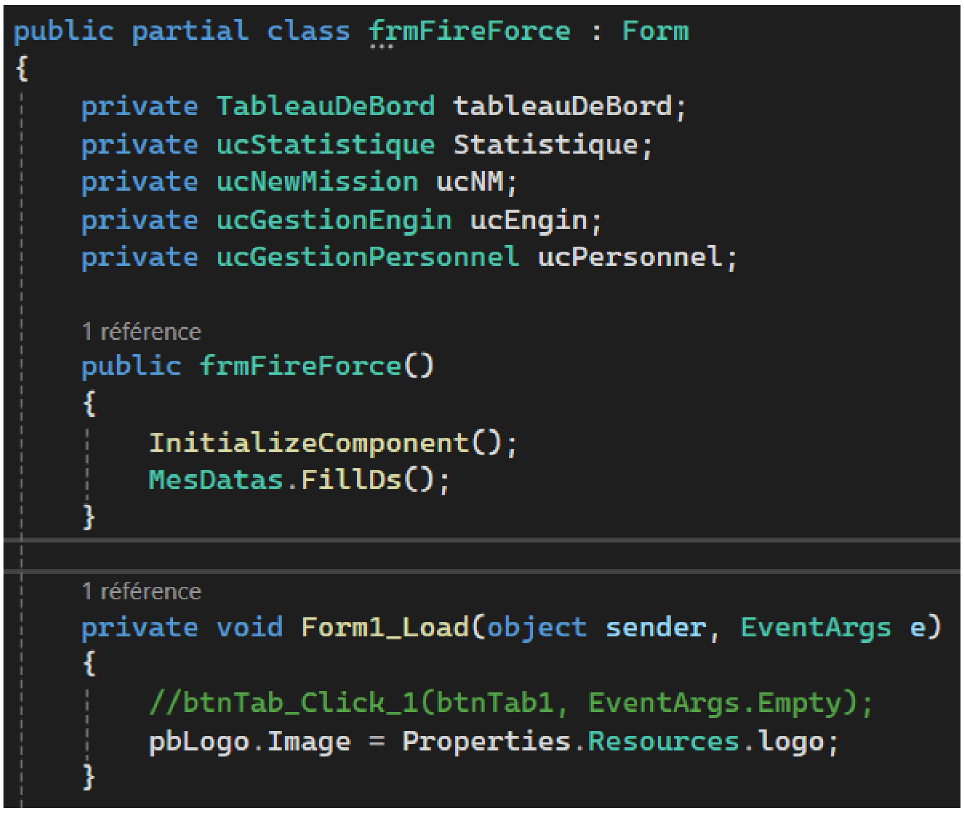
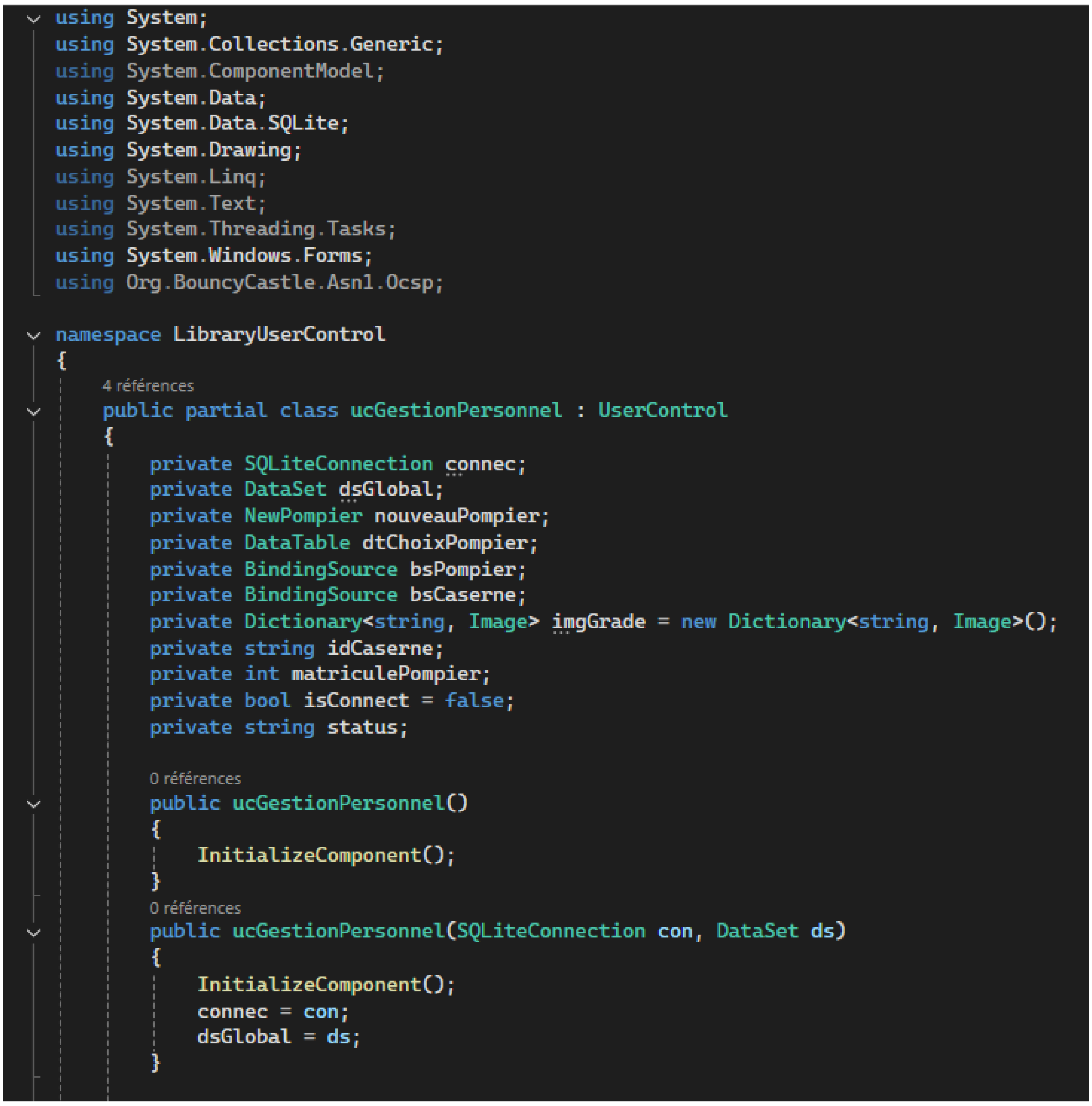
Déclaration des variables globales du formulaire principale :

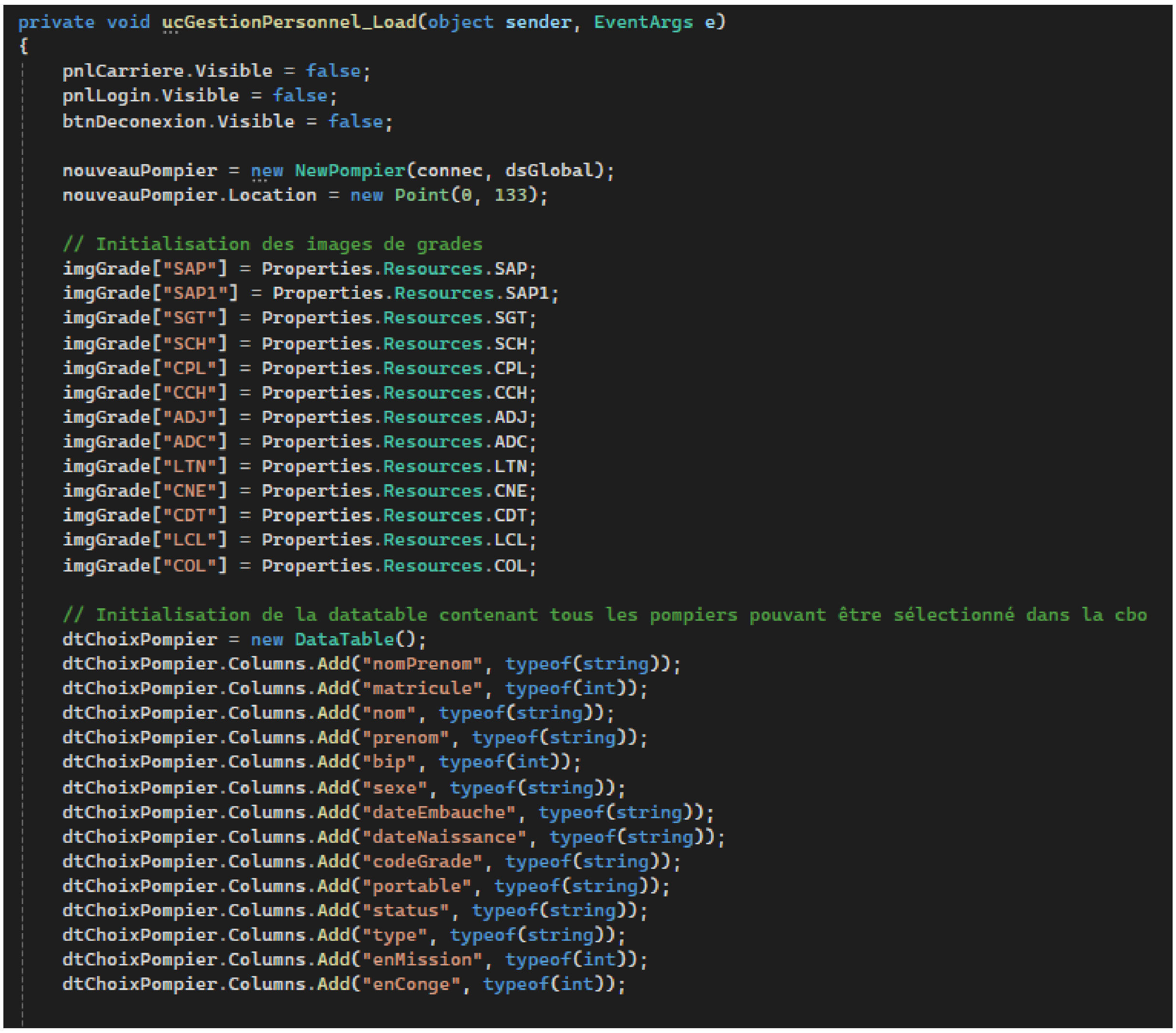


```
public partial class frmFireForce : Form
{
    private TableauDeBord tableauDeBord;
    private ucStatistique Statistique;
    private ucNewMission ucNM;
    private ucGestionEngin ucEngin;
    private ucGestionPersonnel ucPersonnel;

    1 référence
    public frmFireForce()
    {
        InitializeComponent();
        MesDatas.FillDs();
    }

    1 référence
    private void Form1_Load(object sender, EventArgs e)
    {
        //btnTab_Click_1(btnTab1, EventArgs.Empty);
        pbLogo.Image = Properties.Resources.logo;
    }
```

code complet du UserControl dans lequel vous permettez le choix d'un pompier, puis affichez les informations complètes concernant un pompier :

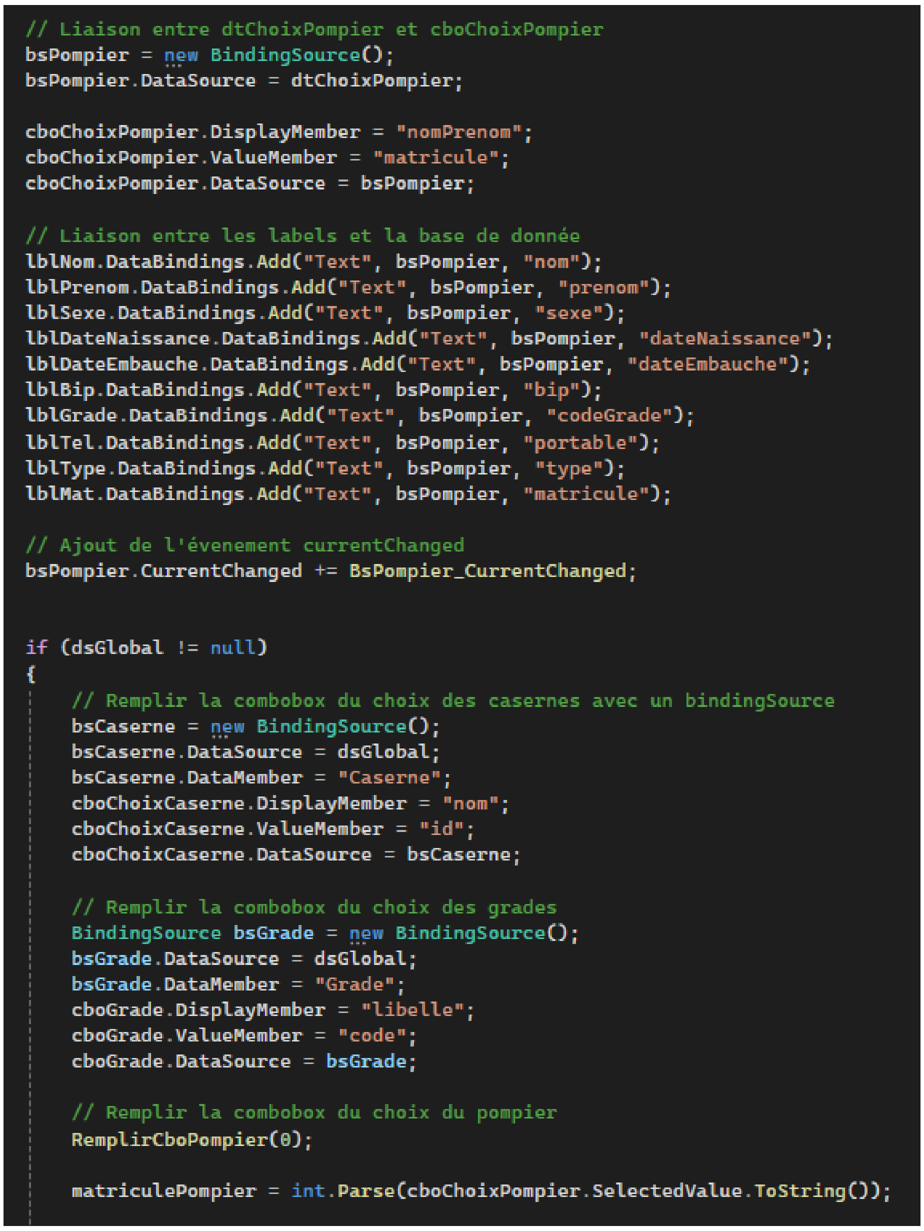


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Org.BouncyCastle.Asn1.Ocsp;

namespace LibraryUserControl
{
    4 références
    public partial class ucGestionPersonnel : UserControl
    {
        private SQLiteConnection connec;
        private DataSet dsGlobal;
        private NewPompier nouveauPompier;
        private DataTable dtChoixPompier;
        private BindingSource bsPompier;
        private BindingSource bsCaserne;
        private Dictionary<string, Image> imgGrade = new Dictionary<string, Image>();
        private string idCaserne;
        private int matriculePompier;
        private bool isConnect = false;
        private string status;

        0 références
        public ucGestionPersonnel()
        {
            InitializeComponent();
        }
        0 références
        public ucGestionPersonnel(SQLiteConnection con, DataSet ds)
        {
            InitializeComponent();
            connec = con;
            dsGlobal = ds;
        }
```



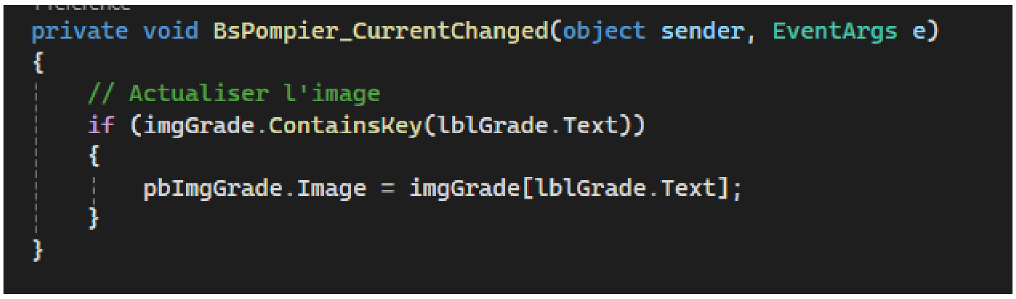
```
private void ucGestionPersonnel_Load(object sender, EventArgs e)
{
    pnlCarriere.Visible = false;
    pnlLogin.Visible = false;
    btnDeconexion.Visible = false;

    nouveauPompier = new NewPompier(connec, dsGlobal);
    nouveauPompier.Location = new Point(0, 133);

