Table des matières

I. Déclaration globale des variables

```
public partial class FormMealEtUnPlat : Form
{
    Dictionary<string, string> IngredientsChoisis = new Dictionary<string, string>();
    // Un dictionnaire pour stocker les ingrédients choisis, où la clé est le nom de l'ingrédient et la valeur est sa quantité ou autre information associée.

    List<string> codeRecette = new List<string>();
    // Une liste pour stocker les codes des recettes sélectionnées.

    string chcon = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\louisschlegel\source\repos\SAE-D21-main\SAE2.4\baseFrigo.accdb";
    //string chcon = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\acker\OneDrive\Bureau\SAE\SAE24_ExploitationBDD_GIT\SAE2.4\baseFrigo.accdb";
    // Initialisation de la chaîne de connexion à la base de données.

    DataSet ds = new DataSet();
    // Création d'un DataSet qui contiendra l'ensemble de nos tables.

    BindingSource bs = new BindingSource();
    // Un BindingSource pour lier les données du DataSet à des contrôles d'interface utilisateur.

    DataSet dsEtapes = new DataSet();
    // Un autre DataSet pour stocker les étapes de la recette.

    OleDbConnection conn = new OleDbConnection();
    // Création d'une connexion OleDb pour se connecter à une base de données.

    string Arguments = "";
    // Une chaîne de caractères pour stocker les arguments de recherche de recette.

    int NbSelectionne = 0;
    // Un entier pour stocker le nombre d'éléments sélectionnés.

    string nomRecette = "";
    // Une chaîne de caractères pour stocker le nom de la recette.

    string NumeroRecette;
    // Une chaîne de caractères pour stocker le numéro de la recette.

    string pseudoNom = "";
    // Une chaîne de caractères pour stocker le pseudo.

    bool connecte = false;
    // Un booléen pour indiquer si l'utilisateur est connecté ou non.
```

II. Procédure événementielle permettant de rechercher les recettes répondant aux critères de l'utilisateur

```
private void btnBasePatisserie_Click(object sender, EventArgs e)
{
    // Convertit le sender en objet Button
    System.Windows.Forms.Button t = (System.Windows.Forms.Button)sender;

    // Récupère le numéro de catégorie à partir de la propriété Tag du bouton
    int numCat = Convert.ToInt16(t.Tag);

    // Efface les éléments de la liste des ingrédients
    clbIngredients.Items.Clear();

    // Met à jour le texte du label "lblFamille" avec le texte du bouton
    lblFamille.Text = t.Text;

    // Parcourt chaque ligne dans le tableau "Ingrédients" de l'objet DataSet "ds"
    foreach (DataRow dr in ds.Tables["Ingrédients"].Rows)
    {
        // Vérifie si la catégorie de l'ingrédient correspond au numéro de catégorie sélectionné
        if (dr[2].ToString() == numCat.ToString())
        {
            // Ajoute le nom de l'ingrédient à la liste des ingrédients
            clbIngredients.Items.Add(dr[1].ToString());
        }
    }

    // Trie la liste des ingrédients dans l'ordre alphabétique
    List<string> ingredientsList = new List<string>();
    foreach (var item in clbIngredients.Items)
    {
        ingredientsList.Add(item.ToString());
    }
    ingredientsList.Sort();

    clbIngredients.Items.Clear();

    // Ajoute les ingrédients triés à la liste des ingrédients
    foreach (var ingredient in ingredientsList)
    {
        clbIngredients.Items.Add(ingredient);
    }

    // Vérifie si l'ingrédient est déjà choisi dans la liste IngredientsChoisis
    foreach (KeyValuePair<string, string> str in IngredientsChoisis)
    {
        for (int i = 0; i < clbIngredients.Items.Count; i++)
        {
            if (str.Key.CompareTo(ds.Tables["Ingrédients"].Rows[i][0].ToString()) == 0)
            {
                // Coche l'élément de la liste correspondant à l'ingrédient choisi
                clbIngredients.SetItemChecked(i, true);
                break;
            }
        }
    }
}
```

III. Procédure événementielle grâce à laquelle nous affichons les ingrédients d'une famille sélectionnée :

```
private void btnValiderModif_Click(object sender, EventArgs e)
