotaleMinutes()
        {
            return dureeTotaleMinutes;
        }
        // ===== SETTERS (optionnels mais cohérents avec ton projet)
=====

        public void setIdT(int unIdT)
        {
            idT = unIdT;
        }

        public void setNomT(string unNomT)
        {
            nom = unNomT;
        }

        public void setPrenomT(string unPrenomT)
        {
            prenom = unPrenomT;
        }
```

```
        public void setNbEnCharge(int unNbEnCharge)
        {
            nbEnCharge = unNbEnCharge;
        }

        public void setNbResolus(int unNbResolus)
        {
            nbResolus = unNbResolus;
        }

        public void setNbClotures(int unNbClotures)
        {
            nbClotures = unNbClotures;
        }

        public void setNbTotalAssignes(int unNbTotalAssignes)
        {
            nbTotalAssignes = unNbTotalAssignes;
        }

        public void setDureeMoyMinutes(double uneDureeMoyMinutes)
        {
            dureeMoyMinutes = uneDureeMoyMinutes;
        }
        public void setDureeTotaleMinutes(double uneDureeTotaleMinutes)
        {
            dureeTotaleMinutes = uneDureeTotaleMinutes;
        }
    }
}
```

Classe Stats Utilisateurs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class StatUtilisateur
    {

        private int idU;
        private string nom;
        private string prenom;

        private int nbIncidentsDeclares;
        private int nbEnCours;
        private int nbResolus;
```

```
        public StatUtilisateur(int unIdU, string unNom, string
unPrenom,
                               int unNbDeclares, int unNbEnCours, int
unNbResolus)
        {
            idU = unIdU;
            nom = unNom;
            prenom = unPrenom;
            nbIncidentsDeclares = unNbDeclares;
            nbEnCours = unNbEnCours;
            nbResolus = unNbResolus;
        }

        // ===== GETTERS (même écriture que tes autres fichiers) =====

        public int getIDU()
        {
            return idU;
        }

        public string getNomU()
        {
            return nom;
        }

        public string getPrenomU()
        {
            return prenom;
        }

        public int getNbIncidentsDeclares()
        {
            return nbIncidentsDeclares;
        }

        public int getNbEnCours()
        {
            return nbEnCours;
        }

        public int getNbResolus()
        {
```

```
            return nbResolus;
        }


        // ===== SETTERS (si tu fais pareil ailleurs) =====

        public void setIDU(int newIdU)
        {
            idU = newIdU;
        }

        public void setNomU(string newNomU)
        {
            nom = newNomU;
        }

        public void setPrenomU(string newPrenomU)
        {
            prenom = newPrenomU;
        }

        public void setNbIncidentsDeclares(int newNbDeclares)
        {
            nbIncidentsDeclares = newNbDeclares;
        }

        public void setNbEnCours(int newNbEnCours)
        {
            nbEnCours = newNbEnCours;
        }

        public void setNbResolus(int newNbResolus)
        {
            nbResolus = newNbResolus;
        }
    }
}
```

Class Base de Donnée

Class Base de Donnée

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

namespace Projet_final_C_
{
    internal class Base_de_donnee
    //création de tous les inserts
    {
        //insert table technicien
        public static void AjouterTechnicien(Technicien UnTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                    INSERT INTO technicien
                    (idT, niv_intervention, formation, competences,
nomT, prenomT, matriculeT,
                     adresseT, villeT, codePT, date_embaucheT, regionT,
idR)
                    VALUES
                    (@idT, @niv_intervention, @formation, @competences,
@nomT, @prenomT, @matriculeT,
                     @adresseT, @villeT, @codePT, @date_embaucheT,
@regionT, @idR);
                    ";
                    cmd.Parameters.AddWithValue("@idT",
UnTechnicien.getIDTech());
                    cmd.Parameters.AddWithValue("@niv_intervention",
UnTechnicien.getNiv_intervention());
```

```
                    cmd.Parameters.AddWithValue("@formation",
UnTechnicien.getFormation());
                    cmd.Parameters.AddWithValue("@competences",
UnTechnicien.getCompetences());
                    cmd.Parameters.AddWithValue("@nomT",
UnTechnicien.getNomt());
                    cmd.Parameters.AddWithValue("@prenomT",
UnTechnicien.getPrenomT());
                    cmd.Parameters.AddWithValue("@matriculeT",
UnTechnicien.getMatriculeT());
                    cmd.Parameters.AddWithValue("@adresseT",
UnTechnicien.getAdresseT());
                    cmd.Parameters.AddWithValue("@villeT",
UnTechnicien.getVilleT());
                    cmd.Parameters.AddWithValue("@codePT",
UnTechnicien.getCodePT());
                    cmd.Parameters.Add("@date_embaucheT",
MySqlDbType.Date).Value = UnTechnicien.getDate_embaucheT().Date;
                    cmd.Parameters.AddWithValue("@regionT",
UnTechnicien.getRegionT());
                    cmd.Parameters.AddWithValue("@idR",
UnTechnicien.getIdr());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTechnicien.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table responsable
        public static void AjouterResponsable(Responsable
UnResponsable)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
```

```
                    cmd.CommandText = "INSERT INTO responsable (idR,
nomR, prenomR, matriculeR, adresseR, villeR, codePR, date_embaucheR,
regionR) VALUES (@idR, @nomR, @prenomR, @matriculeR, @adresseR,
@villeR, @codePR, @date_embaucheR, @regionR)";
                    cmd.Parameters.AddWithValue("@idR",
UnResponsable.getIDR());
                    cmd.Parameters.AddWithValue("@nomR",
UnResponsable.getNomr());
                    cmd.Parameters.AddWithValue("@prenomR",
UnResponsable.getPrenomr());
                    cmd.Parameters.AddWithValue("@matriculeR",
UnResponsable.getMatriculer());
                    cmd.Parameters.AddWithValue("@adresseR",
UnResponsable.getAdresser());
                    cmd.Parameters.AddWithValue("@villeR",
UnResponsable.getViller());
                    cmd.Parameters.AddWithValue("@codePR",
UnResponsable.getCodepr());
                    cmd.Parameters.AddWithValue("@date_embaucheR",
UnResponsable.getDate_embaucher());
                    cmd.Parameters.AddWithValue("@regionR",
UnResponsable.getRegionr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table utilisateur
        public static void AjouterUtilisateur(Utilisateur
UnUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "INSERT INTO utilisateur (idU,
nomU, prenomU, matriculeU, adresseU, villeU, codePU, date_embaucheU,
```

