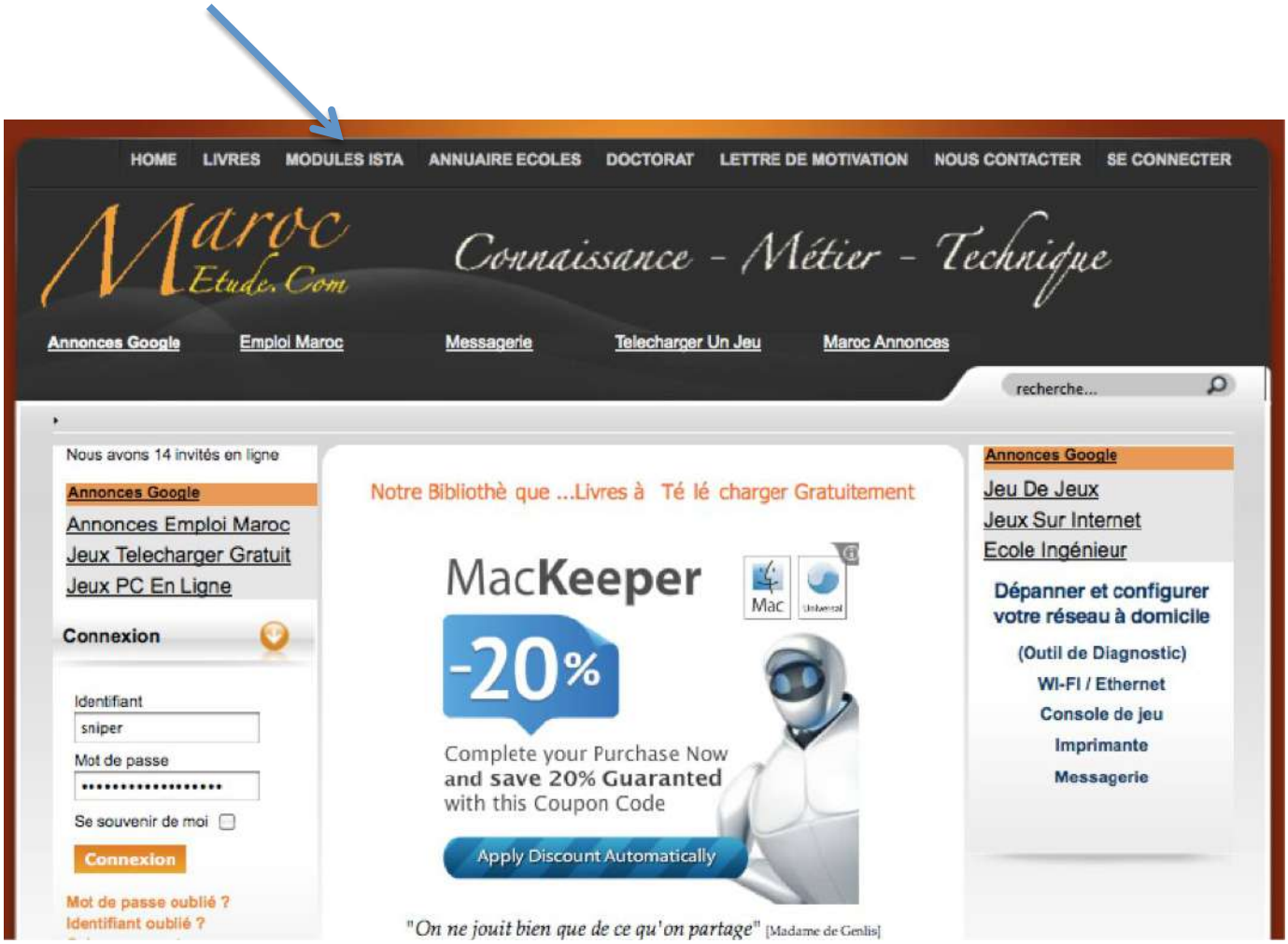
PORTAIL DE LA FORMATION PROFESSIONNELLE AU MAROC

Télécharger tous les modules de toutes les filières de l'OFPPT sur le site dédié à la formation professionnelle au Maroc : **www.marocetude.com**

Pour cela visiter notre site www.marocetude.com et choisissez la rubrique :

MODULES ISTA



PARTIE III / PRATIQUE DE ADO.NET

Pour les exemples de ce cours la base de données **Recettes** est utilisée :

- Description : Cette base de données permet de stocker des recettes de cuisine. Chaque recette se compose d'un ensemble d'ingrédients et chaque ingrédient est fourni par un et un seul fournisseur.
- Structure de la base de données :

- Nom du serveur SQL Server : TEST
- Mode d'authentification : Mixte
- Les comptes utilisateurs suivants ont accès à la base de données Recettes :
  - Les utilisateurs du domaine :

Utilisateur	Mot de passe	Droits accordés
Administrateur	Naoual	tous les droits
Ut1Domaine	1	tous les droits sur la table Fournisseur, aucun droit sur les autres tables

  - Les utilisateurs SQL Server :

Utilisateur	Mot de passe	Droits accordés
Sa	Mot de passe vide	tous les droits
Ut1	1	tous les droits sur la table Fournisseur, aucun droit sur les autres tables

- Le numéro de recette est automatique
- Le nom de recette est unique
- Cette base de données est disponible sur votre serveur de classe pour vous permettre de tester les différents exemples
- Le cours traitera du fournisseur de données .Net Framework pour OleDB et du fournisseur de données .Net Framework optimisé pour SQL Server. L'utilisation du fournisseur de données pour odbc est similaire à celle de OleDB à ceci prés qu'il faut importer l'espace de noms System.Data.odbc au lieu de System.Data.OleDB

ETABLISSEMENT D'UNE CONNEXION

L'objet Connection permet à une application de se connecter et de communiquer avec une source de données.

Instanciation :

Pour utiliser l'objet connection, il faut créer selon le cas une instance de la classe OleDbConnection ou de la classe SqlConnection :

En utilisant le driver Ole Db

```
Imports System.Data.OleDb
Dim Réf_Connexion as New OleDbConnection (déclaration et instanciation)
```

Ou

```
Dim Réf_Connexion  as OleDbConnection  (Déclaration)
Réf_Connexion  = New OleDbConnection (instantiation)
```

Exemple :

```
Dim cn as new OleDbConnection
```

En utilisant le driver optimisé

```
Imports System.Data.SqlClient
Dim Réf_Connexion as New SqlConnection (déclaration et instanciation)
```

Ou

```
Dim Réf_Connexion  as SqlConnection  (Déclaration)
Réf_Connexion  = New SqlConnection (instantiation)
```

Exemple :

```
Dim cn as new SqlConnection
```

Remarques :

- Un autre constructeur est disponible pour la classe oledbConnection :
  New(ConnectionString as String)
- L'objet Connection dispose d'un certain nombre d''événements qui permettent de détecter les résultats obtenus suite à une demande de connexion. Pour utiliser et gérer ces événements, on doit rajouter après Dim le mot clé WithEvents. Une fois l'instruction WithEvents utilisée, la variable de connexion apparaît parmi la liste des objets de la feuille dans la page de code et un ensemble d'événements lui sont associés (Disposed, infoMessage et StateChange).

```
Dim WithEvents cn as new OleDbConnection
```

Quelques propriétés :

ConnectionString : Fournit les informations dont l'objet Connection a besoin pour l'établissement de la connexion entre le client et le serveur

En utilisant le driver Ole Db

Pour un SGBD comme Access (pas de notion de serveur) :

```
Réf_Connexion.ConnectionString="Provider=...;data Source=..."
```

Clause	Rôle
Provider	Nom du fournisseur OLE DB concerné
Data Source	Chemin complet de la base de données (Si la base est sur une autre machine il faut indiquer le nom de la machine dans le chemin \\Nom_Machine\...)

Pour un SGBD comme SQL Server (avec serveur) :

Pour utiliser l'authentification Windows

```
Réf_Connexion.ConnectionString="Provider=...;Server=...; database= ...;
                                Integrated Security = SSPI"
```

Pour utiliser l'authentificaion du SGBD

```
Réf_Connexion.ConnectionString="Provider=...;Server=...; database=...;
                               uid=...;pwd=..."
```

Clause	Rôle
Provider	Le fournisseur OLE DB concerné
Server	Le nom du serveur (Server peut être remplacée par Address, NetworkAdress ou Addr)
Uid	Le nom de l'utilisateur (il est également possible d'utiliser User Id)
pwd	Le mot de passe (il est également possible d'utiliser Password)
DataBase	Nom de la base de données à laquelle on souhaite se connecter. Remarque : En SQL Server il est possible de remplacer DataBase par Initial Catalog

En utilisant le driver optimisé

Pour utiliser les informations d'identification du compte Windows

```
Réf_Connexion.ConnectionString="Server=...; database= ...; Integrated
Security=SSPI"
```

Pour utiliser les information d'identification de SQL Server

```
Réf_Connexion.ConnectionString="Server=...; database=...;uid=...;pwd=..."
```

- **ConnectionTimeout** : Fournit la durée d'attente avant que la tentative de connexion ne soit arrêtée. Cette valeur est par défaut de 15 Secondes

```
Réf_Connexion.ConnectionTimeout=Valeur_En_Secondes
```

- **State** : Retourne l'état actuel de la connexion. Les valeurs possibles sont :

Etat Connexion	Signification
ConnectionState.Connecting	La connexion est en cours (pas encore effectuée)
ConnectionState.Open	La connexion est ouverte
ConnectionState.Broken	La connexion a été coupée
ConnectionState.Executing	La connexion est en train d'exécuter une commande
ConnectionState.Fetching	La connexion est en train d'extraire des données
ConnectionState.Closed	La connexion est fermée

Quelques méthodes :

- **Open** : Permet l'ouverture d'une connection

```
Réf_Connexion.open()
```

- **Close** : Fermeture d'une connexion

```
Réf_Connexion.close()
```

- **BegìnTransaction(Niveau d'isolation)** : Associe une transaction à une connexion en spécifiant un niveau d'isolation.

Le niveau d'isolation indique le comportement de la transaction vis à vis des accès concurrents à la source de données. Plusieurs niveaux d'isolation sont disponibles en Vb.Net :

Niveau	Comportement
Chaos	• Il ne sera pas possible de modifier des données si ces mêmes données ont été modifiées par des transactions mères et que ces données n'ont pas encore été validées
ReadCommitted	• Il ne sera pas possible de lire des données écrites par une transaction concurrente non encore validée • En relisant des données ou en re-exécutant une requête, il est possible d'obtenir des résultats différents de celles obtenues lors de la première lecture ou exécution à cause d'une transaction concurrente qui a été validée entre le moment de la première lecture ou exécution et le moment de la deuxième lecture ou exécution
ReadUncommitted	• Il est possible de lire des données écrites par une transaction concurrente même si celle si n'a pas encore été validée • En relisant des données ou en re-exécutant une requête, il est possible d'obtenir des résultats différents de celles obtenues lors de la première lecture ou exécution à cause d'une transaction concurrente qui a été validée entre le moment de la première lecture ou exécution et le moment de la deuxième lecture ou exécution
RepeatableRead	• Il ne sera pas possible de lire des données écrites par une transaction concurrente non encore validée

	• En relisant des données, une transaction obtient toujours le même résultat même si ces données ont été modifiées par d'autres transactions validées. • En re-exécutant une requête, il est possible d'obtenir des résultats différents de celles obtenues lors de la première exécution à cause d'une transaction concurrente qui a été validée entre le moment de la première exécution et le moment de la deuxième exécution
Serializable	• Il ne sera pas possible de lire des données écrites par une transaction concurrente non encore validée • En relisant des données ou en re-exécutant une requête, une transaction obtient toujours le même résultat même si ces données ont été modifiées par d'autres transactions validées. • Empêche la mise à jour ou l'insertion tant que la transaction n'est pas terminée
Unspecified	

En utilisant le driver Ole Db

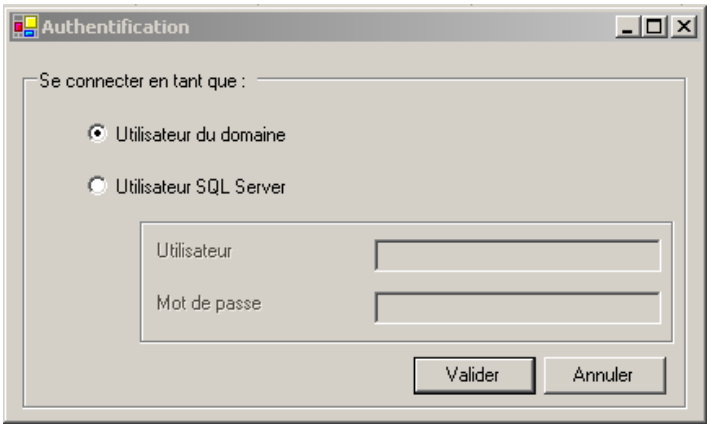
```
Dim Réf_Transaction As OleDbTransaction
Réf_Transaction = Réf_Connexion.BeginTransaction(Niveau isolation)
…
Réf_Transaction.Commit () ou Réf_Transaction.Rollback()
```

En utilisant le driver optimisé

```
Dim Réf_Transaction As SqlTransaction
Réf_Transaction = Réf_Connexion.BeginTransaction(Niveau isolation)
…
Réf_Transaction.Commit() ou Réf_Transaction.Rollback()
```

Exemple : ***Connexion à une base de données SQL Server à l'aide de OLEDB***

Soit le formulaire d'authentification suivant :



Dans cette fenêtre l'utilisateur choisit un mode d'authentification. S'il a choisit l'authentification SQL Server, il devra saisir le nom de l'utilisateur et le mot de passe associé. Après avoir fait son choix l'utilisateur clique sur le bouton Valider. Si la connexion au serveur de base de données s'est correctement déroulée, une feuille MDI FrmMenu va s'afficher.

Remarques :

- Les deux boutons d'options utilisés portent respectivement les noms ChkUserDomaine et ChkUserSQL. Le bouton d'option ChkUserDomaine est coché par défaut
- Les zones utilisateur et mot de passe sont respectivement nommées TxtUserId et TxtPassword. Elles sont contenues avec leurs étiquettes dans un cadre de groupe nommé GpeLogin

Soit le code associé à ce formulaire :

```
Imports System.Data.OleDb
Public Class FrmAuthentification
    Inherits System.Windows.Forms.Form
    ...
    Private Sub ChkUserDomaine_Click(ByVal sender As Object, ByVal e As
                                     System.EventArgs) Handles ChkUserDomaine.Click
        GpeLogin.Enabled = False
    End Sub
    Private Sub ChkUserSQL_Click(ByVal sender As Object, ByVal e As System.EventArgs)
                                 Handles ChkUserSQL.Click
        GpeLogin.Enabled = True
    End Sub
    Private Sub BtnValider_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnValider.Click
        cn = New OleDbConnection
        If ChKUserDomaine.Checked = True Then
            cn.ConnectionString = "Provider=SQLOLEDB;Server=TEST;Database=Recettes;
                                   Integrated security=SSPI"
        Else
            If ChkUserSQL.Checked = True Then
                cn.ConnectionString = "Provider=SQLOLEDB;Server=TEST;Database=Recettes;
                                       Uid=" & TxtUserId.Text & "; Pwd=" &
                                       TxtPassword.Text
            End If
        End If
        Try
            cn.Open()
            Dim f as new FrmMenu
            f.show()
        Catch ex As OleDbException
            MessageBox.Show("Erreur de connexion. L'accès est refusé.")
        End Try
    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
End Class
```

Remarque : Le même exemple peut être utilisé avec le driver optimisé. Il suffit dans ce cas de remplacer :

- System.data.oledb Par System.data.SqlClient
- De remplacer OLEDB par Sql dans le reste du code
- D'enlever la clause Provider de la chaîne de connexion

TRAVAIL A FAIRE :

- **Dans un module, déclarer une variable globale pour la connexion :**

```
Public cn as new OleDbConnection
```

- **Créer le formulaire MDI FrmMenu**
- **Créer le formulaire FrmAuthentification**
- **Ecrire les programmes nécessaires et exécuter l'application**

EXECUTION D'UNE COMMANDE

Pour adresser des commandes au serveur, il faut utiliser l'objet Command

Instantiation :

Pour utiliser l'objet command, il faut créer selon le cas une instance de la classe OleDbCommand ou de la classe SqlCommand :

En utilisant le driver Ole Db

```
Imports System.Data.OleDb

Dim Réf_Command as New OleDbCommand (déclaration et instanciation)
```
Ou
```
Dim Réf_Command  as OleDbCommand  (Déclaration)
Réf_Command  = New OleDbCommand (instantiation)
```

En utilisant le driver optimisé

```
Imports System.Data.SqlClient

Dim Réf_Command as New SqlCommand (déclaration et instanciation)
```
Ou
```
Dim Réf_Command  as SqlCommand  (Déclaration)
Réf_Command  = New SqlCommand (instantiation)
```

Remarque : D'autres constructeurs sont disponibles pour la classe oledbcommand :

- New(CmdText as String)
- New(CmdText as String, Connection as OleDbConnection)
- New(CmdText as String, Connection as OleDbConnection, Transaction as OleDbTransaction)

Quelques propriétés :

Connection : Permet de désigner l'objet Connection auquel sera associée cette commande

```
Réf_Command.Connection=Réf_Connection
```

CommandTimeout : Fournit la durée d'attente de l'exécution d'une commande, avant de générer une erreur

```
Réf_Command.CommandTimeout=Valeur_En_Secondes
```

Transaction : Associe un objet commande à une transaction en cours

```
Réf_Command.Transaction=Réf_Transaction
```

CommandText : Définit l'élément à exécuter au niveau de la source de données. Il peut donc s'agir :

- D'un nom de table
- D'un nom de procédure stockée
- D'une requête SQL

CommandType : Spécifie la nature du contenu de CommandText

Valeur	Signification
CommandType.StoredProcedure	CommandText contient le nom d'une procédure stockée
CommandType.Tabledirect	CommandText contient le nom d'une table
CommandType.text	CommandText contient une requête SQL

Exemples :

```
    Imports System.data.oledb
    Dim Cmd As New OleDbCommand
    Cmd.Connection = cn
Cas d'une table
    Cmd.CommandType = CommandType.TableDirect
    Cmd.CommandText = "Recette"
Cas d'une procédure stockée
    Cmd.CommandType = CommandType.StoredProcedure
    Cmd.CommandText = "SP_1"
Cas d'une requête action
    Cmd.CommandType = CommandType.Text
    Cmd.CommandText = "Delete From Recette"
Cas d'une requête sélection
    Cmd.CommandType = CommandType.Text
```

```
Cmd.CommandText = "Select * from Recette"
```

Quelques méthodes :

Les méthodes utilisées pour une commande dépendent du mode de connexion utilisé :

En mode connecté

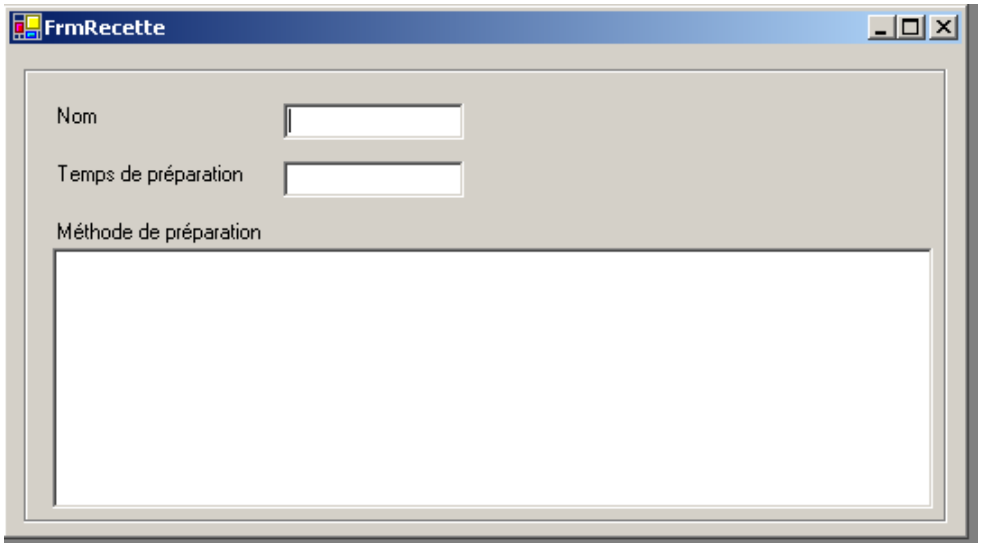
ExecuteNonQuerry : Cette méthode est utilisée pour exécuter une commande de type procédure stockée ou de type Text contenant une instruction SQL de type Insert, Delete, Update ou de description de données (Create Table, Drop Table,...). Après l'exécution, cette méthode retourne le nombre de lignes affectées

Remarques :

Si on tente d'exécuter une commande contenant une instruction select par cette méthode, la méthode retourne la valeur -1.

Exemple : ***Ajout d'une recette***

Soit le formulaire FrmRecette suivant :



Cette fenêtre sera utilisée, via un processus d'héritage, comme base pour la réalisation de plusieurs autres formulaires. Elle comportera certaines vérifications communes à tous ces formulaires :

- Le nom d'une recette ne peut comporter de valeurs numériques : Pour interdire les valeurs numériques dans la zone Nom de recette, l'événement KeyPress sera utilisé
- Lors de la saisie, le nom et la méthode de préparation de la recette doivent obligatoirement être renseignés : Pour vérifier cela, le contrôle ErrorProvider sera exploité au niveau des événement Validating (en cours de validation) et Valideted (une fois la validation terminée).

Remarques :

- Des contrôles de type TextBox seront utilisés pour le nom et le temps de préparation d'une recette (respectivement TxtNom et TxtTemps)
- Un contrôle RichTextBox sera utilisé pour la méthode de préparation puisqu'il peut accepter des textes très longs (RichTxtMethode)
- Le contrôle ErrorProvider utilisé portera le nom ErrorProvider1

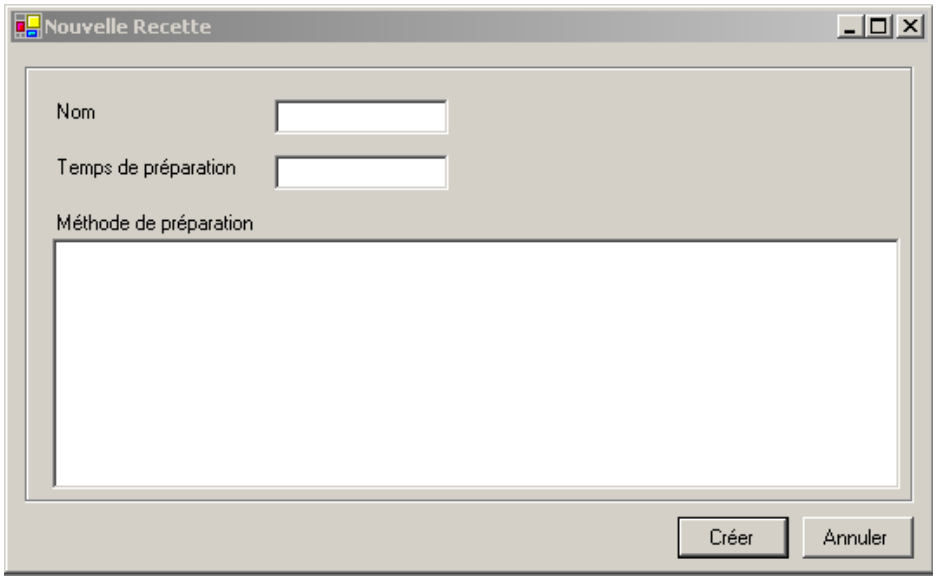
Soit le code associé à ce formulaire :

```
Public Class FrmRecette
    Inherits System.Windows.Forms.Form

    Private Sub TxtNom_KeyPress(ByVal sender As Object, ByVal e As
                                System.Windows.Forms.KeyPressEventArgs) Handles
                                TxtNom.KeyPress
        'Ce code interdit la saisie de nombres
        If Char.IsNumber(e.KeyChar) Then
            e.Handled = True
        End If
    End Sub
    Private Sub TxtNom_Validating(ByVal sender As Object, ByVal e As
                                  System.ComponentModel.CancelEventArgs) Handles
                                  TxtNom.Validating
        If TxtNom.Text = "" Then
            ErrorProvider1.SetError(TxtNom, "Le nom est obligatoire")
            e.Cancel = True
        End If
    End Sub
    Private Sub TxtNom_Validated(ByVal sender As Object, ByVal e As System.EventArgs)
                                  Handles TxtNom.Validated
        ErrorProvider1.SetError(TxtNom, "")
    End Sub
    Private Sub RichTxtMethode_Validating(ByVal sender As Object, ByVal e As
                System.ComponentModel.CancelEventArgs) Handles RichTxtMethode.Validating
        If RichTxtMethode.Text = "" Then
            ErrorProvider1.SetError(RichTxtMethode, "La méthode de préparation est
                                    obligatoire")
            e.Cancel = True
        End If
    End Sub
    Private Sub RichTxtMethode_Validated(ByVal sender As Object, ByVal e As
                                  System.EventArgs) Handles RichTxtMethode.Validated
        ErrorProvider1.SetError(RichTxtMethode, "")
    End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmRecette**
- **Ecrire les programmes nécessaires et tester le formulaire**

Soit le formulaire frmNouvelleRecette suivant :



Ce formulaire hérite du formulaire FrmRecette et contient en plus les boutons Créer (BtnCreer) et Annuler (BtnAnnuler).
En cliquant sur le bouton Créer, le programme enregistre la nouvelle recette dans la base de données.

Soit le code associé à ce formulaire :

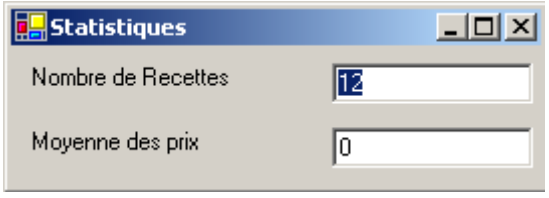
```
Imports System.Data.OleDb
Public Class FrmNouvelleRecette
    Inherits GestionRecette.FrmRecette
    Dim Cmd As New OleDbCommand
    Private Sub FrmNouvelleRecette_Load(ByVal sender As System.Object, ByVal e As
                                        System.EventArgs) Handles MyBase.Load
      'Sans ces deux instructions, chaque fois qu'on clique sur annuler
      'le code contenu dans la procédure évenementielle Validating va se déclencher et
      'donc il faudra obligatoirement remplir les zones de texts pour pouvoir quitter
      'le formulaire

      BtnAnnuler.CausesValidation = False
        Me.CausesValidation = False
    End Sub
    Private Sub BtnCreer_Click(ByVal sender As System.Object, ByVal e As
                               System.EventArgs) Handles BtnCréer.Click
        Cmd.Connection = cn
        Cmd.CommandType = CommandType.Text
        Cmd.CommandText = "insert into Recette (NomRec,MéthodePreparation,
                          TempsPréparation) values ('" & TxtNom.Text & "','" &
                          RichTxtMethode.Text & "', '" & TxtTemps.Text & "')"
        Try
            Cmd.ExecuteNonQuery()
            MessageBox.Show("Recette Créée avec succès", "Gestion recette",
                            MessageBoxButtons.OK, MessageBoxIcon.Information)
        Catch ex As Exception
            MessageBox.Show("Erreur", "Gestion recette", MessageBoxButtons.OK,
                            MessageBoxIcon.Error)
        End Try
    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                               System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmNouvelleRecette**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

ExecuteScalar : Cette méthode sert à exécuter des commandes contenants des requêtes retournant une seule valeur. ExecuteScalar renvoie sous forme d'un objet la première colonne de la première ligne. Pour qu'il soit exploité, l'objet devra être converti dans le type souhaité (effectuer un casting)

Exemple : ***Affichage de statistiques***
Soit le Formulaire FrmStatistiques suivant :



Dans ce formulaire, seront affichés le nombre total de recettes et la moyenne des prix des recettes.

