TP
DEVELOPPEMENT EN COUCHES

Plan :

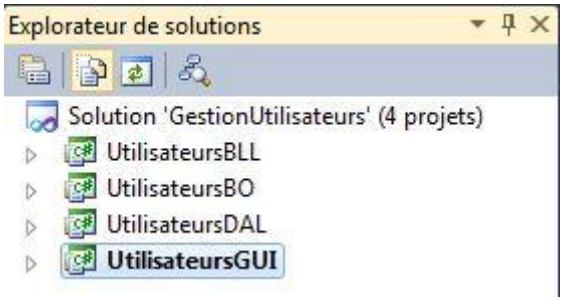
I. Introduction

L'application développée dans ce TP permet de transposer l'application d'un TP précédent, « TP - ADO.net & SQL Server » qui était une application procédurale en mode console, en une application graphique respectant le modèle de développement en couches.

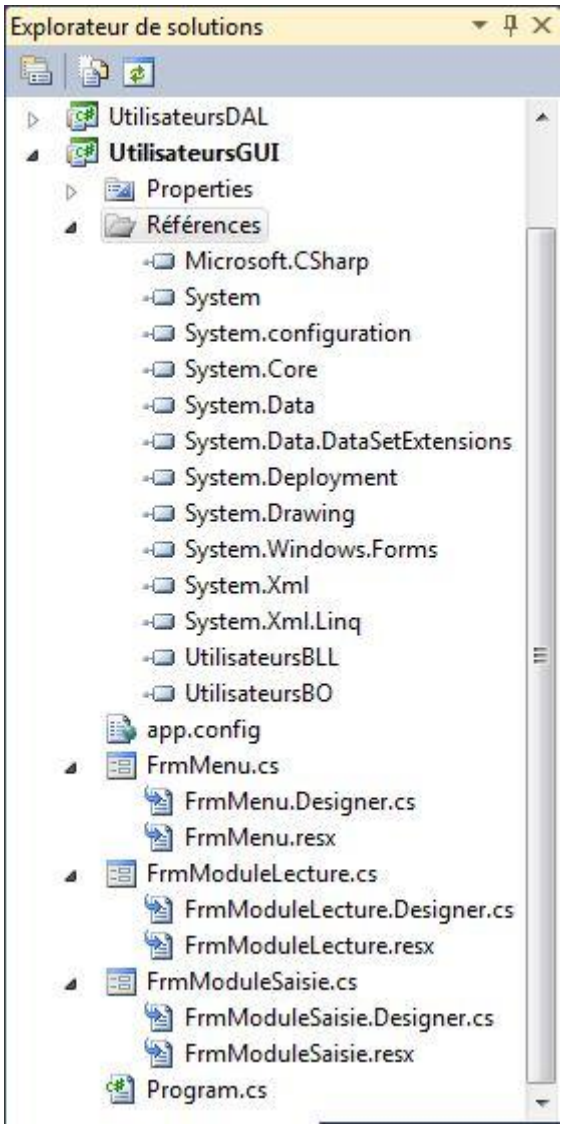
L'application permettra donc, au minimum, de consulter et d'enregistrer dans une base de données Microsoft SQL Server un identifiant et un nom d'utilisateur.

II. Architecture finale de l'application

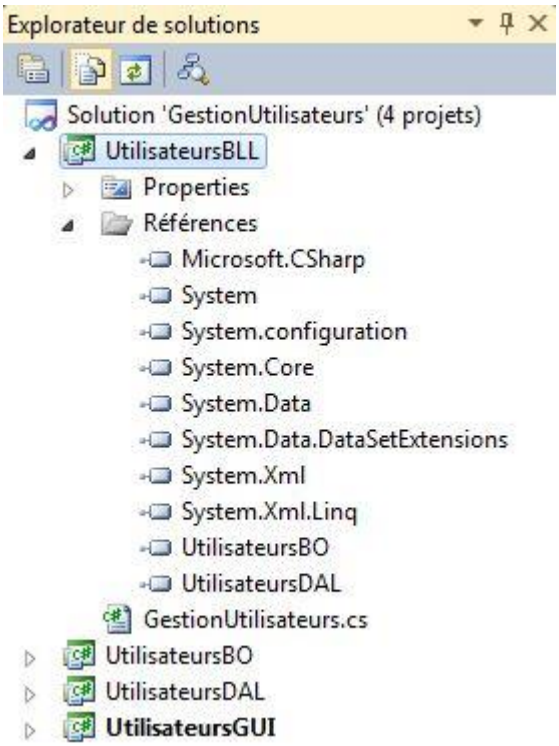
La solution finale contiendra 4 parties (représentants les 4 couches) :



