



Université de Caen Normandie
Institut Universitaire de Technologie de Caen
Département Informatique

Bachelor Universitaire de Technologie

INFORMATIQUE

COMPTE-RENDU - TP Gestion d'une librairie d'éditeur

LEBRASSEUR Thibaud

Année universitaire 2022 / 2023

Résumé

Le projet consiste à développer une application de gestion de bibliothèque en utilisant le langage de programmation C# et le framework .NET. L'application permet aux utilisateurs de se connecter, d'ajouter des auteurs et des livres, de les modifier ou de les supprimer, et de consulter les listes d'auteurs et de livres.

Université de Caen Normandie
Institut Universitaire de Technologie de Caen
Département Informatique

Bachelor Universitaire de Technologie

INFORMATIQUE

COMPTE-RENDU - TP Gestion d'une librairie d'éditeur

LEBRASSEUR Thibaud

SOMMAIRE

I. Introduction.....	6
Introduction de l'objectif du TP.....	7
II. Présentation du logiciel.....	8
Écran principal.....	9
Écran des livres.....	10
Écran de configuration à la base de données.....	11
Connexion.....	12
Écran des utilisateurs.....	14
Modifier un utilisateur.....	14
Ajouter un utilisateur.....	15
Supprimer un utilisateur.....	16
Ajouter un auteur.....	17
Ajouter un livre.....	19
Supprimer un livre.....	21
Erreur champs.....	21
Déconnexion.....	22
III. Conception de l'application.....	23
La classe DLL.....	24
La classe principale.....	29
IV. Conclusion.....	37
Conclusion du TP.....	38

I. Introduction

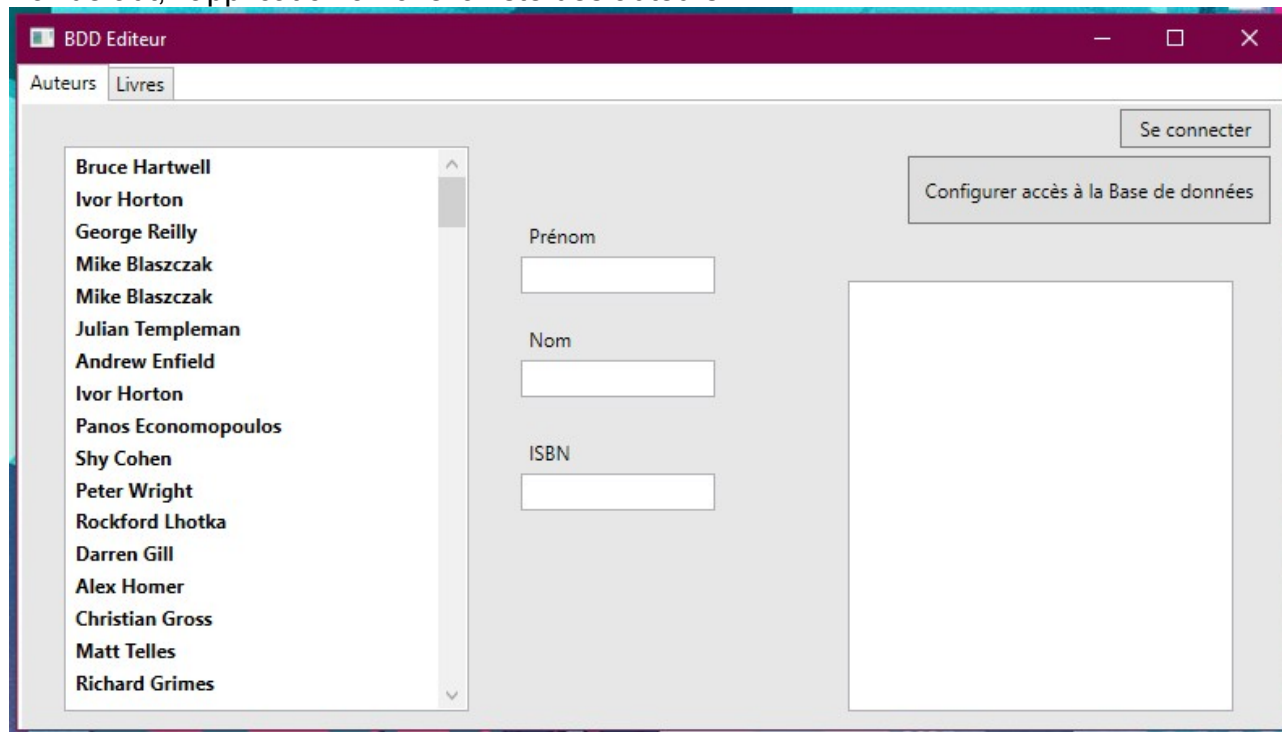
Introduction de l'objectif du TP

Dans ce TP, l'objectif est de réaliser une application, un exécutable, qui va permettre la gestion des auteurs et livres enregistrés informatiquement dans la base de données d'une librairie d'éditeur. Cette application doit fonctionner avec le langage C# qui gère la partie fonctionnelle de l'application (interactions, événements, etc...), Linq qui doit permettre de se connecter à la base de données contenant les informations sur les auteurs, livres et utilisateurs (lire, écrire les données) et WPF (Windows Presentation Foundation) qui doit gérer la partie graphique de l'application.

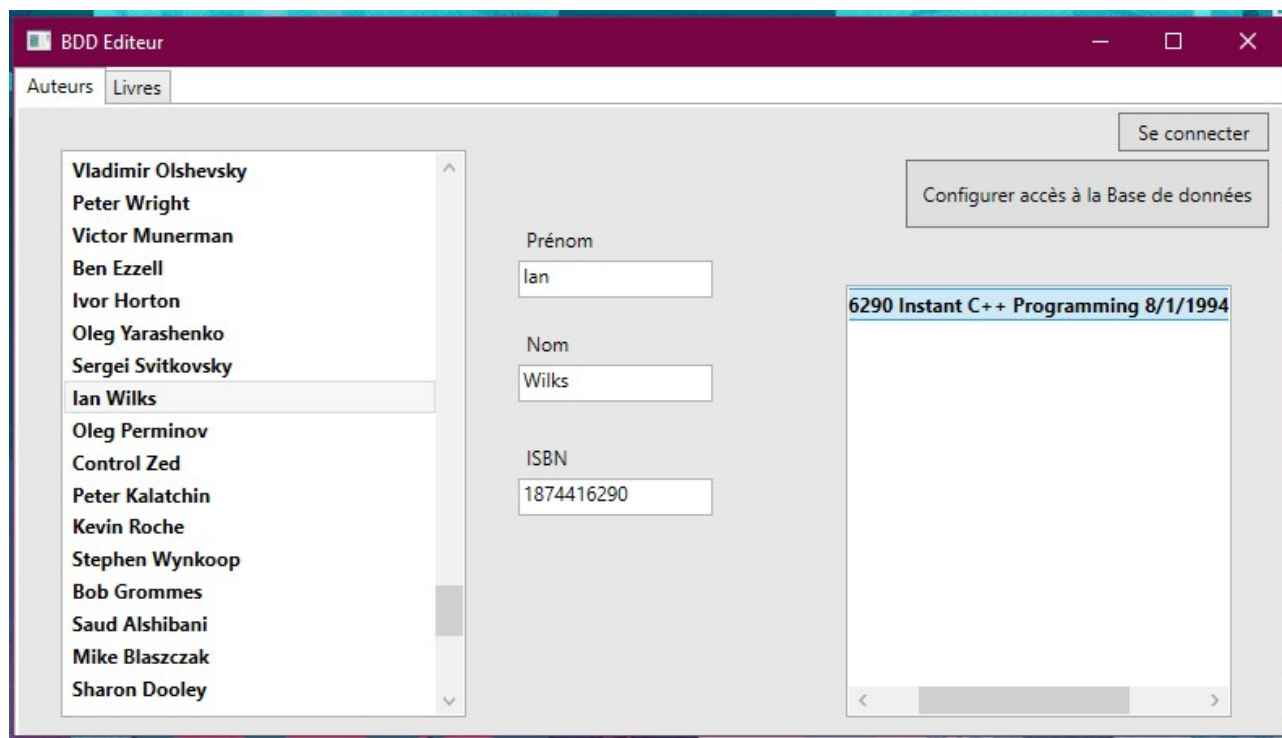
II. Présentation du logiciel

Écran principal

Par défaut, l'application affiche la liste des auteurs.



Si on clique sur un auteur, on peut afficher ses informations : son nom, prénom, ISBN ainsi que ses livres publiés.



