Compte rendu :

- La **déclaration des variables globales du formulaire de départ**

```
private TableauDeBord tableauDeBord;    // Déclarer le volet du tableau de bord
private AjoutMissions ajoutMissions;    // Déclarer le volet d'ajout de missions
private GestionEngins gestionEngins;    // Déclarer le volet de gestion des engins
private GestionPerso gestionPerso;      // Déclarer le volet de gestion du personnel
private Statistique stats;              // Déclarer le volet de statistiques
private SQLiteConnection con;           // Déclarer la connection à la base de données
private DataSet monDs;                  // Déclarer le DataSet Global
private DataTable dtMissionsFormatees;  // Déclarer une table de missions formatées pour le tableau de bord
private Bouton btnActuel;               // Déclarer le bouton du volet actuellement ouvert
```

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

```cs
using System.Data;
using System.Data.SQLite;
using System.Globalization;


namespace UC_GestionPerso
{
    public delegate void validateLogin(Dictionary<string, string> KeyLogin);

    5 références
    public partial class GestionPerso : UserControl
    {
        SQLiteConnection _con;
        UCLogin login;

        int idCaserneInit;
        string codeGradeInit;
        bool dateF;
        int enCongeInit;

        private UCRichButton Selected;
        private UCRichButton SelectedPompier;
        0 références
        public bool Connected
        {
            get { return login != null && login.Connected; }
            set { if(login != null) login.Connected = value; }
        }

        public validateLogin validateLoginDel;

        0 références
        public GestionPerso()
        {
            InitializeComponent();
        }

        1 référence
        public GestionPerso(SQLiteConnection con) : base()
        {
            InitializeComponent();
            _con = con;

            if(con.State == ConnectionState.Closed)
            {
                throw new Exception("Illegal argument : La connexion doit être ouverte 😡");
            }

            login = new UCLogin(_con);
            login.OnLoginResult = HandleLoginResult;
        }
```

```cs
        private void UCGestionPerso_Load(object sender, EventArgs e)
        {
            try
            {
                //On remplit le panel avec des boutons (Caserne)
                string caserneCommand = "SELECT nom, id FROM Caserne";
                SQLiteCommand cmd = new SQLiteCommand(caserneCommand, _con);
                SQLiteDataReader data = cmd.ExecuteReader();

                while(data.Read())
                {
                    string nom = data.GetString(0);
                    string id = data.GetValue(1).ToString();

                    UCRichButton btn = new UCRichButton(nom);
                    btn.Tag = id;
                    btn.clickReturnTag += btnClickForCaserne;

                    flpCaserne.Controls.Add(btn);
                }

                //On charge nos combobox
                ChargercboGrade();
                ChargercboCaserne();
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        1 référence
        private void btnClickForCaserne(UCRichButton clicked, object tag)
        {
            if(Selected != null && Selected != clicked)
            {
                Selected.isSelected = false;
                Selected.BackColor = Color.White;
                Selected.ForeColor = Color.Black;
            }
            clicked.isSelected = true;
            Selected = clicked;

            grpCaserne.Visible = false;
            grpContact.Visible = false;
            grpIdentite.Visible = false;
            lblMatricule.Visible = false;
            btnInfo.Visible = false;

            refreshPompier();
        }
```

```
100         private void btnClickPompier(UCRichButton clicked, object tag)
101         {
102             if(SelectedPompier != null && clicked != SelectedPompier)
103             {
104                 SelectedPompier.isSelected = false;
105                 SelectedPompier.BackColor = Color.White;
106                 SelectedPompier.ForeColor = Color.Black;
107             }
108             clicked.isSelected = true;
109             SelectedPompier = clicked;
110             formPompier();
111         }
112
```

```
114         private void refreshPompier()
115         {
116             try
117             {
118                 if(Selected.Tag == null)
119                 {
120                     return;
121                 }
122                 flpPompier.Controls.Clear();
123                 string cmdPompier = $@"SELECT matriculePompier FROM Affectation WHERE idCaserne = {Selected.Tag}";
124                 SQLiteCommand cd = new SQLiteCommand(cmdPompier, _con);
125                 SQLiteDataReader dr = cd.ExecuteReader();
126
127                 while(dr.Read())
128                 {
129                     int id = dr.GetInt32(0);
130                     string cmdNom = $@"SELECT CONCAT(prenom, ' ', nom) FROM Pompier WHERE matricule = {id}";
131                     SQLiteCommand cd1 = new SQLiteCommand(cmdNom, _con);
132                     string nom = (string)cd1.ExecuteScalar();
133
134                     string cmdGrade = $@"SELECT codeGrade FROM Pompier WHERE matricule = {id}";
