Membres de l'équipe:
Elora Fouilleul
Ivanoe Megnin Preiss
Fella Bennounas

version 5.1
date: 10/06/2025

Sommaire:

Formulaire gestion pompier: 2
Le code complet du User Control : 12
Un extrait du code permettant de générer le pdf : 15
Navigation en mode liaison de données entre les véhicules d'une caserne: BindingSource 18

Formulaire gestion pompier:

```
e PimPomBro
{
    public partial class frmGestionPompiers : Form
    {
        // on peut initialiser à true pour bypass le test de login (
"vrichard", mdp = "mdpVero")
        bool admin = false;
        private int matricule = -1;
        string requete = "";
        SQLiteCommand cmd = null;
        SQLiteDataReader reader = null;

        public frmGestionPompiers()
        {
            InitializeComponent();
        }

        private void gestionPompiers_Load(object sender, EventArgs e
        {

            // on affiche les 2 images
            picLogo.Image =
Image.FromFile(@"../../Images/ImagesGestionPompiers/logo.png");
            picLogo.SizeMode = PictureBoxSizeMode.Zoom;
            picNouveau.Image =
Image.FromFile(@"../../Images/ImagesGestionPompiers/jsp.jpg");

            // on initialise les combobox des casernes,
            cboCaserne.DisplayMember = "nom";
            cboCaserne.ValueMember = "id";
            cboCaserne.DataSource = MesDatas.DsGlobal.Tables["Caserr

            cboCaserneDeRattachement.DisplayMember = "nom";
            cboCaserneDeRattachement.ValueMember = "id";
            // en utilisant une copie pour la seconde afin qu'elles
soient pas synchronisées
            cboCaserneDeRattachement.DataSource =
MesDatas.DsGlobal.Tables["Caserne"].Copy();

            cboGrade.DisplayMember = "libelle";
```

```
            cboGrade.ValueMember = "code";
            cboGrade.DataSource = MesDatas.DsGlobal.Tables["Grade"];

            cboPompier.Text = "choisir pompier";
            cboPompier.SelectedIndex = -1;
        }

        private void cboPompier_Click(object sender, EventArgs e)
        {
            if (cboCaserne.SelectedItem == null)
            {
                MessageBox.Show("Veuillez selectionner une caserne p
pouvoir choisir un pompier.");
            }
        }

        private void cboCaserne_SelectedIndexChanged(object sender,
EventArgs e)
        {
            lblConsigne.Text = "Veuillez sélectionner un pompier.";
            lblConsigne.Font = new Font("Arial", 15, FontStyle.Bold)
            lblConsigne.BackColor = Color.Transparent;
            cboPompier.Text = "choisir pompier";

            if (cboCaserne.SelectedItem == null) { return; }
            int idCaserne = Convert.ToInt32(cboCaserne.SelectedValue

            // on récupère et affiche les pompiers de la caserne dep
dataset global
            DataTable tablePompiers = MesDatas.DsGlobal.Tables["Pomp
            DataTable tableAffectation =
MesDatas.DsGlobal.Tables["Affectation"];

            List<int> listeMatricules = new List<int>();
            foreach (DataRow rowAff in tableAffectation.Rows)
            {
                // on vérifie que le pompier est bien affecté à l'he
actuelle dans la caserne
                // (c'est à dire si la date de fin n'est pas définie
                if (rowAff["DateFin"].ToString() == "" &&
Convert.ToInt32(rowAff["idCaserne"]) == idCaserne)
                {
```

```
listeMatricules.Add(Convert.ToInt32(rowAff["matriculePompier"]));
                }
            }

            string filter = "matricule IN (" + string.Join(",",
listeMatricules) + ")";
            DataView dv = new DataView(tablePompiers, filter, "nom A
prenom ASC", DataViewRowState.CurrentRows);

            cboPompier.ValueMember = "matricule";
            cboPompier.DataSource = dv;

            // on affiche le nom ET le prénom des pompiers dans la
combobox, bien que ce soit des colonnes différentes
            cboPompier.Format += (s, e2) =>
            {
                DataRowView drv = e2.ListItem as DataRowView;
                if (drv != null)
                {
                    e2.Value = drv["nom"] + " " + drv["prenom"];
                }
            };

