ANNEXES

Table des matières

ANNEXES.....I
Classe Bddmanager.....III
Classe Access.....V
Classe PersonnelAccess.....VI
Classe ServiceAccess.....IX
AbsenceAccess.....X
Classe MotifAccess.....XIII
Classe ResponsableAccess.....XIV
Classe Absence.....XV
Classe Motif.....XV
Classe Personnel.....XVI
Classe Responsable.....XVI
Classe Service.....XVII
Classe FrmPersonnelController.....XVIII
Classe FrmAjoutModifPersonnelController.....XIX
Classe FrmAbsenceController.....XX
Classe FrmAuthentificationController.....XXII
Formulaire FrmPersonnel.....XXIII
Formulaire FrmAjoutModifPersonnel.....XXVI
Formulaire FrmAbsence.....XXIX
Formulaire FrmAuthentification.....XXXIV

Classe Bddmanager

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
/// <summary>
/// Package contenant la classe qui gère la connexion et la communication avec la base de
données
/// </summary>
namespace Mediatek86.bddmanager
{
    /// <summary>
    /// Singleton : connexion à la base de données et exécution des requêtes
    /// </summary>
    public class BddManager
    {
        /// <summary>
        /// instance unique de la classe
        /// </summary>
        private static BddManager instance = null;
        /// <summary>
        /// objet de connexion à la BDD à partir d'une chaîne de connexion
        /// </summary>
        private readonly MySqlConnection connection;

        /// <summary>
        /// Constructeur pour créer la connexion à la BDD et l'ouvrir
        /// </summary>
        /// <param name="stringConnect">chaine de connexion</param>
        private BddManager(string stringConnect)
        {
            connection = new MySqlConnection(stringConnect);
            connection.Open();
        }

        /// <summary>
        /// Création d'une seule instance de la classe
        /// </summary>
        /// <param name="stringConnect">chaine de connexion</param>
        /// <returns>instance unique de la classe</returns>
        public static BddManager GetInstance(string stringConnect)
        {
            if (instance == null)
            {
                instance = new BddManager(stringConnect);
            }
            return instance;
        }

        /// <summary>
        /// Exécution d'une requête autre que "select"
        /// </summary>
        /// <param name="stringQuery">requête autre que select</param>
        /// <param name="parameters">dictionnire contenant les parametres</param>
        public void ReqUpdate(string stringQuery, Dictionary<string, object> parameters =
null)
        {
            MySqlCommand command = new MySqlCommand(stringQuery, connection);
            if (!(parameters is null))
            {
```

```
                foreach (KeyValuePair<string, object> parameter in parameters)
                {
                    command.Parameters.Add(new MySqlParameter(parameter.Key,
parameter.Value));
                }
            }
            command.Prepare();
            command.ExecuteNonQuery();
        }

        /// <summary>
        /// Execution d'une requête de type "select"
        /// </summary>
        /// <param name="stringQuery">requête select</param>
        /// <param name="parameters">dictoinnaire contenant les parametres</param>
        /// <returns>liste de tableaux d'objets contenant les valeurs des colonnes</returns>
        public List<Object[]> ReqSelect(string stringQuery, Dictionary<string, object>
parameters = null)
        {
            MySqlCommand command = new MySqlCommand(stringQuery, connection);
            if (!(parameters is null))
            {
                foreach (KeyValuePair<string, object> parameter in parameters)
                {
                    command.Parameters.Add(new MySqlParameter(parameter.Key,
parameter.Value));
                }
            }
            command.Prepare();
            MySqlDataReader reader = command.ExecuteReader();
            int nbCols = reader.FieldCount;
            List<Object[]> records = new List<object[]>();
            while (reader.Read())
            {
                Object[] attributs = new Object[nbCols];
                reader.GetValues(attributs);
                records.Add(attributs);
            }
            reader.Close();
            return records;
        }

    }
}
```

Classe Access

```
using Mediatek86.bddmanager;
using System;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Singleton : classe d'accès à BddManager
    /// </summary>
    public class Access
    {
        /// <summary>
        /// Chaine de connexion à la bdd
        /// </summary>
        private static readonly string connectionString = "server=localhost;user
id=user_mediatek86;password=user_mediatek86_pwd;persistsecurityinfo=True;database=mediatek86
;SslMode=none";
        /// <summary>
        /// Instance unique de la classe
        /// </summary>
        private static Access instance = null;
        /// <summary>
        /// Getter sur l'objet d'accès aux données
        /// </summary>
        public BddManager Manager { get; }
        /// <summary>
        /// Création unique de l'objet de type BddManager
        /// Arrête le programme si l'accès à la BDD a échoué
        /// </summary>
        private Access()
        {
            try
            {
                Manager = BddManager.GetInstance(connectionString);
            }
            catch (Exception)
            {
                Environment.Exit(0);
            }
        }
        /// <summary>
        /// Création d'une seule instance de la classe
        /// </summary>
        /// <returns></returns>
        public static Access GetInstance()
        {
            if (instance == null)
            {
                instance = new Access();
            }
            return instance;
        }
    }
}
```

Classe PersonnelAccess

```
using Mediatek86.modele;
using System;
using System.Collections.Generic;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Classe permettant de gérer les demandes concernant les personnels
    /// </summary>
    public class PersonnelAccess
    {
        /// <summary>
        /// Instance unique de l'accès aux données
        /// </summary>
        private readonly Access access = null;
        /// <summary>
        /// Constructeur pour créer l'accès aux données
        /// </summary>
        public PersonnelAccess()
        {
            access = Access.GetInstance();
        }
        /// <summary>
        /// Récupère et retourne les personnels
        /// </summary>
        /// <returns>Liste des personnels</returns>
        public List<Personnel> GetLesPersonnels()
        {
            List<Personnel> lesPersonnels = new List<Personnel>();
            if (access.Manager != null)
            {
                string req = "select p.idpersonnel as idpersonnel, p.nom as nom, p.prenom as
prenom, p.tel as tel, p.mail as mail, s.idservice as idservice, s.nom as service ";
                req += "from personnel p join service s on (p.idservice = s.idservice) ";
                req += "order by nom, prenom;";
                try
                {
                    List<Object[]> records = access.Manager.ReqSelect(req);
                    if(records != null)
                    {
                        foreach (Object[] record in records)
                        {
                            Service service = new Service((int)record[5],
(string)record[6]);
                            Personnel personnel = new Personnel((int)record[0],
(string)record[1], (string)record[2], (string)record[3], (string)record[4], service);
                            lesPersonnels.Add(personnel);
                        }
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
```