    // Initialisation des images de grades
    imgGrade["SAP"] = Properties.Resources.SAP;
    imgGrade["SAP1"] = Properties.Resources.SAP1;
    imgGrade["SGT"] = Properties.Resources.SGT;
    imgGrade["SCH"] = Properties.Resources.SCH;
    imgGrade["CPL"] = Properties.Resources.CPL;
    imgGrade["CCH"] = Properties.Resources.CCH;
    imgGrade["ADJ"] = Properties.Resources.ADJ;
    imgGrade["ADC"] = Properties.Resources.ADC;
    imgGrade["LTN"] = Properties.Resources.LTN;
    imgGrade["CNE"] = Properties.Resources.CNE;
    imgGrade["CDT"] = Properties.Resources.CDT;
    imgGrade["LCL"] = Properties.Resources.LCL;
    imgGrade["COL"] = Properties.Resources.COL;

    // Initialisation de la datatable contenant tous les pompiers pouvant être sélectionné dans la cbo
    dtChoixPompier = new DataTable();
    dtChoixPompier.Columns.Add("nomPrenom", typeof(string));
    dtChoixPompier.Columns.Add("matricule", typeof(int));
    dtChoixPompier.Columns.Add("nom", typeof(string));
    dtChoixPompier.Columns.Add("prenom", typeof(string));
    dtChoixPompier.Columns.Add("bip", typeof(int));
    dtChoixPompier.Columns.Add("sexe", typeof(string));
    dtChoixPompier.Columns.Add("dateEmbauche", typeof(string));
    dtChoixPompier.Columns.Add("dateNaissance", typeof(string));
    dtChoixPompier.Columns.Add("codeGrade", typeof(string));
    dtChoixPompier.Columns.Add("portable", typeof(string));
    dtChoixPompier.Columns.Add("status", typeof(string));
    dtChoixPompier.Columns.Add("type", typeof(string));
    dtChoixPompier.Columns.Add("enMission", typeof(int));
    dtChoixPompier.Columns.Add("enConge", typeof(int));
```

// Stockage des images de grades dans un dictionnaire

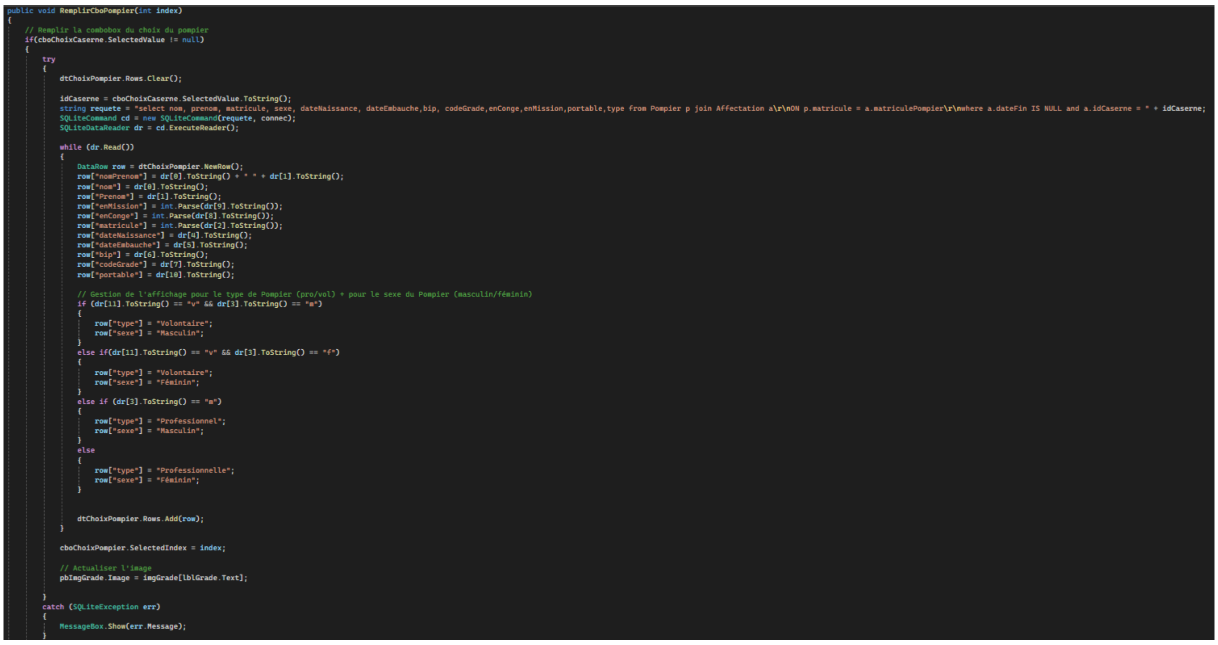


```csharp
// Liaison entre dtChoixPompier et cboChoixPompier
bsPompier = new BindingSource();
bsPompier.DataSource = dtChoixPompier;

cboChoixPompier.DisplayMember = "nomPrenom";
cboChoixPompier.ValueMember = "matricule";
cboChoixPompier.DataSource = bsPompier;

// Liaison entre les labels et la base de donnée
lblNom.DataBindings.Add("Text", bsPompier, "nom");
lblPrenom.DataBindings.Add("Text", bsPompier, "prenom");
lblSexe.DataBindings.Add("Text", bsPompier, "sexe");
lblDateNaissance.DataBindings.Add("Text", bsPompier, "dateNaissance");
lblDateEmbauche.DataBindings.Add("Text", bsPompier, "dateEmbauche");
lblBip.DataBindings.Add("Text", bsPompier, "bip");
lblGrade.DataBindings.Add("Text", bsPompier, "codeGrade");
lblTel.DataBindings.Add("Text", bsPompier, "portable");
lblType.DataBindings.Add("Text", bsPompier, "type");
lblMat.DataBindings.Add("Text", bsPompier, "matricule");

// Ajout de l'évenement currentChanged
bsPompier.CurrentChanged += BsPompier_CurrentChanged;


if (dsGlobal != null)
{
    // Remplir la combobox du choix des casernes avec un bindingSource
    bsCaserne = new BindingSource();
    bsCaserne.DataSource = dsGlobal;
    bsCaserne.DataMember = "Caserne";
    cboChoixCaserne.DisplayMember = "nom";
    cboChoixCaserne.ValueMember = "id";
    cboChoixCaserne.DataSource = bsCaserne;

    // Remplir la combobox du choix des grades
    BindingSource bsGrade = new BindingSource();
    bsGrade.DataSource = dsGlobal;
    bsGrade.DataMember = "Grade";
    cboGrade.DisplayMember = "libelle";
    cboGrade.ValueMember = "code";
    cboGrade.DataSource = bsGrade;

    // Remplir la combobox du choix du pompier
    RemplirCboPompier(0);

    matriculePompier = int.Parse(cboChoixPompier.SelectedValue.ToString());
```

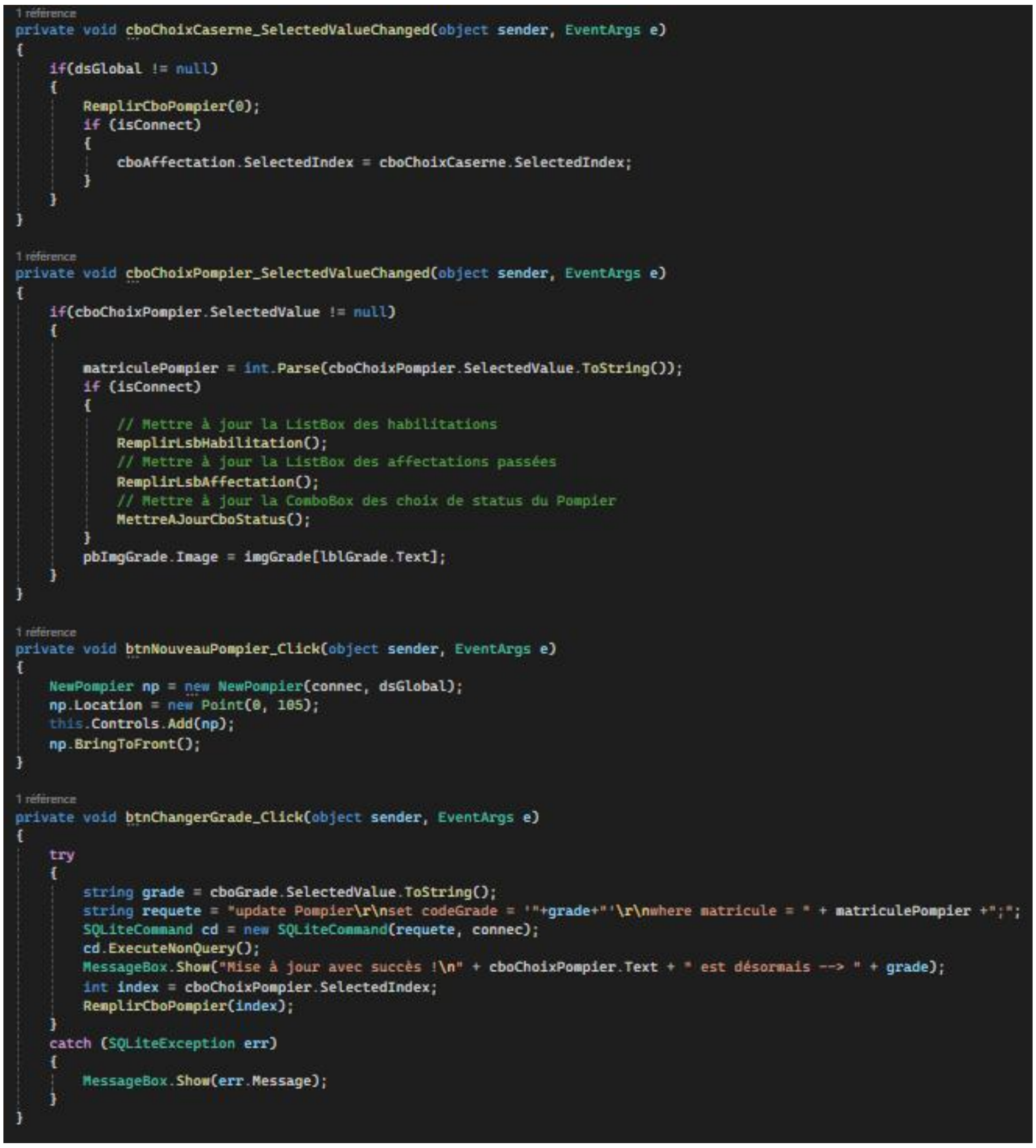
J’utilise des binding sources relié à une datatable qui est remplie en mode connecté pour mettre automatiquement à jour les infos des pompiers



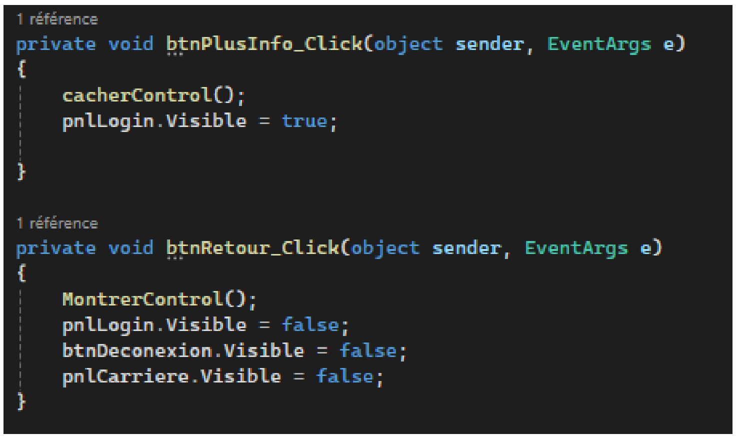
```csharp
private void BsPompier_CurrentChanged(object sender, EventArgs e)
{
    // Actualiser l'image
    if (imgGrade.ContainsKey(lblGrade.Text))
    {
        pbImgGrade.Image = imgGrade[lblGrade.Text];
    }
}
```



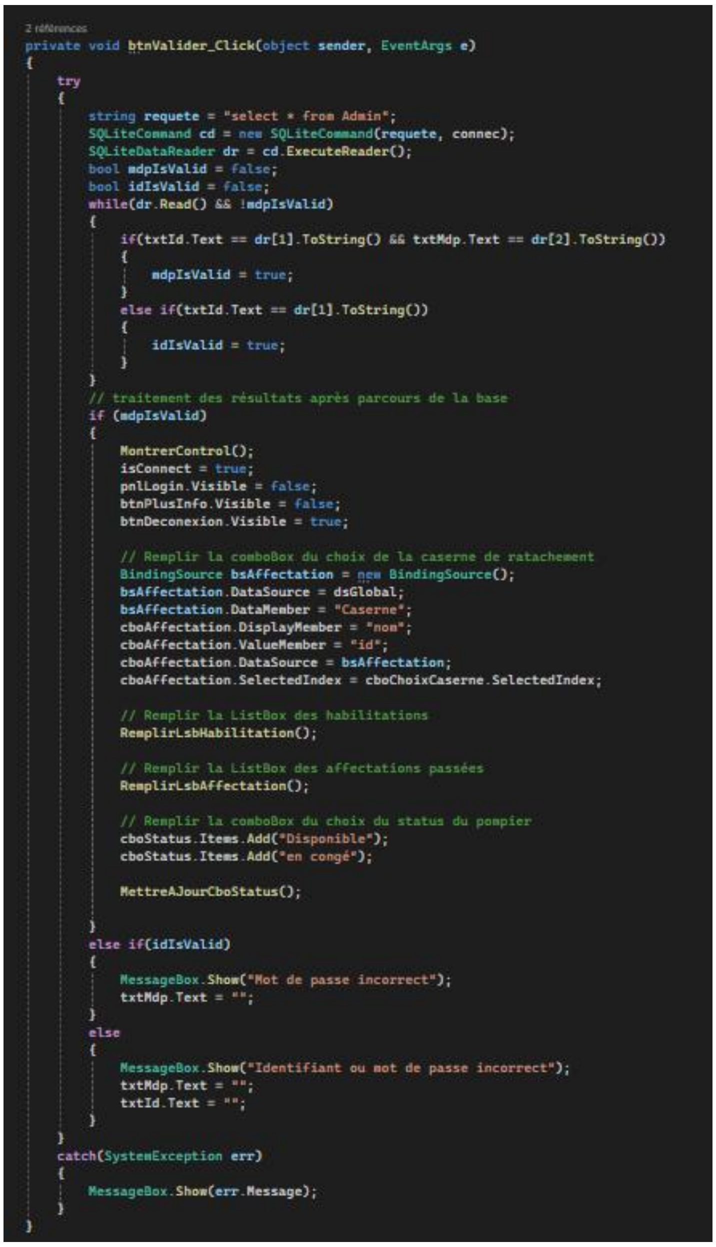
```csharp
public void RemplirCboPompier(int index)
{
    // Remplir la combobox du choix du pompier
    if(cboChoixCaserne.SelectedValue != null)
    {
        try
        {
            dtChoixPompier.Rows.Clear();

            idCaserne = cboChoixCaserne.SelectedValue.ToString();
            string requete = "select nom, prenom, matricule, sexe, dateNaissance, dateEmbauche,bip, codeGrade,enConge,enMission,portable,type from Pompier p join Affectation a\r\nON p.matricule = a.matriculePompier\r\nwhere a.dateFin IS NULL and a.idCaserne = " + idCaserne;
            SQLiteCommand cd = new SQLiteCommand(requete, connec);
            SQLiteDataReader dr = cd.ExecuteReader();

            while (dr.Read())
            {
                DataRow row = dtChoixPompier.NewRow();
                row["nomPrenom"] = dr[0].ToString() + " " + dr[1].ToString();
                row["nom"] = dr[0].ToString();
                row["Prenom"] = dr[1].ToString();
                row["enMission"] = int.Parse(dr[9].ToString());
                row["enConge"] = int.Parse(dr[8].ToString());
                row["matricule"] = int.Parse(dr[2].ToString());
                row["dateNaissance"] = dr[4].ToString();
                row["dateEmbauche"] = dr[5].ToString();
                row["bip"] = dr[6].ToString();
                row["codeGrade"] = dr[7].ToString();
                row["portable"] = dr[10].ToString();