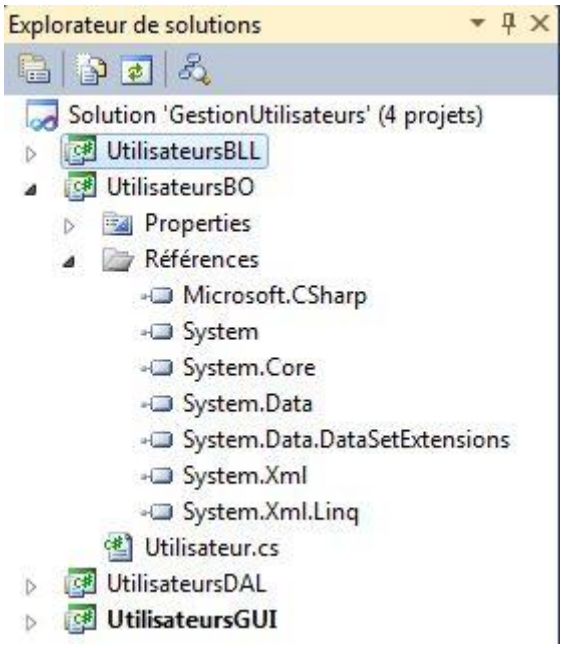
La partie **UtilisateursGUI** : application Windows Forms, représentant l'interface graphique



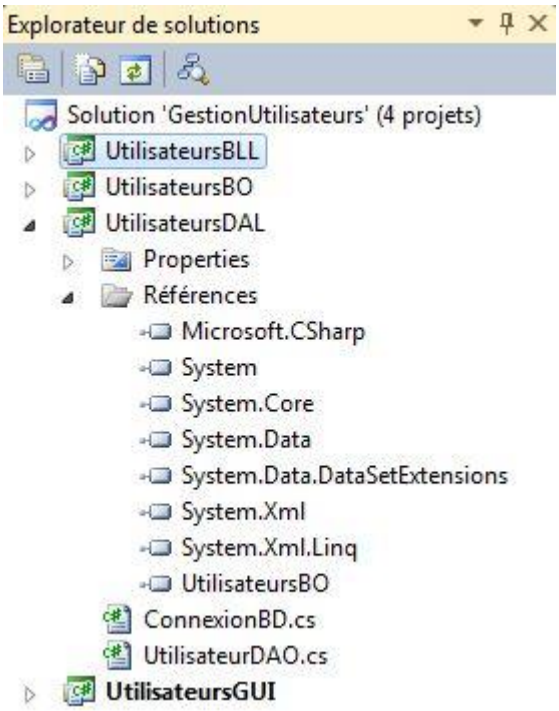
La partie **UtilisateursBLL** : bibliothèque de classes, représentant la couche logique métier



La partie **UtilisateursBO** : bibliothèque de classes, représentant la couche objets métier



La partie **UtilisateursDAL** : bibliothèque de classes, représentant la couche de données



III. Création de la base de données

Exécutez le script ci-dessous sous Microsoft SQL Server 2008 R2 pour obtenir :

- ✓ Une base de données nommée : BD_Utilisateurs
- ✓ Une table dans cette BD nommée : T_Identification
- ✓ 2 champs dans cette table nommés : Id_client, Nom_client de types respectifs : entier : int, chaîne de caractères : NvarChar(50). L'Id en tant que clé primaire (PRIMARY KEY) auto incrémantable (IDENTITY) ne pouvant être « Null » et le nom ne pouvant être « Null ».

```
-- Création de la base de données
CREATE DATABASE BD_Utilisateurs;
GO

-- Indique la BDD à utiliser et dans laquelle on exécute les requêtes
USE BD_Utilisateurs;
GO

-- Création de la table et création et paramétrage des champs
CREATE TABLE T_Identification
(
 Id_utilisateur INT NOT NULL PRIMARY KEY IDENTITY(1,1),
 Nom_utilisateur nvarchar(50) NOT NULL
);
GO
```

IV. Développement de l'application en couches

1. La couche GUI : l'interface graphique

- ✓ Créer un projet de type Visual C# > Windows > Application Windows Forms nommé **UtilisateursGUI** dans une solution nommée **GestionUtilisateurs**.

