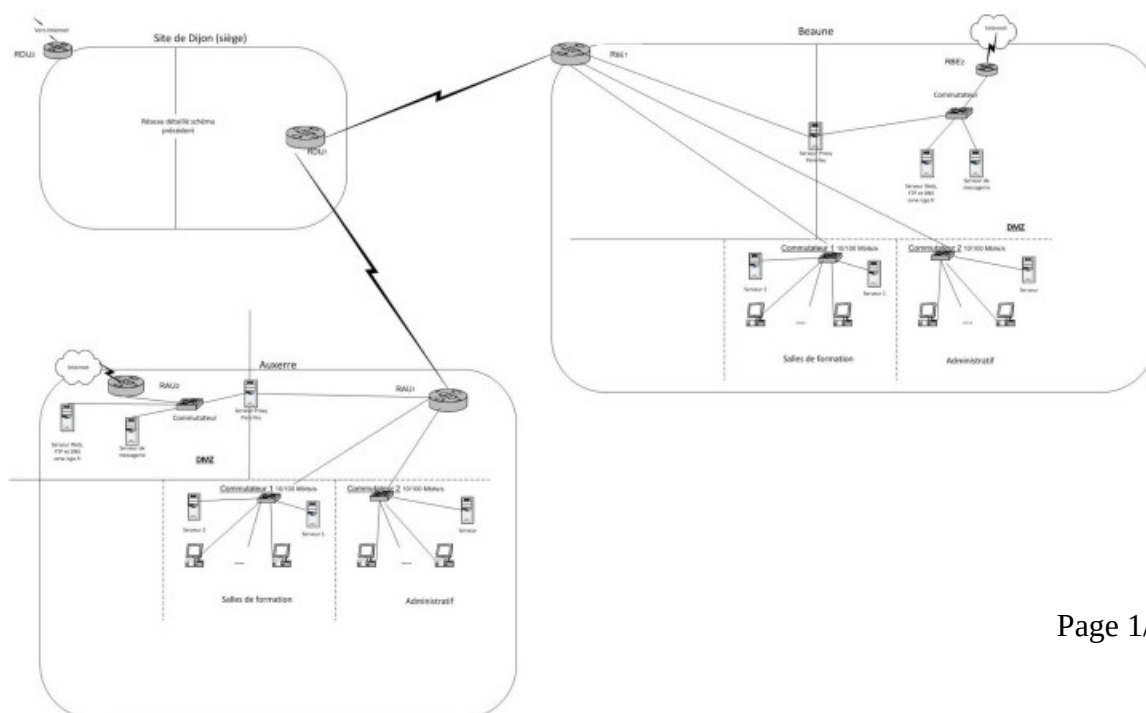
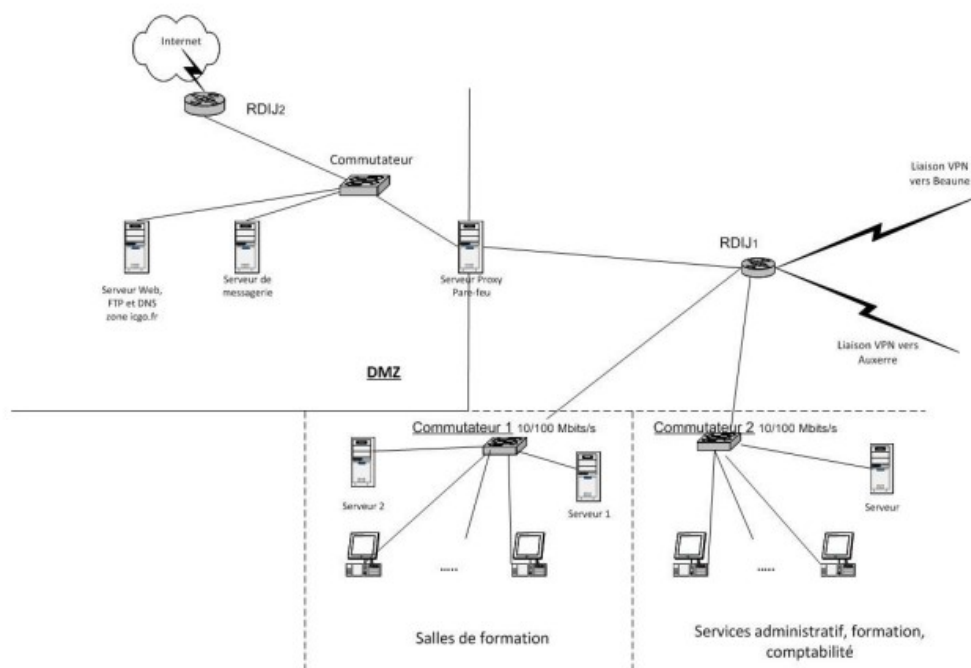