```
regionU, idR) VALUES (@idU, @nomU, @prenomU, @matriculeU, @adresseU,
@villeU, @codePU, @date_embaucheU, @regionU, @idR)";
                    cmd.Parameters.AddWithValue("@idU",
UnUtilisateur.getIDU());
                    cmd.Parameters.AddWithValue("@nomU",
UnUtilisateur.getNomu());
                    cmd.Parameters.AddWithValue("@prenomU",
UnUtilisateur.getPrenomu());
                    cmd.Parameters.AddWithValue("@matriculeU",
UnUtilisateur.getMatriculeu());
                    cmd.Parameters.AddWithValue("@adresseU",
UnUtilisateur.getAdresseu());
                    cmd.Parameters.AddWithValue("@villeU",
UnUtilisateur.getVilleu());
                    cmd.Parameters.AddWithValue("@codePU",
UnUtilisateur.getCodepu());
                    cmd.Parameters.AddWithValue("@date_embaucheU",
UnUtilisateur.getDate_embaucheu());
                    cmd.Parameters.AddWithValue("@regionU",
UnUtilisateur.getRegionu());
                    cmd.Parameters.AddWithValue("@idR",
UnUtilisateur.getIdr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //insert table matériels
        public static void AjouterMateriel(Materiel UnMateriel)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "Insert Into materiel (idM,
type_materiel, date_Achat, date_location, garanti, idU) Values (@idM,
@type_materiel, @date_Achat, @date_location, @garanti, @idU)";
```

```
                    cmd.Parameters.AddWithValue("@idM",
UnMateriel.getIDM());
                    cmd.Parameters.AddWithValue("@type_materiel",
UnMateriel.getType_materiel());
                    //Gestion de la nullité des dates achats et
locations
                    if (UnMateriel.getDate_achat() == null)
                        cmd.Parameters.AddWithValue("@date_Achat",
DBNull.Value);
                    else
                        cmd.Parameters.AddWithValue("@date_Achat",
UnMateriel.getDate_achat());

                    if (UnMateriel.getDate_location() == null)
                        cmd.Parameters.AddWithValue("@date_location",
DBNull.Value);
                    else
                        cmd.Parameters.AddWithValue("@date_location",
UnMateriel.getDate_location());
                    //
                    cmd.Parameters.AddWithValue("@garanti",
UnMateriel.getGaranti());
                    cmd.Parameters.AddWithValue("@idU",
UnMateriel.getIDU());

                    cmd.ExecuteNonQuery();
                }
            }

        }
        //insert table ticket
        public static void AjouterTicket(Ticket UnTicket)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
```

```
                    cmd.CommandText = "INSERT INTO ticket (id_ticket,
objet_demande, niv_urgence, etat_demande, date_ticket, idM, idT, idU)
VALUES (@id_ticket, @objet_demande, @niv_urgence, @etat_demande,
@date_ticket, @idM, @idT, @idU)";
                    cmd.Parameters.AddWithValue("@id_ticket",
UnTicket.getIDTicket());
                    cmd.Parameters.AddWithValue("@objet_demande",
UnTicket.getObjet_demande());
                    cmd.Parameters.AddWithValue("@niv_urgence",
UnTicket.getNiv_urgence());
                    cmd.Parameters.AddWithValue("@etat_demande",
UnTicket.getEtat_demande());
                    cmd.Parameters.AddWithValue("@date_ticket",
UnTicket.getDate_ticket());
                    cmd.Parameters.AddWithValue("@idM",
UnTicket.getIDM());
                    if (UnTicket.getIDTech() == null)
                        cmd.Parameters.AddWithValue("@idT",
DBNull.Value);
                    else
                        cmd.Parameters.AddWithValue("@idT",
UnTicket.getIDTech());
                    cmd.Parameters.AddWithValue("@idU",
UnTicket.getIDU());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTicket.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //insert des phases d'un ticket
        public static void AjouterPhaseTicket(Phase_ticket phase)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
```

```
                {
                    cmd.CommandText = @"
                INSERT INTO phase_ticket
                (id_ticket, idT, date_phase, heure_debut, heure_fin,
description_travail)
                VALUES
                (@idTicket, @idT, @datePhase, @heureDebut, @heureFin,
@description)
            ";

                    cmd.Parameters.AddWithValue("@idTicket",
phase.getIdTicket());
                    cmd.Parameters.AddWithValue("@idT",
phase.getIdTechnicien());
                    cmd.Parameters.AddWithValue("@datePhase",
phase.getDatePhase());
                    cmd.Parameters.AddWithValue("@heureDebut",
phase.getHeureDebut());
                    cmd.Parameters.AddWithValue("@heureFin",
phase.getHeureFin());
                    cmd.Parameters.AddWithValue("@description",
phase.getDescriptionTravail());

                    cmd.ExecuteNonQuery();
                }
            }
        }


        //Création des modifications

        //modification table technicien

        public static void ModifierTechnicien(Technicien UnTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
```

```
                {
                    cmd.CommandText = "UPDATE technicien SET
niv_intervention = @niv_intervention, formation = @formation,
competences = @competences, nomT = @nomT, prenomT = @prenomT,
matriculeT = @matriculeT, adresseT = @adresseT, villeT = @villeT,
codePT = @codePT, date_embaucheT = @date_embaucheT, regionT = @regionT,
idR = @idR WHERE idT = @idT;";
                    cmd.Parameters.AddWithValue("@idT",
UnTechnicien.getIDTech());
                    cmd.Parameters.AddWithValue("@niv_intervention",
UnTechnicien.getNiv_intervention());
                    cmd.Parameters.AddWithValue("@formation",
UnTechnicien.getFormation());
                    cmd.Parameters.AddWithValue("@competences",
UnTechnicien.getCompetences());
                    cmd.Parameters.AddWithValue("@nomT",
UnTechnicien.getNomt());
                    cmd.Parameters.AddWithValue("@prenomT",
UnTechnicien.getPrenomT());
                    cmd.Parameters.AddWithValue("@matriculeT",
UnTechnicien.getMatriculeT());
                    cmd.Parameters.AddWithValue("@adresseT",
UnTechnicien.getAdresseT());
                    cmd.Parameters.AddWithValue("@villeT",
UnTechnicien.getVilleT());
                    cmd.Parameters.AddWithValue("@codePT",
UnTechnicien.getCodePT());
                    cmd.Parameters.AddWithValue("@date_embaucheT",
UnTechnicien.getDate_embaucheT());
                    cmd.Parameters.AddWithValue("@regionT",
UnTechnicien.getRegionT());
                    cmd.Parameters.AddWithValue("@idR",
UnTechnicien.getIdr());
                    //cmd.Parameters.AddWithValue("@id_phase",
UnTechnicien.getIDPhase());

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //Modification d'un utilisateur
```

