

Projet ICGO

Cahier des charges :

Définition du besoin

Définition de l'objet

L'application doit permettre d'enregistrer le catalogue des formations, l'ensemble des formateurs ainsi que les inscriptions des stagiaires aux différentes sessions de stage.

Forme de l'objet

L'application sera développée en C# avec l'environnement de développement Visual Studio auquel sera associée une base de données SQL SERVER.

Accessibilité/Sécurité

L'environnement doit être accessible aux seuls acteurs de l'entreprise. Une authentification préalable sera nécessaire pour l'accès au contenu selon les missions attribuées.

Contraintes

Architecture

L'architecture du code produit doit respecter les conventions d'usage : développement basé sur des bibliothèques de classes ou architecture en couches.

Ergonomie

Les IHM devront respecter les règles ergonomiques fournies dans le document « guidergo ».

Codage

Le document "ICGO-NormesDevelpt" présente des règles de bonnes pratiques de développement utilisées par le service informatique de I-CGO pour encadrer le développement d'applications et en faciliter la maintenance. Les éléments à fournir devront respecter le nommage des fichiers, variables et paramètres.

Fonctionnalités

L'application présente plusieurs fonctionnalités :

- gestion des agences (code fourni)
- recrutement des formateurs (code fourni)
- élaboration du catalogue des formations
- inscription des stagiaires

Documentation

La documentation devra présenter les IHM et cas d'utilisation pour chaque fonctionnalité à réaliser, le descriptif des éléments, classes et bibliothèques utilisées.

Responsabilités

Le commanditaire fournira à la demande toute information sur le contexte nécessaire à la production de l'application.

Le prestataire est à l'initiative de toute proposition technique complémentaire. Le prestataire fournira un système opérationnel, une documentation technique permettant un transfert de compétences et un mode opératoire propre à chaque fonctionnalité.

Début du projet :

Répartition des tâches :

Équipe de développeur de 2 personnes :

- Développement modèles et classes (BiblioTCGODAO et BiblioICGO) :
 - Inscription.cs (Attributs privés, Constructeurs, Accesseurs)
 - Module.cs (Attributs privés, Constructeurs, Accesseurs)
 - Session.cs (Attributs privés, Constructeurs, Accesseurs)
 - InscriptionDAO.cs + TestInscriptionDAO
 - ModuleDAO.cs + TestModuleDAO
 - SessionDAO.cs + TestSessionDAO
- Développement des vues et utilitaires (ProjetTCGO) :
 - Utilitaires.cs (Sessions stage, module)
 - Manager.cs (Sessions stage, module)
 - frmAffecterModule.cs (code)
 - frmConfirmerInscription.cs (code)
 - frmInscription.cs (code)
 - frmModule.cs (code)
 - frmSessionStage.cs (code)

Préparation de l'environnement de travail :

- Regroupement sous une seule solution en MVC "projetEquipeICGO"
- Lien de l'application avec Git pour exporter et mettre en commun sur GitHub Procédure **ici**
- Lien de la Base de donnée SQL Server à l'application (fichier connexion.cs).

```
/// <summary>
/// Ouverture de la connexion à la base de données
/// </summary>
/// </summary>
10 references
public static void OuvrirConnexion()
{
    connexion = new SqlConnection(@"Data Source=srv-tfs;Initial Catalog=ICGO_ETTBS;Integrated Security=True");
    connexion.Open();
}
```

Contenu de l'application effectué :

BiblioICGO :

-*Inscription.cs* :

-Attributs privés :

```
#region Attributs privés

private Session laSession;
private Stagiaire leStagiaire;
private string etatInscription;
#endregion
```

-Constructeurs :

```
#region Constructeurs

/// <summary>
/// Constructeur
/// </summary>
/// </summary>
0 références
public Inscription()
{
    ...
}

/// <summary>
/// Constructeur
/// </summary>
/// <param name="la_Session">La session</param>
/// <param name="le_Stagiaire">Le stagiaire</param>
/// <param name="etat_Inscription">Etat inscription</param>
0 références
public Inscription(Session la_Session, Stagiaire le_Stagiaire, string etat_Inscription)
{
    laSession = la_Session;
    leStagiaire = le_Stagiaire;
    etatInscription = etat_Inscription;
}

#endregion
```

-Accesseurs :

```
#region Accesseurs

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>La session</returns>
3 références
public Session GetLaSession()
{
    return laSession;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="uneSession">La session</param>
0 références
public void SetLaSession(Session uneSession)
{
    laSession = uneSession;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Le stagiaire</returns>
2 références
public Stagiaire GetLeStagiaire()
{
    return leStagiaire;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="unStagiaire">Le stagiaire</param>
0 références
public void SetLeStagiaire(Stagiaire unStagiaire)
{
    leStagiaire = unStagiaire;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>L'état inscription</returns>
1 référence
public string GetEtatInscription()
{
    return etatInscription;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="etat_Inscription">L'état inscription</param>
0 références
public void SetEtatInscription(string etat_Inscription)
{
    etatInscription= etat_Inscription;
}

#endregion
```

-Module.cs

-Attributs privés :

```
#region Attributs privés
/// <summary>
/// Attributs privés
/// </summary>
int numModule;
string nomModule;
string nomSupportCours;
string nomPresentation;
string placeSupportCours;
string placePresentation;
List<Stage> lesStages;
#endregion
```

-Constructeurs :

```
#region Constructeurs
/// <summary>
/// Constructeur
/// </summary>
/// <summary>
/// Constructeur
/// </summary>
public Module(int NumModule, string NomModule, string NomSupportCours, string NomPresentation, string PlaceSupportCours, string PlacePresentation)
{
    this.numModule = NumModule;
    this.nomModule = NomModule;
    this.nomSupportCours = NomSupportCours;
    this.nomPresentation = NomPresentation;
    this.placeSupportCours = PlaceSupportCours;
    this.placePresentation = PlacePresentation;
}

/// <summary>
/// Constructeur
/// </summary>
/// <summary>
/// Constructeur
/// </summary>
public Module()
{
    this.lesStages = new List<Stage>();
}

/// <summary>
/// Constructeur
/// </summary>
/// <summary>
/// Constructeur
/// </summary>
public Module(int NumModule, string NomModule, string NomSupportCours, string NomPresentation, string PlaceSupportCours, string PlacePresentation, List<Stage> lesStages)
{
    this.numModule = NumModule;
    this.nomModule = NomModule;
    this.nomSupportCours = NomSupportCours;
    this.nomPresentation = NomPresentation;
    this.placeSupportCours = PlaceSupportCours;
    this.placePresentation = PlacePresentation;
    this.lesStages = lesStages;
}
#endregion
```

