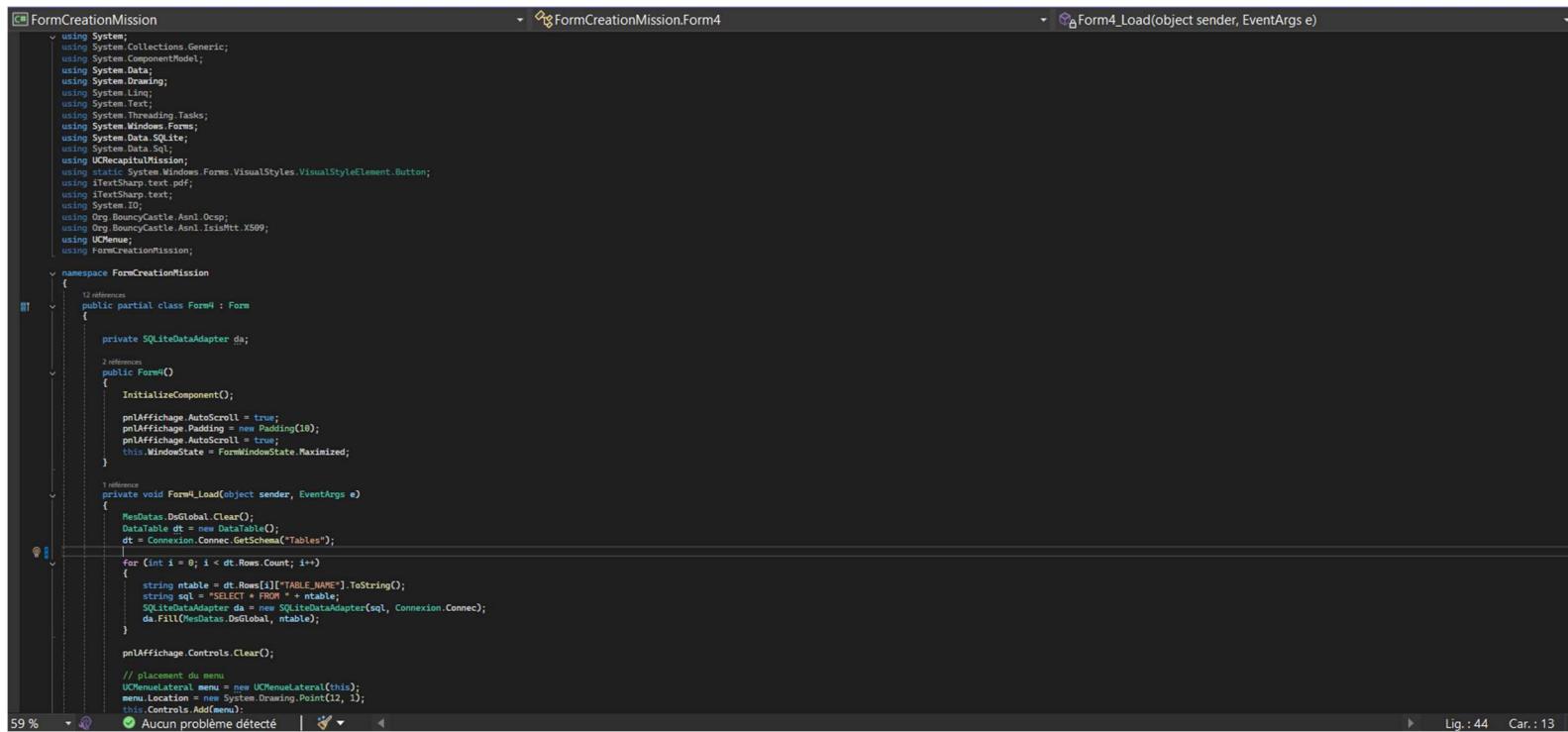


## Resume du code Saé A21-D21 :

### 1) La declaration des variables globales du formulaire de départ :



```
FormCreationMission
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using System.Data.SQLite;
    using System.Data.SqlClient;
    using UCRecapMission;
    using System.Windows.Forms.VisualStyles.VisualStyleElement.Button;
    using iTextSharp.text;
    using System.IO;
    using Org.BouncyCastle.Asn1.Ocsp;
    using Org.BouncyCastle.Asn1.X509;
    using UCNew;
    using FormCreationMission;

namespace FormCreationMission
{
    public partial class Form4 : Form
    {
        private SQLiteDataAdapter da;
        public Form4()
        {
            InitializeComponent();
            pnAffichage.AutoScroll = true;
            pnAffichage.Padding = new Padding(10);
            pnAffichage.AutoScroll = true;
            this.WindowState = FormWindowState.Maximized;
        }

        private void Form4_Load(object sender, EventArgs e)
        {
            MesDatas.DsGlobal.Clear();
            DataTable dt = new DataTable();
            dt = Connexion.Connec.GetSchema("Tables");
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                string ntable = dt.Rows[i]["TABLE_NAME"].ToString();
                string sql = "SELECT * FROM " + ntable;
                SQLiteDataAdapter da = new SQLiteDataAdapter(sql, Connexion.Connec);
                da.Fill(MesDatas.DsGlobal, ntable);
            }
            pnAffichage.Controls.Clear();
            // placement du menu
            UCMenuLateral menu = new UCMenuLateral(this);
            menu.Location = new System.Drawing.Point(12, 1);
            this.Controls.Add(menu);
        }
    }
}
```

### 2) Le code complet du formulaire (ou UserControl) dans lequel vous permettez le choix d'un pompier, puis affichez les informations complètes concernant un pompier

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Drawing;
    using System.Data;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using System.Data.SQLite;
    using System.IO;

    namespace FormCreationMission
    {
        5 références
        public partial class UCPersonnel : UserControl
        {
            Form4 f;
            private bool chargementTermine = false;
            private bool estAdmin;
            1 référence
            public UCPersonnel(Form4 x)
            {
                InitializeComponent();
                f = x;
            }

            1 référence
            private void UCPersonnel_Load(object sender, EventArgs e)
            {
                estAdmin = false; // Indique si l'utilisateur est admin
                gbInformationCarriere.Visible = false; // Masquer le groupe d'informations de carrière au départ
                cbGradeNouveau.Visible = false;
                if (Connexion.Connec.State != ConnectionState.Open)
                    Connexion.Connec.Open();

                string sqlCaserne = "SELECT id, nom FROM Caserne";
                using (SQLiteCommand cmd = new SQLiteCommand(sqlCaserne, Connexion.Connec))
                using (SQLiteDataReader reader = cmd.ExecuteReader())
                {
                    DataTable dtCaserne = new DataTable();
                    dtCaserne.Load(reader);
                    cbCaserne.DataSource = dtCaserne;
                    cbCaserne.DisplayMember = "nom";
                    cbCaserne.ValueMember = "id";
                }

                chargementTermine = true; //Autorise l'événement maintenant
                cbCaserne_SelectedIndexChanged(null, null); //Forcer un premier chargement des pompiers
            }

            2 références
            private void cbCaserne_SelectedIndexChanged(object sender, EventArgs e)
            {

                if (!chargementTermine || cbCaserne.SelectedValue == null)
                    return;

                try
                {
                    cbPompiers.DataSource = null;

```

%



Aucun problème détecté



```
C# FormCreationMission
cbPompiers.DataSource = null;
// Vérifie que l'ID caserne est bien un entier
if (!int.TryParse(cbCaserne.SelectedValue.ToString(), out int idCaserne))
{
    MessageBox.Show("Erreur : ID caserne invalide.");
    return;
}

// Ouvre la connexion si nécessaire
if (Connexion.Connec.State != ConnectionState.Open)
    Connexion.Connec.Open();

// Requête SQL pour récupérer les pompiers de la caserne
string sqlPompier = @"SELECT P.matricule, P.nom || ' ' || P.prenom AS nomComplet FROM Pompier P JOIN Affectation
A ON P.matricule = A.matriculePompier WHERE A.idCaserne = @id"; // Le || ' ' || c'est pour la concaténation

using (SQLiteCommand cmd = new SQLiteCommand(sqlPompier, Connexion.Connec))
{
    cmd.Parameters.AddWithValue("@id", idCaserne); // On ajoute la valeur de @id dans la requête

    using (SQLiteDataReader reader = cmd.ExecuteReader())
    {
        DataTable dtPompiers = new DataTable();
        dtPompiers.Load(reader);
        // Remplir la ComboBox des pompiers
        cbPompiers.DataSource = dtPompiers;
        cbPompiers.DisplayMember = "nomComplet";
        cbPompiers.ValueMember = "matricule";
    }
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur lors du chargement des pompiers : " + ex.Message);
}

private void cbPompiers_SelectedIndexChanged_1(object sender, EventArgs e)
{
    try
    {
        if (cbPompiers.SelectedItem is DataRowView row)
        {
            if (gbInformationCarriere.Visible)
                gbInformationCarriere.Visible = false; // Masquer le groupe d'informations de carrière si visible
            int matricule = Convert.ToInt32(row["matricule"]);

            if (Connexion.Connec.State != ConnectionState.Open)
                Connexion.Connec.Open();

            string sql = @"
SELECT P.nom, P.prenom, P.sex, P.dateNaissance, P.type,
       P.portable, P.bip, P.dateEmbauche, P.codeGrade, P.enConge,
       G.libelle AS gradeLibelle
FROM Pompier P
LEFT JOIN Grade G ON P.codeGrade = G.code
WHERE P.matricule = @mat";

            using (SQLiteCommand cmd = new SQLiteCommand(sql, Connexion.Connec))
            {
                cmd.Parameters.AddWithValue("@mat", matricule);
            }
        }
    }
}
```

C# FormCreationMission