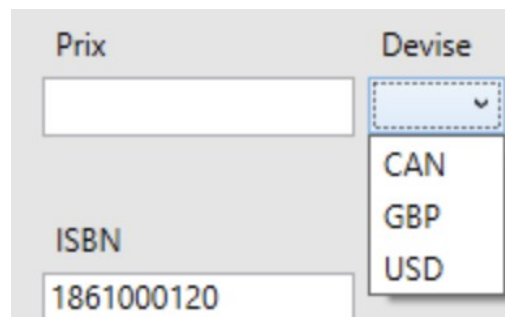
Écran des livres

The screenshot shows the 'BDD Editeur' application window. The 'Livres' tab is active. On the left, a list of books is displayed, including 'Beginning ATL COM', 'Beginning C++', 'Professional MFC With Visual C++ 2nd Editic', 'Professional MFC With Visual C++ 6', 'Beginning NT Programming', 'Instant Visual Basic 5 Activex Control Progra', 'Beginning Java 1.1', 'Professional Visual C++ 5 Activex/COM Cont', 'Professional Java Fundamentals', 'Beginning Visual Basic 5', 'Professional Visual Basic 5 Business Objects', 'Instant VB Script', 'Instant Activex Web Database', 'Professional NT Internet Information Server', and 'Beginning Visual C++ Components Program'. On the right, the details for a selected book are shown. The 'Titre' field is empty. The 'Date de publication' field is empty. The 'Prix' field is empty. The 'Devise' field is a dropdown menu. The 'Auteur' field is empty. The 'ISBN' field is empty. The 'Nom' field is empty. The 'Prénom' field is empty. There are buttons for 'Se connecter' and 'Configurer accès à la Base de données'.

Si on clique sur un livre dans la liste, on peut obtenir plus d'informations sur le livre : son ISBN, titre, date de publications, ses prix en fonction de la devise, et son auteur.

The screenshot shows the 'BDD Editeur' application window. The 'Livres' tab is active. The book 'Beginning C++' is selected in the list. The details for this book are displayed on the right. The 'Titre' field contains 'Beginning C++'. The 'Date de publication' field contains '01/04/1998'. The 'Prix' field is empty. The 'Devise' field is a dropdown menu. The 'Auteur' field is empty. The 'ISBN' field contains '1861000120'. The 'Nom' field contains 'Blaszczak'. The 'Prénom' field contains 'Mike'. There are buttons for 'Se connecter' and 'Configurer accès à la Base de données'.

Si on clique sur liste déroulante “Devise”, on peut sélectionner le prix du livre sélectionné en fonction de la devise sélectionnée.



En Dollar canadien



En Livres sterling

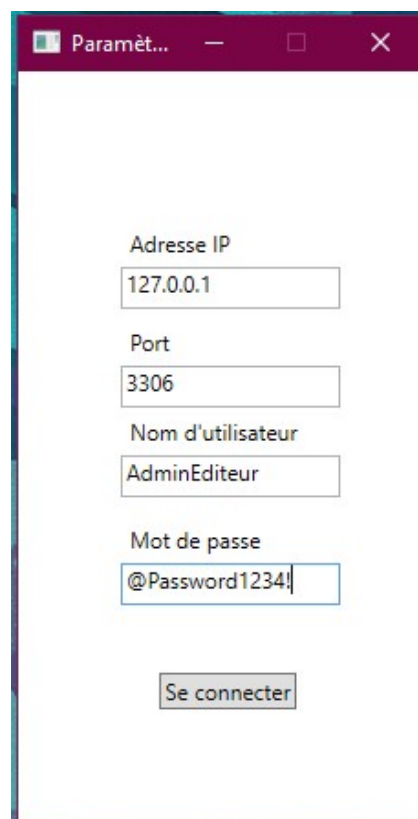


En Dollar américain



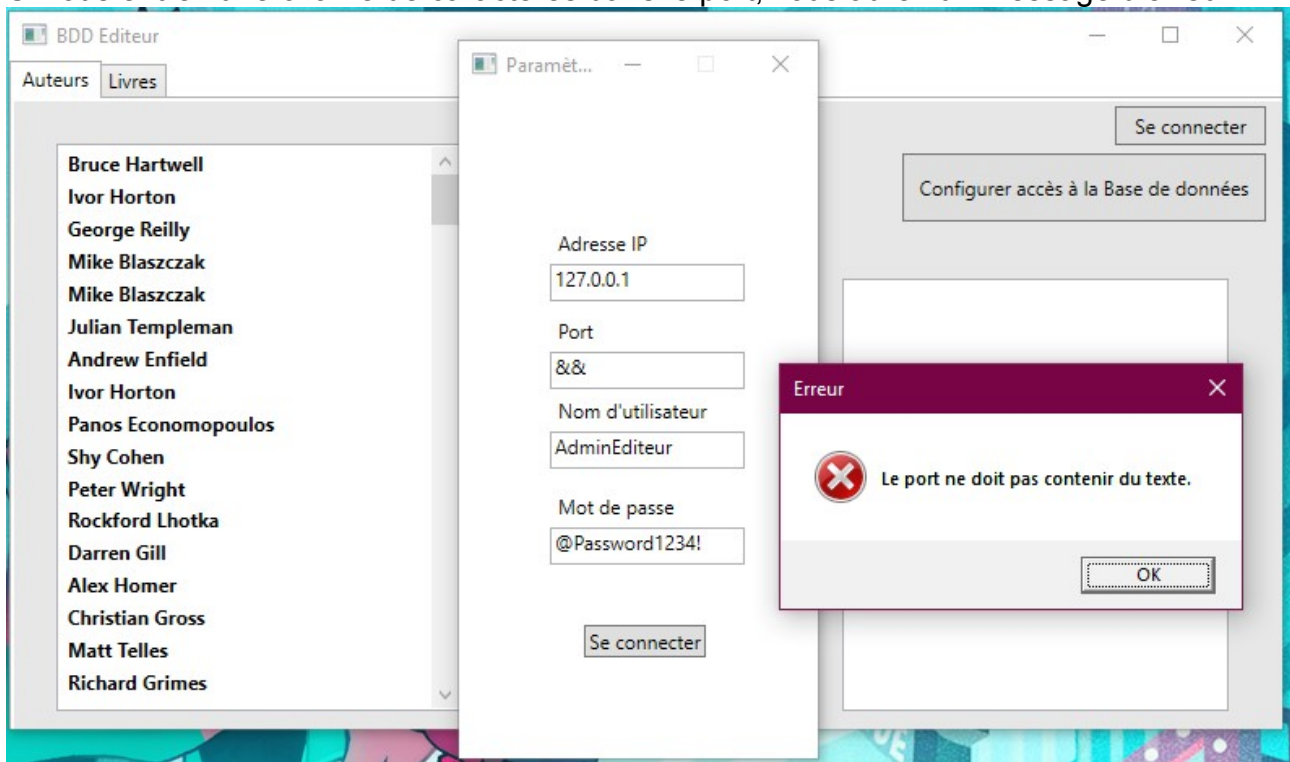
Écran de configuration à la base de données

Si vous cliquez sur le bouton “Configurer accès à la Base de données”, cette fenêtre va apparaître :



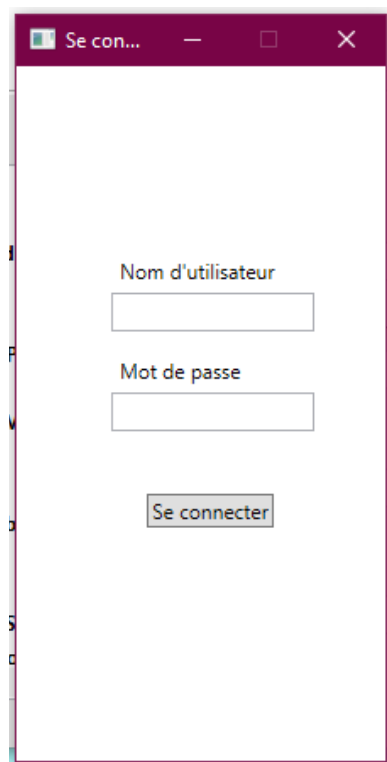
Les informations de connexions à la base de données sont enregistrées dans les settings de l'application.

Si vous entrez une chaîne de caractères dans le port, vous aurez un message d'erreur.