-Accesseurs :

```

#region Accesseurs
/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">numModule</param>
4 références
public int GetNumModule()
{
    return numModule;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">numModule</param>
0 références
public void SetNumModule(int value)
{
    numModule = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">nomModule</param>
2 références
public string GetNomModule()
{
    return nomModule;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">nomModule</param>
0 références
public void SetNomModule(string value)
{
    nomModule = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">nomSupportCours</param>
2 références
public string GetNomSupportCours()
{
    return nomSupportCours;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">nomSupportCours</param>
0 références
public void SetNomSupportCours(string value)
{
    nomSupportCours = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">nomPresentation</param>
2 références
public string GetNomPresentation()
{
    return nomPresentation;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">nomPresentation</param>
0 références
public void SetNomPresentation(string value)
{
    nomPresentation = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">PlaceSupportCours</param>
2 références
public string GetPlaceSupportCours()
{
    return placeSupportCours;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">PlaceSupportCours</param>
0 références
public void SetPlaceSupportCours(string value)
{
    placeSupportCours = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">PlacePresentation</param>
2 références
public string GetPlacePresentation()
{
    return placePresentation;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">PlacePresentation</param>
0 références
public void SetPlacePresentation(string value)
{
    placePresentation = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <param name="value">lesStages</param>
0 références
public List<Stage> GetLesStages()
{
    return lesStages;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">lesStages</param>
0 références
public void SetLesStages(List<Stage> value)
{
    lesStages = value;
}
#endregion

```


-Session.cs

-Attributs privés :

```
#region Attributs privés

private int numSession;
private DateTime dateSession;
private Stage leStage;
private Formateur leFormateur;
private Agence lAgence;
private List<Stagiaire> lesStagiaires;
#endregion
```

-Constructeurs :

```
#region Constructeurs

/// <summary>
/// Constructeur
/// </summary>
0 références
public Session()
{
    lesStagiaires = new List<Stagiaire>();
}

/// <summary>
/// Constructeur
/// </summary>
/// <param name="numero_session">Numéro de la session</param>
/// <param name="date">La date</param>
/// <param name="le_stage">Le stage</param>
/// <param name="le_formateur">Le formateur</param>
/// <param name="lAgence">L'agence</param>
4 références
public Session(int numero_session, DateTime date, Stage le_stage, Formateur le_formateur, Agence lAgence)
{
    numSession = numero_session;
    dateSession = date;
    leStage = le_stage;
    leFormateur = le_formateur;
    lAgence = lAgence;
    lesStagiaires = new List<Stagiaire>();
}

public Session(int numero_session, DateTime date, Stage le_stage, Formateur le_formateur, Agence lAgence, List<Stagiaire> les_stagiaires)
{
    numSession = numero_session;
    dateSession = date;
    leStage = le_stage;
    leFormateur = le_formateur;
    lAgence = lAgence;
    lesStagiaires = les_stagiaires;
}

#endregion
```


-Accesseurs :

```

#Region Accesseurs

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>Le numéro de session</returns>
''' </summary>
2 références
public int GetNumSession()
{
    return numSession;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="numero_session">Le numéro de la session</param>
''' </summary>
0 références
public void SetNumSession(int numero_session)
{
    numSession = numero_session;
}

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>La date</returns>
''' </summary>
2 références
public DateTime GetDateSession()
{
    return dateSession;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="date_Session">La date</param>
''' </summary>
0 références
public void SetDateSession(DateTime date_Session)
{
    dateSession = date_Session;
}

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>Le stage</returns>
''' </summary>
4 références
public Stage GetLeStage()
{
    return leStage;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="le_Stage">Le stage</param>
''' </summary>
0 références
public void SetLeStage(Stage le_Stage)
{
    leStage = le_Stage;
}

```

```

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>L'agence</returns>
''' </summary>
2 références
public Agence GetLAgence()
{
    return lAgence;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="l_agence">L'agence</param>
''' </summary>
0 références
public void SetLAgence(Agence lAgence)
{
    lAgence = lAgence;
}

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>Le formateur</returns>
''' </summary>
2 références
public Formateur GetLeFormateur()
{
    return leFormateur;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="le_Formateur">Le formateur</param>
''' </summary>
0 références
public void SetLeFormateur(Formateur le_Formateur)
{
    leFormateur = le_Formateur;
}

''' <summary>
''' Accesseur Get
''' </summary>
''' <returns>Les stagiaires</returns>
''' </summary>
0 références
public List<Stagiaire> GetLesStagiaires()
{
    return lesStagiaires;
}

''' <summary>
''' Accesseur Set
''' </summary>
''' <param name="les_Stagiaires">Les stagiaires</param>
''' </summary>
0 références
public void SetLesStagiaires(List<Stagiaire> les_Stagiaires)
{
    lesStagiaires = les_Stagiaires;
}

#EndRegion

```

BiblioICGODAO (Liste des fonctions):

InscriptionDAO.cs :

- AjouterUneInscription(Inscription uneInscription)
 - Ajouter une inscription dans la table INSCRIRE
- ConfirmerInscription(Inscription uneInscription)
 - Confirmer une inscription : état définitif
- SupprimerUneInscription(string idCompetence, int idStage, int idSession, int idStagiaire)
 - Supprimer une inscription identifiée par une session et un stagiaire
- VerifierPlacesDisponibles(string idCompetence, int idStage, int idSession)
 - Vérifier si l'inscription d'un stagiaire à une session est possible

ModuleDAO.cs :