ICGO – DOCUMENTATION TECHNIQUE

➤ Contexte ICGO

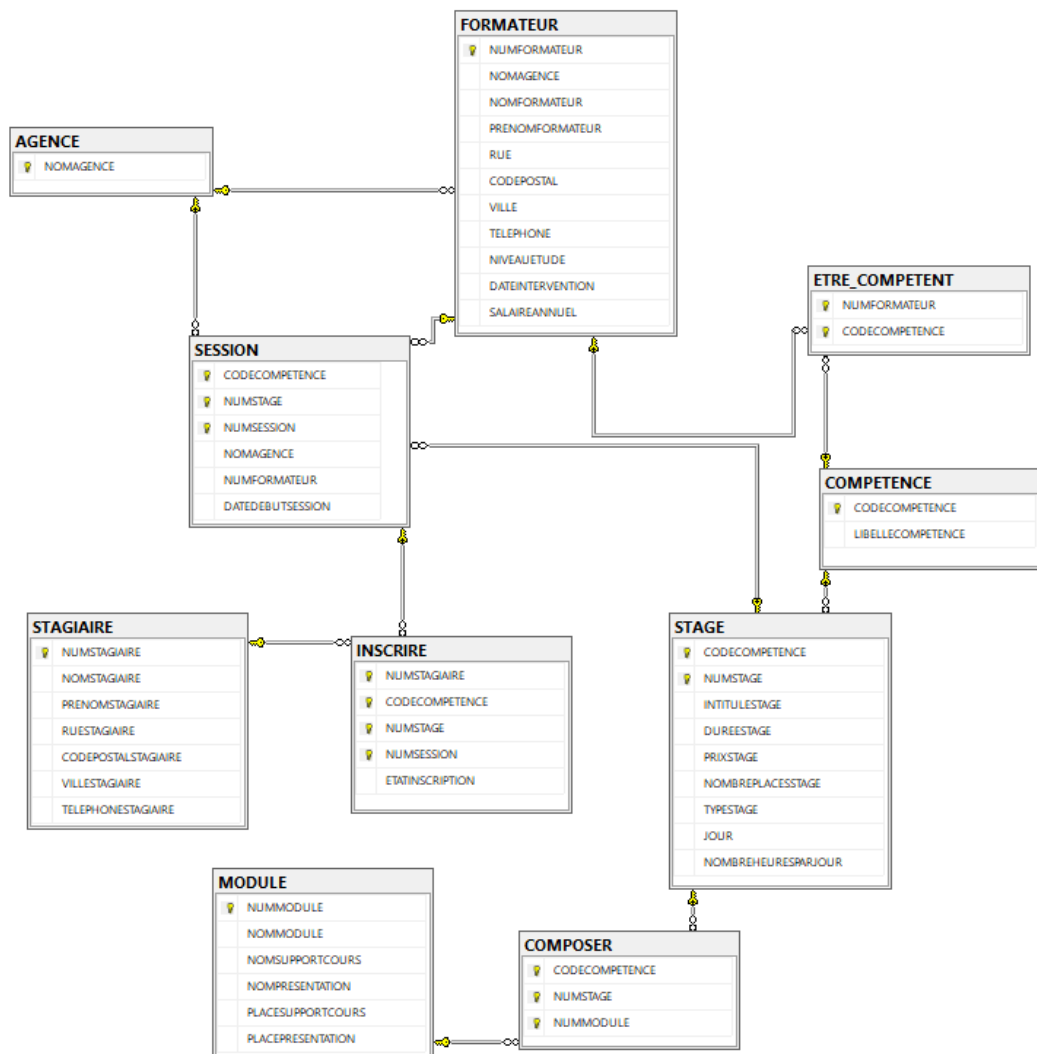
L'institut Claude Gaston Octave (I-CGO) est un organisme de formation situé à Dijon assurant des stages de formation en informatique pour des entreprises clientes et pour des particuliers. L'I-CGO est dirigé par ses trois cofondateurs, Messieurs Claude, Gaston et Octave. Souhaitant se développer et apporter un service au plus près de sa clientèle, deux agences ont été ouvertes à Beaune et à Auxerre afin de permettre le déroulement de stages. Dijon est le siège de l'entreprise.

Systèmes Informatiques :



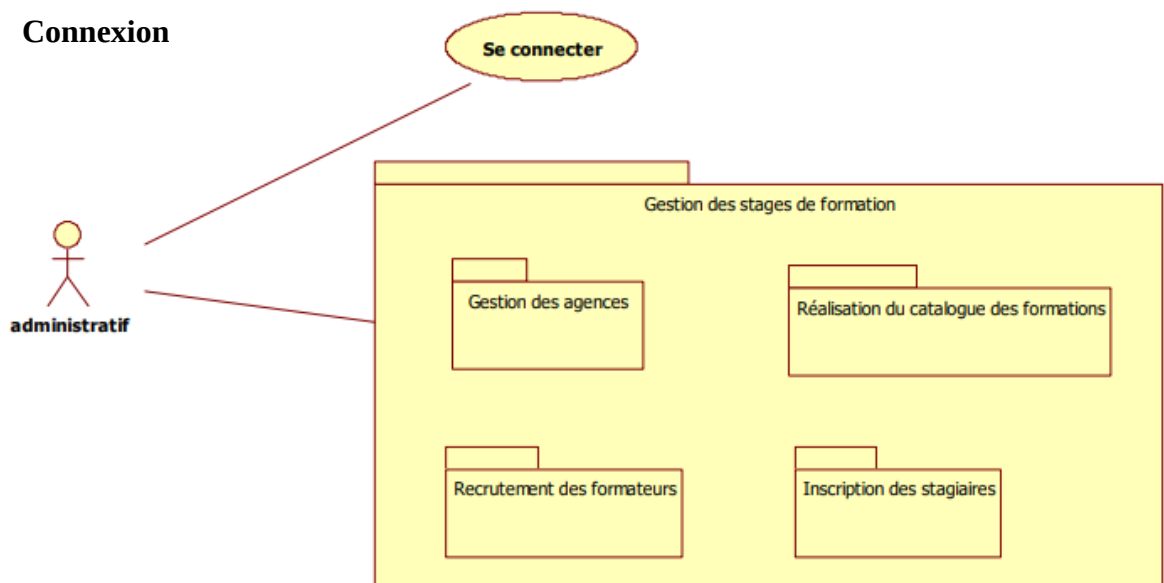
➤ Application

- **Base de donnée**



- *Traitement de l'application*

Connexion



Screenshot of the 'Connexion' window:

- Title bar: **Connexion** with standard window controls (minimize, maximize, close).
- Form fields: **Login** and **Mot de passe** (password).
- Buttons: **Connexion** and **Quitter**.

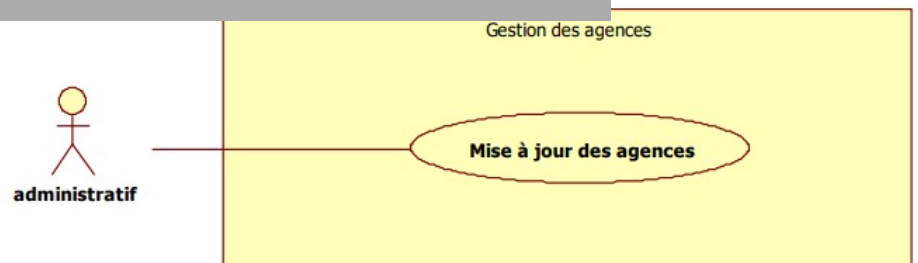
Agence

ICGO Formation

Screenshot of the 'Agence' window:

- Menu bar: **Fichier**, **Agence**, **Recrutement**, **Catalogue**, **Inscription**.
- Main area: A large grey rectangle representing the content area for managing agencies.