- ✓ L'interface graphique de l'application contiendra 3 formulaires :
  - Un formulaire "Menu"
  - Un formulaire "Module de lecture en BD"
  - Un formulaire "Module de saisie en BD"

- ✓ Le formulaire "Module de lecture en BD" contiendra :
  - une DataGridView où seront affichées les données des utilisateurs
  - un bouton "Actualiser" permettra de rafraîchir les données du DataGridView

- ✓ Le formulaire "Module de saisie en BD" contiendra :
  - un champ de type textBox permettant de saisir le nom
  - un bouton "Enregistrer" permettant d'enregistrer les données saisies dans la base de données

- ✓ Le formulaire "Menu" contiendra deux boutons :
  - le premier bouton "Lecture" permettra d'obtenir le formulaire « Module de lecture en BD »
  - le deuxième bouton "Saisie" permettra d'obtenir le formulaire « Module de saisie en BD »

Vous pouvez, bien entendu, selon votre avancement, agrémenter vos formulaires de :
- boutons « Retour » sur chaque formulaire pour fermer le formulaire
- bouton « Quitter » sur le formulaire de menu pour quitter l'application
- Messages de confirmation d'action effectuée du type : Voulez-vous vraiment…

Et le projet de :
- fonctionnalité de modification d'un enregistrement en BD
- fonctionnalité de suppression d'un enregistrement en BD

Pour appeler un formulaire :

```
FrmModuleLecture FrmModLect;
FrmModLect = new FrmModuleLecture();
FrmModLect.ShowDialog(); // ouverture du formulaire
FrmModLect.Close() ; // fermeture du formulaire
```

L'interaction des formulaires avec la couche BLL sera vue un peu plus loin dans ce TP.

2. La couche BLL : la couche logique métier

Dans la solution précédemment créée, ajouter un nouveau projet bibliothèque de classes nommé UtilisateursBLL contenant une unique classe pour le moment : « GestionUtilisateurs ». Cette classe sera implémentée dans la suite du TP.

3. La couche BO : les objets métier

Dans la solution précédemment créée, ajouter un nouveau projet bibliothèque de classes nommé UtilisateursBO contenant une unique classe pour le moment : « Utilisateur ».

Cette classe « Utilisateur » est composée :

- des attributs suivants : Id_utilisateur, Nom_utilisateur (les mêmes informations que dans la table « T_Identification » de la base de données « BD_Utilisateurs »).
- un constructeur à 1 paramètre qui permet de valoriser le nom.
- un constructeur à 2 paramètres qui permet de valoriser l'id et le nom.
- 2 propriétés Id et Nom avec pour chacun d'eux un getter et un setter (accesseur en lecture et écriture)

4. La couche DAL : la couche d'accès aux données

Dans la solution précédemment créée, ajouter un nouveau projet bibliothèque de classes nommé UtilisateursDAL contenant 2 classes :

- classe ConnexionBD qui permet l'accès à la base de données
- classe UtilisateurDAO contenant les requêtes sur notre table T_Identification

Code source de la classe ConnexionBD :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;

namespace UtilisateursDAL
{
    // Classe de gestion de la connexion à la BD
    public class ConnexionBD
    {
        private SqlConnection maConnexion;
        private static ConnexionBD uneConnexionBD; // instance d'une connexion
        private string chaineConnexion; // chaîne de connexion à la BD

        // Accesseur en lecture de la chaîne de connexion
        public string GetchaineConnexion()
        {
            return chaineConnexion;
        }
```

```
        // Accesseur en écriture de la chaîne de connexion
        public void SetchaineConnexion(string ch)
        {
            chaineConnexion = ch;
        }

        // Accesseur en lecture, renvoi une instance de connexion
        public static ConnexionBD GetConnexionBD()
        {
            if (uneConnexionBD == null)
            {
                uneConnexionBD = new ConnexionBD();
            }
            return uneConnexionBD;
        }