- AjouterUnModule(Module unModule)
 - Ajouter un module dans la table MODULE
- ChargerLesModules()
 - Charger les modules de la table MODULE dans une liste de modules
- GetModule(int idModule)
 - Retourne un module identifié par son numéro dans la table MODULE
- ModifierUnModule(Module unModule, int idModule)
 - Modifier les caractéristiques d'un module identifié par son numéro dans la table MODULE
- SupprimerUnModule(int idModule)
 - Supprimer un module identifié par son numéro dans la table MODULE
- ChargerLesModulesDuStage(string idCompetence, int idStage)
 - Charger les modules de la table COMPOSER d'un stage identifié par son code compétence et numéro stage

SessionDAO.cs :

- *AjouterUneSession(Session uneSession)*
 - Ajouter une session dans la table SESSION
- *ChargerLesSessions()*
 - Charger les sessions de la table SESSION dans une liste de sessions
- *GetSession(string idCompetence, int idStage, int idSession)*
 - Retourne une session identifiée dans la table SESSION
- *ModifierUneSession(Session uneSession, string idCompetence, int idStage, int idSession)*
 - Modifier les caractéristiques d'une session identifiée dans la table SESSION
- *SupprimerUneSession(string idCompetence, int idStage, int idSession)*
 - Supprimer une session identifiée dans la table SESSION
- *ChargerLesSessionsDuStagiaire(int idStagiaire)*
 - Charger les sessions de la table INSCRIRE d'un stagiaire identifié par son numéro
- *ChargerLesSessionsNonChoisiesDuStagiaire(int idStagiaire)*
 - Charger les sessions n'appartenant pas à la table INSCRIRE d'un stagiaire identifié par son numéro

ProjetICGO:

Utilitaires.cs :

Ajout de 2 fonctionnalités :

- *ExtraireIdSession (string unlibelleSession, out string codeCompetence, out int numStage, out int numSession)*

```
#region Session stage
3 références
static public void ExtraireIdSession(string unlibelleSession, out string codeCompetence, out int numStage, out int numSession)
{
    string[] strSession;
    string idSession;

    //Récupération dans un tableau strSession des éléments du libellé séparé par le caractère "."
    strSession = unlibelleSession.Split(new String[] { "." }, StringSplitOptions.RemoveEmptyEntries);
    // Récupération du premier élément compétence du tableau strSession
    codeCompetence = strSession[0].ToString();
    // Récupération du deuxième élément numéro stage du tableau strSession
    idSession = strSession[1].ToString();
    // Conversion en int
    numStage = int.Parse(idSession);
    numSession = int.Parse(idSession);
}
#endregion
```

- *ExtraireNumModule* (*string libelleModule*)

```
#region Module
2 références
static public int ExtraireNumModule(string libelleModule)
{
    int numModule;
    string[] strModule;
    string idModule;

    // Récupération dans un tableau strModule des éléments du libellé séparé par le caractère "."
    strModule = libelleModule.Split(new String[] { "." }, StringSplitOptions.RemoveEmptyEntries);
    // Récupération du premier élément compétence du tableau strModule
    idModule = strModule[0].ToString();
    // Conversion en int
    numModule = int.Parse(idModule);
    // Retour du résultat
    return numModule;
}

#endregion
```

Manager.cs :

Ajout de 2 fonctionnalités :

- ChargerLesSessions ([ComboBox cboSession](#))

```
#region Session stage
2 références
static public void ChargerLesSessions(ComboBox cboSession)
{
    List<Session> lesSessions = new List<Session>();

    lesSessions = SessionDAO.ChargerLesSessions();

    cboSession.SelectedIndex = -1;
    cboSession.Items.Clear();

    foreach (dynamic uneSession in lesSessions)
    {
        cboSession.Items.Add(uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() + " - " +
            uneSession.GetLeStage().GetNumStage() + " - " + uneSession.GetNumSession() + " - " + uneSession.GetLeStage().GetNomStage());
    }
}

#endregion
```

ChargerLesModules ([ComboBox cboModule](#))

```
#region Module
/// <summary>
/// Valorisation de cboFormateur
/// </summary>
/// <param name="cboFormateur">Combo cboFormateur</param>
/// </summary>
static public void ChargerLesModules(ComboBox cboModule)
{
    List<Module> lesModules = new List<Module>();

    // Recherche des formateurs dans la base de données
    lesModules = ModuleDAO.ChargerLesModules();
    // Remise à vide de cboFormateur
    cboModule.SelectedIndex = -1;
    cboModule.Items.Clear();
    // Création d'un libellé "numéro, nom prénom" et ajout dans cboFormateur pour chaque formateur
    foreach (Module unModule in lesModules)
    {
        cboModule.Items.Add(unModule.GetNumModule() + " - " + unModule.GetNomModule());
    }
}

#endregion
```

frmAffecterModule.cs :

vue :

The screenshot shows a Windows form titled "Affecter les modules". It has a light gray background. At the top left, there are three input fields: "Code compétence", "Numero stage", and "Nom". Below these is a section labeled "Choisir les modules" containing a list box labeled "lstModule". At the bottom left, there is a label "Liste des modules" above a table. The table has two columns: "Numéro" and "Libellé". At the bottom right, there are two buttons: "Ajouter" and "Fermer".

code :

```
public partial class frmAffecterModule : Form
{
    // Déclaration d'un objet dynamique qui sera soit un stage étalé soit un stage
    // groupé lors de l'exécution
    private Stage unStage;
    /// <summary>
    /// Constructeur du formulaire
    /// </summary>
    /// <param name="leStage">Stage transmis par le formulaire frmStage</param>
    public frmAffecterModule(Stage stage)
    {
        InitializeComponent();
        // Valorisation de l'objet unStage
        unStage = stage;
    }

    private void btnFermer_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

```

private void frmAffecterModule_Load(object sender, EventArgs e)
{
    txtCodeCompetence.Text = unStage.GetLaCompetence().GetCodeCompetence();
    txtNomStage.Text = unStage.GetNomStage();
    txtNumStage.Text = unStage.GetNumStage().ToString();

    foreach (Module unModule in unStage.GetLesModules())
        dgvModule.Rows.Add(unModule.GetNumModule(), unModule.GetNomModule());