Soit le code associé à ce formulaire :

```
Imports System.Data.OleDb
Public Class FrmStatistiques
    Inherits System.Windows.Forms.Form
...
      Dim cmd As New OleDbCommand
      Private Sub FrmStatistiques_Load(ByVal sender As System.Object, ByVal e As
                                       System.EventArgs) Handles MyBase.Load
            Dim o As Object
            Try
                  cmd.Connection = cn
                  cmd.CommandType = CommandType.Text
                  cmd.CommandText = "Select count(numrec) from recette"
                  o = cmd.ExecuteScalar()
                  If Not o Is DBNull.Value Then
                        TxtNbrRecettes.Text = o.ToString
                  Else
                        TxtNbrRecettes.Text = "0"
                  End If
                  cmd.CommandText = "Select Sum(PUIng*QtéUtilisée) From Ingrédient,
                              Ingrédients_Recette where Ingrédient.Numing=
                              Ingrédients_Recette.NumIng"
                  o = cmd.ExecuteScalar()
                  If Not o Is DBNull.Value Then
                       TxtMoyennePrixRecettes.Text = val(o.ToString) /
                                                     val(TxtNbrRecettes.Text)
                  Else
                        TxtMoyennePrixRecettes.Text = "0"
                  End If
            Catch ex As Exception
                  MessageBox.Show("Erreur. Les données ne peuvent être chargées",
                            "Gestion recette", MessageBoxButtons.OK,
                            MessageBoxIcon.Error)
            End Try
    End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmStatistiques**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

ExecuteReader : Cette méthode est utilisée pour exécuter une commande contenant une instruction Select pouvant retourner 0 ou plusieurs lignes. Le résultat est retourné dans un objet OleDbDataReader ou SqlDataReader selon le type de driver utilisé.

Remarque : Seul l'accès séquentiel est autorisé avec ces objets

Stockage du résultat dans un objet DataReader :

En utilisant le driver Ole Db

```
Dim Réf_DataReader as OleDbDataReader
Réf_DataReader = Réf_Command.ExecuteReader()
```

En utilisant le driver optimisé

```
Dim Réf_DataReader as SqlDataReader
Réf_DataReader = Réf_Command.ExecuteReader()
```

Lecture d'un objet DataReader

La méthode read() permet de lire les lignes d'un DataReader ligne par ligne. Elle retourne un

booléen indiquant s'il existe encore des lignes à lire ou non.

Exemple :

```
Dim l As Boolean
l = rdOleDb.Read()
While (l = True)
      MessageBox.Show("Numéro:" & rdOleDb(0) & "Désignation :" & dOleDb(1))
      l = rdOleDb.Read()
End While
```

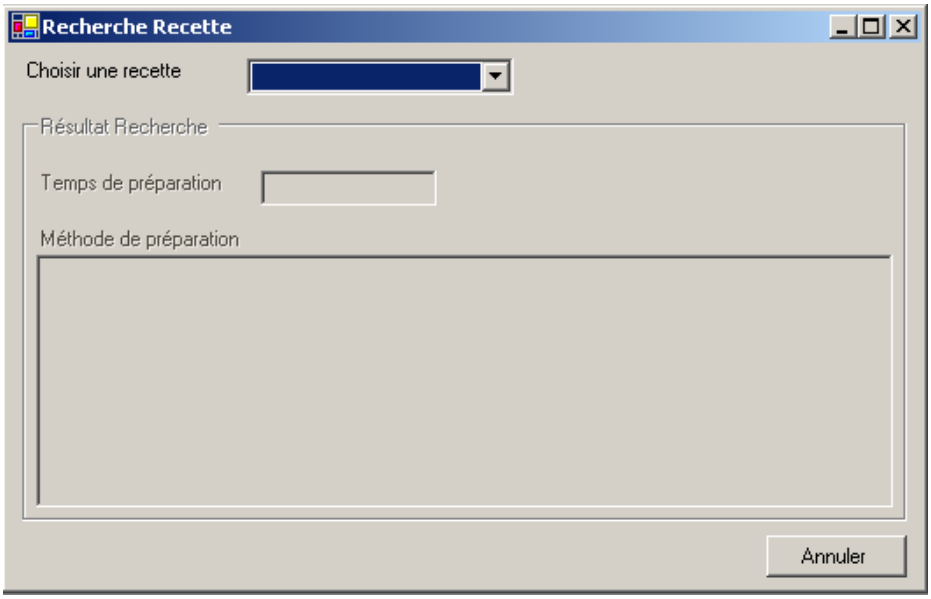
ou

```
While (rdOleDb.Read())
      MessageBox.Show("Numéro:" & rdOleDb(0) & "Désignation :" & dOleDb(1))
End While
```

Remarque :

- Il n'est possible d'avoir qu'un seul DataReader ouvert par connexion.
- Si un DataReader n'est plus utilisé, il faut le fermer

Exemple : ***Consultation d'une recette par Nom***
Soit le formulaire FrmRechercherRecette :



Dans ce formulaire l'utilisateur va choisir un nom de recette dans la zone de liste modifiable (CmbListeNomsRecettes). Au choix d'une recette, les informations concernant cette recette vont apparaître.
Les Zones temps de préparation (TxtTemps) et méthode de préparation (RichTxtMethode) sont contenus dans un cadre nommé GroupeInfos.

Soit le code associé à ce formulaire :

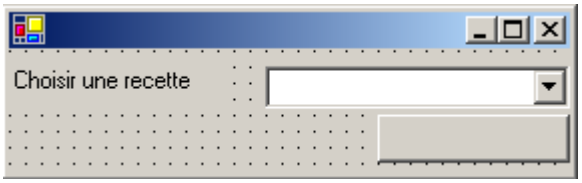
```
Imports System.Data.OleDb
Public Class FrmRechercheRecette
    Inherits System.Windows.Forms.Form
    Dim cmd As New OleDbCommand
    Dim DataR As OleDbDataReader
...
    Private Sub FrmRechercheRecette_Load(ByVal sender As System.Object, ByVal e As
                                         System.EventArgs) Handles MyBase.Load
        GroupeInfos.Enabled = False
        Try
            cmd.Connection = cn
            cmd.CommandType = CommandType.Text
            cmd.CommandText = "Select NomRec from Recette"
```

```
            DataR = cmd.ExecuteReader
            Dim l As Boolean
            l = DataR.Read()
            While (l = True)
                CmbListeNomsRecettes.Items.Add(DataR(0))
                l = DataR.Read()
            End While
            DataR.Close()
        Catch ex As Exception
            CmbListeNomsRecettes.Enabled = False
            MessageBox.Show("Erreur. Les données ne peuvent être chargées", "Gestion
                            recette", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
    Private Sub CmbListeNomsRecettes_SelectedIndexChanged(ByVal sender As
                                    System.Object, ByVal e As System.EventArgs) Handles
                                    CmbListeNomsRecettes.SelectedIndexChanged
        GroupeInfos.Enabled = False
        TxtTemps.Text = ""
        RichTxtMethode.Text = ""
        If CmbListeNomsRecettes.Text <> "" Then
            cmd.Connection = cn
            cmd.CommandType = CommandType.Text
            cmd.CommandText = "Select TempsPréparation,MéthodePreparation from Recette
                               Where NomRec= '" & CmbListeNomsRecettes.Text & "' "
            DataR = cmd.ExecuteReader
            'La boucle n'a pas été utilisée à ce niveau car la requête peut retourner
            ' au maximum un enregistrement
            Dim l As Boolean
            l = DataR.Read()
            If l = True Then
                GroupeInfos.Enabled = True
                TxtTemps.Text = DataR(0)
                RichTxtMethode.Text = DataR(1)
            End If
            DataR.Close()
        End If
    End Sub
    Private Sub CmbListeNomsRecettes_TextChanged(ByVal sender As Object, ByVal e As
                          System.EventArgs) Handles CmbListeNomsRecettes.TextChanged
        GroupeInfos.Enabled = False
        TxtTemps.Text = ""
        RichTxtMethode.Text = ""
        If CmbListeNomsRecettes.Text <> "" Then
            cmd.Connection = cn
            cmd.CommandType = CommandType.Text
            cmd.CommandText = "Select TempsPréparation,MéthodePreparation from Recette
                               Where NomRec= '" & CmbListeNomsRecettes.Text & "' "
            DataR = cmd.ExecuteReader
            'La boucle n'a pas été utilisée à ce niveau car la requête peut retourner
            ' au maximum un enregistrement
            Dim l As Boolean
            l = DataR.Read()
            If l = True Then
                GroupeInfos.Enabled = True
                TxtTemps.Text = DataR(0)
                RichTxtMethode.Text = DataR(1)
            End If
            DataR.Close()
        End If
    End SubPrivate Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmRechercherRecette**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

Exemple : ***Modification et Suppression d'une recette***
Soit le formulaire FrmChoisirRecette suivant :

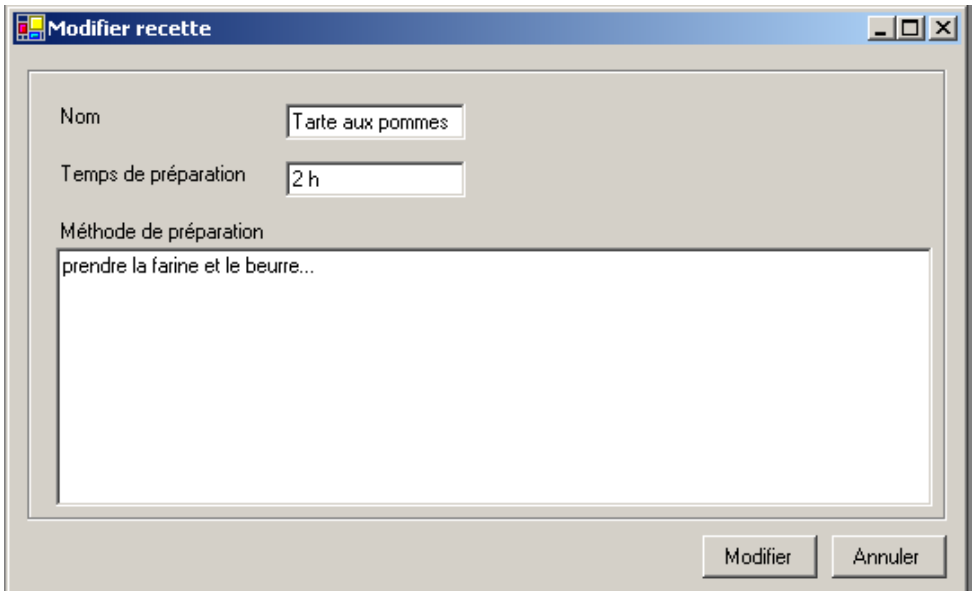


Ce formulaire sera exploité pour rechercher la recette à supprimer ou à modifier. **Une variable globale prendra (à partir du formulaire FrmMenu) la valeur "Supprimer" si l'utilisateur a demandé la suppression d'une recette et "Modifier" si l'utilisateur a demandé la modification d'une recette**.

En cas de suppression le formulaire FrmChoisirRecette aura l'aspect suivant :

En cas de modification le formulaire FrmChoisirRecette aura l'aspect suivant :

Soit le formulaire FrmModifierRecette suivant :



Le formulaire FrmModifierRecette hérite du formulaire FrmRecette. Au choix d'un nom de recette dans le formulaire précédent, les informations concernant cette recette apparaissent dans ce formulaire, l'utilisateur pourra apporter les modifications souhaitées puis appuyer sur le bouton Modifier.

Soit le code associé au formulaire FrmChoisirRecette :

```
Imports System.Data.OleDb
Public Class FrmChoisirRecette
    Inherits System.Windows.Forms.Form
    Dim cmd As New OleDbCommand
    Dim Datar As OleDbDataReader
...
    Private Sub FrmChoisirRecette_Load(ByVal sender As System.Object, ByVal e As
                                System.EventArgs) Handles MyBase.Load
        If Operation = "Supprimer" Then
            Me.Text = "Suppression d'une recette"
            BtnOk.Text = "Supprimer"
        End If
        If Operation = "Modifier" Then
            Me.Text = "Modification d'une recette"
            BtnOk.Text = "Afficher"
        End If
            cmd.Connection = cn
            cmd.CommandType = CommandType.Text
            cmd.CommandText = "Select NomRec from Recette"
        Try
            Datar = cmd.ExecuteReader
            While (Datar.Read())
                CmbListeNomsRecettes.Items.Add(Datar(0))
            End While
            Datar.Close()
        Catch ex As Exception
            MessageBox.Show("Erreur. Les données ne peuvent être chargées", "Gestion
                            recette", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
    Private Sub BtnOk_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                            Handles BtnOk.Click
        If CmbListeNomsRecettes.Text <> "" Then
                If Operation = "Supprimer" Then
                    cmd.Connection = cn
                    cmd.CommandType = CommandType.Text
                    cmd.CommandText = "delete from recette where nomrec= '" &
                                      CmbListeNomsRecettes.Text & "' "
                   Try
                       cmd.ExecuteNonQuery()
                      MessageBox.Show("Suppression effectuée avec succès")

                   Catch ex As OleDbException
                       MessageBox.Show("Echec Suppression")
                   End Try
                End If
                If Operation = "Modifier" Then
                    Dim f As New FrmModifierRecette
                    cmd.Connection = cn
                    cmd.CommandType = CommandType.Text
                    cmd.CommandText = "Select numrec,nomRec, TempsPréparation,
                                       MéthodePreparation from Recette Where NomRec= '"
                                       & CmbListeNomsRecettes.Text & "' "
                  Try
                       Datar = cmd.ExecuteReader
                       If Datar.Read() Then
                             NRec = Datar(0)
                             f.TxtNom.Text = Datar(1)
                             f.TxtTemps.Text = Datar(2)
                             f.RichTxtMethode.Text = Datar(3)
                       End If
                       Datar.Close()
                        f.Show()
                   Catch ex As OleDbException
```

```
                        MessageBox.Show("Recette inaccessible")
                    End Try
                End If
                Operation = ""
        End If
    End Sub
End Class
```

Soit le code associé au formulaire FrmModifierRecette :

```
Imports System.Data.OleDb
Public Class FrmModifierRecette
    Inherits GestionRecette.FrmRecette
    Dim cmd As New OleDbCommand
...
    Private Sub BtnModifier_Click(ByVal sender As System.Object, ByVal e As
                                  System.EventArgs) Handles BtnModifier.Click

        cmd.Connection = cn
        cmd.CommandType = CommandType.Text
        cmd.CommandText = "Update Recette set NomRec='" & TxtNom.Text & "',
                          Méthodepreparation='" & RichTxtMethode.Text & "',
                          Tempspréparation='" & TxtTemps.Text & "' where numrec=" &
                          NRec & ""

        Try
            cmd.ExecuteNonQuery()
            MessageBox.Show("Recette modifiée avec succès", "Gestion recette",
                          MessageBoxButtons.OK, MessageBoxIcon.Information)

        Catch ex As Exception
            'MessageBox.Show(ex.Message)
            MessageBox.Show("Erreur", "Gestion recette", MessageBoxButtons.OK,
                          MessageBoxIcon.Error)
        End Try
    End Sub

    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub

End Class
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmRechercherRecette**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

L'OBJET PARAMETER

Servent à faire passer ou récupérer des paramètres d'une requête SQL ou d'une procédure stockée :

Instantiation :

Pour utiliser l'objet Parameter, il faut créer selon le cas une instance de la classe OleDbParameter ou de la classe SqlParameter :

En utilisant le driver Ole Db

```
Imports System.Data.OleDb
Dim Réf_Param as New OleDbParameter (déclaration et instanciation)
```

Ou

```
Dim Réf_Param  as OleDbParameter  (Déclaration)
Réf_Param  = New OleDbParameter (instantiation)
```

Pour spécifier des paramètres dans une requête SQL, on utilise des points d'interrogation dans oledb (Select * from Recette where NomRec=?)

En utilisant le driver optimisé

```
Imports System.Data.SqlClient
Dim Réf_Param as New SqlParameter (déclaration et instanciation)
```

Ou

```
Dim Réf_Param  as SqlParameter  (Déclaration)
Réf_Param  = New SqlParameter (instantiation)
```

Pour spécifier des paramètres dans une requête SQL, on leur affecte des noms qui commencent par le caractère @ (Select * from Recette where NomRec=@NomRec)

Quelques propriétés :

- **ParameterName** : Permet de définir et d'obtenir le nom du paramètre.
  ```
  Réf_Param.ParameterName=Nom_Paramètre
  ```
- **Value** : Permet de définir et d'obtenir la valeur du paramètre
  ```
  Réf_Param.Value=Valeur
  ```
- **Direction** : Définit le type du paramètre : d'entrée ou de sortie
  ```
  Réf_Param.Direction=Direction_Paramètre
  ```
- **OleDbType** : Définit le type de données du paramètre
  ```
  Réf_Param.OleDbType =Type_Données_Paramètre
  ```

Association d'un paramètre à une commande

Pour faire passer un paramètre à un objet Command :

```
Réf_Command.Parameters.add(Réf_Paramètre)
```

Exemple : ***Suppression d'une recette en utilisant une procédure stockée***

- Soit la procédure stockée suivante :
  Create Procedure SP_SupprimerRecette @a varchar(50) AS
  Delete from Recette where NomRec=@a
- Soit la procédure stockée suivante :
  Create Procedure SP_ListeRecettes AS
  Select NomRec from Recette

Soit le formulaire FrmSupprimerRecetteAvecProcedure :

Soit le code associé au formulaire FrmSupprimerRecetteAvecProcedure :

```
Imports System.Data.OleDb
Public Class FrmSupprimerRecetteAvecProcedure
    Inherits System.Windows.Forms.Form
    Dim cmd As New OleDbCommand
    Dim Datar As OleDbDataReader
...
    Private Sub FrmSupprimerRecetteAvecProcedure_Load(ByVal sender As System.Object,
                                ByVal e As System.EventArgs) Handles MyBase.Load

        chargerListe(cn, CmbListeNomsRecettes, "SP_ListeRecettes")
```

```
    End Sub
    Private Sub BtnSupprimer_Click(ByVal sender As System.Object, ByVal e As
                                   System.EventArgs) Handles BtnSupprimer.Click
        If CmbListeNomsRecettes.Text <> "" Then
            If MessageBox.Show("Etes-vous sûr de vouloir supprimer cette recette ?",
                               "Suppression", MessageBoxButtons.YesNo,
                               MessageBoxIcon.Question) = DialogResult.Yes Then
                Dim Cmd As New OleDbCommand
                Dim P As New OleDbParameter
                Cmd.Connection = cn
                Cmd.CommandType = CommandType.StoredProcedure
                Cmd.CommandText = "SP_SupprimerRecette"
                P.Value = CmbListeNomsRecettes.Text
                Cmd.Parameters.Add(P)
                Try
                    Cmd.ExecuteNonQuery()
                    CmbListeNomsRecettes.Text = ""
                    chargerListe(cn, CmbListeNomsRecettes, "SP_ListeRecettes")
                    MessageBox.Show("Suppression effectuée avec succès")
                Catch ex As OleDbException
                    MessageBox.Show("Echec Suppression")
                End Try
            End If
        End If
    End Sub
End Class
```

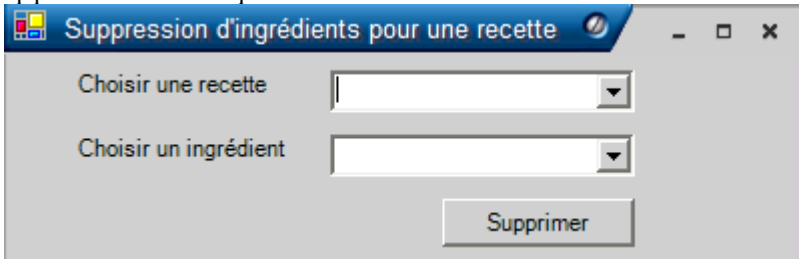
Remarque : chargerListe(ByVal c As OleDbConnection, ByVal cmb As ComboBox, ByVal NomProcedure As String) est une procédure qui devra être créée dans un module pour être accessible à plusieurs feuilles :

```
Public Sub chargerListe(ByVal c As OleDbConnection, ByVal cmb As ComboBox, ByVal
                        NomProcedure As String)
        Dim datar As OleDbDataReader
        cmb.Items.Clear()
        Dim cmd As New OleDbCommand
        cmd.Connection = c
        cmd.CommandType = CommandType.StoredProcedure
        cmd.CommandText = NomProcedure
        datar = cmd.ExecuteReader
        While (datar.Read())
            cmb.Items.Add(datar(0))
        End While
        datar.Close()
    End Sub
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmSupprimerRecetteAvecProcedure**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

Exemple : ***Suppression d'une recette en utilisant une requête paramétrée***
Soit le formulaire FrmSupprimerAvecRequeteParametree :



Soit le code associé au formulaire FrmSupprimerAvecRequeteParametree :

```
Imports System.Data.OleDb
Public Class FrmSupprimerIngredientRecetteAvecRequeteParametree
    Inherits System.Windows.Forms.Form
...
    Private Sub FrmSupprimerIngredientRecetteAvecRequeteParametree_Load(ByVal sender As
                    System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        chargerListe(cn, CmbListeNomsRecettes, "SP_ListeRecettes")
    End Sub
    Private Sub CmbListeNomsRecettes_SelectedIndexChanged(ByVal sender As
                                System.Object, ByVal e As System.EventArgs) Handles
                                cmbListeNomsRecettes.SelectedIndexChanged
        chargerCmbListeIngredients()
    End Sub
    Private Sub CmbListeNomsRecettes_TextChanged(ByVal sender As Object, ByVal e As
                    System.EventArgs) Handles CmbListeNomsRecettes.TextChanged
        chargerCmbListeIngredients()
    End Sub
    Private Sub chargerCmbListeIngredients()
        CmbListeNomsIngredientsRecette.Items.Clear()
        Dim Cmd As New OleDbCommand
        Dim P As New OleDbParameter
        Dim datar As OleDbDataReader
        Cmd.Connection = cn
        Cmd.CommandType = CommandType.Text
        Cmd.CommandText = "select NomIng from Ingrédient, Ingrédients_recette, Recette
                           where Ingrédient.numIng=Ingrédients_recette.numing and
                           Ingrédients_recette.NumRec=Recette.NumRec and NomRec=?"
        P.Value = CmbListeNomsRecettes.Text
        Cmd.Parameters.Add(P)
        Try
            datar = Cmd.ExecuteReader
            While (datar.Read())
                CmbListeNomsIngredientsRecette.Items.Add(datar(0))
            End While
            datar.Close()
        Catch ex As OleDbException
            MessageBox.Show("Erreur chargement")
            CmbListeNomsRecettes.Items.Clear()
            CmbListeNomsIngredientsRecette.Items.Clear()
        End Try
    End Sub
    Private Sub BtnSupprimer_Click(ByVal sender As System.Object, ByVal e As
                                System.EventArgs) Handles BtnSupprimer.Click
        If CmbListeNomsRecettes.Text = "" Or CmbListeNomsIngredientsRecette.Text = ""
              Then
            MessageBox.Show("La recette et l'ingrédient doivent être sélectionnés")
            Exit Sub
        Else
            Dim Cmd As New OleDbCommand
            Dim P1 As New OleDbParameter, P2 As New OleDbParameter
            Cmd.Connection = cn
            Cmd.CommandType = CommandType.Text
            Cmd.CommandText = "delete from Ingrédients_recette where NumRec=(select
                              numRec from Recette where NomRec= ?) and NumIng=(select
                              NumIng from ingrédient where NomIng=?)"
            P1.Value = CmbListeNomsRecettes.Text
            P2.Value = CmbListeNomsIngredientsRecette.Text
            Cmd.Parameters.Add(P1)
            Cmd.Parameters.Add(P2)
            Try
                Cmd.ExecuteNonQuery()
                CmbListeNomsIngredientsRecette.Text = ""
                chargerCmbListeIngredients()
                MessageBox.Show("Suppression effectuée avec succès")
```

```
            Catch ex As OleDbException
                MessageBox.Show("Echec Suppression")
            End Try
        End If
    End Sub
End Class
```

Remarque : D'autres constructeurs sont disponibles pour la classe oledbParameter :

- New(Name as String, value as Object)
- New(Name as String, DataType as Type)
- New(Name as String, DataType as Type, size as Integer)
- …

Remarque : Les mêmes exemples peuvent être utilisés avec le driver optimisé. Il suffit dans ce cas de remplacer :

- System.data.oledb Par System.data.SqlClient
- De remplacer OleDB par Sql dans le reste du code
- De remplacer, dans les requêtes paramérées, les points d'interrogation par des noms de paramétres :

  Une requête formulée ainsi avec le fournisseur OleDB :

  ```
  "delete from Ingrédients_recette where NumRec=(select numRec from
  Recette where NomRec= ?) and NumIng=(select NumIng from ingrédient
  where NomIng=?)"
  ```

  Deviendra avec le fournisseur optimisé:

  ```
  "delete from Ingrédients_recette where NumRec=(select numRec from
  Recette where NomRec= @NomRec) and NumIng=(select NumIng from
  ingrédient where NomIng=@NomIng)"
  ```

- D'indiquer obligatoirement le nom du paramètre avant de l'ajouter à l'objet Command :

```
Dim P1 As New OleDbParameter, P2 As New OleDbParameter
...
P1.ParameterName = "@NomRec"
P2.ParameterName = "@NomIng"
P1.Value = CmbListeNomsRecettes.Text
P2.Value = CmbListeNomsIngredientsRecette.Text
Cmd.Parameters.Add(P)
```

TRAVAIL A FAIRE :

- **Créer le formulaire FrmSupprimerAvecRequeteParametree**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**
- **Ecrire les programmes nécessaires et tester le formulaire**

REALISATION D'ETATS DE SORTIE

Dot Net permet la réalisation d'états de sortie en utilisant Crystal Reports (outil de création d'état de sortie fourni avec Visual Studio.net).

Création d'un état de sortie

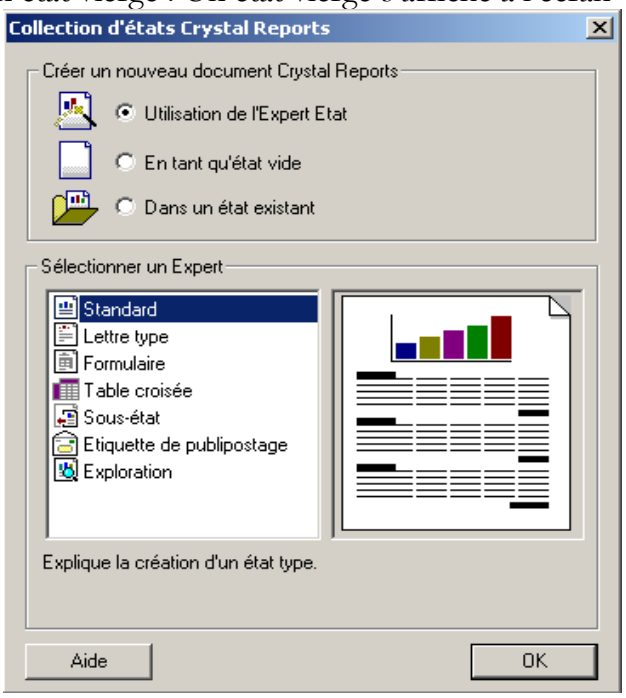
La première étape consiste à insérer au niveau du projet un état Crystal reports : Menu Projet / ajouter un nouvel élément. Dans la fenêtre qui apparaît choisir le modèle Etat Crystal Reports, introduire le nom de l'état et valider.

La fenêtre Collection d'états Crystal Reports de la page suivante apparaît. Elle propose la création d'un nouvel état selon trois cas :

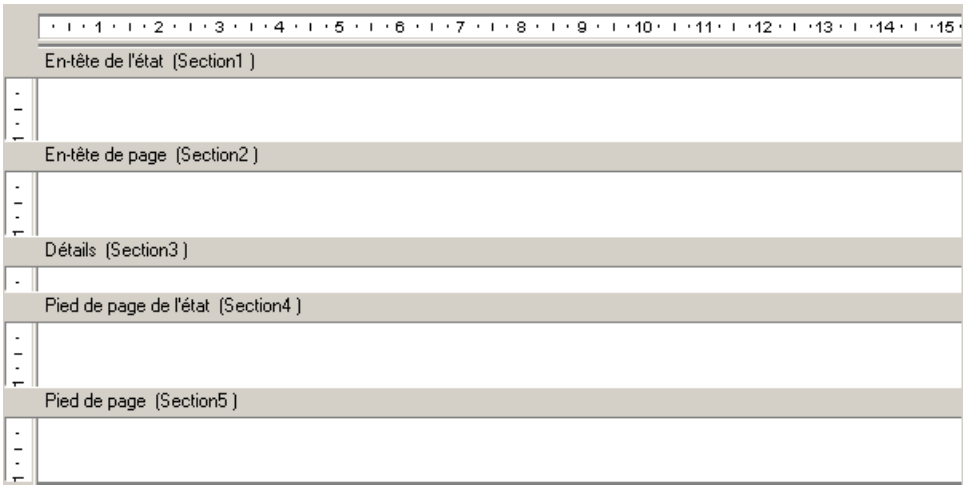
- En utilisant l'expert Etat : L'utilisateur devra choisir un modèle (Lettre, Standard…). En fonction de ce modèle, Crystal Reports nous guidera tout au long de la création et du

paramétrage de l'état

- En utilisant un état existant : Crystal Reports demande le nom de l'état qui va servir comme base de travail. Un nouvel état apparait à l'écran, il affiche le contenu de l'état existant. L'utilisateur peut alors compléter l'état.
- En travaillant sur un état vierge : Un état vierge s'affiche à l'écran



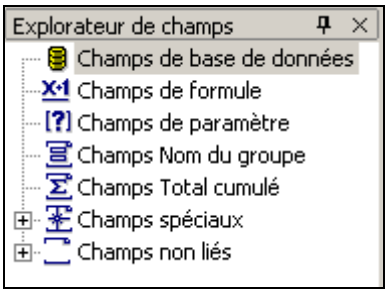
Dans ce cours, on manipulera des états vierges et on vous laissera le soin de tester l'expert Crystal Reports. La fenêtre état vierge a l'apparence suivante :



Elle est découpée en plusieurs sections qu'il est possible d'afficher ou de masquer selon le besoin :

- Entête de l'état : Contient les informations qui s'afficheront une seule fois au début du document même si le document contient plusieurs pages.
- Entête de page : Contient les informations qui s'afficheront au début de chaque page du document.
- Entête de groupe : Contient les informations qui s'afficheront au début de chaque groupe.
- Détails : Contient les lignes de données
- Pied de groupe : Contient les informations qui s'afficheront à la fin de chaque groupe.
- Pied de page : Contient les informations qui s'afficheront à la fin de chaque page du document.
- Pied de l'état : Contient les informations qui s'afficheront une seule fois en fin du document même si le document contient plusieurs pages.