135                     SQLiteCommand cd2 = new SQLiteCommand(cmdGrade, _con);
136                     string grade = (string)cd2.ExecuteScalar();
137                     UCRichButton btn;
138                     if(grade != null)
139                     {
140                         var varimg = SAE_Aparcio_Claudel_Meral.Properties.Resources.ResourceManager.GetObject(grade);
141                         if(varimg is Image img)
142                         {
143                             btn = new UCRichButton(nom, id.ToString(), (Image)varimg);
144                         }
145                         else
146                         {
147                             btn = new UCRichButton(nom, id.ToString(), SAE_Aparcio_Claudel_Meral.Properties.Resources.SAP
148                         }
149                         btn.Width = flpPompier.Width - 30;
150                         btn.Tag = id;
151                         btn.clickReturnTag += btnClickPompier;
152                         flpPompier.Controls.Add(btn);
153                     }
154                 }
155                 dr.Close();
156             }
157             catch(InvalidOperationException)
158             {
159                 MessageBox.Show("Erreur : connexion fermée !");
160             }
161             catch(SQLiteException)
162             {
163                 MessageBox.Show("Erreur dans la requête SQL !");
164             }
165             catch(Exception err)
166             {
167                 MessageBox.Show(err.ToString());
168             }
169             finally { }
```

```
1 reference
private void formPompier()
{
    try
    {
        InitPompierInfo();
        grpIdentite.Visible = true;
        grpContact.Visible = true;
        lblMatricule.Visible = true;
        btnInfo.Visible = true;

        int id = Convert.ToInt32(SelectedPompier.Tag);
        lblMatricule.Text = "Matricule " + id;
        lblMatricule.Tag = id;

        string requete = $@"SELECT * FROM Pompier WHERE matricule = {id}";
        SQLiteCommand cd = new SQLiteCommand(requete, _con);
        SQLiteDataReader dr = cd.ExecuteReader();


        while(dr.Read())
        {
            lblNom.Text = dr.GetString(1);
            lblPrenom.Text = dr.GetString(2);
            lblSexe.Text = dr.GetString(3);
            lblDateNaissance.Text = dr.GetString(4);

            string type = dr.GetString(5);

            if(type == "p")
            {
                rdbPro.Checked = true;
            }
            else if(type == "v")
            {
                rdbVolontaire.Checked = true;
            }

            lblTel.Text = dr.IsDBNull(6) ? "Non renseigné" : dr.GetString(6);
            lblBip.Text = dr.GetInt32(7).ToString();
            int enConge = dr.GetInt32(9);
            string codeGrade = dr.GetString(10);

            txtGradeCode.Text = codeGrade;
            codeGradeInit = codeGrade;
            cboGrade.SelectedValue = codeGrade;

            if(enConge == 1)
            {
                chbConge.Checked = true;
                enCongeInit = 1;
            }
            lblDateEmbauche.Text = dr.GetString(11);
        }
        dr.Close();
```

```
228                 ///Affectations
229                 string cmdCaserne = $@"SELECT * FROM Affectation WHERE matriculePompier = " + id + " ORDER BY dateA";
230                 SQLiteCommand cd1 = new SQLiteCommand(cmdCaserne, _con);
231                 SQLiteDataReader dr1 = cd1.ExecuteReader();
232
233                 int idCaserne = 0;
234
235                 while(dr1.Read())
236                 {
237                     string affectation = "";
238                     if(!dr1.IsDBNull(2))
239                     {
240                         dateF = true; //il existe une date fin
241
242                         DateTime dateNow = DateTime.Now;
243                         DateTime dateFin = DateTime.ParseExact(dr1.GetString(2), "yyyy-MM-dd", CultureInfo.InvariantCulture);
244
245                         int comp = DateTime.Compare(dateNow, dateFin);
246
247                         if(dr1.GetString(2) == null || comp > 0)
248                         {
249                             idCaserne = dr1.GetInt32(3);
250                             affectation += dr1.GetString(1);
251                         }
252                     }
253                     else
254                     {
255                         dateF = false;
256                         idCaserne = dr1.GetInt32(3);
257                         affectation += dr1.GetString(1);
258                     }
259                     cboCaserne.SelectedValue = idCaserne;
260                     affectation += " " + cboCaserne.Text;
261
262                     rtbAffec.Text += affectation + "\n";
263                 }
264                 dr1.Close();
265                 idCaserneInit = idCaserne;
266
267                 ///Habilitations
268                 string cmdHab = $@"SELECT * FROM Passer WHERE matriculePompier = {id}";
269                 SQLiteCommand cd2 = new SQLiteCommand(cmdHab, _con);
270                 SQLiteDataReader dr2 = cd2.ExecuteReader();
271
272                 while(dr2.Read())
273                 {
274                     string rq = $@"SELECT libelle FROM Habilitation WHERE id = {dr2.GetInt32(1)}";
275                     SQLiteCommand cm = new SQLiteCommand(rq, _con);