```
                }
            }
            return lesPersonnels;
        }
        /// <summary>
        /// Demande de modification d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void UpdatePersonnel(Personnel personnel)
        {
            if(access.Manager != null)
            {
                string req = "update personnel set nom = @nom, prenom = @prenom, tel = @tel,
mail = @mail, idservice = @idservice ";
                req += "where idpersonnel = @idpersonnel;";
                Dictionary<string, object> parameters = new Dictionary<string, object>();
                parameters.Add("@idpersonnel", personnel.Idpersonnel);
                parameters.Add("@nom", personnel.Nom);
                parameters.Add("@prenom", personnel.Prenom);
                parameters.Add("@tel", personnel.Tel);
                parameters.Add("@mail", personnel.Mail);
                parameters.Add("@idservice", personnel.Service.IdService);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
        }
        /// <summary>
        /// Demande d'ajout d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void AddPersonnel (Personnel personnel)
        {
            if(access.Manager != null)
            {
                string req = " insert into personnel(nom, prenom, tel, mail, idservice) ";
                req += "values (@nom, @prenom, @tel, @mail, @idservice);";
                Dictionary<string, object> parameters = new Dictionary<string, object>();
                parameters.Add("@nom", personnel.Nom);
                parameters.Add("@prenom", personnel.Prenom);
                parameters.Add("@tel", personnel.Tel);
                parameters.Add("@mail", personnel.Mail);
                parameters.Add("@idservice", personnel.Service.IdService);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
        }
        /// <summary>
        /// Demande de suppresion d'un personnel
```

```
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void DelPersonnel(Personnel personnel)
        {
            if (access.Manager != null)
            {
                string req = "delete from personnel where idpersonnel = @idpersonnel;";
                Dictionary<string, object> parameters = new Dictionary<string, object>();
                parameters.Add("@idpersonnel", personnel.Idpersonnel);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
        }
    }
}
```

Classe ServiceAccess

```
using Mediatek86.modele;
using System;
using System.Collections.Generic;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Classe permettant de gérer les demandes concernant les services
    /// </summary>
    public class ServiceAccess
    {
        /// <summary>
        /// Instance unique de l'accès aux données
        /// </summary>
        private readonly Access access = null;
        /// <summary>
        /// Constructeur pour créer l'accès aux données
        /// </summary>
        public ServiceAccess()
        {
            access = Access.GetInstance();
        }
        /// <summary>
        /// Récupère et retourne les services
        /// </summary>
        /// <returns>Liste des services</returns>
        public List<Service> GetLesServices()
        {
            List<Service> lesServices = new List<Service>();
            if (access.Manager != null)
            {
                string req = "select * from service order by nom";
                try
                {
                    List<Object[]> records = access.Manager.ReqSelect(req);
                    if (records != null)
                    {
                        foreach (object[] record in records)
                        {
                            Service service = new Service((int)record[0],
(string)record[1]);
                            lesServices.Add(service);
                        }
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
            return lesServices;
        }
    }
}
```

AbsenceAccess

```
using Mediatek86.modele;
using System;
using System.Collections.Generic;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Classe permettant de gérer les demandes concernant les absences
    /// </summary>
    public class AbsenceAccess
    {
        /// <summary>
        /// Instance unique de l'accès aux données
        /// </summary>
        private readonly Access access = null;
        /// <summary>
        /// Constructeur pour créer l'accès aux données
        /// </summary>
        public AbsenceAccess()
        {
            access = Access.GetInstance();
        }
        /// <summary>
        /// Récupère et retourne les absences concernant le personnel passé en paramètre
        /// </summary>
        /// <param name="personnel">personnel concerné</param>
        /// <returns>Liste des absences</returns>
        public List<Absence> GetLesAbsences(Personnel personnel)
        {
            List<Absence> lesAbsences = new List<Absence>();
            if (access.Manager != null)
            {
                string req = "SELECT p.idpersonnel as idpersonnel, a.datedebut as datedebut,
a.datefin as datefin, a.idmotif as idmotif, m.libelle as motif ";
                req += "FROM personnel p JOIN absence a ON (p.idpersonnel = a.idpersonnel)
JOIN motif m ON (a.idmotif = m.idmotif) ";
                req += "WHERE p.idpersonnel = @idpersonnel ";
                req += "ORDER BY a.datedebut DESC;";
                Dictionary<string, Object> parameters = new Dictionary<string, Object>();
                parameters.Add("@idpersonnel", personnel.Idpersonnel);
                try
                {
                    List<Object[]> records = access.Manager.ReqSelect(req, parameters);
                    if (records != null)
                    {
                        foreach (Object[] record in records)
                        {
                            Motif motif = new Motif((int)record[3], (string)record[4]);
                            Absence absence = new Absence((int)record[0],
(DateTime)record[1], (DateTime)record[2], motif);
                            lesAbsences.Add(absence);
                        }
                    }
                }
                catch (Exception e)
                {
```