{
    pnlChoixRecette.Controls.Clear();
    // Sélectionne l'onglet 3 de la fenêtre
    tbcFenetre.SelectTab(3);

    // Crée une liste pour stocker les recettes sélectionnées
    List<string> lstRecSelec = new List<string>();

    // Nettoie le code de recette
    codeRecette.Clear();

    // Vérifie si des ingrédients ont été choisis
    try
    {
        int hauteurRec = 40;

        // Ouvre la connexion à la base de données
        conn.Open();

        string requeteValider = "";
        if (IngredientsChoisis.Count > 0)
        {
            // Parcourt chaque paire clé-valeur dans la liste des ingrédients choisis
            foreach (KeyValuePair<string, string> str in IngredientsChoisis)
            {
                // Construit la requête SQL pour sélectionner les recettes correspondantes
                requeteValider = @"SELECT * FROM Recettes
                            WHERE tempsCuisson <= " + ucTempsChoix1.getTemps +
                                    " AND categPrix <= " + ucBudgetJauge1.getBudget +
                                    " AND codeRecette IN (SELECT codeRecette FROM IngrédientsRecette WHERE codeIngredient = " + int.Parse(str.Key) + ")";


                // Crée une commande pour exécuter la requête SQL
                OleDbCommand commandRechercher = new OleDbCommand(requeteValider, conn);

                // Exécute la requête SQL et récupère le lecteur de données
                OleDbDataReader reader = commandRechercher.ExecuteReader();

                // Parcourt les résultats du lecteur de données
                while (reader.Read())
                {
                    // Ajoute le code de recette à la liste des recettes sélectionnées
                    lstRecSelec.Add(reader.GetValue(0).ToString());

                    // Passe au résultat suivant dans le lecteur de données
                    reader.NextResult();
                }

                // Ferme le lecteur de données
                reader.Close();
            }
        }
        else
        {
            // Construit la requête SQL pour sélectionner les recettes correspondantes
            requeteValider = @"SELECT * FROM Recettes
                            WHERE tempsCuisson <= " + ucTempsChoix1.getTemps +
                            " AND categPrix <= " + ucBudgetJauge1.getBudget;

            // Crée une commande pour exécuter la requête SQL
            OleDbCommand commandRechercher = new OleDbCommand(requeteValider, conn);

            // Exécute la requête SQL et récupère le lecteur de données
            OleDbDataReader reader = commandRechercher.ExecuteReader();

            // Parcourt les résultats du lecteur de données
            while (reader.Read())
            {

                // Ajoute le code de recette à la liste des recettes sélectionnées
                lstRecSelec.Add(reader.GetValue(0).ToString());

                // Passe au résultat suivant dans le lecteur de données
                reader.NextResult();
            }

            // Ferme le lecteur de données
            reader.Close();
        }

        // Parcourt la liste des recettes sélectionnées
        foreach (string code in lstRecSelec)
        {
            // Génère dynamiquement des UserControls pour afficher les recettes
            UcAffichageRecette afficheRecette = new UcAffichageRecette(chcon, getImageRec(code), getNomRec(code), getTempsRec(code), getBudget(code), int.Parse(code));

            // Ajoute des gestionnaires d'événements aux UserControls
            afficheRecette.DoubleClick += new System.EventHandler(this.btnLancerRec_Click);
            afficheRecette.CheckedChanged += tempsCout_CheckedChanged;
            afficheRecette.Top = hauteurRec;
            afficheRecette.Left = 90;

            afficheRecette.Tag = code;

            // Génère dynamiquement des boutons pour consulter les avis sur la recette
            Button btnAvisRec = new Button();


            btnAvisRec.BackColor = Color.White;

            btnAvisRec.Top = hauteurRec +10;

            btnAvisRec.Font = lblIngr1.Font;