```
if (estAdmin)
{
    // Si l'utilisateur est admin, on ouvre directement le formulaire de création
    Form3 formCreer = new Form3();
    formCreer.ShowDialog();
}
else
{
    //Demander si il est admin ou pas
    FormConnexion formConnexion = new FormConnexion();
    formConnexion.ShowDialog();

    if (formConnexion.EstConnecte)
    {
        estAdmin = true; // L'utilisateur est maintenant admin
        // Connexion réussie ouvre le formulaire de création du pompier
        Form3 formCreer = new Form3();
        formCreer.ShowDialog();
    }
}

1 référence
private void btnChanger_Click_1(object sender, EventArgs e)
{
    if(estAdmin)
    {
        try
        {
            if (cbGradeNouveau.Visible)
            {
                //Déjà visible on copie le texte + image du grade
                if (cbGradeNouveau.SelectedItem is DataRowView selectedRow)
                {
                    string libelleGrade = selectedRow["Libelle"].ToString();
                    string codeGrade = selectedRow["code"].ToString();
                    int matricule = Convert.ToInt32(lblMatricule.Text); // récupère le matricule

                    //Mise à jour du grade dans la base avec transaction
                    using (SQLiteTransaction transaction = Connexion.Connec.BeginTransaction())
                    {
                        try
                        {
                            using (SQLiteCommand cmd = Connexion.Connec.CreateCommand())
                            {
                                cmd.Transaction = transaction;
                                cmd.CommandText = "UPDATE Pompier SET codeGrade = @codeGrade WHERE matricule = @matricule";
                                cmd.Parameters.AddWithValue("@codeGrade", codeGrade);
                                cmd.Parameters.AddWithValue("@matricule", matricule);
                                cmd.ExecuteNonQuery();
                            }

                            transaction.Commit();

                            //Mise à jour visuelle
                            txtGrade.Text = libelleGrade;
                        }
                    }
                }
            }
        }
    }
}
```

59 % Aucun problème détecté

## C# FormCreationMission

## FormCreationMission.UCPersonne

```
        string imagePath = Path.Combine(@"C:\OMAR\S2\SAE-D21\ImagesGrades", codeGrade + ".png");
        if (File.Exists(imagePath))
        {
            pbGrade.Image = Image.FromFile(imagePath);
            pbGrade.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        else
        {
            pbGrade.Image = null;
        }
    }
    catch (Exception ex)
    {
        transaction.Rollback();
        MessageBox.Show("X Erreur pendant la mise à jour du grade : " + ex.Message);
    }
}
cbGradeNouveau.Visible = false; // cacher après sélection
}
else
{
    //Première fois on affiche et on remplit la combobox
    if (Connexion.Connec.State != ConnectionState.Open)
        Connexion.Connec.Open();

    string sqlGrade = "SELECT code, libelle FROM Grade";
    using (SQLiteCommand cmdGrade = new SQLiteCommand(sqlGrade, Connexion.Connec))
    using (SQLiteDataReader reader = cmdGrade.ExecuteReader())
    {
        DataTable dtGrades = new DataTable();
        dtGrades.Load(reader);
        cbGradeNouveau.DataSource = dtGrades;
        cbGradeNouveau.DisplayMember = "libelle";
        cbGradeNouveau.ValueMember = "code";
    }
    cbGradeNouveau.Visible = true;
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur : " + ex.Message);
}
//Si c'est pas un admin on verifie
else
{
    FormConn formConnexion = new FormConn();
    formConnexion.ShowDialog();
    if (formConnexion.EstConnecte)
    {
        estAdmin = true;
        // L'utilisateur est maintenant admin
        try
        {
            if (cbGradeNouveau.Visible)
            {
                // Déjà visible = on copie le texte + image du grade
                if (cbGradeNouveau.SelectedItem is DataRowView selectedRow)
                {

```

59 %



Aucun problème détecté



```
C# FormCreationMission
    // Déjà visible = on copie le texte + image du grade
    if (cbGradeNouveau.SelectedItem is DataRowView selectedRow)
    {
        string libelleGrade = selectedRow["libelle"].ToString();
        string codeGrade = selectedRow["code"].ToString();
        int matricule = Convert.ToInt32(lblMatricule.Text); // récupère le matricule

        //Mise à jour du grade dans la base avec transaction
        using (SQLiteTransaction transaction = Connexion.Connec.BeginTransaction())
        {
            try
            {
                using (SQLiteCommand cmd = Connexion.Connec.CreateCommand())
                {
                    cmd.Transaction = transaction;
                    cmd.CommandText = "UPDATE Pompier SET codeGrade = @codeGrade WHERE matricule = @matricule";
                    cmd.Parameters.AddWithValue("@codeGrade", codeGrade);
                    cmd.Parameters.AddWithValue("@matricule", matricule);
                    cmd.ExecuteNonQuery();
                }

                transaction.Commit();

                //Mise à jour visuelle (affichage)
                txtGrade.Text = libelleGrade;

                string imagePath = Path.Combine(@"C:\OMAR\S2\SAE-D21\ImagesGrades", codeGrade + ".png");
                if (File.Exists(imagePath))
                {
                    pbGrade.Image = Image.FromFile(imagePath);
                    pbGrade.SizeMode = PictureBoxSizeMode.StretchImage;
                }
                else
                {
                    pbGrade.Image = null;
                }
            }
            catch (Exception ex)
            {
                transaction.Rollback();
                MessageBox.Show("X Erreur pendant la mise à jour du grade : " + ex.Message);
            }
        }

        cbGradeNouveau.Visible = false; // cacher après sélection
    }
    else
    {
        //Première fois on affiche et on remplit la ComboBox
        if (Connexion.Connec.State != ConnectionState.Open)
            Connexion.Connec.Open();

        string sqlGrade = "SELECT code, libelle FROM Grade";
        using (SQLiteCommand cmdGrade = new SQLiteCommand(sqlGrade, Connexion.Connec))
        using (SQLiteDataReader reader = cmdGrade.ExecuteReader())
        {
            DataTable dtGrades = new DataTable();
            dtGrades.Load(reader);
            cbGradeNouveau.DataSource = dtGrades;
        }
    }
}

Aucun problème détecté | 🔍
```

```
FormCreationMission
```

```
    cbGradeNouveau.DisplayMember = "libelle";
    cbGradeNouveau.ValueMember = "code";
}

cbGradeNouveau.Visible = true;
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur : " + ex.Message);
}
}

}

1 référence
private void BtnPlusInformation_Click_1(object sender, EventArgs e)
{
    if (!estAdmin)
    {
        FormConn formConnexion = new FormConn();
        formConnexion.ShowDialog();

        if (formConnexion.EstConnecte)
        {
            estAdmin = true;
            try
            {
                if (!gbInformationCarriere.Visible)
                {
                    gbInformationCarriere.Visible = true;

                    if (cbPompiers.SelectedItem is DataRowView row)
                    {
                        int matricule = Convert.ToInt32(row["matricule"]);

                        if (Connexion.Connec.State != ConnectionState.Open)
                            Connexion.Connec.Open();

                        //Charger toutes les casernes
                        string sqlToutesCaserne = "SELECT id, nom FROM Caserne";
                        SQLiteDataAdapter da = new SQLiteDataAdapter(sqlToutesCaserne, Connexion.Connec);
                        DataTable dtCaserne = new DataTable();
                        da.Fill(dtCaserne);
                        cbCaserneRattachement.DataSource = dtCaserne;
                        cbCaserneRattachement.DisplayMember = "nom";
                        cbCaserneRattachement.ValueMember = "id";

                        //Caserne actuelle
                        lstCaserneActuelle.Items.Clear();
                        string sqlCaserneActuelle = @""
                        SELECT A.dateA, C.nom
                        FROM Affection A JOIN Caserne C ON A.idCaserne = C.id
                        WHERE A.matriculePompier = @mat AND A.dateFin IS NULL";

                        using (SQLiteCommand cmd = new SQLiteCommand(sqlCaserneActuelle, Connexion.Connec))
                        {
                            cmd.Parameters.AddWithValue("@mat", matricule);
                            using (SQLiteDataReader reader = cmd.ExecuteReader())
                            {
                                while (reader.Read())
                                {
                                    string dateA = reader["dateA"].ToString();
                                    string nomCaserne = reader["nom"].ToString();
                                    lstCaserneActuelle.Items.Add(dateA + " - " + nomCaserne);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
FormCreationMission.UCPersonnel

    {
        if (reader.Read())
        {
            string dateA = reader["dateA"].ToString();
            string nomCaserne = reader["nom"].ToString();
            lstCaserneActuelle.Items.Add($"{dateA} : {nomCaserne}");
        }
        else
        {
            lstCaserneActuelle.Items.Add("Aucune affectation en cours.");
        }
    }

    //Habilitations
    lstHabilitations.Items.Clear();
    string sqlHabilitations = @"SELECT H.libelle FROM Habilitation H JOIN Passer P ON H.id = P.idHabilitation WHERE P.matriculePompier = @mat";
    using (SQLiteCommand cmd2 = new SQLiteCommand(sqlHabilitations, Connexion.Connec))
    {
        cmd2.Parameters.AddWithValue("@mat", matricule);
        using (SQLiteDataReader reader = cmd2.ExecuteReader())
        {
            while (reader.Read())
            {
                lstHabilitations.Items.Add(reader["libelle"].ToString());
            }
        }
    }