    ChargerListeModules();
}

/// <summary>
/// Ajout d'un module choisi dans lstModule au stage et mise A jour du datagrid
dgvCompetence et de la base de données
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAjouter_Click(object sender, EventArgs e)
{
    int i;
    int numModule;
    Module unModule;

    for (i = 0; i <= lstModule.Items.Count - 1; i++)
    {
        // Si une compétence est sélectionnée
        if (lstModule.GetSelected(i))
        {
            numModule = int.Parse(lstModule.Items[i].ToString());
            unModule = ModuleDAO.GetModule(numModule);
            txtCodeCompetence.Text = unModule.GetNumModule().ToString() + ", " +
            unStage.GetLaCompetence().GetCodeCompetence() + ", " +
            unStage.GetNumStage().ToString();
            StageDAO.AjouterUnModule(unStage, unModule);
            // Ajout de cette compétence dans le datagrid dgvCompetence
            dgvModule.Rows.Add(numModule, unModule.GetNomModule());
        }
    }
    // Mise A jour de la liste lstCompetence
    ChargerListeModules();
}

```

```

/// <summary>
/// Valorisation de la liste lstModule
/// </summary>
private void ChargerListeModules()
{
    List<Module> lesModules = new List<Module>();
    Boolean trouve;
    int i;
    Module unModule;

    lstModule.Items.Clear();

    unStage.SetLesModules (ModuleDAO.ChargerLesModulesDuStage
(unStage.GetLaCompetence().Get CodeCompetence(), unStage.GetNumStage()));
    // Chargement de l'ensemble des modules
    lesModules = ModuleDAO.ChargerLesModules();
    // Parcours de l'ensemble des modules existantes dans la base de données
    foreach (Module leModule in lesModules)
    {
        trouve = false;
        i = 0;
        // Recherche les modules qui n'ont pas été attribuées au stage
        while ((i <= unStage.GetLesModules().Count - 1) && (!trouve))
        {
            // un module du stage
            unModule = unStage.GetLesModules()[i];
            if (leModule.GetNumModule().Equals(unModule.GetNumModule()))
            {
                trouve = true;
            }
            else
            {
                i = i + 1;
            }
        }
        // Si un module n'a pas été attribuée au stage, ajout de ce module A la
        // liste lstModule
        if (!trouve)
        {
            lstModule.Items.Add(leModule.GetNumModule());
        }
    }
}

```



```

/// <summary>
/// Suppression d'un module d'un stage en cliquant sur le bouton supprimer (X)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void dgvModule_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    DialogResult reponse;
    int numModule;
    Module unModule;
    int index;
    DataGridViewRow uneLigne;

    if (dgvModule.SelectedCells.Count == 1)
    {
        if (dgvModule.CurrentCell.ColumnIndex == 2)
        {
            reponse = MessageBox.Show("Etes vous sûr de vouloir supprimer ce module o", "Suppression d'un module", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (reponse == DialogResult.Yes)
            {
                index = dgvModule.CurrentCell.RowIndex;
                uneLigne = dgvModule.Rows[index];
                // Récupération du code compétence de la ligne sélectionnée
                numModule = int.Parse(uneLigne.Cells["colNumModule"].Value.ToString());
                unModule = ModuleDAO.GetModule(numModule);
                // Supprimer la compétence de la base de données
                StageDAO.SupprimerUnModule(unStage, unModule);
                // Supprimer la compétence du datagrid
                dgvModule.Rows.Remove(uneLigne);
                // Recharger la liste des compétences Istcompotence avec les compétences non attribuées au formateur
                ChargerListeModules();
            }
        }
    }
}

/// <summary>
/// Suppression de modules (sélection de un ou plusieurs) d'un stage par l'intermédiaire de la touche SUPPR du clavier
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void dgvModule_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
{
    DialogResult reponse;
    int numModule;
    Module unModule;

    reponse = MessageBox.Show("Etes vous sûr de vouloir supprimer ce module o", "Suppression d'un module", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
    if (reponse == DialogResult.Yes)
    {
        foreach (DataGridViewRow uneLigne in dgvModule.SelectedRows)
        {
            // Récupération du code compétence de la ligne sélectionnée

```

```

        numModule = int.Parse(uneLigne.Cells
["colNumModule"].Value.ToString());
        unModule = ModuleDAO.GetModule(numModule);
        // Supprimer la compétence de la base de données
        StageDAO.SupprimerUnModule(unStage, unModule);
    }
    // Recharger la liste des compétences Istcompete avec les compétences
    non attribuées au formateur
    ChargerListeModules();
}
}
}

```

frmConfirmerInscription.cs :

vue :

The screenshot shows a Windows-style window titled "Confirmer une inscription". Inside the window, there are two labels with corresponding dropdown menus: "Choisir un stagiaire" and "Choisir une session". Below these, there are three buttons arranged horizontally: "Confirmer", "Supprimer", and "Fermer". The window has standard Windows controls (minimize, maximize, close) in the top right corner.

code :

```

public partial class frmConfirmerInscription : Form
{
    public frmConfirmerInscription()
    {
        InitializeComponent();
    }

    private void btnFermer_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}

```

```

    /// <summary>
    /// Confirmer une inscription (état définitif)
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnConfirmer_Click(object sender, EventArgs e)
    {
        string codeCompetence;
        int numStage, numSession;

        try
        {
            numStage, out numSession);
            Session laSession = SessionDAO.GetSession(codeCompetence, numStage,
            numSession);

            int idStagiaire = Utilitaires.ExtraireNumStagiaire(cboStagiaire.Text);
            Stagiaire leStagiaire = StagiaireDAO.GetStagiaire(idStagiaire);

            Inscription uneInscription = new Inscription(laSession, leStagiaire, "p");

            InscriptionDAO.ConfirmerInscription(uneInscription);