```
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
            return lesAbsences;
        }
        /// <summary>
        /// Demande d'ajout d'une absence
        /// </summary>
        /// <param name="absence">Objet absence à ajouter</param>
        public void AddAbsence(Absence absence)
        {
            if (access.Manager != null)
            {
                string req = "insert into absence(idpersonnel, datedebut, datefin, idmotif)
";
                req += "values (@idpersonnel, @datedebut, @datefin, @idmotif);";
                Dictionary<string, object> parameters = new Dictionary<string, object>();
                parameters.Add("@idpersonnel", absence.Idpersonnel);
                parameters.Add("@datedebut", absence.Datedebut);
                parameters.Add("@datefin", absence.Datefin);
                parameters.Add("@idmotif", absence.Motif.Idmotif);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }

            }
        }
        /// <summary>
        /// Demande de suppression d'une absence
        /// </summary>
        /// <param name="absence">Objet absence à supprimer</param>
        public void DelAbsence(Absence absence)
        {
            if (access.Manager != null)
            {
                string req = "delete from absence where idpersonnel = @idpersonnel and
datedebut = @datedebut;";
                Dictionary<string, Object> parameters = new Dictionary<string, object>();
                parameters.Add("@idpersonnel", absence.Idpersonnel);
                parameters.Add("@datedebut", absence.Datedebut);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
        }
        /// <summary>
        /// Demande de suppression de toutes les absences d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
```

```
        public void DelAllAbsence(Personnel personnel)
        {
            if (access.Manager != null)
            {
                string req = "delete from absence where idpersonnel = @idpersonnel;";
                Dictionary<string, Object> parameters = new Dictionary<string, object>();
                parameters.Add("@idpersonnel", personnel.Idpersonnel);
                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
        }
        /// <summary>
        /// Demande de modification d'un absence
        /// </summary>
        /// <param name="absence">Objet absence à supprimer</param>
        /// <param name="personnel">Personnel concerné</param>
        /// <param name="dateDebut">datedebut originelle</param>
        public void UpdateAbsence(Absence absence, Personnel personnel, DateTime dateDebut )
        {
            if (access.Manager != null)
            {
                string req = "update absence set datedebut = @datedebut, datefin = @datefin,
idmotif = @idmotif ";
                req += "where idpersonnel = @idpersonnel and datedebut = @datedebutref";
                Dictionary<string, Object> parameters = new Dictionary<string, object>();
                parameters.Add("@datedebut", absence.Datedebut);
                parameters.Add("@datefin", absence.Datefin);
                parameters.Add("@idmotif", absence.Motif.Idmotif);
                parameters.Add("@idpersonnel", personnel.Idpersonnel);
                parameters.Add("@datedebutref", dateDebut);

                try
                {
                    access.Manager.ReqUpdate(req, parameters);
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }

            }
        }

    }
}
```

Classe MotifAccess

```
using Mediatek86.modele;
using System;
using System.Collections.Generic;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Classe permettant de gérer les demandes concernant les motifs
    /// </summary>
    public class MotifAccess
    {
        /// <summary>
        /// Instance unique de l'accès aux données
        /// </summary>
        private readonly Access access = null;
        /// <summary>
        /// Constructeur pour créer l'accès aux données
        /// </summary>
        public MotifAccess()
        {
            access = Access.GetInstance();
        }
        /// <summary>
        /// Récupère et retourne les motifs
        /// </summary>
        /// <returns>Liste des motifs</returns>
        public List<Motif> GetLesMotifs()
        {
            List<Motif> lesMotifs = new List<Motif>();
            if (access.Manager != null)
            {
                string req = "select * from motif order by libelle";
                try
                {
                    List<Object[]> records = access.Manager.ReqSelect(req);
                    if (records != null)
                    {
                        foreach (object[] record in records)
                        {
                            Motif motif = new Motif((int)record[0], (string)record[1]);
                            lesMotifs.Add(motif);
                        }
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
            return lesMotifs;
        }
    }
}
```

Classe ResponsableAccess

```
using System;
using System.Collections.Generic;
using Mediatek86.modele;
/// <summary>
/// Package Data Access Layer : exploite bbdmanager en lui envoyer les requêtes SQL
/// </summary>
/// </summary>
namespace Mediatek86.dal
{
    /// <summary>
    /// Classe permettant de gérer les demandes pour les reponsables
    /// </summary>
    public class ResponsableAccess
    {
        /// <summary>
        /// Instance unique de l'accès aux données
        /// </summary>
        private readonly Access access = null;

        /// <summary>
        /// Constructeur pour créer l'accès aux données
        /// </summary>
        public ResponsableAccess()
        {
            access = Access.GetInstance();
        }
        /// <summary>
        /// Controle si l'utilisateur à le droit de se connecter (responsable)
        /// </summary>
        /// <param name="responsable"></param>
        /// <returns>Vrai si l'utilisateur est le responsable</returns>
        public Boolean ControleAuthentification(Responsable responsable)
        {
            if (access.Manager != null)
            {
                string req = "select * from responsable where login = @login and pwd =
SHA2(@pwd,256)";
                Dictionary<string, object> parameters = new Dictionary<string, object>();
                parameters.Add("@login", responsable.Identifiant);
                parameters.Add(@"pwd", responsable.Mdp);
                try
                {
                    List<Object[]> records = access.Manager.ReqSelect(req, parameters);
                    if (records != null)
                    {
                        return (records.Count > 0);
                    }
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    Environment.Exit(0);
                }
            }
            return false;
        }

    }
}
```

Classe Absence

```
using System;
/// <summary>
/// Package contenant les classes métiers
/// </summary>
namespace Mediatek86.modele
{
    /// <summary>
    /// Classe métier liée à la table Absence
    /// </summary>
    public class Absence
    {
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="idpersonnel"></param>
        /// <param name="datedebut"></param>
        /// <param name="datefin"></param>
        /// <param name="motif"></param>
        public Absence (int idpersonnel, DateTime datedebut, DateTime datefin, Motif motif)
        {
            this.Idpersonnel = idpersonnel;
            this.Datedebut = datedebut;
            this.Datefin = datefin;
            this.Motif = motif;
        }
        public int Idpersonnel { get; }
        public DateTime Datedebut { get; set; }
        public DateTime Datefin { get; set; }
        public Motif Motif { get; set; }
    }
```

Classe Motif