    //anciennes affectations
    lstAffectationsPassees.Items.Clear(); //on utilise une autre ListBox ici
    string sqlPassees = @"SELECT A.dateA, A.dateFin, C.nom
    FROM Affectation A
    JOIN Caserne C ON A.idCaserne = C.id
    WHERE A.matriculePompier = @mat AND A.dateFin IS NOT NULL
    ORDER BY A.dateA DESC";
    using (SQLiteCommand cmd3 = new SQLiteCommand(sqlPassees, Connexion.Connec))
    {
        cmd3.Parameters.AddWithValue("@mat", matricule);
        using (SQLiteDataReader reader = cmd3.ExecuteReader())
        {
            bool aDesAnciennes = false;
            while (reader.Read())
            {
                aDesAnciennes = true;
                string dateDebut = reader["dateA"].ToString();
                string dateFin = reader["dateFin"].ToString();
                string nomCaserne = reader["nom"].ToString();
                lstAffectationsPassees.Items.Add($"{dateDebut} à {dateFin} : {nomCaserne}");
            }

            if (!aDesAnciennes)
                lstAffectationsPassees.Items.Add("Aucune affectation passée.");
        }
    }
}
else
{
    lstAffectationsPassees.Items.Add("Aucune affectation en cours.");
}

59 % Aucun problème détecté | ⚡
```

```
C# FormCreationMission
    {
        gbInformationCarriere.Visible = false;
    }
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur : " + ex.Message);
}
}
else
{
    try
    {
        if (!gbInformationCarriere.Visible)
        {
            gbInformationCarriere.Visible = true;
            if (cbPompiers.SelectedItem is DataRowView row)
            {
                int matricule = Convert.ToInt32(row["matricule"]);

                if (Connexion.Connec.State != ConnectionState.Open)
                    Connexion.Connec.Open();

                // Charger toutes les casernes
                string sqlToutesCaserne = "SELECT id, nom FROM Caserne";
                SQLiteDataAdapter da = new SQLiteDataAdapter(sqlToutesCaserne, Connexion.Connec);
                DataTable dtCaserne = new DataTable();
                da.Fill(dtCaserne);
                cbCaserneRattachement.DataSource = dtCaserne;
                cbCaserneRattachement.DisplayMember = "nom";
                cbCaserneRattachement.ValueMember = "id";

                //Caserne actuelle
                lstCaserneActuelle.Items.Clear();
                string sqlCaserneActuelle = @""
                SELECT A.dateA, C.nom
                FROM Affection A JOIN Caserne C ON A.idCaserne = C.id
                WHERE A.matriculePompier = @mat AND A.dateFin IS NULL";

                using (SQLiteCommand cmd = new SQLiteCommand(sqlCaserneActuelle, Connexion.Connec))
                {
                    cmd.Parameters.AddWithValue("@mat", matricule);
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            string nomCaserne = reader["nom"].ToString();
                            string dateA = Convert.ToDateTime(reader["dateA"]).ToString("yyyy-MM-dd");
                            lstCaserneActuelle.Items.Add($"→ Depuis le {dateA} : {nomCaserne}");
                        }
                        else
                        {
                            lstCaserneActuelle.Items.Add("Aucune affectation en cours.");
                        }
                    }
                }

                //Habilitations
                lstHabilitations.Items.Clear();
            }
        }
    }
}
```

```
FormCreationMission.cs
```

```
FormCreationMission.UCPersonnel
```

```
string sqlHabilitations = @"SELECT H.libelle FROM Habilitation H JOIN Passeur P ON H.id = P.idHabilitation WHERE P.matriculePompier = @mat";
using (SQLiteCommand cmd2 = new SQLiteCommand(sqlHabilitations, Connexion.Connec))
{
    cmd2.Parameters.AddWithValue("@mat", matricule);
    using (SQLiteDataReader reader = cmd2.ExecuteReader())
    {
        while (reader.Read())
        {
            lstHabilitations.Items.Add(reader["libelle"].ToString());
        }
    }
}

//Anciennes affectations
lstAffectationsPassées.Items.Clear(); //on utilise une autre ListBox ici
string sqlPasseees = @""
SELECT A.dateA, A.dateFin, C.nom
FROM Affection A
JOIN Caserne C ON A.idCaserne = C.id
WHERE A.matriculePompier = @mat AND A.dateFin IS NOT NULL
ORDER BY A.dateA DESC";

using (SQLiteCommand cmd3 = new SQLiteCommand(sqlPasseees, Connexion.Connec))
{
    cmd3.Parameters.AddWithValue("@mat", matricule);
    using (SQLiteDataReader reader = cmd3.ExecuteReader())
    {
        bool aDesAnciennes = false;
        while (reader.Read())
        {
            aDesAnciennes = true;
            string dateDebut = Convert.ToDateTime(reader["dateA"]).ToString("yyyy-MM-dd");
            string dateFin = Convert.ToDateTime(reader["dateFin"]).ToString("yyyy-MM-dd");
            string nomCaserne = reader["nom"].ToString();
            lstAffectationsPassées.Items.Add($"{dateDebut} à {dateFin} : {nomCaserne}");
        }
        if (!aDesAnciennes)
            lstAffectationsPassées.Items.Add("Aucune affectation passée.");
    }
}
else
{
    gbInformationCarrière.Visible = false;
}
catch (Exception ex)
{
    MessageBox.Show("✖ Erreur : " + ex.Message);
}
}

1 référence
private void btnMettreAJour_Click_1(object sender, EventArgs e)
{
    //
    SQLiteTransaction transaction = null;
```

```
C# FormCreationMission                                FormCreationMission.UCP
try
{
    int matricule = Convert.ToInt32(lblMatricule.Text);
    int nouvelleCaserne = Convert.ToInt32(cbCaserneRattachement.SelectedValue);

    //Récupérer l'ID de la caserne actuelle
    int idCaserneActuelle = -1;
    string sqlSelect = "SELECT idCaserne FROM Affectation WHERE matriculePompier = @mat AND dateFin IS NULL";
    using (SQLiteCommand cmd = new SQLiteCommand(sqlSelect, Connexion.Connec))
    {
        cmd.Parameters.AddWithValue("@mat", matricule);
        object result = cmd.ExecuteScalar();
        if (result != DBNull.Value)
            idCaserneActuelle = Convert.ToInt32(result);
    }

    // Vérifier si la caserne a changé
    bool caserneChange = idCaserneActuelle != nouvelleCaserne;

    transaction = Connexion.Connec.BeginTransaction();

    if (caserneChange)
    {
        //Fermer ancienne affectation
        string sqlUpdate = @"UPDATE Affectation
        SET dateFin = @dateFin
        WHERE matriculePompier = @mat AND dateFin IS NULL";
        using (SQLiteCommand cmdUpdate = new SQLiteCommand(sqlUpdate, Connexion.Connec, transaction))
        {
            cmdUpdate.Parameters.AddWithValue("@dateFin", DateTime.Now.ToString("yyyy-MM-dd"));
            cmdUpdate.Parameters.AddWithValue("@mat", matricule);
            cmdUpdate.ExecuteNonQuery();
        }

        //Nouvelle affectation
        string nouvelleDate = DateTime.Now.AddSeconds(1).ToString("yyyy-MM-dd HH:mm:ss");

        string sqlInsert = @"INSERT INTO Affectation (matriculePompier, dateA, idCaserne)
        VALUES (@matricule, @dateA, @idCaserne)";
        using (SQLiteCommand cmdInsert = new SQLiteCommand(sqlInsert, Connexion.Connec, transaction))
        {
            cmdInsert.Parameters.AddWithValue("@matricule", matricule);
            cmdInsert.Parameters.AddWithValue("@dateA", nouvelleDate);
            cmdInsert.Parameters.AddWithValue("@idCaserne", nouvelleCaserne);
            cmdInsert.ExecuteNonQuery();
        }
    }

    //Mise à jour du champ enConge
    string sqlConge = "UPDATE Pompier SET enConge = @etatConge WHERE matricule = @matricule";
    using (SQLiteCommand cmdConge = new SQLiteCommand(sqlConge, Connexion.Connec, transaction))
    {
        cmdConge.Parameters.AddWithValue("@etatConge", chbConge.Checked ? 1 : 0);
    }
}
59 %   Aucun problème détecté
```

```
C# FormCreationMission                                FormCreationMission.UCPersonnel
    cmdConge.Parameters.AddWithValue("@matricule", matricule);
    cmdConge.ExecuteNonQuery();
}

transaction.Commit();
MessageBox.Show("✅ Mise à jour effectuée avec succès !");
}
catch (Exception ex)
{
    transaction?.Rollback();
    MessageBox.Show("❌ Erreur pendant la mise à jour : " + ex.Message);
    return;
}

//Mise à jour des deux listes d'affichage
try
{
    lstCaserneActuelle.Items.Clear();
    lstAffectationsPassees.Items.Clear();

    //Liste des affectations passées
    string sqlPassees = @""
SELECT A.dateA, A.dateFin, C.nom
FROM Affectation A
JOIN Caserne C ON A.idCaserne = C.id
WHERE A.matriculePompier = @mat AND A.dateFin IS NOT NULL
ORDER BY A.dateFin DESC";

    using (SQLiteCommand cmdPassees = new SQLiteCommand(sqlPassees, Connexion.Connec))
    {
        cmdPassees.Parameters.AddWithValue("@mat", Convert.ToInt32(lblMatricule.Text));
        using (SQLiteDataReader reader = cmdPassees.ExecuteReader())
        {
            if (!reader.HasRows)
            {
                lstAffectationsPassees.Items.Add("Aucune affectation passée.");
            }
            else
            {
                while (reader.Read())
                {
                    DateTime dateDebut = Convert.ToDateTime(reader["dateA"]);
                    DateTime dateFin = Convert.ToDateTime(reader["dateFin"]);
                    string nomCaserne = reader["nom"].ToString();
                    lstAffectationsPassees.Items.Add($"{dateDebut:yyyy-MM-dd} à {dateFin:yyyy-MM-dd} : {nomCaserne}");
                }
            }
        }
    }