**Page pour
Ajouter et
supprimer des
agences**



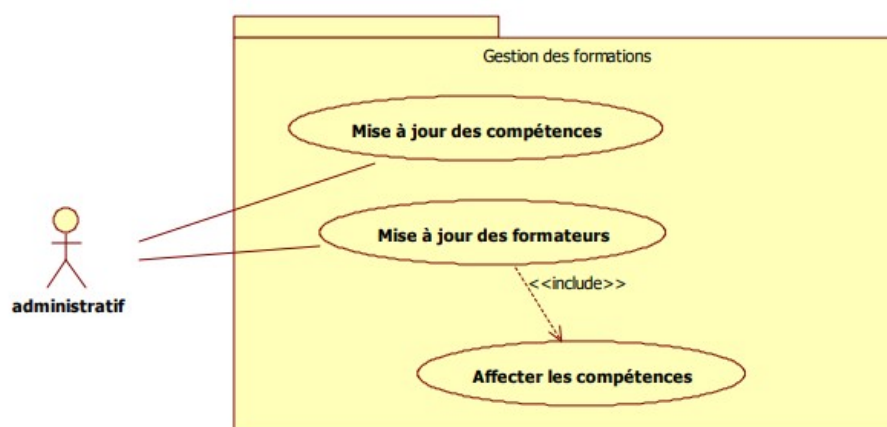
The screenshot shows a web application window titled "Agence". It contains two main sections: "Rechercher" and "Mise à jour".

In the "Rechercher" section, there is a label "Rechercher" and a text input field containing "une agence". To the right of the input field is a dropdown arrow.

In the "Mise à jour" section, there is a label "Mise à jour" and a text input field labeled "Nom".

At the bottom of the "Mise à jour" section, there are four buttons: "Ajouter", "Annuler", "Supprimer", and "Fermer".

Formations et Compétences



Compétence

Rechercher

une compétence

Mise à jour

Code compétence

Libellé

Ajouter

Annuler

Modifier

Supprimer

Fermer

Formateur

Rechercher

un formateur

Mise à

Numén

Nom

Prénom

Choisir les compétences

(

Liste des compétences

Nive

Sal

Affecter les compétences

Numéro formateur

BENARD

Claude

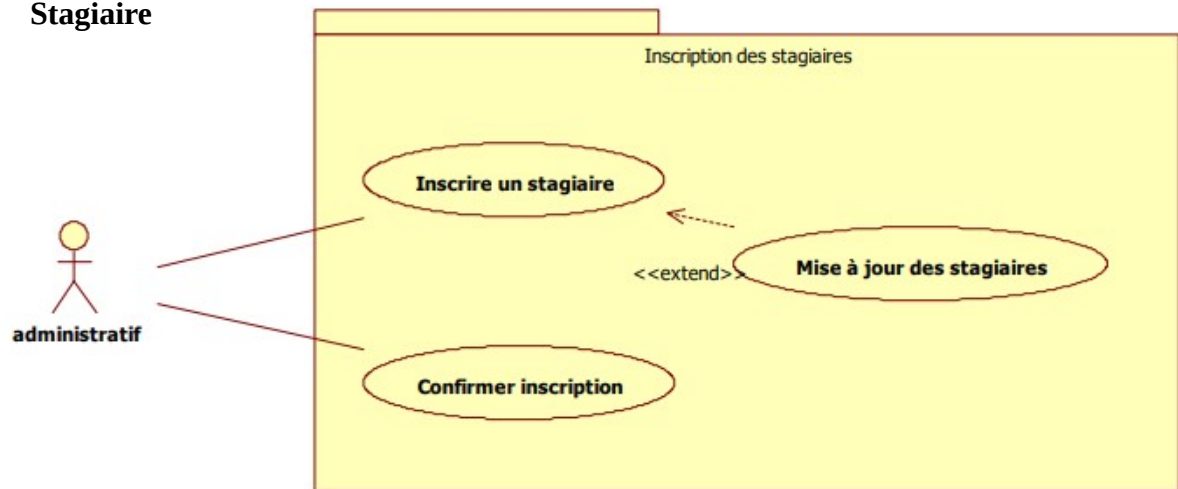
BUR
RES
SYS

	Code	Libellé	
►	BDD	Bases de données	✗
	DEV	Développements	✗
	WEB	Sites Web	✗

Ajouter

Fermer

Stagiaire



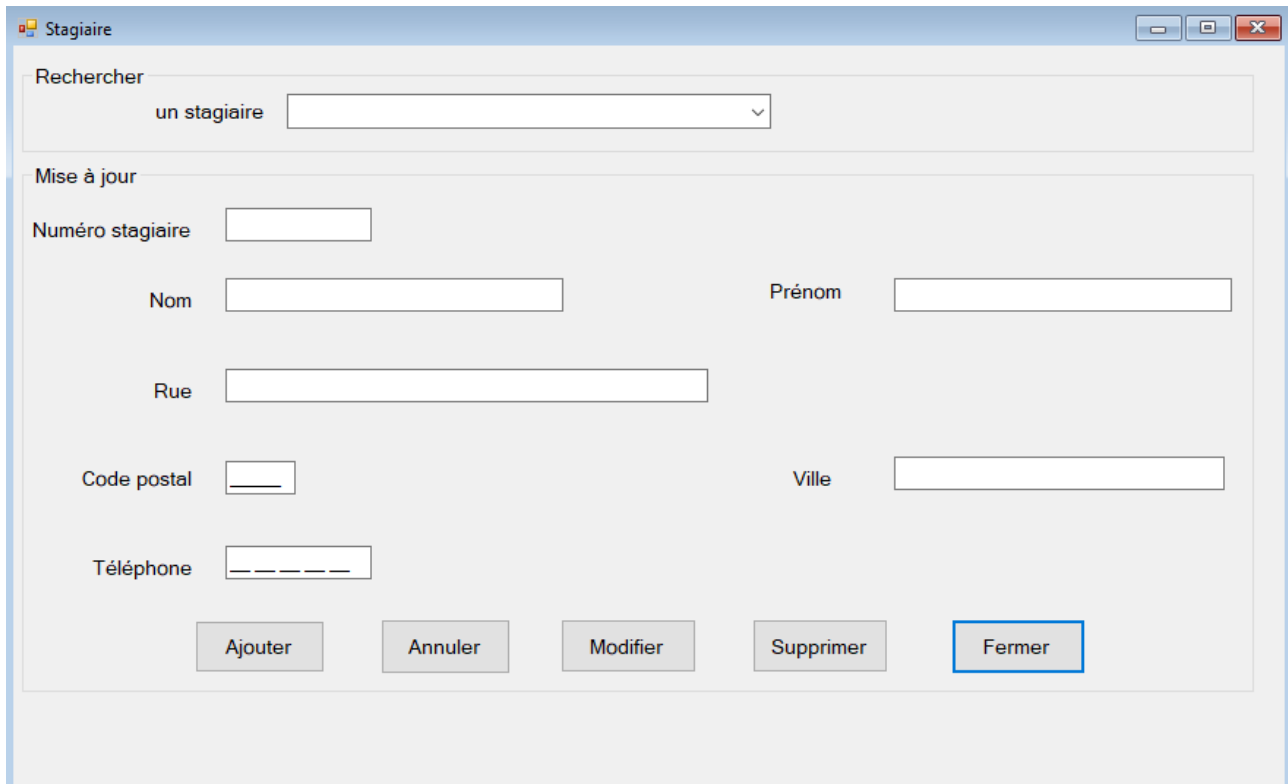
Screenshot of the "Inscription d'un stagiaire" (Intern Registration) web application interface.