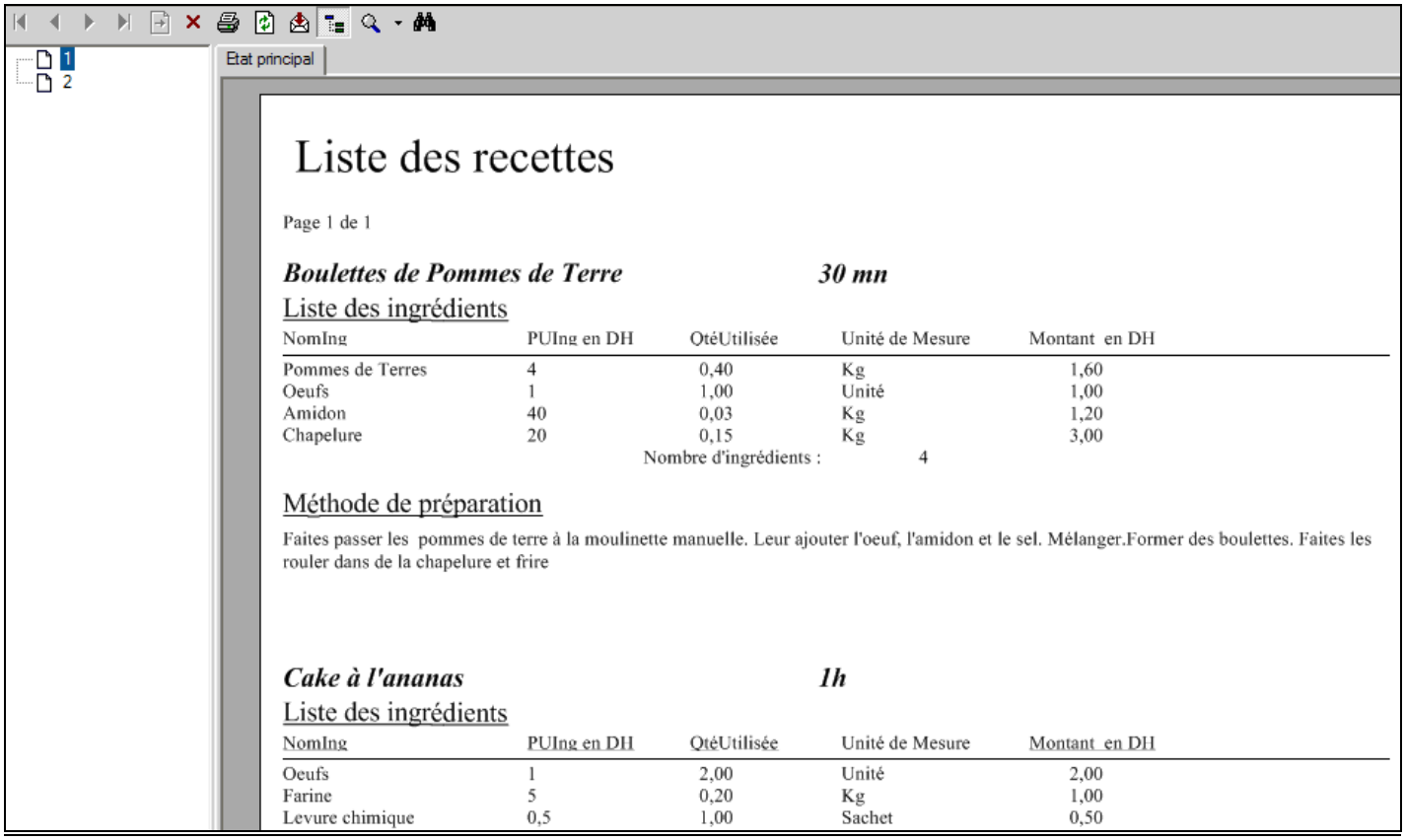
Une boîte à outils Explorateur de champs permettant le paramétrage des états de sortie apparaît :



Paramétrage de la source de données de l'état de sortie

Notre objectif sera de réaliser l'état de sortie suivant représentant la liste des recettes avec pour chaque recette :

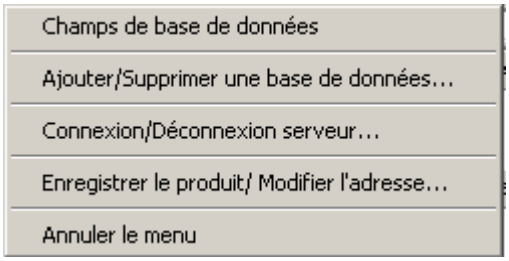
- La liste des ingrédients
- Le nombre d'ingrédients
- La méthode et le temps de préparation



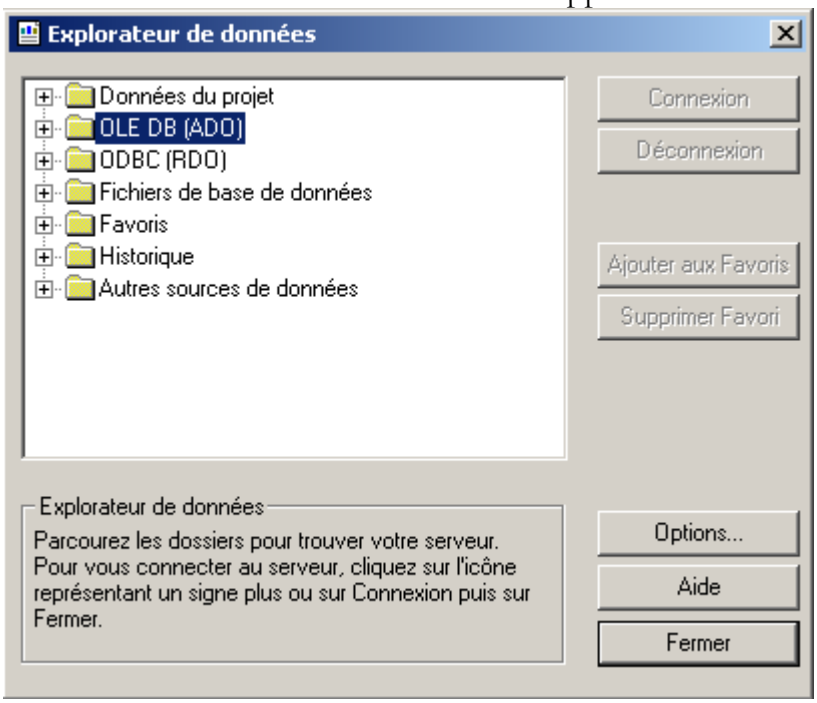
Le paramétrage de tout état de sortie passe, en fonction du besoin, par un certain nombre d'étapes :

Pour paramétrer la connexion avec le serveur de base de données :

Dans la boîte à outils "Explorateur de champs", cliquer avec le bouton droit de la souris sur Champs de base de données :



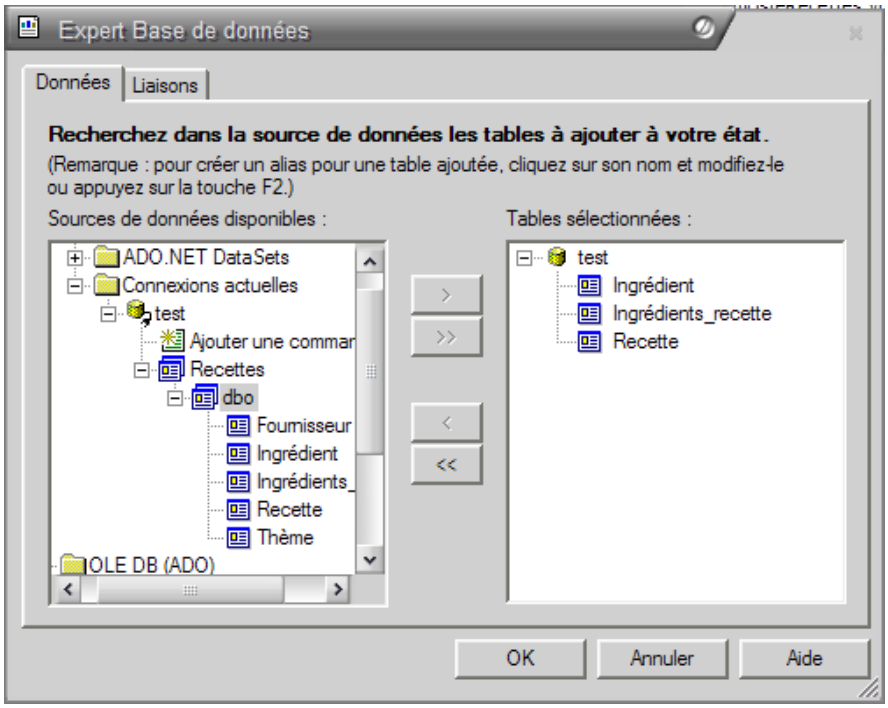
Choisir Connexion / Déconnexion Serveur. La fenêtre suivante apparaît :



En cliquant sur le dossier OLEDB (ADO), une fenêtre de paramétrage de la connexion apparaît : Il faut renseigner les informations de connexion : Pilote (Microsoft OLEDB Provider for SQL Server dans le cas de SQL Server), Nom du serveur, Informations de sécurité, Nom de la base de données. Si les informations communiquées sont correctes, une connexion est établie avec le serveur.

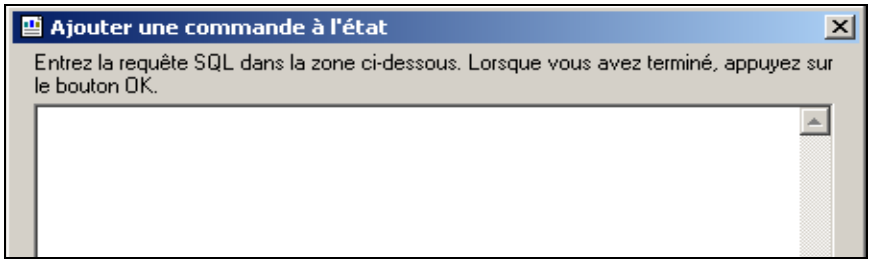
Pour déterminer les champs de l'état de sortie

Dans la boîte à outils "Explorateur de champs", cliquer avec le bouton droit de la souris sur "Champs de base de données" et choisir Ajouter / Supprimer une base de données. La fenêtre Expert Base de Données apparaît. Accéder à la connexion créée. A ce niveau, il est possible de déterminer les champs de l'état de sortie soit en séléctionnant les tables à utiliser graphiquement, soit en utilisant une requête SQL :

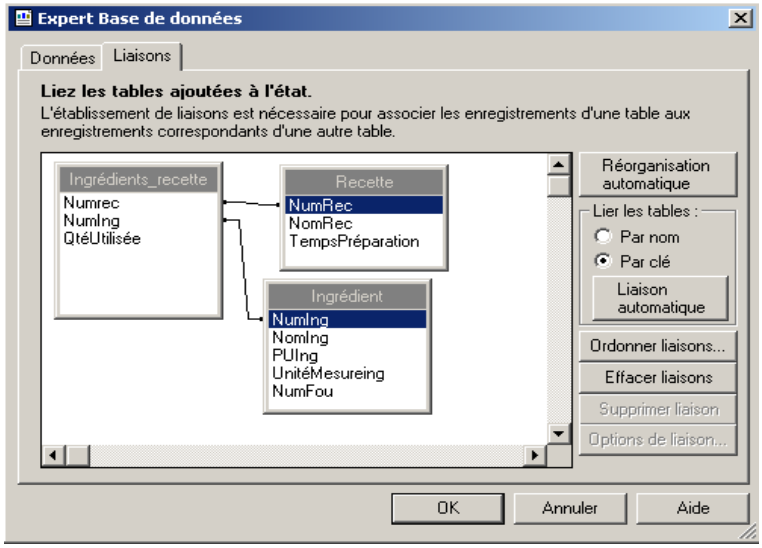


Pour séléctionner les tables à utiliser graphiquement, il suffit de fixer en fonction des champs à utiliser, les tables dont l'utilisateur a besoin. Dans notre exemple nous souhaitons créer un état de sortie affichant la liste des recettes avec pour chaque recette le nom, le temps de préparation, la méthode de préparation, la liste des ingrédients et le Prix de reviens. On aura donc besoin des tables Recette, Ingrédients_Recette et Ingrédient. Ces tables doivent apparaître dans la liste des tables sélectionnées.

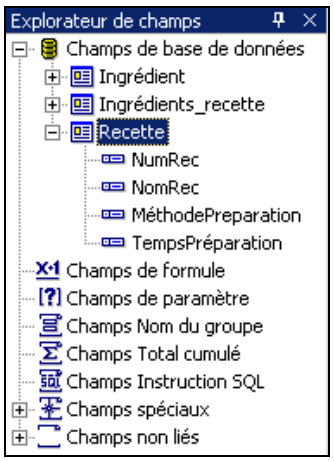
Pour utiliser une requête SQL , il faut choisir l'option Ajouer une commande, un élémeent Commande est alors créé sur le volet droit de la fenêtre. En double-cliquant sur cet élément, la fenêtre ci-dessous apparaît :



Une fois les champs sélectionnés que cela soit avec la première ou la deuxième méthode, l'uilisateur valide en cliquant sur OK, l'onglet liaison apparaît alors, un aperçu des jointures reliant les différentes tables apparaît. Il est possible de supprimer des jointures ou d'ajouter d'autres jointures dans cette fenêtre :

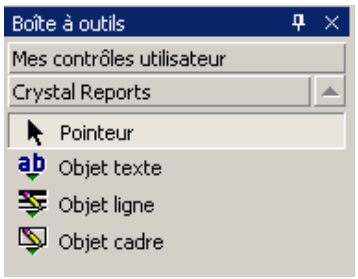


La boîte à outils Explorateur de champs change. La liste des champs de base de données apparaît :



Pour insérer un champ dans l'état, il faut cliquer dessus dans l'explorateur de champs et cliquer-glisser sur

l'état à l'endroit où on souhaite l'insérer dans l'état (Il est important de bien choisir la section où on souhaite déposer le champ).
Pour insérer des étiquettes au niveau de l'état, il faut utiliser l'objet texte de la boîte à outils Crystal Reports :



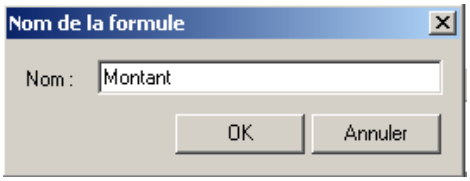
Cette barre d'outils donne également la possibilité de créer des lignes et des cadres sur l'état.

Pour effectuer des regroupements
Si au niveau de l'état, on a besoin d'effectuer des regroupements. Il faut insérer des groupes au niveau de l'état de sortie.
Pour créer un groupe, Cliquer dans l'explorateur de champs avec le Bouton droit de la souris sur l'option Champs Nom du Groupe. Choisir Insérer un groupe. Dans la fenêtre qui apparaît choisir le champ de regroupement et valider par Ok. Dans notre exemple nous allons effectuer des regroupements par numéro de recette puisqu'on souhaite obtenir la liste des ingrédients par recette.

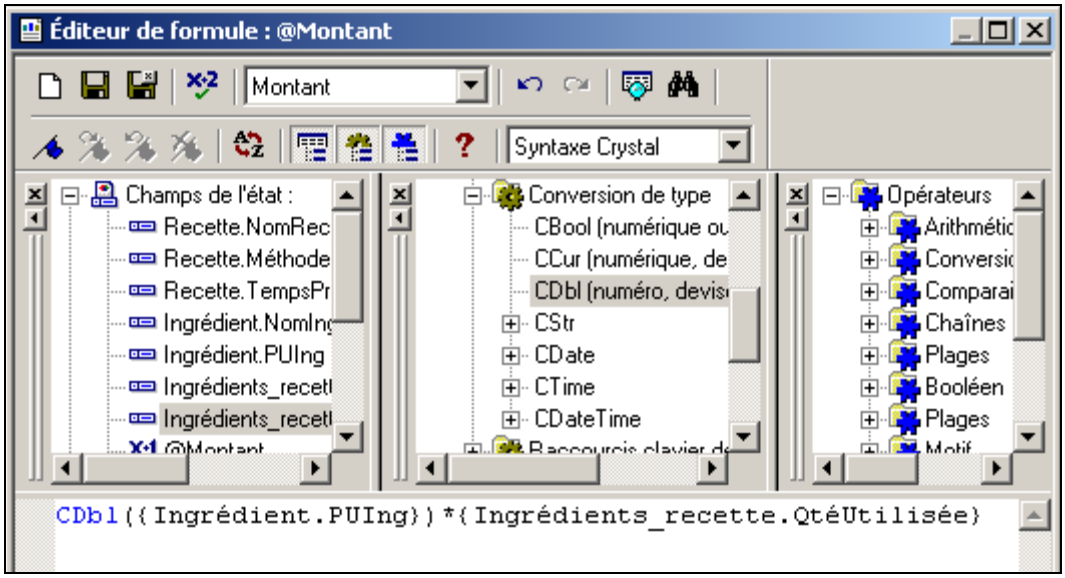
Pour utiliser des Champs spéciaux :
Ce sont des champs offerts par Crystal Reports permettant d'insérer au niveau de l'état des informations supplémentaires telles la date, l'heure et le numéro de page.

TRAVAIL A FAIRE :
- **Créer l'état de sortie RptListeRecettes**
- **Paramétrer la connexion avec le serveur et définir les champs à utiliser**
- **Créer les regroupements nécessaires**
- **Faites glisser les champs nécessaires**

Pour créer des champs calculés (de Formules)
Dans notre exemple, on souhaite calculer le montant de chaque ingrédient d'une recette. On créera alors un champ de formule nommé Montant dont la formule de calcul sera PUIng*QtéUtilisée.
Pour créer un champ calculé, cliquer dans l'explorateur de champs avec le Bouton droit de la souris sur l'option Champs de formule. Choisir Nouveau. Dans la fenêtre qui apparaît introduire le nom à attribuer à ce champ.



En appuyant sur le bouton OK, l'éditeur de formule apparaît pour assister l'utilisateur dans la réalisaion de sa formule :

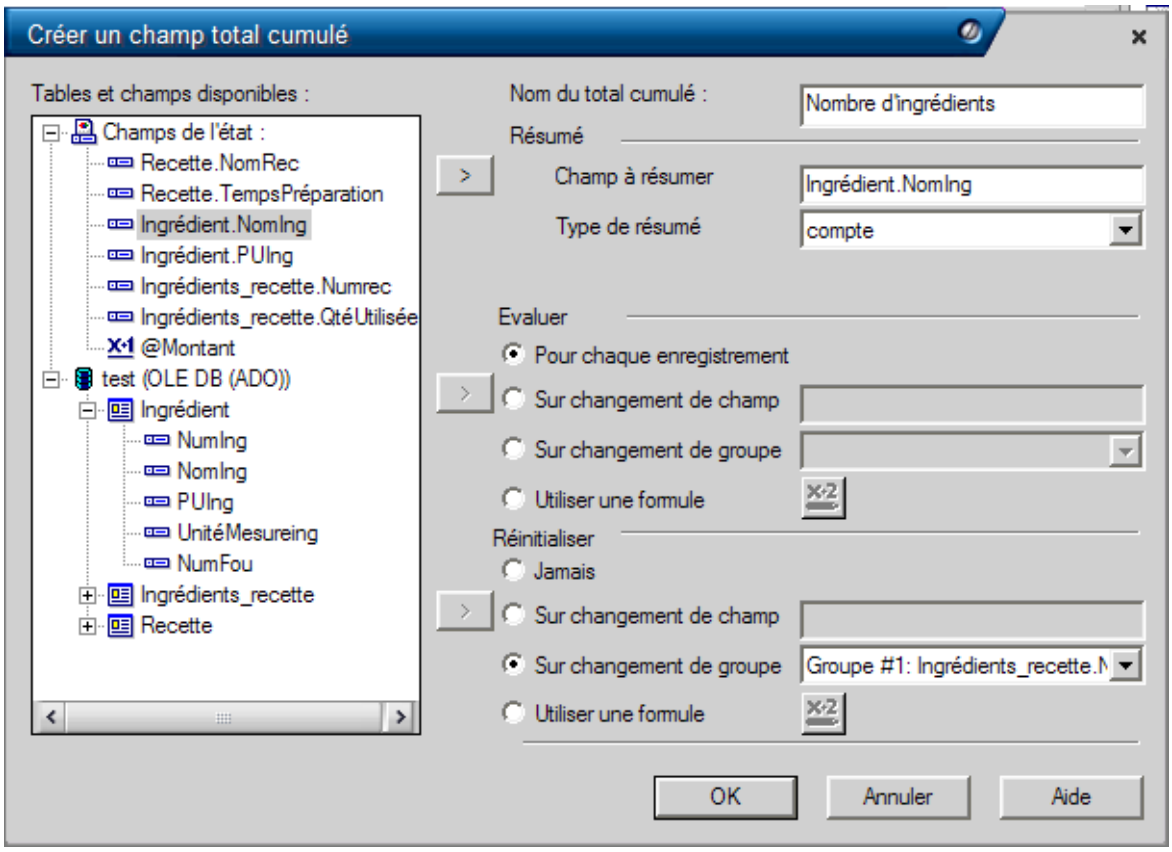


Une fois créé, ce nouveau champ apparaît dans la liste Champs de formule, pour l'insérer il suffit de cliquer dessus et de cliquer-Glisser sur l'état.

Pour créer des champs de synthèse (Champ Total Cumulé)

Dans notre exemple, on souhaite calculer le nombre d'ingrédients par recette. On créera alors un champ de total cumulé nommé Nombre d'ingrédients.

Pour créer un champ calculé, cliquer dans l'explorateur de champs avec le Bouton droit de la souris sur l'option Champs de Total Cumulé. Choisir Nouveau. Dans la fenêtre qui apparaît introduire le nom à attribuer à ce champ, le type de cumul souhaité (dans notre ca le nombre d'ingrédients) et le moment de réinitialisation des compteurs (pour obtenir un total général, par groupe…) :



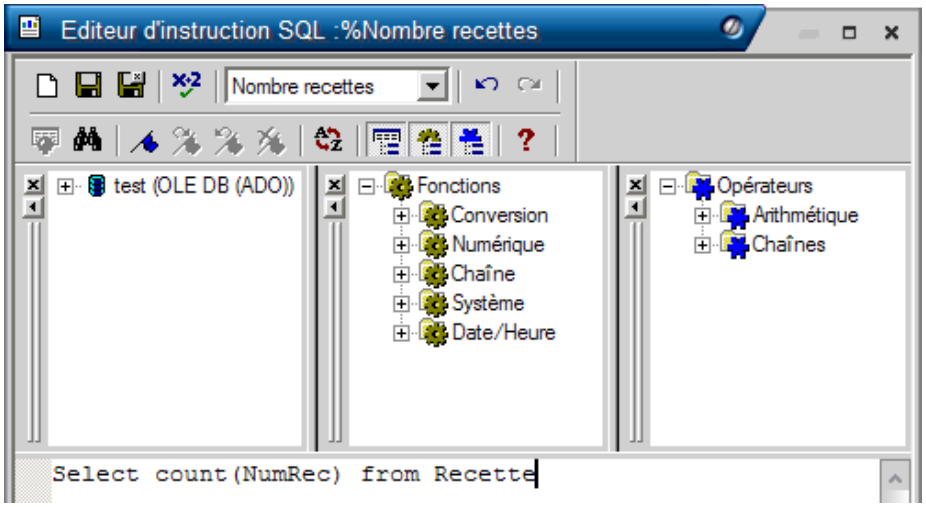
Une fois créé ce nouveau champ apparaît dans la liste Champs de total cumulé, pour l'insérer il suffit de cliquer dessus et de cliquer-Glisser sur l'état.

Pour créer des champs utilisant des requêtes SQL (Champs instruction SQL)

Dans notre exemple, on souhaite calculer le nombre total de recettes. On peut créer une instruction SQL pour le calcul de ce nombre.

Pour créer un champ instruction SQL, cliquer dans l'explorateur de champs avec le Bouton droit de la souris sur l'option Champs Instruction SQL. Choisir Nouveau. Dans la fenêtre qui apparaît introduire le nom à attribuer à ce champ.

En appuyant sur le bouton OK, l'éditeur d'instruction SQL apparaît pour assister l'utilisateur dans la création de sa requête SQL :



TRAVAIL A FAIRE :

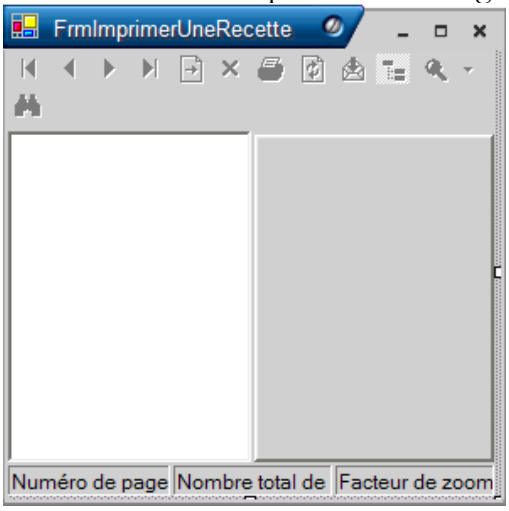
- **Ajouter le montant de chaque ingrédient composant une recette, le nombre d'ingrédients par recette et le nombre total de recettes**

Pour afficher un état de sortie

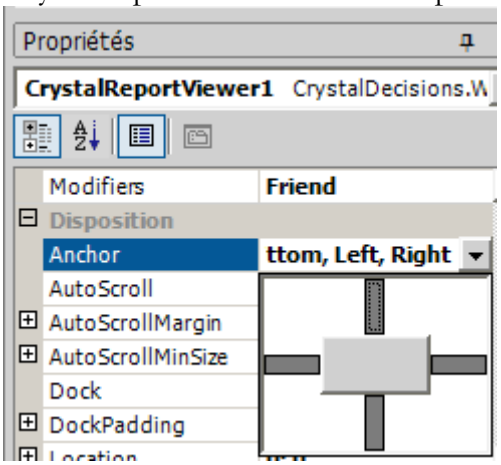
Pour afficher l'état de sortie, il faut insérer, à partir de la boîte à outils Windows Forms, un contrôle CrystalReportViewer dans un formulaire Windows.