        // Constructeur privé
        private ConnexionBD()
        {

        }

        public SqlConnection GetSqlConnexion()
        {
            if (maConnexion == null)
            {
                maConnexion = new SqlConnection();
            }
            maConnexion.ConnectionString = chaineConnexion;

            // Si la connexion est fermée, on l'ouvre
            if (maConnexion.State == System.Data.ConnectionState.Closed)
            {
                maConnexion.Open();
            }
            return maConnexion;
        }

        public void CloseConnexion()
        {
            // Si la connexion est ouverte, on la ferme
            if (this.maConnexion != null && this.maConnexion.State !=
System.Data.ConnectionState.Closed)
            {
                this.maConnexion.Close();
            }
        }
    }
}
```

Code source de la classe UtilisateurDAO :

La classe UtilisateurDAO utilise la classe Utilisateur de la bibliothèque de classes UtilisateursBO.
Pour ce faire, ajouter une référence depuis la classe UtilisateursDAL vers la bibliothèque de classes UtilisateursBO (dans l'explorateur de solution, clique sur Références de UtilisateursDAL → Ajouter une référence → Onglet Projets → Choisir UtilisateursBO → OK).

Attention : aux 2 using nécessaires au début de la classe UtilisateurDAO.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using UtilisateursBO; // Référence la couche BO
using System.Data.SqlClient;


namespace UtilisateursDAL
{
    public class UtilisateurDAO
    {
        private static UtilisateurDAO unUtilisateurDAO;

        // Accesseur en lecture, renvoi une instance
        public static UtilisateurDAO GetunUtilisateurDAO()
        {
            if (unUtilisateurDAO == null)
            {
                unUtilisateurDAO = new UtilisateurDAO();
            }
            return unUtilisateurDAO;
        }

        // Cette méthode retourne une List contenant les objets Utilisateurs
contenus dans la table Identification
        public static List<Utilisateur> GetUtilisateurs()
        {
            int id;
            string nom;
            Utilisateur unUtilisateur;

            // Connexion à la BD
            SqlConnection                    maConnexion                    =
ConnexionBD.GetConnexionBD().GetSqlConnexion();

            // Création d'une liste vide d'objets Utilisateurs
            List<Utilisateur> lesUtilisateurs = new List<Utilisateur>();

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = maConnexion;
            cmd.CommandText = " SELECT * FROM T_Identification";

            SqlDataReader monReader = cmd.ExecuteReader();
```

```
            // Remplissage de la liste
            while (monReader.Read())
            {
                id = Int32.Parse(monReader["Id_Utilisateur"].ToString());
                if (monReader["Nom_Utilisateur"] == DBNull.Value)
                {
                    nom = default(string);
                }
                else
                {
                    nom = monReader["Nom_utilisateur"].ToString();
                }

                unUtilisateur = new Utilisateur(id,nom);
                lesUtilisateurs.Add(unUtilisateur);
            }

            // Fermeture de la connexion
            maConnexion.Close();

            return lesUtilisateurs;
        }

        // Cette méthode insert un nouvel utilisateur passé en paramètre dans
la BD
        public static int AjoutUtilisateur(Utilisateur unUtilisateur)
        {
            int nbEnr;

            // Connexion à la BD
            SqlConnection                    maConnexion                    =
ConnexionBD.GetConnexionBD().GetSqlConnexion();

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = maConnexion;
            cmd.CommandText  =  "INSERT  INTO  T_Identification  values('"  +
unUtilisateur.Nom + "')";

            nbEnr = cmd.ExecuteNonQuery();

            // Fermeture de la connexion
            maConnexion.Close();

            return nbEnr;
        }

        // Cette méthode modifie un utilisateur passé en paramètre dans la BD
        public static int UpdateUtilisateur(Utilisateur unUtilisateur)
        {
            int nbEnr;

            // Connexion à la BD
            SqlConnection                    maConnexion                    =
ConnexionBD.GetConnexionBD().GetSqlConnexion();

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = maConnexion;
            cmd.CommandText = "UPDATE T_Identification SET Nom_utilisateur = '"
+ unUtilisateur.Nom + "' WHERE Id_utilisateur = " + unUtilisateur.Id;
```

```
            nbEnr = cmd.ExecuteNonQuery();

            // Fermeture de la connexion
            maConnexion.Close();

            return nbEnr;
        }

        // Cette méthode supprime de la BD un utilisateur dont l'id est passé
en paramètre
        public static int DeleteUtilisateur(int id)
        {
            int nbEnr;

            // Connexion à la BD
            SqlConnection                    maConnexion                    =
ConnexionBD.GetConnexionBD().GetSqlConnexion();

            SqlCommand cmd = new SqlCommand();
            cmd.Connection = maConnexion;
            cmd.CommandText    =    "DELETE    FROM    T_Identification    WHERE
Id_utilisateur = " + id;

            nbEnr = cmd.ExecuteNonQuery();

            // Fermeture de la connexion
            maConnexion.Close();

            return nbEnr;
        }
    }
}
```