276                     string nom = (string)cm.ExecuteScalar();
277
278                     string hab = nom.Split('-')[0];
279
280                     rtbHab.Text += hab + " " + dr2.GetString(2) + "\n";
281                 }
```

```
                var varimg = SAE_Aparcio_Claudel_Meral.Properties.Resources.ResourceManager.GetObject(codeGradeInit);
                if(varimg is Image img)
                {
                    pic.Image = img;
                }
                else
                {
                    pic.Image = SAE_Aparcio_Claudel_Meral.Properties.Resources.Default;
                }
                if(login.Connected)
                {
                    cboCaserne.Enabled = true;
                    cboGrade.Enabled = true;
                    chbConge.Enabled = true;
                }
            }
            catch(InvalidOperationException)
            {
                MessageBox.Show("Erreur : la connexion est fermé !");
            }
            catch(SQLiteException)
            {
                MessageBox.Show("Erreur dans la requête SQL !");
            }
            catch(Exception err)
            {
                MessageBox.Show(err.ToString());
            }
            finally { }
        }

        1 référence
        private void btnInfo_Click(object sender, EventArgs e)
        {
            if(!login.Connected)
            {
                connexionRequest();
                return;
            }
            grpCaserne.Visible = true;
            btnInfo.Visible = false;
            btnModif.Visible = true;
        }
        2 références
        private void ChargementCbo(DataTable dt, String col1, String col2, ComboBox c)
        {
            c.Items.Clear();
            c.DataSource = dt;
            c.DisplayMember = col1;
            c.ValueMember = col2;
        }
```

```
        1 référence
        private void ChargercboGrade()
        {
            string cmdGrade = $@"SELECT code, libelle FROM Grade";
            SQLiteCommand cd1 = new SQLiteCommand(cmdGrade, _con);
            SQLiteDataReader dr1 = cd1.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Columns.Add("code");
            dt.Columns.Add("libelle");
            while(dr1.Read())
            {
                DataRow ligne = dt.NewRow();
                ligne[0] = dr1.GetString(0);
                ligne[1] = dr1.GetString(1);
                dt.Rows.Add(ligne);
            }
            dr1.Close();
            ChargementCbo(dt, "libelle", "code", cboGrade);
        }

        1 référence
        private void ChargercboCaserne()
        {
            string cmdCaserne = $@"SELECT * FROM Caserne";
            SQLiteCommand cd1 = new SQLiteCommand(cmdCaserne, _con);
            SQLiteDataReader dr1 = cd1.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Columns.Add("id");
            dt.Columns.Add("nom");
            while(dr1.Read())
            {
                DataRow ligne = dt.NewRow();
                ligne[0] = dr1.GetInt32(0);
                ligne[1] = dr1.GetString(1);
                dt.Rows.Add(ligne);
            }
            dr1.Close();
            ChargementCbo(dt, "nom", "id", cboCaserne);
        }
```

```
352     private void ChargercboCaserne()
353     {
354         string cmdCaserne = $@"SELECT * FROM Caserne";
355         SQLiteCommand cd1 = new SQLiteCommand(cmdCaserne, _con);
356         SQLiteDataReader dr1 = cd1.ExecuteReader();
357         DataTable dt = new DataTable();
358         dt.Columns.Add("id");
359         dt.Columns.Add("nom");
360         while(dr1.Read())
361         {
362             DataRow ligne = dt.NewRow();
363             ligne[0] = dr1.GetInt32(0);
364             ligne[1] = dr1.GetString(1);
365             dt.Rows.Add(ligne);
366         }
367         dr1.Close();
368         ChargementCbo(dt, "nom", "id", cboCaserne);
369     }
370
        1 référence
371     private void InitPompierInfo()
372     {
373
374         //Info
375         lblNom.Text = String.Empty;
376         lblPrenom.Text = String.Empty;
377         lblMatricule.Text = String.Empty;
378         lblSexe.Text = String.Empty;
379         lblDateNaissance.Text = String.Empty;
380         lblDateEmbauche.Text = String.Empty;
381         rdbPro.Checked = false;
382         rdbVolontaire.Checked = false;
383
384         //Contact
385         lblBip.Text = String.Empty;
386         lblTel.Text = String.Empty;
387
388         //Carrière
389         txtGradeCode.Text = String.Empty;
390         cboCaserne.SelectedIndex = -1;
391         cboGrade.SelectedIndex = -1;
392         rtbAffec.Text = String.Empty;
393         rtbHab.Text = String.Empty;
394         chbConge.Checked = false;
395         grpCaserne.Visible = false;
396         btnModif.Visible = false;
397         btnModif.Text = "Modifier";
398
399     }
400
```

```
401     private void cboGrade_SelectedIndexChanged(object sender, EventArgs e)
402     {
403         if(cboGrade.SelectedIndex != -1)
404         {
405             txtGradeCode.Text = cboGrade.SelectedValue.ToString();
406         }
407     }
408
409
410
411
412     private bool _pendingCreatePompier = false;
413
        1 référence
414     private void btnCreate_Click(object sender, EventArgs e)
415     {
416         if(!login.Connected)
417         {
418             _pendingCreatePompier = true;