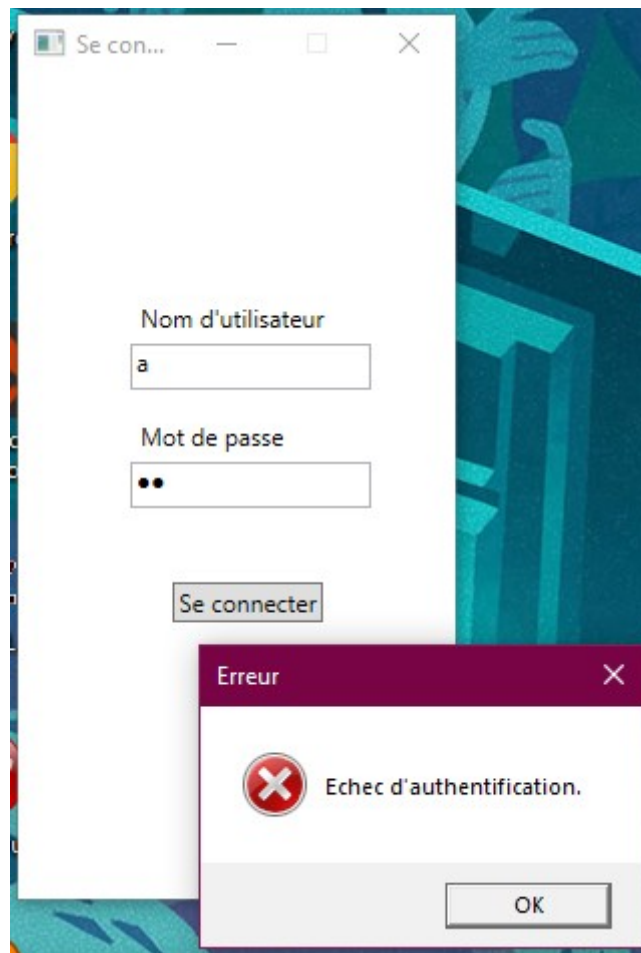


Connexion

Si vous cliquez sur le bouton “Se connecter”, ce menu va apparaître :

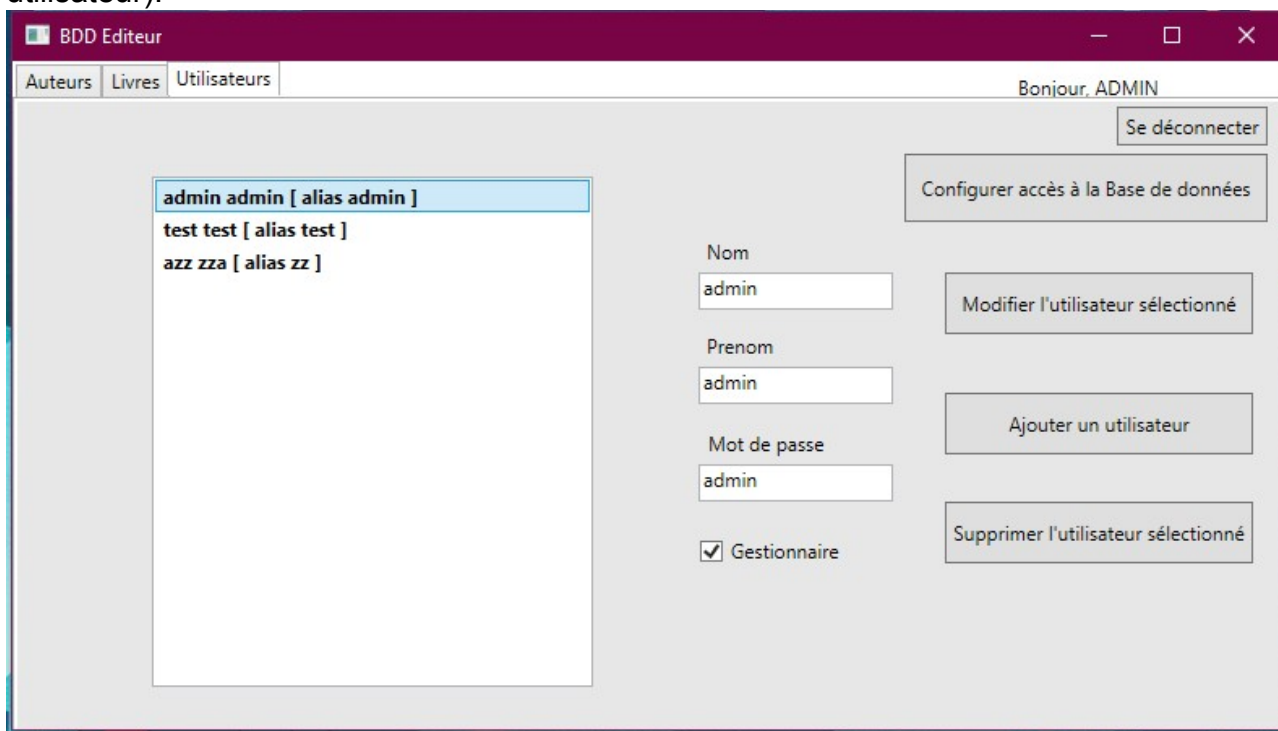


Si vous tentez de vous connecter alors que l'utilisateur n'existe pas ou que vous vous êtes trompé de mot de passe, vous aurez ce message d'erreur.



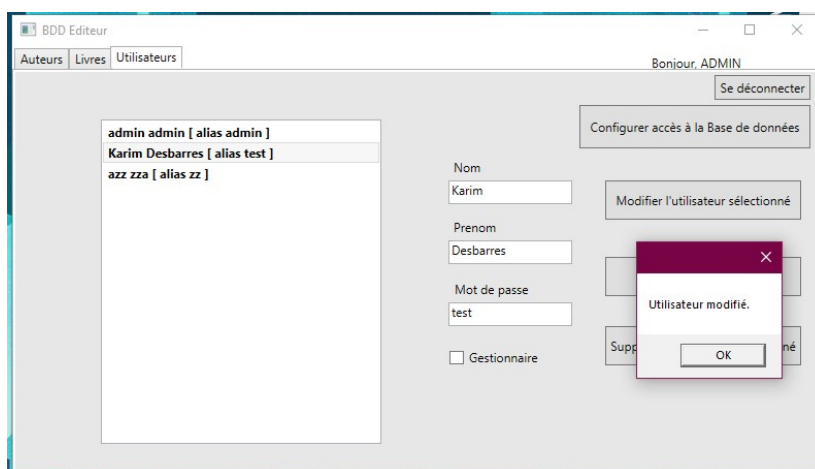
Écran des utilisateurs

Si vous vous connectez en tant que gestionnaire, vous aurez un nouvel onglet “Utilisateurs” sur lequel vous pourrez gérer tous les utilisateurs de l’application. Les utilisateurs sont affichés avec leur nom, prénom et leur login (unique à chaque utilisateur).



Modifier un utilisateur

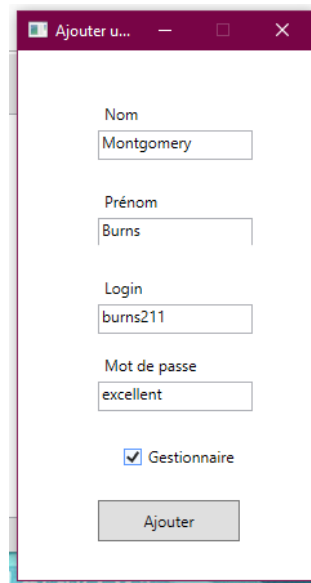
Lorsqu’un utilisateur est sélectionné dans la liste, vous pouvez modifier ses informations directement depuis les zones de textes, puis de cliquer sur “Modifier l’utilisateur sélectionné”



Dans cette image, l'utilisateur “test test” a été changé en “Karim Desbarres”

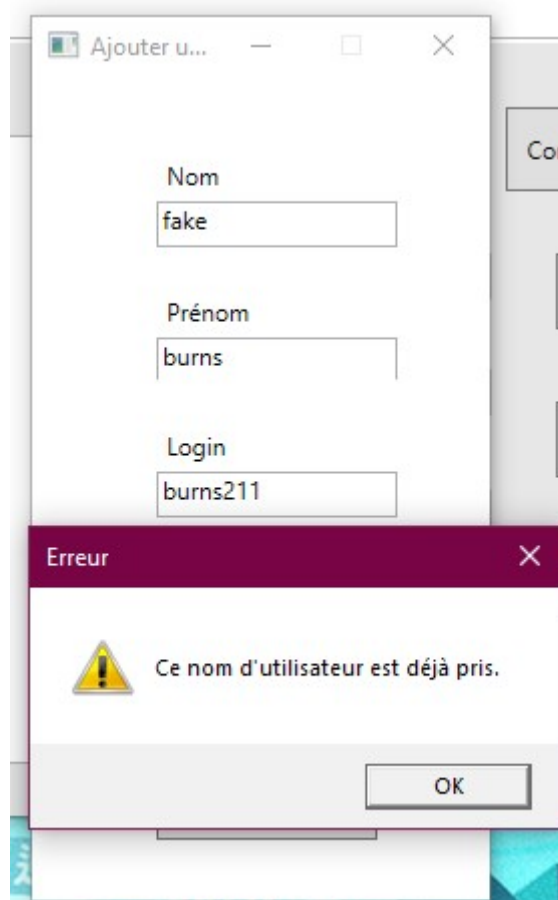
Ajouter un utilisateur

Vous pouvez cliquer sur le bouton “Ajouter un utilisateur” :