The interface features three dropdown menus for selection:

- Choisir un stagiaire
- Choisir une session
- Choisir un état

Buttons are located as follows:

- Créer stagiaire**: Located next to the "Choisir un stagiaire" dropdown.
- Ajouter**: Located at the bottom left.
- Annuler**: Located at the bottom center.
- Fermer**: Located at the bottom right and is highlighted with a blue border.



The screenshot shows a window titled "Stagiaire" with standard Windows window controls. It contains two main sections: "Rechercher" and "Mise à jour".

Rechercher

un stagiaire

Mise à jour

Numéro stagiaire

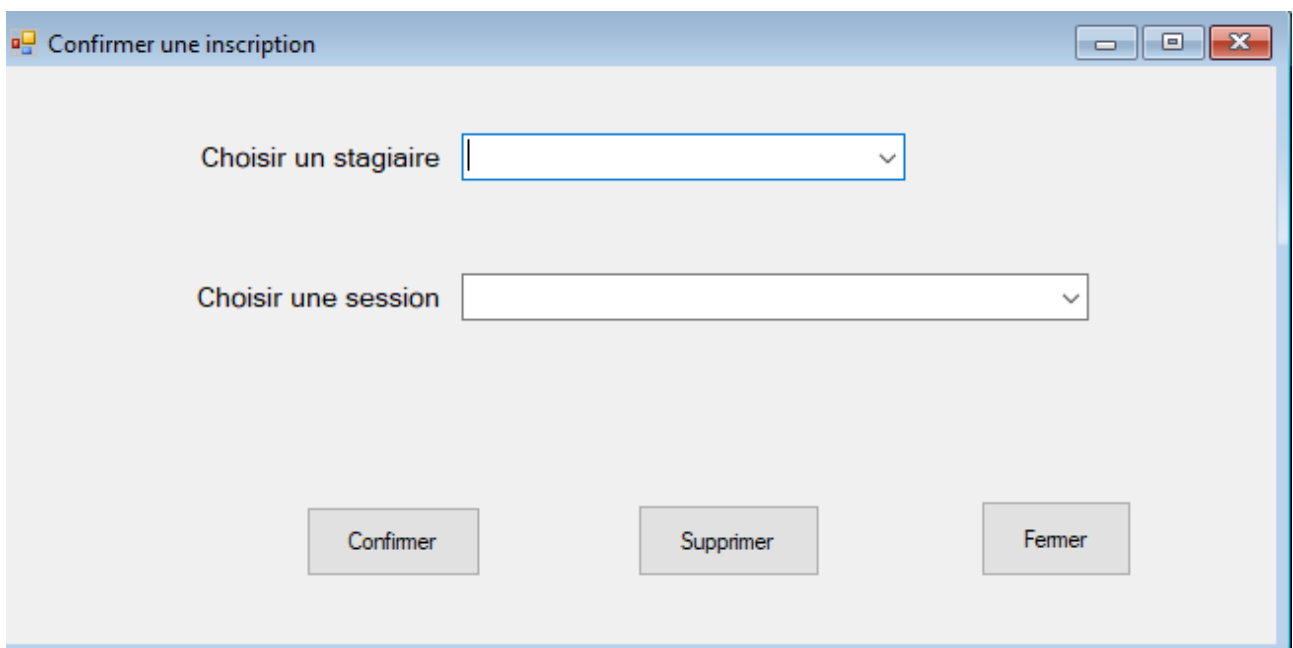
Nom Prénom

Rue

Code postal Ville

Téléphone

Ajouter Annuler Modifier Supprimer Fermer



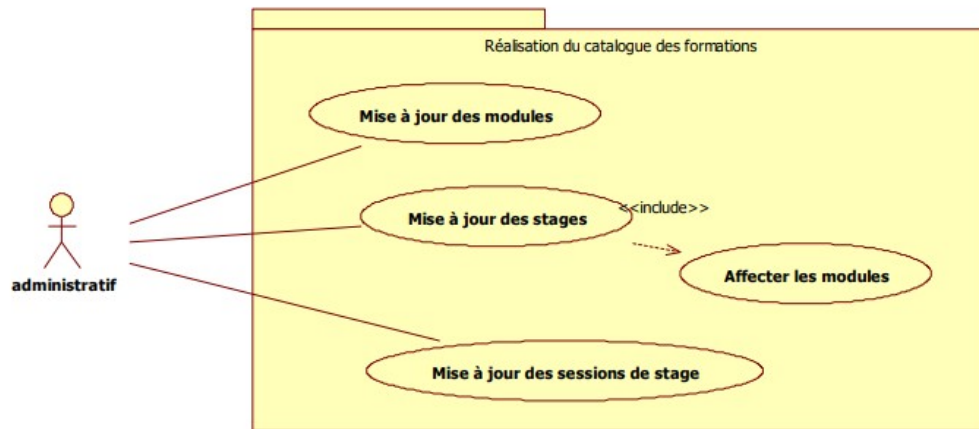
The screenshot shows a window titled "Confirmer une inscription" with standard Windows window controls. It contains two selection fields and three action buttons.

Choisir un stagiaire

Choisir une session

Confirmer Supprimer Fermer

Formations



Screenshot of the "Module" application window. The window has a title bar with standard Windows controls (minimize, maximize, close).