```
    /// <summary>
    /// Classe métier liée à la table Motif
    /// </summary>
    public class Motif
    {
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="idmotif"></param>
        /// <param name="libelle"></param>
        public Motif (int idmotif, string libelle)
        {
            this.Idmotif = idmotif;
            this.Libelle = libelle;
        }
        public int Idmotif { get; }
        public string Libelle { get; }
        /// <summary>
        /// Définit l'information à afficher (Libelle)
        /// </summary>
        /// <returns>nom du service</returns>
        public override string ToString()
```

```
        {
            return this.Libelle;
        }
    }
```

Classe Personnel

```
    /// <summary>
    /// Classe métier liée à la table Personnel
    /// </summary>
    public class Personnel
    {
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="idpersonnel"></param>
        /// <param name="nom"></param>
        /// <param name="prenom"></param>
        /// <param name="tel"></param>
        /// <param name="mail"></param>
        /// <param name="idservice"></param>
        /// <param name="service"></param>
        public Personnel(int idpersonnel, string nom, string prenom, string tel, string
mail, Service service)
        {
            this.Idpersonnel = idpersonnel;
            this.Nom = nom;
            this.Prenom = prenom;
            this.Tel = tel;
            this.Mail = mail;
            this.Service = service;
        }
        public int Idpersonnel { get; }
        public string Nom { get; set; }
        public string Prenom { get; set; }
        public string Tel { get; set; }
        public string Mail { get; set; }
        public Service Service { get; set; }
    }
```

Classe Responsable

```
    /// <summary>
    /// Classe métier liée à la table Responsable
    /// </summary>
    public class Responsable
    {
        public string Identifiant { get; }
        public string Mdp { get; }
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="identifiant"></param>
        /// <param name="mdp"></param>
```

```
        public Responsable(string identifiant, string mdp)
        {
            this.Identifiant = identifiant;
            this.Mdp = mdp;
        }
    }
```

Classe Service

```
    /// <summary>
    /// Classe métier liée à la table Service
    /// </summary>
    public class Service
    {
        public int IdService { get; }
        public string Nom { get; }
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="idprofil"></param>
        /// <param name="nom"></param>
        public Service(int idprofil, string nom)
        {
            this.IdService = idprofil;
            this.Nom = nom;
        }
        /// <summary>
        /// Définit l'information à afficher (nom)
        /// </summary>
        /// <returns>Nom du service</returns>
        public override string ToString()
        {
            return this.Nom;
        }
    }

}
```

Classe FrmPersonnelController

```
using Mediatek86.modele;
using Mediatek86.dal;
using System.Collections.Generic;
/// <summary>
/// Package qui contient les classes qui servent d'intermédiaire entre la Vue et le Modèle
/// </summary>
namespace Mediatek86.contoleur
{
        /// <summary>
        /// Controlleur de FrmPersonnels
        /// </summary>
        public class FrmPersonnelsController
    {
        /// <summary>
        /// Objet d'accès aux opérations possibles sur Personnel
        /// </summary>
        private readonly PersonnelAccess personnelAccess;
        /// <summary>
        /// Objet d'accès aux opérations possibles sur Absences
        /// </summary>
        private readonly AbsenceAccess absenceAccess;
        /// <summary>
        /// Récupère l'accès aux données
        /// </summary>
        public FrmPersonnelsController()
        {
            personnelAccess = new PersonnelAccess();
            absenceAccess = new AbsenceAccess();
        }
        /// <summary>
        /// Récupère et retourne les infos des personnels
        /// </summary>
        /// <returns></returns>
        public List<Personnel> GetLesPersonnels()
        {
            return personnelAccess.GetLesPersonnels();
        }
        /// <summary>
        /// Demande de suppression de toutes les absences d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void DelAllAbsence(Personnel personnel)
        {
            absenceAccess.DelAllAbsence(personnel);
        }
        /// <summary>
        /// Demande de suppression d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void DelPersonnel(Personnel personnel)
        {
            personnelAccess.DelPersonnel(personnel);
        }
    }
}
```

Classe FrmAjoutModifPersonnelController

```
using Mediatek86.dal;
using Mediatek86.modele;
using System.Collections.Generic;
/// <summary>
/// Package qui contient les classes qui servent d'intermédiaire entre la Vue et le Modèle
/// </summary>
namespace Mediatek86.contoleur
{
    /// <summary>
    /// Controleur de FrmAjoutModifPersonnel
    /// </summary>
    public class FrmAjoutModifPersonnelController
    {
        /// <summary>
        /// Objet d'accès aux opérations possibles sur Service
        /// </summary>
        private readonly ServiceAccess serviceAccess;
        private readonly PersonnelAccess personnelAccess;
        /// <summary>
        /// Récupère l'accès aux données
        /// </summary>
        public FrmAjoutModifPersonnelController()
        {
            serviceAccess = new ServiceAccess();
            personnelAccess = new PersonnelAccess();
        }
        /// <summary>
        /// Récupère et retourne les infos des services
        /// </summary>
        /// <returns>Liste des services</returns>
        public List<Service> GetLesServices()
        {
            return serviceAccess.GetLesServices();
        }
        /// <summary>
        /// Demande de modification d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void UpdatePersonnel(Personnel personnel)
        {
            personnelAccess.UpdatePersonnel(personnel);
        }
        /// <summary>
        /// Demande d'ajout d'un personnel
        /// </summary>
        /// <param name="personnel">Personnel concerné</param>
        public void AddPersonnel(Personnel personnel)
        {
            personnelAccess.AddPersonnel(personnel);
        }