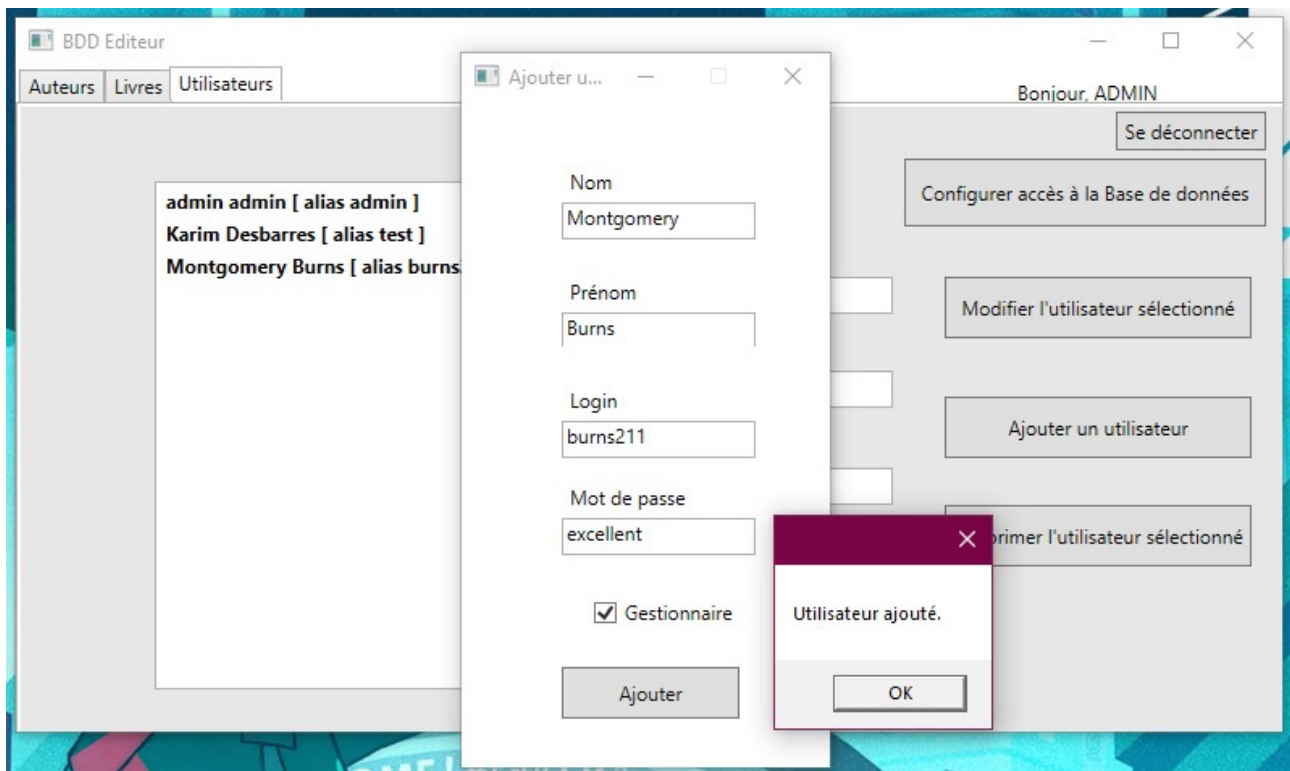


A screenshot of a Windows-style dialog box titled "Ajouter u...". It contains several text input fields: "Nom" with the value "Montgomery", "Prénom" with the value "Burns", "Login" with the value "burns211", and "Mot de passe" with the value "excellent". Below these fields is a checkbox labeled "Gestionnaire" which is checked. At the bottom of the dialog is a button labeled "Ajouter".

Attention, le login est unique à chaque utilisateur, vous ne pouvez pas utiliser celui d'un autre, auquel cas vous aurez ce message d'erreur :

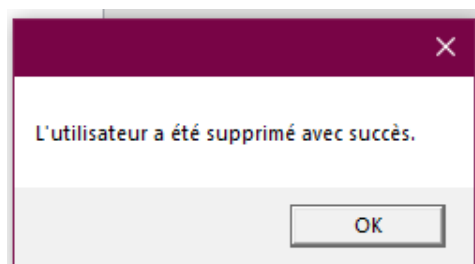


Si vous créez un utilisateur, il s'ajoute dans la liste et vous recevez ce message.



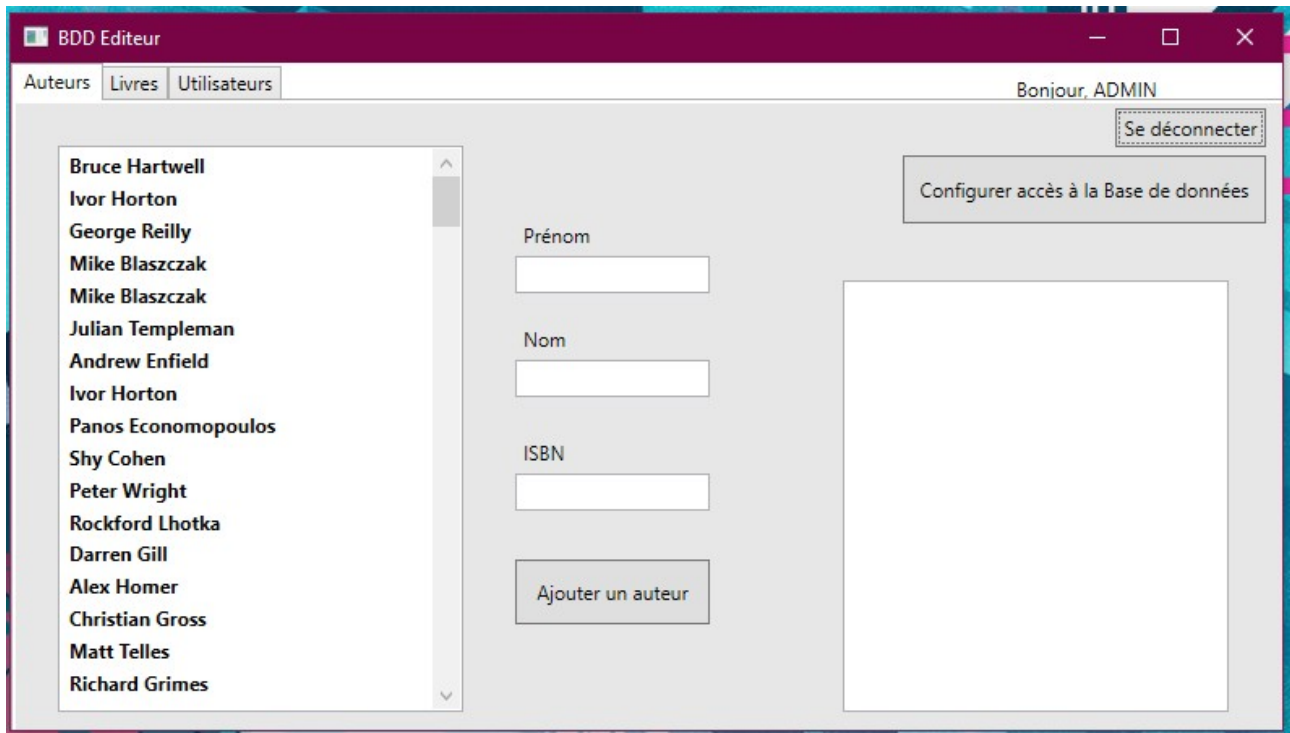
Supprimer un utilisateur

Si vous avez sélectionné un utilisateur dans la liste, et que vous cliquez sur "Supprimer l'utilisateur sélectionné", il sera alors supprimé et ce message s'affichera :



Ajouter un auteur

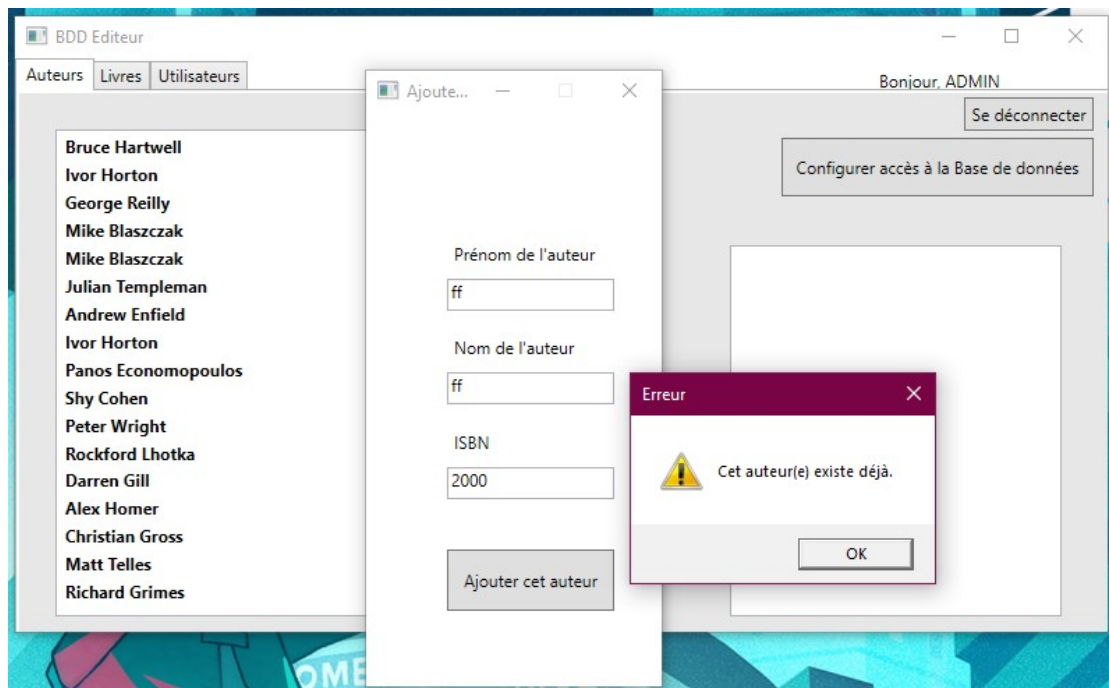
Si vous êtes connecté en tant que gestionnaire, vous verrez un nouveau bouton apparaître : “Ajouter un auteur”