Remarques :

- Le contrôle CrystalReportViewer devra être placé en haut à gauche du formulaire Windows :



- La propriété Anchor du CrystalReportViewer devra être positionnée ainsi :



L'étape suivante consiste à associer au CrystalReportViewer l'état de sortie à ouvrir. Deux méthodes sont possibles :

- Indiquer le nom de l'état de sortie dans la propriété Report source de l'objet CrystalReportViewer (cette méthode est déconseillée car elle pose beaucoup de problème lors du déploiement de l'application du fait que l'état de sortie sera référencé par un chemin absolue)
- Indiquer le nom de l'état de sortie par code : En supposant que le formulaire Windows a été nommé FrmImprimerListeRecettes et que le contrôle CrystalReportViewer a été nommé CrystalReportViewer1, le code à associer à l'événement Load de ce contrôle sera :

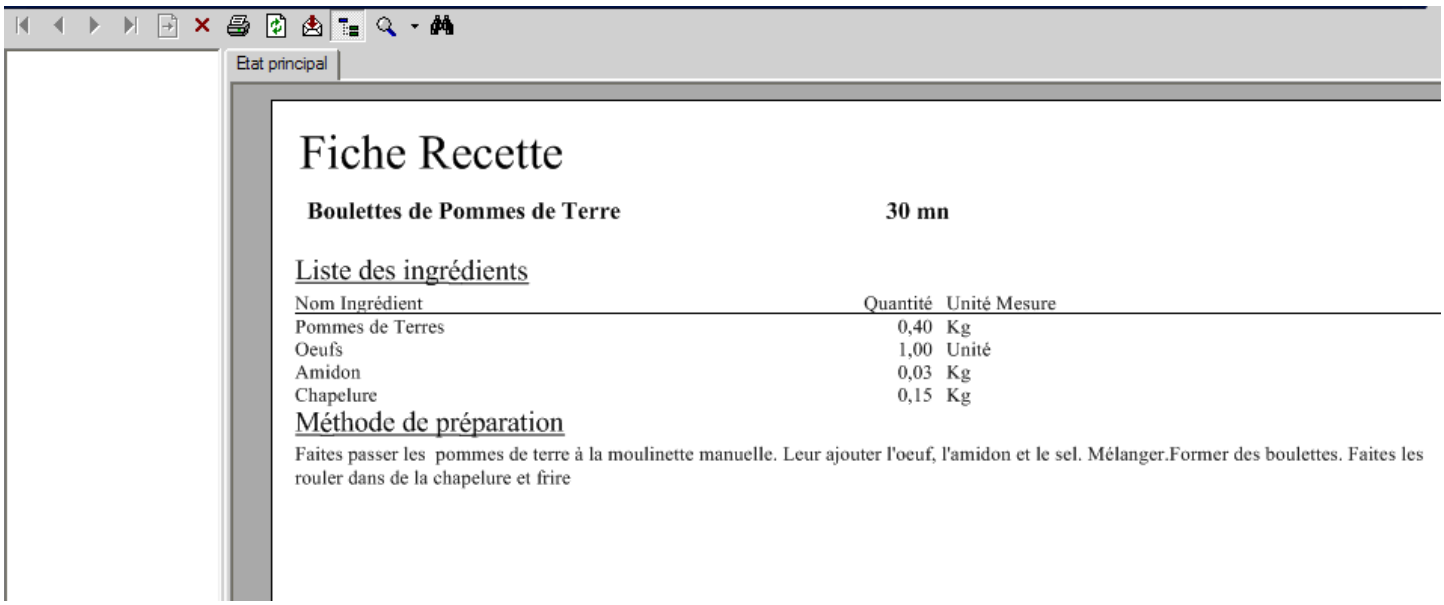
```
Imports CrystalDecisions.Shared
Public Class FrmImprimerListeRecettes
         Inherits System.Windows.Forms.Form
...
    Private Sub CrystalReportViewer1_Load(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles CrystalReportViewer1.Load
      Dim c As New RptListeRecettes
      CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

TRAVAIL A FAIRE :

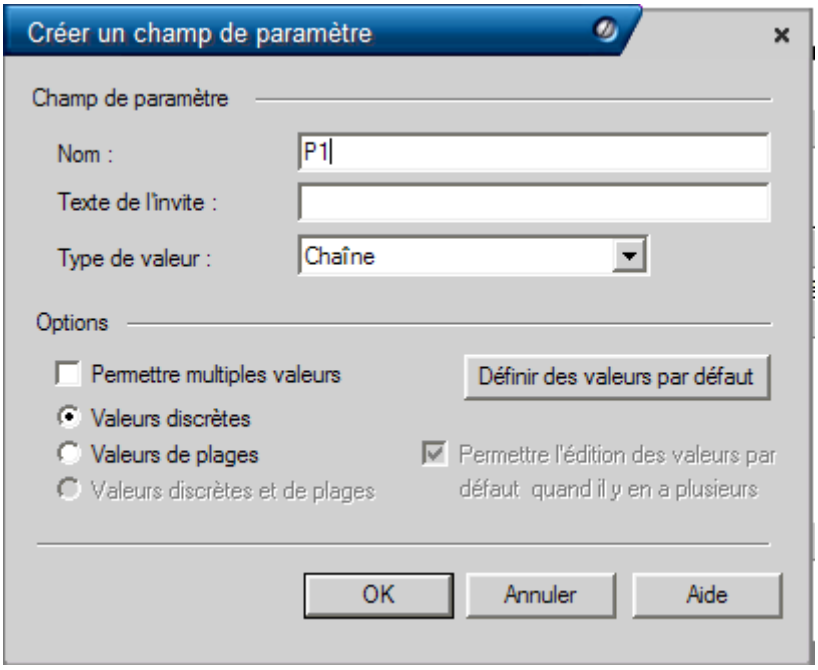
- **Créer le formulaire FrmImprimerListeRecettes**
- **Dans la feuille MDI FrmMenu, créer un élément du Menu pour accéder à ce formulaire**

Pour créer des champs de paramétres :

Un paramètre sert à filtrer la source de données. Notre objectif sera de réaliser un état de sortie qui affiche les informations sur une recette dont le nom est donné en paramètre. L'état aura l'apparence suivante :



La première étape consiste à créer l'état de sortie comme décrit précedemment. Il faut ensuite lui associer un paramètre.
Pour créer un paramètre, cliquer dans l'explorateur de champs avec le bouton droit de la souris sur l'option Champs de Paramètre. Choisir Nouveau. Dans la fenêtre qui apparaît introduire le nom du paramètre et son type :



En plus du nom et du type, il faut indiquer la nature des valeurs à donner pour ce paramètre (valeurs discrètes, valeurs de plages, valeurs discrètes multiples, valeurs de plages multiples ou valeurs discrètes et valeurs de plages).
Pour illuster les différentes options offertes, on supposera qu'on souhaite afficher les informations sur des fournisseurs dont les numéros sont donnés en paramètre :

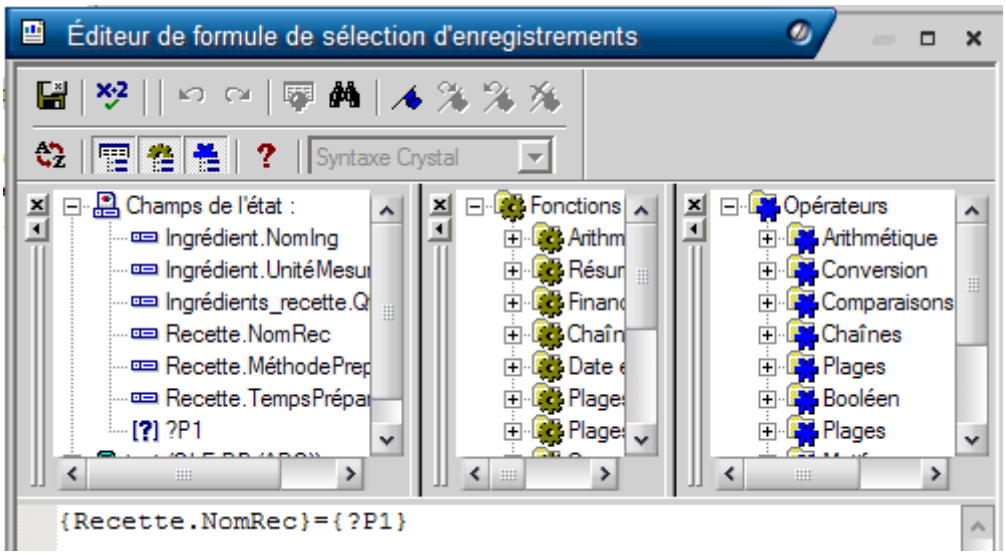
- Dans le cas où la case "Permettre muliples valeurs" n'est pas activée :
  - Avec l'option "Valeurs discrètes" active : L'utilisateur pourra introduire la valeur 3 et obtenir la fiche du fournisseur n° 3 ou (le ou exclusif) introduire la valeur 2 pour obtenir la fiche du fournisseur n° 2
  - Avec l'option "Valeurs de plages" active : L'utilisateur pourra introduire la valeur 1 et la valeur 3 pour obtenir les fiches des fournisseurs dont les numéros sont entre 1 et 3 ou (le

ou exclusif) introduire la valeur 5 et la valeur 15 pour obtenir les fiches des fournisseurs dont les numéros sont entre 5 et 15

- Dans le cas où la case "Permettre muliples valeurs" est activée :
  - Avec l'option "Valeurs discrètes" active : L'utilisateur pourra introduire la valeur 3 et la valeur 5 pour obtenir la fiche du fournisseur n° 3 et la fiche du fournisseur n° 5
  - Avec l'option "Valeurs de plages" active : L'utilisateur pourra introduire la plage de valeurs de 1 à 3 et la plage de valeurs de 5 à 15 pour obtenir les fiches des fournisseurs dont les numéros sont entre 1 et 3 ainsi que les fiches des fournisseurs dont les numéros sont entre 5 et 15.
  - Valeurs discrètes et de plages : L'utilisateur pourra introduire la valeur discrète 3, la valeur discrète 5 et la plage de valeurs entre 5 et 15 pour obtenir les fiches des fournisseurs 3, 5 et les fiches des fournisseurs dont les numéros sont compris entre 5 et 15.

Remarque : Il est possible d'éviter d'utilisation des valeurs discrètes et de plages multiples en multipliant le nombre de paramètres utilisé. Ansi au lieu d'utiliser un seul paramètre avec une valeur de plage pour indiquer qu'on souhaite avoir la liste des fournisseurs dont les numéros sont compris entre 1 et 5, on utilise deux paramètres, le premier prendra la valeur discrète 1 et le second la valeur discrète 5. Dot Net devra afficher la liste des fournisseurs dont les numéros sont compris entre la valeur du premier paramètre et celle du second paramètre.

Une fois créé, un paramètre doit être associé à une condition de sélection des enregistrements. Pour cela, appuyer avec le bouton droit de la souris sur l'état Crystal Reports et choisir l'option état / Modifier la formule de sélection / Enregistrements. Dans la fenêtre qui apparaît saisir la condition et fermer la fenêtre.



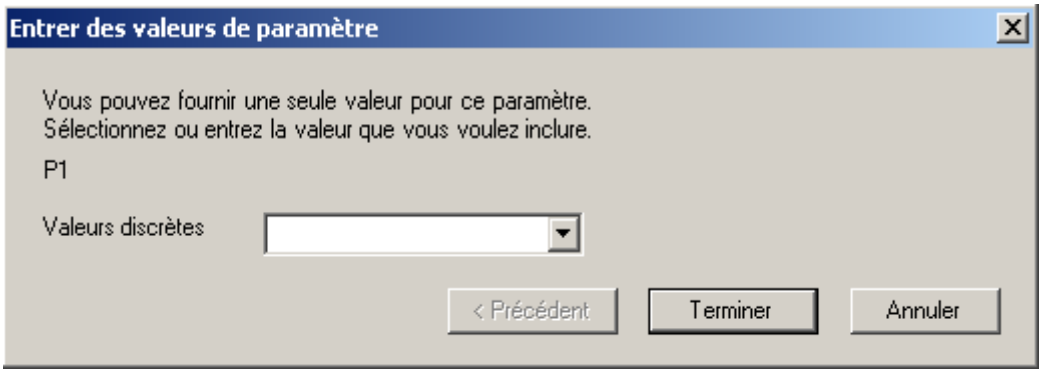
Remarque : Notez que le paramètre est spécifié avec la syntaxe suivante : {?Nom_Paramètre}

Pour afficher un état de sortie demandant des paramètres

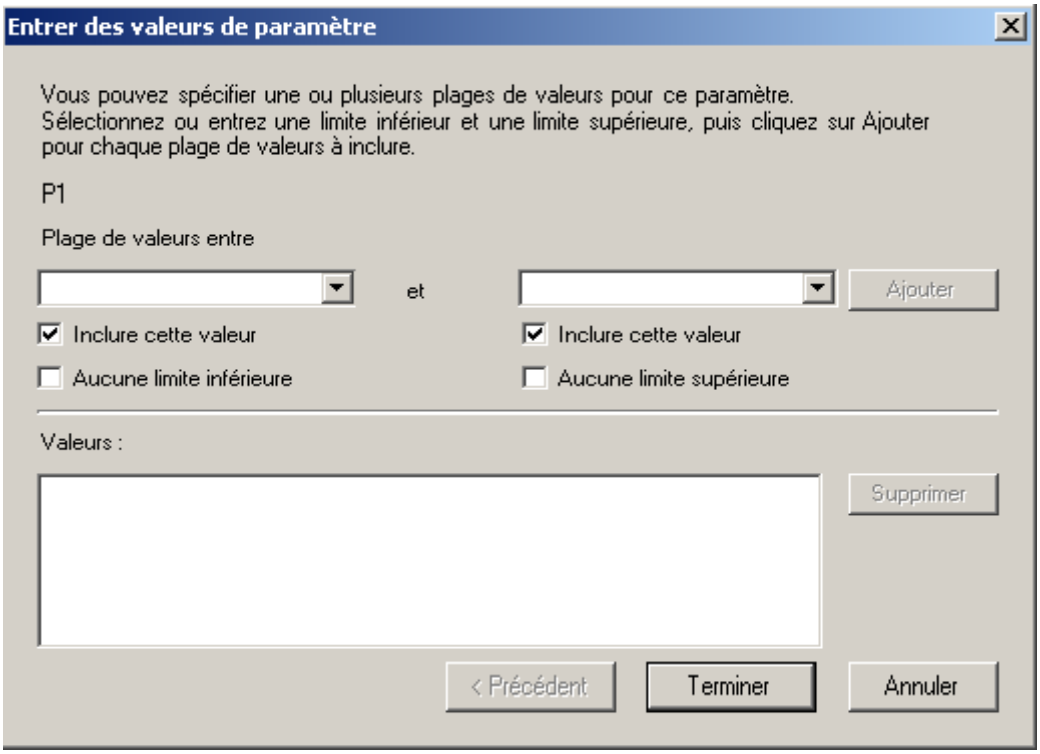
Comme dans le cas d'un état de sortie sans paramètres, il faut insérer, à partir de la boîte à outils Windows Forms, un contrôle CrystalReportViewer dans un formulaire Windows.
Pour associer au CrystalReportViewer l'état de sortie à ouvrir. Deux méthodes sont possibles :

- Indiquer le nom de l'état de sortie dans la propriété Report source de l'objet CrystalReportViewer (cette méthode est déconseillée pour des raisons liées au déploiement). A l'ouverture de l'état l'utilisateur devra saisir les valeurs pour les paramètres dans une fenêtre système proposée par Dot Net. Cette fenêtre change en fonction des paramétrages effectués pour les options valeurs discrètes et plages de valeurs :

- Dans le cas d'un paramètre P1 à valeur discrète unique, par exemple, la fenêtre système a la forme suivante :



- Dans le cas d'un paramètre P1 à valeurs de plages multiples, par exemple, la fenêtre système a la forme suivante :



- Indiquer les valeurs des paramètres et le nom de l'état de sortie par code : En supposant que le formulaire Windows a été nommé FrmImprimerUneRecette et que le contrôle CrystalReportViewer a été nommé CrystalReportViewer1, le code à associer à l'événement Load de ce contrôle sera :
  - Cas d'une valeur discrète unique : (afficher la fiche d'une recette dont le nom est donné en paramètre)

Solution 1 (plus simple)

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private Sub CrystalReportViewer1_Load(ByVal sender As
                        System.Object, ByVal e As System.EventArgs)
                        Handles CrystalReportViewer1.Load
```

```
        Dim c As New RptUneRecette
        ' Passer le paramètre à l'état
        c.SetParameterValue("P1", Variable_Imp_NomRec)
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

Solution 2

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private     Sub     CrystalReportViewer1_Load(ByVal     sender     As
                  System.Object, ByVal e As System.EventArgs) Handles
                  CrystalReportViewer1.Load
        Dim c As New RptUneRecette
        Dim Collection_Param As New ParameterFields
        Dim P As New ParameterField
           'Déclarer une valeur discrète
        Dim Val_Param_Discret As New ParameterDiscreteValue
           'Nom du paramètre tel qu'il a été déclaré dans l'état
        P.ParameterFieldName = "P1"
           'Pour attribuer une valeur à cet objet
        Val_Param_Discret.Value = Variable_Imp_NomRec
           'Pour ajouter la  valeur discrète au paramètre
        P.CurrentValues.Add(Val_Param_Discret)
           'Pour ajouter le paramètre à la collection des paramètres
        Collection_Param.Add(P)
           'Pour associer le collection de paramètre au contrôle CrystalReportViewer
        CrystalReportViewer1.ParameterFieldInfo = Collection_Param
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

Remarque : Pour utiliser les classes ParameterFields, ParameterField, ParameterDiscreteValue et ParameterRangeValue, il faut importer l'espace de nom CrystalDecisions.Shared :

TRAVAIL A FAIRE :

- **Créer l'état de sortie RptUneRecette**
- **Créer les paramètres nécessaires**
- **Ajouter au formulaire FrmRechercherRecette un bouton BtnImprimerRecette**
- **Dans la procédure évenementielle associée au bouton BtnImprimerRecette, la variable globale Variable_Imp_NomRec stockera le nom de la recette à imprimer et fera appel au formulaire FrmImprimerUneRecette**

    - Cas d'une valeur de plage unique : (les fiches des fournisseurs dont les numéros sont entre 1 et 3)

Solution 1 : Utiliser deux paramétres P1 et P2 au niveau de l'état. La formule de sélection des enregistrements d el'état de sortie sera alors : {Fournisseur.NumFou} between {?P1} and {?P2}

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
```

```
...
    Private Sub CrystalReportViewer1_Load(ByVal sender As
                        System.Object, ByVal e As System.EventArgs)
                        Handles CrystalReportViewer1.Load
        Dim p As New OleDb.OleDbParameter
        Dim c As New RptUneRecette
        ' Passer le paramètre à l'état
        c.SetParameterValue("P1", 1)
        c.SetParameterValue("P2", 3)
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

Solution 2

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private    Sub    CrystalReportViewer1_Load(ByVal    sender    As
                System.Object, ByVal e As System.EventArgs) Handles
                CrystalReportViewer1.Load
        Dim c As New RptUneRecette
        Dim Collection_Param As New ParameterFields
        Dim P As New ParameterField
           'Déclarer une valeur de plage
        Dim Val_Param_Plage As New ParameterRangeValue
        P.ParameterFieldName = "P1"
           'Spécifier les bornes de la plage
        Val_Param_Plage.StartValue = 1
        Val_Param_Plage.EndValue = 3
        P.CurrentValues.Add(Val_Param_Plage)
        Collection_Param.Add(P)
          'Pour associer le collection de paramètre au contrôle
                    CrystalReportViewer
        CrystalReportViewer1.ParameterFieldInfo = Collection_Param
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

- Cas de plusieurs valeurs discrètes : (fiche du fournisseur n° 3 et fiche du fournisseur n° 5)

Solution 1 : Consiste à utiliser plusieurs paramètres

Solution 2

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private    Sub    CrystalReportViewer1_Load(ByVal    sender    As
              System.Object, ByVal e As System.EventArgs) Handles
              CrystalReportViewer1.Load
        Dim c As New RptUneRecette
        Dim Collection_Param As New ParameterFields
        Dim P As New ParameterField
         'Première valeur discrète
        Dim Val_Param_Discret As ParameterDiscreteValue
        P.ParameterFieldName = "P1"
        Val_Param_Discret = New ParameterDiscreteValue
```

```
        Val_Param_Discret.Value = 3
        P.CurrentValues.Add(Val_Param_Discret)
         'Deuxième valeur discrète
        Val_Param_Discret = New ParameterDiscreteValue
        Val_Param_Discret.Value = 5
        P.CurrentValues.Add(Val_Param_Discret)
        Collection_Param.Add(P)
        CrystalReportViewer1.ParameterFieldInfo = Collection_Param
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

- Cas de plusieurs valeurs de plages : (les fiches des fournisseurs dont les numéros sont entre 1 et 3 ainsi que les fiches des fournisseurs dont les numéros sont entre 5 et 15)

Solution 1 : Consiste à utiliser plusieurs paramètres

Solution 2 :

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private     Sub     CrystalReportViewer1_Load(ByVal     sender     A
                System.Object,  ByVal  e  As  System.EventArgs)  Handle
                CrystalReportViewer1.Load
        Dim c As New RptUneRecette
        Dim Collection_Param As New ParameterFields
        Dim P As New ParameterField
          'Première plage
        Dim Val_Param_Plage As ParameterRangeValue
        P.ParameterFieldName = "P1"
        Val_Param_Plage = New ParameterRangeValue
        Val_Param_Plage.StartValue = 1
        Val_Param_Plage.EndValue = 3
        P.CurrentValues.Add(Val_Param_Plage)
          'Deuxième plage
        Val_Param_Plage = New ParameterRangeValue
        Val_Param_Plage.StartValue = 5
        Val_Param_Plage.EndValue = 15
        P.CurrentValues.Add(Val_Param_Plage)
        Collection_Param.Add(P)
        CrystalReportViewer1.ParameterFieldInfo = Collection_Param
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

- Cas de plusieurs valeurs discrètes et de plusieurs valeurs de plages : (la fiche du fournisseur n°1, les fiches des fournisseurs dont les numéros sont entre 5 et 8 ainsi que les fiches des fournisseurs dont les numéros sont entre 10 et 15)

Solution 1 : Consiste à utiliser plusieurs paramètres

Solution 2 :

```
Imports CrystalDecisions.Shared
Public Class FrmImprimerUneRecette
    Inherits System.Windows.Forms.Form
...
    Private     Sub     CrystalReportViewer1_Load(ByVal     sender     A
```

```
                System.Object, ByVal e As System.EventArgs) Handles
                CrystalReportViewer1.Load
        Dim c As New RptUneRecette
        Dim Collection_Param As New ParameterFields
        Dim P As New ParameterField
        Dim Val_Param_Discret As New ParameterDiscreteValue
        Dim Val_Param_Plage As ParameterRangeValue
        P.ParameterFieldName = "P1"
        Val_Param_Discret.Value = 1
        P.CurrentValues.Add(Val_Param_Discret)
        Val_)Param_Plage = New ParameterRangeValue
        Val_Param_Plage.StartValue = 5
        Val_Param_Plage.EndValue = 8
        P.CurrentValues.Add(Val_Param_Plage)
        Val_Param_Plage = New ParameterRangeValue
        Val_Param_Plage.StartValue = 10
        Val_Param_Plage.EndValue = 15
        P.CurrentValues.Add(Val_Param_Plage)
        Collection_Param.Add(P)
        CrystalReportViewer1.ParameterFieldInfo = Collection_Param
        CrystalReportViewer1.ReportSource = c
    End Sub
End Class
```

En mode déconnecté

Le client se connecte au serveur, formule sa requête et reçoit la réponse. Cette réponse sera stockée dans des objets DataTable ou DataSet via un objet DataAdapter sur l'ordinateur local. La connexion est ensuite coupée. Le client travaillera alors sur la copie en local. Toutes les modifications apportées à la copie en local pourraient être transmises via le même objet DataAdapter vers le serveur.

UTILISATION DES OBJETS DATATABLE ET DATASET

Présentation des objets DataTable et DataSet

Les objets DataTable et DataSet sont des structures de stockage de données indépendantes des bases de données et donc indépendantes des fournisseurs utilisés pour l'accès aux données.

- Un DataTable est semblable à une table dans une base de données. C'est une collection de lignes (DataRow) et chaque DataRow est une collection de colonnes (DataColumn). Pour utiliser un objet DataTable :

```
Dim Réf_DataTable as New DataTable (déclaration et instanciation)
```

- Un DataSet est plûtot semblable à une base de données. Il se compose d'un ou de plusieurs objets DataTable reliés éventuellement par des relations DataRelation et auxquelles on peut appliquer des contraintes (Clé Primaire, Unique...). Pour utiliser un objet DataSet :

```
Dim Réf_DataSet as New DataSet (déclaration et instanciation)
```

En plus ces structures permettent de remplir facilement des contrôles DataGrid, ListBox, ComboBox...

Utilisation des objets DataTable

Détermination des colonnes

Pour utiliser un objet DataTable, il faut déterminer ses colonnes

```
Dim Réf_DataTable as New DataTable
Réf_DataTable.Columns.add(NomPremièreColonne,TypePremièreColonne, Expression)
Réf_DataTable.Columns.add(NomDeuxièmeColonne,TypeDeuxièmeColonne, Expression)
Réf_DataTable.Columns.add(NomTroisièmeColonne,TypeTroisièmeColonne, Expression)
...
```

Remarque:

- Seul le nom de colonne est obligatoire, le type est utilisé pour déterminer le type de données et Expression est utilisée pour indiquer qu'une colonne est calculée à partir d'une expression
- Pour faire incrémenter une colonne automatiquement :

```
Réf_DataTable.Columns(Index_Col|"Nom_Col").AutoIncrement = True
Réf_DataTable.Columns(Index_Col|"Nom_Col").AutoIncrementStep = valeur
```

- Pour masquer une colonne :

```
Réf_DataTable.Columns(Index_Col|"Nom_Col").ColumnMapping=MappingType.Hidden
```

Définition des contraintes sur les colonnes

Clé primaire

Dans un DataTable, comme dans une table de base de données, il est possible de désigner certaines colonnes comme colonne clé primaire.

Pour le faire, il faut tout d'abord déclarer un tableau contenant autant de cases qu'il y a de champs composant la clé primaire ensuite il faut associer à chaque case du tableau, le nom du champ qui contribuera à la clé primaire.