                // Gestion de l'affichage pour le type de Pompier (pro/vol) + pour le sexe du Pompier (masculin/féminin)
                if (dr[11].ToString() == "v" && dr[3].ToString() == "m")
                {
                    row["type"] = "Volontaire";
                    row["sexe"] = "Masculin";
                }
                else if(dr[11].ToString() == "v" && dr[3].ToString() == "f")
                {
                    row["type"] = "Volontaire";
                    row["sexe"] = "Féminin";
                }
                else if (dr[3].ToString() == "m")
                {
                    row["type"] = "Professionnel";
                    row["sexe"] = "Masculin";
                }
                else
                {
                    row["type"] = "Professionnelle";
                    row["sexe"] = "Féminin";
                }


                dtChoixPompier.Rows.Add(row);
            }

            cboChoixPompier.SelectedIndex = index;

            // Actualiser l'image
            pbImgGrade.Image = imgGrade[lblGrade.Text];

        }
        catch (SQLiteException err)
        {
            MessageBox.Show(err.Message);
        }
```



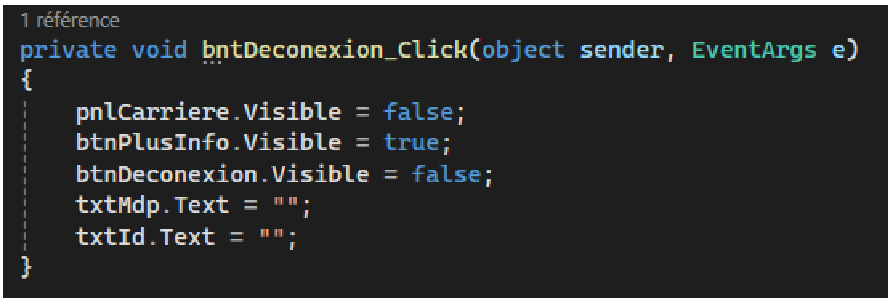
```csharp
1 référence
private void cboChoixCaserne_SelectedValueChanged(object sender, EventArgs e)
{
    if(dsGlobal != null)
    {
        RemplirCboPompier(0);
        if (isConnect)
        {
            cboAffectation.SelectedIndex = cboChoixCaserne.SelectedIndex;
        }
    }
}

1 référence
private void cboChoixPompier_SelectedValueChanged(object sender, EventArgs e)
{
    if(cboChoixPompier.SelectedValue != null)
    {
        matriculePompier = int.Parse(cboChoixPompier.SelectedValue.ToString());
        if (isConnect)
        {
            // Mettre à jour la ListBox des habilitations
            RemplirLsbHabilitation();
            // Mettre à jour la ListBox des affectations passées
            RemplirLsbAffectation();
            // Mettre à jour la ComboBox des choix de status du Pompier
            MettreAJourCboStatus();
        }
        pbImgGrade.Image = imgGrade[lblGrade.Text];
    }
}

1 référence
private void btnNouveauPompier_Click(object sender, EventArgs e)
{
    NewPompier np = new NewPompier(connec, dsGlobal);
    np.Location = new Point(0, 105);
    this.Controls.Add(np);
    np.BringToFront();
}

1 référence
private void btnChangerGrade_Click(object sender, EventArgs e)
{
    try
    {
        string grade = cboGrade.SelectedValue.ToString();
        string requete = "update Pompier\r\nset codeGrade = '"+grade+"'\r\nwhere matricule = " + matriculePompier +";";
        SQLiteCommand cd = new SQLiteCommand(requete, connec);
        cd.ExecuteNonQuery();
        MessageBox.Show("Mise à jour avec succès !\n" + cboChoixPompier.Text + " est désormais --> " + grade);
        int index = cboChoixPompier.SelectedIndex;
        RemplirCboPompier(index);
    }
    catch (SQLiteException err)
    {
        MessageBox.Show(err.Message);
    }
}
```



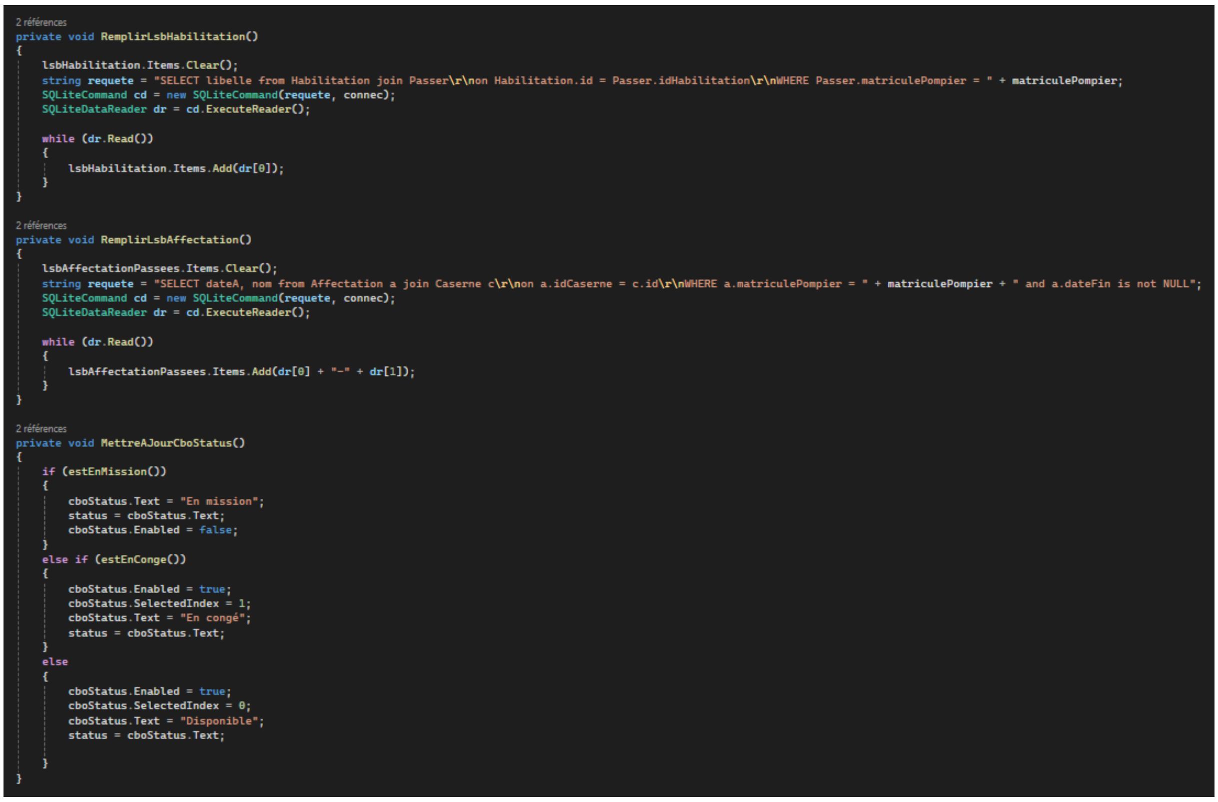
```
1 référence
private void btnPlusInfo_Click(object sender, EventArgs e)
{
    cacherControl();
    pnlLogin.Visible = true;

}

1 référence
private void btnRetour_Click(object sender, EventArgs e)
{
    MontrerControl();
    pnlLogin.Visible = false;
    btnDeconexion.Visible = false;
    pnlCarriere.Visible = false;
}
```



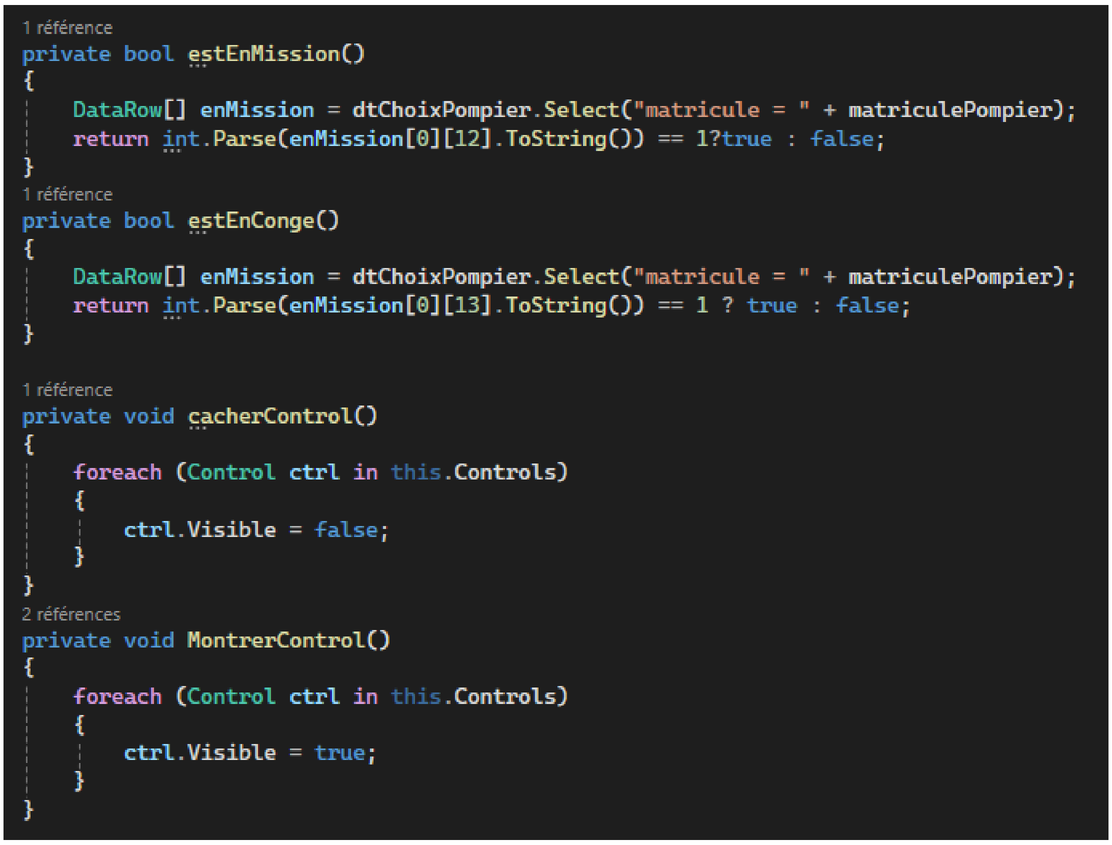
```
2 références
private void btnValider_Click(object sender, EventArgs e)
{
    try
    {
        string requete = "select * from Admin";
        SQLiteCommand cd = new SQLiteCommand(requete, connec);
        SQLiteDataReader dr = cd.ExecuteReader();
        bool mdpIsValid = false;
        bool idIsValid = false;
        while(dr.Read() && !mdpIsValid)
        {
            if(txtId.Text == dr[1].ToString() && txtMdp.Text == dr[2].ToString())
            {
                mdpIsValid = true;
            }
            else if(txtId.Text == dr[1].ToString())
            {
                idIsValid = true;
            }
        }
        // traitement des résultats après parcours de la base
        if (mdpIsValid)
        {
            MontrerControl();
            isConnect = true;
            pnlLogin.Visible = false;
            btnPlusInfo.Visible = false;
            btnDeconexion.Visible = true;

            // Remplir la comboBox du choix de la caserne de ratachement
            BindingSource bsAffectation = new BindingSource();
            bsAffectation.DataSource = dsGlobal;
            bsAffectation.DataMember = "Caserne";
            cboAffectation.DisplayMember = "nom";
            cboAffectation.ValueMember = "id";
            cboAffectation.DataSource = bsAffectation;
            cboAffectation.SelectedIndex = cboChoixCaserne.SelectedIndex;

            // Remplir la ListBox des habilitations
            RemplirLsbHabilitation();

            // Remplir la ListBox des affectations passées
            RemplirLsbAffectation();

            // Remplir la comboBox du choix du status du pompier
            cboStatus.Items.Add("Disponible");
            cboStatus.Items.Add("en congé");

            MettreAJourCboStatus();

        }
        else if(idIsValid)
        {
            MessageBox.Show("Mot de passe incorrect");
            txtMdp.Text = "";
        }
        else
        {
            MessageBox.Show("Identifiant ou mot de passe incorrect");
            txtMdp.Text = "";
            txtId.Text = "";
        }
    }
    catch(SystemException err)
    {
        MessageBox.Show(err.Message);
    }
}
```



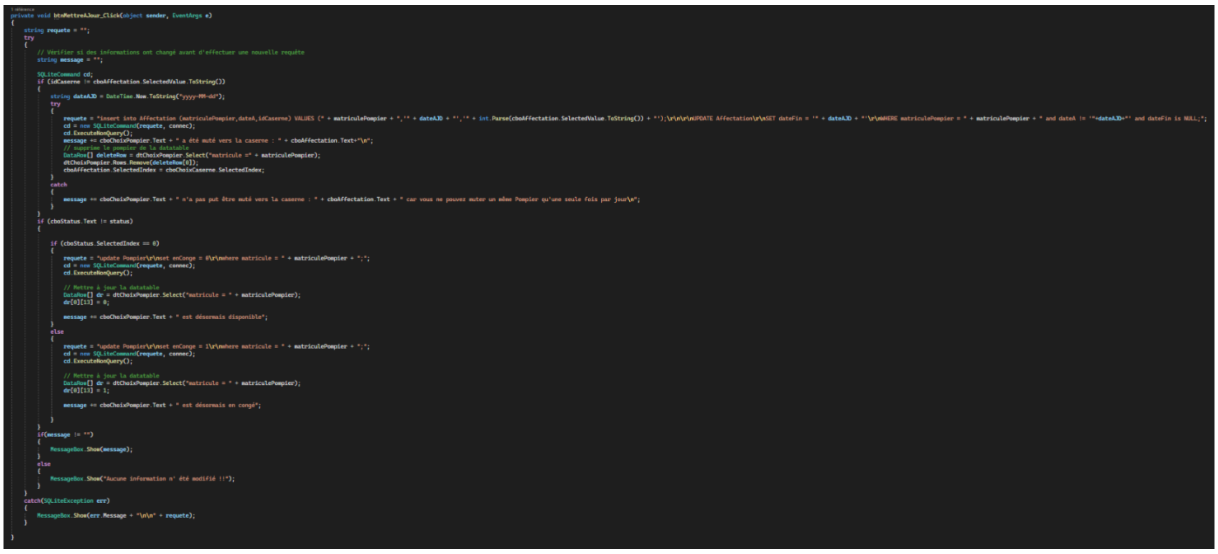
```
1 référence
private void bntDeconexion_Click(object sender, EventArgs e)
{
    pnlCarriere.Visible = false;
    btnPlusInfo.Visible = true;
    btnDeconexion.Visible = false;
    txtMdp.Text = "";
    txtId.Text = "";
}
```



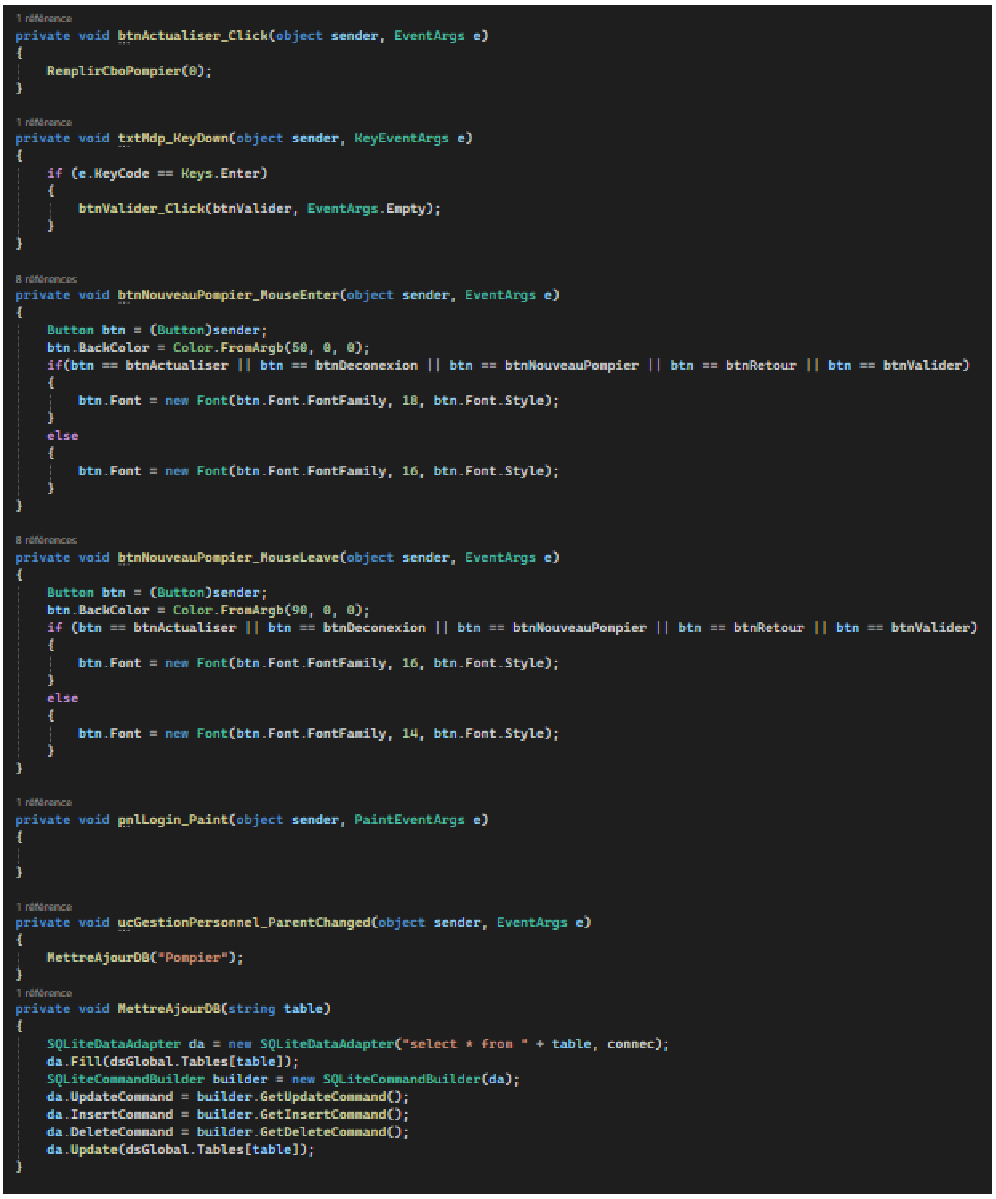
```
2 références
private void RemplirLsbHabilitation()
{
    lsbHabilitation.Items.Clear();
    string requete = "SELECT libelle from Habilitation join Passer\r\non Habilitation.id = Passer.idHabilitation\r\nWHERE Passer.matriculePompier = " + matriculePompier;
    SQLiteCommand cd = new SQLiteCommand(requete, connec);
    SQLiteDataReader dr = cd.ExecuteReader();