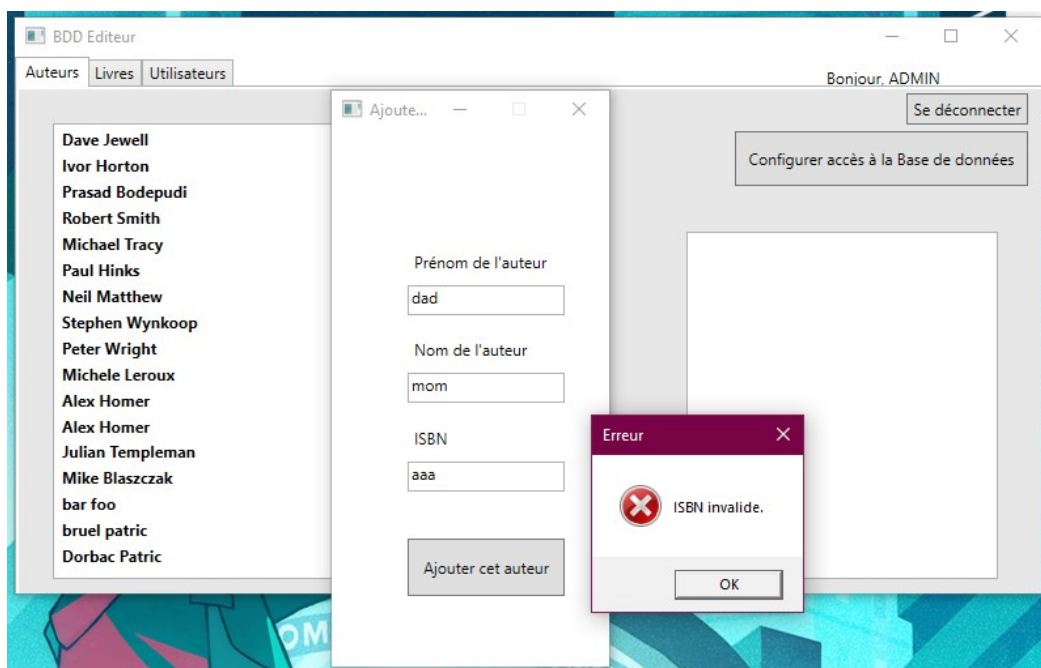
Vous verrez donc une fenêtre s'ouvrir, similaire à l'ajout d'un utilisateur.

Sur cette fenêtre, vous allez pouvoir ajouter un auteur en saisissant son nom, prénom et son ISBN. Attention, vous ne pouvez pas faire n'importe quoi, il y a quelques messages d'erreurs en fonction de certaines actions.

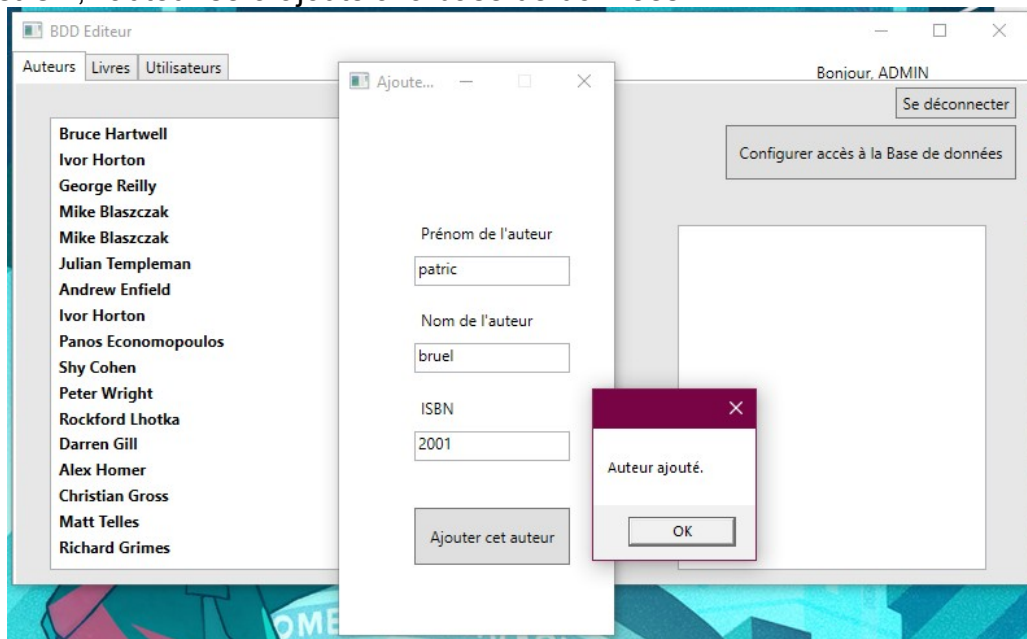
Si l'ISBN entré est déjà utilisé pour un autre auteur :



Vous ne pouvez pas non plus utiliser une chaîne de caractères pour désigner un ISBN :

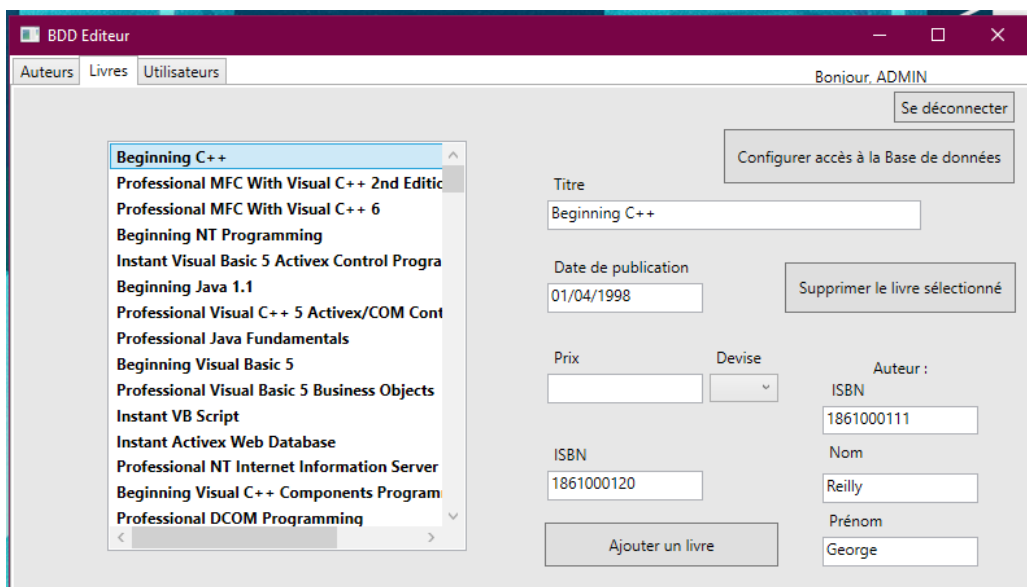


Si tout est OK, l'auteur sera ajouté à la base de données.



Ajouter un livre

Vous pouvez ajouter un livre, pour cela, il vous suffit d'être connecté en tant que gestionnaire, d'aller sur l'onglet "Livres" et de cliquer sur le bouton "Ajouter un livre".



Cette fenêtre va s'ouvrir :



Ajouter...

Titre du livre
Bien débiter à Minecraft

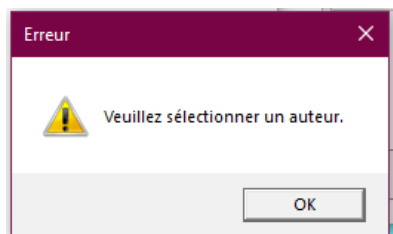
Date de publication
(format dd/mm/yyyy)
29/03/2011

Sélectionnez un auteur
Bob Fanta

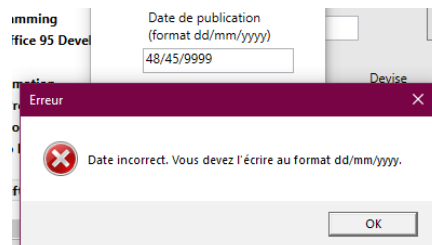
Ajouter ce livre

Évidemment, comme avec les menus d'ajout précédents, il y a quelques messages d'erreurs.