```
Dim Réf_Tableau(Nbr_Champs_Clé_Primaire) As DataColumn
Réf_Tableau(0) = Réf_DataTable.Columns(index_Col|"Nom_Col")
Réf_Tableau(1) = Réf_DataTable.Columns(index_Col|"Nom_Col")
...
Réf_DataTable.PrimaryKey = Réf_Tableau
```

Valeurs nulles

Pour autoriser ou interdire des valeurs nulles

```
Réf_DataTable.Columns(Index_Col|"Nom_Col").AllowDBNull=True|False
```

Colonnes en lecture seule

Pour autoriser ou interdire la modification des données d'une colonne

```
Réf_DataTable.Columns(Index_Col|"Nom_Col").ReadOnly=True|False
```

Contrainte Unique

Pour autoriser ou interdire que des valeurs de cette même colonne puissent se répéter

```
Réf_DataTable.Columns(Index_Col|"Nom_Col").Unique = True|False
```

Vidage d'un DataTable

Pour supprimer le contenu d'un DataTable sans supprimer ses colonnes :

```
Réf_DataTable.Clear()
```

Ajout de lignes à un DataTable

Pour ajouter une ligne à un objet DataTable, il faut d'abord préparer la ligne à ajouter et puis l'insérer dans le DataTable

```
Dim DR as DataRow
DR=Réf_DataTable.newRow()
'Affectation des valeurs aux colonnes de la ligne (Les colonnes doivent
'correspondre aux colonnes du DataTable
DR(0)=...
DR(1)=...
DR(2)=...
...
Réf_DataTable.Rows.add(DR)
```

Recherche d'un élément dans un DataTable

Pour rechercher un élément dans un DataTable, plusieurs méthodes peuvent être utilisées :

Accès séquentiel

Méthode 1 :

```
Dim i As Int16
For i = 0 To Réf_DataTable.Rows.Count - 1
   If Réf_DataTable.Rows(i)(index_Col) = Valeur Then
         ...
```

```
        End If
    Next
```

Méthode 2 :

```
Dim DR as DataRow
For Each DR In Réf_DataTable.Rows
    if DR(index_Col) = valeur Then
          ...
    End If
Next
```

Accès Direct

Si on connaît la position :

```
Réf_DataTable.Rows(position)...
```

Pour accèder à la position en cours dans un DataTable :

```
Dim p as int32
p = BindingContext(Réf_DataTable).Position
```

Pour positionner le DataTable :

```
BindingContext(Réf_DataTable).Position = p
```

Suppression de lignes à partir d'un DataTable

Pour supprimer une ligne à partir d'un DataTable :

```
Réf_DataTable.Rows(position_ligne_A_Supprimer).Delete()
```

Modification d'une ligne dans un DataTable

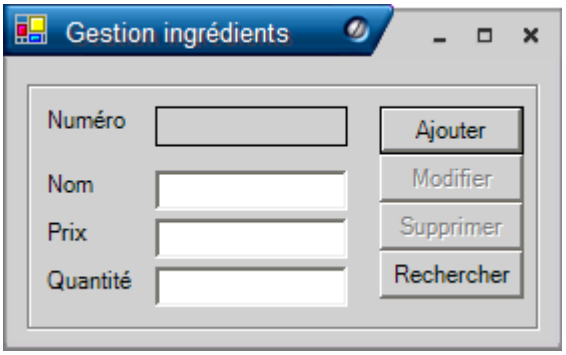
Pour modifier la ligne d'un DataTable :

```
Réf_DataTable.Rows(position_ligne_A_Modifier).BeginEdit()
Réf_DataTable.Rows(position_ligne_A_Modifier)(position_colonne_A_Modifier) = ...
Réf_DataTable.Rows(position_ligne_A_Modifier)(position_colonne_A_Modifier) = ...
...
Réf_DataTable.Rows(position_ligne_A_Modifier).EndEdit()
```

Exemple : ***Utilisation des DataTable pour la gestion d'une liste d'ingrédients***

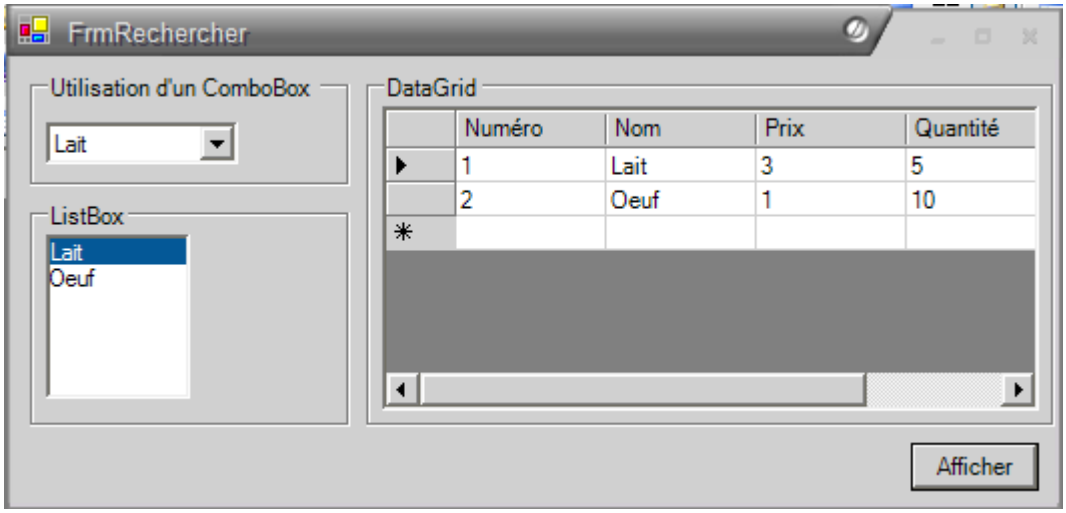
Cet exemple sera totalement indépendant des bases de données. On apprendra à manipuler des objets DataTable et à les exploiter pour remplir des objets Combobox, ListBox et DataGrid :

Soit le formulaire FrmUtiliserDataTable :



Dans ce formulaire, l'utilisateur saisit les informations sur un ingrédient et appuie sur le bouton "Ajouter" pour enregistrer cet ingrédient dans un DataTable.

En cliquant sur le bouton Rechercher le formulaire FrmRechercher s'affiche :



L'utilisateur choisit dans ce formulaire un ingrédient (dans les contrôles comboBox, ListBox ou DataGrid). En cliquant sur le bouton "Afficher" les informations sur l'ingrédient choisi s'affichent dans le premier formulaire et les boutons "Modifier" et "Supprimer" seront actifs.

Soit le code associé à un module :

```
Module Module1
    Public frm1 As FrmUtilserDataTable
    Public frm2 As FrmRechercher
    Public p As Integer
    Public Datat As New DataTable
    Sub main()
        Datat.Columns.Add("Numéro", GetType(Int32))
        Datat.Columns.Add("Nom", GetType(String))
        Datat.Columns.Add("Prix", GetType(Decimal))
        Datat.Columns.Add("Quantité", GetType(Int64))
        Datat.Columns.Add("Montant", GetType(Decimal), "Prix*Quantité")
        'Créer une colonne numéro automatique
        Datat.Columns(0).AutoIncrement = True
        Datat.Columns(0).AutoIncrementSeed = 1
        Datat.Columns(0).AutoIncrementStep = 1
        'Définir une contrainte clé primaire
        Datat.Columns(1).Unique = True
        'Définir une clé primaire
        Dim T(0) As DataColumn
        T(0) = Datat.Columns(0)
        Datat.PrimaryKey = T

        frm1 = New FrmTesterDataTable
        Application.Run(frm1)
    End Sub
End Module
```

Soit le code associé à FrmUtiliserDataTable :

```
Public Class FrmTesterDataTable
    Inherits System.Windows.Forms.Form
...
   Private Sub FrmTesterDataTable_Load(ByVal sender As System.Object, ByVal e As
                                       System.EventArgs) Handles MyBase.Load
        p = -1
   End Sub
   Private Sub BtnRechercher_Click(ByVal sender As System.Object, ByVal e As
                                   System.EventArgs) Handles BtnRechercher.Click
        If Datat.Rows.Count = 0 Then
            MessageBox.Show("Aucun élément à rechercher")
        Else
```

```
            frm2 = New FrmRechercher
            frm2.Show()
        End If
    End Sub
    Private Sub BtnAjouter_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAjouter.Click
        Dim DR As DataRow
        DR = Datat.NewRow
        DR(1) = TxtNom.Text
        DR(2) = Val(TxtPrix.Text)
        DR(3) = Val(TxtQte.Text)
        Datat.Rows.Add(DR)
        vider()
    End Sub
    Sub vider()
        TxtNom.Text = ""
        TxtPrix.Text = ""
        TxtQte.Text = ""
    End Sub
    Private Sub BtnModifier_Click(ByVal sender As System.Object, ByVal e As
                                  System.EventArgs) Handles BtnModifier.Click
        If p <> -1 Then
            Datat.Rows(p).BeginEdit()
            Datat.Rows(p)(1) = TxtNom.Text
            Datat.Rows(p)(2) = Val(TxtPrix.Text)
            Datat.Rows(p)(3) = Val(TxtQte.Text)
            Datat.Rows(p).EndEdit()
            vider()
            p = -1
            BtnModifier.Enabled = False
            BtnSupprimer.Enabled = False
        End If
        'ou
        'For i As Integer = 0 To Datat.Rows.Count - 1
        '    If Datat.Rows(i)(0) = LabelNuméro.Text Then
        '        Datat.Rows(i).BeginEdit()
        '        Datat.Rows(i)(1) = TxtNom.Text
        '        Datat.Rows(i)(2) = Val(TxtPrix.Text)
        '        Datat.Rows(i)(3) = Val(TxtQte.Text)
        '        Datat.Rows(i).EndEdit()
        '        BtnModifier.Enabled = False
        '        Exit Sub
        '    End If
        'Next
    End Sub
    Private Sub BtnSupprimer_Click(ByVal sender As System.Object, ByVal e As
                                   System.EventArgs) Handles BtnSupprimer.Click
        If p <> -1 Then
            Datat.Rows(p).Delete()
            vider()
            p = -1
            BtnSupprimer.Enabled = False
            BtnModifier.Enabled = False
        End If
        'ou
        'For i As Integer = 0 To Datat.Rows.Count - 1
        '    If Datat.Rows(i)(0) = LabelNuméro.Text Then
        '        Datat.Rows(i).delete()
        '        Exit Sub
        '    End If
        'Next
    End Sub
End Class
```

Soit le code associé à FrmRechercher :

```
Public Class FrmRechercher
    Inherits System.Windows.Forms.Form
   Private Sub FrmRechercher_Load(ByVal sender As System.Object, ByVal e As
                                  System.EventArgs) Handles MyBase.Load
        'Remplir un DataGrid
        DataGrid1.DataSource = DataT
        'Remplir un ComboBox
        ComboBox1.DataSource = DataT
        ComboBox1.DisplayMember = DataT.Columns(1).ColumnName
        'Remplir un ListBox
        ListBox1.DataSource = DataT
        ListBox1.DisplayMember = DataT.Columns(1).ColumnName
   End Sub
   Private Sub BtnAfficher_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAfficher.Click
        p = BindingContext(Datat).Position
        frm1.LabelNuméro.Text = Datat.Rows(p)(0)
        frm1.TxtNom.Text = Datat.Rows(p)(1)
        frm1.TxtPrix.Text = Datat.Rows(p)(2)
        frm1.TxtQte.Text = Datat.Rows(p)(3)
        frm1.BtnModifier.Enabled = True
        frm1.BtnSupprimer.Enabled = True
        Me.Hide()
   End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer un nouveau projet VB.Net**
- **Dans un module, déclarer les variables nécessaires et créer la procédure main**
- **Démarrer le projet sur Sub Main (Menu Projet / Propriétés de … / Objet de démarrage/ Sub main)**
- **Créer le formulaire FrmUtiliserDataTable**
- **Créer le formulaire FrmRechercher**
- **Ecrire les programmes nécessaires et exécuter l'application**

Utilisation des objets DataSet

La DataSet est un groupe de DataTable et donc chaque DataTable membre du DataSet sera manipulé comme tout DataTable normal sauf que pour y accéder, on utilisera la syntaxe suivante :

```
Réf_DatSet.Tables(index_DataTable|"nomDataTable")….
```

Détermination des relations entre les DataTable membres d'un DataSet

Il est possible de mettre en relation les DataTable composant un DataSet :

```
Réf_DatSet.Relations.add("NomRelation", Colonne_parent , Colonne_fille)
```

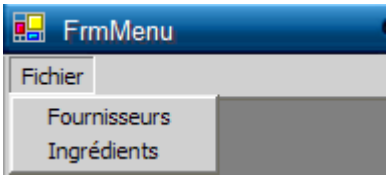
Vidage d'un DataSet

Pour supprimer le contenu d'un DataSet sans supprimer sa structure :

```
Réf_Dataset.Clear()
```

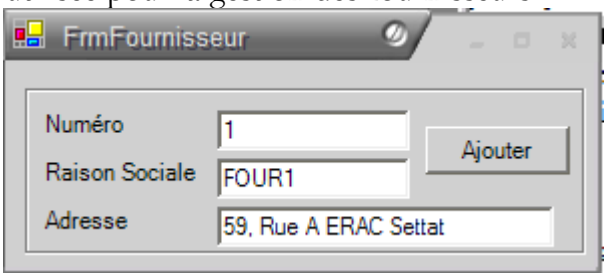
Exemple : ***Utilisation des DataSet pour la gestion d'une liste d'ingrédients et de leurs fournisseurs***
Cet exemple est également indépendant des bases de données. On apprendra à manipuler des objets DataTable appartenant à des DataSet.

Soit le formulaire FrmMenu :

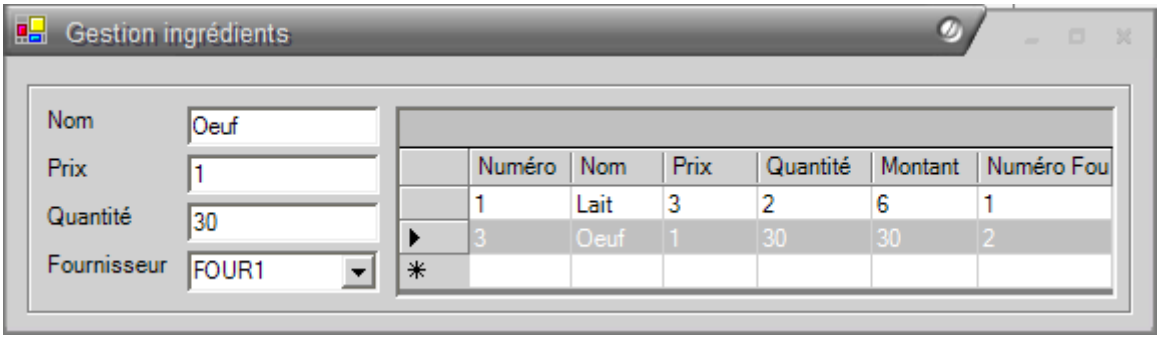


Ce formulaire permet d'accèder à une première fenêtre pour la gestion des fournisseurs et à une seconde fenêtre pour la gestion des ingrédients :

Soit la fenêtre FrmFournisseur utilisée pour la gestion des fournisseurs :



Soit la fenêtre FrmIngrédient utilisée pour la gestion des ingrédients :



Soit le code associé à un module :

```
Module Module1
    Public p As Integer
    Public DataS As New DataSet
    Sub main()
        Dim DataT As DataTable
        'Remplir le premier DataTable du DataSet
        DataT = New DataTable
        DataT.TableName = "DT_Ingrédient"
        DataT.Columns.Add("Numéro", GetType(Int32))
        Datat.Columns.Add("Nom", GetType(String))
        Datat.Columns.Add("Prix", GetType(Decimal))
        Datat.Columns.Add("Quantité", GetType(Int64))
        DataT.Columns.Add("Montant", GetType(Decimal), "Prix*Quantité")
        DataT.Columns.Add("Numéro Fournisseur", GetType(Integer))
        DataS.Tables.Add(DataT)
        'Remplir le deuxième DataTable du DataSet
        DataT = New DataTable
        DataT.TableName = "DT_Fournisseur"
        DataT.Columns.Add("Numéro Fournisseur", GetType(Integer))
        DataT.Columns.Add("Raison Sociale", GetType(String))
        DataT.Columns.Add("Adresse", GetType(String))
        DataS.Tables.Add(DataT)
        'définir les contraintes
        'Numéro Auto pour le premier DataTable
        'DataS.Tables(0).Columns(0).AutoIncrement = True
        'ou
        DataS.Tables("DT_Ingrédient").Columns("Numéro").AutoIncrement = True
        DataS.Tables(0).Columns(0).AutoIncrementSeed = 1
        DataS.Tables(0).Columns(0).AutoIncrementStep = 1
```

```
        'Contrainte Unique
        DataS.Tables(0).Columns(1).Unique = True
        'Clé primaire du premier DataTable
        Dim PK_DT_Ingrédient(0) As DataColumn
        PK_DT_Ingrédient(0) = DataS.Tables(0).Columns(0)
        DataS.Tables(0).PrimaryKey = PK_DT_Ingrédient
        'Clé primaire du second DataTable
        Dim PK_DT_Fournisseur(0) As DataColumn
        PK_DT_Fournisseur(0) = DataS.Tables(1).Columns(0)
        DataS.Tables(1).PrimaryKey = PK_DT_Fournisseur
        'Création d'une relation
        DataS.Relations.Add("Rel_Ing_Four", DataS.Tables(1).Columns(0),
                            DataS.Tables(0).Columns(5))
        Dim FrmMenu As New FrmMenu
        Application.Run(FrmMenu)
    End Sub
End Module
```

Soit le code associé à FrmMenu :

```
Public Class FrmMenu
    Inherits System.Windows.Forms.Form
...
    Private Sub MnuFour_Click(ByVal sender As System.Object, ByVal e As
                          System.EventArgs) Handles MnuFour.Click
        Dim FrmFour As New FrmFournisseur
        FrmFour.Show()
    End Sub

    Private Sub MnuIng_Click(ByVal sender As System.Object, ByVal e As
                           System.EventArgs) Handles MnuIng.Click
        Dim FrmIng As New FrmIngrédient
        FrmIng.Show()
    End Sub
End Class
```

Soit le code associé à FrmFournisseur :

```
Public Class FrmFournisseur
    Inherits System.Windows.Forms.Form
    Private Sub BtnAjouter_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAjouter.Click
        Dim DR As DataRow
        DR = DataS.Tables(1).NewRow
        DR(0) = Val(TxtNum.Text)
        DR(1) = TxtRS.Text
        DR(2) = TxtAdresse.Text
        DataS.Tables(1).Rows.Add(DR)
        vider()
    End Sub
    Sub vider()
        TxtNum.Text = ""
        TxtRS.Text = ""
        TxtAdresse.Text = ""
    End Sub
End Class
```

Soit le code associé à FrmIngrédient :

```
Public Class FrmIngrédient
    Inherits System.Windows.Forms.Form

    Private Sub FrmIngrédient_Load(ByVal sender As System.Object, ByVal e As
```

```
                                System.EventArgs) Handles MyBase.Load
        'Remplissage d'une zone de liste modifiable avec les raisons sociales des
        ' fournisseurs présents dans le DataTable des fournisseurs
        CmbFournisseur.DataSource = DataS.Tables(1)
        CmbFournisseur.DisplayMember = DataS.Tables(1).Columns(1).ColumnName

        'Remplissage d'un DataGrid avec la liste des ingrédients présents dans le
        'DataTable des ingrédients
        DataGrid1.DataSource = DataS.Tables(0)

        'Pour rendre les zones d'affichage et de saisie dépendantes du contenu du
        'DataTable
       TxtNom.DataBindings.Add("text",DataS.Tables(0),
                                              DataS.Tables(0).Columns(1).ColumnName)
        TxtPrix.DataBindings.Add("text", DataS.Tables(0),
                                              DataS.Tables(0).Columns(2).ColumnName)
        TxtQte.DataBindings.Add("text", DataS.Tables(0),
                                              DataS.Tables(0).Columns(3).ColumnName)

    End Sub
End Class
```

TRAVAIL A FAIRE :

- **Créer un nouveau projet VB.Net**
- **Dans un module, déclarer les variables nécessaires et créer la procédure main**
- **Démarrer le projet sur Sub Main (Menu Projet / Propriétés de … / Objet de démarrage/ Sub main)**
- **Créer le formulaire FrmMain**
- **Créer le formulaire FrmFournisseur**
- **Créer le formulaire FrmIngrédient**
- **Ecrire les programmes nécessaires et exécuter l'application**

L'OBJET DATAADAPTER

L'objet DataAdapter sert de liaison entre un objet de stockage de données (DataTable ou DataSet) et une Source de Données. Il permet :

- De remplir un DataTable ou un DataSet à partir des données contenus dans la base de données
- De mettre à jour une base de données à partir d'un DataTable ou d'un DataSet

Instantiation :

Pour utiliser l'objet DataAdapter, il faut créer selon le cas une instance de la classe OleDbDataAdapter ou SqlDataAdapter :

En utilisant Ole db

```
Imports System.Data.OleDb
Dim Réf_DataAdapter as New OleDbDataAdapter (déclaration et instanciation)
```

Ou

```
Dim Réf_DataAdapter  as OleDbDataAdapter  (Déclaration)
Réf_DataAdapter  = New OleDbDataAdapter (instantiation)
```

Exemple :

```
Public DataA as new OleDbDataAdapter
```

En utilisant le driver optimisé

```
Imports System.Data.SqlClient
Dim Réf_DataAdapter as New SqlDataAdapter (déclaration et instanciation)
```

Ou

```
Dim Réf_DataAdapter  as SqlDataAdapter  (Déclaration)
Réf_DataAdapter  = New SqlDataAdapter (instantiation)
```

Exemple :

```
Public DataA as new SqlDataAdapter
```

Remarque :

Dans ce qui va suivre nous allons travailler avec le fournisseur de données OleDb. Rappellons que pour passer au fournisseur de données optimisé pour SQL Server il suffit de remplacer OleDb par SQL dans toutes les instructions et d'utiliser l'espace de noms System.data.SqlClient au lieu de System.data.oleDb. Toutes les autres différences, si elles existent, seront signalées au fur et à mesure.

Paramétrage d'un DataAdapter

La chaîne de connexion

Pour fonctionner, un DataAdapter a **obligatoirement** besoin d'une chaîne de connexion valide. Il faut alors declarer un objet Connection et définir sa chaîne de connexion (par constructeur ou en utilisant la propriété ConnectionString).

La requête de selection de données à partir de la base de données

Un DataAdapter a aussi besoin d'une requête pour la récupération des données à partir de la base de données. Cette requête peut être déterminée de deux manières :

- En utilisant un constructeur de la classe OleDBDataAdapter :
  ```
  Dim Réf_DataAdapter as New OleDbDataAdapter("Requête",Réf_Connexion)
  ```
- En utilisant l'objet SelectCommand (cet objet sera décrit plus târd)

Les requêtes de Mise à jour de la base de données à partir des DataTable et DataSet

Un DataAdapter peut avoir besoin, selon chaque situation, des objets InsertCommand, DeleteCommand ou UpdateCommand qui lui permettront respectivement d'insérer, de modifier ou de supprimer des lignes da la source de données à partir d'objets DataTable ou DataSet.

Remarque :

- Les objets SelectCommand, InsertCommand, DeleteCommand et UpdateCommand sont des objets Command et sont donc traités comme les objets Command précédemment étudiés et bénéficient des mêmes propriétés et méthodes (instantiation, Connexion, CommandText...)
- Supposons que l'utilisateur n'a que le droit d'ajouter des recettes sans avoir le droit de les supprimer ou de les modifier, nous allons donc uniquement définir les objets SelectCommad et InsertCommand.

Utilisation des Objets SelectCommand, InsertCommand, DeleteCommand et UpdateCommand

Objet	Utilisation
SelectCommand	Réf_DataAdapter.SelectCommand=new OleDBCommand() Réf_DataAdapter.SelectCommand.Connection=Réf_Connexion Réf_DataAdapter.SelectCommand.CommandType=... Réf_DataAdapter.SelectCommand.CommandText="Requête_Select|Nom_Procédure" Remarque : Une requête sélection est obligatoire dans tous les cas (en utilisant le constructeur ou SelectCommand)
InsertCommand	Réf_DataAdapter.insertCommand=new OleDBCommand() Réf_DataAdapter.insertCommand.Connection=Réf_Connexion Réf_DataAdapter.insertCommand.CommandType=... Réf_DataAdapter.insertCommand.CommandText="Requête_insert|Nom_Procédure" Remarque : Utilisée si l'utilisateur ajoute des lignes en local (à un objet DataTable ou DataSet) et que ces données doivent être transférées à la base de données
DeleteCommand	Réf_DataAdapter.DeleteCommand=new OleDBCommand() Réf_DataAdapter.DeleteCommand.Connection=Réf_Connexion

	Réf_DataAdapter.DeleteCommand.CommandType=... Réf_DataAdapter.DeleteCommand.CommandText="Requête_delete|Nom_Procédure" Remarque : Utilisée si l'utilisateur supprime des lignes en local (à partir d'un objet DataTable ou DataSet) et que ces lignes doivent également être supprimées de la base de données
UpdateCommand	Réf_DataAdapter.UpdateCommand=new OleDBCommand() Réf_DataAdapter.UpdateCommand.Connection=Réf_Connexion Réf_DataAdapter.UpdateCommand.CommandType=... Réf_DataAdapter.UpdateCommand.CommandText="Requête_update|Nom_Procédure" Remarque : Utilisée si l'utilisateur modifie des lignes en local (à un objet DataTable ou DataSet) et que ces modifications doivent être également apportées à la base de données

Remplissage d'un DataTable ou d'un DataSet à partir d'une base de données

La méthode Fill de l'objet DataAdapter permet de remplir automatiquement un objet DataTable ou DataSet. A l'appel de la méthode Fill, le système recherche la requête de sélection associée à l'objet DataAdapter concerné et l'exécute (objet selectCommand). Le résultat retourné va être stocké dans un DataTable ou un DataSet.

- Pour remplir un DataTable
  ```
  Réf_DataAdapter.Fill(Réf_DataTable)
  ```
- Pour remplir un DataSet
  ```
  Réf_DataAdapter.Fill(Réf_DataSet,Nom_DataTable)
  ```

Remarque :

- Les colonnes des DataTable seront automatiquement créées et le développeur dans ce cas n'a pas besoin de les définir. Par contre les clés primaires, les contraintes spéciales et les relations entre les DataTable membres d'un DataSet devront être créées.
- En général, on a besoin d'un DataAdapter par DataTable

Mise à jour d'une base de données à partir d'un DataTable ou d'un DataSet

L'utilisateur travaille en local sur les données contenues dans le DataTable ou le DataSet. Selon les fonctionnalités offertes par l'application, il peut ajouter, modifier ou supprimer des données du DataTable ou du DataSet. Pour transmettre les modifications apportées par l'utilisateur à la source de données, il faut utiliser la méthode Update de l'objet DataAdapter.

```
Réf_DataAdapter.Update(Réf_DataTable)
ou
Réf_DataAdapter.Update(Réf_DataSet.Tables(index_DataTable|"Nom_DataTable"))
```

Pour comprendre comment le DataAdapter peut effectuer les opérations de mise à jour de la base de données, il est important de comprendre comment les opérations de mise à jour se font au niveau des DataTable et des DataSet :

- Quand un utilisateur supprime des lignes d'un DataTable ou d'un DataSet remplis à partir d'un DataAdapter, le système marque ces lignes comme deleted (RowState = DataRowState.Deleted)
- Quand un utilisateur ajoute des lignes à un DataTable ou à un DataSet remplis à partir d'un DataAdapter, le système marque ces lignes comme Added (RowState = DataRowState.Added)
- Quand un utilisateur modifie les lignes d'un DataTable ou d'un DataSet remplis à partir d'un DataAdapter, le système marque ces lignes comme modified (RowState = DataRowState.Modified)

A l'appel de la méthode Update, le DataAdapter exploite les informations de statut pour apporter les modifications demandées à la base de données.

La méthode Update prend en charge tous les changements apportés aux lignes originales récupérées par la méthode Fill (ajout, suppression et modification) et l'utilisateur n'a aucune maîtrise sur l'ordre d'éxécution de ces commandes. Si l'ordre d'éxéction est important il est préférable d'exécuter la

méthode Update en indiquant le type d'opération souhaitée :

- Pour ne prendre en charge que les lignes qui ont été supprimées :

```
Réf_DataAdapter.Update(DataSet.Tables(…).Select(Nothing, Nothing,
                                                 DataViewRowState .Deleted))
```

- Pour ne prendre en charge que les lignes qui ont été ajoutées :

```
Réf_DataAdapter.Update(DataSet.Tables(…).Select(Nothing, Nothing,
                                                 DataViewRowState.Added))
```

- Pour ne prendre en charge que les lignes qui ont été modifiées :

```
Réf_DataAdapter.Update(DataSet.Tables(…).Select(Nothing, Nothing,
                                                 DataViewRowState .ModifiedCurrent))
```

Exemple : ***Travailler en local sur des recettes d'un thème donné récupérées à partir de la base de données***

Dans ce qui va suivre, la base de données Recettes aura la structure suivante :

On supposera que plusieurs utilisateurs auront à manipuler cette application de gestion des recettes et que chaque utilisateur est spécialisé dans un thème de cuisine distinct. L'utilisateur qui se connecte va donc choisir le thème sur lequel il souhaite travailler. Le programme récupérera toutes les recettes concernant ce thème et les placera dans un DataSet.

Nous avons choisi dans cet exemple d'utiliser la procédure main dans un module pour :

- Déterminer la chaine de connexion
- Paramètrer les objets DataAdapter
- Ouvrir une fenêtre FrmChoisir Thème

Soit le code associé à un module :

```
Imports System.Data.OleDb
Module ModuleGeneral
    Public cn As New OleDbConnection
    Public DataS As New DataSet
    Public VarTheme As Integer, VarPosition As Int32, VarOperation As String
    Public DA_Recettes As New OleDbDataAdapter
    Public DA_Ingrédients As New OleDbDataAdapter
    Public DA_Ingrédients_Recette As OleDbDataAdapter
    Sub main()
        cn.ConnectionString = "Provider=SQLOLEDB;Server=test;Database=Recettes;
                              Integrated security=SSPI"

        'On souhaite donner à l'utilisateur la possibilité de mettre à jour les
        'recettes ainsi que les ingrédients qui leurs sont associés. Il n'a pas le
        'droit de mettre à jour les ingrédients
        '_____________________________________________________________
        'Paramétrage du DataAdapter pour la gestion des recettes

        DA_Recettes.SelectCommand = New OleDbCommand
        DA_Recettes.SelectCommand.Connection = cn
        DA_Recettes.SelectCommand.CommandText = ("Select Numrec, NomRec,
                                    Méthodepreparation,tempspréparation from recette
                                    where codeThème=?")
```

```
        DA_Recettes.InsertCommand = New OleDbCommand
        DA_Recettes.InsertCommand.Connection = cn
        DA_Recettes.InsertCommand.CommandText = "insert into Recette values
                                                (?,?,?,?,?)"

        DA_Recettes.DeleteCommand = New OleDbCommand
        DA_Recettes.DeleteCommand.Connection = cn
        DA_Recettes.DeleteCommand.CommandText = "delete from Recette where numrec=?"

        DA_Recettes.UpdateCommand = New OleDbCommand
        DA_Recettes.UpdateCommand.Connection = cn
        DA_Recettes.UpdateCommand.CommandText = "update Recette set
                                                 NomRec=?,Méthodepreparation=?,tempsprép
                                                 aration=? where  numrec=?"

        '_____________________________________________________________
        'Paramétrage du DataAdapter pour la consultation des ingrédients

        DA_Ingrédients = New OleDbDataAdapter("Select NumIng,NomIng,
                                              PUIng,UnitéMesureIng,RSFou from
                                              Ingrédient I, Fournisseur F where
                                              I.NumFou=F.NumFou", cn)


'_____________________________________________________________________________
        'Paramétrage du DataAdapter pour la gestion des ingrédients d'une recette
 '_____________________________________________________________________________

        DA_Ingrédients_Recette = New OleDbDataAdapter("Select R.NumRec, NumIng,
                                              Qtéutilisée from Recette R,
                                              Ingrédients_Recette IR where
                                              R.NumRec=IR.NumRec and CodeThème=?", cn)

        DA_Ingrédients_Recette.InsertCommand = New OleDbCommand
        DA_Ingrédients_Recette.InsertCommand.Connection = cn
        DA_Ingrédients_Recette.InsertCommand.CommandText = "insert into
                                              Ingrédients_Recette values (?,?,?)"

        DA_Ingrédients_Recette.DeleteCommand = New OleDbCommand
        DA_Ingrédients_Recette.DeleteCommand.Connection = cn
        DA_Ingrédients_Recette.DeleteCommand.CommandText = "delete from
                                                Ingrédients_Recette where NumRec=? and
                                                NumIng=?"

        Dim FrmTheme = New FrmChoisirTheme
        Application.Run(FrmTheme)
    End Sub
End Module
```

Soit le formulaire FrmChoisirTheme :

Ce formulaire récupére, à partir de la base de données, la liste des thèmes disponibles, les stocke dans un DataTable DT_Thème et les affiche dans une zone de liste modifiable (ComboBox). Dans cette zone, on affichera les noms des thèmes mais on utilisera dans les programmes les codes thèmes.