419             connexionRequest();
420             return;
421         }
422
423         OpenCreatePompier();
424     }
        2 références
425     private void OpenCreatePompier()
426     {
427         frmCreationPompier creerPompier = new frmCreationPompier(_con);
428         DialogResult res = creerPompier.ShowDialog();
429
430         if(res == DialogResult.OK)
431         {
432             MessageBox.Show("Vous avez créé un nouveau pompier");
433             refreshPompier();
434         }
435     }
```

```
    private void btnChanger_Click(object sender, EventArgs e)
    {
        SQLiteTransaction transaction = null;
        transaction = _con.BeginTransaction();

        try
        {
            int id = Convert.ToInt32(lblMatricule.Tag);

            int enConge = 0;
            if(chbConge.Checked)
            {
                enConge = 1;
            }

            int idCaserne = Convert.ToInt32(cboCaserne.SelectedValue);
            string codeGrade = txtGradeCode.Text;

            if(codeGrade != codeGradeInit || enCongeInit != enConge)
            {
                SQLiteCommand cmdTabPompier = new SQLiteCommand($@"UPDATE Pompier SET enConge = {enConge}, codeGrade = '{codeGrade}' WHERE matricule = {id}", _con, transaction);
                cmdTabPompier.ExecuteNonQuery();

                MessageBox.Show("Les données ont bien été mises à jour.");

                codeGradeInit = codeGrade;
                enCongeInit = enConge;
            }

            if(idCaserne != idCaserneInit)
            {
                string date = DateTime.Now.ToString("yyyy-MM-dd", CultureInfo.InvariantCulture);
                if(!dateF)
                {
                    SQLiteCommand cmdAffectation = new SQLiteCommand($@"UPDATE Affectation SET dateFin = '{date}' WHERE matriculePompier = {id} AND idCaserne = {idCaserneInit}", _con, trans
                    cmdAffectation.ExecuteNonQuery();
                }
                idCaserneInit = idCaserne;

                SQLiteCommand cmdAffectationAdd = new SQLiteCommand($@"INSERT INTO Affectation (matriculePompier, dateA, idCaserne) VALUES ({id}, '{date}', {idCaserneInit})", _con, transact
                cmdAffectationAdd.ExecuteNonQuery();

                MessageBox.Show("Le pompier a bien été affecté à une caserne.");
            }

            transaction.Commit();
        }
        catch(SQLiteException ex)
        {
            if(ex.ResultCode == SQLiteErrorCode.Constraint)
            {
                MessageBox.Show("Erreur : Vous ne pouvez pas affecter le pompier à une autre caserne le même jour!");
            }
            else
            {
                MessageBox.Show("Erreur SQLite : " + ex.Message);
            }
```

```
            }
            catch(SQLiteException ex)
            {
                if(ex.ResultCode == SQLiteErrorCode.Constraint)
                {
                    MessageBox.Show("Erreur : Vous ne pouvez pas affecter le pompier à une autre caserne le même jour!");
                }
                else
                {
                    MessageBox.Show("Erreur SQLite : " + ex.Message);
                }
            }
            catch(Exception ex)
            {
                transaction.Rollback();
                MessageBox.Show(ex.Message);
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                transaction.Dispose();
            }
        }

        1 référence
        private void btnModif_Click(object sender, EventArgs e)
        {
            if(!login.Connected)
            {
                connexionRequest();
                return;
            }

            bool isInEditMode = cboCaserne.Enabled;

            if(isInEditMode)
            {
                cboCaserne.Enabled = false;
                cboGrade.Enabled = false;
                chbConge.Enabled = false;
                btnModif.Text = "Modifier";
                btnChanger.Visible = false;

                cboGrade.SelectedValue = codeGradeInit;
                cboCaserne.SelectedValue = idCaserneInit;
                chbConge.Checked = (enCongeInit == 1);
            }
            else
            {
                cboCaserne.Enabled = true;
                cboGrade.Enabled = true;
                chbConge.Enabled = true;
                btnModif.Text = "Annuler";
                btnChanger.Visible = true;
            }
```

```
3 références
private void connexionRequest()
{
    if(login == null)
    {
        login = new UCLogin(_con);
    }
    login.OnLoginResult = HandleLoginResult;
    login.OnLoginCancelled = HandleLoginCancelled;

    Parent.Controls.Add(login);
    login.BringToFront();
    login.Left = (this.Width - login.Width) / 2;
    login.Top = (this.Height - login.Height) / 2;
    login.Visible = true;
    this.Enabled = false;
}

1 référence
private void HandleLoginCancelled()
{
    this.Enabled = true;
    login.Visible = false;
}

2 références
private void HandleLoginResult(bool success)
{
    this.Enabled = true;
    login.Visible = false;

    if(success)
    {
        btnModif.Text = "Annuler";
        cboCaserne.Enabled = true;
        cboGrade.Enabled = true;
        chbConge.Enabled = true;

        if(_pendingCreatePompier)
        {
            _pendingCreatePompier = false;
            OpenCreatePompier();
        }
    }
}

}
}
```