```
        public static void ModifierUtilisateur(Utilisateur
UnUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "UPDATE utilisateur SET nomU =
@nomU, prenomU = @prenomU, matriculeU = @matriculeU, adresseU =
@adresseU, villeU = @villeU, codePU = @codePU, date_embaucheU =
@date_embaucheU, regionU = @regionU, idR = @idR WHERE idU = @idU;";
                    cmd.Parameters.AddWithValue("@idU",
UnUtilisateur.getIDU());
                    cmd.Parameters.AddWithValue("@nomU",
UnUtilisateur.getNomu());
                    cmd.Parameters.AddWithValue("@prenomU",
UnUtilisateur.getPrenomu());
                    cmd.Parameters.AddWithValue("@matriculeU",
UnUtilisateur.getMatriculeu());
                    cmd.Parameters.AddWithValue("@adresseU",
UnUtilisateur.getAdresseu());
                    cmd.Parameters.AddWithValue("@villeU",
UnUtilisateur.getVilleu());
                    cmd.Parameters.AddWithValue("@codePU",
UnUtilisateur.getCodepu());
                    cmd.Parameters.AddWithValue("@date_embaucheU",
UnUtilisateur.getDate_embaucheu());
                    cmd.Parameters.AddWithValue("@regionU",
UnUtilisateur.getRegionu());
                    cmd.Parameters.AddWithValue("@idR",
UnUtilisateur.getIdr());

                    cmd.ExecuteNonQuery();
                }
            }
        }
        //Création des suppressions
        //Suppression d'un technicien
```

```
        public static void SupprimerTechnicien(int idTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM technicien WHERE idT
= @idT;";
                    cmd.Parameters.AddWithValue("@idT", idTechnicien);

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //Suppression d'un utilisateur
        public static void SupprimerUtilisateur(int idUtilisateur)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM utilisateur WHERE
idU = @idU;";
                    cmd.Parameters.AddWithValue("@idU", idUtilisateur);

                    cmd.ExecuteNonQuery();
                }
            }
        }

        //Suppression d'un matériel
        // Suppression d'un matériel (CORRIGÉ)
```

```
        public static bool SupprimerMateriel(string idM)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "DELETE FROM materiel WHERE idM =
@idM;";
                    cmd.Parameters.AddWithValue("@idM", idM);

                    return cmd.ExecuteNonQuery() > 0;
                }
            }
        }


        //Consultation des matériels

        public static List<Materiel> GetTousLesMateriels()
        {
            List<Materiel> lesMateriels = new List<Materiel>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idM, type_materiel, date_Achat, date_location,
garanti, idU
                FROM materiel
                ORDER BY idM;";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
```

```
                    {
                        while (reader.Read())
                        {
                            string idM = reader.GetString("idM");
// string
                            string type =
reader.GetString("type_materiel");      // string

                            DateTime? dateAchat =
reader.IsDBNull(reader.GetOrdinal("date_Achat"))
                                ? (DateTime?)null
                                : reader.GetDateTime("date_Achat");
// DateTime?

                            DateTime? dateLoc =
reader.IsDBNull(reader.GetOrdinal("date_location"))
                                ? (DateTime?)null
                                : reader.GetDateTime("date_location");
// DateTime?

                            string garanti =
reader.GetString("garanti");          // string
                            int idU = reader.GetInt32("idU");
// int

                            Materiel m = new Materiel(idM, type,
dateAchat, dateLoc, garanti, idU);
                            lesMateriels.Add(m);
                        }
                    }
                }
            }

            return lesMateriels;
        }

        //Consultation des incidents / tickets
        public static List<Ticket>
GetIncidentsPourTechnicien(Technicien tech)
        {
            List<Ticket> lesTickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
```

```
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // Tech niveau 1 : tickets niveau 1 + même région
(de l'utilisateur)
                    if (tech.getNiv_intervention() == "Niveau 1")
                    {
                        cmd.CommandText = @"
                    SELECT t.id_ticket, t.objet_demande, t.niv_urgence
                    FROM ticket t, utilisateur u
                    WHERE t.idU = u.idU
                      AND u.regionU = @region
                      AND t.niv_urgence = 'Niveau 1'
                    ORDER BY t.date_ticket DESC; ";

                        cmd.Parameters.AddWithValue("@region",
tech.getRegionT());
                    }
                    else
                    {
                        // Tech niveau > 1 : tous les tickets
                        cmd.CommandText = @"
                    SELECT id_ticket, objet_demande, niv_urgence,
etat_demande, date_ticket, idM, idT, idU,  id_phase
                    FROM ticket
                    ORDER BY date_ticket DESC;";
                    }

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string nivUrgence =
reader.GetString("niv_urgence");
```

```
                            string etat =
reader.GetString("etat_demande");
                            string dateTicket =
reader.GetString("date_ticket");
                            int idM = reader.GetInt32("idM");

                            // Si dans ta table idT peut être NULL, on
sécurise :
                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                            ? (int?)null
                            : reader.GetInt32("idT");

                            int idU = reader.GetInt32("idU");
                            int id_phase = reader.GetInt32("id_phase");

                            Ticket t = new Ticket(idTicket, objet,
nivUrgence, etat, dateTicket, idM, idT, idU, id_phase);
                            lesTickets.Add(t);
                        }
                    }
                }
            }

            return lesTickets;
        }

        //suivi des incidents déclaré par les utilisateurs
        public static List<Ticket> GetIncidentsPourUtilisateur(int
idUtilisateur)
        {
            List<Ticket> lesTickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
```

```
                        SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                               date_ticket, idM, idT, idU,
                               0 AS id_phase
                        FROM ticket
                        WHERE idU = @idU
                        ORDER BY date_ticket DESC;
                    ";


                    cmd.Parameters.AddWithValue("@idU", idUtilisateur);

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string nivUrgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string dateTicket =
reader.GetString("date_ticket");
                            int idM = 0; // valeur par défaut si NULL /
vide / mauvais format
                            object rawIdM = reader["idM"];

                            if (rawIdM != DBNull.Value &&
int.TryParse(rawIdM.ToString(), out int parsedIdM))
                                idM = parsedIdM;

                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                                ? (int?)null
                                : reader.GetInt32("idT");

                            int idU = reader.GetInt32("idU");
                            int id_phase = reader.GetInt32("id_phase");
```

```
                            Ticket t = new Ticket(
                                idTicket, objet, nivUrgence, etat,
                                dateTicket, idM, idT, idU, id_phase);

                            lesTickets.Add(t);
                        }
                    }
                }
            }

            return lesTickets;
        }

        //prise en charge d'un incident par un technicien

        public static bool PrendreEnChargeIncident(int idTicket, int
idTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET idT = @idT,
                    etat_demande = 'En cours de traitement'
                WHERE id_ticket = @idTicket
                  AND idT IS NULL;
            ";

                    cmd.Parameters.AddWithValue("@idT", idTechnicien);
                    cmd.Parameters.AddWithValue("@idTicket", idTicket);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }
```

```
        //Passer le ticket en version résolu et enregistrement du
travail

        public static bool EnregistrerTravailEtResoudre(int idTicket,
Technicien tech, Phase_ticket phaseFinale)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (var tx = conn.BeginTransaction())
                {
                    try
                    {
                        // 1) Insert phase
                        using (MySqlCommand cmdPhase =
conn.CreateCommand())
                        {
                            cmdPhase.Transaction = tx;
                            cmdPhase.CommandText = @"
                        INSERT INTO phase_ticket
                        (id_ticket, idT, date_phase, heure_debut,
heure_fin, description_travail)
                        VALUES
                        (@idTicket, @idT, @datePhase, @heureDebut,
@heureFin, @description);
                    ";


cmdPhase.Parameters.AddWithValue("@idTicket",
phaseFinale.getIdTicket());
                            cmdPhase.Parameters.AddWithValue("@idT",
phaseFinale.getIdTechnicien());

cmdPhase.Parameters.AddWithValue("@datePhase",
phaseFinale.getDatePhase());

cmdPhase.Parameters.AddWithValue("@heureDebut",
phaseFinale.getHeureDebut());
```