            MessageBox.Show("Inscription Modifié", "Mise A jour réussie !",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }
    }

    private void frmConfirmerInscription_Load(object sender, EventArgs e)
    {
        // Valorisation de cboStagiaire
        Manager.ChargerLesStagiaires(cboStagiaire);
    }

    /// <summary>
    /// Valorisation de cboSession : chargement des sessions du stagiaire
    /// </summary>
    private void ChargerLesSessionsDuStagiaire()
    {
        cboSession.Items.Clear();

        int idStagiaire = Utilitaires.ExtraireNumStagiaire(cboStagiaire.Text);

        foreach (Session uneSession in
        SessionDAO.ChargerLesSessionsDuStagiaire(idStagiaire))
        {
            cboSession.Items.Add(uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() +
            ". " + uneSession.GetLeStage().GetNumStage() + ". " + uneSession.GetNumSession() + ".
            " + uneSession.GetLeStage().GetNomStage());
        }
    }
  
```

```

private void cboStagiaire_SelectedIndexChanged(object sender, EventArgs e)
{
    // Valorisation de cboSession
    ChargerLesSessionsDuStagiaire();
}

/// <summary>
/// Suppression d'une inscription
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnSupprimer_Click(object sender, EventArgs e)
{
    string codeCompetence;
    int numStage, numSession;
    int idStagiaire = Utilitaires.ExtraireNumStagiaire(cboStagiaire.Text);
    try {
        Utilitaires.ExtraireIdSession(cboSession.Text, out codeCompetence, out
numStage, out numSession);

        InscriptionDAO.SupprimerUneInscription(codeCompetence, numStage,
numSession, idStagiaire);

        MessageBox.Show("Inscription supprimé", "Mise A jour réussie !",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

}
frmInscription.cs :

```

vue :

code :

```

public partial class frmInscription : Form
{
    public frmInscription()
    {
        InitializeComponent();
    }

    private void btnFermer_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    /// <summary>
    /// Ouverture du formulaire frmStagiaire
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnStagiaire_Click(object sender, EventArgs e)
    {
        frmStagiaire fs = new frmStagiaire();
        cboStagiaire.SelectedIndex = -1;
        fs.Show();
    }

    /// <summary>
    /// Valorisation de cboSession : chargement des sessions non choisies du stagiaire
    /// </summary>
    private void ChargerLesSessionsNonChoisiesDuStagiaire()
    {
        List<Session> lesSessions = new List<Session>();
        int idStagiaire;
        idStagiaire = Utilitaires.ExtraireNumStagiaire(cboStagiaire.Text);
        lesSessions =
SessionDAO.ChargerLesSessionsNonChoisiesDuStagiaire(idStagiaire);

        cboSession.SelectedIndex = -1;
        cboSession.Items.Clear();

        foreach (Session uneSession in lesSessions)
        {
            cboSession.Items.Add(uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() +
            "- " + uneSession.GetLeStage().GetNumStage() + "- " + uneSession.GetNumSession() + "- "
            + uneSession.GetLeStage().GetNomStage());
        }
    }

    /// <summary>
    /// Ajout d'une inscription
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnAjouter_Click(object sender, EventArgs e)
    {
        int numStagiaire;
        string codeCompetence;
        int numStage;
    }
}

```

```

    int numSession;
    Stagiaire unStagiaire;
    Session uneSession;
    string unEtat;
    Inscription uneInscription;

    try
    {
        if (cboStagiaire.SelectedIndex >= 0 && cboSession.SelectedIndex >= 0)
        {
            numStagiaire = Utilitaires.ExtraireNumStagiaire(cboStagiaire.Text);
            unStagiaire = StagiaireDAO.GetStagiaire(numStagiaire);

            Utilitaires.ExtraireIdSession(cboSession.Text, out codeCompetence, out
numStage, out numSession);

            uneSession = SessionDAO.GetSession(codeCompetence, numStage,
numSession);

            unEtat = cboEtat.Text;

            if (unEtat == "Définitif")
            {
                uneInscription = new Inscription(uneSession, unStagiaire, "D");
            }
            else
            {
                uneInscription = new Inscription(uneSession, unStagiaire, "f");
            }

            if (InscriptionDAO.VerifierPlacesDisponibles(codeCompetence, numStage,
numStagiaire))
            {
                InscriptionDAO.AjouterUneInscription(uneInscription);
                MessageBox.Show("La session a bien été enregistré",
"Information !", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("Il n'y a plus de place disponible",
"Information !", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            // Remise A vide des zones : déclenchement du bouton annuler
            btnAnnuler_Click(null, EventArgs.Empty);
        }
        else
        {
            MessageBox.Show("Aucune session ou aucun stagiaire n'a été choi si it !",
"Attention !", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ajout échouée !", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

```

```

/// <summary>
/// Remise A vide de l'ensemble des zones de saisie
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAnnuler_Click(object sender, EventArgs e)
{
    cboStagiaire.SelectedIndex = -1;
    cboSession.SelectedIndex = -1;
    cboEtat.SelectedIndex = -1;
}

private void frmInscription_Load(object sender, EventArgs e)
{
    // Valorisation de cboEtat
    cboEtat.Items.Add("Défini");
    cboEtat.Items.Add("florovisoire");
    // Valorisation de cboStagiaire