            btnAvisRec.Text = "Avis";

            btnAvisRec.Height = 80;

            btnAvisRec.Width = 80;

            btnAvisRec.Tag = code;

            btnAvisRec.Click += new System.EventHandler(this.btnVoirAvis_Click);

            RadioButton ra = new RadioButton();
            ra.Width = 150;
            ra.Tag = code;
            ra.Text = "";
            ra.Left = 520;
            ra.Top = hauteurRec + 50;

            // Ajoute les UserControls et les boutons aux groupes de choix de recettes
            pnlChoixRecette.Controls.Add(afficheRecette);
            pnlChoixRecette.Controls.Add(btnAvisRec);
            pnlChoixRecette.Controls.Add(ra);
            hauteurRec += 120;
        }
    }
    catch (OleDbException)
    {
        MessageBox.Show("Erreur dans la requête SQL");
    }
    catch (InvalidOperationException)
    {
        MessageBox.Show("Erreur d'accès à la base de données");
    }
    finally
    {
        if (conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
    }
}
```

IV. Le code du UserControl permettant de choisir le temps de la recette :

```
public partial class UcTempsChoix : UserControl
{
    0 références
    public UcTempsChoix()
    {
        InitializeComponent();
    }

    int temps = 30;

    1 référence
    private void btnPlus15_Click(object sender, EventArgs e)
    {
        if (temps < 180 && temps + 15 < 180)
        {
            temps += 15;
        }
        else
        {
            temps = 180;
        }
        lblTempsChrono.Text = temps.ToString();
        picBoxChrono.Image = imgListTemps.Images[temps.ToString() + ".png"];
    }

    1 référence
    private void btnPlus5_Click(object sender, EventArgs e)
    {
        if (temps < 180 && temps + 5 < 180)
        {
            temps += 5;
        }
        else
        {
            temps = 180;
        }
        lblTempsChrono.Text = temps.ToString();
        picBoxChrono.Image = imgListTemps.Images[temps.ToString() + ".png"];
    }
```

```
    private void btnMoins15_Click(object sender, EventArgs e)
    {
        if (temps > 0 && temps - 15 > 0)
        {
            temps -= 15;
        }
        else
        {
            temps = 0;
        }
        lblTempsChrono.Text = temps.ToString();
        picBoxChrono.Image = imgListTemps.Images[temps.ToString() + ".png"];
    }

    1 référence
    private void btnMoins5_Click(object sender, EventArgs e)
    {
        if (temps > 0 && temps - 5 > 0)
        {
            temps -= 5;
        }
        else
        {
            temps = 0;
        }
        lblTempsChrono.Text = temps.ToString();
        picBoxChrono.Image = imgListTemps.Images[temps.ToString() + ".png"];
    }

    1 référence
    private void UcTempsChoix_Load(object sender, EventArgs e)
    {
        lblTempsChrono.Text = temps.ToString();
        picBoxChrono.Image = imgListTemps.Images["30.png"];
    }

    0 références
    public int getTemps
    {
        get { return  temps; }
        set { }
    }
}
```

v. Le code permettant de générer le pdf :

```
private void btnPDF_Click(object sender, EventArgs e)
{
    // Initialisation des variables
    string nomFichier = nomRecette + "PDF"; // Nom du fichier PDF à créer
    int hauteur = 10; // Hauteur de la page 1
    int hauteur2 = 40; // Hauteur de la page 2
    int hauteur3 = 40; // Hauteur de la page 3
    PdfDocument DocumentPdf = new PdfDocument(); // Création du document PDF

    // Création et configuration de la première page du PDF
    PdfPage page = DocumentPdf.AddPage();
    XGraphics gfx = XGraphics.FromPdfPage(page);

    // Création et configuration de la deuxième page du PDF
    PdfPage pageNb2 = DocumentPdf.AddPage();
    XGraphics gfx2 = XGraphics.FromPdfPage(pageNb2);

    // Création et configuration de la troisième page du PDF
    PdfPage pageNb3 = DocumentPdf.AddPage();
    XGraphics gfx3 = XGraphics.FromPdfPage(pageNb3);

    // Définition des polices à utiliser dans le document
    XFont PoliceTitre = new XFont("Verdana", 16, XFontStyle.Bold);
    XFont PoliceSousTitre = new XFont("Arial", 14, XFontStyle.Underline);