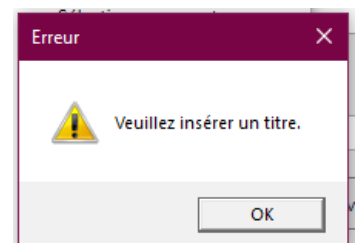
Si vous ne sélectionnez pas un auteur dans la liste



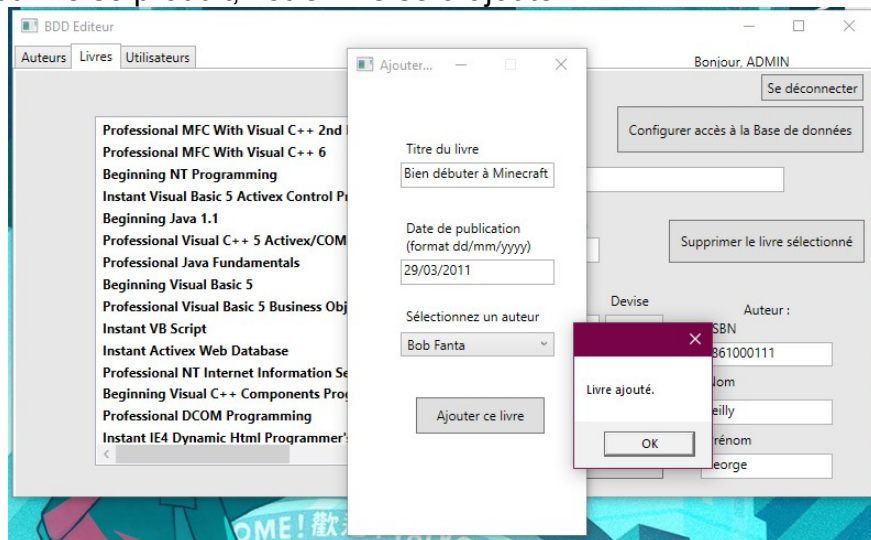
Si vous entrez une date au mauvais format



Si n'entrez pas de titre

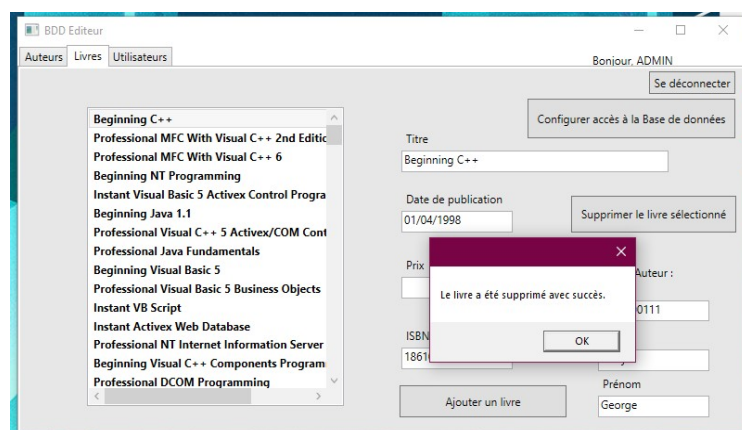


Si aucune erreur ne se produit, votre livre sera ajouté.



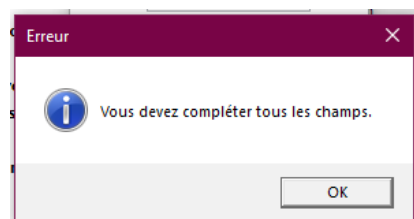
Supprimer un livre

Comme vous avez pu le remarquer sur l'écran des livres et que l'on est connecté en tant que gestionnaire, lorsque l'on sélectionne un livre dans la liste, on peut le supprimer.



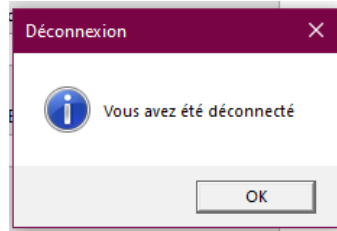
Erreur champs

Dans toutes les boîtes de dialogue, soit ajouter/modifier des utilisateurs/auteurs/livres, vous aurez toujours une boîte de dialogue qui va s'afficher si vous ne remplissez pas tous les champs.



Déconnexion

Vous pouvez cliquer sur le bouton “Se déconnecter”.



III. Conception de l'application

La classe DLL

L'application utilise Linq afin de gérer l'accès à la base de données. L'application est conçue pour fonctionner avec une référence (une DLL : Dynamic Link Library) nommée DIIBddEditeur qui va comporter tout ce qui est en rapport avec Linq et la base de données. Voici la classe métier de la DLL.

```
namespace DIIBddEditeur
{
    public class BddEditeur
    {
        private BddediteurDataContext bdd = null;
        public BddEditeur()
        {
            try
            {
                bdd = new BddediteurDataContext();
            }
            catch
            {
                throw new Exception("problème pour instancier l'attribut bdd");
            }
        }
        public BddEditeur(string serveurIp, string port, string user, string mdp)
        {
            try
            {
                bdd = new BddediteurDataContext("User Id=" + user + ";Password=" + mdp + ";Host=" + serveurIp + ";Port=" + port +
                ";Database=BddEditeur;Persist Security Info=True");
            }
            catch
            {
                throw;
            }
        }
        public List<Bookauthor> getAllAuthors()
        {
            try
            {
                return bdd.Bookauthors.ToList();
            }
            catch { return new List<Bookauthor>(); }
        }
        public List<Booklist> getAllBooks()
        {
            try
            {
                return bdd.Booklists.ToList();
            }
            catch { return new List<Booklist>(); }
        }
        public List<User> getAllUsers()
```



```
{
    try
    {
        return bdd.Users.ToList();
    }
    catch { return new List<User>(); }
}

public List<Bookprice> GetBookprices()
{
    try
    {
        return bdd.Bookprices.ToList();
    }
    catch { return new List<Bookprice>(); }
}

public bool authorExiste(string isbn)
{
    try
    {
        Bookauthor aut = bdd.Bookauthors.Where(s => s.ISBN.ToLower() == isbn.ToLower()).FirstOrDefault();
        if (aut != null)
            return true;
        return false;
    }
    catch { throw; }
}

public bool userExiste(string login)
{
    try
    {
        User user = bdd.Users.Where(s => s.Login == login).FirstOrDefault();
        if (user != null)
            return true;
        return false;
    }
    catch { throw; }
}

public Bookauthor getAuthor(string isbn)
{
    try
    {
        Bookauthor aut = bdd.Bookauthors.Where(s => s.ISBN.ToLower() == isbn.ToLower()).FirstOrDefault();
        return aut;
    }
    catch { throw; }
}

public User userConnecte(string login, string mdp)
{
    try
    {
        User user = bdd.Users.Where(s => s.Login.ToLower() == login.ToLower() && s.Mdp == mdp).FirstOrDefault();
```

```
        return user;
    }
    catch { throw; }
}

public bool addAuthor(string n, string p, string isb)
{
    bool Result;
    try
    {
        Bookauthor aut = new Bookauthor();
        aut.FirstName = n;
        aut.LastName = p;
        aut.ISBN = isb;
        bdd.Bookauthors.InsertOnSubmit(aut);
        bdd.SubmitChanges();
        Result = true;
    }
    catch { Result = false; }
    return Result;
}

public bool addUser(string n, string p, string lo, string mdp, bool auto)
{
    bool Result;
    try
    {
        User u = new User();
        u.Nom = n;
        u.Prenom = p;
        u.Login = lo;
        u.Mdp = mdp;
        u.Autorisation = auto;
        bdd.Users.InsertOnSubmit(u);
        bdd.SubmitChanges();
        Result = true;
    }
    catch { Result = false; }
    return Result;
}

public bool AddBook(string isb, string titre, DateTime dateP)
{
    bool Result;
    try
    {
        Booklist livre = new Booklist();
        livre.ISBN = isb;
        livre.Title = titre;
        livre.PublicationDate = dateP;
        bdd.Booklists.InsertOnSubmit(livre);
        bdd.SubmitChanges();
        Result = true;
    }
    catch { Result = false; }
    return Result;
}
```

```
public Bookprice getBookPrice(string isbn)
{
    return bdd.Bookprices.Where(s => s.ISBN == isbn).FirstOrDefault();
}

public List<Bookprice> getCurrencies(string isbn)
{
    return bdd.Bookprices.Where(s => s.ISBN == isbn).ToList();
}

public Bookprice getPriceFromCurrency(string isbn, string curr)
{
    return bdd.Bookprices.Where(s => s.ISBN == isbn && s.Currency == curr).FirstOrDefault();
}

public bool UpdateUser(User user, string nom, string prenom, string mdp, bool autorisation)
{
    bool result;
    try
    {
        if (user != null)
        {
            user.Nom = nom;
            user.Prenom = prenom;
            user.Mdp = mdp;
            user.Autorisation = autorisation;
            bdd.SubmitChanges();
            result = true;
        }
        else
        {
            result = false;
        }
    }
    catch
    {
        result = false;
    }
    return result;
}

public List<Booklist> getBooksFromAuthor(string isbn)
{
    var query = from book in bdd.Booklists
                join author in bdd.Bookauthors on book.ISBN equals author.ISBN
                where author.ISBN == isbn
                select book;

    return query.ToList();
}

public bool deleteUser(User user)
{
}
```

```
bool Result;
try
{
    bdd.Users.DeleteOnSubmit(user);
    bdd.SubmitChanges();
    Result = true;
}
catch { Result = false; }
return Result;
}

public bool deleteUBook(Booklist book)
{
    bool Result;
    try
    {
        bdd.Booklists.DeleteOnSubmit(book);
        bdd.SubmitChanges();
        Result = true;
    }
    catch { Result = false; }
    return Result;
}
}
```