```
cmdPhase.Parameters.AddWithValue("@heureFin",
phaseFinale.getHeureFin());

cmdPhase.Parameters.AddWithValue("@description",
phaseFinale.getDescriptionTravail());

                            cmdPhase.ExecuteNonQuery();
                        }

                        // 2) Update ticket => Résolue
                        using (MySqlCommand cmdTicket =
conn.CreateCommand())
                        {
                            cmdTicket.Transaction = tx;
                            cmdTicket.CommandText = @"
                        UPDATE ticket
                        SET etat_demande = 'Résolue'
                        WHERE id_ticket = @idTicket
                          AND idT = @idT
                          AND etat_demande = 'En cours de traitement';
                    ";


cmdTicket.Parameters.AddWithValue("@idTicket", idTicket);
                            cmdTicket.Parameters.AddWithValue("@idT",
tech.getIDTech());

                            int rows = cmdTicket.ExecuteNonQuery();
                            if (rows != 1)
                            {
                                tx.Rollback();
                                return false;
                            }
                        }

                        tx.Commit();
                        return true;
                    }
                    catch
                    {
                        tx.Rollback();
                        throw;
```

```
                    }
                }
            }
        }

        //Nombre total d'incidents
        public static int GetNombreTotalIncidents()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT COUNT(*) FROM ticket;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        //Nombre d'incidents par état
        public static int GetNombreIncidentsParEtat(string etat)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT COUNT(*)
                FROM ticket
                WHERE etat_demande = @etat;";
                    cmd.Parameters.AddWithValue("@etat", etat);
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        //Nombre de jours moyen d'intervention
        public static double GetDureeMoyenneInterventionEnMinutes()
```

```
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT AVG(TIMESTAMPDIFF(MINUTE, heure_debut,
heure_fin))
                FROM phase_ticket pt, ticket t
                WHERE pt.id_ticket = t.id_ticket
                AND t.etat_demande = 'Résolue';
            ";

                    object result = cmd.ExecuteScalar();
                    if (result == DBNull.Value)
                        return 0;

                    return Convert.ToDouble(result);
                }
            }
        }

        //statistique sur les techniciens
        public static List<StatTechnicien> GetStatsTechniciens()
        {
            List<StatTechnicien> stats = new List<StatTechnicien>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // IMPORTANT :
                    // - LEFT JOIN ticket : pour garder les techniciens
même s'ils n'ont aucun ticket
                    // - LEFT JOIN phase_ticket : pour calculer des
durées (si aucune phase => NULL => COALESCE)
```

```
                    // - Le DISTINCT dans les comptages évite de
compter plusieurs fois un ticket à cause des phases
                    cmd.CommandText = @"
                    SELECT
                        te.idT,
                        te.nomT,
                        te.prenomT,

                        COALESCE(COUNT(ti.id_ticket), 0) AS
nbTotalAssignes,

                        COALESCE(SUM(CASE WHEN ti.etat_demande = 'En
cours de traitement' THEN 1 ELSE 0 END), 0) AS nbEnCharge,
                        COALESCE(SUM(CASE WHEN ti.etat_demande =
'Résolue' THEN 1 ELSE 0 END), 0) AS nbResolus,
                        COALESCE(SUM(CASE WHEN ti.etat_demande =
'Clôturée' THEN 1 ELSE 0 END), 0) AS nbClotures

                    FROM technicien te
                    LEFT JOIN ticket ti ON ti.idT = te.idT
                    GROUP BY te.idT, te.nomT, te.prenomT
                    ORDER BY te.idT;";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idT = reader.GetInt32("idT");
                            string nom = reader.GetString("nomT");
                            string prenom =
reader.GetString("prenomT");

                            int nbEnCharge =
Convert.ToInt32(reader["nbEnCharge"]);
                            int nbResolus =
Convert.ToInt32(reader["nbResolus"]);
                            int nbClotures =
Convert.ToInt32(reader["nbClotures"]);
                            int nbTotalAssignes =
Convert.ToInt32(reader["nbTotalAssignes"]);

                            double dureeMoy = 0;
```

```
                            double dureeTotale = 0;

                            stats.Add(new StatTechnicien(
                                idT, nom, prenom,
                                nbEnCharge, nbResolus, nbClotures,
nbTotalAssignes,
                                dureeMoy, dureeTotale

                            ));
                        }
                    }
                }
            }
            return stats;
        }

        //Visualisation des statistique sur les utilisateurs

        public static List<StatUtilisateur> GetStatsUtilisateurs()
        {
            List<StatUtilisateur> stats = new List<StatUtilisateur>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT
                    u.idU,
                    u.nomU,
                    u.prenomU,

                    COUNT(t.id_ticket) AS nbDeclares,
                    COALESCE(SUM(CASE WHEN t.etat_demande = 'En cours
de traitement' THEN 1 ELSE 0 END), 0) AS nbEnCours,
                    COALESCE(SUM(CASE WHEN t.etat_demande = 'Résolue'
THEN 1 ELSE 0 END), 0) AS nbResolus

                FROM utilisateur u
```

```
                LEFT JOIN ticket t ON t.idU = u.idU
                GROUP BY u.idU, u.nomU, u.prenomU
                ORDER BY u.idU;
            ";

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idU = reader.GetInt32("idU");
                            string nom = reader.GetString("nomU");
                            string prenom =
reader.GetString("prenomU");

                            int nbDeclares =
Convert.ToInt32(reader["nbDeclares"]);
                            int nbEnCours =
Convert.ToInt32(reader["nbEnCours"]);
                            int nbResolus =
Convert.ToInt32(reader["nbResolus"]);

                            stats.Add(new StatUtilisateur(
                                idU, nom, prenom,
                                nbDeclares, nbEnCours, nbResolus
                            ));
                        }
                    }
                }
            }
            return stats;
        }
        // ====== ID AUTOMATIQUES ======

        public static int GetNextIdResponsable()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
```

```
                    cmd.CommandText = "SELECT IFNULL(MAX(idR), 0) + 1
FROM responsable;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        public static int GetNextIdUtilisateur()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(idU), 0) + 1
FROM utilisateur;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }
        // ====== TECHNICIEN : ID AUTOMATIQUE ======
        public static int GetNextIdTechnicien()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(idT), 0) + 1
FROM technicien;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        // ========Matériel : Id automatique ======
        // enregistrer un matériel

        public static string GetNextIdMateriel()
```

```
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    // idM contient "1", "2", "3" ...
                    cmd.CommandText = "SELECT IFNULL(MAX(CAST(idM AS
UNSIGNED)), 0) + 1 FROM materiel;";
                    int next = Convert.ToInt32(cmd.ExecuteScalar());
                    return next.ToString();
                }
            }
        }