Rechercher

un module

Mise à jour

Número module

Nom

Nom support cours

Nom présentation

Place support cours

Place présentation

Buttons:

Stage

Rechercher un stage

Mise à jour

Code compétence

Numéro stage

Nom

Durée (nombre de jours)

Type stage

☐ Etalé

☐ Groupé

Prix

Nombre de places

Ajouter Annuler Modifier Supprimer Affecter les modules Fermer

Affecter les modules

Code compétence BDD

Numero stage 1

Nom Conception base de données

Choisir les modules

1
2
3

Liste des modules

Numéro	Libellé
--------	---------

Ajouter Fermer

Session de stage

Rechercher

une session

Mise à jour

Choisir un stage

Numéro session

Date session

jeudi 2 mai 2024

Choisir une agence

Choisir un formateur

Ajouter

Annuler

Modifier

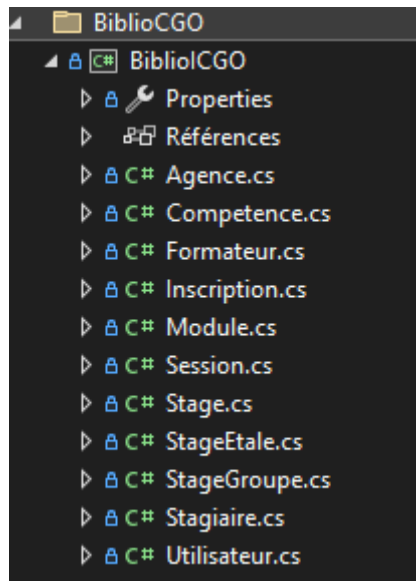
Supprimer

Fermer

➤ Code

L'application est en trois partie :

La première partie concerne les données de l'application avec tous les accesseurs, constructeurs et attribues privés.



Exemple de code avec Formateur.cs

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

namespace BiblioICGO
{
    56 références
    public class Formateur
    {
        #region Attributs privés

        private int numFormateur;
        private string nomFormateur;
        private string prenom;
        private string rue;
        private string codePostal;
        private string ville;
        private string telephone;
        private string niveauEtude;
        private DateTime dateIntervention;
        private string salaireAnnuel;
        private Agence lAgence;
        private List<Competence> lesCompetences;

        #endregion

        Constructeurs

        Accesseurs

    }
}
```

```
#region Constructeurs

/// <summary>
/// Constructeur
/// </summary>
3 références
public Formateur()
{
    lesCompetences = new List<Competence>();
}

/// <summary>
/// Constructeur
/// </summary>
/// <param name="unNumFormateur">Numéro formateur</param>
/// <param name="unNomFormateur">Nom formateur</param>
/// <param name="unPrenom">Prénom formateur</param>
/// <param name="uneRue">Rue</param>
/// <param name="unCodePostal">Code postal</param>
/// <param name="uneVille">Ville</param>
/// <param name="unTelephone">Téléphone</param>
/// <param name="unNiveauEtude">Niveau d'étude</param>
/// <param name="uneDate">Date de 1ère intervention</param>
/// <param name="unSalaire">Salaire annuel</param>
/// <param name="uneAgence">Agence de rattachement du formateur</param>
3 références
public Formateur(int unNumFormateur, string unNomFormateur, string unPrenom, string uneRue, string unCodePostal, string uneVille, string unTelephone, string unNiveauEtude, DateTime uneDateIntervention, double unSalaireAnnuel, string uneAgence)
{
    numFormateur = unNumFormateur;
    nomFormateur = unNomFormateur;
    prenom = unPrenom;
    rue = uneRue;
    codePostal = unCodePostal;
    ville = uneVille;
    telephone = unTelephone;
    niveauEtude = unNiveauEtude;
    dateIntervention = uneDate;
    salaireAnnuel = unSalaire;
    lAgence = uneAgence;
    lesCompetences = new List<Competence>();
}

}
```

```
/// <summary>
/// Constructeur
/// </summary>
/// <param name="unNumFormateur">Numéro formateur</param>
/// <param name="unNomFormateur">Nom formateur</param>
/// <param name="unPrenom">Prénom formateur</param>
/// <param name="uneRue">Rue</param>
/// <param name="unCodePostal">Code postal</param>
/// <param name="uneVille">Ville</param>
/// <param name="unTelephone">Téléphone</param>
/// <param name="unNiveauEtude">Niveau d'étude</param>
/// <param name="uneDate">Date de 1ère intervention</param>
/// <param name="unSalaire">Salaire annuel</param>
/// <param name="uneAgence">Agence de rattachement du formateur</param>
/// <param name="desCompetences">Liste des compétences attribuées au formateur</param>
2 références
public Formateur(int unNumFormateur, string unNomFormateur, string unPrenom, string uneRue, string unCodePostal, string uneVille, string unTelephone, string unNiveauEtude, DateTime uneDateIntervention, double unSalaireAnnuel, string uneAgence, List<Competence> desCompetences)
{
    numFormateur = unNumFormateur;
    nomFormateur = unNomFormateur;
    prenom = unPrenom;
    rue = uneRue;
    codePostal = unCodePostal;
    ville = uneVille;
    telephone = unTelephone;
    niveauEtude = unNiveauEtude;
    dateIntervention = uneDate;
    salaireAnnuel = unSalaire;
    lAgence = uneAgence;
    lesCompetences = new List<Competence>();
    lesCompetences = desCompetences;
}

#endregion
```

```
#region Accesseurs

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Numéro formateur</returns>
11 références
public int GetNumFormateur()
{
    return numFormateur;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Numéro formateur</param>
0 références
public void SetNumFormateur(int value)
{
    numFormateur = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Nom formateur</returns>
6 références
public string GetNomFormateur()
{
    return nomFormateur;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Nom formateur</param>
0 références
public void SetNomFormateur(string value)
{
    nomFormateur = value;
}
```

```
/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Prénom formateur</returns>
6 références
public string GetPrenom()
{
    return prenom;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Prénom formateur</param>
0 références
public void SetNom(string value)
{
    prenom = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Rue formateur</returns>
3 références
public string GetRue()
{
    return rue;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Rue formateur</param>
0 références
public void SetRue(string value)
{
    rue = value;
}
```

```
/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Code postal formateur</returns>
3 références
public string GetCodePostal()
{
    return codePostal;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Code postal formateur</param>
0 références
public void SetCodePostal(string value)
{
    codePostal = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Ville formateur</returns>
3 références
public string GetVille()
{
    return ville;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Ville formateur</param>
0 références
public void SetVille(string value)
{
    ville = value;
}
```