- **Le code complet du User Control** que vous jugez le plus intéressant.:
- Tableau de Bord:

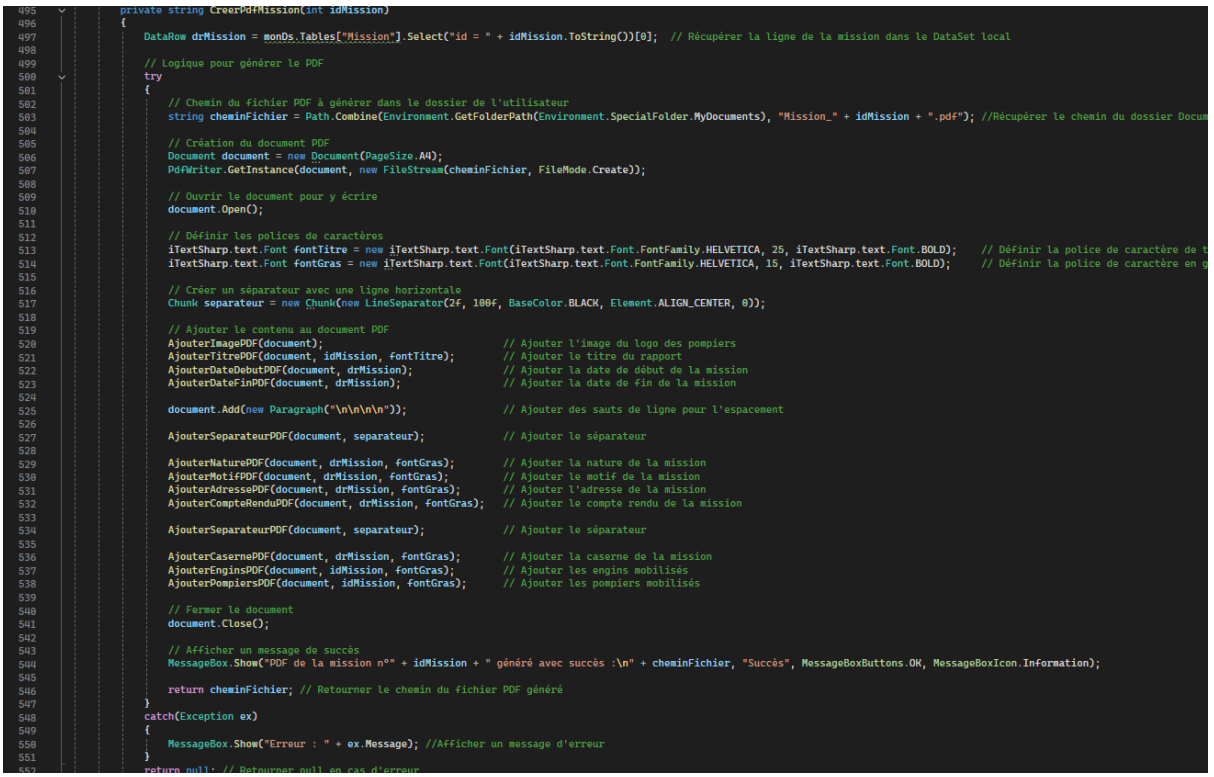
```
namespace UC_TableauDeBord
{

    public delegate void AjouterMissionBD(Mission mission, string compteRendu, DataTable enginsEnPanne); //Déclaration de la signature du délégué pour ajouter une mission à la base de donnée
    public delegate DataTable GetEnginsMission(int idMission); //Déclaration de la signature du délégué pour récupérer les engins d'une mission
    public delegate string CreerPdfMission(int idMission); //Déclaration de la signature du délégué pour créer le PDF de la mission


    public partial class TableauDeBord : UserControl
    {
        1 référence
        public TableauDeBord()
        {
            InitializeComponent();
        }

        private List<Mission> listMissions = new List<Mission>();       //Liste des missions
        public AjouterMissionBD ajouterMissionBD;                       //Instance du délégué pour ajouter une mission à la base de données
        public GetEnginsMission getEnginsMission;                       //Instance du délégué pour récupérer les engins d'une mission
        public CreerPdfMission creerPdfMission;                         //Instance du délégué pour créer le PDF de la mission
        private bool switchCouleur = true;                              //Variable pour alterner les couleurs des missions
        private Color couleurSecondaire = Color.FromArgb(234, 234, 234);  //Couleur grise pour les missions

        2 références
        private Mission InitMission(DataRow dr)
        {
            Mission mission = new Mission(dr);          //Création d'une nouvelle mission à partir de la ligne du DataTable
            switchCouleurMission(mission);              //Appel de la méthode pour alterner la couleur de la mission
            mission.terminerMission = TerminerMission;  //Ajout de l'événement pour terminer la mission
            mission.creerPdfMission = CreerPdfMission;  //Ajout de l'événement pour créer le PDF de la mission
            return mission; //Retourne la mission créée
        }

        1 référence
        public void LoadMissions(DataTable dt)
        {
            listMissions.Clear(); //Vider la liste avant de la remplir
            foreach (DataRow dr in dt.Rows)
            {
                Mission mission = InitMission(dr);  //Création d'une nouvelle mission à partir de la ligne du DataTable
                listMissions.Add(mission);          //Ajout de la mission à la liste
            }
            TriParIdDecroissant(listMissions);  //Tri des missions par ID décroissant
            DisplayMissions();                  //Appel de la méthode pour afficher les missions
        }
```

```
60          public void RemoveMission(Mission mission)
61          {
62              listMissions.Remove(mission);   //Suppression de la mission de la liste
63              DisplayMissions();              //Appel de la méthode pour afficher les missions
64          }
65
66          private void TriParIdDecroissant(List<Mission> missions)
67          {
68              missions.Sort((x, y) => y.MissionID.CompareTo(x.MissionID)); //Tri des missions par ID décroissant
69          }
70
71          public void DisplayMissions()
72          {
73              panelMissions.Controls.Clear(); //Vider le panel avant d'ajouter les missions
74              int top = 6;                    //Position en hauteur de la première mission
75              int left = 30;                  //Position en largeur de la première mission
76              foreach (Mission mission in listMissions)
77              {