        // ====== TECHNICIEN : retrouver l'id via matricule ======
        public static int? GetIdTechnicienParMatricule(string
matricule)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idT FROM technicien WHERE
matriculeT = @matricule LIMIT 1;";
                    cmd.Parameters.AddWithValue("@matricule",
matricule);

                    object result = cmd.ExecuteScalar();
                    if (result == null || result == DBNull.Value)
return null;

                    return Convert.ToInt32(result);
                }
            }
        }
```

```
        public static Technicien GetTechnicienParId(int idT)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idT, nomT, prenomT, formation, competences,
niv_intervention,
                       matriculeT, adresseT, villeT, codePT,
date_embaucheT, regionT, idR
                FROM technicien
                WHERE idT = @idT;";
                    cmd.Parameters.AddWithValue("@idT", idT);

                    using (var r = cmd.ExecuteReader())
                    {
                        if (!r.Read()) return null;

                        return new Technicien(
                            r.GetInt32("idT"),
                            r.GetString("nomT"),
                            r.GetString("prenomT"),
                            r.GetString("formation"),
                            r.GetString("competences"),
                            r.GetString("niv_intervention"),
                            r.GetString("matriculeT"),
                            r.GetString("adresseT"),
                            r.GetString("villeT"),
                            r.GetString("codePT"),
                            r.GetDateTime("date_embaucheT"),
                            r.GetString("regionT"),
                            r.GetInt32("idR")
                        );
                    }
                }
            }
        }
```

```
        public static int? GetIdUtilisateurParMatricule(string
matricule)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idU FROM utilisateur
WHERE matriculeU = @matricule LIMIT 1;";
                    cmd.Parameters.AddWithValue("@matricule",
matricule);

                    object result = cmd.ExecuteScalar();
                    if (result == null || result == DBNull.Value)
return null;

                    return Convert.ToInt32(result);
                }
            }
        }

        //Modifier ou supprimer un utilisateur

        // ===== UTILISATEUR : charger un utilisateur par id =====
        public static Utilisateur GetUtilisateurParId(int idU)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idU, nomU, prenomU, matriculeU, adresseU,
villeU, codePU, date_embaucheU, regionU, idR
                FROM utilisateur
                WHERE idU = @idU;";
```

```
                    cmd.Parameters.AddWithValue("@idU", idU);

                    using (var r = cmd.ExecuteReader())
                    {
                        if (!r.Read()) return null;

                        return new Utilisateur(
                            r.GetInt32("idU"),
                            r.GetString("nomU"),
                            r.GetString("prenomU"),
                            r.GetString("matriculeU"),
                            r.GetString("adresseU"),
                            r.GetString("villeU"),
                            r.GetString("codePU"),
                            r.GetString("date_embaucheU"),
                            r.GetString("regionU"),
                            r.GetInt32("idR")
                        );
                    }
                }
            }
        }

        // ===== UTILISATEUR : liste simple pour remplir la ComboBox
=====
        public static List<int> GetTousLesIdsUtilisateurs()
        {
            List<int> ids = new List<int>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idU FROM utilisateur
ORDER BY idU;";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetInt32("idU"));
                    }
```

```
                }
            }

            return ids;
        }


        //Liste déroulante des ids des matériel pour les supprimer
        // ===== MATERIEL : liste des IDs pour ComboBox =====
        public static List<string> GetTousLesIdsMateriels()
        {
            List<string> ids = new List<string>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idM FROM materiel ORDER
BY CAST(idM AS UNSIGNED);";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetString("idM"));
                    }
                }
            }

            return ids;
        }



        ///
        ///
        /// selection d'un responsable pour en attribuer un à un
technicien
        ///
        ///
        public static List<int> GetTousLesIdsResponsables()
        {
```

```
            List<int> ids = new List<int>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT idR FROM responsable
ORDER BY idR;";
                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                            ids.Add(r.GetInt32("idR"));
                    }
                }
            }
            return ids;
        }




        //Enregistrer un ticket
        public static int GetNextIdTicket()
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT IFNULL(MAX(id_ticket), 0)
+ 1 FROM ticket;";
                    return Convert.ToInt32(cmd.ExecuteScalar());
                }
            }
        }

        //Consulter les ticket non attribuer
        public static List<Ticket> GetTicketsNonAttribues()
```

```
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();

                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                        SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                               date_ticket, idM, idT, idU
                        FROM ticket
                        WHERE idT IS NULL
                        ORDER BY date_ticket DESC;
                    ";


                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string urgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string date =
reader.GetString("date_ticket");
                            int idM = reader.GetInt32("idM");

                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT"))
                                ? (int?)null
                                : reader.GetInt32("idT");
```

```
                            int idU = reader.GetInt32("idU");

                            tickets.Add(new Ticket(idTicket, objet,
urgence, etat, date, idM, idT, idU, 0));
                        }
                    }
                }
            }

            return tickets;
        }
        //récupération de l'id d'un technicien
        public static List<Technicien> GetTousLesTechniciens()
        {
            List<Technicien> techs = new List<Technicien>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT idT, nomT, prenomT, formation, competences,
niv_intervention,
                       matriculeT, adresseT, villeT, codePT,
date_embaucheT, regionT, idR
                FROM technicien
                ORDER BY idT;";

                    using (var r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            techs.Add(new Technicien(
                                r.GetInt32("idT"),
                                r.GetString("nomT"),
                                r.GetString("prenomT"),
                                r.GetString("formation"),
                                r.GetString("competences"),
                                r.GetString("niv_intervention"),
                                r.GetString("matriculeT"),
```

```
                                r.GetString("adresseT"),
                                r.GetString("villeT"),
                                r.GetString("codePT"),
                                r.GetDateTime("date_embaucheT"),
                                r.GetString("regionT"),
                                r.GetInt32("idR")
                            ));
                        }
                    }
                }
            }

            return techs;
        }

        //Passer le ticke en résolu
        public static bool PasserTicketEnResolu(int idTicket, int
idTechnicien)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET etat_demande = 'Résolue'
                WHERE id_ticket = @idTicket
                  AND idT = @idT
                  AND etat_demande = 'En cours de traitement';
            ";

                    cmd.Parameters.AddWithValue("@idTicket", idTicket);
                    cmd.Parameters.AddWithValue("@idT", idTechnicien);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }
```