```
/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Telephone formateur</returns>
3 références
public string GetTelephone()
{
    return telephone;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Telephone formateur</param>
0 références
public void SetTelephone(string value)
{
    telephone = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Niveau étude formateur</returns>
3 références
public string GetNiveauEtude()
{
    return niveauEtude;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Niveau étude formateur</param>
0 références
public void SetNiveauEtude(string value)
{
    niveauEtude = value;
}
```

```
/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Date lère intervention formateur</returns>
3 références
public DateTime GetDateIntervention()
{
    return dateIntervention;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Date lère intervention formateur</param>
0 références
public void SetDateIntervention(DateTime value)
{
    dateIntervention = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Salaire annuel formateur</returns>
3 références
public string GetSalaireAnnuel()
{
    return salaireAnnuel;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Salaire annuel formateur</param>
0 références
public void SetSalaireAnnuel(string value)
{
    salaireAnnuel = value;
}
```

```
/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Agence formateur</returns>
3 références
public Agence GetLAgence()
{
    return lAgence;
}

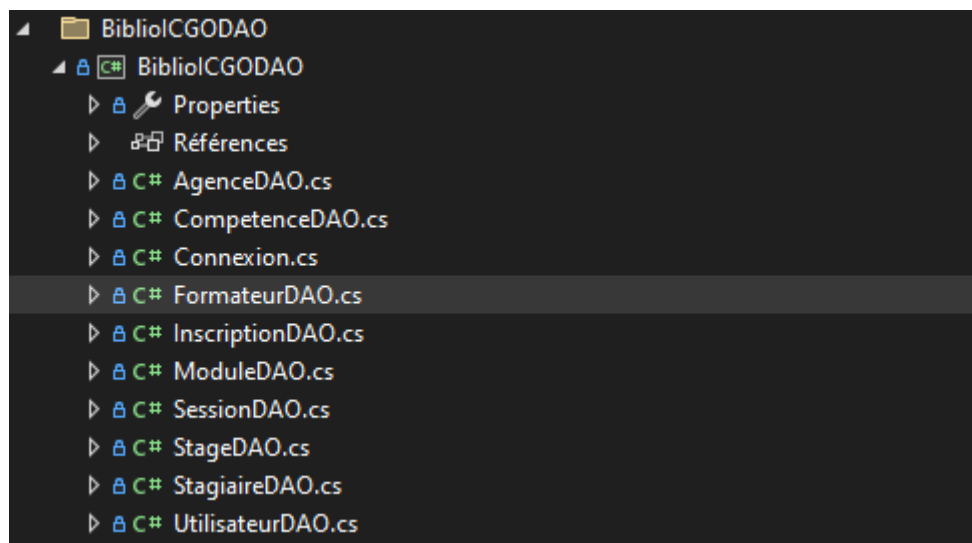
/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Agence formateur</param>
0 références
public void SetLAgence(Agence value)
{
    lAgence = value;
}

/// <summary>
/// Accesseur Get
/// </summary>
/// <returns>Liste des compétences formateur</returns>
3 références
public List<Competeence> GetLesCompetences()
{
    return lesCompetences;
}

/// <summary>
/// Accesseur Set
/// </summary>
/// <param name="value">Liste des compétences formateur</param>
1 référence
public void SetLesCompetences(List<Competeence> value)
{
    lesCompetences = value;
}

#endregion
```


La deuxième partie concerne tous les DAO, c'est à dire tous les fonctions et requêtes qui seront utilisé pour la page concerné.



Exemple de code avec FormateurDAO.cs

```
/// <summary>
/// Ajouter un formateur dans la table FORMATEUR
/// </summary>
/// <param name="unFormateur">Un formateur</param>
public static void AjouterUnFormateur(Formateur unFormateur)
{
    // Exécuter la requête d'insertion
    string requete = "INSERT INTO FORMATEUR VALUES ( " + unFormateur.GetNumFormateur() +
        ", '" + unFormateur.GetLAgence().GetNomAgence() + "', '" +
        unFormateur.GetNomFormateur() + "', '" + unFormateur.GetPrenom() + "', '" +
        unFormateur.GetRue() + "', '" + unFormateur.GetCodePostal() + "', '" +
        unFormateur.GetVille() + "', '" + unFormateur.GetTelephone() + "', '" +
        unFormateur.GetNiveauEtude() + "', '" + unFormateur.GetDateIntervention() + "', CAST("
        + unFormateur.GetSalaireAnnuel() + " AS float))";
    Connexion.ExecuterRequeteMaj(requete);
}
```



```

/// <summary>
/// Charger les formateurs de la table FORMATEUR dans une liste de formateurs
/// </summary>
/// <returns></returns>
1 référence
public static List<Formateur> ChargerLesFormateurs()
{
    List<Formateur> lesFormateurs = new List<Formateur>();
    Formateur unFormateur;
    int numFormateur;
    string nomFormateur, prenomFormateur;
    string nomAgence;
    string rue, codePostal, ville;
    string telephone, niveauEtude;
    DateTime dateIntervention;
    string salaire;
    Agence uneAgence;

    lesFormateurs.Clear();

    // Exécuter la requête de sélection
    string requete = "SELECT NUMFORMATEUR, NOMAGENCE, NOMFORMATEUR, PRENOMFORMATEUR, RUE, CODEPOSTAL, VILLE, TELEPHONE, NIVEAUETUDE, DATEINTERVENTION, SALAIREANNUEL FROM FORMATEUR";
    DataTable dt = Connexion.ExecuterRequete(requete);

    // Parcours du résultat de la requête
    foreach (DataRow uneLigne in dt.Rows)
    {
        // Récupération des caractéristiques d'un formateur à partir du résultat de la requête
        numFormateur = int.Parse(uneLigne["NUMFORMATEUR"].ToString());
        nomAgence = uneLigne["NOMAGENCE"].ToString();
        nomFormateur = uneLigne["NOMFORMATEUR"].ToString();
        prenomFormateur = uneLigne["PRENOMFORMATEUR"].ToString();
        rue = uneLigne["RUE"].ToString();
        codePostal = uneLigne["CODEPOSTAL"].ToString();
        ville = uneLigne["VILLE"].ToString();
        telephone = uneLigne["TELEPHONE"].ToString();
        niveauEtude = uneLigne["NIVEAUETUDE"].ToString();
        dateIntervention = DateTime.Parse(uneLigne["DATEINTERVENTION"].ToString());
        salaire = uneLigne["SALAIREANNUEL"].ToString();
        uneAgence = new Agence(nomAgence);
        // Construction de l'objet unFormateur avec chargement des compétences attribuées à ce formateur
        unFormateur = new Formateur(numFormateur, nomFormateur, prenomFormateur, rue, codePostal, ville, telephone, niveauEtude, dateIntervention, salaire, uneAgence, CompetenceDAO.ChargerLesCompetencesDuFormateur(numFormateur));
        // Ajout du formateur dans la liste lesFormateurs
        lesFormateurs.Add(unFormateur);
    }

    return lesFormateurs;
}

```

```

/// <summary>
/// Retourne un formateur identifié par son numéro dans la table FORMATEUR
/// </summary>
/// <param name="idFormateur">Numéro formateur</param>
/// <returns></returns>
9 références
public static Formateur GetFormateur(int idFormateur)
{
    Formateur unFormateur;
    string nomFormateur, prenomFormateur;
    string nomAgence;
    string rue, codePostal, ville;
    string telephone, niveauEtude;
    DateTime dateIntervention;
    string salaire;
    Agence uneAgence;