V. Gestion des relations entre couches

1. Implémentation, dans la couche BLL, des actions disponibles sur la GUI

La couche BLL va permettre de faire les traitements sur les données et actions saisies/effectués, par l'opérateur, au travers de l'interface graphique ainsi que sur les données provenant de la base de données.

Cette couche BLL va donc jouer le rôle d'intermédiaire entre l'interface graphique (GUI) et la couche d'accès aux données (DAL).

Ajouter ces références au projet UtilisateursBLL :

- Référence .net à System.Configuration (rajouter le using à ce namespace)
- Référence à la bibliothèque UtilisateursDAL (rajouter le using)
- Référence à la bibliothèque UtilisateursBO (rajouter le using)

Code source de la classe GestionUtilisateurs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;
using UtilisateursBO; // Référence la couche BO
using UtilisateursDAL; // Référence la couche DAL

namespace UtilisateursBLL
{
    public class GestionUtilisateurs
    {
        private static GestionUtilisateurs uneGestionUtilisateurs; // objet BLL

        // Accesseur en lecture
        public static GestionUtilisateurs GetGestionUtilisateurs()
        {
            if (uneGestionUtilisateurs == null)
            {
                uneGestionUtilisateurs = new GestionUtilisateurs();
            }
            return uneGestionUtilisateurs;
        }

        // Définit la chaîne de connexion grâce à la méthode SetchaineConnexion
de la DAL
        public static void SetchaineConnexion(ConnectionStringSettings chset)
        {
            string chaine = chset.ConnectionString;
            ConnexionBD.GetConnexionBD().SetchaineConnexion(chaine);
        }

        // Méthode qui renvoit une List d'objets Utilisateur en faisant appel à
la méthode GetUtilisateurs() de la DAL
        public static List<Utilisateur> GetUtilisateurs()
        {
            return UtilisateurDAO.GetUtilisateurs();
        }