Remarque : La colonne dont on souhaite afficher les valeurs sera attachée à la propriété DisplayMember de la zone de liste modifiable (ou de la zone de liste) et la colonne dont on souhaite manipuler les valeurs sera attachée à la propriété ValueMember de la zone de liste modifiable (ou de la zone de liste). Quand un utilisateur sélectionne une valeur dans le ComboBox, la valeur de DisplayMember pourra être récupérée par la propriété Text et Sa valeur correspondante de ValueMember sera récupérée par la propriété SelectedValue.

En cliquant sur le bouton Valider, le système stocke le code du thème séléctionné dans une vaiable globale Var_Theme (il pourra ainsi être utilisé à n'importe quel endroit de l'application créée), place les informations correspondant à ce thème dans un DataSet et affiche un formulaire FrmMenu

Dans ce DataSet seront stockées :

- La liste des recettes concernant le thème spécifié dans un DataTable DT_Recettes
- La liste des ingrédients associés à ces recettes dans un DataTable DT_Ingrédients_Recette
- La liste compléte des ingrédients dans un DataTable DT_Ingrédients (On transmettra tous les ingrédients et pas seulement ceux du thème concerné au cas où l'utilisateur souhaiterait créer de nouvelles recettes utilisant ces ingrédients).

Soit le code à associer au formulaire FrmChoisirTheme :

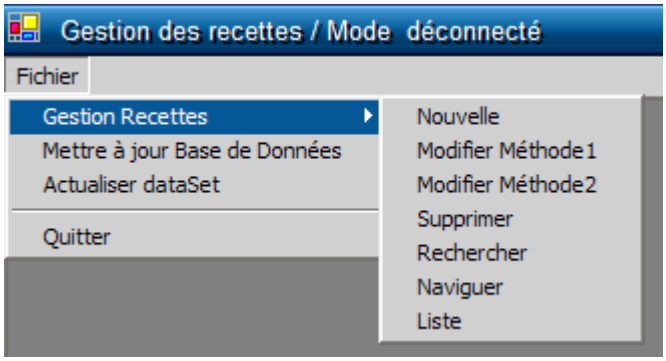
```
Imports System.Data.OleDb
Public Class FrmChoisirTheme
    Inherits System.Windows.Forms.Form
...
Private Sub FrmChoisirTheme_Load(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles MyBase.Load
        Dim DA_Thème As New OleDbDataAdapter("Select CodeThème, Nomthème from thème",
                                             cn)
        Dim DT_Thème As New DataTable
        Try
            DA_Thème.Fill(DT_Thème)
            CmbThemes.DataSource = DT_Thème
            CmbThemes.DisplayMember = DT_Thème.Columns(1).ColumnName
            CmbThemes.ValueMember = DT_Thème.Columns(0).ColumnName
        Catch ex As Exception
            MessageBox.Show("Erreur. Les données ne peuvent être chargées", "Gestion
                            recette", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
    Private Sub BtnOk_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                            Handles BtnOk.Click
        If CmbThemes.Text <> "" Then
            DataS = New DataSet
            VarTheme = CmbThemes.SelectedValue
            Dim P As OleDbParameter
            'Stocker la liste des recettes du thème choisi
            Try
                P = New OleDbParameter
                P.Value = VarTheme
                DA_Recettes.SelectCommand.Parameters.Clear()
                DA_Recettes.SelectCommand.Parameters.Add(P)
                DA_Recettes.Fill(DataS, "DT_Recette")
            Catch ex As Exception
                MessageBox.Show(ex.Message)
                MessageBox.Show("erreur sélection Recette")
            End Try

            'Stocker la liste complète des ingrédients
            Try
                DA_Ingrédients.Fill(DataS, "DT_Ingrédient")
            Catch ex As Exception
```

```
                MessageBox.Show("erreur sélection Ingrédient")
            End Try

            'Stocker la liste des ingrédients associés aux recettes du thème choisi
            Try
                P = New OleDbParameter
                P.Value = VarTheme
                DA_Ingrédients_Recette.SelectCommand.Parameters.Clear()
                DA_Ingrédients_Recette.SelectCommand.Parameters.Add(P)
                DA_Ingrédients_Recette.Fill(DataS, "DT_Ingrédients_Recette")
            Catch ex As Exception
                MessageBox.Show("erreur sélection Ingrédients des recette")
            End Try

            'détermination de la clé primaire pour le DataTable Recette
            Dim PKRec(0) As DataColumn
            PKRec(0) = DataS.Tables(0).Columns(0)
            DataS.Tables(0).PrimaryKey = PKRec

            'détermination de la clé primaire pour le DataTable Ingrédient
            Dim PKIng(0) As DataColumn
            PKIng(0) = DataS.Tables(1).Columns(0)
            DataS.Tables(1).PrimaryKey = PKIng

            'détermination de la clé primaire pour le DataTable Ingrédients_Recette
            Dim PKIngRec(1) As DataColumn
            PKIngRec(0) = DataS.Tables(2).Columns(0)
            PKIngRec(1) = DataS.Tables(2).Columns(1)
            DataS.Tables(2).PrimaryKey = PKIngRec

            'Détermination des relations
            DataS.Relations.Add("Rel_Rec_IngRec", DataS.Tables("DT_Recette").
                                Columns("NumRec"), DataS.Tables("DT_Ingrédients_
                                Recette").Columns("NumRec"))
            DataS.Relations.Add("Rel_Ing_IngRec", DataS.Tables("DT_Ingrédient").
                                Columns("NumIng"), DataS.Tables("DT_Ingrédients_
                                Recette").Columns("NumIng"))
            'ou
            'DataS.Relations.Add("Rel_Rec_IngRec", DataS.Tables(0).Columns(0),
            '                    DataS.Tables(2).Columns(0))
            'DataS.Relations.Add("Rel_Ing_IngRec", DataS.Tables(1).Columns(0),
            '                     DataS.Tables(2).Columns(1))


            Dim f As New FrmMenu
            f.WindowState = FormWindowState.Maximized
            f.Show()
        End If
    End Sub
End Class
```

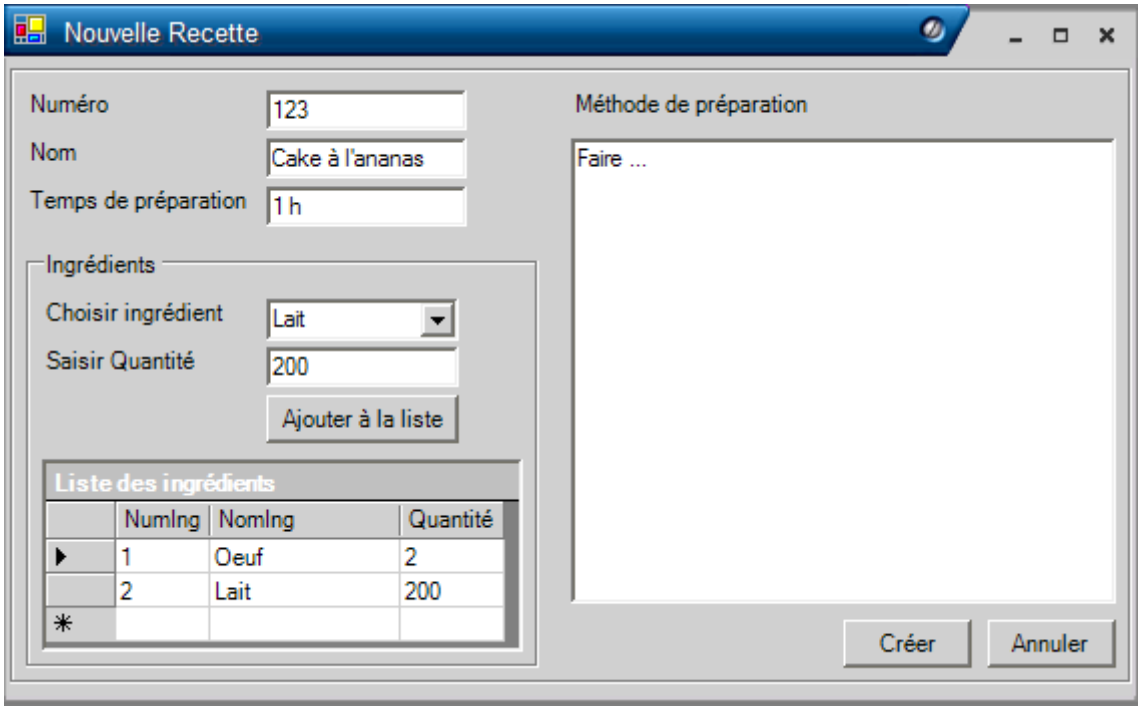
TRAVAIL A FAIRE :

- Créer une nouvelle solution
- Dans le menu Projet / Propriétés de projet Choisir de démarrer sur Sub Main
- Insérer un module dans le projet. Effectuer les déclarations nécessaires et créer la procédure main
- Insérer un formulaire FrmChoisirThème et écrire le code associé

Soit le formulaire FrmMenu :



Le sous-Menu "Nouvelle" ouvre le formulaire FrmNouvelleRecette :



Dans ce formulaire, l'utilisateur saisit les informations sur une nouvelle recette ainsi que les ingrédients associés à cette recette et appuie sur le bouton Créer pour ajouter cette recette au DataSet.

Soit le code associé au sous-Menu MnuNouvelleRecette:

```
    Private Sub MnuNouvelleRecette_Click(ByVal sender As System.Object, ByVal e As
                                         System.EventArgs) Handles MnuNouvelleRecette.Click
        Dim f As New FrmNouvelleRecette
        f.MdiParent = Me
        f.Show()
    End Sub
```

Soit le code associé à FrmNouvelleRecette

```
Public Class FrmNouvelleRecette
    Inherits System.Windows.Forms.Form
    Dim DataTtmp As New DataTable
...
    Private Sub FrmNouvelleRecette_Load(ByVal sender As System.Object, ByVal e As
                                        System.EventArgs) Handles MyBase.Load
        CmbIngrédients.DataSource = DataS.Tables(1)
        CmbIngrédients.DisplayMember = DataS.Tables(1).Columns(1).ColumnName
        CmbIngrédients.ValueMember = DataS.Tables(1).Columns(0).ColumnName
        DataTtmp.Columns.Add("NumIng", GetType(Integer))
```

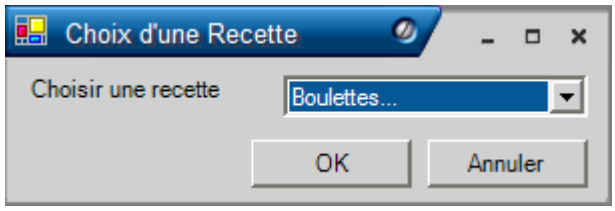
```
        DataTtmp.Columns.Add("NomIng", GetType(String))
        DataTtmp.Columns.Add("Quantité", GetType(Decimal))
        Dim t(0) As DataColumn
        t(0) = DataTtmp.Columns(0)
        DataTtmp.PrimaryKey = t
        DataGridIngrédients.DataSource = DataTtmp
        TxtNumero.Focus()
    End Sub
    Private Sub BtnAjouterIngredient_Click(ByVal sender As System.Object, ByVal e As
                                System.EventArgs) Handles BtnAjouterIngredient.Click
        Try
            Dim dr As DataRow
            dr = DataTtmp.NewRow()
            dr(0) = Val(CmbIngrédients.SelectedValue)
            dr(1) = CmbIngrédients.Text
            dr(2) = Val(TxtQté.Text)
            DataTtmp.Rows.Add(dr)
        Catch ex As Exception
            MessageBox.Show("Cet ingrédient a peut être été déjà cité pour cette
                            recette")
        End Try
    End Sub

    Private Sub BtnCréer_Click(ByVal sender As System.Object, ByVal e As
                          System.EventArgs) Handles BtnCréer.Click
        If TxtNumero.Text = "" Then
            ErrorProvider1.SetError(TxtNumero, "A remplir obligatoirement")
            TxtNumero.Focus()
            Exit Sub
        End If
        If TxtNom.Text = "" Then
            ErrorProvider1.SetError(TxtNom, "A remplir obligatoirement")
            TxtNom.Focus()
            Exit Sub
        End If
        If RichTxtMethode.Text = "" Then
            ErrorProvider1.SetError(RichTxtMethode, "A remplir obligatoirement")
            RichTxtMethode.Focus()
            Exit Sub
        End If
        If DataGridIngrédients.VisibleRowCount <= 0 Then
            ErrorProvider1.SetError(CmbIngrédients, "Au moins un ingrédient compose une
                                    recette")
            CmbIngrédients.Focus()
            Exit Sub
        End If

        Dim dr As DataRow

        Try
            ' Ajouter une recette
            dr = DataS.Tables(0).NewRow()
            dr(0) = TxtNumero.Text
            dr(1) = TxtNom.Text
            dr(2) = RichTxtMethode.Text
            dr(3) = TxtTemps.Text
            DataS.Tables(0).Rows.Add(dr)
        Catch ex As Exception
            MessageBox.Show("Erreur Ajout Recette")
            Exit Sub
        End Try

        Try
            ' Ajouter les ingrédients de la recette
            For i As Integer = 0 To DataTtmp.Rows.Count - 1
```

```
                dr = DataS.Tables(2).NewRow()
                dr(0) = TxtNumero.Text 'Le numéro de recette
                dr(1) = DataTtmp.Rows(i)(0) 'Le numéro d'ingrédient
                dr(2) = DataTtmp.Rows(i)(2) 'La quantité
                DataS.Tables(2).Rows.Add(dr)
            Next

            'ou
            'For i As Integer = 0 To DataGridIngrédients.VisibleRowCount - 1
            '    dr = DataS.Tables(2).NewRow()
            '    dr(0) = TxtNumero.Text
            '    dr(1) = DataGridIngrédients.Item(i, 0)
            '    dr(2) = DataGridIngrédients.Item(i, 2)
            '    DataS.Tables(2).Rows.Add(dr)
            'Next
        Catch ex As Exception
            MessageBox.Show("Erreur Ajout Ingrédients de la recette")
            Exit Sub
        End Try
        MessageBox.Show("Ajout effectué avec succès")
        vider()
    End Sub
    Sub vider()
        TxtNumero.Text = ""
        TxtNom.Text = ""
        TxtTemps.Text = ""
        RichTxtMethode.Text = ""
        TxtQté.Text = ""
        DataTtmp.Clear()
    End Sub

    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
End Class
```

TRAVAIL A FAIRE :

- Créer le formulaire FrmMenu
- Créer le formulaire FrmNouvelleRecette
- Ecrire les codes correspondants

Les sous-Menu "Modifier1", "Modifier2", "Supprimer" et "Rechercher" ouvrent tous le formulaire FrmChoisir :



Ce formulaire permet d'afficher l'élément à Modifier ou à rechercher et permet également de supprimer des recettes. Pour connaître l'opération à effectuer, ce formulaire utilise une variable globale VarOperation. L'utilisateur choisit une recette dans la liste et appuie sur le bouton OK, le formulaire effectue un test sur la variable VarOperation (les valeurs prises par cette variable sont affectées au niveau des sous-Menu) et effectue le traitement associé à chaque cas.
Une autre variable globale VarPosition permet de récupérer la position en cours du DataTable

Soit les codes suivants associés aux sous-Menus dans le formulaire frmMenu :

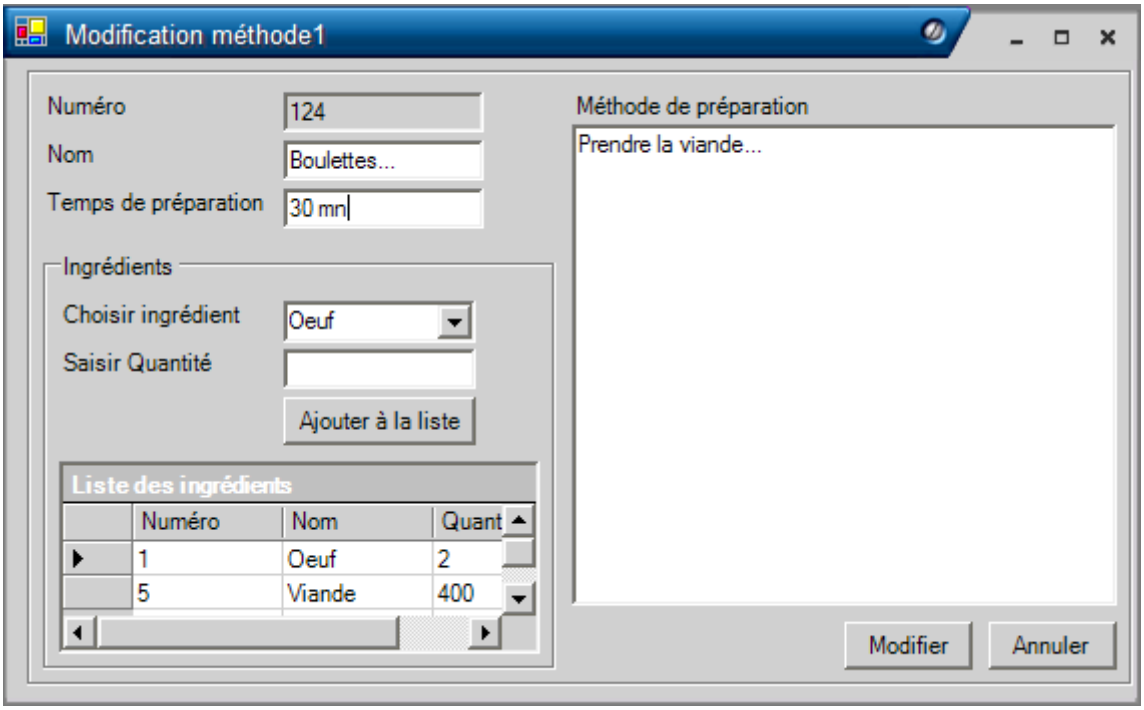
```
Imports System.Data.OleDb
Public Class FrmMenu
    Inherits System.Windows.Forms.Form
...
    Private Sub MnuModifier1Recette_Click(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles MnuModifier1Recette.Click
        Dim f As New FrmChoisir
        f.MdiParent = Me
        VarOperation = "Modifier1"
        f.Show()
    End Sub
    Private Sub MnuModifier2Recette_Click(ByVal sender As System.Object, ByVal e As
                                     System.EventArgs) Handles MnuModifier2Recette.Click
        Dim f As New FrmChoisir
        f.MdiParent = Me
        VarOperation = "Modifier2"
        f.Show()
    End Sub
    Private Sub MnuRechercherRecette_Click(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles MnuRechercherRecette.Click
        Dim f As New FrmChoisir
        f.MdiParent = Me
        VarOperation = "Rechercher"
        f.Show()
    End Sub
    Private Sub MnuSupprimerRecette_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles MnuSupprimerRecette.Click
        Dim f As New FrmChoisir
        f.MdiParent = Me
        f.Text = "Rechercher la recette à supprimer"
        VarOperation = "Supprimer"
        f.Show()
    End Sub
End class
```

Soit le code associé au formulaire frmChoisir :

```
Imports System.Data.OleDb
Public Class FrmChoisir
    Inherits System.Windows.Forms.Form
...
    Private Sub FrmRechercherRecette_Load(ByVal sender As System.Object, ByVal e As
                                      System.EventArgs) Handles MyBase.Load
        Try
            CmbRecettes.DataSource = DataS.Tables(0)
            CmbRecettes.DisplayMember = DataS.Tables(0).Columns(1).ColumnName
            CmbRecettes.ValueMember = DataS.Tables(0).Columns(0).ColumnName
        Catch ex As Exception
            MessageBox.Show("Erreur. Les données ne peuvent être chargées", "Gestion
                                 recette", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
    Private Sub BtnOk_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
                             Handles BtnOk.Click
          For i As Integer = 0 To DataS.Tables(0).Rows.Count - 1
            If DataS.Tables(0).Rows(i).RowState <> DataRowState.Deleted Then
                If DataS.Tables(0).Rows(i)(0) = CmbRecettes.SelectedValue Then
                    VarPosition = i
                End If
            End If
        Next
```

```
        If VarPosition >= 0 Then
            Select Case VarOperation
                Case "Rechercher"

                    Dim f As New FrmRechercherRecette
                    f.Show()
                Case "Modifier1"
                    Dim f As New FrmModifierRecetteMéthode1
                    f.Show()
                Case "Modifier2"
                    Dim f As New FrmModifierRecetteMéthode2
                    f.Show()
                Case "Supprimer"
                    DataS.Tables(0).Rows(VarPosition).Delete()
                    'ou DataS.Tables(0).Rows(VarPosition).Delete()
                    'Les ingrédients associés à cette recette seront automatriquement
                    'supprimés puisque une relation a été créée entre ces deux
                    'DataTable et que par défaut la suppression en cascade est activée
                    'pour la désactiver il faut créer la relation ainsi :
                    ' DataS.Relations.Add("Rel_Rec_IngRec", DataS.Tables("DT_Recette").
                    '               Columns("NumRec"), DataS.Tables("DT_Ingrédients_
                    '               Recette").Columns("NumRec"), False)
                    ' et à ce moment là la suppression ne sera pas automatique

            End Select
            Me.Hide()
        Else
            MessageBox.Show("Aucun élément disponible")
        End If
    End Sub
End Class
```

- Cas VarOperation ="Modifier1" : Le formulaire FrmModifierRecette1 s'affiche à l'écran avec les informations sur la recette à modifier :



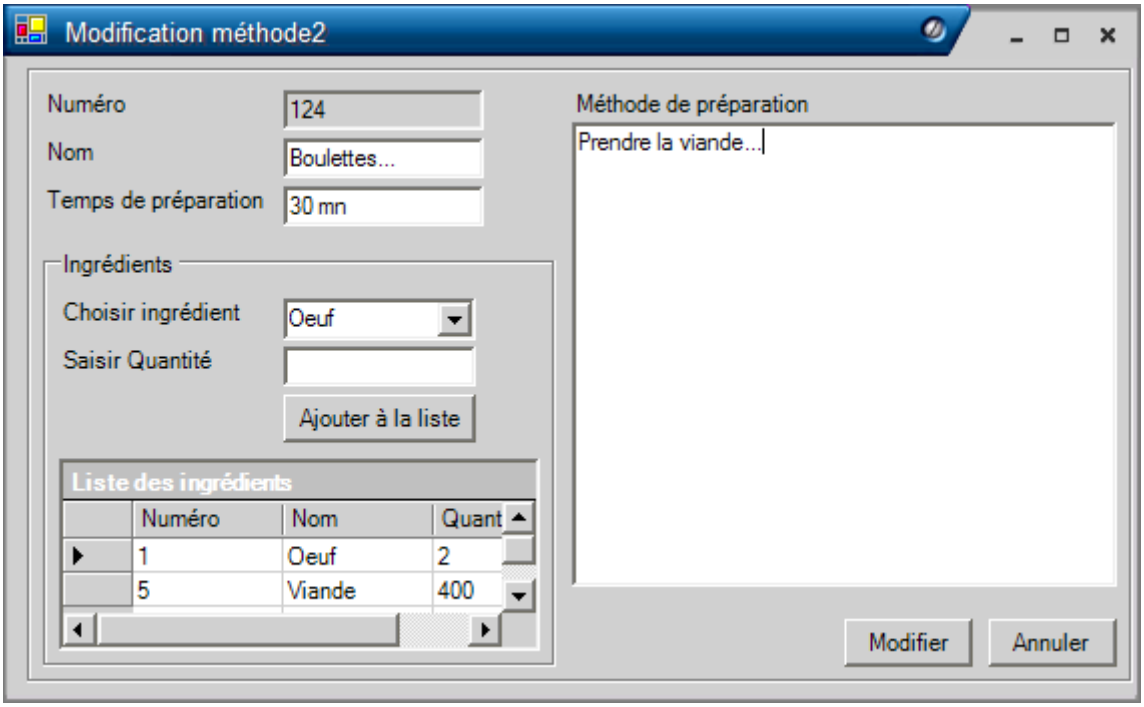
Soit le code associé à ce formulaire

```
Public Class FrmModifierRecetteMéthode1
    Inherits System.Windows.Forms.Form
...
```

```
Dim DataTtmp As New DataTable
    Private Sub FrmModifierRecetteMéthode1_Load(ByVal sender As System.Object, ByVal
                                e As System.EventArgs) Handles MyBase.Load
        If VarPosition <> -1 Then
            TxtNumero.Text = DataS.Tables(0).Rows(VarPosition)(0)
            'Dans notre exemple, on interdit la modification du numéro de recette
            TxtNumero.ReadOnly = True
            TxtNom.Text = DataS.Tables(0).Rows(VarPosition)(1)
            RichTxtMethode.Text = DataS.Tables(0).Rows(VarPosition)(2)
            TxtTemps.Text = DataS.Tables(0).Rows(VarPosition)(3)

            CmbIngrédients.DataSource = DataS.Tables(1)
            CmbIngrédients.DisplayMember = DataS.Tables(1).Columns(1).ColumnName
            CmbIngrédients.ValueMember = DataS.Tables(1).Columns(0).ColumnName

            DataTtmp.Columns.Add("Numéro", GetType(Integer))
            DataTtmp.Columns.Add("Nom", GetType(String))
            DataTtmp.Columns.Add("Quantité", GetType(Decimal))

            For Each DataR_Ing_Rec As DataRow In DataS.Tables(2).Rows
                If DataR_Ing_Rec.RowState <> DataRowState.Deleted Then
                    If DataR_Ing_Rec(0) = DataS.Tables(0).Rows(VarPosition)(0) Then
                        Dim DataR2 As DataRow
                        DataR2 = DataTtmp.NewRow()
                        DataR2(0) = DataR_Ing_Rec(1)
                        For Each DataR_Ing As DataRow In DataS.Tables(1).Rows
                            If DataR_Ing(0) = DataR_Ing_Rec(1) Then
                                DataR2(1) = DataR_Ing(1)
                                Exit For
                            End If
                        Next
                        DataR2(2) = DataR_Ing_Rec(2)
                        DataTtmp.Rows.Add(DataR2)
                    End If
                End If
            Next
            DataGridIngrédients.DataSource = DataTtmp
        Else
            MessageBox.Show("erreur chargement")
        End If
    End Sub
    Private Sub BtnAjouterIngredient_Click(ByVal sender As System.Object, ByVal e As
                        System.EventArgs) Handles BtnAjouterIngredient.Click
        Try
            Dim dr As DataRow
            dr = DataTtmp.NewRow()
            dr(0) = Val(CmbIngrédients.SelectedValue)
            dr(1) = CmbIngrédients.Text
            dr(2) = Val(TxtQté.Text)
            DataTtmp.Rows.Add(dr)
        Catch ex As Exception
            MessageBox.Show("Cet ingrédient a peut être été déjà cité pour cette
                            recette")
        End Try
    End Sub
    Private Sub BtnModifier_Click(ByVal sender As System.Object, ByVal e As
                            System.EventArgs) Handles BtnModifier.Click
        If TxtNom.Text = "" Then
            ErrorProvider1.SetError(TxtNom, "A remplir obligatoirement")
            RichTxtMethode.Focus()
            Exit Sub
        End If
        If RichTxtMethode.Text = "" Then
            ErrorProvider1.SetError(RichTxtMethode, "A remplir obligatoirement")
            RichTxtMethode.Focus()
```

```
            Exit Sub
        End If
        Try
            ' Modifier une ligne du DataSet
            DataS.Tables(0).Rows(VarPosition).BeginEdit()
            DataS.Tables(0).Rows(VarPosition)(1) = TxtNom.Text
            DataS.Tables(0).Rows(VarPosition)(2) = RichTxtMethode.Text
            DataS.Tables(0).Rows(VarPosition)(3) = TxtTemps.Text
            DataS.Tables(0).Rows(VarPosition).EndEdit()

            Dim j As Int16 = DataS.Tables(2).Rows.Count - 1
            For i As Integer = 0 To j
                If DataS.Tables(2).Rows(i).RowState <> DataRowState.Deleted Then
                    If DataS.Tables(2).Rows(i)(0) = Val(TxtNumero.Text) Then
                        DataS.Tables(2).Rows(i).Delete()
                        j = DataS.Tables(2).Rows.Count - 1
                        i = i - 1
                    End If
                End If
                If i = j Then Exit For
            Next
            For Each DataRTmp As DataRow In DataTtmp.Rows
                Dim DataRIngRec As DataRow
                DataRIngRec = DataS.Tables(2).NewRow
                DataRIngRec(0) = Val(TxtNumero.Text)
                DataRIngRec(1) = DataRTmp(0)
                DataRIngRec(2) = DataRTmp(2)
                DataS.Tables(2).Rows.Add(DataRIngRec)
            Next
        Catch ex As Exception
            MessageBox.Show("Erreur Modification Recette")
            Exit Sub
        End Try
        MessageBox.Show("Modification effectuée avec succès")
    End Sub

    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
End Class
```