            // on cache le panel tant qu'on pompier n'est pas sélect
            pnlPompier.Visible = false;
            cboPompier.SelectedIndex = -1;
            cboPompier.Select();
        }

        private void cboPompier_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (cboPompier.SelectedItem == null || cboPompier.Select
== -1 || cboPompier.Text == "")
            {
                return;
            }

            pnlInformationsDetaillees.Visible = false;
            cboGrade.Enabled = false;
```

```
            // une fois qu'un pompier est sélectionné on met à jours
les informations
            matricule = Convert.ToInt32(cboPompier.SelectedValue);
            DataTable tablePompier = MesDatas.DsGlobal.Tables["Pompi
            DataRow rowPompier = tablePompier.Rows.Find(matricule);

            lblMatricule.Text = matricule.ToString();
            lblNom.Text = rowPompier["nom"].ToString();
            lblPrenom.Text = rowPompier["prenom"].ToString();
            lblSexe.Text = rowPompier["sexe"].ToString().ToUpper();
            lblDateEmbauche.Text = rowPompier["dateEmbauche"].ToStri
            txtGrade.Text = rowPompier["codeGrade"].ToString();
            lblDateDeNaissance.Text =
rowPompier["dateNaissance"].ToString();
            if (rowPompier["type"].ToString() == "p") {
rdbProfessionnel.Checked = true; } else { rdbVolontaire.Checked = tr
            try
            {
                lblTelephone.Text = rowPompier["portable"].ToString(
                lblBip.Text = rowPompier["bip"].ToString();
            }
            catch (Exception ex) {
              lblTelephone.Text = "";
              lblBip.Text = "";
            }

            try
            {
                requete = "SELECT libelle FROM Grade WHERE code =
@codeGrade";
                cmd = new SQLiteCommand(requete, Connexion.Connec);
                cmd.Parameters.AddWithValue("@codeGrade", txtGrade.T
                reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    cboGrade.Text = reader["libelle"].ToString();
                }
            } catch (Exception ex) {
                cboGrade.Visible = false;
            }
```

```
            // on affiche les informations une fois que tout a été
initialisé
            pnlPompier.Visible = true;
        }
        private void btnModifications_Click(object sender, EventArgs
        {
            if (!admin) { connexionAdmin(); }
            if (!admin) { return; }

            cboCaserneDeRattachement.Text = cboCaserne.Text;

            // on récupère la liste de toutes les habilitations
            lstHabilitations.Items.Clear();
            requete = "SELECT h.libelle,p.dateObtention FROM Passer
Habilitation h ON p.idHabilitation = h.id WHERE p.matriculePompier =
@matricule";
            cmd = new SQLiteCommand(requete, Connexion.Connec);
            cmd.Parameters.AddWithValue("@matricule", matricule);
            reader = cmd.ExecuteReader();
            while (reader.Read()) {
                DateTime date =
DateTime.Parse(reader["dateObtention"].ToString());
                lstHabilitations.Items.Add(date.ToString("dd/MM/yyyy
- " + reader["libelle"]);
            }

            // on récupère la liste des affectations
            lstAffectations.Items.Clear();
            requete = "SELECT a.dateA,a.dateFin,c.nom FROM Affectati
JOIN Caserne c ON a.idCaserne = c.id WHERE a.matriculePompier = @mat
AND a.dateFin IS NOT NULL";
            cmd = new SQLiteCommand(requete, Connexion.Connec);
            cmd.Parameters.AddWithValue("@matricule", matricule);
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                string caserne = reader["nom"].ToString();
                DateTime dA = DateTime.Parse(reader["dateA"].ToStrir
                string dateA = dA.ToString("dd/MM/yyyy");
                DateTime dF = DateTime.Parse(reader["dateFin"].ToStr
                string dateFin = dF.ToString("dd/MM/yyyy");
                lstAffectations.Items.Add(caserne + " : " + dateA +
```

```
dateFin);
            }

            requete = "SELECT enConge FROM Pompier WHERE matricule =
@matricule";