    }
}
```

Classe FrmAbsenceController

```
using Mediatek86.dal;
using Mediatek86.modele;
using System;
using System.Collections.Generic;
/// <summary>
/// Package qui contient les classes qui servent d'intermédiaire entre la Vue et le Modèle
/// </summary>
namespace Mediatek86.contoleur
{
    /// <summary>
    /// Controleur de FrmAbsence
    /// </summary>
    public class FrmAbsenceController
    {
        /// <summary>
        /// Objet d'accès aux opérations possibles sur Absence
        /// </summary>
        private readonly AbsenceAccess absenceAccess;
        /// <summary>
        /// Objet d'accès aux opération possibles sur Motif
        /// </summary>
        private readonly MotifAccess motifAccess;
        /// <summary>
        /// Récupère les accès aux données
        /// </summary>
        public FrmAbsenceController()
        {
            absenceAccess = new AbsenceAccess();
            motifAccess = new MotifAccess();
        }
        /// <summary>
        /// Récupère et retourne les infos des développeurs
        /// </summary>
        /// <returns>liste des absences</returns>
        public List<Absence> GetLesAbsences(Personnel personnel)
        {
            return absenceAccess.GetLesAbsences(personnel);
        }
        /// <summary>
        /// Récupère et retourne les infos des motifs
        /// </summary>
        /// <returns></returns>
        public List<Motif> GetLesMotifs()
        {
            return motifAccess.GetLesMotifs();
        }
        /// <summary>
        /// Demande d'ajout d'une absence
        /// </summary>
        /// <param name="absence">objet absence à ajouter</param>
        public void AddAbsence(Absence absence)
        {
            absenceAccess.AddAbsence(absence);
        }
        /// <summary>
        /// Demande de suppresion d'une absence
        /// </summary>
        /// <param name="absence">objet absence à supprimer</param>
```

```
        public void DelAbsence (Absence absence)
        {
            absenceAccess.DelAbsence(absence);
        }
        /// <summary>
        /// Demande de modification d'une absence
        /// </summary>
        /// <param name="absence">objet absence à supprimer</param>
        /// <param name="personnel">personnel concerné</param>
        /// <param name="dateDebut">datedebut originelle</param>
        public void UpdateAbsence(Absence absence, Personnel personnel, DateTime dateDebut)
        {
            absenceAccess.UpdateAbsence(absence, personnel, dateDebut);
        }

    }
}
```

Classe FrmAuthentificationController

```
using Mediatek86.dal;
using Mediatek86.modele;
using System;
/// <summary>
/// Package qui contient les classes qui servent d'intermédiaire entre la Vue et le Modèle
/// </summary>
namespace Mediatek86.contoleur
{
    /// <summary>
    /// Controleur de FrmAUthentification
    /// </summary>
    public class FrmAuthentificationController
    {
        /// <summary>
        /// Objet d'accès aux opérations possibles sur Responsable
        /// </summary>
        private readonly ResponsableAccess responsableAccess;

        /// <summary>
        /// Récupère l'accès aux données
        /// </summary>
        public FrmAuthentificationController()
        {
            responsableAccess = new ResponsableAccess();
        }
        /// <summary>
        /// Vérifie l'authentification
        /// </summary>
        /// <param name="responsable">objet contenant les informations de connexion</param>
        /// <returns>vrai si les informations de connexion sont correctes</returns>
        public Boolean ControleAuthentification(Responsable responsable)
        {
            return responsableAccess.ControleAuthentification(responsable);
        }
    }
}
```

Formulaire FrmPersonnel

```
using Mediatek86.contoleur;
using Mediatek86.modele;
using System;
using System.Collections.Generic;
using System.Windows.Forms;
/// <summary>
/// Package contenant les interfaces
/// </summary>
namespace Mediatek86.vue
{
    /// <summary>
    /// Fentre d'affichage des personnels et de leurs infos
    /// </summary>
    public partial class FrmPersonnels : Form
    {
        /// <summary>
        /// Objet pour gérer la liste des personnels
        /// </summary>
        private BindingSource bdgPersonnels = new BindingSource();
        /// <summary>
        /// Controleur de la fenêtre
        /// </summary>
        private FrmPersonnelsController controller;
        /// <summary>
        /// Demande de modification de personnel
        /// </summary>
        public Button BtnModifierPersonnel
        {
            get { return btnModifierPersonnel; }
        }
        /// <summary>
        /// Construction des composants graphiques et appel des autres initialisations
        /// </summary>
        /// <param name="frmauthentification">Instance de la fenêtre
d'authentification</param>
        public FrmPersonnels(FrmAuthentification frmauthentification)
        {
            InitializeComponent();
            if (frmauthentification != null)
            {
                frmauthentification.Visible = false;
            }
            Init();
        }
        /// <summary>
        /// Initialisations : création du controleur et remplissage de la liste
        /// </summary>
        private void Init()
        {
            controller = new FrmPersonnelsController();
            this.StartPosition = FormStartPosition.CenterScreen;
            RemplirListePersonnels();
        }
        /// <summary>
        /// Affiche les personnels
        /// </summary>
        public void RemplirListePersonnels()
        {
```

```
            List<Personnel> lesPersonnels = controller.GetLesPersonnels();
            bdgPersonnels.DataSource = lesPersonnels;
            dgvPersonnels.DataSource = bdgPersonnels;
            dgvPersonnels.Columns["idpersonnel"].Visible = false;
            dgvPersonnels.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
        }
        /// <summary>
        /// Demande de création d'un personnel
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnAjouterPersonnel_Click(object sender, EventArgs e)
        {
            FrmAjoutModifPersonnel frm = new FrmAjoutModifPersonnel(null, sender, this);
            frm.ShowDialog();
        }
        /// <summary>
        /// Demande de modification d'un personnel
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnModifierPersonnel_Click(object sender, EventArgs e)
        {
            if (dgvPersonnels.SelectedRows.Count > 0)
            {
                Personnel personnel = (Personnel)bdgPersonnels.List[bdgPersonnels.Position];
                FrmAjoutModifPersonnel frm = new FrmAjoutModifPersonnel(personnel, sender,
this);
                frm.ShowDialog();
            }