    // Recherche du formateur identifié par son numéro dans la table FORMATEUR
    string requete = "SELECT NOMAGENCE, NOMFORMATEUR, PRENOMFORMATEUR, RUE, CODEPOSTAL, VILLE, TELEPHONE, NIVEAUETUDE, DATEINTERVENTION, SALAIREANNUEL FROM FORMATEUR WHERE NUMFORMATEUR = " + idFormateur;
    DataTable dt = Connexion.ExecuterRequete(requete);

    if (dt.Rows.Count == 1)
    {
        nomAgence = dt.Rows[0]["NOMAGENCE"].ToString();
        nomFormateur = dt.Rows[0]["NOMFORMATEUR"].ToString();
        prenomFormateur = dt.Rows[0]["PRENOMFORMATEUR"].ToString();
        rue = dt.Rows[0]["RUE"].ToString();
        codePostal = dt.Rows[0]["CODEPOSTAL"].ToString();
        ville = dt.Rows[0]["VILLE"].ToString();
        telephone = dt.Rows[0]["TELEPHONE"].ToString();
        niveauEtude = dt.Rows[0]["NIVEAUETUDE"].ToString();
        dateIntervention = DateTime.Parse(dt.Rows[0]["DATEINTERVENTION"].ToString());
        salaire = dt.Rows[0]["SALAIREANNUEL"].ToString();
        uneAgence = new Agence(nomAgence);
        // Construction de l'objet unFormateur avec chargement des compétences attribuées à ce formateur
        unFormateur = new Formateur(idFormateur, nomFormateur, prenomFormateur, rue, codePostal, ville, telephone, niveauEtude, dateIntervention, salaire, uneAgence, CompetenceDAO.ChargerLesCompetencesDuFormateur(idFormateur));
    }
    else
    {
        unFormateur = new Formateur();
    }

    return unFormateur;
}

```

```

/// <summary>
/// Modifier les caractéristiques d'un formateur identifié par son numéro dans la table FORMATEUR
/// </summary>
/// <param name="unFormateur">Un formateur</param>
/// <param name="idFormateur">Numéro formateur</param>
1 référence
public static void ModifierUnFormateur(Formateur unFormateur, int idFormateur)
{
    // Exécution de la requête de modification
    string requete = "UPDATE FORMATEUR SET NUMFORMATEUR = " + unFormateur.GetNumFormateur() + ", NOMAGENCE = " + unFormateur.GetNomAgence().GetNomAgence() + ", NOMFORMATEUR = " + unFormateur.GetNomFormateur() + ", PRENOMFORMATEUR = " + unFormateur.GetPrenomFormateur() + " WHERE NUMFORMATEUR = " + idFormateur;
    Connexion.ExecuterRequeteMaj(requete);
}

/// <summary>
/// Supprimer un formateur identifié par son numéro dans la table FORMATEUR
/// </summary>
/// <param name="idFormateur">Numéro formateur</param>
1 référence
public static void SupprimerUnFormateur(int idFormateur)
{
    // Exécution de la requête de suppression
    string requete = "DELETE FROM FORMATEUR WHERE NUMFORMATEUR = " + idFormateur;
    Connexion.ExecuterRequeteMaj(requete);
}

/// <summary>
/// Ajouter une compétence à un formateur dans la table ETRE_COMPETENT
/// </summary>
/// <param name="unFormateur">Un formateur</param>
/// <param name="uneCompetence">Une compétence</param>
1 référence
public static void AjouterUneCompetence(Formateur unFormateur, Competence uneCompetence)
{
    // Exécution de la requête d'insertion
    string requete = "INSERT INTO ETRE_COMPETENT VALUES ( " + unFormateur.GetNumFormateur() + ", " + uneCompetence.GetCodeCompetence() + " )";
    Connexion.ExecuterRequeteMaj(requete);
}

```

```

/// <summary>
/// Supprimer une compétence à un formateur dans la table ETRE_COMPETENT
/// </summary>
/// <param name="unFormateur">Un formateur</param>
/// <param name="uneCompetence">Une compétence</param>
2 références
public static void SupprimerUneCompetence(Formateur unFormateur, Competence uneCompetence)
{
    // Exécuter la requête de suppression
    string requete = "DELETE FROM ETRE_COMPETENT WHERE NUMFORMATEUR = " + unFormateur.GetNumFormateur() + " AND CODECOMPETENCE = " + uneCompetence.GetCodeCompetence() + " ";
    Connexion.ExecuterRequeteMaj(requete);
}

```

```

/// <summary>
/// Charger les formateurs de la table ETRE_COMPETENT d'une compétence identifié par son code
/// </summary>
/// <param name="idCompetence">Code compétences</param>
/// <returns></returns>
3 références
public static List<Formateur> ChargerLesFormateursCompetents(string idCompetence)
{
    List<Formateur> lesFormateurs = new List<Formateur>();
    Formateur unFormateur;
    int numFormateur;
    string nomFormateur, prenomFormateur;
    string nomAgence;
    string rue, codePostal, ville;
    string telephone, niveauEtude;
    DateTime dateIntervention;
    string salaire;
    Agence uneAgence;

    lesFormateurs.Clear();

    // Recherche des formateurs de la compétence
    string requete = "SELECT F.NUMFORMATEUR, NOMAGENCE, NOMFORMATEUR, PRENOMFORMATEUR, RUE, CODEPOSTAL, VILLE, TELEPHONE, NIVEAUETUDE, DATEINTERVENTION, SALAIREANNUEL FROM FORMATEUR AS F INNER JOIN ETRE_COMPETENT AS E ON F.NUMFORMATEUR = E.NUMFORMATEUR WHERE E.CODECOMPETENCE = " + idCompetence;
    DataTable dt = Connexion.ExecuterRequete(requete);