    // Valorisation de cboSession
}

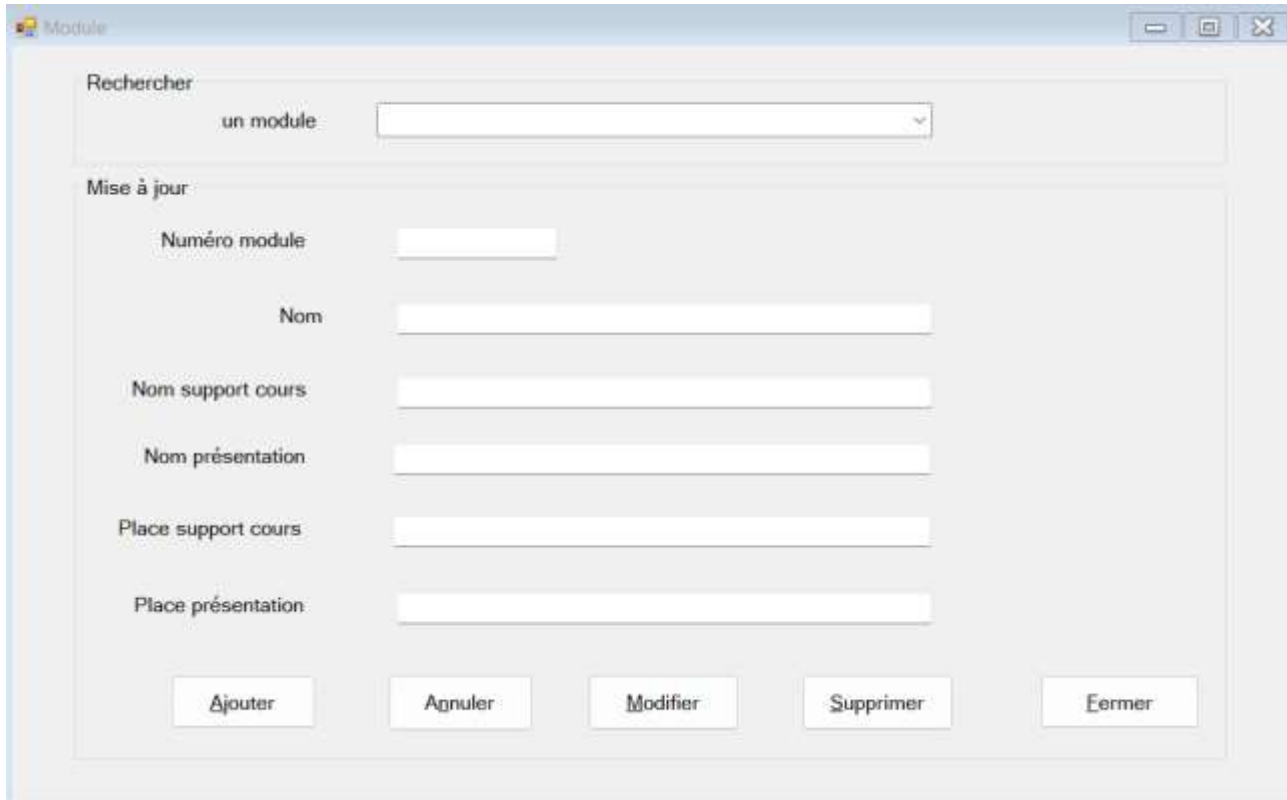
/// <summary>
/// Valorisation de cboSession après la sélection d'un stagiaire
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void cboStagiaire_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboStagiaire.SelectedIndex >= 0)
    {
        // Charger les sessions auxquelles le stagiaire n'est pas encore inscrit
        ChargerLesSessionsNonChoisiesDuStagiaire();
    }
    else
    {
        cboStagiaire.SelectedIndex = -1;
    }
}

/// <summary>
/// Valorisation de cboStagiaire sur le click de la liste
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void cboStagiaire_Click(object sender, EventArgs e)
{
    // Valorisation de cboStagiaire
    Manager.ChargerLesStagiaires(cboStagiaire);
}
}

```


frmModule.cs :

vue :



code :

```
public partial class frmModule : Form
{
    public frmModule()
    {
        InitializeComponent();
    }

    private void btnFermer_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

```
/// <summary>
/// Remise A vide de l'ensemble des zones de saisie
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAnnuler_Click(object sender, EventArgs e)
{
    cboModule.Items.Clear();
    txtNomModule.Clear();
    txtNumModule.Clear();
    txtNomSupportCours.Clear();
    txtNomfresentation.Clear();
    txtfllaceSupportCours.Clear();
    txtfllacefresentation.Clear();
    // Valorisation de cboModule
    Manager.ChargerLesModules(cboModule);
}
```

```

/// <summary>
/// Modification d'un module
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnModifier_Click(object sender, EventArgs e)
{
    int NumModule;
    string NomModule, NomSupportCours, Nomfresentation, fllaceSupportCours,
    fllacefresentation;
    Module unModule;

    // Si un module est choisi dans cboModule
    if (cboModule.SelectedIndex >= 0)
    {
        if (!int.TryParse(txtNumModule.Text, out NumModule))
        {
            MessageBox.Show("Le numéro de formateur est incorrect", "Erreur de
saisie", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            try
            {
                // Récupération du numéro de Module choisi dans cboModule
                NumModule = Utilitaires.ExtraireNumFormateur(cboModule.Text);
                // Récupération des informations des zones de saisie et ajout du
caractère ' en double si nécessaire pour construire une requête SQL
                NomModule = txtNomModule.Text;
                NomSupportCours = txtNomSupportCours.Text.Replace("'", "''");
                Nomfresentation = txtNomfresentation.Text.Replace("'", "''");
                fllaceSupportCours = txtfllaceSupportCours.Text;
                fllacefresentation = txtfllacefresentation.Text;
                // Création de l'objet unModule
                unModule = new Module(NumModule, NomModule, NomSupportCours,
Nomfresentation, fllaceSupportCours, fllacefresentation);
                // Mise A jour du Module dans la base de données
                ModuleDAO.ModifierUnModule(unModule, NumModule);
                // Valorisation de cboModule
                ModuleDAO.ChargerLesModules();
                // Valorisation de cboModule
                Manager.ChargerLesModules(cboModule);
                // Message
                MessageBox.Show("Module enregistré", "Mise A jour réussie !",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Mise A jour échouée !",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    else
    {
        MessageBox.Show("Aucun Module est choisi dans la liste", "Attention !",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