    //Affectation en cours
    string sqlEnCours = @""
SELECT A.dateA, C.nom
FROM Affectation A
JOIN Caserne C ON A.idCaserne = C.id
WHERE A.matriculePompier = @mat AND A.dateFin IS NULL";

    using (SQLiteCommand cmdCours = new SQLiteCommand(sqlEnCours, Connexion.Connec))
    {
        cmdCours.Parameters.AddWithValue("@mat", Convert.ToInt32(lblMatricule.Text));
        ...
    }
}
}

59 %  🔍  Aucun problème détecté | ⚡
```

```
C# FormCreationMission
using (SQLiteDataReader reader = cmdCours.ExecuteReader())
{
    if (reader.Read())
    {
        DateTime dateDebut = Convert.ToDateTime(reader["dateA"]);
        string nomCaserne = reader["nom"].ToString();
        lstCaserneActuelle.Items.Add($"Depuis le {dateDebut:yyyy-MM-dd HH:mm:ss} : {nomCaserne}");
    }
    else
    {
        lstCaserneActuelle.Items.Add("Aucune affectation en cours.");
    }
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur d'affichage : " + ex.Message);
}

1 référence
private void btnQuitter_Click_1(object sender, EventArgs e)
{
    f.actualiser();
}

1 référence
private void rdbProfessionnel_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true; // Ignore key press events
}

1 référence
private void rdbVolontaire_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true; // Ignore key press events
}
```

3) Le code complet du User Control que vous jugez le plus intéressant : Créer Mission :

C# FormCreationMission

```
namespace FormCreationMission
{
    public partial class UCCreerMission : UserControl
    {
        Form4 tableauDeBord;
        public UCCreerMission(Form4 tableauDeBord)
        {
            InitializeComponent();
            this.tableauDeBord = tableauDeBord;
        }

        private void UCCreerMission_Load(object sender, EventArgs e)
        {
            flpEngins.Visible = false;
            flpPompiers.Visible = false;
            gbMobilisation.Visible = false;
            //btnQuitter.Visible = false;
            btnRapport.Visible = false;
            // --- Initialisation des labels et comboBox
            lblId.Text = (MesDatas.DsGlobal.Tables["Mission"].Rows.Count + 1).ToString();
            lblDateDeclanhee.Text = DateTime.Now.ToString();
            //Premiere comboBox : NatureSinistre
            cbNatureSinistre.DataSource = MesDatas.DsGlobal.Tables["NatureSinistre"];
            cbNatureSinistre.DisplayMember = "libelle";
            cbNatureSinistre.ValueMember = "id";
            //Deuxieme comboBox : TypeIntervention
            cbCaserneImmobiliser.DataSource = MesDatas.DsGlobal.Tables["Caserne"];
            cbCaserneImmobiliser.DisplayMember = "nom";
            cbCaserneImmobiliser.ValueMember = "id";
        }

        private void cbNatureSinistre_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = true;
        }

        private void cbCaserneImmobiliser_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = true;
        }

        private bool estEnMission(int matricule)
        {
    
```

72 %

Aucun problème détecté

C# FormCreationMission

FormCreationMission.UCCreerMission

```
0 références
private bool estEnMission(int matricule)
{
    // On récupère toutes les lignes de la table Mobiliser où le pompier est mobilisé
    DataRow[] mobilisations = MesDatas.DsGlobal.Tables["Mobiliser"]
        .Select("matriculePompier = " + matricule);
    //On parcourt chaque ligne de mobilisation pour vérifier si le pompier est en mission
    foreach (DataRow mobilisation in mobilisations)
    {
        int idMission = Convert.ToInt32(mobilisation["idMission"]);

        // On vérifie si la mission correspondante n'est pas encore terminée
        DataRow[] missions = MesDatas.DsGlobal.Tables["Mission"]
            .Select("id = " + idMission + " AND terminee = 0");
        //On regarde si il y a des missions actives
        if (missions.Length > 0)
        {
            return true; // Le pompier est en mission
        }
    }

    return false; // Aucune mission active trouvée
}

1 référence
private bool estEnConge(DataRow pompier)
{
    // On vérifie d'abord si la valeur est vide
    if (pompier["enConge"] == DBNull.Value)
        return false;
    //On transforme la valeur en boolean
    return Convert.ToBoolean(pompier["enConge"]);
}

12 références
private string Nettoyer(string input)
{
    // Vérifie si la chaîne est vide ou nulle
    if (string.IsNullOrEmpty(input)) return "";
    // Normalise la chaîne pour enlever les accents et caractères spéciaux
    string normalise = input.Normalize(System.Text.NormalizationForm.FormD);
    StringBuilder sb = new StringBuilder();
    // Parcourt chaque caractère de la chaîne normalisée
    foreach (char c in normalise)
    {
        System.Globalization.UnicodeCategory uc = System.Globalization.CharUnicodeInfo.GetUnicodeCategory(c);
        if (uc != System.Globalization.UnicodeCategory.NonSpacingMark && c <= 127)
            //UC = caractère unicode
    }
}
```

72 %

Aucun problème détecté

```

# FormCreationMission
    if (uc != System.Globalization.UnicodeCategory.NonSpacingMark && c <= 127)
        //UC = caractère unicode
        //NonSpacingMark = les accents et autres marques de diacritiques
        //c <= 127 = on garde les caractères ASCII
        {
            //On l'ajoute dans le stringbuilder
            sb.Append(c);
        }
    return sb.ToString();
}

0 références
private void dgvEngins_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}

0 références
private void dgvPompiers_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}

0 références
private void button2_Click(object sender, EventArgs e)
{
    Form4 f4 = new Form4();
    f4.ShowDialog();
}

1 référence
private void btnConstituerEquipe_Click_1(object sender, EventArgs e)
{
    //Verifier si les champs sont remplis
    if (string.IsNullOrWhiteSpace(txtMotif.Text) || string.IsNullOrWhiteSpace(txtRue.Text) || string.IsNullOrWhiteSpace(txtNom.Text))
    {
        MessageBox.Show("X Veuillez remplir tous les champs avant de continuer.");
        return;
    }
    //On affiche les panels des pompiers et des engins
    flpEngins.Visible = true;
    flpPompiers.Visible = true;
    // Liste finale des engins nécessaires
    List<String> enginsNecessaires = new List<String>();
    // Récupération des valeurs depuis les ComboBox
    int idNatureSinistre = Convert.ToInt32(cbNatureSinistre.SelectedValue);
    int idCaserne = Convert.ToInt32(cbCaserneImobiliser.SelectedValue);
    //On parcourt la table Necessiter pour trouver les engins nécessaires à l'id de NatureSinistre
    foreach (DataRow row in MesDatas.DsGlobal.Tables["Necessiter"].Select("idNatureSinistre = " + idNatureSinistre))
    {
        string codeTypeEngin = row["codeTypeEngin"].ToString();
        int nb = Convert.ToInt32(row["nb"]);
        //On part chercher maintenant dans la table des engin
        DataRows[] enginsDispo = MesDatas.DsGlobal.Tables["Engin"].Select($"codeTypeEngin = '{codeTypeEngin}' AND idCaserne = {idCaserne} AND enMission = 0 AND enPanne = 0");
        //On regarde si on a assez d'engins disponibles
        if (enginsDispo.Length > 0)
        {
            int nbRequis = Math.Min(nb, enginsDispo.Length); //On prend le nombre minimum dont on a besoin car c'est possible d'avoir 2 engins mais on a besoin de 5 alors on prend
            enginsNecessaires.Add(codeTypeEngin, nbRequis);
        }
        if (enginsDispo.Length < nb)
        {
            MessageBox.Show($"⚠ Pas assez d'engins pour le type {codeTypeEngin}. Nécessaires : {nb}, disponibles : {enginsDispo.Length}.", "Attention", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    //On prépare le datagridview que on utilise pas pour l'affichage pour les engins
    dgvEngins.Rows.Clear();
    if (dgvEngins.Columns.Count == 0)
    {
        dgvEngins.Columns.Add("codeTypeEngin", "Type d'engin");
        dgvEngins.Columns.Add("nb", "Quantité requise");
        dgvEngins.Columns.Add("equipage", "Équipage requis");
    }
    //On ajoute dans le dgv
    foreach ((String type, int nb) in enginsNecessaires)
    {
        int equipage = 0;
        DataRows[] rowType = MesDatas.DsGlobal.Tables["TypeEngin"].Select($"code = '{type}'");
        if (rowType.Length > 0)
            equipage = Convert.ToInt32(rowType[0]["equipage"]);

        dgvEngins.Rows.Add(type, nb, equipage);
    }
    //On prépare le datagridview que on utilise pas pour l'affichage pour les pompiers
    dgvPompiers.Rows.Clear();
    if (dgvPompiers.Columns.Count == 0)
    {
        dgvPompiers.Columns.Add("matricule", "Matricule");
        dgvPompiers.Columns.Add("nom", "Nom");
        dgvPompiers.Columns.Add("prenom", "Prénom");
        dgvPompiers.Columns.Add("pourEngin", "Type Engin");
    }
    //////////////// Maintenant pour les pompiers
}

```

```
C# FormCreationMission
    ///////////////////////////////////////////////////////////////////Maintenant pour les pompiers
    foreach ((string typeEngin, int nombre) in enginsNecessaires)
    {
        List<int> habilitations = new List<int>();
        DataRow[] rowsEmbarquer = MesDatas.DsGlobal.Tables["Embarquer"].Select($"codeTypeEngin = '{typeEngin}'");
        //On ajoute l id des habilitation dans la liste habilitations
        foreach (DataRow row in rowsEmbarquer)
        {
            int idHab = Convert.ToInt32(row["idHabilitation"]);
            if (!habilitations.Contains(idHab))
                habilitations.Add(idHab);
        }