    XFont PoliceParagraphe = new XFont("Arial", 12);

    // Ajout du titre principal du document
    TextePDF("Récapitulatif " + nomRecette + " :", 0, hauteur, 600, 50, gfx, PoliceTitre);
    hauteur += 50;

    // Ajout de la section "Informations"
    TextePDF("Informations :", 0, hauteur, 600, 50, gfx, PoliceSousTitre);
    hauteur += 40;

    // Parcours des données de recettes pour trouver les informations de la recette en cours
    foreach (DataRow dtr in ds.Tables["Recettes"].Rows)
    {
        if (dtr["codeRecette"].ToString() == btnLancerRec.Tag.ToString())
        {
            // Ajout des informations de temps et de budget de la recette
            TextePDF("Temps : " + dtr[3].ToString(), 100, hauteur, 100, 50, gfx, PoliceParagraphe);
            string TypeBudget = "";
            if (dtr[5].ToString() == "0")
            {
                TypeBudget = "Peu onéreux";
            }
            if (dtr[5].ToString() == "1")
            {
                TypeBudget = "Abordable";
            }
            if (dtr[5].ToString() == "2")
            {
                TypeBudget = "Onéreux";
            }
            TextePDF("Budget : " + TypeBudget, 200, hauteur, 100, 50, gfx, PoliceParagraphe);
        }
    }
    hauteur += 40;

    // Vérification de la sélection des listes d'ingrédients et d'ustensiles
    if (rdbListeNon1.Checked == true || rdbListeNon2.Checked == true)
    {
        int hauteurprec = hauteur;

        // Ajout de la section "Ingrédients"
        TextePDF("Ingrédients :", 0, hauteur, 100, 50, gfx, PoliceSousTitre);
```

```
    // Parcours des données d'ingrédients de la recette
    foreach (DataRow dtr in ds.Tables["IngrédientsRecette"].Rows)
    {
        if (dtr[0].ToString() == btnLancerRec.Tag.ToString())
        {
            // Recherche de l'ingrédient correspondant dans la table des ingrédients
            foreach (DataRow dtr2 in ds.Tables["Ingrédients"].Rows)
            {
                if (dtr[1].ToString().CompareTo(dtr2[0].ToString()) == 0)
                {
                    hauteur += 40;
                    // Ajout de l'ingrédient à la liste
                    TextePDFGauche("- " + dtr2[1], 40, hauteur, 100, 50, gfx, PoliceParagraphe);
                }
            }
        }
    }
    hauteur += 50;

    // Ajout de la section "Ustensiles"
    TextePDF("Ustensiles :", 300, hauteurprec, 100, 50, gfx, PoliceSousTitre);
    List<String> Ustensiles = new List<string>();

    // Parcours des données d'ustensiles nécessaires à la recette
    foreach (DataRow dtr in ds.Tables["BesoinsUstensiles"].Rows)
    {
        if (dtr[0].ToString() == btnLancerRec.Tag.ToString())
        {
            Ustensiles.Add(dtr[1].ToString());
            // Recherche de l'ustensile correspondant dans la table des ustensiles
            foreach (DataRow dtr2 in ds.Tables["Ustensiles"].Rows)
            {
                if (Ustensiles.Contains(dtr2[0].ToString()))
                {
                    hauteurprec += 40;
                    // Ajout de l'ustensile à la liste
                    TextePDFGauche("- " + dtr2[1], 340, hauteurprec, 100, 50, gfx, PoliceParagraphe);
                    Ustensiles.Remove(dtr2[0].ToString());
                }
            }
        }
    }
}

int i = 1; // Compteur d'étapes
bool page1 = false; // Indicateur de passage à la deuxième page

// Connexion à la base de données
try
{
    conn.Open();
    ChargementDsLocal();
}
catch (OleDbException)
{
    MessageBox.Show("Erreur dans la requête SQL");
}
catch (InvalidOperationException)
{
    MessageBox.Show("Erreur d'accès à la base");
}
```

```
finally
{
    if (conn.State == ConnectionState.Open)
    {
        conn.Close();
    }
}

// Parcours des étapes de la recette
foreach (DataRow numRecette in ds.Tables["EtapesRecette"].Rows)
{
    if (numRecette[0].ToString() == btnLancerRec.Tag.ToString() && int.Parse(numRecette[1].ToString()) >= i)
    {
        if (hauteur > 600)
        {
            // Passage à la deuxième page en raison d'un trop grand nombre d'étapes
            TextePDF("Etape " + i + " :", 300, hauteur2, 100, 50, gfx2, PoliceSousTitre);