- Cas VarOperation ="Modifier2" : Le formulaire FrmModifierRecette2 s'affiche à l'écran avec les informations sur la recette à modifier (ce formulaire offre une deuxième méthode pour opérer des modifications sur les données)

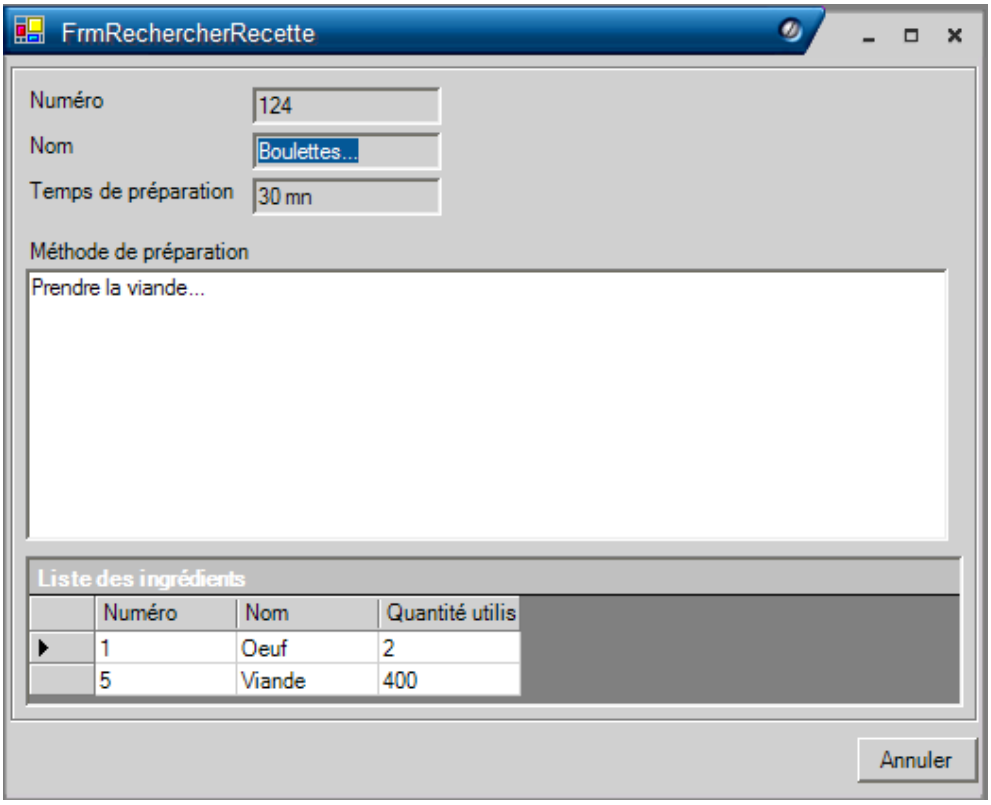


Soit le code associé à ce formulaire

```
Public Class FrmModifierRecetteMéthode2
    Inherits System.Windows.Forms.Form
    Dim DataTtmp As New DataTable
...
    Private Sub FrmModifierRecetteMéthode2_Load(ByVal sender As System.Object, ByVal
                                    e As System.EventArgs) Handles MyBase.Load
        Try
            Me.BindingContext(DataS.Tables(0)).Position() = VarPosition
            TxtNumero.DataBindings.Add("text", DataS.Tables(0), DataS.Tables(0).
                                    Columns(0).ColumnName)
            TxtNumero.ReadOnly = True
            TxtNom.DataBindings.Add("text", DataS.Tables(0), DataS.Tables(0).
                                    Columns(1).ColumnName)
            RichTxtMethode.DataBindings.Add("text", DataS.Tables(0), DataS.
                                    Tables(0).Columns(2).ColumnName)
            TxtTemps.DataBindings.Add("text", DataS.Tables(0), DataS.Tables(0).
                                    Columns(3).ColumnName)

            CmbIngrédients.DataSource = DataS.Tables(1)
            CmbIngrédients.DisplayMember = DataS.Tables(1).Columns(1).ColumnName
            CmbIngrédients.ValueMember = DataS.Tables(1).Columns(0).ColumnName


            DataTtmp.Columns.Add("Numéro", GetType(Integer))
            DataTtmp.Columns.Add("Nom", GetType(String))
            DataTtmp.Columns.Add("Quantité utilisée", GetType(Decimal))

            For Each DataR_Ing_Rec As DataRow In DataS.Tables(2).Rows
                If DataR_Ing_Rec(0) = DataS.Tables(0).Rows(VarPosition)(0) Then
                    Dim DataR2 As DataRow
                    DataR2 = DataTtmp.NewRow()
                    DataR2(0) = DataR_Ing_Rec(1)
                    For Each DataR_Ing As DataRow In DataS.Tables(1).Rows
                        If DataR_Ing(0) = DataR_Ing_Rec(1) Then
                            DataR2(1) = DataR_Ing(1)
                            Exit For
                        End If
                    Next
```

```
                    DataR2(2) = DataR_Ing_Rec(2)
                    DataTtmp.Rows.Add(DataR2)
                End If
            Next
            DataGridIngrédients.DataSource = DataTtmp
        Catch ex As Exception
            MessageBox.Show("erreur chargement")
        End Try

    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
    Private Sub BtnAjouterIngredient_Click(ByVal sender As System.Object, ByVal e As
                           System.EventArgs) Handles BtnAjouterIngredient.Click
        Try
            Dim dr As DataRow
            dr = DataTtmp.NewRow()
            dr(0) = Val(CmbIngrédients.SelectedValue)
            dr(1) = CmbIngrédients.Text
            dr(2) = Val(TxtQté.Text)
            DataTtmp.Rows.Add(dr)
        Catch ex As Exception
            MessageBox.Show("Cet ingrédient a peut être été déjà cité pour cette
                             recette")
        End Try
    End Sub

    Private Sub BtnModifier_Click(ByVal sender As System.Object, ByVal e As
                                  System.EventArgs) Handles BtnModifier.Click
        Try
            DataS.Tables(0).rows(varPosition).BeginEdit()
            DataS.Tables(0).rows(varPosition).EndEdit()
            Dim j As Int16 = DataS.Tables(2).Rows.Count - 1
            For i As Integer = 0 To j
                If DataS.Tables(2).Rows(i).RowState <> DataRowState.Deleted Then
                    If DataS.Tables(2).Rows(i)(0) = Val(TxtNumero.Text) Then
                        DataS.Tables(2).Rows(i).Delete()
                        j = DataS.Tables(2).Rows.Count - 1
                        i = i - 1
                    End If
                End If
                If i = j Then Exit For
            Next

            For Each DataRTmp As DataRow In DataTtmp.Rows
                Dim DataRIngRec As DataRow
                DataRIngRec = DataS.Tables(2).NewRow
                DataRIngRec(0) = Val(TxtNumero.Text)
                DataRIngRec(1) = DataRTmp(0)
                DataRIngRec(2) = DataRTmp(2)
                DataS.Tables(2).Rows.Add(DataRIngRec)
            Next
        Catch ex As Exception
            MessageBox.Show("Erreur Modification Recette")
            Exit Sub
        End Try
        MessageBox.Show("Modification effectuée avec succès")
    End Sub
End Class
```

- Cas VarOperation ="Rechercher", le formulaire FrmRechercherRecette affiche les informations sur la recette choisie



Soit le code associé à ce formulaire

```
Public Class FrmRechercherRecette
    Inherits System.Windows.Forms.Form
...
    Private Sub FrmRechercherRecette_Load(ByVal sender As System.Object, ByVal e As
                            System.EventArgs) Handles MyBase.Load
        Try
            TxtNumero.Text = DataS.Tables(0).Rows(VarPosition)(0)
            TxtNom.Text = DataS.Tables(0).Rows(VarPosition)(1)
            RichTxtMethode.Text = DataS.Tables(0).Rows(VarPosition)(2)
            TxtTemps.Text = DataS.Tables(0).Rows(VarPosition)(3)

            'Affichage liste des ingrédients associés
            Dim DataTtmp As New DataTable
            DataTtmp.Columns.Add("Numéro", GetType(Integer))
            DataTtmp.Columns.Add("Nom", GetType(String))
            DataTtmp.Columns.Add("Quantité utilisée", GetType(Decimal))
            Dim NumRec As Integer = DataS.Tables(0).Rows(VarPosition)(0)
            For Each DataR_Ing_Rec As DataRow In DataS.Tables(2).Rows
                'Puisque la suppression se fait en cascade, à la suppression d'une
                'recette, les ingrédients associés sont automatiquement supprimés
                'et le système marque leur état comme deleted
                If DataR_Ing_Rec.RowState <> DataRowState.Deleted Then
                    If DataR_Ing_Rec(0) = NumRec Then
                        Dim DataR2 As DataRow
                        DataR2 = DataTtmp.NewRow()
                        DataR2(0) = DataR_Ing_Rec(1)
                        For Each DataR_Ing As DataRow In DataS.Tables(1).Rows
                            If DataR_Ing(0) = DataR_Ing_Rec(1) Then
                                DataR2(1) = DataR_Ing(1)
                                Exit For
                            End If
                        Next
                        DataR2(2) = DataR_Ing_Rec(2)
                        DataTtmp.Rows.Add(DataR2)
                    End If
                End If
```

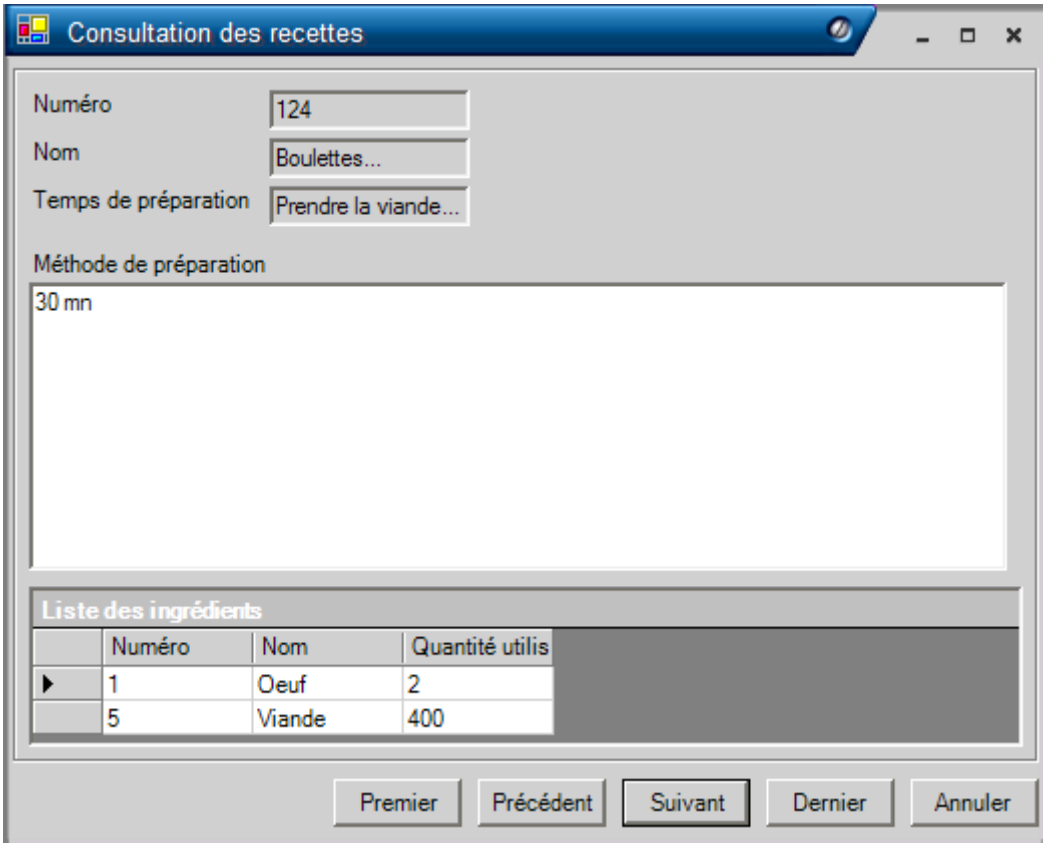
```
            Next
            DataGridIngrédients.DataSource = DataTtmp

            'interdire écriture
            TxtNumero.ReadOnly = True
            TxtNom.ReadOnly = True
            TxtTemps.ReadOnly = True
            RichTxtMethode.ReadOnly = True
            DataGridIngrédients.ReadOnly = True
        Catch ex As Exception
            MessageBox.Show("Erreur Chargement")
        End Try
    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub

End Class
```

- Si VarOperation ="Supprimer", la recette choisie est supprimée. Si l'option de suppression en cascade est activée, la recette ainsi que les ingrédients qui lui sont associée sont supprimés. Si par contre cette option n'est pas activée et que l'utilisateur demande à supprimer une recette qui a des ingrédients associés, la suppression échoue.

TRAVAIL A FAIRE :

- Déclarer les variables VarOperation et VarPosition dans un module
- Créer le formulaire FrmMenu
- Créer les formulaires FrmChoisir, FrmModifierRecetteMethode1, FrmModifierRecetteMethode2 et FrmRechercherRecette
- Associer les programmes aux sous-Menus
- Ecrire les programmes correspondants à chaque formulaire

Le sous-Menu "Naviguer" ouvre le formulaire FrmNaviguerEntreRecettes :



Ce formulaire permet de parcourir de manière séquentielle l'ensemble des recettes avec pour chacune les ingrédients correspondants.

Soit le code associé au sous-Menu MnuNouvelleRecette:

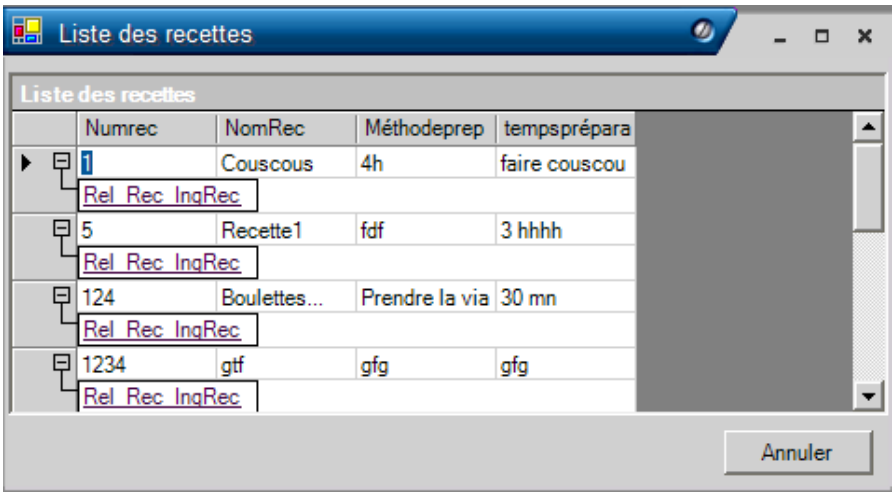
```
Private Sub MnuNaviguerEntrerecettes_Click(ByVal sender As System.Object, ByVal e As
                            System.EventArgs) Handles MnuNaviguerEntrerecettes.Click
        Dim f As New FrmRechercheSéquentielleRecette
        f.MdiParent = Me
        f.Show()
    End Sub
```

Soit le code associé à ce formulaire :

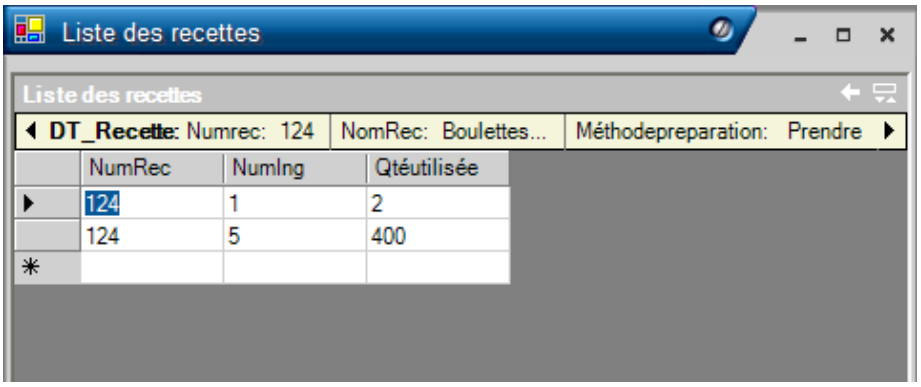
```
Public Class FrmNaviguerEntreRecettes
    Inherits System.Windows.Forms.Form
    Dim p As Integer
    Dim DataTtmp As New DataTable
...
    Private Sub FrmNaviguerEntreRecettes_Load(ByVal sender As System.Object, ByVal e As
                                          System.EventArgs) Handles MyBase.Load
        p = 0
        DataTtmp.Columns.Add("Numéro", GetType(Integer))
        DataTtmp.Columns.Add("Nom", GetType(String))
        DataTtmp.Columns.Add("Quantité utilisée", GetType(Decimal))
        'interdire écriture
        TxtNumero.ReadOnly = True
        TxtNom.ReadOnly = True
        TxtTemps.ReadOnly = True
        RichTxtMethode.ReadOnly = True
        DataGridIngrédients.ReadOnly = True
        afficher(p)

    End Sub
    Private Sub BtnAnnuler_Click(ByVal sender As System.Object, ByVal e As
```

```
                                System.EventArgs) Handles BtnAnnuler.Click
        Me.Hide()
    End Sub
    Sub afficher(ByVal n As Integer)
        TxtNumero.Text = DataS.Tables(0).Rows(n)(0)
        TxtNom.Text = DataS.Tables(0).Rows(n)(1)
        TxtTemps.Text = DataS.Tables(0).Rows(n)(2)
        RichTxtMethode.Text = DataS.Tables(0).Rows(n)(3)
        Dim numrec As Integer = DataS.Tables(0).Rows(n)(0)

        DataTtmp.Clear()
        For Each DataR_Ing_Rec As DataRow In DataS.Tables(2).Rows
            If DataR_Ing_Rec.RowState <> DataRowState.Deleted Then
                If DataR_Ing_Rec(0) = numrec Then
                    Dim DataR2 As DataRow
                    DataR2 = DataTtmp.NewRow()
                    DataR2(0) = DataR_Ing_Rec(1)
                    For Each DataR_Ing As DataRow In DataS.Tables(1).Rows
                        If DataR_Ing(0) = DataR_Ing_Rec(1) Then
                            DataR2(1) = DataR_Ing(1)
                            Exit For
                        End If
                    Next
                    DataR2(2) = DataR_Ing_Rec(2)
                    DataTtmp.Rows.Add(DataR2)
                End If
            End If
        Next
        DataGridIngrédients.DataSource = DataTtmp
    End Sub
    Private Sub BtnPremier_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnPremier.Click
        p = 0
        If DataS.Tables(0).Rows(p).RowState <> DataRowState.Deleted Then
            afficher(p)
        End If
    End Sub
    Private Sub BtnPrecedent_Click(ByVal sender As System.Object, ByVal e As
                                   System.EventArgs) Handles BtnPrecedent.Click
        If p > 0 Then
            p = p - 1
            While DataS.Tables(0).Rows(p).RowState = DataRowState.Deleted
                p = p - 1
            End While
            If p >= 0 Then afficher(p)
        End If
    End Sub
    Private Sub BtnSuivant_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnSuivant.Click
        If p < DataS.Tables(0).Rows.Count - 1 Then
            p = p + 1
            While DataS.Tables(0).Rows(p).RowState = DataRowState.Deleted
                p = p + 1
            End While
            If p <= DataS.Tables(0).Rows.Count - 1 Then afficher(p)
        End If
    End Sub
    Private Sub BtnDernier_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles BtnDernier.Click
        p = DataS.Tables(0).Rows.Count - 1
        If DataS.Tables(0).Rows(p).RowState <> DataRowState.Deleted Then
            afficher(p)
        End If
    End Sub
End Class
```

Le sous-Menu "Liste" ouvre le formulaire FrmListeRecettes :



Ce formulaire affiche dans un DataGrid la liste des recettes et de leurs ingrédients. En cliquant sur la relation correspondant à une recette, le système affichera les ingrédients correspondants (ce procédé est automatique) :



Soit le code associé au sous-Menu FrmListeRecettes :

```
    Private Sub MnuListeRecettes_Click(ByVal sender As System.Object, ByVal e As
                                System.EventArgs) Handles MnuListeRecettes.Click
        Dim f As New FrmListeRecettes
        f.MdiParent = Me
        f.Show()
    End Sub
```

Soit le code associé à ce formulaire :

```
Public Class FrmListeRecettes
    Inherits System.Windows.Forms.Form
    Dim dt As New DataTable
...
    Private Sub FrmListeRecettes_Load(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles MyBase.Load
        DataGridRecettes.DataSource = DataS
        DataGridRecettes.DataMember = DataS.Tables(0).TableName
        DataGridRecettes.Expand(-1)
        DataGridRecettes.ParentRowsBackColor = Color.Beige()
    End Sub
```

Pour Mettre à jour la base de données, soit le code associé au Menu MnuMiseAJourBD :

```
    Private Sub MnuMiseAJourBD_Click(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles MnuMiseAJourBD.Click
        Try
```

```
        Dim P1, P2, P3, P4, P5 As OleDbParameter

        '_____________________________
        'DA_Recettes

        'Spécifier les paramètres de la commande delete pour le DataAdapter
        'DataARecette
        P1 = New OleDbParameter("Param_Numéro", OleDbType.Integer, 50, DataS.
                                Tables(0).Columns(0).ColumnName)
        DA_Recettes.DeleteCommand.Parameters.Clear()
        DA_Recettes.DeleteCommand.Parameters.Add(P1)

        'Spécifier les paramètres de la commande update pour le DataAdapter
        'DataARecette
        P1 = New OleDbParameter("Param_NomRec", OleDbType.VarChar, 50, DataS.
                                Tables(0).Columns(1).ColumnName)
        P2 = New OleDbParameter("Param_Méthode", OleDbType.LongVarChar, 500, DataS.
                                Tables(0).Columns(2).ColumnName)
        P3 = New OleDbParameter("Param_Temps", OleDbType.VarChar, 50, DataS.
                                Tables(0).Columns(3).ColumnName)
        P4 = New OleDbParameter("Param_Numéro", OleDbType.Integer, 4, DataS.
                                Tables(0).Columns(0).ColumnName)
        DA_Recettes.UpdateCommand.Parameters.Clear()
        DA_Recettes.UpdateCommand.Parameters.Add(P1)
        DA_Recettes.UpdateCommand.Parameters.Add(P2)
        DA_Recettes.UpdateCommand.Parameters.Add(P3)
        DA_Recettes.UpdateCommand.Parameters.Add(P4)

        'Spécifier les paramètres de la commande insert pour le DataAdapter
        'DataARecette
        P1 = New OleDbParameter("Param_Numéro", OleDbType.Integer, 4, DataS.
                                Tables(0).Columns(0).ColumnName)
        P2 = New OleDbParameter("Param_NomRec", OleDbType.VarChar, 50, DataS.
                                Tables(0).Columns(1).ColumnName)
        P3 = New OleDbParameter("Param_Méthode", OleDbType.LongVarChar, 500, DataS.
                                Tables(0).Columns(2).ColumnName)
        P4 = New OleDbParameter("Param_Temps", OleDbType.VarChar, 50, DataS.
                                Tables(0).Columns(3).ColumnName)
        P5 = New OleDbParameter
        P5.Value = VarTheme

        DA_Recettes.InsertCommand.Parameters.Clear()
        DA_Recettes.InsertCommand.Parameters.Add(P1)
        DA_Recettes.InsertCommand.Parameters.Add(P2)
        DA_Recettes.InsertCommand.Parameters.Add(P3)
        DA_Recettes.InsertCommand.Parameters.Add(P4)
        DA_Recettes.InsertCommand.Parameters.Add(P5)

        '_____________________________
        'DA_Ingrédients_Recette

        'Spécifier les paramètres de la commande delete pour le DataAdapter
        'DataARecetteIngrédients
        P1 = New OleDbParameter("Param_NumRec", OleDbType.Integer, 4,
                                DataS.Tables(2).Columns(0).ColumnName)
        P2 = New OleDbParameter("Param_NumIng", OleDbType.Integer, 4,
                                DataS.Tables(2).Columns(1).ColumnName)
        DA_Ingrédients_Recette.DeleteCommand.Parameters.Clear()
        DA_Ingrédients_Recette.DeleteCommand.Parameters.Add(P1)
        DA_Ingrédients_Recette.DeleteCommand.Parameters.Add(P2)

        'Spécifier les paramètres de la commande insert pour le DataAdapter
        'DataARecetteIngrédients
        P1 = New OleDbParameter("Param_NumRec", OleDbType.Integer, 4,
                                DataS.Tables(2).Columns(0).ColumnName)
```

```
            P2 = New OleDbParameter("Param_NumIng", OleDbType.Integer, 4,
                                    DataS.Tables(2).Columns(1).ColumnName)
            P3 = New OleDbParameter("Param_Qté", OleDbType.Decimal, 13,
                                    DataS.Tables(2).Columns(2).ColumnName)
            DA_Ingrédients_Recette.InsertCommand.Parameters.Clear()
            DA_Ingrédients_Recette.InsertCommand.Parameters.Add(P1)
            DA_Ingrédients_Recette.InsertCommand.Parameters.Add(P2)
            DA_Ingrédients_Recette.InsertCommand.Parameters.Add(P3)

            'Mise à jour de la base de données
            'L'ordre dans lequel les modifications apportées dans le DataSet
            ' sont transmises à la source de données est important dans notre cas
            'La proriété DataViewRowState retourne l'état en cours d'un objet dataRow
            'et permet de gérer l'ordre des opérations de mise à jour à effectuer
            'dans la base de données
            DA_Ingrédients_Recette.Update(DataS.Tables(2).Select(Nothing, Nothing,
                                          DataViewRowState.Deleted))
            DA_Recettes.Update(DataS.Tables(0).Select(Nothing, Nothing,
                                          DataViewRowState.Deleted))
            DA_Recettes.Update(DataS.Tables(0).Select(Nothing, Nothing,
                                          DataViewRowState.Added))
            DA_Ingrédients_Recette.Update(DataS.Tables(2).Select(Nothing, Nothing,
                                          DataViewRowState.Added))
            DA_Recettes.Update(DataS.Tables(0).Select(Nothing, Nothing,
                                          DataViewRowState.ModifiedCurrent))

        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End Sub
```

Pour récupére d'éventuelle modifications qui ont été apportées à la base de données depuis le dernier chargement avec la méthode Fill:

```
    Private Sub MnuActualiserDataset_Click(ByVal sender As System.Object, ByVal e As
                            System.EventArgs) Handles MnuActualiserDataset.Click
        Dim P As OleDbParameter
        Try
            P = New OleDbParameter
            P.Value = VarTheme
            DA_Recettes.SelectCommand.Parameters.Clear()
            DA_Recettes.SelectCommand.Parameters.Add(P)
            DA_Recettes.Fill(DataS, "DT_Recette")

            DA_Ingrédients.Fill(DataS, "DT_Ingrédient")

            P = New OleDbParameter
            P.Value = VarTheme
            DA_Ingrédients_Recette.SelectCommand.Parameters.Clear()
            DA_Ingrédients_Recette.SelectCommand.Parameters.Add(P)
            DA_Ingrédients_Recette.Fill(DataS, "DT_Ingrédients_Recette")
        Catch ex As Exception
            MessageBox.Show("Erreur Chargement")
        End Try
    End Sub
```

Soit le code associé au sous-Menu FrmListeRecettes :

```
    Private Sub MnuListeRecettes_Click(ByVal sender As System.Object, ByVal e As
                              System.EventArgs) Handles MnuListeRecettes.Click
        Dim f As New FrmListeRecettes
        f.MdiParent = Me
```

```
        f.Show()
    End Sub
```

Soit le code associé à ce formulaire :

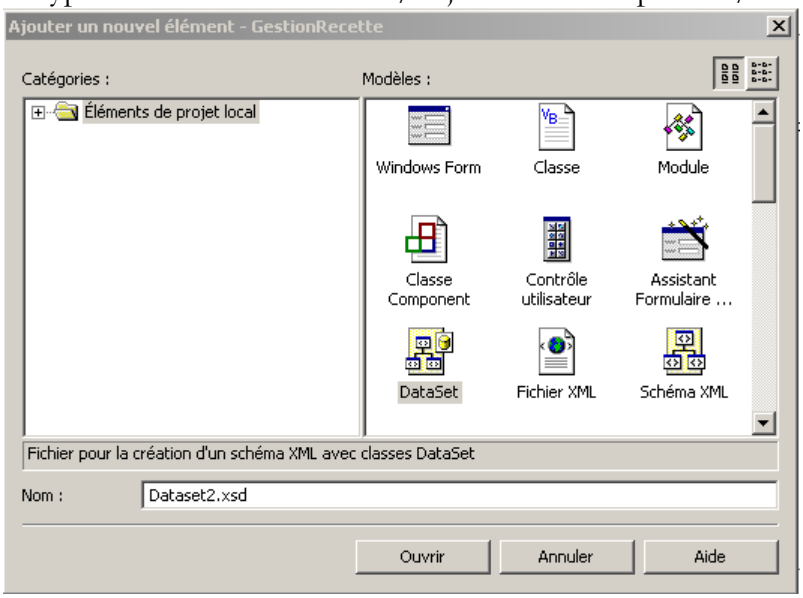
```
Public Class FrmListeRecettes
    Inherits System.Windows.Forms.Form
    Dim dt As New DataTable
...
    Private Sub FrmListeRecettes_Load(ByVal sender As System.Object, ByVal e As
                                      System.EventArgs) Handles MyBase.Load
        DataGridRecettes.DataSource = DataS
        DataGridRecettes.DataMember = DataS.Tables(0).TableName
        DataGridRecettes.Expand(-1)
        DataGridRecettes.ParentRowsBackColor = Color.Beige()
    End Sub
```

LES DATASET TYPES

Un DataSet typé est une classe dérivée d'un DataSet. Il hérite de l'ensemble des méthodes, événements et propriétés d'un DataSet mais permet d'accéder à des tables et à des colonnes par leur nom au lieu d'utiliser les méthodes associées à des collections. Ceci permet d'améliorer la lisibilité du code, de bénéficier des fonctions d'assistance à la frappe et de détecter les erreurs d'incompatibilité de types au moment de la compilation.