        List<DataRow> pompiersEligibles = new List<DataRow>();
        //On parcourt les habilitations pour trouver les pompiers éligibles
        foreach (int idHab in habilitations)
        {
            //Tous les pompiers ayant cette habilitation
            DataRow[] rowsPasser = MesDatas.DsGlobal.Tables["Passer"].Select("idHabilitation = " + idHab);

            foreach (DataRow passerRow in rowsPasser)
            {
                int matricule = Convert.ToInt32(passerRow["matriculePompier"]);

                //Vérifier qu'il est bien affecté à la caserne sélectionnée, et encore en poste
                DataRow[] affectations = MesDatas.DsGlobal.Tables["Affectation"]
                    .Select("matriculePompier = " + matricule + " AND idCaserne = " + idCaserne + " AND dateFin IS NULL");

                if (affectations.Length > 0)
                {
                    //Récupération du pompier depuis la table Pompier
                    DataRow[] pompiers = MesDatas.DsGlobal.Tables["Pompier"].Select("matricule = " + matricule);
                    if (pompiers.Length > 0)
                    {
                        DataRow pompier = pompiers[0];
                        //On vérifie si il sont en mission ou en congé
                        if (Convert.ToInt32(pompier["enMission"]) == 0 && !estEnConge(pompier))
                        {
                            if (!pompiersEligibles.Contains(pompier))
                                pompiersEligibles.Add(pompier);
                        }
                    }
                }
            }
        }

        // On parcourt les pompiers éligibles pour les ajouter au DataGridView
        int equipage = 0;
        DataRow[] rowType = MesDatas.DsGlobal.Tables["TypeEngin"].Select($"code = '{typeEngin}'");
        foreach (DataRow rowType in rowType)
        {
            if (equipage < nombre)
            {
                foreach (DataRow pompier in pompiersEligibles)
                {
                    if (equipage < nombre)
                    {
                        // Ajout du pompier au DataGridView
                        // ...
                    }
                }
            }
        }
    }
}

// Aucun problème détecté
```

```
FormCreationMission    FormCreationMission.UCCreerMission
    int equipage = 0;
    DataRow[] rowType = MesDatas.DsGlobal.Tables["TypeEngin"].Select($"code = '{typeEngin}'");
    if (rowType.Length > 0)
        equipage = Convert.ToInt32(rowType[0]["equipage"]);

    int totalPompiers = equipage * nombre;
    //selection c est la liste finale
    List<DataRow> selection = pompiersEligibles.Take(totalPompiers).ToList(); //Elle prend les n premiers pompiers éligibles

    foreach (DataRow p in selection)
    {
        dgvPompiers.Rows.Add(p["matricule"], p["nom"], p["prenom"], typeEngin);
    }
    //Réaffichage visuel des engin depuis dgvEngins
    flpEngins.Controls.Clear();
    foreach (DataGridViewRow row in dgvEngins.Rows)
    {
        if (row.IsNewRow || row.Cells[0].Value == null) continue;

        string type = row.Cells[0].Value.ToString();
        string quantite = row.Cells[1].Value.ToString();
        string equipage = row.Cells[2].Value.ToString();
        //Panel pour chaque engin
        Panel panel = new Panel
        {
            Width = flpEngins.Width - 25,
            Height = 40,
            BackColor = Color.LightGray,
            Margin = new Padding(3),
            Padding = new Padding(5),
            Tag = type
        };
        //Label que on va ajouter dans le panel
        Label lbl = new Label
        {
            Text = $"• {type} | Quantité : {quantite} | Équipage : {equipage}",
            Dock = DockStyle.Fill,
            TextAlign = ContentAlignment.MiddleLeft
        };

        panel.Controls.Add(lbl);
        flpEngins.Controls.Add(panel);
    }

    //Réaffichage visuel des pompiers depuis dgvPompiers
    flpPompiers.Controls.Clear();
    foreach (DataGridViewRow row in dgvPompiers.Rows)
    {
        if (row.IsNewRow || row.Cells[0].Value == null) continue;
```

```
72 %    Aucun problème détecté | ⚙️ ▾
FormCreationMission    FormCreationMission.UCCreerMission
    if (row.IsNewRow || row.Cells[0].Value == null) continue;

    int matricule = Convert.ToInt32(row.Cells[0].Value);
    string nom = row.Cells[1].Value.ToString();
    string prenom = row.Cells[2].Value.ToString();
    string typeEngin = row.Cells[3].Value.ToString();

    Panel panel = new Panel
    {
        Width = flpPompiers.Width - 25,
        Height = 40,
        BackColor = Color.LightSteelBlue,
        Margin = new Padding(4),
        Padding = new Padding(5),
        Tag = new Tuple<int, string>(matricule, typeEngin)
    };

    Label lbl = new Label
    {
        Text = $"{prenom} {nom} - Matricule : {matricule} - Engin : {typeEngin}",
        Dock = DockStyle.Fill,
        TextAlign = ContentAlignment.MiddleLeft
    };

    panel.Controls.Add(lbl);
    flpPompiers.Controls.Add(panel);
}

1 référence
private void btnMAJ_Click_1(object sender, EventArgs e)
{
    //On vérifie si les champs sont remplis
    if (string.IsNullOrWhiteSpace(txtMotif.Text) || string.IsNullOrWhiteSpace(txtRue.Text) || string.IsNullOrWhiteSpace(txtCodePostale.Text) || string.IsNullOrWhiteSpace(txtVille.Text))
    {
        MessageBox.Show("Veuillez remplir tous les champs avant de continuer.");
        return;
    }
    try
    {
        // --- Récupération des DataTables nécessaires
        DataTable dtMission = MesDatas.DsGlobal.Tables["Mission"];
        DataTable dtEngin = MesDatas.DsGlobal.Tables["Engin"];
        DataTable dtPompier = MesDatas.DsGlobal.Tables["Pompier"];
        DateTime date = DateTime.Now;
        // --- Créeation de la nouvelle ligne en mémoire
        DataRow nouvelleMission = dtMission.NewRow();
```

```
C# FormCreationMission
    // --- Ajout de la nouvelle ligne en mémoire
    DataRow nouvelleMission = dtMission.NewRow();
    nouvelleMission["motifAppel"] = txtMotif.Text.Trim();
    nouvelleMission["adresse"] = txtRue.Text.Trim();
    nouvelleMission["cp"] = txtCodePostale.Text.Trim();
    nouvelleMission["ville"] = txtVille.Text.Trim();
    nouvelleMission["dateHeureDepart"] = date;
    nouvelleMission["terminee"] = 0; // Mission non terminée par défaut
    // --- Vérification ID mission
    if (string.IsNullOrWhiteSpace(lblId.Text))
    {
        MessageBox.Show("X L'ID de la mission est vide.");
        return;
    }
    //On nettoie l'ID pour éviter les espaces ou caractères indésirables
    string id = lblId.Text.Trim();
    nouvelleMission["id"] = id;

    //Vérification comboBox
    if (cbCaserneImmobiliser.SelectedValue == null || cbNatureSinistre.SelectedValue == null)
    {
        MessageBox.Show("X Veuillez sélectionner une caserne et une nature de sinistre.");
        return;
    }
    //Ajout des valeurs sélectionnées dans les ComboBox
    nouvelleMission["idCaserne"] = Convert.ToInt32(cbCaserneImmobiliser.SelectedValue);
    nouvelleMission["idNatureSinistre"] = Convert.ToInt32(cbNatureSinistre.SelectedValue);
    //Ajout dans le DataSet
    dtMission.Rows.Add(nouvelleMission);
    // --- Ajouter les pompiers dans la table Mobiliser
    DataTable dtMobiliser = MesDatas.DsGlobal.Tables["Mobiliser"];
    DataTable dtEmbarquer = MesDatas.DsGlobal.Tables["Embarquer"];

    foreach (DataGridViewRow row in dgvPompiers.Rows)
    {
        if (row.Cells["matricule"].Value != null && row.Cells["pourEngin"].Value != null)
        {
            int matricule = Convert.ToInt32(row.Cells["matricule"].Value);
            string codeTypeEngin = row.Cells["pourEngin"].Value.ToString();
            //On récupère l'habilitation associée à ce type d'engin
            DataRow[] habRows = dtEmbarquer.Select($"codeTypeEngin = '{codeTypeEngin}'");
            if (habRows.Length > 0)
            {
                int idHabilitation = Convert.ToInt32(habRows[0]["idHabilitation"]);

                // + Nouvelle ligne dans Mobiliser
                DataRow ligneMobiliser = dtMobiliser.NewRow();
                ligneMobiliser["matriculePompier"] = matricule;
                ligneMobiliser["idMission"] = nouvelleMission["id"];
                ligneMobiliser["idHabilitation"] = idHabilitation;
            }
        }
    }
}

72 %  Aucun problème détecté
```

```
FormCreationMission
```

```
    ligneMobiliser["idHabilitation"] = idHabilitation;
    ligneMobiliser["idMission"] = idMission;
    dtMobiliser.Rows.Add(ligneMobiliser);

}

//Mise à jour enMission pour les pompiers
foreach (DataGridViewRow row in dgvPOMPIERS.Rows)
{
    if (row.Cells["Matricule"].Value != null)
    {
        int matricule = Convert.ToInt32(row.Cells["Matricule"].Value);
        DataRow[] pompierRow = dtPompier.Select($"matricule = {matricule}");
        if (pompierRow.Length > 0)
        {
            pompierRow[0]["enMission"] = 1;
        }
    }
}