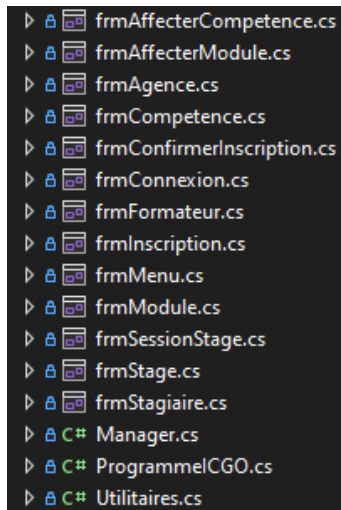
    foreach (DataRow uneLigne in dt.Rows)
    {
        // Récupération des caractéristiques d'un formateur à partir du résultat de la requête
        numFormateur = int.Parse(uneLigne["NUMFORMATEUR"].ToString());
        nomAgence = uneLigne["NOMAGENCE"].ToString();
        nomFormateur = uneLigne["NOMFORMATEUR"].ToString();
        prenomFormateur = uneLigne["PRENOMFORMATEUR"].ToString();
        rue = uneLigne["RUE"].ToString();
        codePostal = uneLigne["CODEPOSTAL"].ToString();
        ville = uneLigne["VILLE"].ToString();
        telephone = uneLigne["TELEPHONE"].ToString();
        niveauEtude = uneLigne["NIVEAUETUDE"].ToString();
        dateIntervention = DateTime.Parse(uneLigne["DATEINTERVENTION"].ToString());
        salaire = uneLigne["SALAIREANNUEL"].ToString();
        uneAgence = new Agence(nomAgence);
        // Création de l'objet unFormateur et ajout dans la liste des formateurs
        unFormateur = new Formateur(numFormateur, nomFormateur, prenomFormateur, rue, codePostal, ville, telephone, niveauEtude, dateIntervention, salaire, uneAgence);
        lesFormateurs.Add(unFormateur);
    }

    return lesFormateurs;
}

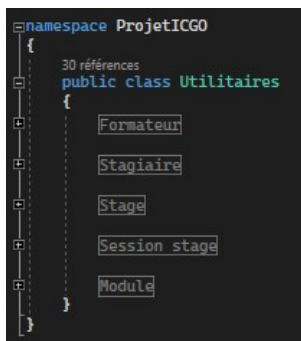
```

La troisième partie concerne l'interface de l'application avec les actions correspondant à chaque bouton de l'application, on ajoute aussi dans cette partie les pages Manager.cs, Utilitaire.cs et ProgrammellICGO.cs.

Exemple avec la page frmFormateur.cs



et la page Utilitaire.cs



```
#region Formateur

/// <summary>
/// Extraction du numéro formateur à partir du libellé choisi dans un comboBox
/// </summary>
/// <param name="unLibelleFormateur"></param>
/// <returns></returns>
7 références
static public int ExtraireNumFormateur(string unLibelleFormateur)
{
    int numFormateur;
    string[] strFormateur;
    string idFormateur;

    // Récupération dans un tableau strFormateur des éléments du libellé séparé par le caractère "."
    strFormateur = unLibelleFormateur.Split(new String[] { "." }, StringSplitOptions.RemoveEmptyEntries);
    // Récupération du premier élément du tableau strFormateur
    idFormateur = strFormateur[0].ToString();
    // Conversion de l'élément en valeur de type int
    numFormateur = int.Parse(idFormateur);
    // Retour du résultat
    return numFormateur;
}

#endregion
```

```
#region Stagiaire

/// <summary>
/// Extraction du numéro stagiaire à partir du libellé choisi dans un comboBox
/// </summary>
/// <param name="unLibelleStagiaire"></param>
/// <returns></returns>
8 références
static public int ExtraireNumStagiaire(string unLibelleStagiaire)
{
    int numStagiaire;
    string[] strStagiaire;
    string idStagiaire;

    // Récupération dans un tableau strStagiaire des éléments du libellé séparé par le caractère "."
    strStagiaire = unLibelleStagiaire.Split(new String[] { "." }, StringSplitOptions.RemoveEmptyEntries);
    // Récupération du premier élément du tableau strStagiaire
    idStagiaire = strStagiaire[0].ToString();
    // Conversion de l'élément en valeur de type int
    numStagiaire = int.Parse(idStagiaire);
    // Retour du résultat
    return numStagiaire;
}

#endregion
```

```
#region Stage

/// <summary>
/// Extraction de l'identifiant stage (code compétence, numéro stage) à partir du libellé choisi dans un comboBox
/// </summary>
/// <param name="unLibelleStage">Libellé</param>
/// <param name="codeCompetence">Code compétence (en sortie)</param>
/// <param name="numStage">Numéro stage (en sortie)</param>
7 références
static public void ExtraireIdStage(string unLibelleStage, out string codeCompetence, out int numStage)
{
    string[] strStage;
    string idStage;

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

#endregion
```

```
#region Session stage

5 références
static public void ExtraireIdSession(string unLibelleSession, out string codeCompetence, out int numStage, out int numSession)
{
    string[] strSession;
    string idStage, idSession;

    // Récupération dans un tableau strStage 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
    idStage = strSession[1].ToString();
    // Conversion en int
    numStage = int.Parse(idStage);
    // Récupération du troisième élément numéro Session du tableau strSession
    idSession = strSession[2].ToString();
    // Conversion en int
    numSession = int.Parse(idSession);
}

#endregion
```



```
#region Module
/// <summary>
/// Extraction du numéro module à partir du libellé choisi dans un comboBox
/// </summary>
/// <param name="unLibelleModule"></param>
/// <returns></returns>
3 références
static public int ExtraireNumModule(string unLibelleModule)
{
    int numModule;
    string[] strModule;
    string idModule;

    // Récupération dans un tableau strModule des éléments du libellé séparé par le caractère "."
    strModule = unLibelleModule.Split(new string[] { "." }, StringSplitOptions.RemoveEmptyEntries);
    //Récupération du premier élément du tableau strModule
    idModule = strModule[0].ToString();
    //Conversion de l'élément en valeur de type int
    numModule = int.Parse(idModule);
    // retour du résultat
    return numModule;
}

#endregion
```