```
        //Ne charger (dans la liste des tickets) que les ticket en
cours de traitement

        public static List<Ticket> GetTicketsEnCoursPourTechnicien(int
idTechnicien)
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                       date_ticket, idM, idT, idU
                FROM ticket
                WHERE idT = @idT
                  AND TRIM(etat_demande) LIKE 'En cours%'
                ORDER BY date_ticket DESC;
            ";
                    cmd.Parameters.AddWithValue("@idT", idTechnicien);

                    using (MySqlDataReader reader =
cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            int idTicket =
reader.GetInt32("id_ticket");
                            string objet =
reader.GetString("objet_demande");
                            string urgence =
reader.GetString("niv_urgence");
                            string etat =
reader.GetString("etat_demande");
                            string date =
reader.GetString("date_ticket");
                            int idM = reader.GetInt32("idM");
```

```
                            int? idT =
reader.IsDBNull(reader.GetOrdinal("idT")) ? (int?)null :
reader.GetInt32("idT");
                            int idU = reader.GetInt32("idU");

                            tickets.Add(new Ticket(idTicket, objet,
urgence, etat, date, idM, idT, idU, 0));
                        }
                    }
                }
            }

            return tickets;
        }

        //débug le nom "En cours de traitement
        public static string DebugEtatTicket(int idTicket)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";
            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = "SELECT CONCAT('[', etat_demande,
']') FROM ticket WHERE id_ticket = @id;";
                    cmd.Parameters.AddWithValue("@id", idTicket);
                    object res = cmd.ExecuteScalar();
                    return res == null ? "NULL" : res.ToString();
                }
            }
        }

        //Clôturer un ticket (UNIQUEMENT si Résolu)
        public static bool CloturerTicket(int idTicket)
        {
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
```

```
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                UPDATE ticket
                SET etat_demande = 'Clôturée'
                WHERE id_ticket = @idTicket
                  AND etat_demande = 'Résolue';
            ";

                    cmd.Parameters.AddWithValue("@idTicket", idTicket);

                    return cmd.ExecuteNonQuery() == 1;
                }
            }
        }
        //Charger UNIQUEMENT les tickets résolus (nouvelle ComboBox)
        // ===== Tickets résolus =====
        public static List<Ticket> GetTicketsResolus()
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandText = @"
                SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                       date_ticket, idM, idT, idU
                FROM ticket
                WHERE etat_demande = 'Résolue'
                ORDER BY date_ticket DESC;
            ";

                    using (MySqlDataReader r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            tickets.Add(new Ticket(
                                r.GetInt32("id_ticket"),
```

```
                                r.GetString("objet_demande"),
                                r.GetString("niv_urgence"),
                                r.GetString("etat_demande"),
                                r.GetString("date_ticket"),
                                r.GetInt32("idM"),
                                r.IsDBNull(r.GetOrdinal("idT")) ?
(int?)null : r.GetInt32("idT"),
                                r.GetInt32("idU"),
                                0
                            ));
                        }
                    }
                }
            }
            return tickets;
        }

        //gestion du niveau du technicien (selon lequel est cocher

        public static List<Ticket> GetTicketsVisiblesParNiveau(string
niveauSelectionne, bool estResponsable)
        {
            List<Ticket> tickets = new List<Ticket>();
            string connStr =
"Server=127.0.0.1;Database=projet_c#;Uid=root;Pwd=;CharSet=utf8mb4;";

            using (MySqlConnection conn = new MySqlConnection(connStr))
            {
                conn.Open();
                using (MySqlCommand cmd = conn.CreateCommand())
                {
                    if (estResponsable)
                    {
                        // Responsable : TOUS les tickets (attribués +
non attribués)
                        cmd.CommandText = @"
                    SELECT id_ticket, objet_demande, niv_urgence,
etat_demande,
                           date_ticket, idM, idT, idU
                    FROM ticket
                    ORDER BY date_ticket DESC;";
                    }
                    else
```

```
                    {
                        // Détermine quels niveaux sont visibles
                        string niveauxVisibles;
                        if (niveauSelectionne == "Niveau 1")
niveauxVisibles = "('Niveau 1')";
                        else if (niveauSelectionne == "Niveau 2")
niveauxVisibles = "('Niveau 1','Niveau 2')";
                        else niveauxVisibles = "('Niveau 1','Niveau
2','Niveau 3')"; // Niveau 3

                        // Ici : on affiche les tickets attribués à un
technicien dont le niveau est dans la liste
                        cmd.CommandText = $@"
                    SELECT t.id_ticket, t.objet_demande, t.niv_urgence,
t.etat_demande,
                           t.date_ticket, t.idM, t.idT, t.idU
                    FROM ticket t
                    INNER JOIN technicien te ON te.idT = t.idT
                    WHERE te.niv_intervention IN {niveauxVisibles}
                    ORDER BY t.date_ticket DESC;";
                    }

                    using (MySqlDataReader r = cmd.ExecuteReader())
                    {
                        while (r.Read())
                        {
                            int idTicket =
Convert.ToInt32(r["id_ticket"]);
                            string objet =
r["objet_demande"]?.ToString();
                            string urgence =
r["niv_urgence"]?.ToString();
                            string etat =
r["etat_demande"]?.ToString();
                            string date = r["date_ticket"]?.ToString();

                            int idM = SafeToNullableInt(r["idM"]) ?? 0;
                            int? idT = SafeToNullableInt(r["idT"]);
                            int idU = SafeToNullableInt(r["idU"]) ?? 0;

                            // Si tu n'as pas id_phase dans la requête,
laisse 0
                            int idPhase = 0;
```

```
                            tickets.Add(new Ticket(
                                    idTicket,
                                    objet,
                                    urgence,
                                    etat,
                                    date,
                                    idM,
                                    idT,
                                    idU,
                                    idPhase
                                    ));
                        }
                    }
                }
            }

            return tickets;
        }

        private static int? SafeToNullableInt(object value)
        {
            if (value == null || value == DBNull.Value) return null;

            // si c'est déjà un int/long, on convertit proprement
            if (value is int i) return i;
            if (value is long l) return (int)l;

            string s = value.ToString().Trim();

            // gère chaînes vides ou "null"
            if (string.IsNullOrWhiteSpace(s) || s.Equals("null",
StringComparison.OrdinalIgnoreCase))
                return null;

            // tentative de conversion
            if (int.TryParse(s, out int result))
                return result;

            return null; // si c'est pas un nombre, on retourne null au
lieu de planter
        }
```

```
        }
}
```

Class Utilisateur

Class Utilisateur

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Utilisateur
    {
        private int idu;
        private string nomu;
        private string prenomu;
        private string matriculeu;
        private string adresseu;
        private string villeu;
        private string codepu;
        private string date_embaucheu;
        private string regionu;
        private int idr;


        public Utilisateur(int UnIdu, string UnNomu, string UnPrenomu,
string UnMatriculeu, string UneAdresseu, string UneVilleu, string
UnCodepu, string UneDate_embauche, string UneRegionu, int UnIdr)
        {
            idu = UnIdu;
            nomu = UnNomu;
            prenomu = UnPrenomu;
            matriculeu = UnMatriculeu;
            adresseu = UneAdresseu;
            villeu = UneVilleu;
            codepu = UnCodepu;
            date_embaucheu = UneDate_embauche;
            regionu = UneRegionu;
            idr = UnIdr;
        }
        public int getIDU()
        {
            return idu;
        }
        public string getNomu()
```

```
        {
            return nomu;
        }

        public string getPrenomu()
        {
            return prenomu;
        }

        public string getMatriculeu()
        {
            return matriculeu;
        }

        public string getAdresseu()
        {
            return adresseu;
        }

        public string getVilleu()
        {
            return villeu;
        }

        public string getCodepu()
        {
            return codepu;
        }

        public string getDate_embaucheu()
        {
            return date_embaucheu;
        }

        public string getRegionu()
        {
            return regionu;
        }

        public int getIdr()
        {
            return idr;
        }
```

```
        public void setRegionu(string newregion)
        {
            regionu = newregion;
        }
    }
}
```