        // Méthode qui renvoi l'objet Utilisateur en l'ajoutant à la
        // BD avec la méthode AjoutUtilisateur de la DAL
        public static int CreerUtilisateur(Utilisateur ut)
        {
            return UtilisateurDAO.AjoutUtilisateur(ut);
        }
        //  Méthode qui modifie un nouvel Utilisateur avec la méthode
UpdateUtilisateur de la DAL
        public static int ModifierUtilisateur(Utilisateur ut)
        {
            return UtilisateurDAO.UpdateUtilisateur(ut);
        }
        //  Méthode qui supprime un Utilisateur avec la méthode
DeleteUtilisateur de la DAL
        public static int SupprimerUtilisateur(int id)
        {
            return UtilisateurDAO.DeleteUtilisateur(id);
        }
    }
}
```

2. Appel de la couche BLL par la couche GUI

La couche GUI va maintenant pouvoir faire appel à la couche BLL pour appliquer les traitements se cachant derrière les boutons.

Pour ce faire, la couche GUI doit informer la couche BLL de la source de données (BD, fichiers, etc.) qui va être utilisée pour stocker les données de l'application.

Cette communication va se faire au travers d'un fichier de configuration. Ce fichier de configuration contiendra les informations nécessaires pour effectuer la connexion à cette source de données. (Nous aurions pu écrire ces informations dans le code de notre application mais le fait de passer par un fichier de configuration rend la maintenance applicative plus facile par la suite).

a. Création d'un fichier de configuration

Dans le projet GUI :
Menu Données > Ajouter une nouvelle source de données…

Dans la fenêtre qui apparaît :
Choisissez « Base de données », puis cliquer sur Suivant > Suivant >

Dans la fenêtre qui apparaît :
Cliquer sur le bouton « Nouvelle connexion… »

Dans la fenêtre qui apparaît :
Source de données : choisir *Microsoft SQL Server (SqlClient) > Continuer*
Nom du serveur : nom*_du_serveur_utilisé_pour_créer_la_BD*
Connexion au serveur : *Utiliser l'authentification Windows*
Connexion à la base de données : *sélectionner le nom de votre base de donnés, à savoir BD_Utilisateurs*

Cliquer sur le bouton pour tester la connexion.
Cliquer sur OK pour enregistrer.

La fenêtre qui apparaît vous fournit la chaîne de connexion que vous pouvez utiliser dans votre code. Cependant, nous avons choisit d'utiliser un fichier de configuration qui utilisera cette chaine de connexion.

De cette manière vous centralisez les informations nécessaires à la connexion à votre base de données. En cas de changements, seul le fichier de configuration sera à modifier au lieu de devoir modifier votre code à chaque fois que cette connexion apparaît.

Appuyer sur Suivant et dans la fenêtre qui apparaît, indiquer le nom de la chaîne de connexion : simplifier le nom proposé en *Utilisateur*.

Cliquer sur Suivant, puis sur Terminer et répondez Oui à la question.

Supprimer le fichier XSD créer (qui contient le dataset).

Le fichier de configuration ainsi créé s'appelle app.config et est d'extension XML. Ouvrez le pour renommer la valeur de l'attribut name (mettez Utilisateur).

Voici notre fichier de configuration :

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
    <!-- Chaine de connexion définie pour la connexion à la BD-->
    <connectionStrings>
        <add            name="Utilisateur"           connectionString="Data
    Source=localhost;Initial            Catalog=BD_Utilisateurs;Integrated
    Security=True"
            providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

b. Connexion à la base de données

La couche GUI doit donc informer la couche BLL de la connexion à la base de données, et ce, le plus tôt possible dans l'application.
Nous allons donc ajouter cette instruction dans le constructeur des formulaires qui auront besoin d'un accès à la base de données.
(par exemple dans FrmModuleSaisie.cs juste après l'instruction : InitializeComponent();)

Commencer par ajouter dans le projet GUI, les références nécessaires :

- Clic droit sur le projet UtilisateursGUI > Ajouter une référence > Onglet .Net > System.Configuration > Ajouter
- Clic droit sur le projet UtilisateursGUI > Ajouter une référence > Onglet Projet > Utilisateurs.BLL > Ajouter
- Clic droit sur le projet UtilisateursGUI > Ajouter une référence > Onglet Projet > UtilisateursBO > Ajouter

3. Formulaire de saisie : enregistrement des données en base

Ajouter les using nécessaires :

Dans le fichier FrmModuleSaisie.cs, ajouter :

```
using System.Configuration;
using UtilisateursBO; // Référence la couche BO
using UtilisateursBLL; // Référence la couche BLL
```

Après l'instruction InitializeComponent() du constructeur, ajouter :

```
// Récupération de chaîne de connexion à la BD à l'ouverture du formulaire
GestionUtilisateurs.SetchaineConnexion(ConfigurationManager.ConnectionStrings
["Utilisateur"]);
```

Code de la méthode appelée lors du clic sur le bouton « Enregistrer » du formulaire FrmModuleSaisie.cs :

```
// Code exécuté sur l'évènement Click du bouton Enregistrer
private void btnEnregistrer_Click(object sender, EventArgs e)
{
   // Création de l'objet Utilisateur avec le nom récupéré dans la GUI
   Utilisateur uti = new Utilisateur(txtNom.Text);

   // Appel de la méthode CreerUtilisateur de la couche BLL
   GestionUtilisateurs.CreerUtilisateur(uti);
 }
```

4. Formulaire de lecture : remplissage du Datagridview

Ajouter les using nécessaires :

Dans le fichier FrmModuleLecture.cs, ajouter :