            hauteur2 += 40;

            // Ajout de l'image de l'étape (ou une image par défaut si l'image n'est pas disponible)
            try
            {
                ImgPDF("../../Properties/" + numRecette["imageEtape"].ToString(), 490, hauteur2, 75, 75, gfx2);
            }
            catch
            {
                ImgPDF("../../Properties/cocotte.jpg", 490, hauteur2, 75, 75, gfx2);
            }

            // Ajout du texte de l'étape avec alignement justifié
            TextePDFAlignement(numRecette["texteEtape"].ToString(), XParagraphAlignment.Justify, 20, hauteur2, 450, 50, gfx2, PoliceParagraphe);
            hauteur2 = hauteur2 + 50;
            page1 = true;
            i++;
        }
        else
        {
            // Ajout de l'étape à la première page
            TextePDF("Etape " + i + " :", 300, hauteur, 100, 50, gfx, PoliceSousTitre);
            hauteur += 40;

            // Ajout de l'image de l'étape (ou une image par défaut si l'image n'est pas disponible)
            try
            {
                ImgPDF("../../Properties/" + numRecette["imageEtape"].ToString(), 490, hauteur, 75, 75, gfx);
            }
            catch
            {
                ImgPDF("../../Properties/cocotte.jpg", 490, hauteur, 75, 75, gfx);
            }

            // Ajout du texte de l'étape avec alignement justifié
            TextePDFAlignement(numRecette["texteEtape"].ToString(), XParagraphAlignment.Justify, 20, hauteur, 450, 50, gfx, PoliceParagraphe);
            hauteur += 50;
            i++;
        }
    }
}

// Vérification de la sélection des listes de courses
if (rdbListeOui1.Checked == true || rdbListeOui2.Checked == true)
{
    // Ajout de la section "Liste de course"
    TextePDF("Liste de course :", 100, hauteur3, 100, 50, gfx3, PoliceTitre);
    hauteur3 += 50;
    int hauteurprec = hauteur3;
```

```
        // Ajout de la section "Ingrédients"
        TextePDF("Ingrédients :", 0, hauteur3, 100, 50, gfx3, PoliceSousTitre);
        hauteur3 += 50;
        List<string> ingredientSelect = new List<string>();

        // Parcours des ingrédients de la recette
        foreach (DataRow dtr in ds.Tables["IngrédientsRecette"].Rows)
        {
            if (dtr[0].ToString() == "1")
            {
                foreach (DataRow dtr2 in ds.Tables["Ingrédients"].Rows)
                {
                    if (dtr[1].ToString().CompareTo(dtr2[0].ToString()) == 0 && !ingredientSelect.Contains(dtr[1].ToString()))
                    {
                        hauteur3 += 40;
                        ingredientSelect.Add(dtr[1].ToString());
                        // Ajout de l'ingrédient à la liste de courses
                        TextePDFGauche("- " + dtr2[1], 40, hauteur3, 100, 50, gfx3, PoliceParagraphe);
                    }
                }
            }
        }
        hauteur3 += 50;

        // Ajout de la section "Ustensiles"
        TextePDF("Ustensiles :", 300, hauteurprec, 100, 50, gfx3, PoliceSousTitre);
        List<String> Ustensiles = new List<string>();

        // Parcours des ustensiles nécessaires à la recette
        foreach (DataRow dtr in ds.Tables["BesoinsUstensiles"].Rows)
        {
            if (dtr[0].ToString() == "1")
            {
                Ustensiles.Add(dtr[1].ToString());
                foreach (DataRow dtr2 in ds.Tables["Ustensiles"].Rows)
                {
                    if (Ustensiles.Contains(dtr2[0].ToString()))
                    {
                        hauteurprec += 50;
                        // Ajout de l'ustensile à la liste de courses
                        TextePDFGauche("- " + dtr2[1], 340, hauteurprec, 100, 50, gfx3, PoliceParagraphe);
                        Ustensiles.Remove(dtr2[0].ToString());
                    }
                }
            }
        }
    }

    DocumentPdf.Save(nomFichier); // Sauvegarde du PDF avec le nom de fichier spécifié
    Process.Start(nomFichier); // Ouverture automatique du PDF
}

// Affiche l'image associée à l'ingrédient sélectionné dans un PictureBox.