78                  if (!sckbEnCours.Checked || (sckbEnCours.Checked && mission.EstEnCours)) //Si la checkbox n'est pas cochée, ou que la checkbox est cochée et que la mission est en cours :
79                  {
80                      mission.Location = new Point(left, top);    //Position de la mission
81                      panelMissions.Controls.Add(mission);        //Ajout de la mission au panel
82                      top += mission.Height + 10;                 //Espace entre les missions
83                  }
84              }
85          }
86
87          private void TerminerMission(object sender, EventArgs e, int idMission)
88          {
89              frmCompteRendu frm = new frmCompteRendu();  //Création d'une nouvelle instance de la fenêtre de terminaison de mission
90              if (frm.ShowDialog() == DialogResult.OK)    //Affichage de la fenêtre et gestion de la fermeture du formulaire
91              {
92                  Mission mission = (Mission)sender;  //Récupération de la mission sélectionnée
93                  mission.Terminer();                 //La mission n'est plus en cours, ajout automatique de la date de fin à la mission sélectionnée
94                  if (ajouterMissionBD != null)       //Si le délégué n'est pas nul
95                  {
96                      DataTable engins = getEngins(idMission);            //Récupération des engins
97                      ajouterMissionBD(mission, frm.CompteRendu, engins); //Appel du délégué pour ajouter la mission à la base de données
98                  }
99                  DisplayMissions(); //Appel de la méthode pour afficher les missions
100             }
101         }
102
103
104         public void AjouterMission(DataRow dr)
105         {
106             Mission item = InitMission(dr);
107             listMissions.Insert(0, item);
108             DisplayMissions();
109         }
110
```

```
        1 référence
        private DataTable getEngins(int idMission)
        {
            DataTable engins = new DataTable();         //Création d'un DataTable pour stocker les engins de la mission
            frmPanneEngins frm = new frmPanneEngins();  //Création d'une nouvelle instance de la fenêtre de panne d'engins
            frm.Remplir(getEnginsMission(idMission));   //Remplissage de la fenêtre avec les engins de la mission
            if (frm.ShowDialog() == DialogResult.OK)    //Affichage de la fenêtre et gestion de la fermeture du formulaire
            {
                //Récupération des engins sélectionnés dans le formulaire
                engins = frm.getEngins(); //Appel de la méthode pour récupérer les engins en panne
            }
            return engins; //Retourne le DataTable
        }

        1 référence
        private void switchCouleurMission(Mission mission)
        {
            switchCouleur = !switchCouleur; //Alterner la couleur
            if (switchCouleur)
            {
                mission.Couleur = couleurSecondaire; //Couleur grise
            }
        }

        1 référence
        private void sckbEnCours_CheckedChanged(object sender, EventArgs e)
        {
            DisplayMissions(); //Appel de la méthode pour afficher les missions
        }

        1 référence
        private string CreerPdfMission(int idMission)
        {
            if (creerPdfMission != null) //Si le délégué n'est pas nul
            {
                return creerPdfMission(idMission); //Appel du délégué pour créer le PDF de la mission
            }
            else
            {
                return string.Empty; //Retourne une chaîne vide si le délégué est nul
            }
        }
    }
}
```