    while (dr.Read())
    {
        lsbHabilitation.Items.Add(dr[0]);
    }
}

2 références
private void RemplirLsbAffectation()
{
    lsbAffectationPassees.Items.Clear();
    string requete = "SELECT dateA, nom from Affectation a join Caserne c\r\non a.idCaserne = c.id\r\nWHERE a.matriculePompier = " + matriculePompier + " and a.dateFin is not NULL";
    SQLiteCommand cd = new SQLiteCommand(requete, connec);
    SQLiteDataReader dr = cd.ExecuteReader();

    while (dr.Read())
    {
        lsbAffectationPassees.Items.Add(dr[0] + "-" + dr[1]);
    }
}

2 références
private void MettreAJourCboStatus()
{
    if (estEnMission())
    {
        cboStatus.Text = "En mission";
        status = cboStatus.Text;
        cboStatus.Enabled = false;
    }
    else if (estEnConge())
    {
        cboStatus.Enabled = true;
        cboStatus.SelectedIndex = 1;
        cboStatus.Text = "En congé";
        status = cboStatus.Text;
    }
    else
    {
        cboStatus.Enabled = true;
        cboStatus.SelectedIndex = 0;
        cboStatus.Text = "Disponible";
        status = cboStatus.Text;

    }
}
```



```
1 référence
private bool estEnMission()
{
    DataRow[] enMission = dtChoixPompier.Select("matricule = " + matriculePompier);
    return int.Parse(enMission[0][12].ToString()) == 1?true : false;
}
1 référence
private bool estEnConge()
{
    DataRow[] enMission = dtChoixPompier.Select("matricule = " + matriculePompier);
    return int.Parse(enMission[0][13].ToString()) == 1 ? true : false;
}

1 référence
private void cacherControl()
{
    foreach (Control ctrl in this.Controls)
    {
        ctrl.Visible = false;
    }
}
2 références
private void MontrerControl()
{
    foreach (Control ctrl in this.Controls)
    {
        ctrl.Visible = true;
    }
}
```



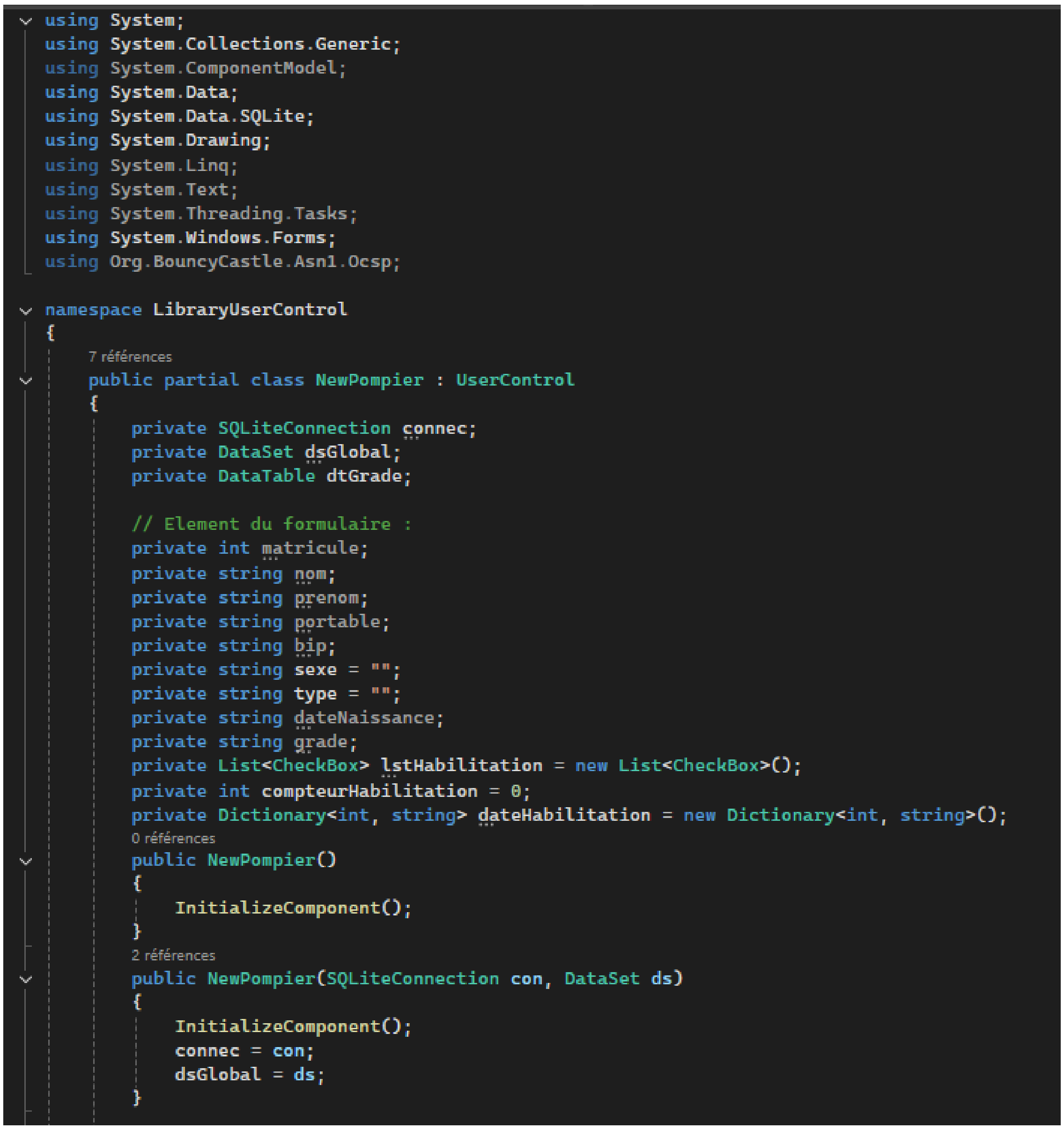
```
1 référence
private void btnMettreAJour_Click(object sender, EventArgs e)
{
    string requete = "";
    try
    {
        // Vérifier si des informations ont changé avant d'effectuer une nouvelle requête
        string message = "";

        SQLiteCommand cd;
        if (idCaserne != cboAffectation.SelectedValue.ToString())
        {
            string dateAJD = DateTime.Now.ToString("yyyy-MM-dd");
            try
            {
                requete = "insert into Affectation (matriculePompier,dateA,idCaserne) VALUES (" + matriculePompier + ",'" + dateAJD + "','" + int.Parse(cboAffectation.SelectedValue.ToString()) + "');\r\n\r\nUPDATE Affectation\r\nSET dateFin = '" + dateAJD + "'\r\nWHERE matriculePompier = " + matriculePompier + " and dateA != '"+dateAJD+"' and dateFin is NULL;";
                cd = new SQLiteCommand(requete, connec);
                cd.ExecuteNonQuery();
                message += cboChoixPompier.Text + " a été muté vers la caserne : " + cboAffectation.Text+"\n";
                // supprime le pompier de la datatable
                DataRow[] deleteRow = dtChoixPompier.Select("matricule =" + matriculePompier);
                dtChoixPompier.Rows.Remove(deleteRow[0]);
                cboAffectation.SelectedIndex = cboChoixCaserne.SelectedIndex;
            }
            catch
            {
                message += cboChoixPompier.Text + " n'a pas put être muté vers la caserne : " + cboAffectation.Text + " car vous ne pouvez muter un même Pompier qu'une seule fois par jour\n";
            }
        }
        if (cboStatus.Text != status)
        {

            if (cboStatus.SelectedIndex == 0)
            {
                requete = "update Pompier\r\nset enConge = 0\r\nwhere matricule = " + matriculePompier + ";";
                cd = new SQLiteCommand(requete, connec);
                cd.ExecuteNonQuery();

                // Mettre à jour la datatable
                DataRow[] dr = dtChoixPompier.Select("matricule = " + matriculePompier);
                dr[0][13] = 0;

                message += cboChoixPompier.Text + " est désormais disponible";
            }
            else
            {
                requete = "update Pompier\r\nset enConge = 1\r\nwhere matricule = " + matriculePompier + ";";
                cd = new SQLiteCommand(requete, connec);
                cd.ExecuteNonQuery();

                // Mettre à jour la datatable
                DataRow[] dr = dtChoixPompier.Select("matricule = " + matriculePompier);
                dr[0][13] = 1;

                message += cboChoixPompier.Text + " est désormais en congé";

            }
        }
        if(message != "")
        {
            MessageBox.Show(message);
        }
        else
        {
            MessageBox.Show("Aucune information n' été modifié !!");
        }
    }
    catch(SQLiteException err)
    {
        MessageBox.Show(err.Message + "\n\n" + requete);
    }

}
```



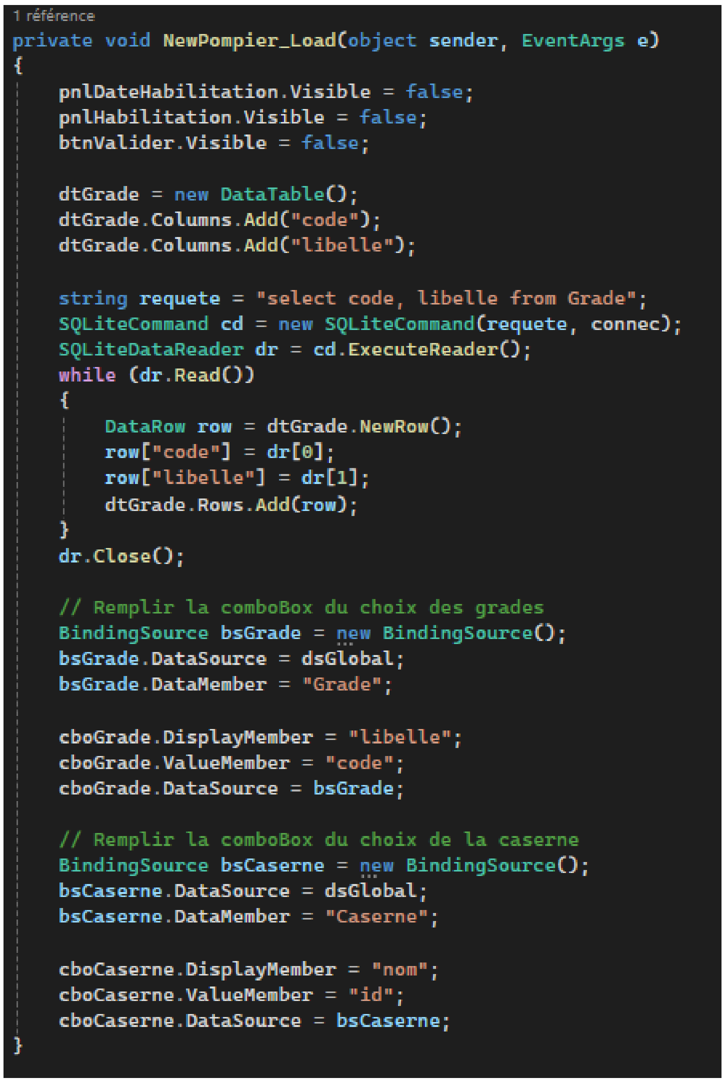
```csharp
1 référence
private void btnActualiser_Click(object sender, EventArgs e)
{
    RemplirCboPompier(0);
}

1 référence
private void txtMdp_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        btnValider_Click(btnValider, EventArgs.Empty);
    }
}

8 références
private void btnNouveauPompier_MouseEnter(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.FromArgb(50, 0, 0);
    if(btn == btnActualiser || btn == btnDeconexion || btn == btnNouveauPompier || btn == btnRetour || btn == btnValider)
    {
        btn.Font = new Font(btn.Font.FontFamily, 18, btn.Font.Style);
    }
    else
    {
        btn.Font = new Font(btn.Font.FontFamily, 16, btn.Font.Style);
    }
}

8 références
private void btnNouveauPompier_MouseLeave(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.FromArgb(90, 0, 0);
    if (btn == btnActualiser || btn == btnDeconexion || btn == btnNouveauPompier || btn == btnRetour || btn == btnValider)
    {
        btn.Font = new Font(btn.Font.FontFamily, 16, btn.Font.Style);
    }
    else
    {
        btn.Font = new Font(btn.Font.FontFamily, 14, btn.Font.Style);
    }
}