            cmd = new SQLiteCommand(requete, Connexion.Connec);
            cmd.Parameters.AddWithValue("@matricule", matricule);
            reader = cmd.ExecuteReader();
            reader.Read();
            if (Convert.ToInt32(reader["enConge"]) == 1)
            {
                chkConge.Checked = true;
            } else
            {
                chkConge.Checked = false;
            }

            reader.Close();
            cboGrade.Enabled = true;
            pnlInformationsDetaillees.Visible = true;
        }

       private void btnAppliquerModif_Click(object sender, EventArgs
        {
            using (SQLiteTransaction tran =
Connexion.Connec.BeginTransaction())
            {
                try
                {
                    requete = "UPDATE Pompier SET codeGrade = @codeG
enConge = @enConge WHERE matricule = @matricule";
                    cmd = new SQLiteCommand(requete, Connexion.Conne
tran);
                    cmd.Parameters.AddWithValue("@codeGrade",
txtGrade.Text);
                    cmd.Parameters.AddWithValue("@matricule", matric
                    cmd.Parameters.AddWithValue("@enConge",
(chkConge.Checked) ? 1 : 0);
                    cmd.ExecuteNonQuery();

                    requete = "UPDATE Affectation SET dateFin = @dat
WHERE matriculePompier = @matricule AND (dateFin IS NULL or dateFin
```

```
                    cmd = new SQLiteCommand(requete, Connexion.Conne
tran);
                    cmd.Parameters.AddWithValue("@matricule", matric
                    cmd.Parameters.AddWithValue("@dateFin", DateTime
                    cmd.ExecuteNonQuery();

                    requete = "INSERT INTO Affectation VALUES (@matr
@dateA, NULL,@caserne)";
                    cmd = new SQLiteCommand(requete, Connexion.Conne
tran);
                    cmd.Parameters.AddWithValue("@matricule", matric
                    cmd.Parameters.AddWithValue("@dateA", DateTime.N
                    cmd.Parameters.AddWithValue("@caserne",
cboCaserneDeRattachement.SelectedValue);
                    cmd.ExecuteNonQuery();

                    tran.Commit();
                }
                catch (Exception ex)
                {
                    tran.Rollback();
                    MessageBox.Show(ex.Message);
                    return;
                }
            }

            // après avoir modifié les tables dans la base de donnée
mets à jours dans le dataset
            MesDatas.refreshTable("Pompier");
            MesDatas.refreshTable("Affectation");


            // puis on reload le formulaire pour récupérer le datase
jours
            this.gestionPompiers_Load(sender, e);
            this.cboCaserne_SelectedIndexChanged(sender, e);
        }

       // on met à jours le code du grade quand l'utilisateur en
sélectionne un autre dans la combobox
        private void cboGrade_SelectedIndexChanged(object sender, Ev
e)
```

```
        {
            requete = "SELECT code FROM Grade WHERE libelle = @libel
            cmd = new SQLiteCommand(requete, Connexion.Connec);
            cmd.Parameters.AddWithValue("@libelle", cboGrade.Text);
            reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                txtGrade.Text = reader["code"].ToString();
            }

            reader.Close();
        }
        private void connexionAdmin()
        {
            frmConnexionAdmin connexionAdmin = new frmConnexionAdmir
            var result = connexionAdmin.ShowDialog();
            if (result == DialogResult.OK) {
                admin = true;
            } else
            {
                admin = false;
            }

        }

        private void btnNouveau_Click(object sender, EventArgs e)
        {
            // si l'utilisateur n'est pas admin on lui demande de ce
connecter
            if (!admin) { connexionAdmin(); }
            // s'il n'est toujours pas connecté on quitte la fonctio
            if (!admin) { return; }

            frmCreationPompier creationPompier = new frmCreationPomp
            creationPompier.ShowDialog();

            this.gestionPompiers_Load(sender, e);
            this.cboCaserne_SelectedIndexChanged(sender, e);