- **Un extrait du code permettant de générer le pdf**

```
private void btnCreerPdf_Click(object sender, EventArgs e)
{
    if (creerPdfMission != null) // Si le délégué n'est pas nul
    {
        string cheminFichier = creerPdfMission(MissionID); // On appelle le délégué pour créer un PDF de la mission
        // Ouvrir le PDF généré
        if (File.Exists(cheminFichier))
        {
            frmOuverturePDF frmOuverturePDF = new frmOuverturePDF(); // On crée une instance de la fenêtre de confirmation
            if (frmOuverturePDF.ShowDialog() == DialogResult.OK)
            {
                Process.Start(new ProcessStartInfo(cheminFichier) { UseShellExecute = true }); // On ouvre le PDF si l'utilisateur a cliqué sur Oui
            }
        }
        else
        {
            MessageBox.Show("Le fichier n'existe pas.", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```



```
private string CreerPdfMission(int idMission)
{
    DataRow drMission = monDs.Tables["Mission"].Select("id = " + idMission.ToString())[0];  // Récupérer la ligne de la mission dans le DataSet local

    // Logique pour générer le PDF
    try
    {
        // Chemin du fichier PDF à générer dans le dossier de l'utilisateur
        string cheminFichier = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments), "Mission_" + idMission + ".pdf"); //Récupérer le chemin du dossier Docum

        // Création du document PDF
        Document document = new Document(PageSize.A4);
        PdfWriter.GetInstance(document, new FileStream(cheminFichier, FileMode.Create));

        // Ouvrir le document pour y écrire
        document.Open();