//Mise à jour enMission pour les engins
DataTable dtPartirAvec = MesDatas.DsGlobal.Tables["PartirAvec"];
int idMission = Convert.ToInt32(id); // déjà défini au-dessus
int idCaserne = Convert.ToInt32(cbCaserneImmobiliser.SelectedValue);
foreach (DataGridViewRow row in dgvEngins.Rows)
{
    if (row.Cells["typeEngin"].Value != null & row.Cells["nombre"].Value != null)
    {
        string codeTypeEngin = row.Cells["typeEngin"].Value.ToString();
        int nombreRequis = Convert.ToInt32(row.Cells["nombre"].Value);

        // Récupère les engins disponibles
        DataRow[] enginsDispo = dtEngin.Select($"codeTypeEngin = '{codeTypeEngin}' AND idCaserne = {idCaserne} AND enMission = 0");

        //Prendre uniquement les N premiers nécessaires
        for (int i = 0; i < Math.Min(nombreRequis, enginsDispo.Length); i++)
        {
            DataRow engin = enginsDispo[i];
            engin["enMission"] = 1;

            DataRow ligne = dtPartirAvec.NewRow();
            ligne["idMission"] = idMission;
            ligne["idCaserne"] = idCaserne;
            ligne["codeTypeEngin"] = codeTypeEngin;
            ligne["numeroEngin"] = engin["numero"];
            ligne["reparationsEventuelles"] = DBNull.Value;
            dtPartirAvec.Rows.Add(ligne);
        }
    }
}
```

72 % Aucun problème détecté

```
C# FormCreationMission          ▾ FormCreationMission.UCCreerMission

    //Nettoyage du formulaire
    txtMotif.Text = "";
    txtRue.Text = "";
    txtVille.Text = "";
    txtCodePostale.Text = "";

    cbNatureSinistre.SelectedIndex = -1;
    cbCaserneImmobiliser.SelectedIndex = -1;
    dgvEngins.Rows.Clear();
    dgvPompiers.Rows.Clear();

    //Prochain ID
    int prochainId = 1;
    if (dtMission.Rows.Count > 0)
    {
        var lastRow = dtMission.Rows[dtMission.Rows.Count - 1];
        int lastId = Convert.ToInt32(lastRow["id"]);
        prochainId = lastId + 1;
    }

    lblId.Text = prochainId.ToString();

    MesDatas.DsGlobal.AcceptChanges();

    // --- Rafraîchir tableau de bord
    if (tableauDeBord != null)
    {
        tableauDeBord.btnActualiser.PerformClick();
    }
    else
    {
        MessageBox.Show("⚠ Le tableau de bord est introuvable.");
    }
    //On les remets à 0
    flpEngins.Visible = false;
    flpPompiers.Visible = false;
}
catch (Exception ex)
{
    MessageBox.Show("✖ Erreur :\n" + ex.Message, "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private void btnRapport_Click_1(object sender, EventArgs e)
{
    try
    {
        string nomFichier = "Dernier_Mission.pdf";
        ...
    }
    catch (Exception ex)
    {
        MessageBox.Show("✖ Erreur :\n" + ex.Message, "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

72 %
```

Aucun problème détecté | ⚡

```
FormCreationMission
```

```
private void btnRapport_Click_1(object sender, EventArgs e)
{
    try
    {
        string nomFichier = "Rapport_Mission.pdf";
        string chemin = Path.Combine(Environment.SpecialFolder.Desktop), nomFichier);

        if (File.Exists(chemin))
        {
            File.Delete(chemin);
        }

        // creation du pdf avec iTextSharp
        Document document = new Document(PageFormat.A4, 50, 50, 25, 25);
        PdfWriter.GetInstance(document, new FileStream(chemin, FileMode.Create));
        document.Open();

        // titre
        iTextSharp.text.Font titreFont = FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 18);
        iTextSharp.text.Paragraph titre = new iTextSharp.text.Paragraph("RAPPORT DE MISSION", titreFont);

        titre.Alignment = Element.ALIGN_CENTER;
        document.Add(titre);
        document.Add(new Paragraph("\n"));

        // information mission
        document.Add(new Paragraph("Numero de mission : " + Nettoyer(lblId.Text)));
        document.Add(new Paragraph("Date declenchement : " + Nettoyer(lblDateDeclanchee.Text)));
        document.Add(new Paragraph("Nature du sinistre : " + Nettoyer(cbNatureSinistre.Text)));
        document.Add(new Paragraph("Caserne : " + Nettoyer(cbCaserneImmobiliser.Text)));
        document.Add(new Paragraph("Adresse : " + Nettoyer(txtRue.Text + ", " + txtCodePostale.Text + " " + txtVille.Text)));
        document.Add(new Paragraph("Description : " + Nettoyer(txtMotif.Text)));

        document.Add(new Paragraph("\n-----\n"));

        // Engins mobilises
        document.Add(new Paragraph("Engins mobilises :"));
        if (dgvEngins.Rows.Count == 0)
        {
            document.Add(new Paragraph("- Aucun engin mobilise."));
        }
        else
        {
            foreach (DataGridViewRow row in dgvEngins.Rows)
            {
                if (!row.IsNewRow & row.Cells[0].Value != null)
                {
                    string type = Nettoyer(row.Cells[0].Value?.ToString() ?? "Inconnu");
                    string quantite = row.Cells.Count > 1 & row.Cells[1].Value != null ? Nettoyer(row.Cells[1].Value.ToString()) : "N/A";
                    string numero = row.Cells.Count > 2 & row.Cells[2].Value != null ? Nettoyer(row.Cells[2].Value.ToString()) : "N/A";
                    string equipage = row.Cells.Count > 3 & row.Cells[3].Value != null ? Nettoyer(row.Cells[3].Value.ToString()) : "N/A";
                    document.Add(new Paragraph("- " + type + " : " + quantite + " engin(s), " + equipage + " pompier(s) par engin"));
                }
            }
        }
    }
}

// Pompiers mobilisés
document.Add(new Paragraph("\nPompiers mobilises :"));
if (dgvPompiers.Rows.Count == 0)
{
    document.Add(new Paragraph("- Aucun pompier affecté."));
}
else
{
    foreach (DataGridViewRow row in dgvPompiers.Rows)
    {
        if (!row.IsNewRow & row.Cells[0].Value != null)
        {
            string nom = row.Cells.Count > 1 ? Nettoyer(row.Cells[1].Value?.ToString() ?? "Nom") : "Nom";
            string prenom = row.Cells.Count > 2 ? Nettoyer(row.Cells[2].Value?.ToString() ?? "Prenom") : "Prenom";
            string engin = row.Cells.Count > 3 ? Nettoyer(row.Cells[3].Value?.ToString() ?? "?") : "?";
            document.Add(new Paragraph("- " + prenom + " " + nom + " (engin : " + engin + ")"));
        }
    }
}

document.Add(new Paragraph("\nRapport genere le : " + DateTime.Now.ToString("dd/MM/yyyy HH:mm")));
document.Close();

MessageBox.Show("PDF genere avec succes sur le bureau !");
}
catch (Exception ex)
{
    MessageBox.Show("Erreur PDF : " + ex.Message);
}
}

1 référence
private void cbNatureSinistre_KeyPress_1(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}

1 référence
private void cbCaserneImmobiliser_KeyPress_1(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}
```

```
FormCreationMission
```

```
document.Add(new Paragraph("- " + type + " : " + quantite + " engin(s), " + equipage + " pompier(s) par engin"));

}

// Pompiers mobilisés
document.Add(new Paragraph("\nPompiers mobilises :"));
if (dgvPompiers.Rows.Count == 0)
{
    document.Add(new Paragraph("- Aucun pompier affecté."));
}
else
{
    foreach (DataGridViewRow row in dgvPompiers.Rows)
    {
        if (!row.IsNewRow & row.Cells[0].Value != null)
        {
            string nom = row.Cells.Count > 1 ? Nettoyer(row.Cells[1].Value?.ToString() ?? "Nom") : "Nom";
            string prenom = row.Cells.Count > 2 ? Nettoyer(row.Cells[2].Value?.ToString() ?? "Prenom") : "Prenom";
            string engin = row.Cells.Count > 3 ? Nettoyer(row.Cells[3].Value?.ToString() ?? "?") : "?";
            document.Add(new Paragraph("- " + prenom + " " + nom + " (engin : " + engin + ")"));
        }
    }
}

document.Add(new Paragraph("\nRapport genere le : " + DateTime.Now.ToString("dd/MM/yyyy HH:mm")));
document.Close();

MessageBox.Show("PDF genere avec succes sur le bureau !");
}
catch (Exception ex)
{
    MessageBox.Show("Erreur PDF : " + ex.Message);
}
}

1 référence
private void cbNatureSinistre_KeyPress_1(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}

1 référence
private void cbCaserneImmobiliser_KeyPress_1(object sender, KeyPressEventArgs e)
{
    e.Handled = true;
}
```

**4) Un extrait du code permettant de générer le pdf**

```
C# UCRecapitulMission
    }

    1 référence
    private void btnRapport_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrWhiteSpace(this.dateRetour))
        {
            string bureauPath = Path.Combine(
                Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
                "Rapports Missions"
            );

            string id = lblID2.Text.Trim();
            string nomFichier = $"Mission_{id}_Rapport.pdf";
            if (!Directory.Exists(bureauPath)) Directory.CreateDirectory(bureauPath);
            string cheminPDF = Path.Combine(bureauPath, nomFichier);

            if (File.Exists(cheminPDF))
            {
                MessageBox.Show(" ! Ce rapport existe déjà dans le Bureau.");
                return;
            }