- Visibilité des attributs : l'attribut bdd est déclaré en privé, ce qui signifie qu'il n'est pas accessible à partir de l'extérieur de la classe. Cela permet de garantir l'encapsulation et de contrôler les accès à la base de données.
- Choix des propriétés : la classe ne possède pas de propriétés, elle expose uniquement des méthodes pour récupérer les informations de la base de données et en ajouter.
- Choix des méthodes : la classe contient des méthodes pour récupérer des listes d'auteurs, de livres et d'utilisateurs à partir de la base de données, pour vérifier l'existence d'un auteur ou d'un utilisateur dans la base de données, pour récupérer un auteur ou un utilisateur à partir de son ISBN ou son login, et pour ajouter un auteur, un utilisateur ou un livre dans la base de données.
- Types de relations utilisées : la classe BddEditeur utilise Linq pour accéder à la base de données, ce qui permet d'utiliser des requêtes LINQ pour interroger la base de données et récupérer les informations. La classe utilise également des classes d'entités générées automatiquement pour représenter les tables de la base de données (Bookauthor, Booklist, User, etc.). Les relations entre les tables de la base de données sont gérées par Linq.
- Cardinalité : la classe BddEditeur est conçue pour gérer les accès en lecture et en écriture à la base de données, il n'y a donc pas de cardinalité particulière entre cette classe et les autres classes du projet.
- Résultats attendus : l'application devrait permettre de gérer les auteurs et les livres enregistrés dans la base de données d'une librairie d'éditeur. Elle devrait permettre de récupérer des listes d'auteurs, de livres et d'utilisateurs à partir de la base de données, de vérifier l'existence d'un auteur ou d'un utilisateur, de récupérer un auteur ou un utilisateur à partir de son ISBN ou de son login, et d'ajouter un auteur,

- un utilisateur ou un livre dans la base de données.
- Résultats obtenus : la classe BddEditeur fournit les fonctionnalités attendues et permet à l'application de gérer les informations de la base de données. Des modifications pourraient être apportées en fonction des besoins de l'application, mais la classe fournit une base solide pour la gestion de la base de données.

La classe principale

La classe MainWindow est la fenêtre principale de l'application et implémente l'interface utilisateur graphique (GUI) de l'application. Elle gère les événements utilisateur et appelle les méthodes appropriées pour gérer les interactions avec la base de données.

```
namespace TpBddEditeur
{
    /// <summary>
    /// Logique d'interaction pour MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {

        //entités bdd
        public static List<Bookauthor> listeauteurs;
        private static List<Booklist> listelivres;
        private static List<User> listeutilisateurs;
        public static BddEditeur bdd;

        public User utilisateur = null;
        public MainWindow()
        {
            InitializeComponent();
            this.helloUserLabel.Visibility = Visibility.Hidden;
            this.addAuthorButton.Visibility = Visibility.Hidden;
            this.addBookButton.Visibility = Visibility.Hidden;
            this.deleteBookButon.Visibility = Visibility.Hidden;
            this.deleteUserButton.Visibility = Visibility.Hidden;
            this.modifUserButton.Visibility = Visibility.Hidden;
            this.tabs.Items.Remove(this.userTab);
            try
            {
                bdd = new BddEditeur(Properties.Settings.Default.AdresseIP, Properties.Settings.Default.Port, Properties.Settings.Default.NomUtilisateur,
                Properties.Settings.Default.MotDePasse);
                listeauteurs = bdd.getAllAuthors();
                listelivres = bdd.getAllBooks();
                this.listViewAuthors.ItemsSource = listeauteurs;
                this.listViewBooks.ItemsSource = listelivres;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

```
private void listViewAuthors_SelChanged(object sender, RoutedEventArgs e)
{
    // récupérer l'auteur sélectionné dans la liste vue
    Bookauthor auteur = (Bookauthor)(sender as ListView).SelectedItem;
    if (auteur != null)
    {
        this.textBoxAuthorsFirstName.Text = auteur.FirstName;
        this.textBoxAuthorsLastName.Text = auteur.LastName;
        this.textBoxAuthorsISBN.Text = auteur.ISBN;
        List<Booklist> seslivres = bdd.getBooksFromAuthor(auteur.ISBN);
        this.authorsSelectedBooks.ItemsSource = seslivres;
    }
}

private void listViewBooks_SelChanged(object sender, RoutedEventArgs e)
{
    // récupérer le livre sélectionné dans la liste vue
    Booklist book = (Booklist)(sender as ListView).SelectedItem;
    this.textBoxBookPrice.Text = "";
    if (book != null)
    {
        if (this.utilisateur != null)
        {
            if (this.utilisateur.Autorisation == true)
            {
                this.deleteBookButon.Visibility = Visibility.Visible;
            }
        }
        this.textBoxBookISBN.Text = book.ISBN;
        this.textBoxBookDate.Text = book.PublicationDate.ToString().Split(' ')[0];
        this.textBoxBookTitle.Text = book.Title;
        List<Bookprice> bookprices = bdd.getCurrencies(book.ISBN);
        this.currenciesCombo.ItemsSource = bookprices;
        Bookauthor auteur = bdd.getAuthor(book.ISBN);
        if (auteur != null)
        {
            this.isbnAuthorFromBook.Text = auteur.ISBN;
            this.nomFromBookBox.Text = auteur.LastName;
            this.prenomFromBookBox.Text = auteur.FirstName;
        }
    }
}

private void Connect_Button(object sender, RoutedEventArgs e)
{
    ConnexionPage conn = new ConnexionPage(this);
    conn.ShowDialog();
    reloadWindow();
}

public void reloadWindow()
{
    this.reloadAuthors();
    this.reloadBooksList();
}
```

```
this.reloadUsers();
if (!listeauteurs.Any())
{
    this.textBoxAuthorsFirstName.Text = "";
    this.textBoxAuthorsLastName.Text = "";
    this.textBoxAuthorsISBN.Text = "";
}
if (!listelivres.Any())
{
    this.textBoxBookTitle.Text = "";
    this.textBoxBookDate.Text = "";
    this.textBoxBookISBN.Text = "";
    this.textBoxBookPrice.Text = "";
}
}

private void Login_Button(object sender, RoutedEventArgs e)
{
    if (this.utilisateur == null)
    {
        LoginPage conn = new LoginPage(this);
        conn.ShowDialog();
    }
    else
    {
        disconnect();
        MessageBox.Show("Vous avez été déconnecté", "Déconnexion", MessageBoxButton.OK, MessageBoxImage.Information);
    }
}

private void disconnect()
{
    this.helloUserLabel.Visibility = Visibility.Hidden;
    this.addAuthorButton.Visibility = Visibility.Hidden;
    this.addBookButton.Visibility = Visibility.Hidden;
    this.deleteBookButon.Visibility = Visibility.Hidden;
    this.deleteUserButton.Visibility = Visibility.Hidden;
    this.modifUserButton.Visibility = Visibility.Hidden;
    this.tabs.Items.Remove(this.userTab);
    this.login_button.Content = "Se connecter";
    this.utilisateur = null;
}

private void AddAuthorClick(object sender, RoutedEventArgs e)
{
    AddAuthorWindow window = new AddAuthorWindow(this);
    window.ShowDialog();
}

public void reloadAuthors()
{
    listeauteurs = bdd.getallAuthors();
    this.listViewAuthors.ItemsSource = listeauteurs;
}
```