        }
        /// <summary>
        /// Demande de suppresion d'un personnel (et préalablement, de toutes ces absences)
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnSupprimerPersonnel_Click(object sender, EventArgs e)
        {
            if (dgvPersonnels.SelectedRows.Count > 0)
            {
                Personnel personnel = (Personnel)bdgPersonnels.List[bdgPersonnels.Position];
                if (MessageBox.Show("Voulez-vous vraiment supprimer " + personnel.Nom + " "
+ personnel.Prenom + " ?", "Confirmation de suppression", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
                {
                    controller.DelAllAbsence(personnel);
                    controller.DelPersonnel(personnel);
                    RemplirListePersonnels();
                }
            }
            else
            {
                MessageBox.Show("Une ligne doit être sélectionnée.", "Information");
            }
        }
        /// <summary>
        /// Demande d'affichage des absences
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnAbsence_Click(object sender, EventArgs e)
```

```
        {
            if (dgvPersonnels.SelectedRows.Count > 0)
            {
                Personnel personnel = (Personnel)bdgPersonnels.List[bdgPersonnels.Position];
                FrmAbsence frm = new FrmAbsence(personnel);
                frm.ShowDialog();
            }
        }
    }
}
```

Formulaire FrmAjoutModifPersonnel

```
using Mediatek86.contoleur;
using Mediatek86.modele;
using System;
using System.Collections.Generic;
using System.Windows.Forms;
/// <summary>
/// Package contenant les interfaces
/// </summary>
namespace Mediatek86.vue
{
    /// <summary>
    /// Fenêtre d'ajout ou de modification de personnel
    /// </summary>
    public partial class FrmAjoutModifPersonnel : Form
    {
        /// <summary>
        /// Booléen pour savoir si une modification est demandée
        /// </summary>
        private Boolean enCoursDeModifDeveloppeur = false;
        /// <summary>
        /// Controleur de la fenêtre
        /// </summary>
        private FrmAjoutModifPersonnelController controller;
        private FrmPersonnels frmPersonnels;
        private Personnel personnel;
        /// <summary>
        /// Objet pour gérer la liste des services
        /// </summary>
        private BindingSource bdgServices = new BindingSource();
        /// <summary>
        /// Construction des composants graphiques et appel des autres initialisations
        /// </summary>
        public FrmAjoutModifPersonnel(Personnel personnel, Object sender, FrmPersonnels frm)
        {
            InitializeComponent();
            Init(personnel, sender, frm);
        }
        /// <summary>
        /// Initialisations : création du controller
        /// </summary>
        /// <param name="personnel">personnel de la ligne selectionnée</param>
        /// <param name="sender">bouton ayant déclenché l'ouverture de la frame (ajouter ou
modifier)</param>
        /// <param name="frm"></param>
        private void Init(Personnel personnel, Object sender, FrmPersonnels frm)
        {

            controller = new FrmAjoutModifPersonnelController();
            this.StartPosition = FormStartPosition.CenterScreen;
            this.frmPersonnels = frm;
            this.personnel = personnel;
            RemplirListServices();
            if ((Button)sender == frm.BtnModifierPersonnel)
            {
                grpActionPersonnel.Text = "modifier un personnel";
                txtNom.Text = personnel.Nom;
                txtPrenom.Text = personnel.Prenom;
                txtTelephone.Text = personnel.Tel;
```

```
                txtMail.Text = personnel.Mail;
                cbbServices.SelectedIndex =
cbbServices.FindStringExact(personnel.Service.Nom);
                enCoursDeModifDeveloppeur = true;
            }
            else
            {
                grpActionPersonnel.Text = "ajouter un personnel";
                txtNom.Text = "";
                txtPrenom.Text = "";
                txtTelephone.Text = "";
                txtMail.Text = "";
            }

        }
        /// <summary>
        /// Affiche les services
        /// </summary>
        private void RemplirListServices()
        {
            List<Service> lesServices = controller.GetLesServices();
            bdgServices.DataSource = lesServices;
            cbbServices.DataSource = bdgServices;

        }
        /// <summary>
        /// Demande d'enregistrement de l'ajout ou de la modification d'un développeur
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnEnregistrerPersonnel_Click(object sender, EventArgs e)
        {
            if (!txtNom.Text.Equals("") && !txtPrenom.Text.Equals("") && !
txtTelephone.Text.Equals("") && !txtMail.Text.Equals("") && cbbServices.SelectedIndex != -1)
            {
                Service service = (Service)bdgServices.List[bdgServices.Position];
                if (enCoursDeModifDeveloppeur)
                {
                    if (MessageBox.Show("Voulez-vous vraiment modifier " + personnel.Nom + "
" + personnel.Prenom + " ?", "Confirmation de suppression", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
                    {
                        personnel.Nom = txtNom.Text;
                        personnel.Prenom = txtPrenom.Text;
                        personnel.Tel = txtTelephone.Text;
                        personnel.Mail = txtMail.Text;
                        personnel.Service = service;
                        controller.UpdatePersonnel(personnel);
                    }
                }
                else
                {
                    Personnel personnel = new Personnel(0, txtNom.Text, txtPrenom.Text,
txtTelephone.Text, txtMail.Text, service);
                    controller.AddPersonnel(personnel);
                }
                frmPersonnels.RemplirListePersonnels();
                enCoursDeModifDeveloppeur=false;
                this.Close();
            }
            else
            {
```