```
using System.Configuration;
using UtilisateursBLL; // Référence la couche BLL
using UtilisateursBO; // Référence la couche BO
```

Après l'instruction InitializeComponent() du constructeur, ajouter :

```
// Récupération de chaîne de connexion à la BD à l'ouverture du formulaire
GestionUtilisateurs.SetchaineConnexion(ConfigurationManager.ConnectionStrings
["Utilisateur"]);
```

Ainsi que la définition des propriétés du Datagridview :

```
// Blocage de la génération automatique des colonnes
dgv.AutoGenerateColumns = false;

// Création d'une en-tête de colonne pour la colonne 1
DataGridViewTextBoxColumn IdColumn = new DataGridViewTextBoxColumn();
IdColumn.DataPropertyName = "Id";
IdColumn.HeaderText = "Identifiant";

// Création d'une en-tête de colonne pour la colonne 2
DataGridViewTextBoxColumn NomColumn = new DataGridViewTextBoxColumn();
NomColumn.DataPropertyName = "Nom";
NomColumn.HeaderText = "Nom de l'utilisateur";

// Ajout des 2 en-têtes de colonne au datagridview
dgv.Columns.Add(IdColumn);
dgv.Columns.Add(NomColumn);

// Définition du style apporté au datagridview
dgv.ColumnHeadersVisible = true;
DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
columnHeaderStyle.BackColor = Color.Beige;
columnHeaderStyle.Font = new Font("Verdana", 10, FontStyle.Bold);
dgv.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
```

Ainsi que le chargement de la source du Datagridview :

```
// Création d'un objet List d'Utilisateur à afficher dans le datagridview
List<Utilisateur> liste = new List<Utilisateur>();
liste = GestionUtilisateurs.GetUtilisateurs();

// Rattachement de la List à la source de données du datagridview
dgv.DataSource = liste;
```

Code de la méthode appelée lors du clic sur le bouton « Actualiser » du formulaire FrmModuleLecture.cs :

```
// Création d'un objet List d'Utilisateur à afficher dans le datagridview
List<Utilisateur> liste = new List<Utilisateur>();
liste = GestionUtilisateurs.GetUtilisateurs();

// Rattachement de la List à la source de données du datagridview
dgv.DataSource = liste;

//// CI-DESSOUS - CODE ALTERNATIF DE REMPLISSAGE DU DATAGRIDVIEW MAIS MOINS
"PROPRE"

//// Effacement de toutes les lignes
//dgv.Rows.Clear();

//// On définit le nombre de lignes nécessaires en comptant le nombre
d'éléments dans la liste
//dgv.Rows.Add(GestionUtilisateurs.GetUtilisateurs().Count);

//// remplissage des lignes du datagridview
//for (int i = 0; i < GestionUtilisateurs.GetUtilisateurs().Count; i++)
//{            //    dgv[0, i].Value =
GestionUtilisateurs.GetUtilisateurs()[i].Id;            //    dgv[1,
i].Value = GestionUtilisateurs.GetUtilisateurs()[i].Nom;
//}

//// CI-DESSUS - CODE ALTERNATIF DE REMPLISSAGE DU DATAGRIDVIEW MAIS MOINS
"PROPRE"
```

VI. Test de l'application

Il ne reste plus qu'à définir le projet de démarrage de l'application ainsi que le premier formulaire qui sera ouvert au lancement de l'application.

Clic droit sur le projet UtilisateursGUI > Définir comme projet de démarrage

Ouvrir la classe Program.cs contenant la méthode static voir Main() et y rajouter (si cela n'est pas déjà présent) :

```
Application.Run(new FrmMenu());
```

Lancer l'application (bouton lecture vert) et tester les différents formulaires ainsi que les boutons :

- vérifier que les données saisies sont bien enregistrées dans la base de données.
- vérifier que les données s'affichent bien dans le Datagridview.

VII. Aller plus loin

Vous pouvez compléter l'application avec :

- Des contrôles de saisies sur les zones de saisies et/ou les boutons
- Des messages de demande de confirmation (sur la fermeture de l'application, sur l'enregistrement, la modification, la suppression, etc.)
- Un message de confirmation sur le bon enregistrement des infos en base de données.
- L'ajout d'un formulaire de modification des données d'un utilisateur.