        }

        private void btnAnnuler_Click(object sender, EventArgs e)
        {
```

```
            cboPompier_SelectedIndexChanged(sender, e);
            btnModifications_Click(sender, e);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}



//CONNEXION ADMIN

namespace PimPomBro
{
    public partial class frmConnexionAdmin : Form
    {

        public bool admin { get; private set; } = false;
        public frmConnexionAdmin()
        {
            InitializeComponent();
        }

        private void btnSeConnecter_Click(object sender, EventArgs e
        {
            try
            {
                String requete = "SELECT mdp FROM Admin WHERE login
@login";
                SQLiteCommand cmd = new SQLiteCommand(requete,
Connexion.Connec);
                cmd.Parameters.AddWithValue("@login", txtLogin.Text)
                IDataReader reader = cmd.ExecuteReader();
                reader.Read();
                if (txtMDP.Text == reader["mdp"].ToString()) {
                    admin = true;
                    this.DialogResult = DialogResult.OK;
                } else
                {
```

```
                    connexionRatee();
                }
            } catch
            {
                connexionRatee();
            }
        }

        private void connexionRatee() {
                admin = false;
                MessageBox.Show("Le login et/ou le mot de passe est
erroné.");
                txtMDP.Text = "";
                txtLogin.Select();
                txtLogin.SelectAll();

        }

        private void txtMDP_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                btnSeConnecter_Click(sender, e);
            }
        }

        private void txtLogin_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                txtMDP.Select();
            }
        }
    }
}
```

Le code complet du User Control :

```
namespace PimPomBro
{

      public delegate void ClotureMission(object sender, EventArgs
e);
      public delegate void PDF(object sender, EventArgs e);


      public partial class AfficheMission: UserControl
      {
      public AfficheMission()
      {
            InitializeComponent();
      }

      public AfficheMission(int id, String type, String caserne,
String date, String heure, String adresse, String status, String
motif, String chemin1, String chemin2, String chemin3)
      {
            InitializeComponent();
            lblId.Text = id.ToString();
            lblNatureSinistre.Text = type;
            lblCaserne.Text = caserne;
            lblDate.Text = date;
            lblHeure.Text = heure;
            lblStatus.Text = status;
            lblMotif.Text = motif;
            // pictureBoxMission.Image =
Image.FromFile("../../Ressources/Mission.png");
            pictureBoxMission.Image = Image.FromFile(chemin1);
            pictureBoxMission.SizeMode =
PictureBoxSizeMode.StretchImage;


            btnPDF.BackgroundImage = Image.FromFile(chemin2);
            btnPDF.BackgroundImageLayout = ImageLayout.Stretch;
            btnPDF.Text = ""; // Aucun texte


            btnTerminer.BackgroundImage = Image.FromFile(chemin3);
            btnTerminer.BackgroundImageLayout =
```

```
ImageLayout.Stretch;
            btnTerminer.Text = ""; // Pas de texte
      }


      public ClotureMission ClotureMission;
      public PDF PDF;

      private void btnPDF_MouseHover(object sender, EventArgs e)
      {
            System.Windows.Forms.Button btn = sender as
System.Windows.Forms.Button;
            if (btn != null)
            {
                  btn.BackColor = Color.LightGray;
            }
      }


      private void btnPDF_MouseLeave(object sender, EventArgs e)
      {

            System.Windows.Forms.Button btn = sender as
System.Windows.Forms.Button;
            if (btn != null)
            {
                  btn.BackColor = Color.White;
            }
      }

      private void btnTerminer_Click(object sender, EventArgs e)
      {
            if (this.ClotureMission != null)
            {
                  this.ClotureMission(sender, e);
            }
      }