```
                MessageBox.Show("Tous les champs doivent être remplis.", "Information");
            }

        }
        /// <summary>
        /// annuler la demande d'ajout ou de modification d'un personnel
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnAnnulerPersonnel_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Voulez-vous vraiment annuler ?", "Confirmation",
MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                this.Close();
            }
        }
    }
}
```

Formulaire FrmAbsence

```
using Mediatek86.contoleur;
using Mediatek86.modele;
using System;
using System.Collections.Generic;
using System.Windows.Forms;
/// <summary>
/// Package contenant les interfaces
/// </summary>
namespace Mediatek86.vue
{
    /// <summary>
    /// Fenêtre de gestion des absences du personnel selectionné
    /// </summary>
    public partial class FrmAbsence : Form
    {
        /// <summary>
        /// Date de début originelle de l'absence à modifier
        /// </summary>
        private DateTime dateDebut;
        /// <summary>
        /// Personnel concerné par les absences
        /// </summary>
        private Personnel personnel;
        /// <summary>
        /// Booléen pour savoir si une modification est demandée
        /// </summary>
        private Boolean enCoursDeMotdifAbsence = false;
        /// <summary>
        /// Booléen poue savoir si un ajout est demandé
        /// </summary>
        private Boolean enCoursAjoutAbsence = false;
        /// <summary>
        /// Objet pour gérer la liste des absences
        /// </summary>
        private BindingSource bdgAbsences = new BindingSource();
        /// <summary>
        /// Objet pour gérer la liste des motifs
        /// </summary>
        private BindingSource bdgMotifs = new BindingSource();
        /// <summary>
        /// Controleur de la fenêtre
        /// </summary>
        private FrmAbsenceController controller;
        /// <summary>
        /// Construction des composants graphiques et appel des autres initialisations
        /// </summary>
        public FrmAbsence(Personnel personnel)
        {
            InitializeComponent();
            Init(personnel);
        }
        /// <summary>
        /// Initialisations : création du controleur et remplissage de la liste des absences
        /// </summary>
        private void Init(Personnel personnel)
        {
            controller = new FrmAbsenceController();
            this.StartPosition = FormStartPosition.CenterScreen;
```

```
        this.personnel = personnel;
        grpActionsAbsence.Enabled = false;
        grpAbsences.Text = personnel.Nom + personnel.Prenom;
        RemplirListeAbsences(personnel);
        RemplirListeMotifs();
        InitialisationActionAbsence();
    }
    /// <summary>
    /// Affiche les absences
    /// </summary>
    /// <param name="personnel"></param>
    private void RemplirListeAbsences(Personnel personnel)
    {
        List<Absence> lesAbsences = controller.GetLesAbsences(personnel);
        bdgAbsences.DataSource = lesAbsences;
        dgvAbsences.DataSource = bdgAbsences;
        dgvAbsences.Columns["idpersonnel"].Visible = false;
        dgvAbsences.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

    }
    /// <summary>
    /// Affiche les motifs
    /// </summary>
    private void RemplirListeMotifs()
    {
        List<Motif> lesMotifs = controller.GetLesMotifs();
        bdgMotifs.DataSource = lesMotifs;
        cboMotifs.DataSource = bdgMotifs;

    }
    /// <summary>
    /// Initialise et rend inaccessible la zone d'ajout/modification
    /// </summary>
    private void InitialisationActionAbsence()
    {
        grpAbsences.Enabled = true;
        grpActionsAbsence.Enabled = false;
        grpActionsAbsence.Text = "";
        dtpDebut.Value = DateTime.Today;
        dtpFin.Value = DateTime.Today;
        cboMotifs.SelectedItem = cboMotifs.Items[0];

    }
    /// <summary>
    /// Modification d'affichage si on est en cours de modif
    /// </summary>
    /// <param name="modif"></param>
    private void EnCoursModifAbsence(Boolean modif)
    {
        enCoursDeMotdifAbsence = modif;
        grpActionsAbsence.Text = "modifer une absence";
    }
    /// <summary>
    /// Demande d'enregistrement de l'ajout d'un absence
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void btnEnregistrerAbsence_Click(object sender, EventArgs e)
    {
        if (cboMotifs.SelectedIndex != -1)
        {
            if (dtpFin.Value >= dtpDebut.Value)
```

```
                {
                    if (!ControleAbsencesSimultanees())
                    {
                        Motif motif = (Motif)bdgMotifs.List[bdgMotifs.Position];
                        if (enCoursDeMotdifAbsence)
                        {
                            if (MessageBox.Show("Voulez-vous vraiment modifier : " +
dgvAbsences.SelectedRows[0].Cells[3].Value + " du " + ((DateTime)
(dgvAbsences.SelectedRows[0].Cells[1].Value)).ToString("dd/MM/yyyy") + " au " + ((DateTime)
(dgvAbsences.SelectedRows[0].Cells[2].Value)).ToString("dd/MM/yyyy") + " ?", "Confirmation
de suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
                            {
                                Absence absence =
(Absence)bdgAbsences.List[bdgAbsences.Position];
                                absence.Datedebut = dtpDebut.Value;
                                absence.Datefin = dtpFin.Value;
                                absence.Motif = motif;
                                controller.UpdateAbsence(absence, personnel, dateDebut);
                                EnCoursModifAbsence(false);
                            }
                        }
                        else if (enCoursAjoutAbsence)
                        {
                            Absence absence = new Absence(personnel.Idpersonnel,
dtpDebut.Value, dtpFin.Value, motif);
                            controller.AddAbsence(absence);
                            EnCoursAjoutAbsence(false);
                        }
                        RemplirListeAbsences(personnel);
                        InitialisationActionAbsence();
                    }
                    else
                    {
                        MessageBox.Show("Une absence est déjà enregistrée sur cette
période", "Information");
                    }
                }
                else
                {
                    MessageBox.Show("La date de fin doit être ultérieure à la date de
début", "Information");
                }
            }
            else
            {
                MessageBox.Show("Un motif doit être sélectionné.", "Information");
            }
        }
        /// <summary>
        /// Annuler la demande d'ajout d'une absence
        /// Réinitialise les zones de saisie de l'absence
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnAnnulerAbsence_Click(object sender, EventArgs e)
        {
                if (MessageBox.Show("Voulez-vous vraiment annuler ?", "Confirmation",
MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    InitialisationActionAbsence();
                }
        }
```