            Document doc = new Document(PageSize.A4, 50, 50, 50, 50);
            try
            {
                PdfWriter.GetInstance(doc, new FileStream(cheminPDF, FileMode.Create));
                doc.Open();

                // Fonts
                iTextFont titreFont = FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 18, BaseColor.BLACK);
                iTextFont sectionFont = FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 14, BaseColor.DARK_GRAY);
                iTextFont normalFont = FontFactory.GetFont(FontFactory.HELVETICA, 12, BaseColor.BLACK);
                iTextFont smallFont = FontFactory.GetFont(FontFactory.HELVETICA, 10, BaseColor.GRAY);

                // Logo
                string cheminLogo = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Images", "logo.png");
                if (File.Exists(cheminLogo))
                {
                    iTextSharp.text.Image logo = iTextSharp.text.Image.GetInstance(cheminLogo);
                    logo.ScaleToFit(100f, 100f);
                    logo.Alignment = Element.ALIGN_CENTER;
                    doc.Add(logo);
                }

                doc.Add(new Paragraph("\n"));
                doc.Add(new Paragraph($"RAPPORT DE MISSION N° {id}", titreFont) { Alignment = Element.ALIGN_CENTER });
                doc.Add(new Paragraph("\n"));
            }
        }
    }
}

72 %  Aucun problème détecté
```

```
UCRecapitulMission.UCAffichageMission
string debut = lblDateDebut.Text.Split(':')[1].Trim();
string description = lblDescription.Text;
string caserne = lblCaserne.Text.Split(':')[1].Trim();

PdfPTable tableInfo = new PdfPTable(1);
tableInfo.WidthPercentage = 100;
tableInfo.SpacingAfter = 10f;

tableInfo.AddCell(new PdfPCell(new Phrase("Informations Générales", sectionFont)) { Colspan = 1 });
tableInfo.AddCell(new Phrase($"Début : {debut}", normalFont));
tableInfo.AddCell(new Phrase($"Fin : {this.dateRetour}", normalFont));
tableInfo.AddCell(new Phrase($"Caserne : {caserne}", normalFont));
tableInfo.AddCell(new Phrase($"Motif : {type}", normalFont));
tableInfo.AddCell(new Phrase($"Nature : {description}", normalFont));
doc.Add(tableInfo);

// Engins mobilisés
doc.Add(new Paragraph("Engins mobilisés :", sectionFont));
var lignesEngins = ds.Tables["PartirAvec"].Select($"idMission = {id}");
var dtEngin = ds.Tables["Engin"];
var dtTypeEngin = ds.Tables["TypeEngin"];

if (lignesEngins.Length == 0)
{
    doc.Add(new Paragraph("Aucun engin mobilisé.", normalFont));
}
else
{
    foreach (var ligne in lignesEngins)
    {
        int num = Convert.ToInt32(ligne["numeroEngin"]);
        var engin = dtEngin.Select($"numero = {num}").FirstOrDefault();
        if (engin != null)
        {
            string code = engin["codeTypeEngin"].ToString();
            string libelle = dtTypeEngin.Select($"code = '{code}'").FirstOrDefault()?[ "nom" ].ToString() ?? code;
            doc.Add(new Paragraph($"→ {libelle}", normalFont));
        }
    }

    doc.Add(new Paragraph("\n"));

    // Pompiers mobilisés
    doc.Add(new Paragraph("Pompiers mobilisés :", sectionFont));
    string lignes = recapTableMobiliser(id);
    doc.Add(new Paragraph(lignes, normalFont));

    // Date de génération
}
```

72 %

Aucun problème détecté

```

UCRecapitulMission                                         UCRecapitulMission.UCAffichageMission
    string debut = lblDateDebut.Text.Split(':')[1].Trim();
    string description = lblDescription.Text;
    string caserne = lblCaserne.Text.Split(':')[1].Trim();

    PdfPTable tableInfo = new PdfPTable(1);
    tableInfo.WidthPercentage = 100;
    tableInfo.SpacingAfter = 10f;

    tableInfo.AddCell(new PdfPCell(new Phrase("Informations Générales", sectionFont)) { Colspan = 1 });
    tableInfo.AddCell(new Phrase($"Début : {debut}", normalFont));
    tableInfo.AddCell(new Phrase($"Fin : {this.dateRetour}", normalFont));
    tableInfo.AddCell(new Phrase($"Caserne : {caserne}", normalFont));
    tableInfo.AddCell(new Phrase($"Motif : {type}", normalFont));
    tableInfo.AddCell(new Phrase($"Nature : {description}", normalFont));
    doc.Add(tableInfo);

    // Engins mobilisés
    doc.Add(new Paragraph("Engins mobilisés :", sectionFont));
    var lignesEngins = ds.Tables["PartirAvec"].Select($"idMission = {id}");
    var dtEngin = ds.Tables["Engin"];
    var dtTypeEngin = ds.Tables["TypeEngin"];

    if (lignesEngins.Length == 0)
    {
        doc.Add(new Paragraph("Aucun engin mobilisé.", normalFont));
    }
    else
    {
        foreach (var ligne in lignesEngins)
        {
            int num = Convert.ToInt32(ligne["numeroEngin"]);
            var engin = dtEngin.Select($"numero = {num}").FirstOrDefault();
            if (engin != null)
            {
                string code = engin["codeTypeEngin"].ToString();
                string libelle = dtTypeEngin.Select($"code = '{code}'").FirstOrDefault()["nom"].ToString() ?? code;
                doc.Add(new Paragraph($"{libelle}", normalFont));
            }
        }

        doc.Add(new Paragraph("\n"));

        // Pompiers mobilisés
        doc.Add(new Paragraph("Pompiers mobilisés :", sectionFont));
        string lignes = recapTableMobiliser(id);
        doc.Add(new Paragraph(lignes, normalFont));

        // Date de génération
        doc.Add(new Paragraph("\n Rapport généré le : {DateTime.Now:dd/MM/yyyy HH:mm}", smallFont));
    }

    doc.Close();
    MessageBox.Show("Rapport généré avec succès !");
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur lors de la génération du rapport :\n" + ex.Message);
}
}
else
{
    MessageBox.Show("X La mission doit d'abord être terminée pour générer le rapport.");
}

```

```

doc.Add(new Paragraph("\n"));

// Pompiers mobilisés
doc.Add(new Paragraph("Pompiers mobilisés :", sectionFont));
string lignes = recapTableMobiliser(id);
doc.Add(new Paragraph(lignes, normalFont));

// Date de génération
doc.Add(new Paragraph("\n Rapport généré le : {DateTime.Now:dd/MM/yyyy HH:mm}", smallFont));

doc.Close();
MessageBox.Show("Rapport généré avec succès !");
}
catch (Exception ex)
{
    MessageBox.Show("X Erreur lors de la génération du rapport :\n" + ex.Message);
}
}
else
{
    MessageBox.Show("X La mission doit d'abord être terminée pour générer le rapport.");
}

```

UC pour les engins (Binding source) :

```
FormCreationMission    ▾ FormCreationMission.gestionEngins
{
    5 références
    public partial class gestionEngins: UserControl
    {
        private DataSet ds;
        private BindingSource bsEngins;
        0 références
        public gestionEngins()
        {
            InitializeComponent();
            this.bsEngins = new BindingSource();
            ChargerTout();
        }

        1 référence
        public gestionEngins(DataSet dsp, Form4 f)
        {
            InitializeComponent();
            this.bsEngins = new BindingSource();
            this.ds = dsp;
        }

        1 référence
        private void gestionEngins_Load(object sender, EventArgs e)
        {
            if (ds != null && ds.Tables.Contains("Caserne"))
            {
                pbmax.Image = Properties.Resources.final_page;
                pbavant.Image = Properties.Resources.previous_page;
                pbapres.Image = Properties.Resources.next_page;
                pbmin.Image = Properties.Resources.first_page;

                cboChoix.DataSource = ds.Tables["Caserne"];
                cboChoix.DisplayMember = "nom";

                if (ds.Tables["Caserne"].Columns.Contains("id"))
                {
                    cboChoix.ValueMember = "id";

                    if (cboChoix.Items.Count > 0)
                    {
                        cboChoix.SelectedIndex = 0; // Ca déclenchera cboChoix_SelectedIndexChanged
                    }
                }
                else
                {
                    MessageBox.Show("Table non trouvée", "Erreur de Données", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}
```

```
FormCreationMission    ▾ FormCreationMission.gestionEngins
{
    // Méthode qui établit la première connexion et récupère toutes les données que l'on viendra rechercher après (3 tables importantes) dans le DataSet
    // Voir si il faut adapter avec le fichier MesDatas
    1 référence
    private void ChargerTout()
    {
        string chaine = "Data Source=SDIS67.db;Version=3;";
        ds = new DataSet();

        try
        {
            using (SQLiteConnection connection = new SQLiteConnection(chaine))
            {
                connection.Open();

                // Charger la table Caserne
                using (SQLiteDataAdapter daCaserne = new SQLiteDataAdapter("SELECT id, nom FROM Caserne", connection))
                {
                    daCaserne.Fill(ds, "Caserne");
                }

                // 2. Charger la table Engin
                using (SQLiteDataAdapter daEngin = new SQLiteDataAdapter("SELECT idCaserne, codeTypeEngin, numero, dateReception, enMission, enPanne FROM Engin", connection))
                {
                    daEngin.Fill(ds, "Engin");
                }