Class Materiel

Class Materiel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Materiel
    {
        private string idm;
        private string type_materiel;
        private DateTime? date_achat;//datetime ? --> veut dire qu'il
peut être nullable
        private DateTime? date_location;//datetime ? --> veut dire
qu'il peut être nullable
        private string garanti;
        private int idu;



        public Materiel(string UnIdm, string UnType_materiel, DateTime?
UneDate_achat, DateTime? Unedate_location, string UneGaranti, int
UnIdu)
        {
            idm = UnIdm;
            type_materiel = UnType_materiel;
            date_achat = UneDate_achat;
            date_location = Unedate_location;
            garanti = UneGaranti;
            idu = UnIdu;
        }
        public string getIDM()
        {
            return idm;
        }
        public string getType_materiel()
        {
            return type_materiel;
        }
        public DateTime? getDate_achat()
        {
```

```
            return date_achat;
        }
        public DateTime? getDate_location()
        {
            return date_location;
        }
        public string getGaranti()
        {
            return garanti;
        }

        public int getIDU()
        {
            return idu;
        }
        public override string ToString()
        {
            return $"{idm} - {type_materiel} - {garanti}";
        }

    }
}
```

Class Phase ticket

Phase ticket

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Phase_ticket
    {
        private int idPhase;
        private int idTicket;
        private int idTechnicien;

        private DateTime datePhase;
        private TimeSpan heureDebut;
        private TimeSpan heureFin;

        private string descriptionTravail;

        public Phase_ticket(int unIdTicket, int unIdTech, DateTime
uneDate,
                            TimeSpan debut, TimeSpan fin, string
description)
        {
            idTicket = unIdTicket;
            idTechnicien = unIdTech;
            datePhase = uneDate;
            heureDebut = debut;
            heureFin = fin;
            descriptionTravail = description;
        }

        public int getIdTicket()
        {
            return idTicket;
        }

        public int getIdTechnicien()
        {
            return idTechnicien;
        }
```

```
        public DateTime getDatePhase()
        {
            return datePhase;
        }

        public TimeSpan getHeureDebut()
        {
            return heureDebut;
        }
        public TimeSpan getHeureFin()
        {
            return heureFin;
        }

        public string getDescriptionTravail()
        {
            return descriptionTravail;
        }

    }
}
```

Class Responsable

Class Responsable

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Responsable
    {
        private int idr;
        private string nomr;
        private string prenomr;
        private string matriculer;
        private string adresser;
        private string viller;
        private string codepr;
        private string date_embaucher;
        private string regionr;


        public Responsable(int UnIdr, string UnNomr, string UnPrenomr,
string UnMatriculer, string UneAdresser, string UneViller, string
UnCodepr, string UneDate_embaucher, string UneRegionr)
        {
            idr = UnIdr;
            nomr = UnNomr;
            prenomr = UnPrenomr;
            matriculer = UnMatriculer;
            adresser = UneAdresser;
            viller = UneViller;
            codepr = UnCodepr;
            date_embaucher = UneDate_embaucher;
            regionr = UneRegionr;
        }

        public int getIDR()
        {
            return idr;
        }
        public string getNomr()
        {
```

```
            return nomr;
        }
        public string getPrenomr()
        {
            return prenomr;
        }
        public string getDate_embaucher()
        {
            return date_embaucher;
        }
        public string getMatriculer()
        {
            return matriculer;
        }
        public string getAdresser()
        {
            return adresser;
        }
        public string getViller()
        {
            return viller;
        }
        public string getCodepr()
        {
            return codepr;
        }
        public string getRegionr()
        {
            return regionr;
        }
        public void setRegionr(string newregionr)
        {
            regionr = newregionr;
        }
    }
}
```

Classe Session

Classe Session

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Session
    {
        public static Technicien TechnicienConnecte { get; set; }
        public static Utilisateur UtilisateurConnecte { get; set; }
    }
}
```

Class Stats technicien

Class Stats technicien

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class StatTechnicien
    {
        private int idT;
        private string nom;
        private string prenom;

        private int nbEnCharge;   // "En cours de traitement"
        private int nbResolus;    // "Résolue"
        private int nbClotures;   // "Clôturée" (si tu l'utilises)
        private int nbTotalAssignes;

        private double dureeMoyMinutes;   // moyenne des phases sur ses
tickets
        private double dureeTotaleMinutes; // somme des phases sur ses
tickets

        public StatTechnicien(int UnIdT, string UnNom, string UnPrenom,
                              int UnNbEnCharge, int UnNbResolus, int
UnNbClotures, int UnNbTotalAssignes,
                              double UneDureeMoyMinutes, double
UneDureeTotaleMinutes)
        {
            idT = UnIdT;
            nom = UnNom;
            prenom = UnPrenom;
            nbEnCharge = UnNbEnCharge;
            nbResolus = UnNbResolus;
            nbClotures = UnNbClotures;
            nbTotalAssignes = UnNbTotalAssignes;
            dureeMoyMinutes = UneDureeMoyMinutes;
            dureeTotaleMinutes = UneDureeTotaleMinutes;
        }

        // ===== GETTERS (même écriture que dans Phase_ticket) =====
```

```
        public int getIdT()
        {
            return idT;
        }

        public string getNomT()
        {
            return nom;
        }

        public string getPrenomT()
        {
            return prenom;
        }

        public int getNbEnCharge()
        {
            return nbEnCharge;
        }

        public int getNbResolus()
        {
            return nbResolus;
        }

        public int getNbClotures()
        {
            return nbClotures;
        }

        public int getNbTotalAssignes()
        {
            return nbTotalAssignes;
        }

        public double getDureeMoyMinutes()
        {
            return dureeMoyMinutes;
        }

        public double getDureeTotaleMinutes()
        {
```

```
            return dureeTotaleMinutes;
        }
        // ===== SETTERS (optionnels mais cohérents avec ton projet)
=====

        public void setIdT(int unIdT)
        {
            idT = unIdT;
        }

        public void setNomT(string unNomT)
        {
            nom = unNomT;
        }

        public void setPrenomT(string unPrenomT)
        {
            prenom = unPrenomT;
        }

        public void setNbEnCharge(int unNbEnCharge)
        {
            nbEnCharge = unNbEnCharge;
        }

        public void setNbResolus(int unNbResolus)
        {
            nbResolus = unNbResolus;
        }

        public void setNbClotures(int unNbClotures)
        {
            nbClotures = unNbClotures;
        }

        public void setNbTotalAssignes(int unNbTotalAssignes)
        {
            nbTotalAssignes = unNbTotalAssignes;
        }

        public void setDureeMoyMinutes(double uneDureeMoyMinutes)
        {
            dureeMoyMinutes = uneDureeMoyMinutes;
```