```
        /// <summary>
        /// Demande de suppression d'une absence
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnSupprimerAbsence_Click(object sender, EventArgs e)
        {
            if (dgvAbsences.SelectedRows.Count > 0)
            {
                Absence absence = (Absence)bdgAbsences.List[bdgAbsences.Position];
                if (MessageBox.Show("Voulez-vous vraiment supprimer : " +
dgvAbsences.SelectedRows[0].Cells[3].Value + " du " + ((DateTime)
(dgvAbsences.SelectedRows[0].Cells[1].Value)).ToString("dd/MM/yyyy") + " au " + ((DateTime)
(dgvAbsences.SelectedRows[0].Cells[2].Value)).ToString("dd/MM/yyyy") + " ?", "Confirmation
de suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    controller.DelAbsence(absence);
                    RemplirListeAbsences(personnel);
                }
            }
            else
            {
                MessageBox.Show("Une ligne doit être sélectionnée.", "Information");
            }
        }
        /// <summary>
        /// Demande de modification d'un développeur
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void bntModifierAbsence_Click(object sender, EventArgs e)
        {
            if (dgvAbsences.SelectedRows.Count > 0)
            {
                grpActionsAbsence.Enabled = true;
                EnCoursModifAbsence(true);
                Absence absence = (Absence)bdgAbsences.List[bdgAbsences.Position];
                dtpDebut.Value = (DateTime)dgvAbsences.SelectedRows[0].Cells[1].Value;
                dateDebut = (DateTime)dgvAbsences.SelectedRows[0].Cells[1].Value;
                dtpFin.Value = (DateTime)dgvAbsences.SelectedRows[0].Cells[2].Value;
                cboMotifs.SelectedIndex = cboMotifs.FindStringExact(absence.Motif.Libelle);
            }
            else
            {
                MessageBox.Show("Une ligne doit être selectionnée.", "Information");
            }
        }
        /// <summary>
        ///  Demande d'ajout d'une absence
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnAjouter_Click(object sender, EventArgs e)
        {
            grpActionsAbsence.Enabled = true;
            EnCoursAjoutAbsence(true);
        }
        /// <summary>
        /// Modification d'affichage si on est en cours d'ajout
        /// </summary>
        /// <param name="modif"></param>
        private void EnCoursAjoutAbsence(Boolean modif)
```

```
        {
            enCoursAjoutAbsence = modif;
            grpActionsAbsence.Text = "ajouter une absence";
        }
        /// <summary>
        /// Controle d'absences simultanées
        /// </summary>
        /// <returns></returns>
        private Boolean ControleAbsencesSimultanees()
        {
            Boolean test = false;
            if (enCoursDeMotdifAbsence)
            {
                foreach (DataGridViewRow row in dgvAbsences.Rows)
                {
                    if (!row.Selected && (dtpDebut.Value <= (DateTime)row.Cells[2].Value &&
dtpFin.Value >= (DateTime)row.Cells[1].Value))
                    {
                        test = true;
                    }
                }
                return test;
            }
            else if(enCoursAjoutAbsence)
            {
                foreach (DataGridViewRow row in dgvAbsences.Rows)
                {
                    if ((dtpDebut.Value <= (DateTime)row.Cells[2].Value && dtpFin.Value >=
(DateTime)row.Cells[1].Value) || dtpDebut.Value == (DateTime)row.Cells[1].Value)
                    {
                        test = true;
                    }
                }
            }
            return test;
        }
        /// <summary>
        /// Fermeture de la fenêtre de gestion des absences
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnFermerAbsences_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulaire FrmAuthentification

```
using Mediatek86.contoleur;
using Mediatek86.modele;
using System;
using System.Windows.Forms;
/// <summary>
/// Package contenant les interfaces
/// </summary>
namespace Mediatek86.vue
{
    /// <summary>
    /// Fenêtre d'authentification (seul le responsable peut accéder à l'application)
    /// </summary>
    public partial class FrmAuthentification : Form
    {
        /// <summary>
        /// Contrôleur de la fenêtre
        /// </summary>
        private FrmAuthentificationController controller;
        /// <summary>
        /// Construction des composants graphiques et appel des autres initialisations
        /// </summary>
        public FrmAuthentification()
        {
            InitializeComponent();
            Init();
        }

        /// <summary>
        /// Initialisation : création du controleur
        /// </summary>
        private void Init()
        {
            controller = new FrmAuthentificationController();
            this.StartPosition = FormStartPosition.CenterScreen;
        }

        /// <summary>
        /// Demande au controleur de controler l'authentification
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnSeConnecter_Click(object sender, EventArgs e)
        {
            String identifiant = txtIdentifiant.Text;
            String mdp = txtMdp.Text;
            if (String.IsNullOrEmpty(identifiant) || String.IsNullOrEmpty(mdp))
            {
                MessageBox.Show("Tous les champs doivent être remplis.", "Information");
            }
            else
            {
                Responsable responsable = new Responsable(identifiant, mdp);
                if (controller.ControleAuthentification(responsable))
                {
                    FrmPersonnels frm = new FrmPersonnels(this);
                    frm.ShowDialog();

                }
```

```
                else
                {
                    MessageBox.Show("Authentification incorrecte ou vous n'êtes pas admin",
"Alerte");
                }
            }
        }
    }
}

                                                }
```