                // 3. Charger la table TypeEngin
                using (SQLiteDataAdapter daTypeEngin = new SQLiteDataAdapter("SELECT code, nom FROM TypeEngin", connection))
                {
                    daTypeEngin.Fill(ds, "TypeEngin");
                }
            }
        }
        catch (SQLiteException err)
        {
            MessageBox.Show($"Erreur chargement de toutes les données: {err.Message}", "Erreur de Base de Données", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    1 référence
    private void cboChoix_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (cboChoix.SelectedItem == null)
        {
            return;
        }

        int idCaserne = Convert.ToInt32(cboChoix.SelectedValue);
    }
}
```

```
int idCaserneSelectionne;
try
{
    DataRowView selectedRow = cboChoix.SelectedItem as DataRowView;
    if (selectedRow != null && selectedRow.Row.Table.Columns.Contains("id"))
    {
        idCaserneSelectionne = Convert.ToInt32(selectedRow["id"]);
    }
    else
    {
        MessageBox.Show("Impossible de récupérer l'ID de la caserne sélectionnée. La colonne 'id' est introuvable ou l'élément sélectionné n'est pas une ligne de tableau.");
        return;
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Erreur lors de la récupération de l'ID de la caserne: {ex.Message}", "Erreur de Conversion", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}

DataTable dtEngins = new DataTable(); // DataTable pour les engins filtrés

if (ds.Tables.Contains("Engin"))
{
    // Fait une datatable seulement avec les engins de la caserne choisi
    DataRow[] filtre = ds.Tables["Engin"].Select($"idCaserne = {idCaserneSelectionne}");

    // Clone les colonnes de la table Engin pour le nouveau DataTable
    dtEngins = ds.Tables["Engin"].Clone();

    foreach (DataRow row in filtre)
    {
        dtEngins.ImportRow(row);
    }

    if (ds.Tables.Contains("TypeEngin") && ds.Tables["TypeEngin"].Columns.Contains("nom"))
    {
        if (!dtEngins.Columns.Contains("NomTypeEngin"))
        {
            dtEngins.Columns.Add("NomTypeEngin", typeof(string));
        }

        foreach (DataRow rowEngin in dtEngins.Rows)
        {
            string codeTypeEngin = rowEngin["codeTypeEngin"].ToString();
            DataRow[] typeEnginRows = ds.Tables["TypeEngin"].Select($"code = '{codeTypeEngin}'");

```

```
        if (typeEnginRows.Length > 0)
        {
            rowEngin["NomTypeEngin"] = typeEnginRows[0]["nom"];
        }
        else
        {
            rowEngin["NomTypeEngin"] = $"Type Inconnu ({codeTypeEngin})";
        }
    }

    //creation du matricule
    if (dtEngins.Columns.Contains("idCaserne") && dtEngins.Columns.Contains("numero") && dtEngins.Columns.Contains("codeTypeEngin"))
    {
        dtEngins.Columns.Add("matricule", typeof(string));
        foreach (DataRow row in dtEngins.Rows)
        {
            string idCaserne = row.IsNull("idCaserne") ? string.Empty : row["idCaserne"].ToString();
            string codeTypeEngin = row.IsNull("codeTypeEngin") ? string.Empty : row["codeTypeEngin"].ToString();
            string numeroEngin = row.IsNull("numero") ? string.Empty : row["numero"].ToString();
            row["matricule"] = $"{idCaserne}-{codeTypeEngin}-{numeroEngin}";
        }
    }
    else
    {
        MessageBox.Show("Les colonnes 'idCaserne', 'numero' ou 'codeTypeEngin' sont manquantes dans la table Engin du DataSet en mémoire. Impossible de construire le matricule.");
    }

    // Créeation d'une colonne Image avec toutes les images correspondant à la colonne code car nomImage= code + ".png" et ajouter dans la bonne ligne du DataGridView
    if (dtEngins.Columns.Contains("codeTypeEngin"))
    {
        if (!dtEngins.Columns.Contains("ImageEngin"))
        {
            dtEngins.Columns.Add("ImageEngin", typeof(Image));
        }

        foreach (DataRow row in dtEngins.Rows)
        {
            string codeTypeEngin = row["codeTypeEngin"].ToString();
            Image enginImage = (Image)Properties.Resources.ResourceManager.GetObject(codeTypeEngin);

            // cas où l'image n'est pas trouvée pour ce codeTypeEngin
            if (enginImage != null)
            {
                row["ImageEngin"] = enginImage;
            }
        }
    }
}
else
{
    MessageBox.Show("Aucun résultat trouvé pour la recherche de caserne.");
}
```

```
C# FormCreationMission                                     FormCreationMission.gestionEngins

    // cas où l'image n'est pas trouvée pour ce codeTypeEngin
    if (enginImage != null)
    {
        row["ImageEngin"] = enginImage;
    }
    else
    {
        Console.WriteLine($"Avertissement: Image non trouvée pour le code d'engin '{codeTypeEngin}' .");
    }

    // Affichage des labels pour la disponibilité du véhicule
    if (!dtEngins.Columns.Contains("Dispo"))
    {
        dtEngins.Columns.Add("Dispo", typeof(string)); // Ajout de la colonne Dispo dans le DataTable
    }
    if (!dtEngins.Columns.Contains("MotifIndispo"))
    {
        dtEngins.Columns.Add("MotifIndispo", typeof(string)); // Idem Ajout MotifIndispo
    }

    foreach (DataRow row in dtEngins.Rows)
    {
        bool enMission = row.Field<long>("enMission") == 1; // si enMission vaut 1 alors true
        bool enPanne = row.Field<long>("enPanne") == 1;

        if (!enMission && !enPanne) // tout 2 false alors Engin disponible
        {
            row["Dispo"] = "Oui";
            row["MotifIndispo"] = ""; // Pas de motif d'indisponibilité
        }
        else // dans le cas où au moins l'un des 2 est true
        {
            row["Dispo"] = "Non";
            List<string> motifs = new List<string>();
            if (enMission)
            {
                motifs.Add("Motif : En mission");
            }
            if (enPanne)
            {
                motifs.Add("Motif : En panne");
            }
            row["MotifIndispo"] = string.Join(" et ", motifs);
        }
    }

    // Comme le nombre d'Engins est écrit dans un tableau on passe la ligne 1/n à chaque changement

```

72 %

Aucun problème détecté

```
    // Compte le nombre d'Engins et écrit dans un label en bas de la page 1/n à chaque changement
    if (!dtEngins.Columns.Contains("Index"))
    {
        dtEngins.Columns.Add("Index", typeof(string));
    }

    int total = dtEngins.Rows.Count;
    int i = 0;
    foreach (DataRow row in dtEngins.Rows)
    {
        i++;
        row["Index"] = i.ToString() + " / " + total.ToString();
    }
}
else
{
    MessageBox.Show("table Engin pas chargée dans le DataSet", "Erreur Interne", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}

bsEngins.DataSource = dtEngins; // Lie la DataTable filtré au BindingSource

// Liaison des contrôles individuels au BindingSource
txtNumero.DataBindings.Clear();
txtReception.DataBindings.Clear();
pbEnginImage.DataBindings.Clear(); // synchro picturebox avec bindingsource
lblOuiNon.DataBindings.Clear(); // label Etat disponibilité
lblMotif.DataBindings.Clear(); // label Motif Non disponibilité
lblIndex.DataBindings.Clear(); // label index page Engin dans la table
lblNomEngin.DataBindings.Clear(); // lbl qui affiche le nom complet de l'engin

txtNumero.DataBindings.Add("Text", bsEngins, "matricule");
txtReception.DataBindings.Add("Text", bsEngins, "dateReception");
pbEnginImage.DataBindings.Add("Image", bsEngins, "ImageEngin", true, DataSourceUpdateMode.OnPropertyChanged);
lblOuiNon.DataBindings.Add("Text", bsEngins, "Dispo");
lblMotif.DataBindings.Add("Text", bsEngins, "MotifIndispo");
lblIndex.DataBindings.Add("Text", bsEngins, "Index");
lblNomEngin.DataBindings.Add("Text", bsEngins, "NomTypeEngin");

if (dgvCaserne != null)
{
    dgvCaserne.DataSource = bsEngins;
    dgvCaserne.Visible = false;
}

if (dtEngins.Rows.Count == 0)
{
    MeccanoRoy.Show("Aucun engin trouvé pour cette caserne", "Information", MeccanoRoyButtons.OK, MeccanoRoyIcon.Information);
}
```

```
FormCreationMission    ↴ FormCreationMission.gestionEngins
}
}

if (dtEngins.Rows.Count == 0)
{
    MessageBox.Show("Aucun engin trouvé pour cette caserne.", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    bsEngins.MoveFirst(); // se replace toujours au debut quand on change de "table" et donc de Caserne
}

// Chercher dans la table TypeEngin les noms complets des Engins pour les afficher dans le label correspondant
if (ds.Tables.Contains("TypeEngin"))
{
}

}

1 référence
private void pbmin_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {
        bsEngins.MoveFirst();
    }
}

1 référence
private void pbavant_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {
        bsEngins.MovePrevious();
    }
}

1 référence
private void pbapres_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {
        bsEngins.MoveNext();
    }
}

1 référence
private void pbmax_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {

}
72 %    ↴ Aucun problème détecté
```

```
C# FormCreationMission
    }
}

1 référence
private void pbapres_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {
        bsEngins.MoveNext();
    }
}

1 référence
private void pbmax_Click(object sender, EventArgs e)
{
    if (bsEngins.Count > 0)
    {
        bsEngins.MoveLast();
    }
}

1 référence
private void cboChoix_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true; // Empêche la saisie de texte dans le ComboBox
}

1 référence
private void txtNumero_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true; // Empêche la saisie de texte dans le TextBox
}

1 référence
private void txtReception_KeyPress(object sender, KeyPressEventArgs e)
{
    e.Handled = true; // Empêche la saisie de texte dans le TextBox
}
```