```
public void reloadBooksList()
{
    listelivres = bdd.getallBooks();
    this.listeViewBooks.ItemsSource = listelivres;
}

public void reloadUsers()
{
    listeutilisateurs = bdd.getallUsers();
    this.usersListView.ItemsSource = listeutilisateurs;
}

public void reloadGestionnaireButton()
{
    if (this.utilisateur.Autorisation == true)
    {
        this.addAuthorButton.Visibility = Visibility.Visible;
        this.addBookButton.Visibility = Visibility.Visible;
        this.tabs.Items.Add(this.userTab);
        listeutilisateurs = bdd.getallUsers();
        this.usersListView.ItemsSource = listeutilisateurs;
    }
    else
    {
        this.tabs.Items.Remove(this.userTab);
    }
}

private void currenciesCombo_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (!this.currenciesCombo.Items.IsEmpty && this.currenciesCombo.SelectedItem != null && this.currenciesCombo.SelectedItem.ToString() != "")
    {
        Bookprice book = (Bookprice)(this.currenciesCombo as ComboBox).SelectedItem;
        this.textBoxBookPrice.Text = book.Price.ToString();
    }
}

private void addBookButton_Click(object sender, RoutedEventArgs e)
{
    AddBookWindow window = new AddBookWindow(this);
    window.ShowDialog();
}

private void modifUserButton_Click(object sender, RoutedEventArgs e)
{
    if (this.usersListView.SelectedItem != null)
    {
        if (this.nomBox.Text != "" && this.prenomBox.Text != "" && this.mdpBox.Text != "")
        {
            User user = (User)(this.usersListView as ListView).SelectedItem;
            bdd.UpdateUser(user, this.nomBox.Text, this.prenomBox.Text, this.mdpBox.Text, this.gestionnaireCheckbox.IsChecked.Value);
            MessageBox.Show("Utilisateur modifié.");
        }
        else
        {
        }
    }
}
```



```
        MessageBox.Show("Veuillez remplir tous les champs.", "Erreur", MessageBoxButton.OK, MessageBoxImage.Information);
    }
}

private void usersListView_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    // récupérer l'utilisateur sélectionné dans la liste vue
    User user = (User)(sender as ListView).SelectedItem;
    if (user != null)
    {
        this.deleteUserButton.Visibility = Visibility.Visible;
        this.modifUserButton.Visibility = Visibility.Visible;
        this.nomBox.Text = user.Nom;
        this.prenomBox.Text = user.Prenom;
        this.mdpBox.Text = user.Mdp;
        this.gestionnaireCheckbox.IsChecked = user.Autorisation;
    }
}

private void addUserButton_Click(object sender, RoutedEventArgs e)
{
    AddUserWindow window = new AddUserWindow(this);
    window.ShowDialog();
}

private void authorsSelectedBooks_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
}

private void deleteUserButton_Click(object sender, RoutedEventArgs e)
{
    User user = (User)(this.usersListView as ListView).SelectedItem;
    if (user != null && bdd.deleteUser(user))
    {
        MessageBox.Show("L'utilisateur a été supprimé avec succès.");
        this.reloadUsers();
        this.deleteBookButton.Visibility = Visibility.Hidden;
    }
    else
    {
        MessageBox.Show("Un problème est survenu...", "Erreur", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}

private void deleteBookButton_Click(object sender, RoutedEventArgs e)
{
    Booklist book = (Booklist)(this.listViewBooks as ListView).SelectedItem;
    if (book != null && bdd.deleteUBook(book))
    {
        MessageBox.Show("Le livre a été supprimé avec succès.");
        this.reloadBooksList();
        this.deleteUserButton.Visibility = Visibility.Hidden;
    }
}
```

```
else
{
    MessageBox.Show("Un problème est survenu...", "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}
```

- Les attributs de la classe MainWindow sont :
 - **listeauteurs**, une liste de Bookauthor qui stocke tous les auteurs de la base de données.
 - **listelivres**, une liste de Booklist qui stocke tous les livres de la base de données.
 - **listeutilisateurs**, une liste de User qui stocke tous les utilisateurs de l'application.
 - **bdd**, un objet de type BddEditeur qui représente la base de données d'éditeur.
 - **utilisateur**, un objet de type User qui stocke l'utilisateur actuellement connecté à l'application.
- Les propriétés de la classe MainWindow sont les contrôles de l'interface graphique (Label, Button, TextBox, etc.) qui permettent à l'utilisateur d'interagir avec l'application.
- Les méthodes de la classe MainWindow sont :
 - Le constructeur de la classe MainWindow() qui initialise les attributs de la classe et la connexion à la base de données.
 - listViewAuthors_SelChanged() qui est appelée lorsque l'utilisateur sélectionne un auteur dans la liste des auteurs. Elle affiche les informations de l'auteur sélectionné et les livres associés.
 - listViewBooks_SelChanged() qui est appelée lorsque l'utilisateur sélectionne un livre dans la liste des livres. Elle affiche les informations du livre sélectionné et les prix associés.
 - Connect_Button() qui est appelée lorsque l'utilisateur clique sur le bouton de connexion. Elle ouvre la fenêtre de connexion pour permettre à l'utilisateur de se connecter à l'application.
 - reloadWindow() qui recharge les données de la fenêtre.
 - Login_Button() qui est appelée lorsque l'utilisateur clique sur le bouton de connexion/déconnexion. Elle connecte/déconnecte l'utilisateur en fonction de son état de connexion.
- Les relations entre la classe MainWindow et les autres classes du projet sont les suivantes :
 - La classe MainWindow utilise la classe BddEditeur pour se connecter à la base de données.
 - La classe MainWindow utilise les classes Bookauthor, Booklist, User pour stocker les données de la base de données.
 - La classe MainWindow utilise les classes ConnexionPage et LoginPage pour gérer les fenêtres de connexion et de déconnexion de l'application.
 - La classe MainWindow utilise les contrôles de l'interface graphique de la classe MainWindow.xaml.

Les résultats attendus sont que l'application permette à l'utilisateur de :

- Se connecter à l'application.
- Visualiser les auteurs et les livres stockés dans la base de données.
- Ajouter ou modifier des auteurs et des livres dans la base de données.
- Supprimer des auteurs et des livres de la base de données.
- Consulter les informations de prix pour chaque livre.
- Visualiser les utilisateurs de l'application et leurs autorisations.

Les modifications éventuelles apportées à la classe MainWindow peuvent inclure :

1. Ajout de nouveaux éléments d'interface utilisateur, tels que des boutons, des zones de texte, des menus déroulants, etc. Ces éléments peuvent être utilisés pour afficher ou modifier des informations dans l'application.
2. Implémentation de nouvelles fonctionnalités de l'application. Par exemple, si l'application est un éditeur de texte, vous pouvez ajouter une fonctionnalité de recherche et de remplacement de texte.
3. Ajout de nouvelles méthodes à la classe MainWindow pour gérer les événements de l'interface utilisateur. Les événements sont des actions effectuées par l'utilisateur, telles que cliquer sur un bouton ou saisir du texte dans une zone de texte. Les méthodes associées à ces événements sont appelées pour gérer ces actions.
4. Modification des méthodes existantes de la classe MainWindow pour améliorer la logique de l'application ou pour résoudre des bogues.
5. Ajout de nouvelles classes pour gérer des fonctionnalités spécifiques de l'application. Par exemple, si l'application utilise une base de données, vous pourrez créer une classe pour gérer les opérations de la base de données.

IV. Conclusion

Conclusion du TP

En conclusion, le travail réalisé sur cette application de gestion de bibliothèque est assez satisfaisant. Toutefois, il y a quelques modifications à apporter pour améliorer son fonctionnement.

Tout d'abord, il serait préférable d'ajouter un message d'erreur lorsque la connexion au serveur échoue, afin d'informer l'utilisateur de l'erreur et de lui donner la possibilité de réessayer.

Ensuite, il y a un problème avec le contrôle de l'onglet "TabControl" qui a du mal à s'agrandir avec la fenêtre. Cela doit être corrigé pour une meilleure expérience utilisateur.

Les pages auraient également pu être concaténées pour permettre un héritage plus facile entre les différentes pages. Cela aurait simplifié la gestion de l'interface utilisateur et permis une meilleure organisation du code.

Un autre point à noter est que si les paramètres de connexion sont modifiés et qu'un auteur est sélectionné, la liste de ses livres est conservée. Cela peut causer de la confusion pour l'utilisateur et doit être corrigé.

Enfin, il a été remarqué qu'il manquait un champ pour entrer le prix lors de l'ajout d'un livre. Cette fonctionnalité doit être ajoutée pour que l'application soit plus complète.

Il y a également un autre défaut, en tant que gestionnaire, on peut supprimer n'importe quel utilisateur, même présentement l'utilisateur connecté.

En somme, cette application de gestion de bibliothèque est un projet intéressant et réalisable. Il y a encore des améliorations à apporter, mais dans l'ensemble, le travail réalisé est satisfaisant.