1 référence
private void pnlLogin_Paint(object sender, PaintEventArgs e)
{

}

1 référence
private void ucGestionPersonnel_ParentChanged(object sender, EventArgs e)
{
    MettreAjourDB("Pompier");
}
1 référence
private void MettreAjourDB(string table)
{
    SQLiteDataAdapter da = new SQLiteDataAdapter("select * from " + table, connec);
    da.Fill(dsGlobal.Tables[table]);
    SQLiteCommandBuilder builder = new SQLiteCommandBuilder(da);
    da.UpdateCommand = builder.GetUpdateCommand();
    da.InsertCommand = builder.GetInsertCommand();
    da.DeleteCommand = builder.GetDeleteCommand();
    da.Update(dsGlobal.Tables[table]);
}
```

Code complet du User Control que vous jugez le plus intéressant :

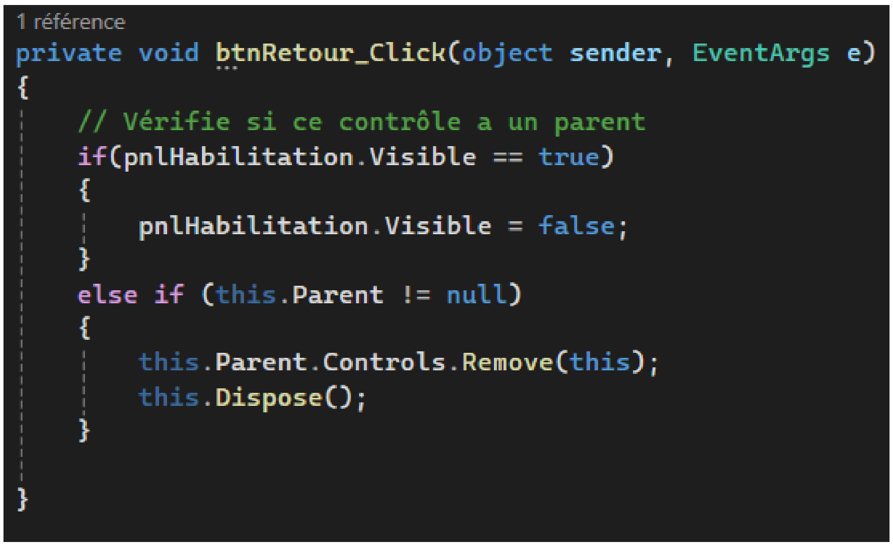
Le code que nous avons jugé le plus intéressant est celui permettant de créer un nouveau pompier car c’est celui qui possède l’interface la plus développé avec le plus de fonctionnalité comme par exemple la possibilité d’accorder des habilitations aux nouveaux pompiers



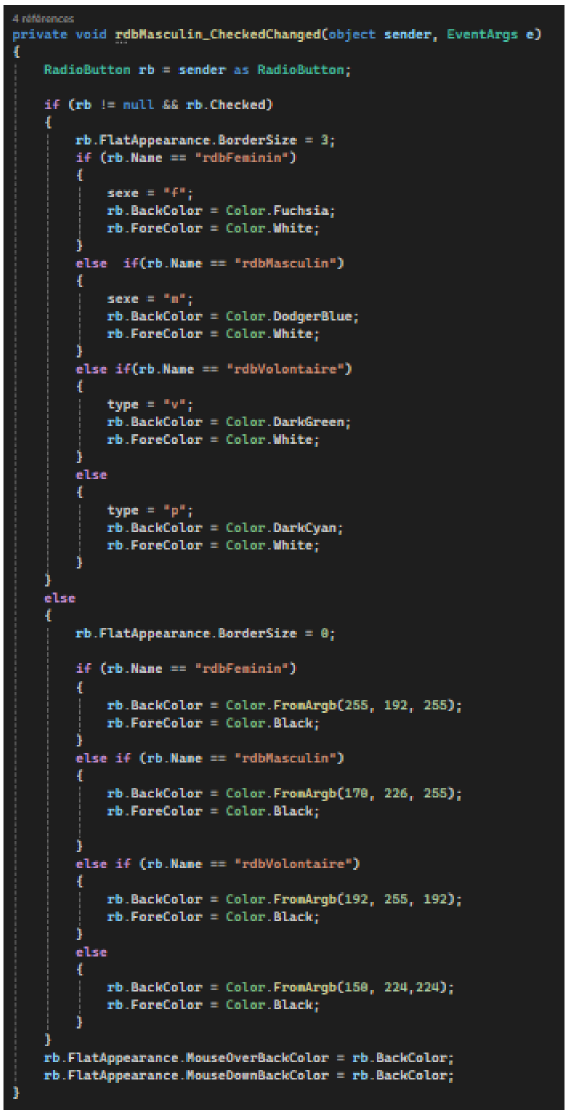
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Org.BouncyCastle.Asn1.Ocsp;

namespace LibraryUserControl
{
    7 références
    public partial class NewPompier : UserControl
    {
        private SQLiteConnection connec;
        private DataSet dsGlobal;
        private DataTable dtGrade;

        // Element du formulaire :
        private int matricule;
        private string nom;
        private string prenom;
        private string portable;
        private string bip;
        private string sexe = "";
        private string type = "";
        private string dateNaissance;
        private string grade;
        private List<CheckBox> lstHabilitation = new List<CheckBox>();
        private int compteurHabilitation = 0;
        private Dictionary<int, string> dateHabilitation = new Dictionary<int, string>();
        0 références
        public NewPompier()
        {
            InitializeComponent();
        }
        2 références
        public NewPompier(SQLiteConnection con, DataSet ds)
        {
            InitializeComponent();
            connec = con;
            dsGlobal = ds;
        }
```



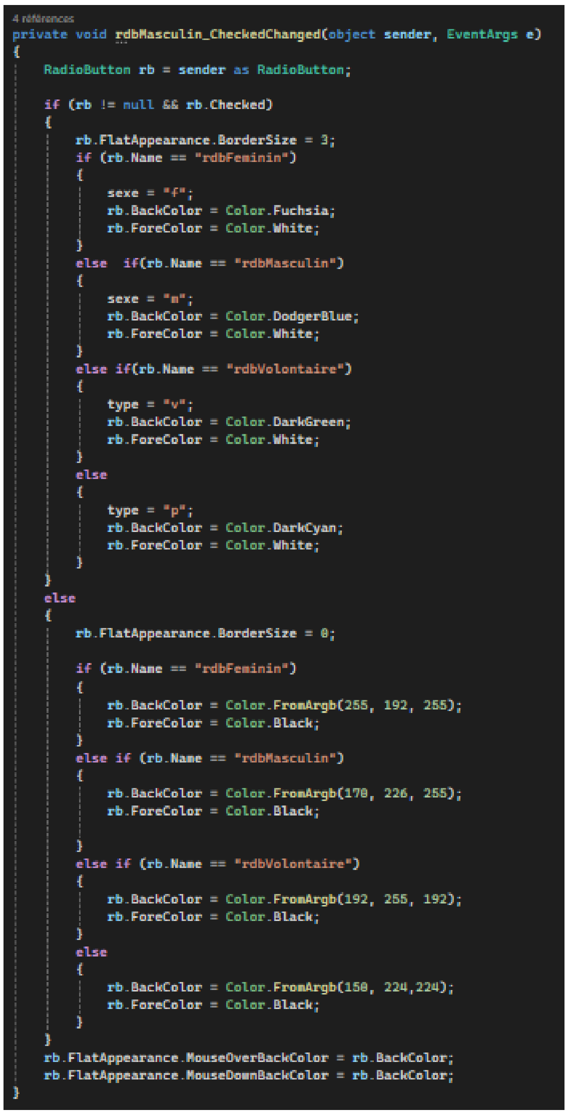
```
1 référence
private void NewPompier_Load(object sender, EventArgs e)
{
    pnlDateHabilitation.Visible = false;
    pnlHabilitation.Visible = false;
    btnValider.Visible = false;

    dtGrade = new DataTable();
    dtGrade.Columns.Add("code");
    dtGrade.Columns.Add("libelle");

    string requete = "select code, libelle from Grade";
    SQLiteCommand cd = new SQLiteCommand(requete, connec);
    SQLiteDataReader dr = cd.ExecuteReader();
    while (dr.Read())
    {
        DataRow row = dtGrade.NewRow();
        row["code"] = dr[0];
        row["libelle"] = dr[1];
        dtGrade.Rows.Add(row);
    }
    dr.Close();

    // Remplir la comboBox du choix des grades
    BindingSource bsGrade = new BindingSource();
    bsGrade.DataSource = dsGlobal;
    bsGrade.DataMember = "Grade";

    cboGrade.DisplayMember = "libelle";
    cboGrade.ValueMember = "code";
    cboGrade.DataSource = bsGrade;

    // Remplir la comboBox du choix de la caserne
    BindingSource bsCaserne = new BindingSource();
    bsCaserne.DataSource = dsGlobal;
    bsCaserne.DataMember = "Caserne";

    cboCaserne.DisplayMember = "nom";
    cboCaserne.ValueMember = "id";
    cboCaserne.DataSource = bsCaserne;
}
```



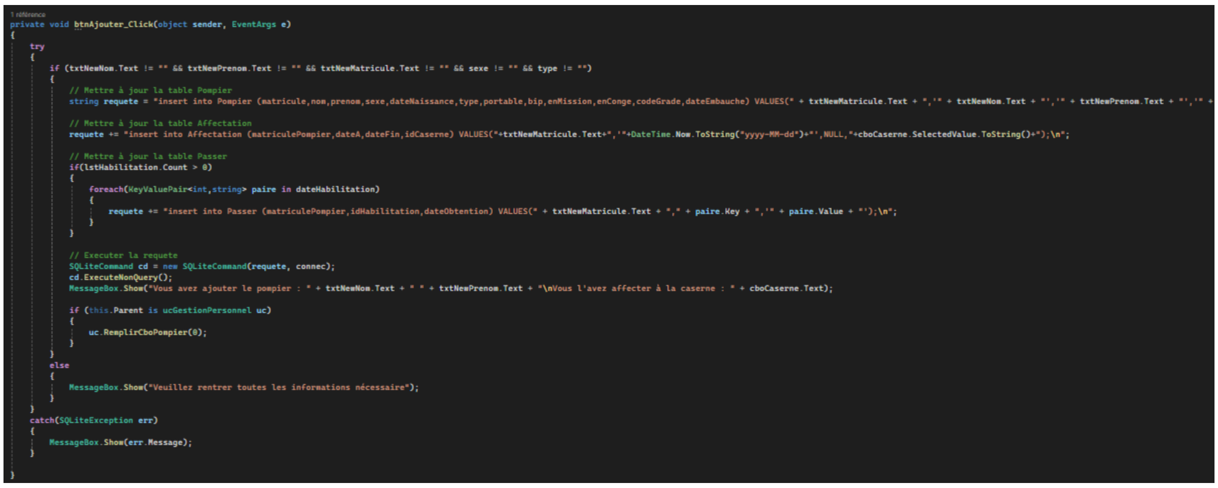
```
1 référence
private void btnRetour_Click(object sender, EventArgs e)
{
    // Vérifie si ce contrôle a un parent
    if(pnlHabilitation.Visible == true)
    {
        pnlHabilitation.Visible = false;
    }
    else if (this.Parent != null)
    {
        this.Parent.Controls.Remove(this);
        this.Dispose();
    }

}
```



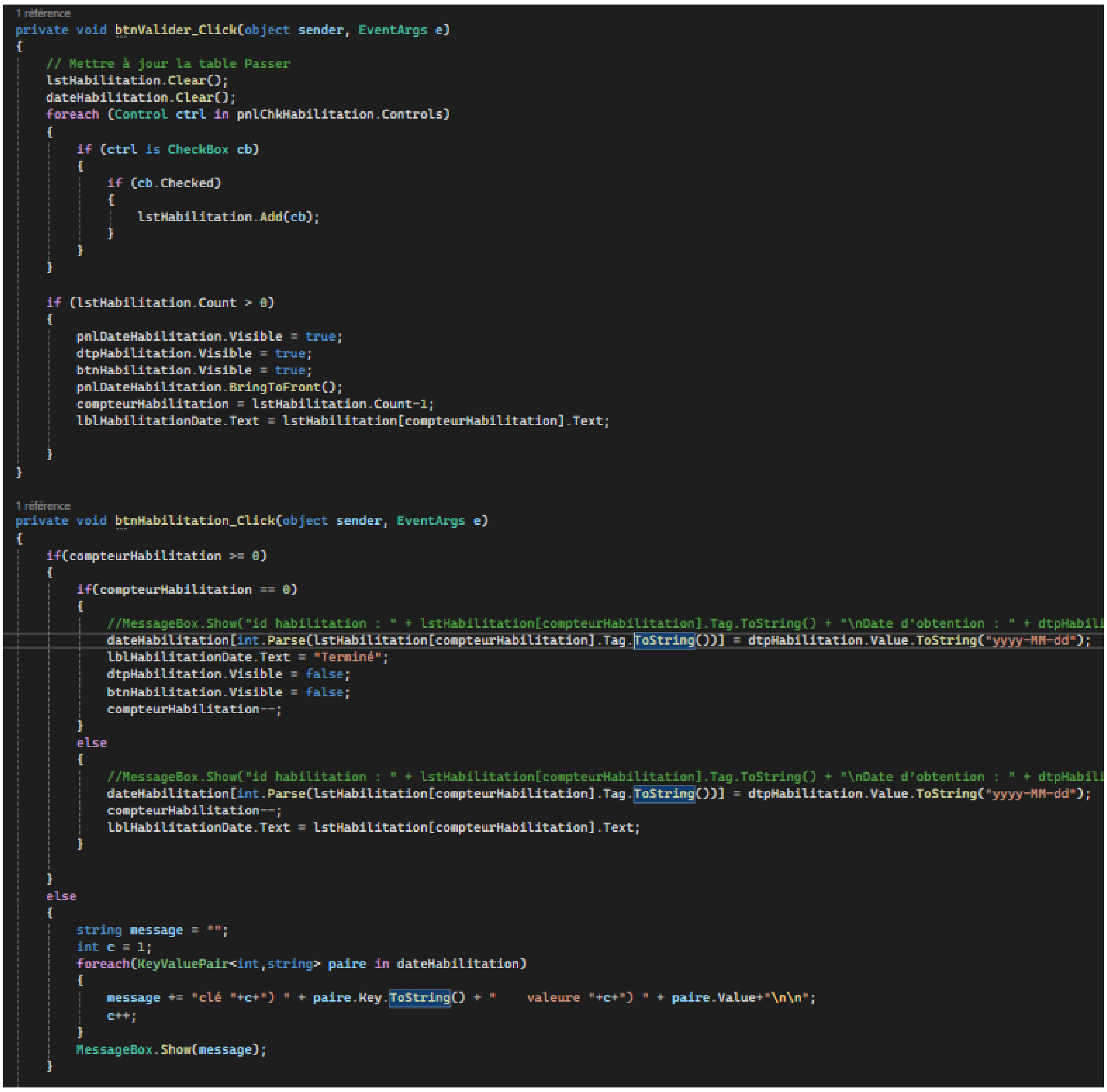
```
4 références
private void rdbMasculin_CheckedChanged(object sender, EventArgs e)
{
    RadioButton rb = sender as RadioButton;

    if (rb != null && rb.Checked)
    {
        rb.FlatAppearance.BorderSize = 3;
        if (rb.Name == "rdbFeminin")
        {
            sexe = "f";
            rb.BackColor = Color.Fuchsia;
            rb.ForeColor = Color.White;
        }
        else  if(rb.Name == "rdbMasculin")
        {
            sexe = "m";
            rb.BackColor = Color.DodgerBlue;
            rb.ForeColor = Color.White;
        }
        else if(rb.Name == "rdbVolontaire")
        {
            type = "v";
            rb.BackColor = Color.DarkGreen;
            rb.ForeColor = Color.White;
        }
        else
        {
            type = "p";
            rb.BackColor = Color.DarkCyan;
            rb.ForeColor = Color.White;
        }
    }
    else
    {
        rb.FlatAppearance.BorderSize = 0;

        if (rb.Name == "rdbFeminin")
        {
            rb.BackColor = Color.FromArgb(255, 192, 255);
            rb.ForeColor = Color.Black;
        }
        else if (rb.Name == "rdbMasculin")
        {
            rb.BackColor = Color.FromArgb(170, 226, 255);
            rb.ForeColor = Color.Black;

        }
        else if (rb.Name == "rdbVolontaire")
        {
            rb.BackColor = Color.FromArgb(192, 255, 192);
            rb.ForeColor = Color.Black;
        }
        else
        {
            rb.BackColor = Color.FromArgb(150, 224,224);
            rb.ForeColor = Color.Black;
        }
    }
    rb.FlatAppearance.MouseOverBackColor = rb.BackColor;
    rb.FlatAppearance.MouseDownBackColor = rb.BackColor;
}
```