// <param name="pictureBox">Le PictureBox où afficher l'image.</param>
3 références
private void AfficherImageCompte(PictureBox pictureBox)
{
    pictureBox.Visible = true;

    try
    {
        // Tente de charger une image avec l'extension .jpg correspondant à l'ingrédient choisi
        pictureBox.ImageLocation = "../../Properties/" + IngredientsChoisis.Last().Value + ".jpg";
        pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox.Load();
    }
    catch
    {
        try
        {
            // Si le chargement précédent a échoué, tente de charger une image avec l'extension .png
            pictureBox.ImageLocation = "../../Properties/" + IngredientsChoisis.Last().Value + ".png";
            pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
            pictureBox.Load();
        }
        catch
        {
            // Si les chargements précédents ont échoué, affiche une image par défaut (lama.png)
            pictureBox.ImageLocation = "../../Properties/lama.png";
            pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
            pictureBox.Load();
        }
    }
}
```

```
private void TextePDF(string contenu, int x, int y, int largeur, int hauteur, XGraphics Xgfx, XFont police)
{
    XRect ZoneTexte = new XRect(x, y, largeur, hauteur);
    // Crée un rectangle définissant la zone de texte avec les coordonnées (x, y), la largeur et la hauteur spécifiées.

    XTextFormatter tfTitre = new XTextFormatter(Xgfx);
    // Crée un formateur de texte en utilisant le contexte graphique spécifié.

    Xgfx.DrawRectangle(XBrushes.White, ZoneTexte);
    // Dessine un rectangle blanc à l'emplacement spécifié pour effacer toute zone de texte précédente.

    Xgfx.DrawString(contenu, police, XBrushes.Black, ZoneTexte, XStringFormats.Center);
    // Dessine le contenu spécifié à l'intérieur du rectangle avec la police et la couleur spécifiées, en centrant le texte.
}

4 références
private void TextePDFGauche(string contenu, int x, int y, int largeur, int hauteur, XGraphics Xgfx, XFont police)
{
    XRect ZoneTexte = new XRect(x, y, largeur, hauteur);
    // Crée un rectangle définissant la zone de texte avec les coordonnées (x, y), la largeur et la hauteur spécifiées.

    XTextFormatter tfTitre = new XTextFormatter(Xgfx);
    // Crée un formateur de texte en utilisant le contexte graphique spécifié.

    Xgfx.DrawRectangle(XBrushes.White, ZoneTexte);
    // Dessine un rectangle blanc à l'emplacement spécifié pour effacer toute zone de texte précédente.

    Xgfx.DrawString(contenu, police, XBrushes.Black, ZoneTexte, XStringFormats.TopLeft);
    // Dessine le contenu spécifié à l'intérieur du rectangle avec la police et la couleur spécifiées, en alignant le texte en haut à gauche.
}

2 références
private void TextePDFAlignement(string contenu, XParagraphAlignment disposition, int x, int y, int largeur, int hauteur, XGraphics Xgfx, XFont police)
{
    XTextFormatter tfPara = new XTextFormatter(Xgfx);
    // Crée un formateur de texte en utilisant le contexte graphique spécifié.