```

```

/// <summary>
/// Supprimer un module
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnSupprimer_Click(object sender, EventArgs e)
{
    int NumModule;
    DialogResult reponse;

    // Si un formateur est choisi dans cboFormateur
    if (cboModule.SelectedIndex >= 0)
    {
        reponse = MessageBox.Show("Etes vous sûr de vouloir supprimer ce Module o", "Suppression d'un Module", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        if (reponse == DialogResult.Yes)
        {
            try
            {
                // Récupération du numéro de Module choisi dans cboModule
                NumModule = Utilitaires.ExtraireNumModule(cboModule.Text);
                // Supprimer le formateur identifié dans la base de données
                ModuleDAO.SupprimerUnModule(NumModule);
                // Valorisation de cboModule
                Manager.ChargerLesModules(cboModule);
                // Message
                MessageBox.Show("Module supprimé", "Mise A jour réussie !", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show("Aucun Module choisi dans la liste", "Attention !", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
}

/// <summary>
/// Ajout d'un module
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAjouter_Click(object sender, EventArgs e)
{
    int NumModule;
    string NomModule, NomSupportCours, Nomfresentation, flaceSupportCours, flacefresentation;
    Module unModule;

```

```

        if (!int.TryParse(txtNumModule.Text, out NumModule))
        {
            MessageBox.Show("Le numéro de Module est incorrect", "Erreur de saisie",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            try
            {
                // Récupération des informations saisies et ajout du caractère ' en
double si nécessaire pour construire une requête SQL
                NomModule = txtNomModule.Text;
                NomSupportCours = txtNomSupportCours.Text.Replace("'", "''");
                NomfIrepresentation = txtNomfIrepresentation.Text.Replace("'", "''");
                fIfaceSupportCours = txtfIfaceSupportCours.Text;
                fIfacefIrepresentation = txtfIfacefIrepresentation.Text;
                // Création de l'objet unModule
                unModule = new Module(NumModule, NomModule, NomSupportCours,
NomfIrepresentation, fIfaceSupportCours, fIfacefIrepresentation);
                // Création du formateur dans la base de données
                ModuleDAO.AjouterUnModule(unModule);
                // Message
                MessageBox.Show("Module enregistré", "Mise A jour réussie !",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
                // Remise A vide des zones : déclenchement du bouton annuler
                btnAnnuler_Click(null, EventArgs.Empty);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Mise A jour échouée !",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }

    private void frmModule_Load(object sender, EventArgs e)
    {
        Manager.ChargerLesModules(cboModule);
    }

```

```

private void cboModule_SelectedIndexChanged(object sender, EventArgs e)
{
    int NumModule;
    Module unModule;

    // si sélection d'un formateur : extraction de son identifiant et recherche et
    // affichage de ses coordonnées
    if (cboModule.SelectedIndex >= 0)
    {
        NumModule = Utilitaires.ExtraireNumModule(cboModule.Text);
        unModule = ModuleDAO.GetModule(NumModule);
        txtNumModule.Text = NumModule.ToString();
        txtNomModule.Text = unModule.GetNomModule();
        txtNomfresentation.Text = unModule.GetNomfresentation();
        txtNomSupportCours.Text = unModule.GetNomSupportCours();
        txtfllaceSupportCours.Text = unModule.GetfllaceSupportCours();
        txtfllacefresentation.Text = unModule.Getfllacefresentation();
    }
    else // remise A vide des zones et affichage de la date du jour
    {
        cboModule.Items.Clear();
        txtNomModule.Clear();
        txtNumModule.Clear();
        txtNomSupportCours.Clear();
        txtNomfresentation.Clear();
        txtfllaceSupportCours.Clear();
        txtfllacefresentation.Clear();
    }
}

}
frmSessionStage.cs :

```

vue :

code :

```
public partial class frmSessionStage : Form
{
    // Booléen indiquant si une session a été choisie dans cboSession
    private bool choixSession;

    public frmSessionStage()
    {
        InitializeComponent();
    }

    private void btnFermer_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void frmSessionStage_Load(object sender, EventArgs e)
    {
        // Valorisation de cboAgence
        Manager.ChargerLesAgences(cboAgence);
        // Valorisation de cboStage
        Manager.ChargerLesStages(cboStage);
        // Valorisation de cboSession
        Manager.ChargerLesSessions(cboSession);
        // Valorisation de cboFormateur
        Manager.ChargerLesFormateurs(cboFormateur);
    }
}
```



```
/// <summary>
/// Valorisation de cboFormateur avec les formateurs compétents pour la session de
stage
/// </summary>
/// <param name="idCompetence">Code compétence</param>
private void ChargerLesFormateursCompetents(string idCompetence)
{
    List<Formateur> resultat = new List<Formateur>();
    bool competant;

    resultat.Clear();

    foreach (Formateur unFormateur in FormateurDAO.ChargerLesFormateurs())
    {
        competant = false;

        foreach(Competence uneCompetence in unFormateur.GetLesCompetences())
        {
            if(uneCompetence.GetCodeCompetence() == idCompetence)
            {
                competant = true;
            }
        }

        if(competant == true)
        {
            cboFormateur.Items.Add(unFormateur.GetNumFormateur() + ". " +
unFormateur.GetNomFormateur() + ". " + unFormateur.Getflrenom());
        }
    }
}
```

```

/// <summary>
/// Ajout d'une session de stage
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnAjouter_Click(object sender, EventArgs e)
{
    Session laSession;
    int numero_session;
    DateTime date;
    Stage le_stage = new Stage();
    Formateur le_formateur = new Formateur();
    Agence laagence = new Agence();

    try
    {
        numero_session = int.Parse(txtNumSession.Text);
        date = dtpDateSession.Value;

        foreach (Stage unStage in StageDAO.ChargerLesStages())
        {
            if(cboStage.Text == unStage.GetLaCompetence().GetCodeCompetence() + "."
            + unStage.GetNumStage() + ". " + unStage.GetNomStage())
            {
                le_stage = unStage;
            }
        }

        foreach (Formateur unFormateur in FormateurDAO.ChargerLesFormateurs())
        {
            if(cboFormateur.Text == unFormateur.GetNumFormateur() + ". " +
            unFormateur.GetNomFormateur() + ". " + unFormateur.Getflrenom())
            {
                le_formateur = unFormateur;
            }
        }

        foreach (Agence uneAgence in AgenceDAO.ChargerLesAgences())
        {
            if(cboAgence.Text == uneAgence.GetNomAgence())
            {
                laagence = uneAgence;
            }
        }

        laSession = new Session(numero_session, date, le_stage, le_formateur,
        laagence);

        SessionDAO.AjouterUneSession(laSession);

        btnAnnuler_Click(null, EventArgs.Empty);

        MessageBox.Show("Session enregistré", "Mise A jour réussie !",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }
    }

    /// <summary>
    /// Remise A vide de l'ensemble des zones de saisie
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnAnnuler_Click(object sender, EventArgs e)
    {
        choixSession = false;
        txtNumSession.Clear();
        dtpDateSession.Value = DateTime.Now;
        cboAgence.SelectedIndex = -1;
        cboFormateur.SelectedIndex = -1;
        cboSession.SelectedIndex = -1;
        cboStage.SelectedIndex = -1;
        // Valorisation de cboSession
        Manager.ChargerLesSessions(cboSession);
        // Valorisation de cboFormateur
        Manager.ChargerLesFormateurs(cboFormateur);
        // Valorisation de cboStage
        Manager.ChargerLesStages(cboStage);
    }

    /// <summary>
    /// Modification de la session
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnModifier_Click(object sender, EventArgs e)
    {
        Session laSession;
        int numero_session;
        DateTime date;
        Stage le_stage = new Stage();
        Formateur le_formateur = new Formateur();
        Agence laAgence = new Agence();

        try
        {
            numero_session = int.Parse(txtNumSession.Text);
            date = dtpDateSession.Value;

            foreach (Stage unStage in StageDAO.ChargerLesStages())
            {
                if (cboStage.Text == unStage.GetLaCompetence().GetCodeCompetence() +
                ". " + unStage.GetNumStage() + ". " + unStage.GetNomStage())
                {
                    le_stage = unStage;
                }
            }
        }
    }