      private void pictureBoxMission_Click(object sender, EventArgs
e)
      {
            if (this.ClotureMission != null)
```

```
            {
                  this.ClotureMission(sender, e);
            }
      }



      private void grpBox1_Enter(object sender, EventArgs e)
      {

      }

      private void btnPDF_Click(object sender, EventArgs e)
      {
            if (this.PDF != null)
            {
                  this.PDF(sender, e);
            }
      }
      }
}
```

Un extrait du code permettant de générer le pdf :

```
private void generationPDF(object sender, EventArgs e)
{
      //Recuperation des informations
      {...};
      //Génération du PDF
      Document doc = new Document();
      // initialisation des polices
{...};

      try
      {
      string chemin =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.D
esktop), $"Mission_{idMission}.pdf");
      PdfWriter writer = PdfWriter.GetInstance(doc, new
FileStream(chemin, FileMode.Create));
      doc.Open();


      iTextSharp.text.Font titre =
FontFactory.GetFont(FontFactory.HELVETICA_BOLD, 20f,
BaseColor.RED);
      Paragraph pTitre = new Paragraph("Mission " + idMission + "
(" + status + ")", titre);
      pTitre.Alignment = Element.ALIGN_CENTER; // centrer le texte
      doc.Add(pTitre);
      doc.Add(new Paragraph(" ")); // espace après le titre

      Paragraph pInfos = new Paragraph("Informations générales : ",
maPolice);
      doc.Add(pInfos);

      doc.Add(new Paragraph("Caserne mobilisée : " + caserne,
police));
      doc.Add(new Paragraph("Déclenchée le : " + date + " à " +
heure, police));
      if (status == "terminée")
      {
            doc.Add(new Paragraph("Terminée le : " + dateRetour + "
à " + heureRetour, police));
```

```
      }

      doc.Add(new Paragraph("Adresse du sinistre : " + adr,
police));


      doc.Add(new Paragraph(" "));
      Paragraph pNatureSinistre = new Paragraph("Nature du sinistre
: " + sinistre, maPolice);
      doc.Add(pNatureSinistre);

      Paragraph pMotif = new Paragraph("Motif : " + motif, police);
      pMotif.Alignment = Element.ALIGN_JUSTIFIED; // Justifier le
texte
      doc.Add(pMotif);

      Paragraph pCompteRendu = new Paragraph("Compte rendu : " +
compteRendu, police);
      pCompteRendu.Alignment = Element.ALIGN_JUSTIFIED;
      doc.Add(pCompteRendu);

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

      Paragraph pEnginsMobilises = new Paragraph("Engins mobilisés
:", maPolice);
      doc.Add(pEnginsMobilises);


      // Récupération des engins mobilisés
      DataRow[] partis =
MesDatas.DsGlobal.Tables["PartirAvec"].Select("idMission = " +
idMission);

      foreach (DataRow engin in partis)
      {
            string nomEngin =
MesDatas.DsGlobal.Tables["TypeEngin"].Select("code = '" +
engin["codeTypeEngin"] + "'")[0]["nom"].ToString();
            string texte = $"* {nomEngin}
{idCaserne}-{engin["codeTypeEngin"]}-{engin["numeroEngin"]}";

            if (engin["reparationsEventuelles"] != DBNull.Value &&
```

```
engin["reparationsEventuelles"].ToString() != "")
            {
                  texte += " (Réparations : " +
engin["reparationsEventuelles"] + ")";
            }

            else
            {
                  texte += " (Aucune réparation nécessaire)";
            }
            Paragraph pEngin = new Paragraph(texte, police);
            pEngin.Alignment = Element.ALIGN_JUSTIFIED;
            doc.Add(pEngin);
      }

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

      Paragraph pPompierMobilises = new Paragraph("Pompiers
mobilisés :", maPolice);
      pPompierMobilises.PaddingTop = 10f; // espace entre les
sections
      doc.Add(pPompierMobilises);

      // Récupération des pompiers mobilisés
      {...};
      catch (Exception ex)
      {
            MessageBox.Show("Erreur : " + ex.Message);
      }

      finally
      {
            doc.Close();
      }
}
```

Navigation en mode liaison de données entre les véhicules d'une caserne: BindingSource

```
public partial class frmGestionEngins : Form
{
      public frmGestionEngins()
      {
      InitializeComponent();
      }