    XRect ZoneTexte = new XRect(x, y, largeur, hauteur);
    // Crée un rectangle définissant la zone de texte avec les coordonnées (x, y), la largeur et la hauteur spécifiées.

    Xgfx.DrawRectangle(XBrushes.White, ZoneTexte);
    // Dessine un rectangle blanc à l'emplacement spécifié pour effacer toute zone de texte précédente.

    tfPara.Alignment = disposition;
    // Définit l'alignement du texte en utilisant la disposition spécifiée.

    tfPara.DrawString(contenu, police, XBrushes.Black, ZoneTexte, XStringFormats.TopLeft);
    // Dessine le contenu spécifié à l'intérieur du rectangle avec la police et la couleur spécifiées, en utilisant l'alignement spécifié.
}
4 références
private void ImgPDF(string Chemin, int x, int y, int largeur, int hauteur, XGraphics Xgfx)
{
    XRect rcImage = new XRect(x, y, largeur, hauteur);
    // Crée un rectangle définissant la position et les dimensions de l'image.

    Xgfx.DrawRectangle(XBrushes.LightGray, rcImage);
    // Dessine un rectangle gris clair à l'emplacement spécifié pour représenter l'espace réservé à l'image.

    Xgfx.DrawImage(XImage.FromFile(Chemin), rcImage);
    // Dessine l'image spécifiée à l'intérieur du rectangle aux coordonnées et dimensions spécifiées.
}
```

VI. Le code permettant la navigation en mode liaison de données entre les étapes d'une recette.

```
private void btnBSAvant_Click(object sender, EventArgs e)
{
    // Définit la source de données du BindingSource comme étant la table "EtapesRecette" du DataSet
    bs.DataSource = dsEtapes.Tables["EtapesRecette"];

    // Convertit le sender en objet Button
    System.Windows.Forms.Button t = (System.Windows.Forms.Button)sender;

    // Récupère le numéro de catégorie à partir de la propriété Tag du bouton et le convertit en entier
    int numCat = int.Parse(t.Tag.ToString());

    // Effectue une action en fonction de la valeur de numCat
    if (numCat == 0) // GAUCHE
    {
        bs.MovePrevious();
    }
    else if (numCat == 1) // DROITE
    {
        bs.MoveNext();
    }
    else if (numCat == 2) // DEBUT
    {
        bs.MoveFirst();
    }
    else // FIN
    {
        bs.MoveLast();
    }

    // Met à jour le texte du Label avec la valeur actuelle du BindingSource
    lblEtapes.Text = "Etape : " + ((DataRowView)bs.Current)["NumEtape"].ToString();

    try
    {
        // Charge l'image de l'étape actuelle dans le PictureBox
        pcbEtapeEtape.ImageLocation = "../../Properties/" + ((DataRowView)bs.Current)["imageEtape"].ToString();
        pcbEtapeEtape.SizeMode = PictureBoxSizeMode.StretchImage;
        pcbEtapeEtape.Load();
    }
    catch
    {
        // Charge une image par défaut si aucune image n'est disponible pour l'étape actuelle
        pcbEtapeEtape.ImageLocation = "../../Properties/cocotte.jpg";
        pcbEtapeEtape.SizeMode = PictureBoxSizeMode.StretchImage;
        pcbEtapeEtape.Load();
    }

    // Affiche le contenu textuel de l'étape actuelle dans le RichTextBox
    rctEtapeContenu.Text = ((DataRowView)bs.Current)["texteEtape"].ToString();
}

/*
 * Fonction: btnLogout_Click
 * Description: Cette fonction est appelée lorsque le bouton de déconnexion est cliqué.
 *              Elle effectue les actions nécessaires pour se déconnecter de l'application.
*/
1 référence
private void btnLogout_Click(object sender, EventArgs e)
{
    // Nettoie les contrôles dans le panneau pntlFavoris
    pntlFavoris.Controls.Clear();
    txtPseudoMenu.Text = String.Empty;
    txtPseudoMenu.Enabled = true;
    // Marque l'utilisateur comme déconnecté en définissant la variable connecte à false
    connecte = false;
}
```