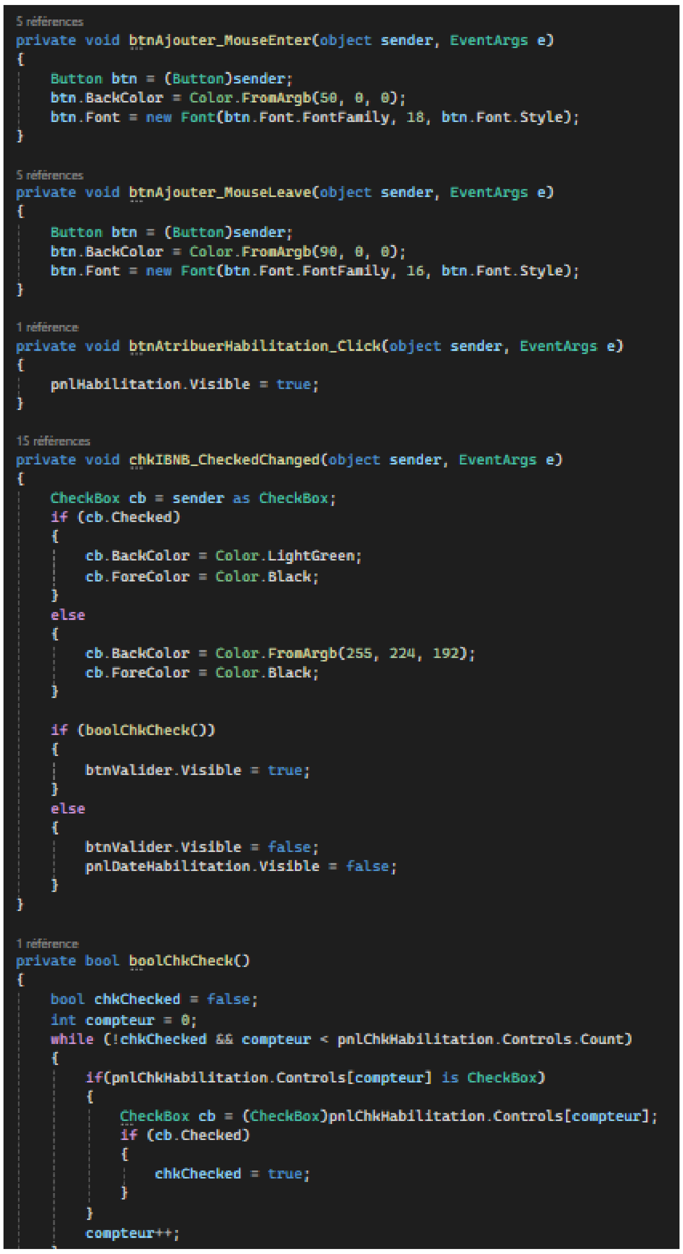
```
4 références
private void rdbMasculin_CheckedChanged(object sender, EventArgs e)
{
    RadioButton rb = sender as RadioButton;

    if (rb != null && rb.Checked)
    {
        rb.FlatAppearance.BorderSize = 3;
        if (rb.Name == "rdbFeminin")
        {
            sexe = "f";
            rb.BackColor = Color.Fuchsia;
            rb.ForeColor = Color.White;
        }
        else  if(rb.Name == "rdbMasculin")
        {
            sexe = "m";
            rb.BackColor = Color.DodgerBlue;
            rb.ForeColor = Color.White;
        }
        else if(rb.Name == "rdbVolontaire")
        {
            type = "v";
            rb.BackColor = Color.DarkGreen;
            rb.ForeColor = Color.White;
        }
        else
        {
            type = "p";
            rb.BackColor = Color.DarkCyan;
            rb.ForeColor = Color.White;
        }
    }
    else
    {
        rb.FlatAppearance.BorderSize = 0;

        if (rb.Name == "rdbFeminin")
        {
            rb.BackColor = Color.FromArgb(255, 192, 255);
            rb.ForeColor = Color.Black;
        }
        else if (rb.Name == "rdbMasculin")
        {
            rb.BackColor = Color.FromArgb(170, 226, 255);
            rb.ForeColor = Color.Black;

        }
        else if (rb.Name == "rdbVolontaire")
        {
            rb.BackColor = Color.FromArgb(192, 255, 192);
            rb.ForeColor = Color.Black;
        }
        else
        {
            rb.BackColor = Color.FromArgb(150, 224,224);
            rb.ForeColor = Color.Black;
        }
    }
    rb.FlatAppearance.MouseOverBackColor = rb.BackColor;
    rb.FlatAppearance.MouseDownBackColor = rb.BackColor;
}
```



```
1 référence
private void btnAjouter_Click(object sender, EventArgs e)
{
    try
    {
        if (txtNewNom.Text != "" && txtNewPrenom.Text != "" && txtNewMatricule.Text != "" && sexe != "" && type != "")
        {
            // Mettre à jour la table Pompier
            string requete = "insert into Pompier (matricule,nom,prenom,sexe,dateNaissance,type,portable,bip,enMission,enConge,codeGrade,dateEmbauche) VALUES(" + txtNewMatricule.Text + ",'" + txtNewNom.Text + "','" + txtNewPrenom.Text + "','" +

            // Mettre à jour la table Affectation
            requete += "insert into Affectation (matriculePompier,dateA,dateFin,idCaserne) VALUES("+txtNewMatricule.Text+",'"+DateTime.Now.ToString("yyyy-MM-dd")+"',NULL,"+cboCaserne.SelectedValue.ToString()+");\n";

            // Mettre à jour la table Passer
            if(lstHabilitation.Count > 0)
            {
                foreach(KeyValuePair<int,string> paire in dateHabilitation)
                {
                    requete += "insert into Passer (matriculePompier,idHabilitation,dateObtention) VALUES(" + txtNewMatricule.Text + "," + paire.Key + ",'" + paire.Value + "');\n";
                }
            }

            // Executer la requete
            SQLiteCommand cd = new SQLiteCommand(requete, connec);
            cd.ExecuteNonQuery();
            MessageBox.Show("Vous avez ajouter le pompier : " + txtNewNom.Text + " " + txtNewPrenom.Text + "\nVous l'avez affecter à la caserne : " + cboCaserne.Text);

            if (this.Parent is ucGestionPersonnel uc)
            {
                uc.RemplirCboPompier(0);
            }
        }
        else
        {
            MessageBox.Show("Veuillez rentrer toutes les informations nécessaire");
        }
    }
    catch(SQLiteException err)
    {
        MessageBox.Show(err.Message);
    }

}
```

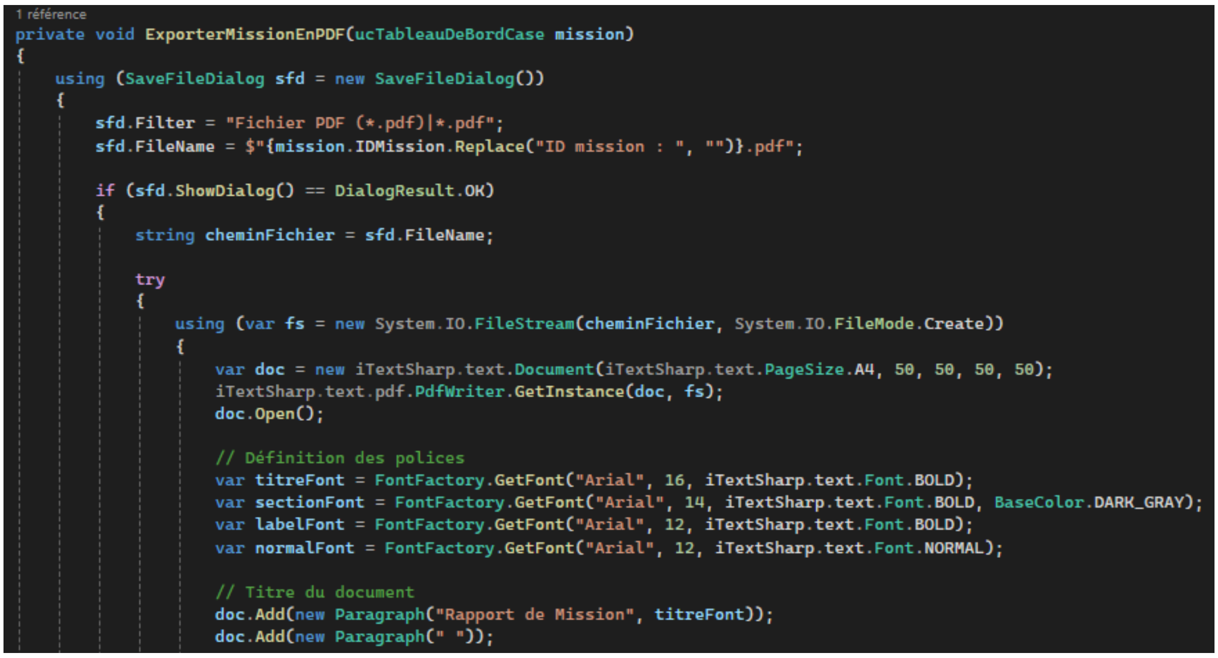


```
1 référence
private void btnValider_Click(object sender, EventArgs e)
{
    // Mettre à jour la table Passer
    lstHabilitation.Clear();
    dateHabilitation.Clear();
    foreach (Control ctrl in pnlChkHabilitation.Controls)
    {
        if (ctrl is CheckBox cb)
        {
            if (cb.Checked)
            {
                lstHabilitation.Add(cb);
            }
        }
    }

    if (lstHabilitation.Count > 0)
    {
        pnlDateHabilitation.Visible = true;
        dtpHabilitation.Visible = true;
        btnHabilitation.Visible = true;
        pnlDateHabilitation.BringToFront();
        compteurHabilitation = lstHabilitation.Count-1;
        lblHabilitationDate.Text = lstHabilitation[compteurHabilitation].Text;

    }
}

1 référence
private void btnHabilitation_Click(object sender, EventArgs e)
{
    if(compteurHabilitation >= 0)
    {
        if(compteurHabilitation == 0)
        {
            //MessageBox.Show("id habilitation : " + lstHabilitation[compteurHabilitation].Tag.ToString() + "\nDate d'obtention : " + dtpHabili
            dateHabilitation[int.Parse(lstHabilitation[compteurHabilitation].Tag.ToString())] = dtpHabilitation.Value.ToString("yyyy-MM-dd");
            lblHabilitationDate.Text = "Terminé";
            dtpHabilitation.Visible = false;
            btnHabilitation.Visible = false;
            compteurHabilitation--;
        }
        else
        {
            //MessageBox.Show("id habilitation : " + lstHabilitation[compteurHabilitation].Tag.ToString() + "\nDate d'obtention : " + dtpHabili
            dateHabilitation[int.Parse(lstHabilitation[compteurHabilitation].Tag.ToString())] = dtpHabilitation.Value.ToString("yyyy-MM-dd");
            compteurHabilitation--;
            lblHabilitationDate.Text = lstHabilitation[compteurHabilitation].Text;
        }

    }
    else
    {
        string message = "";
        int c = 1;
        foreach(KeyValuePair<int,string> paire in dateHabilitation)
        {
            message += "clé "+c+") " + paire.Key.ToString() + "    valeure "+c+") " + paire.Value+"\n\n";
            c++;
        }
        MessageBox.Show(message);
    }
```

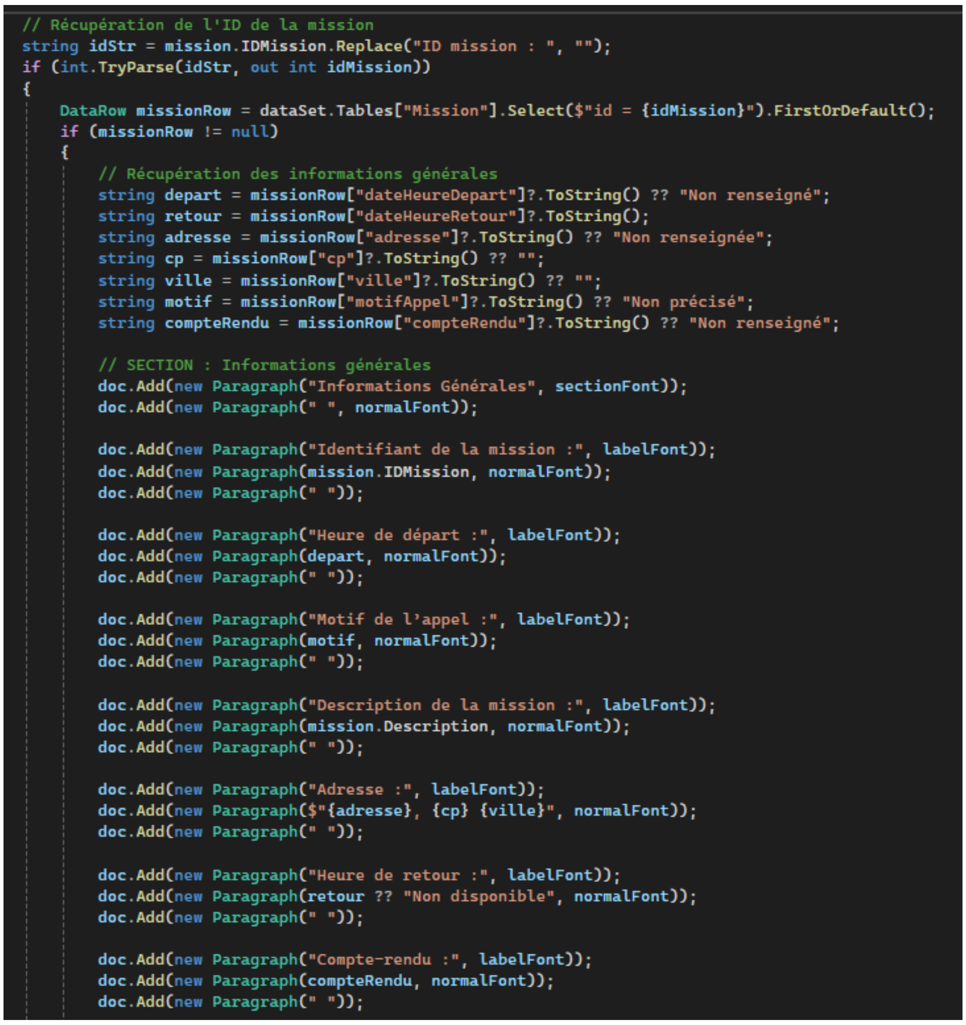


```
5 références
private void btnAjouter_MouseEnter(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.FromArgb(50, 0, 0);
    btn.Font = new Font(btn.Font.FontFamily, 18, btn.Font.Style);
}

5 références
private void btnAjouter_MouseLeave(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.FromArgb(90, 0, 0);
    btn.Font = new Font(btn.Font.FontFamily, 16, btn.Font.Style);
}

1 référence
private void btnAtribuerHabilitation_Click(object sender, EventArgs e)
{
    pnlHabilitation.Visible = true;
}

15 références
private void chkIBNB_CheckedChanged(object sender, EventArgs e)
{
    CheckBox cb = sender as CheckBox;
    if (cb.Checked)
    {
        cb.BackColor = Color.LightGreen;
        cb.ForeColor = Color.Black;
    }
    else
    {
        cb.BackColor = Color.FromArgb(255, 224, 192);
        cb.ForeColor = Color.Black;
    }

    if (boolChkCheck())
    {
        btnValider.Visible = true;
    }
    else
    {
        btnValider.Visible = false;
        pnlDateHabilitation.Visible = false;
    }
}

1 référence
private bool boolChkCheck()
{
    bool chkChecked = false;
    int compteur = 0;
    while (!chkChecked && compteur < pnlChkHabilitation.Controls.Count)
    {
        if(pnlChkHabilitation.Controls[compteur] is CheckBox)
        {
            CheckBox cb = (CheckBox)pnlChkHabilitation.Controls[compteur];
            if (cb.Checked)
            {
                chkChecked = true;
            }
        }
        compteur++;
```

