TP 2 – Mon Smart Agenda

Objectif du TP :

Manipuler des fichiers xaml, créer des windows, réutiliser la bibliothèque de classe du TP n°1.

Créer une smart application WPF attrayante qui gère le stockage de ses dimensions.

Etape 1 : Création de la solution

Ouvrir Visual Studio. Fichier => nouveau projet, Choisir Visual C# => Application WPF. La nommer « MyWPFAgenda».

Dans la solution nouvellement créée, ajouter la référence aux dlls « BusinessLayer», et « EntitiesLayers » créées dans le TP précédent. Le but de cette application est de permettre les options suivantes :

- Connecter l'utilisateur via un écran de connexion.
- Afficher la liste des évènements prévus classés par date.
- Afficher la liste des Artistes en représentation par ordre alphabétique.
- Afficher la liste des lieux pour lesquels au moins un évènement est programmé.
- Pour un lieu donné, afficher les évènements associés triés par date.

Bref pareil qu'au TP1 mais en beaucoup plus sympathique.

Etape 2 : Sérialisation

Créer une classe destinée au pilotage des dimensions de votre programme. Elle doit permettre de conserver, **pour chaque utilisateur**, les dimensions et la position de la fenêtre principale ainsi que de chaque fenêtre du programme.

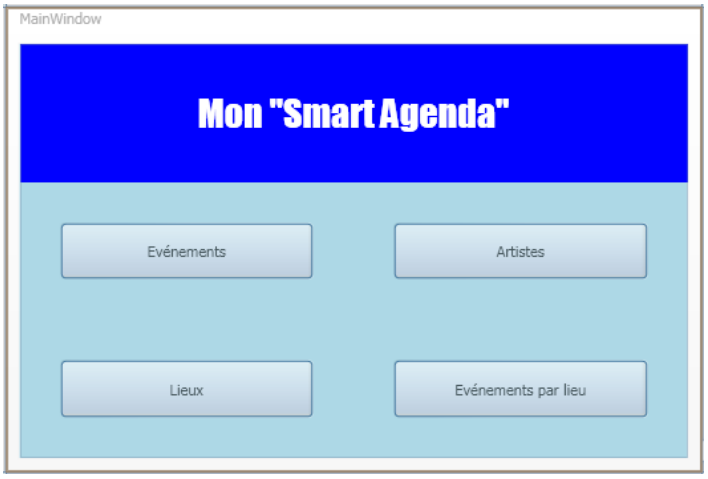
Avant de coder cette classe, il faut répondre aux questions suivantes :

- Dans quelle couche créer cette classe ?
- Quand la lecture des dimensions doit-elle être déclenchée ?
- Quand les dimensions doivent-elles être mises à jour ?
- Quand la sauvegarde des dimensions doit-elle être déclenchée ?

Etape 3 : MainWindows

Créer la fenêtre principale pour qu'elle permette l'accès aux 4 fonctionnalités qu'elle doit satisfaire. Choisir le container qui convient le mieux à une représentation aérée et customisable.

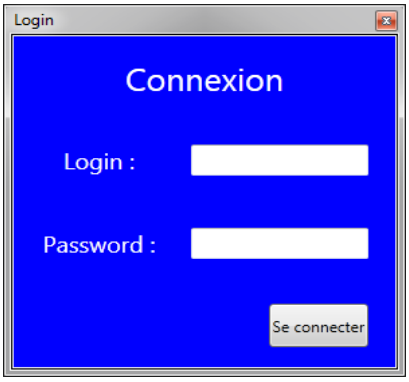
Le résultat pourra ressembler au visuel ci-dessous :



Faire en sorte que l'application se redimensionne de façon élégante. Sa taille ne doit pas être inférieure à une valeur donnée. Lors du 1e lancement, la fenêtre doit s'ouvrir au centre de l'écran.

Etape 4 : Ecran de connexion

Créer la fenêtre de connexion : Le champ password doit être de type « PasswordBox ».



Créer une nouvelle classe dans EntitiesLayer « Utilisateur » (avec nom, prenom, login et password) et créer une méthode GetUtilisateurByLogin() dans StubDataAccessLayer. La vérification du mot de passe devra se faire dans la couche BusinessLayer.

Le code derrière le bouton de connexion devrait ressembler à ça :

```
void btnConnexion_Click(object sender, RoutedEventArgs e)
{
    if (BusinessManager.CheckConnexionUser(txtLogin.Text.ToLower(), txtPassword.Password))
    {
        MainWindow win = new MainWindow();
        win.Show();

        this.Close();
    }
}
```

Enfin, pour que cet écran de login se lance au démarrage de votre application en lieu et place de la mainWindow vous devez modifier le fichier « app.xaml ».

Etape 5 : Gestion des erreurs

La gestion des erreurs en WPF est assez simple : tout se passe dans la classe app.cs :

```
/// <summary>
/// Logique d'interaction pour App.xaml
/// </summary>
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
        DispatcherUnhandledException += App_DispatcherUnhandledException;
    }

    void App_DispatcherUnhandledException(object sender, System.Windows.Threading.DispatcherUnhandledExceptionEventArgs e)
    {
        //ici votre gestion d'erreur
    }
}
```

A vous d'imaginer une gestion efficace…

Etape 6 : Gestion des fenêtres enfants

Créer un objet Window par fonctionnalité à implémenter. Les fenêtres Filles doivent s'ouvrir lors du 1^{e} lancement au centre de la fenêtre principale.

Liste des évènements classés par date

Le formulaire doit permettre d'afficher la liste des évènements prévus classés par date.

Utiliser un StackPanel pour cette fenêtre. Le formulaire pourra ressembler au modèle ci-dessous :

Gérer le redimensionnement de la fenêtre.

Remarque : Une méthode simple pour gérer l'affichage d'un événement consiste à implémenter la méthode ToString() pour la classe abstraite Evénement.

Liste des artistes en représentation

Le formulaire doit permettre d'afficher la liste des Artistes en représentation par ordre alphabétique. Utiliser un DockPanel. Le formulaire pourra ressembler au modèle ci-dessous :

Gérer le redimensionnement de la fenêtre.

Remarque : Une méthode simple pour gérer l'affichage d'un Artiste consiste à implémenter la méthode ToString() pour la classe Artiste.

Liste des lieux de représentation

Le formulaire doit permettre d'afficher la liste des lieux pour lesquels au moins un évènement est programmé.

Utiliser le container que vous préférez. Le formulaire pourra ressembler au modèle ci-dessous :

Gérer le redimensionnement de la fenêtre.

Remarque : Une méthode simple pour gérer l'affichage d'un Lieu consiste à implémenter la méthode ToString() pour la classe abstraite Artiste.

Evénements par lieu

Le formulaire doit permettre, pour une salle donnée, d'afficher les évènements associés triés par date.

Utiliser le container que vous préférez. Pour ce formulaire, il faut pouvoir choisir un lieu parmi la liste des lieux existants. Ajouter une combobox permettant le choix du lieu.

Gérer le redimensionnement de la fenêtre.

A la fin de ce TP, votre application permet d'afficher le contenu des collections issues de la base de données qui sont, pour le moment, « stubées ».