      BindingSource bsEngins = new BindingSource();
      private string cheminImage = @"../../Images/ImagesEngins/";
      DataTable dtEngins = new DataTable();

      private void frmGestionEngins_Load(object sender, EventArgs
e)
      {
      string req = @"select *, idCaserne || '-' || codeTypeEngin ||
'-' || numero as ID " +
                        "from Engin";
      SQLiteDataAdapter daEngins = new SQLiteDataAdapter(req,
Connexion.Connec);

      daEngins.Fill(dtEngins);
      bsEngins.DataSource = dtEngins;
      bsEngins.Filter = "idCaserne = 1";
      bsEngins.MoveFirst();
      lblNom.DataBindings.Add("Text", bsEngins, "ID");
      lblAcquisition.DataBindings.Add("Text", bsEngins,
"dateReception");
      lblCodeType.DataBindings.Add("Text", bsEngins,
"codeTypeEngin");
      lblEnPanne.DataBindings.Add("Text", bsEngins, "enPanne");
      lblEnMission.DataBindings.Add("Text", bsEngins, "enMission");

      disponibiliteVehicule();
      chargerImage();

      cboCaserne.DataSource = MesDatas.DsGlobal.Tables["Caserne"];
      cboCaserne.DisplayMember = "nom";
      cboCaserne.ValueMember = "id";

      Connexion.FermerConnexion();
```

```
      }

      private void disponibiliteVehicule()
      {
      if (lblEnPanne.Text == "1")
      {
            lblDisponibilite.Text = "En panne";
      }
      else if (lblEnMission.Text == "1")
      {
            lblDisponibilite.Text = "En mission";
      }
      else
      {
            lblDisponibilite.Text = "Disponible";
      }
      }

      private void chargerImage()
      {
      switch (lblCodeType.Text)
      {
            case "FPT":
                  pctEngin.Image = Image.FromFile(cheminImage +
"FPT.png");
                  break;
            case "CCF":
                  pctEngin.Image = Image.FromFile(cheminImage +
"CCF.png");
                  break;
            case "BRS":
                  pctEngin.Image = Image.FromFile(cheminImage +
"BRS.jpg");
                  break;
            case "EPA":
                  pctEngin.Image = Image.FromFile(cheminImage +
"EPA.png");
                  break;
            case "FCYN":
                  pctEngin.Image = Image.FromFile(cheminImage +
"FCYN.png");
                  break;
```

```
            case "VID":
                  pctEngin.Image = Image.FromFile(cheminImage +
"VID.png");
                  break;
            case "VPC":
                  pctEngin.Image = Image.FromFile(cheminImage +
"VPC.png");
                  break;
            case "VSAV":
                  pctEngin.Image = Image.FromFile(cheminImage +
"VSAV.png");
                  break;
            case "VSR":
                  pctEngin.Image = Image.FromFile(cheminImage +
"VSR.png");
                  break;
            case "VSS":
                  pctEngin.Image = Image.FromFile(cheminImage +
"VSS.jpg");
                  break;
      }
      }

      private void btnRetour_Click(object sender, EventArgs e)
      {
      this.Close();
      }

      private void btnFirst_Click(object sender, EventArgs e)
      {
      bsEngins.MoveFirst();
      chargerImage();
      disponibiliteVehicule();
      }

      private void btnLast_Click(object sender, EventArgs e)
      {
      bsEngins.MoveLast();
      chargerImage();
      disponibiliteVehicule();
      }
```

```
      private void btnSuivant_Click(object sender, EventArgs e)
      {
      if (bsEngins.Position == bsEngins.Count - 1)
      {
            bsEngins.MoveFirst();
      }
      else
      {
            bsEngins.MoveNext();
      }

      chargerImage();
      disponibiliteVehicule();
      }

      private void btnPrecedent_Click(object sender, EventArgs e)
      {
      if (bsEngins.Position == 0)
      {
            bsEngins.MoveLast();
      }
      else
      {
            bsEngins.MovePrevious();
      }
      chargerImage();
      disponibiliteVehicule();
      }

      private void cboCaserne_SelectionChangeCommitted(object
sender, EventArgs e)
      {
      bsEngins.Filter = $"idCaserne = {cboCaserne.SelectedValue}";
      bsEngins.MoveFirst();
      chargerImage();
      disponibiliteVehicule();
      }
}
```