        // Définir les polices de caractères
        iTextSharp.text.Font fontTitre = new iTextSharp.text.Font(iTextSharp.text.Font.FontFamily.HELVETICA, 25, iTextSharp.text.Font.BOLD);    // Définir la police de caractère de t
        iTextSharp.text.Font fontGras = new iTextSharp.text.Font(iTextSharp.text.Font.FontFamily.HELVETICA, 15, iTextSharp.text.Font.BOLD);     // Définir la police de caractère en g

        // Créer un séparateur avec une ligne horizontale
        Chunk separateur = new Chunk(new LineSeparator(2f, 100f, BaseColor.BLACK, Element.ALIGN_CENTER, 0));

        // Ajouter le contenu au document PDF
        AjouterImagePDF(document);                                  // Ajouter l'image du logo des pompiers
        AjouterTitrePDF(document, idMission, fontTitre);            // Ajouter le titre du rapport
        AjouterDateDebutPDF(document, drMission);                   // Ajouter la date de début de la mission
        AjouterDateFinPDF(document, drMission);                     // Ajouter la date de fin de la mission

        document.Add(new Paragraph("\n\n\n\n"));                    // Ajouter des sauts de ligne pour l'espacement

        AjouterSeparateurPDF(document, separateur);                 // Ajouter le séparateur

        AjouterNaturePDF(document, drMission, fontGras);            // Ajouter la nature de la mission
        AjouterMotifPDF(document, drMission, fontGras);             // Ajouter le motif de la mission
        AjouterAdressePDF(document, drMission, fontGras);           // Ajouter l'adresse de la mission
        AjouterCompteRenduPDF(document, drMission, fontGras);       // Ajouter le compte rendu de la mission

        AjouterSeparateurPDF(document, separateur);                 // Ajouter le séparateur

        AjouterCasernePDF(document, drMission, fontGras);           // Ajouter la caserne de la mission
        AjouterEnginsPDF(document, idMission, fontGras);            // Ajouter les engins mobilisés
        AjouterPompiersPDF(document, idMission, fontGras);          // Ajouter les pompiers mobilisés

        // Fermer le document
        document.Close();

        // Afficher un message de succès
        MessageBox.Show("PDF de la mission n°" + idMission + " généré avec succès :\n" + cheminFichier, "Succès", MessageBoxButtons.OK, MessageBoxIcon.Information);

        return cheminFichier; // Retourner le chemin du fichier PDF généré
    }
    catch(Exception ex)
    {
        MessageBox.Show("Erreur : " + ex.Message); //Afficher un message d'erreur
    }
    return null; // Retourner null en cas d'erreur
```

- Le code permettant la **navigation en mode liaison de données entre les véhicules d'une caserne.**

```
public EnginsDisplay(BindingSource baseEngin)
{
    InitializeComponent();
    EnginsList = baseEngin;
    lblCodeEngin.DataBindings.Add("Text", EnginsList, "numero");
    lblDate.DataBindings.Add("Text", EnginsList, "dateReception");
    chkEnPanne.DataBindings.Add("Checked", EnginsList,"enPanne");
    chkEnMission.DataBindings.Add("Checked", EnginsList, "enMission");
}
```

```
private void refreshImage()
{
    Visible = false;
    timer.Start();
    DataRowView currentEngin = (DataRowView)EnginsList.Current;
    string codeType = currentEngin["codeTypeEngin"].ToString();
    var varimg = Properties.Resources.ResourceManager.GetObject(codeType);
    if(varimg is Image img)
    {
        pctEnginImage.Image = img;
    }
    else
    {
        pctEnginImage.Image = Properties.Resources.Default;
    }
}
```

```
private void UCGestionEngin_Load(object sender, EventArgs e)
{
    if(monDs == null)
    {
        return;
    }
    if (!monDs.Relations.Contains("RelationsCaserneEngins"))
    {

        monDs.Relations.Add("RelationsCaserneEngins",
            monDs.Tables["Caserne"].Columns["id"],
            monDs.Tables["Engin"].Columns["idCaserne"]);
    }

    BindingSource baseCaserne = new BindingSource(monDs,"Caserne");
    cbxChoixCaserne.DataSource = baseCaserne;
    cbxChoixCaserne.DisplayMember = "nom";
    cbxChoixCaserne.ValueMember = "id";

    BindingSource baseEngin = new BindingSource(baseCaserne, "RelationsCaserneEngins");

    display = new EnginsDisplay(baseEngin);
    display.Dock = DockStyle.Fill;
    cbxChoixCaserne.Focus();
    setNext(false);
    setPrevious(false);


}
```