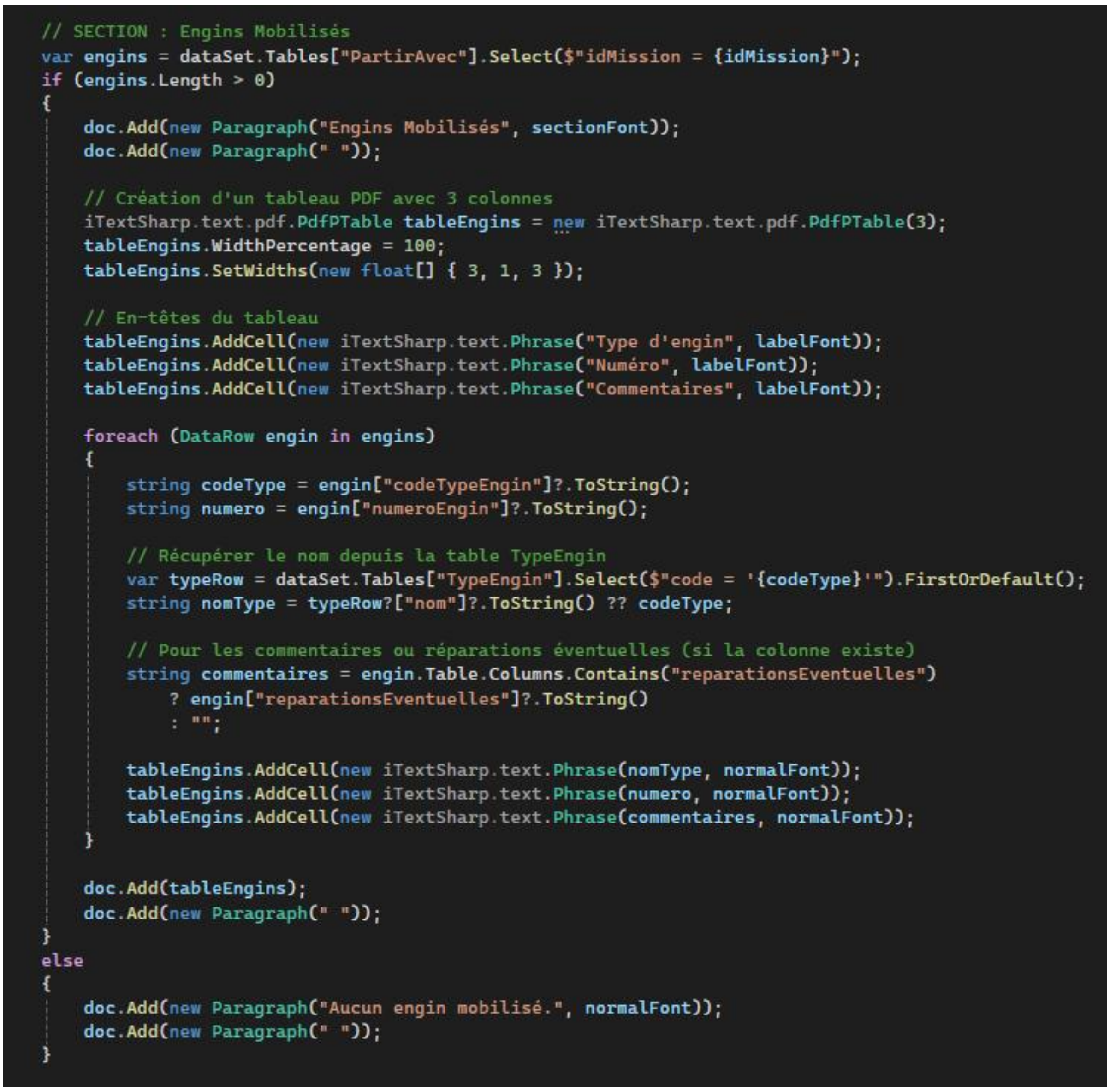
Extrait du code permettant de générer le PDF :



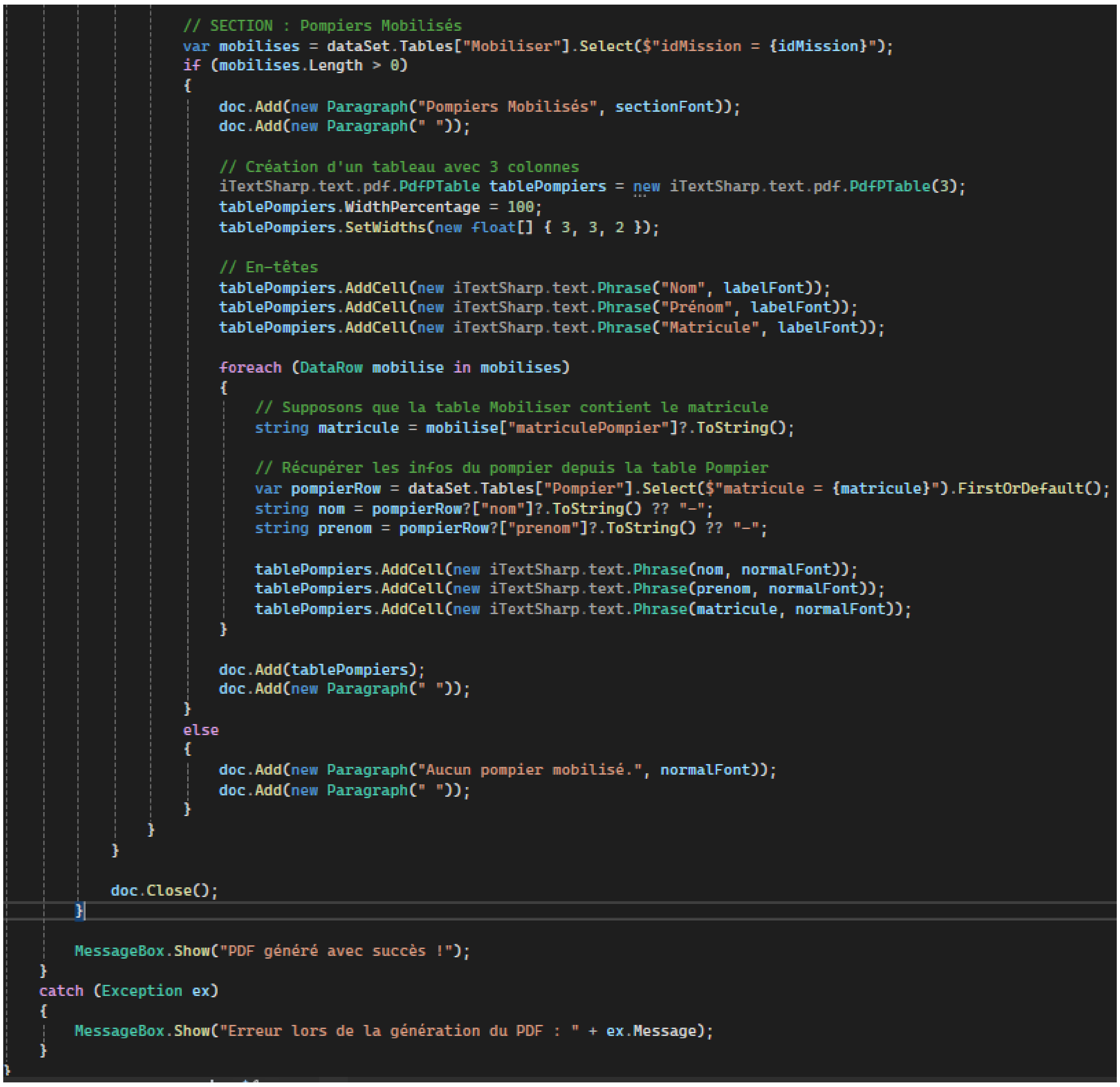
```
1 référence
private void ExporterMissionEnPDF(ucTableauDeBordCase mission)
{
    using (SaveFileDialog sfd = new SaveFileDialog())
    {
        sfd.Filter = "Fichier PDF (*.pdf)|*.pdf";
        sfd.FileName = $"{mission.IDMission.Replace("ID mission : ", "")}.pdf";

        if (sfd.ShowDialog() == DialogResult.OK)
        {
            string cheminFichier = sfd.FileName;

            try
            {
                using (var fs = new System.IO.FileStream(cheminFichier, System.IO.FileMode.Create))
                {
                    var doc = new iTextSharp.text.Document(iTextSharp.text.PageSize.A4, 50, 50, 50, 50);
                    iTextSharp.text.pdf.PdfWriter.GetInstance(doc, fs);
                    doc.Open();

                    // Définition des polices
                    var titreFont = FontFactory.GetFont("Arial", 16, iTextSharp.text.Font.BOLD);
                    var sectionFont = FontFactory.GetFont("Arial", 14, iTextSharp.text.Font.BOLD, BaseColor.DARK_GRAY);
                    var labelFont = FontFactory.GetFont("Arial", 12, iTextSharp.text.Font.BOLD);
                    var normalFont = FontFactory.GetFont("Arial", 12, iTextSharp.text.Font.NORMAL);

                    // Titre du document
                    doc.Add(new Paragraph("Rapport de Mission", titreFont));
                    doc.Add(new Paragraph(" "));
```



```
// Récupération de l'ID de la mission
string idStr = mission.IDMission.Replace("ID mission : ", "");
if (int.TryParse(idStr, out int idMission))
{
    DataRow missionRow = dataSet.Tables["Mission"].Select($"id = {idMission}").FirstOrDefault();
    if (missionRow != null)
    {
        // Récupération des informations générales
        string depart = missionRow["dateHeureDepart"]?.ToString() ?? "Non renseigné";
        string retour = missionRow["dateHeureRetour"]?.ToString();
        string adresse = missionRow["adresse"]?.ToString() ?? "Non renseignée";
        string cp = missionRow["cp"]?.ToString() ?? "";
        string ville = missionRow["ville"]?.ToString() ?? "";
        string motif = missionRow["motifAppel"]?.ToString() ?? "Non précisé";
        string compteRendu = missionRow["compteRendu"]?.ToString() ?? "Non renseigné";

        // SECTION : Informations générales
        doc.Add(new Paragraph("Informations Générales", sectionFont));
        doc.Add(new Paragraph(" ", normalFont));

        doc.Add(new Paragraph("Identifiant de la mission :", labelFont));
        doc.Add(new Paragraph(mission.IDMission, normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Heure de départ :", labelFont));
        doc.Add(new Paragraph(depart, normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Motif de l'appel :", labelFont));
        doc.Add(new Paragraph(motif, normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Description de la mission :", labelFont));
        doc.Add(new Paragraph(mission.Description, normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Adresse :", labelFont));
        doc.Add(new Paragraph($"{adresse}, {cp} {ville}", normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Heure de retour :", labelFont));
        doc.Add(new Paragraph(retour ?? "Non disponible", normalFont));
        doc.Add(new Paragraph(" "));

        doc.Add(new Paragraph("Compte-rendu :", labelFont));
        doc.Add(new Paragraph(compteRendu, normalFont));
        doc.Add(new Paragraph(" "));
```

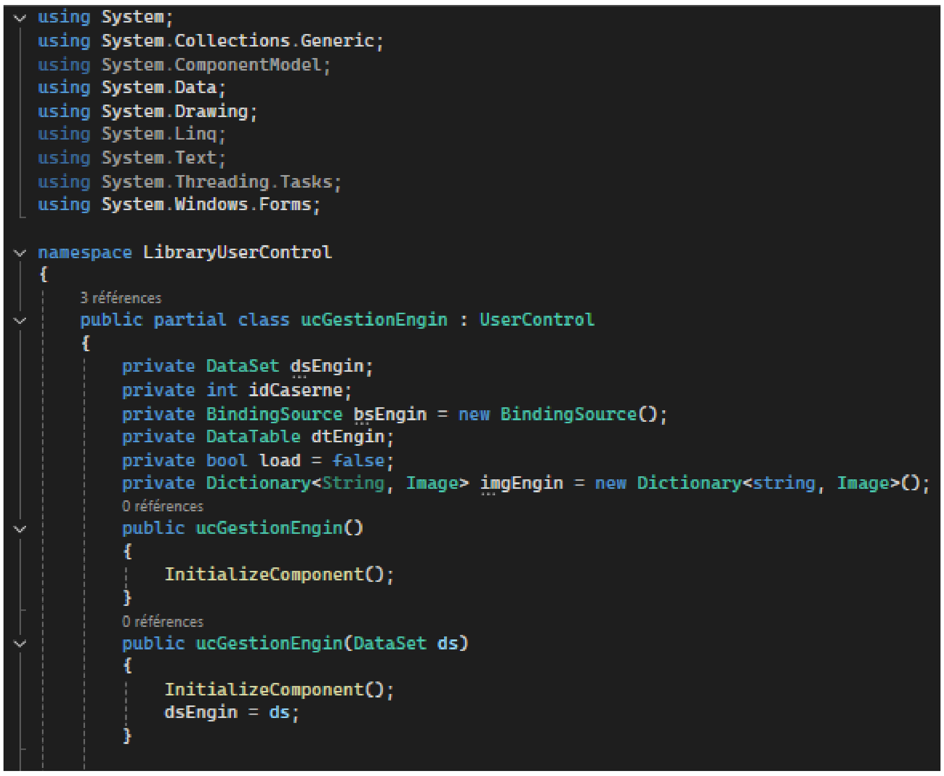


```
// SECTION : Engins Mobilisés
var engins = dataSet.Tables["PartirAvec"].Select($"idMission = {idMission}");
if (engins.Length > 0)
{
    doc.Add(new Paragraph("Engins Mobilisés", sectionFont));
    doc.Add(new Paragraph(" "));

    // Création d'un tableau PDF avec 3 colonnes
    iTextSharp.text.pdf.PdfPTable tableEngins = new iTextSharp.text.pdf.PdfPTable(3);
    tableEngins.WidthPercentage = 100;
    tableEngins.SetWidths(new float[] { 3, 1, 3 });

    // En-têtes du tableau
    tableEngins.AddCell(new iTextSharp.text.Phrase("Type d'engin", labelFont));
    tableEngins.AddCell(new iTextSharp.text.Phrase("Numéro", labelFont));
    tableEngins.AddCell(new iTextSharp.text.Phrase("Commentaires", labelFont));

    foreach (DataRow engin in engins)
    {
        string codeType = engin["codeTypeEngin"]?.ToString();
        string numero = engin["numeroEngin"]?.ToString();

        // Récupérer le nom depuis la table TypeEngin
        var typeRow = dataSet.Tables["TypeEngin"].Select($"code = '{codeType}'").FirstOrDefault();
        string nomType = typeRow?["nom"]?.ToString() ?? codeType;

        // Pour les commentaires ou réparations éventuelles (si la colonne existe)
        string commentaires = engin.Table.Columns.Contains("reparationsEventuelles")
            ? engin["reparationsEventuelles"]?.ToString()
            : "";

        tableEngins.AddCell(new iTextSharp.text.Phrase(nomType, normalFont));
        tableEngins.AddCell(new iTextSharp.text.Phrase(numero, normalFont));
        tableEngins.AddCell(new iTextSharp.text.Phrase(commentaires, normalFont));
    }

    doc.Add(tableEngins);
    doc.Add(new Paragraph(" "));
}
else
{
    doc.Add(new Paragraph("Aucun engin mobilisé.", normalFont));
    doc.Add(new Paragraph(" "));
}
```