```

```

        foreach (Formateur unFormateur in FormateurDAO.ChargerLesFormateurs())
        {
            if (cboFormateur.Text == unFormateur.GetNumFormateur() + ". " +
unFormateur.GetNomFormateur() + ". " + unFormateur.Getflrenom())
            {
                le_formateur = unFormateur;
            }
        }

        foreach (Agence uneAgence in AgenceDAO.ChargerLesAgences())
        {
            if (cboAgence.Text == uneAgence.GetNomAgence())
            {
                laagence = uneAgence;
            }
        }

        laSession = new Session(numero_session, date, le_stage, le_formateur,
laagence);

        SessionDAO.ModifierUneSession(laSession,
laSession.GetLeStage().GetLaCompetence().GetCodeCompetence(), le_stage.GetNumStage(),
numero_session);

        //btnAnnuler_Click(null, EventArgs.Empty);

        MessageBox.Show("Session modifiée", "Mise A jour réussie !",
MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

```

```

/// <summary>
/// Supprimer une session de stage
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnSupprimer_Click(object sender, EventArgs e)
{
    Session laSession = new Session();

    try
    {
        foreach (Session uneSession in SessionDAO.ChargerLesSessions())
        {
            if (cboSession.Text ==
uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() + "_" +
uneSession.GetLeStage().GetNumStage() + "_" + uneSession.GetNumSession() + "_" +
uneSession.GetLeStage().GetNomStage())
            {
                laSession = uneSession;
            }
        }

        SessionDAO.SupprimerUneSession(laSession.GetLeStage().GetLaCompetence().GetCodeCompetence(), laSession.GetLeStage().GetNumStage(), laSession.GetNumSession());

        btnAnnuler_Click(null, EventArgs.Empty);

        MessageBox.Show("Session supprimée", "Mise A jour réussie !",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Mise A jour échouée !", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

```

```

    /// <summary>
    /// Valorisation des zones de saisie A la sélection d'une session choisie dans
    cboSession

```

```

    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void cboSession_SelectedIndexChanged(object sender, EventArgs e)
    {
        if(cboSession.SelectedIndex != -1)
        {
            Session laSession = new Session();

            foreach (Session uneSession in SessionDAO.ChargerLesSessions())
            {
                if (cboSession.Text ==
                uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() + ". " +
                uneSession.GetLeStage().GetNumStage() + ". " + uneSession.GetNumSession() + ". " +
                uneSession.GetLeStage().GetNomStage())
                {
                    laSession = uneSession;
                }
            }

            cboFormateur.Items.Clear();
            cboStage.Items.Clear();

```

```

ChargerLesFormateursCompetents(laSession.GetLeStage().GetLaCompetence().GetCodeCompetence());

```

```

        ChargerLesStagesSession(laSession);
        txtNumSession.Text = laSession.GetNumSession().ToString();
    }

```

```

}

```

```

private void ChargerLesStagesSession(Session laSession)
{

```

```

    foreach (Session uneSession in SessionDAO.ChargerLesSessions())
    {
        if(uneSession.GetLeStage().GetNumStage() ==
        laSession.GetLeStage().GetNumStage() &&
        uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() ==
        laSession.GetLeStage().GetLaCompetence().GetCodeCompetence())
        {
            cboStage.Items.Add(uneSession.GetLeStage().GetLaCompetence().GetCodeCompetence() + ".
            " + uneSession.GetLeStage().GetNumStage() + ". " +
            uneSession.GetLeStage().GetNomStage());
        }
    }
}

```

```

    /// <summary>
    /// Valorisation des zones de saisie A la sélection d'un stage choisi dans
cboStage
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void cboStage_SelectedIndexChanged(object sender, EventArgs e)
    {
        if(cboStage.SelectedIndex != -1)
        {
            cboFormateur.Items.Clear();

            Utilitaires.ExtraireIdStage(cboStage.Text, out string codeCompetence, out int
numStage);

            Stage leStage = new Stage();

            foreach (Stage unStage in StageDAO.ChargerLesStages())
            {
                if (unStage.GetLaCompetence().GetCodeCompetence() == codeCompetence &&
unStage.GetNumStage() == numStage)
                {
                    leStage = unStage;
                }
            }

            ChargerLesFormateursCompetents(leStage.GetLaCompetence().GetCodeCompetence());
        }
    }
}

```