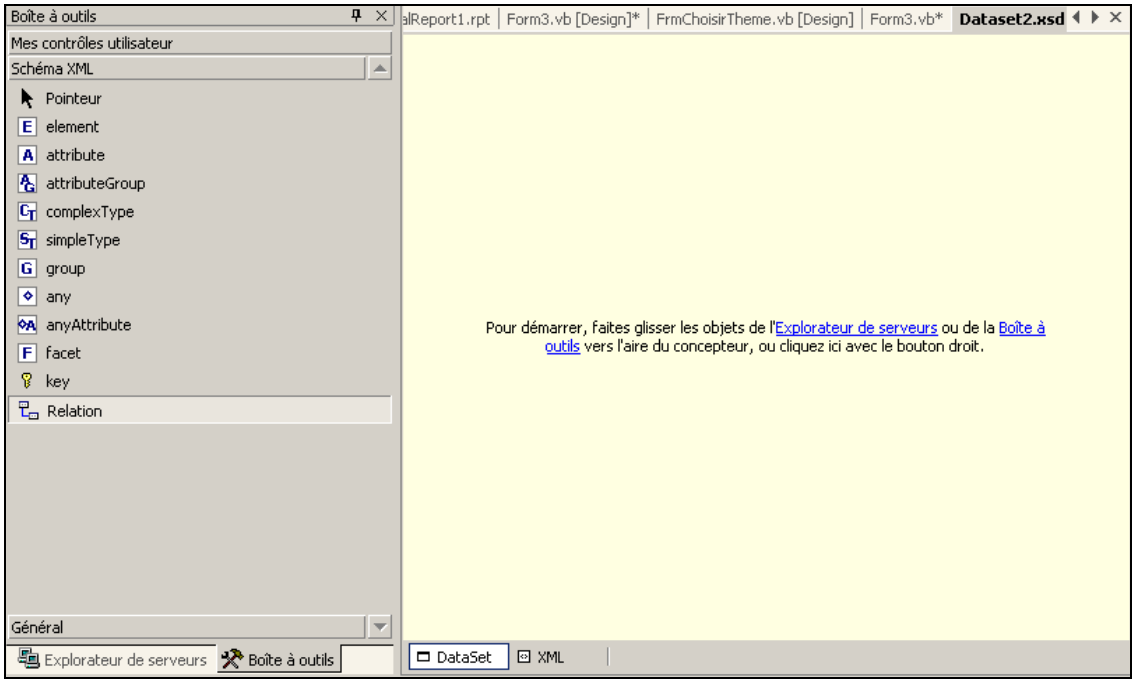
Création

Pour créer un DataSet typé aller au Menu Fichier / Ajouter un composant / DataSet :



Remarque : Le .xsd contient toutes les informations concernant le DataSet créé et va contenir les modifications qui seront apportées (XML).

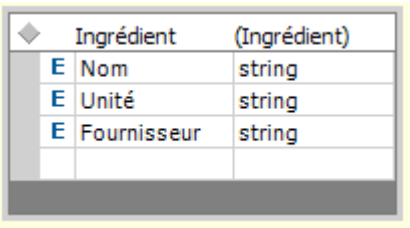
A l'insertion d'un DataSet typé l'écran suivant apparaît :



Il est possible de commencer à ajouter des DataTable (Element) à l'aide de la boite à outils 'Schéma XML' ou à l'aide de l'explorateur de serveurs. La solution de l'explorateur de serveurs est plus rapide et plus facile. On fait glisser les éléments à partir d'une connexion à une source de données préalablement paramétrée, **on opère les modifications souhaitées (ajout de champs, suppression de champs, création de relations...)**

Remarque : Les DataTables sont dit Element et les colonnes sont dits attributs

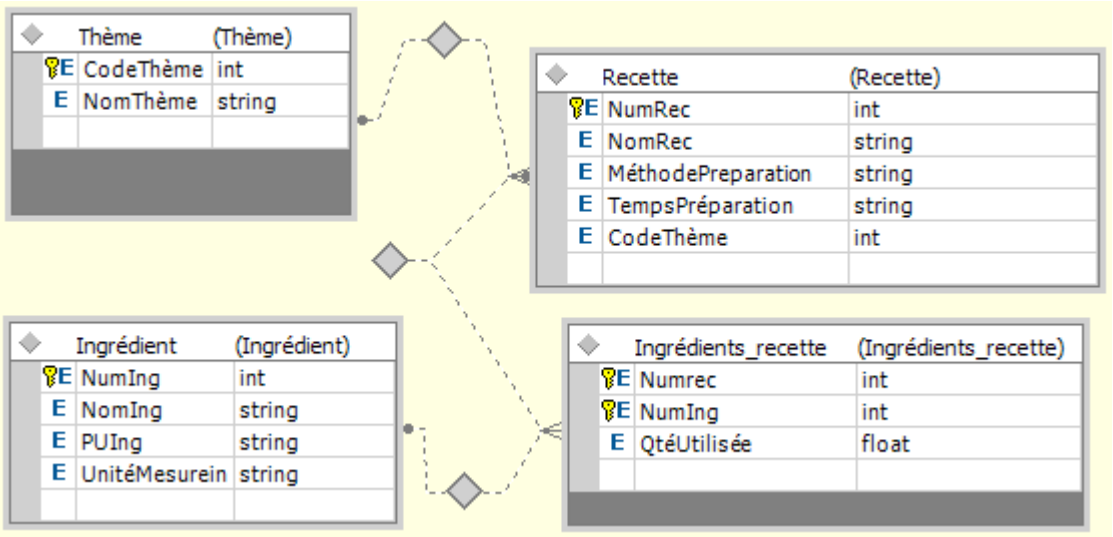
Exemple : ***Créer un DataSetTypé sans utiliser l'explorateur de Serveur***
Le DataSet créé va s'appeler DataSetIngrédient. Il va contenir un seul DataTable nommé Ingrédient contenant les champs qu'on souhaite afficher (nom ingrédient, unité de mesure et raison sociale du fournisseur).
Ce DataSet aura l'aspect suivant :



TRAVAIL A FAIRE :

- **Créer ce DataSet Typé**

Exemple : ***Créer un DataSetTypé e utilisant l'explorateur de Serveur***
Le DataSet créé va s'appeler DataSetRecette. Il va contenir quatre DataTables (Recette, Ingrédients_Recette, Ingrédient et thème).
Ce DataSet aura l'aspect suivant :



Remarque :

- Un dataTable concernant le thème sera utilisé pour nous permettre d'afficher le nom du thème sur l'état de sortie
- Le numéro de fournisseur a été enlevé du DataTable Ingrédient parce qu'on ne souhaite pas l'afficher

TRAVAIL A FAIRE :

- **Créer ce DataSet Typé**

Utilisation

Pour utiliser un DataSet typé, il faut créer une instance du DataSet typé en question. Une fois instancié, il sera traité comme n'importe quel autre DataSet. On pourra le remplir soit en utilisant des objets DataRow, soit en utilisant la méthode Fill.
Remarque : La méthode Update peut également être utilisée pour permettre à l'objet DataAdapter de mettre à jour la base de données en fonction du nouveau contenu du dataSet Typé.

- Pour créer une instance d'un DataSet Typé:

```
Dim|Public Réf_DataSetTypé as new Nom_DataSetTypé
```

- Pour Remplir un DataSet Typé
  - En utilisant des objets DataRow

```
Dim Réf_DataRow as DataRow
Réf_DataRow=Réf_DataSet.Tables(index|Nom_DataTable).NewRow()
Réf_DataRow(0)=…
Réf_DataRow(1)=…
Réf_DataSet.Tables(index|Nom_DataTable).rows.add(Réf_DataRow)
```

  - En utilisant un objet DataAdapter auquel on a associé une requête de sélection soit au moment de l'instanciation soit en utilisant un objet SelectCommand

```
Réf_DataAdapter.Fill(Réf_DataSetTypé, NomDataTable)
```

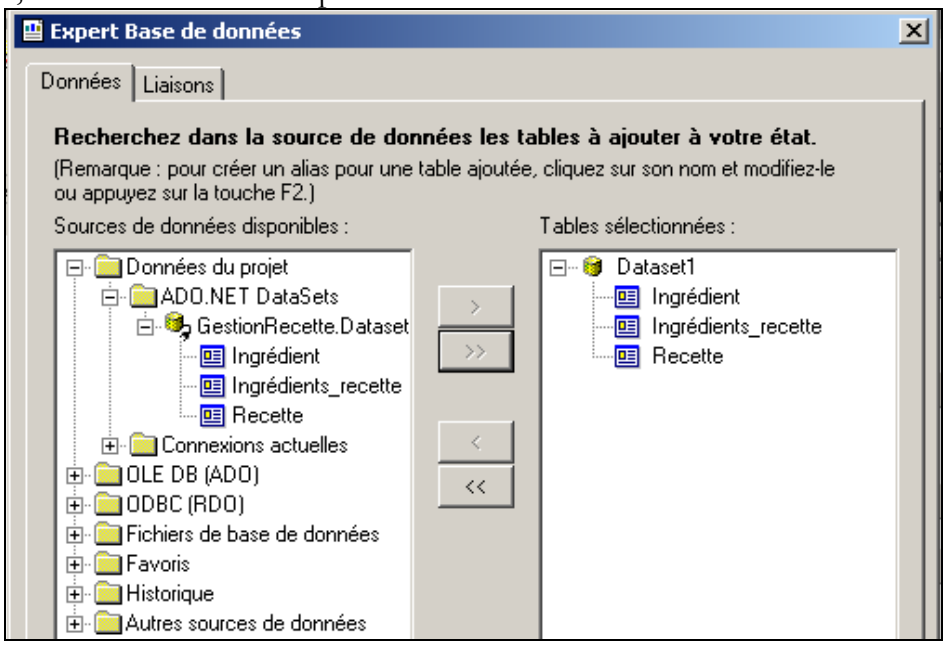
Remarque : Pour faire référence à un DataTable contenu dans un DataSet typé , peut utiliser la syntaxe suivante : `Dim Réf_DataTable as new  NomDataSetTypé.Nom_DataTable`

ETAT DE SORTIE BASES SUR DES DATASET

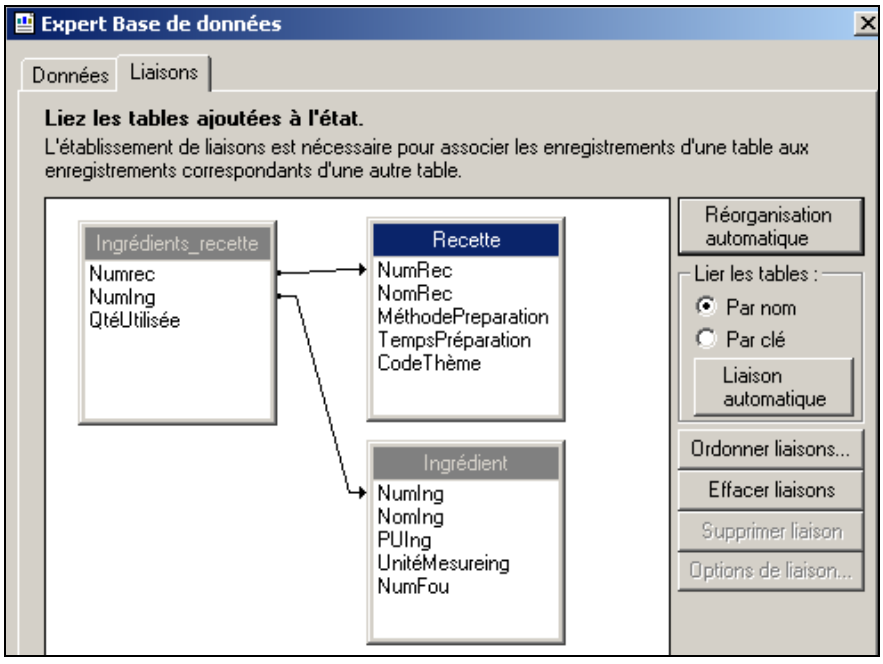
Source

Les objets DataTable et DataSet sont les seules structures complètement indépendantes de la source de données. Leur utilisation comme source de données d'un état Crystal Reports le détache de la connexion et le rend indépendant.

Les états de sortie ne fonctionnent qu'avec des objets DataSet Typés. Cet objet doit être créé avant la création de l'état de sortie ainsi il sera reconnu par l'état au moment de la construction de celui ci.
La première étape consiste à insérer un composant Crystal reports. Dans la fenêtre Explorateur de champs cliquer avec le bouton droit de la souris et choisir l'option Ajouter / Supprimer Base de données. Dans la fenêtre qui apparaît, sélectionner les champs souhaités dans ADO.NET DataSets :



Ensuite il faut confirmer, modifier ou supprimer les liaisons entre les objets DataTable dans la fenêtre qui apparaît :



Paramètrage

Le paramétrage d'un état basé sur un DataSet se fait de la même manière que les états précédemment traités (champs, groupes, Champs spéciaux, champs cumul, formules, paramètres, conditions sur les paramètres et la sélection des enregistrements ...)

Appel d'un état :

Pour faire appel à un état, il faut insérer un contrôle CrystalReportViewer au niveau d'un formulaire Windows et indiquer l'état Crystal Reports souhaité dans sa propriété Report Source.

Si cet état est non paramétré, l'affichage se fait directement après l'ouverture du formulaire sinon une fenêtre proposée pour la saisie des paramètres apparaît. Sa forme change en fonction des paramétrages effectués pour les options valeurs discrètes et plages de valeurs.
Si on souhaite que l'utilisateur saisisse les paramètres à partir d'une fenêtre que nous avons programmer. Il ne faut pas renseigner la propriété ReportSource dans la fenêtre du code et il faut avant l'appel de l'état de sortie :

- Remplir le DataSet Typé concerné
- Instanciation de l'état de sortie

```
Dim RéfEtat As New Nom_Etat_CrystalReport
```

- Si l'état contient des paramètres :

```
Réf_Etat.SetParameterValue("Nom_Param_Dans_L'état", Valeur_Param)
```

- Associer le Dataset à l'état Crystal Reports

```
Réf_Etat.SetDataSource(Réf_DataSetTypé)
```

- Charger l'état

```
NomCrystalReport.ReportSource = Réf_Etat
```

Exemple : ***Créer un état représentant pour chaque ingrédient le nom, l'unité de mesure et la raison sociale du fournisseur***

TRAVAIL A FAIRE :

- **Créer un état de sortie nommé RptListeIngrédients se basant sur le DataSet Typé DataSetIngrédient précédemment créé**
- **Disposez les champs au niveau de l'état comme souhaité**
- **Créer un formulaire FrmImprimerListeIngrédient**
- **Placer dans ce formulaire un contrôle CrystalViewer nommé CrystalReportViewer1**
- **Ecrire la procédure évenementielle associée à l'événement Load du CrystalViewer**

```
Private Sub CrystalReportViewer1_Load(ByVal sender As System.Object, ByVal e As
                                System.EventArgs) Handles CrystalReportViewer1.Load
        im DataS_T As New DatasetIngrédient
        Dim Da_Tmp As New OleDb.OleDbDataAdapter("select noming,unitémesureing, RSFou
                        from ingrédient I, Fournisseur F where I.NumFou=F.NumFou", cn)
        Da_Tmp.Fill(DataS_T, "ingrédient")
        Dim c As New RptListeIngrédients
        c.SetDataSource(DataS_T)
        CV.ReportSource = c
End Sub
```

- **Créer dans le formulaire FrmMenu un SousMenu MnuImprimerListeIngrédients qui appelle le formulaire FrmImprimerListeIngrédient**

Exemple : ***Créer un état représentant la liste des recettes relatives au thème sélectionné au démarrage de l'application.***
Cet état aura le même aspect que l'état de sortie représentant la liste des recettes précédemment créées en mode connecté mais il affichera en plus le nom du thème concerné

TRAVAIL A FAIRE :

- **Créer un état de sortie nommé RptListeIngrédients se basant sur le DataSet Typé DataSetRecette précedemment créé**
- **Disposez les champs, les formules de calcul et de synthèse au niveau de l'état comme souhaité**
- **Créer un formulaire FrmImprimerListeRecettes**
- **Placer dans ce formulaire un contrôle CrystalViewer nommé CrystalReportViewer1**

- **Ecrire la procédure évenementielle associée à l'événement Load du CrystalViewer**

```
Private Sub CrystalReportViewer1_Load(ByVal sender As System.Object, ByVal e As
                                      System.EventArgs) Handles CrystalReportViewer1.Load
        Dim DataS_T As New DataSetRecette
        DA_Recettes.Fill(DataS_T, "Recette")
        DA_Ingrédients.Fill(DataS_T, "Ingrédient")
        DA_Ingrédients_Recette.Fill(DataS_T, "Ingrédients_Recette")
        Dim DA_Thème As New OleDb.OleDbDataAdapter("Select CodeThème,NomThème from
                                      Thème where CodeThème=" & VarTheme & "", cn)
        DA_Thème.Fill(DataS_T, "Thème")

        Dim c As New RptListeRecettes
        c.SetDataSource(DataS_T)
        CrystalReportViewer1.ReportSource = c
End Sub
```

- **Créer dans le formulaire FrmMenu un SousMenu MnuImprimerListeRecettes qui appelle le formulaire FrmImprimerListeRecettes**

Exemple : ***Créer un état représentant la fiche d'une recette***
Cet état aura le même aspect que l'état de sortie représentant la liste des recettes précédemment créée mais concernera une seule recette. On pourra accéder à cet état de sortie à partir du formulaire FrmRechercherRecette où on rajoutera un bouton BtnImprimerRecette pour l'impression directe.

Dans le cas d'une impression directe, on a pas besoin du contrôle CrystalReportViewer. Il suffit de paramétrer l'état de sortie et de l'imprimer. Pour demander une impression directe :

```
Dim réf_EtatSortie as new Nom_EtatSortie
Réf_EtatSortie.PrintToPrinter(Nbr_Copies,Regroupement(True|False) , Page_Début,
                              Page_Fin)
```

TRAVAIL A FAIRE :

- **Créer l'état de sortie RptUneRecette**
- **Créer les paramètres nécessaires**
- **Ajouter au formulaire FrmRechercherRecette un bouton BtnImprimerRecette**
- **Dans la procédure évenementielle associée au bouton BtnImprimerRecette, écrire le code suivant :**

```
Private Sub BtnImprimerRecette_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BtnImprimerRecette.Click
        Dim DataS_T As New DataSetRecette
        DA_Recettes.Fill(DataS_T, "Recette")
        DA_Ingrédients.Fill(DataS_T, "Ingrédient")
        DA_Ingrédients_Recette.Fill(DataS_T, "Ingrédients_Recette")
        Dim DA_Thème As New OleDb.OleDbDataAdapter("Select CodeThème,NomThème from
                                      Thème where CodeThème=" & VarTheme & "", cn)
        DA_Thème.Fill(DataS_T, "Thème")

        Dim c As New RptListeRecettes
        c.SetParameterValue("P1",val(TxtNuméro.Text))
        c.SetDataSource(DataS_T)
        c.PrintToPrinter(1,Nothing,Nothing,Nothing)
End Sub
```

Exemple : ***Créer un état représentant la liste des recettes avec pour chaque recette une évaluation du prix***

Cet état aura la forme suivante :

Evaluation des prix des recettes du thème Dessert

Nom Recette	Prix Bas	Prix Moyen	Prix Elevé
Macédoine de fruit			X
Milk shake à la banane			X
Poires à l'anglaise	X		
Tarte aux prunes		X	

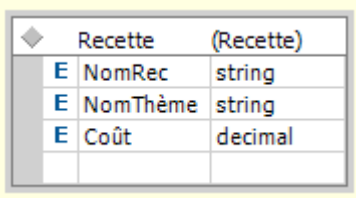
On créera cet état de deux manières :

- La première en créant des champs de formule dans le CrystralReport
- La deuxième en utilisant des champs de DataSet

Méthode1 :

TRAVAIL A FAIRE :

- **Créer un DataSet Typé nommé DataSetRecetteMéthode1 qui a la structure suivante :**



- **Soit un état CrystalReport nommé RptClassificationRecettesMéthode1 basé sur ce DataSet Typé. Dans cet état on fait glisser les champs Nom de recette et nom de thème. On crée ensuite trois champs de formules nommés Catégorie1, Catégorie2 et Catégorie3 qui vont servir à classer les recettes. Ces champs représentent respectivement les catégories 'Prix Bas', 'Prix moyen' et 'Prix élévé'. On leur associe respectivement les formules suivantes :**

- Formule 1

```
if {Recette.Coût}<=20 then
        "X"
else
        ""
```

- Formule 2

```
if {Recette.Coût}>20 and  {Recette.Coût}<=50  then
        "X"
else
        ""
```

- Formule 3

```
if {Recette.Coût}>50 then
        "X"
else
        ""
```

On fait glisser ces trois champs de formules dans l'état de sortie chacun à l'endroit édéquat

- **Créer un formulaire FrmImprimerClassificationRecettesMéthode1**
- **Placer dans ce formulaire un contrôle CrystalViewer nommé CrystalReportViewer1**

- **Ecrire la procédure évenementielle associée à l'événement Load du CrystalViewer**

```
Private Sub CrystalReportViewer1_Load(ByVal sender As System.Object, ByVal e
                    As System.EventArgs) Handles CrystalReportViewer1.Load
    Dim DataS_T As New DatasetRecetteMéthode1
    Dim da_tmp As New OleDb.OleDbDataAdapter("select NomRec,NomThème,
                    sum(puing*Qtéutilisée) as Coût From Recette R, Ingrédient
                    I, Ingrédients_Recette IR , Thème T where R.NumRec =
                    IR.NumRec And IR.NumIng = I.NumIng And
                    R.CodeThème=T.COdethème and T.codethème = " & VarTheme & "
                    group by NomRec, NomThème", cn)
    da_tmp.Fill(DataS_T.Tables(0))
    Dim c As New RptClassificationRecettesMéthode1
    c.SetDataSource(DataS_T)
    CrystalReportViewer1.ReportSource = c
End Sub
```

- **Créer dans le formulaire FrmMenu un SousMenu MnuImprimerClassification1 qui appelle le formulaire FrmImprimerClassificationRecettesMéthode1**

Méthode2 :

TRAVAIL A FAIRE :

- **Créer un DataSet Typé nommé DataSetRecetteMéthode2 qui a la structure suivante :**

Recette	(Recette)
E NomRec	string
E NomThème	string
E Coût	decimal
E CatégoriePrixElevé	string
E CatégorieBonPrix	string
E CatégoriePrixMoyen	string

Soit un état CrystalReport nommé RptClassificationRecettesMéthode2 basé sur ce DataSet Typé. Dans cet état on fait glisser les champs Nom de recette, nom de thème, CatégoriePrixElevé, CatégoriePrixMoyen et CatégorieBonPrix.

- **Créer un formulaire FrmImprimerClassificationRecettesMéthode2**
- **Placer dans ce formulaire un contrôle CrystalViewer nommé CrystalReportViewer1**
- **Ecrire la procédure évenementielle associée à l'événement Load du CrystalViewer**

```
Private Sub CrystalReportViewer1_Load(ByVal sender As System.Object, ByVal e
                As System.EventArgs) Handles CrystalReportViewer1.Load
    Dim DataS_T As New DatasetRecetteMéthode2
    Dim da_tmp As New OleDb.OleDbDataAdapter("select NomRec,NomThème,
                    sum(puing*Qtéutilisée) as Coût From Recette R, Ingrédient
                    I, Ingrédients_Recette IR , Thème T where R.NumRec =
                    IR.NumRec And IR.NumIng = I.NumIng And
                    R.CodeThème=T.COdethème and T.codethème = " & VarTheme & "
                    group by NomRec, NomThème", cn)
    da_tmp.Fill(DataS_T.Tables(0))
    For i As Integer = 0 To DataS_T.Tables(0).Rows.Count - 1
        If Not DataS_T.Tables(0).Rows(i)(2) Is DBNull.Value Then
              DataS.Tables(0).Rows(i).BeginEdit()
                    'Détermination de la catégorie pour chaque recette
                    'On mettra une croix sur la bonne catégorie
                Select Case DataS_T.Tables(0).Rows(i)(2)
                  Case Is > 50
                       DataS_T.Tables(0).Rows(i)(3) = "X"
```

```
                    Case Is <= 20
                         DataS_T.Tables(0).Rows(i)(4) = "X"
                    Case Else
                         DataS_T.Tables(0).Rows(i)(5) = "X"
                End Select
               DataS.Tables(0).Rows(i).EndEdit()
            End If
        Next
        Dim c As New RptClassificationRecettesMéthode2
        c.SetDataSource(DataS_T)
        CrystalReportViewer1.ReportSource = c
    End Sub
```

- **Créer dans le formulaire FrmMenu un SousMenu MnuImprimerClassification2 qui appelle le formulaire FrmImprimerClassificationRecettesMéthode2**

COMPLEMENT MODE CONNECTE

Ceci est un exemple d'utilisation d'une transaction.

```
Private Sub MnuUtilisationTransaction_Click(ByVal sender As System.Object, ByVal e As
                                    System.EventArgs) Handles MnuTransaction.Click
        Dim T As OleDb.OleDbTransaction
        T = cn.BeginTransaction
        Dim cmd As New OleDb.OleDbCommand
        cmd.Transaction = T
        cmd.Connection = cn
        cmd.CommandType = CommandType.Text
        cmd.CommandText = "update ingrédient set PUIng=PUIng + ((PUIng*10)/100)"
        cmd.ExecuteNonQuery()
        If MessageBox.Show("Etes-vous sûr de vouloir mettre à jour ? ", "Augmentation
                        des prix", MessageBoxButtons.YesNo) = DialogResult.Yes Then
            T.Commit()
            MessageBox.Show("Les prix des ingrédients ont été augmentés de 10%")
        Else
            T.Rollback()
        End If
    End Sub
```