```
        }
        public void setDureeTotaleMinutes(double uneDureeTotaleMinutes)
        {
            dureeTotaleMinutes = uneDureeTotaleMinutes;
        }
    }
}
```

Classe Stats Utilisateurs

Classe Stats Utilisateurs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class StatUtilisateur
    {

        private int idU;
        private string nom;
        private string prenom;

        private int nbIncidentsDeclares;
        private int nbEnCours;
        private int nbResolus;

        public StatUtilisateur(int unIdU, string unNom, string
unPrenom,
                               int unNbDeclares, int unNbEnCours, int
unNbResolus)
        {
            idU = unIdU;
            nom = unNom;
            prenom = unPrenom;
            nbIncidentsDeclares = unNbDeclares;
            nbEnCours = unNbEnCours;
            nbResolus = unNbResolus;
        }

        // ===== GETTERS (même écriture que tes autres fichiers) =====

        public int getIDU()
        {
            return idU;
        }

        public string getNomU()
        {
            return nom;
```

```
        }

        public string getPrenomU()
        {
            return prenom;
        }

        public int getNbIncidentsDeclares()
        {
            return nbIncidentsDeclares;
        }

        public int getNbEnCours()
        {
            return nbEnCours;
        }

        public int getNbResolus()
        {
            return nbResolus;
        }


        // ===== SETTERS (si tu fais pareil ailleurs) =====

        public void setIDU(int newIdU)
        {
            idU = newIdU;
        }

        public void setNomU(string newNomU)
        {
            nom = newNomU;
        }

        public void setPrenomU(string newPrenomU)
        {
            prenom = newPrenomU;
        }

        public void setNbIncidentsDeclares(int newNbDeclares)
        {
            nbIncidentsDeclares = newNbDeclares;
```

```
        }

        public void setNbEnCours(int newNbEnCours)
        {
            nbEnCours = newNbEnCours;
        }

        public void setNbResolus(int newNbResolus)
        {
            nbResolus = newNbResolus;
        }
    }
}
```

Class Technicien

Class Technicien

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projet_final_C_
{
    internal class Technicien
    {
        private int idT;
        private string nomT;
        private string prenomT;
        private string formation;
        private string competences;
        private string niv_intervention;
        private string matriculeT;
        private string adresseT;
        private string villeT;
        private string codePT;
        private DateTime date_embaucheT;
        private string regionT;
        private int idR;
        //private int id_phase;

        public Technicien(int UnIdt, string UnNomt, string UnPrenomT,
string UneFormation, string UneCompetence, string UnNiv_intervention,
string UnMatricule, string UneAdresse, string UneVille, string
UnCodePT, DateTime UneDate_embaucheT, string UneRegion, int UnIdR)
        {
            idT = UnIdt;
            nomT = UnNomt;
            prenomT = UnPrenomT;
            formation = UneFormation;
            competences = UneCompetence;
            niv_intervention = UnNiv_intervention;
            matriculeT = UnMatricule;
            adresseT = UneAdresse;
            villeT = UneVille;
            codePT = UnCodePT;
```

```
            date_embaucheT = UneDate_embaucheT;
            regionT = UneRegion;
            idR = UnIdR;
            //id_phase = UnId_phase;

        }
        public int getIDTech()
        {
            return idT;
        }
        public string getNomt()
        {
            return nomT;
        }

        public string getPrenomT()
        {
            return prenomT;
        }

        public string getMatriculeT()
        {
            return matriculeT;
        }

        public string getAdresseT()
        {
            return adresseT;
        }

        public string getVilleT()
        {
            return villeT;
        }

        public string getCodePT()
        {
            return codePT;
        }

        public DateTime getDate_embaucheT()
        {
            return date_embaucheT;
```

```
        }

        public string getRegionT()
        {
            return regionT;
        }

        public string getFormation()
        {
            return formation;
        }

        public string getCompetences()
        {
            return competences;
        }

        public string getNiv_intervention()
        {
            return niv_intervention;
        }
        public int getIdr()
        {
            return idR;
        }
        //public int getIDPhase()
        //{
        //    return id_phase;
        //}


        public void setRegionT (string newregion)
        {
            regionT = newregion;
        }
        public void setFormationT(string newformation)
        {
            formation = newformation;
        }
        public override string ToString()
        {
            return $"{idT} - {nomT} {prenomT} ({niv_intervention})";
        }
```

```
    }

}
```

Class Ticket

Class Ticket

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet_final_C_
{
    internal class Ticket
    {
        private int id_ticket;
        private string objet_demande;
        private string niv_urgence;
        private string etat_demande;
        private string date_ticket;
        private int idM;
        private int? idT;
        private int idU;
        private int id_phase;

        public Ticket(int UnId_ticket, string UnObjet_demande, string
UnNiveau_urgence, string UnEtat_demande, string UneDate_ticket, int
UnIdM, int? UnIdT, int UnIdU, int UnId_phase)
        {
            id_ticket = UnId_ticket;
            objet_demande = UnObjet_demande;
            niv_urgence = UnNiveau_urgence;
            etat_demande = UnEtat_demande;
            date_ticket = UneDate_ticket;
            idM = UnIdM;
            idT = UnIdT;
            idU = UnIdU;
            id_phase = UnId_phase;
        }
        public int getIDTicket()
        {
            return id_ticket;
        }
        public string getObjet_demande()
        {
            return objet_demande;
        }
```

```
        public string getNiv_urgence()
        {
            return niv_urgence;
        }

        public string getEtat_demande()
        {
            return etat_demande;
        }
        public string getDate_ticket()
        {
            return date_ticket;
        }

        public int getIDM()
        {
            return idM;
        }

        public int? getIDTech()
        {
            return idT;
        }

        public int getIDU()
        {
            return idU;
        }
        public int getIDPhase()
        {
            return id_phase;
        }
        public void setNouvelle_Objet_demande(string Nouvelle_demande)
        {
            objet_demande = Nouvelle_demande;
        }

        // (optionnel) setter pour affecter un technicien après
création
        public void setIDTech(int? nouvelIdT)
        {
            idT = nouvelIdT;
        }
```

```
        public override string ToString()
        {
            return $"#{id_ticket} - {objet_demande} ({niv_urgence}) -
{etat_demande}";
        }









    }
}
```

Onglet 12

Bonsoir, voici le document avec les code en copier coller, j’esper qu’il vous conviendra.
Cordialement
Adrien Bernady et Valentine pincho