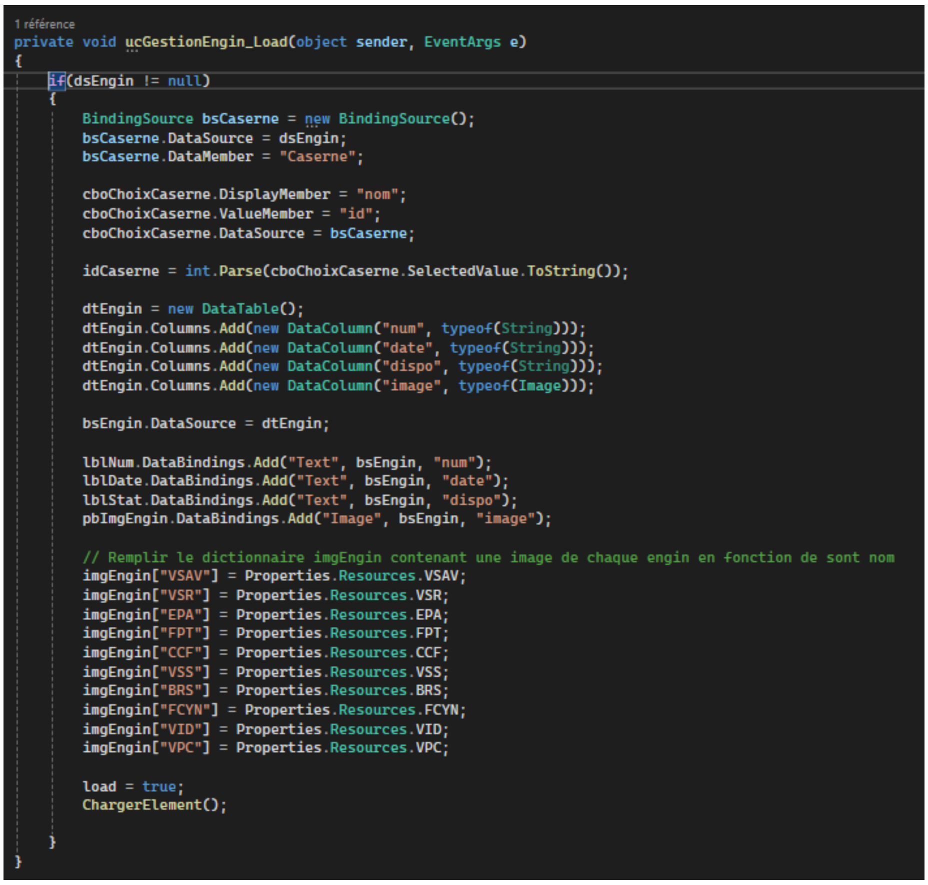


```
                    // SECTION : Pompiers Mobilisés
                    var mobilises = dataSet.Tables["Mobiliser"].Select($"idMission = {idMission}");
                    if (mobilises.Length > 0)
                    {
                        doc.Add(new Paragraph("Pompiers Mobilisés", sectionFont));
                        doc.Add(new Paragraph(" "));

                        // Création d'un tableau avec 3 colonnes
                        iTextSharp.text.pdf.PdfPTable tablePompiers = new iTextSharp.text.pdf.PdfPTable(3);
                        tablePompiers.WidthPercentage = 100;
                        tablePompiers.SetWidths(new float[] { 3, 3, 2 });

                        // En-têtes
                        tablePompiers.AddCell(new iTextSharp.text.Phrase("Nom", labelFont));
                        tablePompiers.AddCell(new iTextSharp.text.Phrase("Prénom", labelFont));
                        tablePompiers.AddCell(new iTextSharp.text.Phrase("Matricule", labelFont));

                        foreach (DataRow mobilise in mobilises)
                        {
                            // Supposons que la table Mobiliser contient le matricule
                            string matricule = mobilise["matriculePompier"]?.ToString();

                            // Récupérer les infos du pompier depuis la table Pompier
                            var pompierRow = dataSet.Tables["Pompier"].Select($"matricule = {matricule}").FirstOrDefault();
                            string nom = pompierRow?["nom"]?.ToString() ?? "-";
                            string prenom = pompierRow?["prenom"]?.ToString() ?? "-";

                            tablePompiers.AddCell(new iTextSharp.text.Phrase(nom, normalFont));
                            tablePompiers.AddCell(new iTextSharp.text.Phrase(prenom, normalFont));
                            tablePompiers.AddCell(new iTextSharp.text.Phrase(matricule, normalFont));
                        }

                        doc.Add(tablePompiers);
                        doc.Add(new Paragraph(" "));
                    }
                    else
                    {
                        doc.Add(new Paragraph("Aucun pompier mobilisé.", normalFont));
                        doc.Add(new Paragraph(" "));
                    }
                }
            }

            doc.Close();
        }

        MessageBox.Show("PDF généré avec succès !");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erreur lors de la génération du PDF : " + ex.Message);
    }
}
```

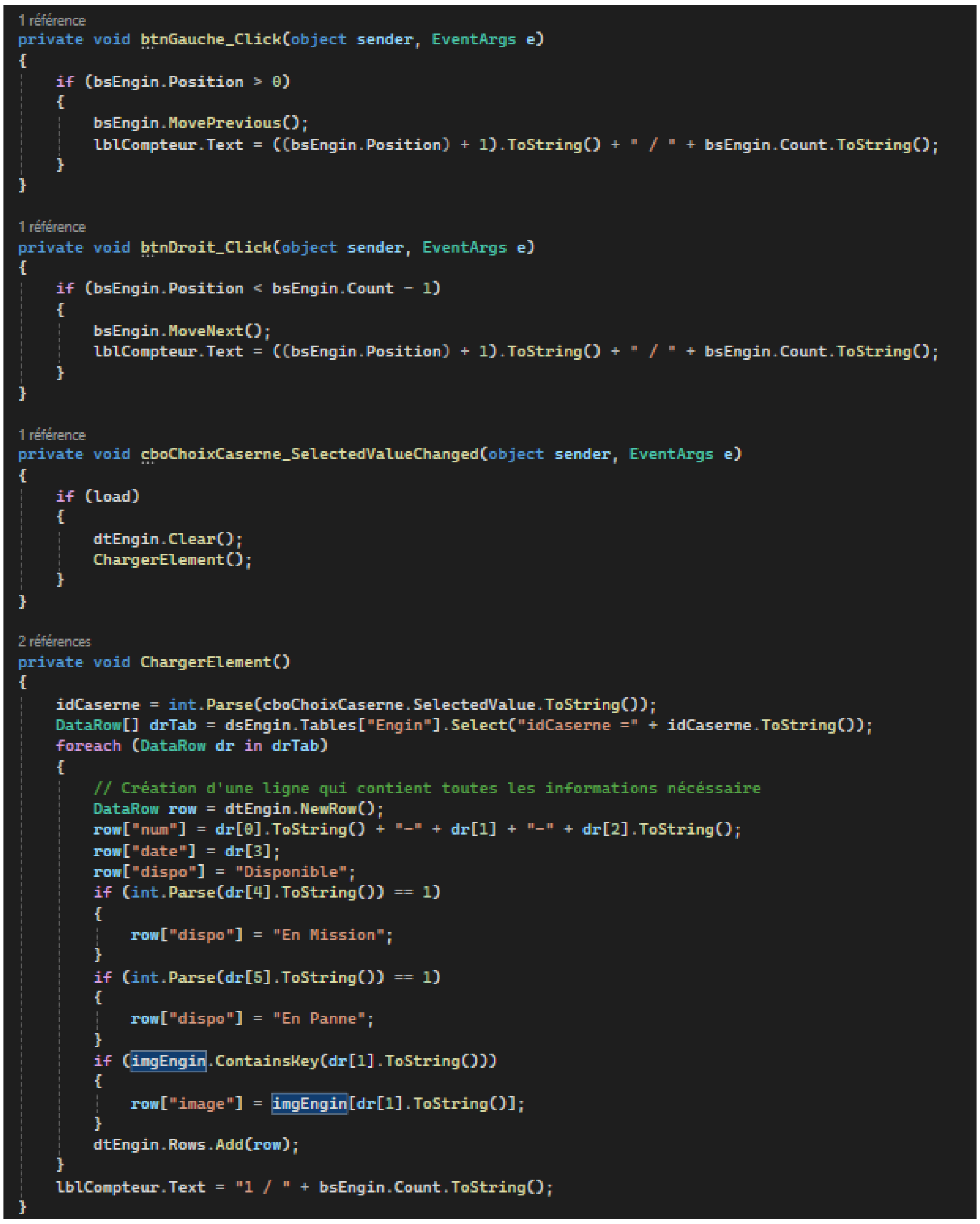
Code permettant la navigation en mode liaison de données entre les véhicules d’une caserne :



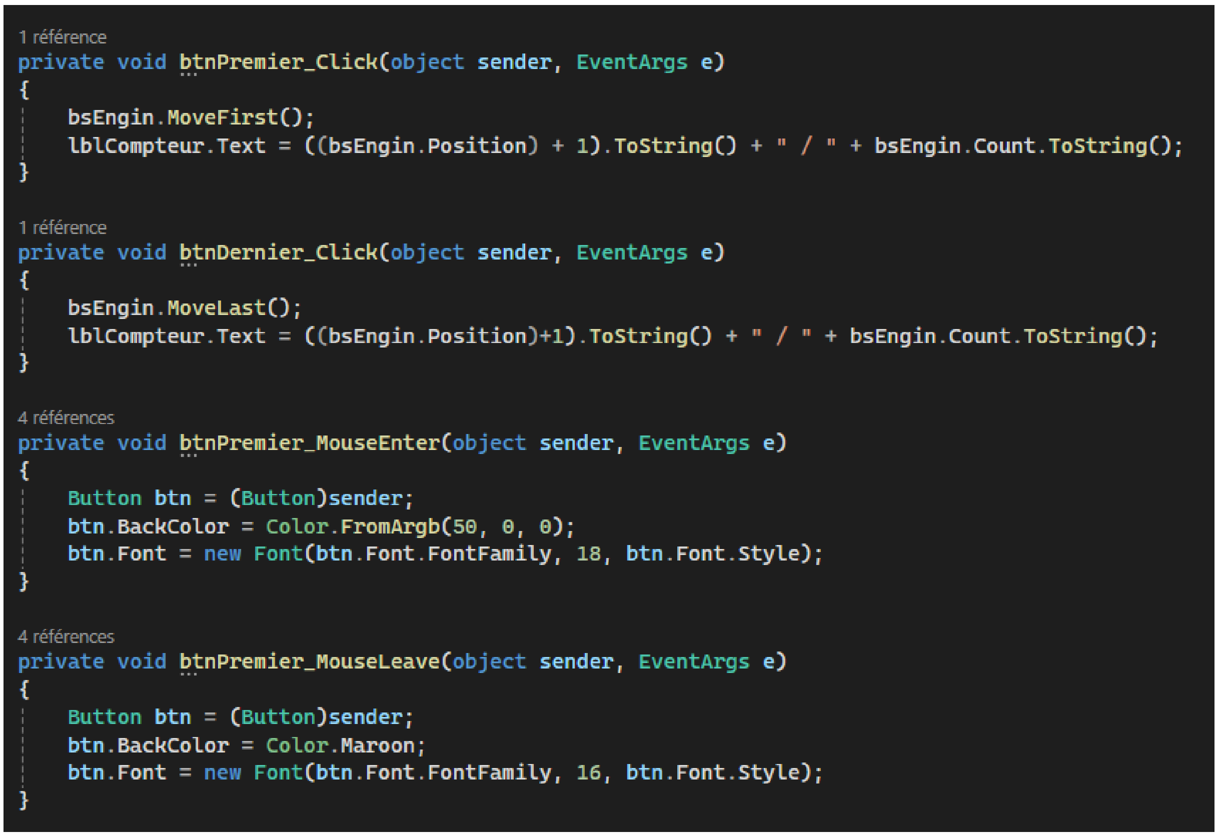
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LibraryUserControl
{
    3 références
    public partial class ucGestionEngin : UserControl
    {
        private DataSet dsEngin;
        private int idCaserne;
        private BindingSource bsEngin = new BindingSource();
        private DataTable dtEngin;
        private bool load = false;
        private Dictionary<String, Image> imgEngin = new Dictionary<string, Image>();
        0 références
        public ucGestionEngin()
        {
            InitializeComponent();
        }
        0 références
        public ucGestionEngin(DataSet ds)
        {
            InitializeComponent();
            dsEngin = ds;
        }
```



```
1 référence
private void ucGestionEngin_Load(object sender, EventArgs e)
{
    if(dsEngin != null)
    {
        BindingSource bsCaserne = new BindingSource();
        bsCaserne.DataSource = dsEngin;
        bsCaserne.DataMember = "Caserne";

        cboChoixCaserne.DisplayMember = "nom";
        cboChoixCaserne.ValueMember = "id";
        cboChoixCaserne.DataSource = bsCaserne;

        idCaserne = int.Parse(cboChoixCaserne.SelectedValue.ToString());

        dtEngin = new DataTable();
        dtEngin.Columns.Add(new DataColumn("num", typeof(String)));
        dtEngin.Columns.Add(new DataColumn("date", typeof(String)));
        dtEngin.Columns.Add(new DataColumn("dispo", typeof(String)));
        dtEngin.Columns.Add(new DataColumn("image", typeof(Image)));

        bsEngin.DataSource = dtEngin;

        lblNum.DataBindings.Add("Text", bsEngin, "num");
        lblDate.DataBindings.Add("Text", bsEngin, "date");
        lblStat.DataBindings.Add("Text", bsEngin, "dispo");
        pbImgEngin.DataBindings.Add("Image", bsEngin, "image");

        // Remplir le dictionnaire imgEngin contenant une image de chaque engin en fonction de sont nom
        imgEngin["VSAV"] = Properties.Resources.VSAV;
        imgEngin["VSR"] = Properties.Resources.VSR;
        imgEngin["EPA"] = Properties.Resources.EPA;
        imgEngin["FPT"] = Properties.Resources.FPT;
        imgEngin["CCF"] = Properties.Resources.CCF;
        imgEngin["VSS"] = Properties.Resources.VSS;
        imgEngin["BRS"] = Properties.Resources.BRS;
        imgEngin["FCYN"] = Properties.Resources.FCYN;
        imgEngin["VID"] = Properties.Resources.VID;
        imgEngin["VPC"] = Properties.Resources.VPC;

        load = true;
        ChargerElement();

    }
}
```



```
1 référence
private void btnGauche_Click(object sender, EventArgs e)
{
    if (bsEngin.Position > 0)
    {
        bsEngin.MovePrevious();
        lblCompteur.Text = ((bsEngin.Position) + 1).ToString() + " / " + bsEngin.Count.ToString();
    }
}

1 référence
private void btnDroit_Click(object sender, EventArgs e)
{
    if (bsEngin.Position < bsEngin.Count - 1)
    {
        bsEngin.MoveNext();
        lblCompteur.Text = ((bsEngin.Position) + 1).ToString() + " / " + bsEngin.Count.ToString();
    }
}

1 référence
private void cboChoixCaserne_SelectedValueChanged(object sender, EventArgs e)
{
    if (load)
    {
        dtEngin.Clear();
        ChargerElement();
    }
}

2 références
private void ChargerElement()
{
    idCaserne = int.Parse(cboChoixCaserne.SelectedValue.ToString());
    DataRow[] drTab = dsEngin.Tables["Engin"].Select("idCaserne =" + idCaserne.ToString());
    foreach (DataRow dr in drTab)
    {
        // Création d'une ligne qui contient toutes les informations nécéssaire
        DataRow row = dtEngin.NewRow();
        row["num"] = dr[0].ToString() + "-" + dr[1] + "-" + dr[2].ToString();
        row["date"] = dr[3];
        row["dispo"] = "Disponible";
        if (int.Parse(dr[4].ToString()) == 1)
        {
            row["dispo"] = "En Mission";
        }
        if (int.Parse(dr[5].ToString()) == 1)
        {
            row["dispo"] = "En Panne";
        }
        if (imgEngin.ContainsKey(dr[1].ToString()))
        {
            row["image"] = imgEngin[dr[1].ToString()];
        }
        dtEngin.Rows.Add(row);
    }
    lblCompteur.Text = "1 / " + bsEngin.Count.ToString();
}
```



```
1 référence
private void btnPremier_Click(object sender, EventArgs e)
{
    bsEngin.MoveFirst();
    lblCompteur.Text = ((bsEngin.Position) + 1).ToString() + " / " + bsEngin.Count.ToString();
}

1 référence
private void btnDernier_Click(object sender, EventArgs e)
{
    bsEngin.MoveLast();
    lblCompteur.Text = ((bsEngin.Position)+1).ToString() + " / " + bsEngin.Count.ToString();
}

4 références
private void btnPremier_MouseEnter(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.FromArgb(50, 0, 0);
    btn.Font = new Font(btn.Font.FontFamily, 18, btn.Font.Style);
}

4 références
private void btnPremier_MouseLeave(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    btn.BackColor = Color.Maroon;
    btn.Font = new Font(btn.Font.FontFamily, 16, btn.Font.Style);
}
```