Compte rendu atelier n°2

-Somaire :

Etape #1 :

- Mise en place de la base de données

Etape #2 :

-Creation du projet et de ces dossiers
-Creation du kanban et du repos sur github
-Test de la connection a la base de données
-Prototypage des vus
-Mise en places des vus

Etape #3 :

-Code de la parti modèle, classes metiers
-Implementation du dal
-Génération de la premiere documentation technique au format chm

Etape #4 :

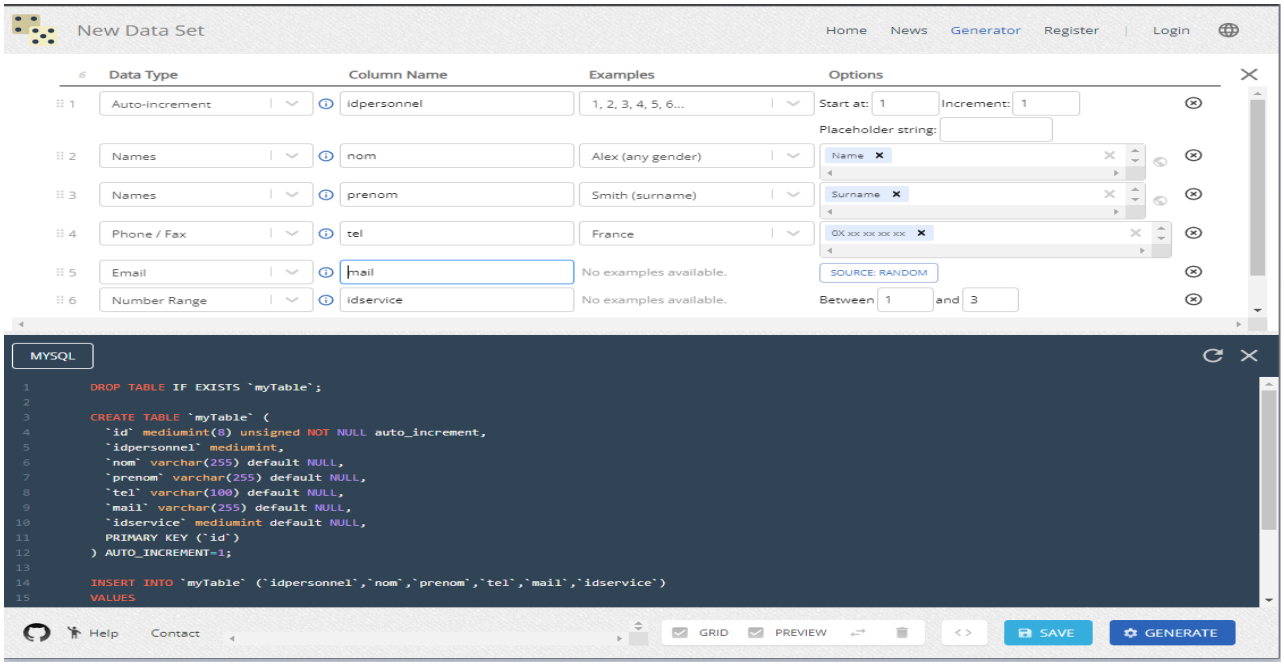
-Code de la connection (de la form jusqu'au dal):
creation et code de dal.connectionAcces, code des fonctionnalité de la vue et de son controlleur
-Code de la parti personnel (de la form jusqu'au dal) :
creation d'un deuxiéme controlleur, code de la parti personnel de la vue, du controlleur, creation et code de dal.personnelAcces
-Code de la parti absence (de la form jusqu'au dal) :
code de la parti absence de la vue, du controlleur, creation et code de dal.absenceAcces
-Code de l'initialisation des parti graphique pour les services et les motifs
code du dal initAcces et du code necessaire a son fonctionnement dans la vue et le controlleur
-Finalisation :
modifications, testes et verifications des diverses fonctionnalités

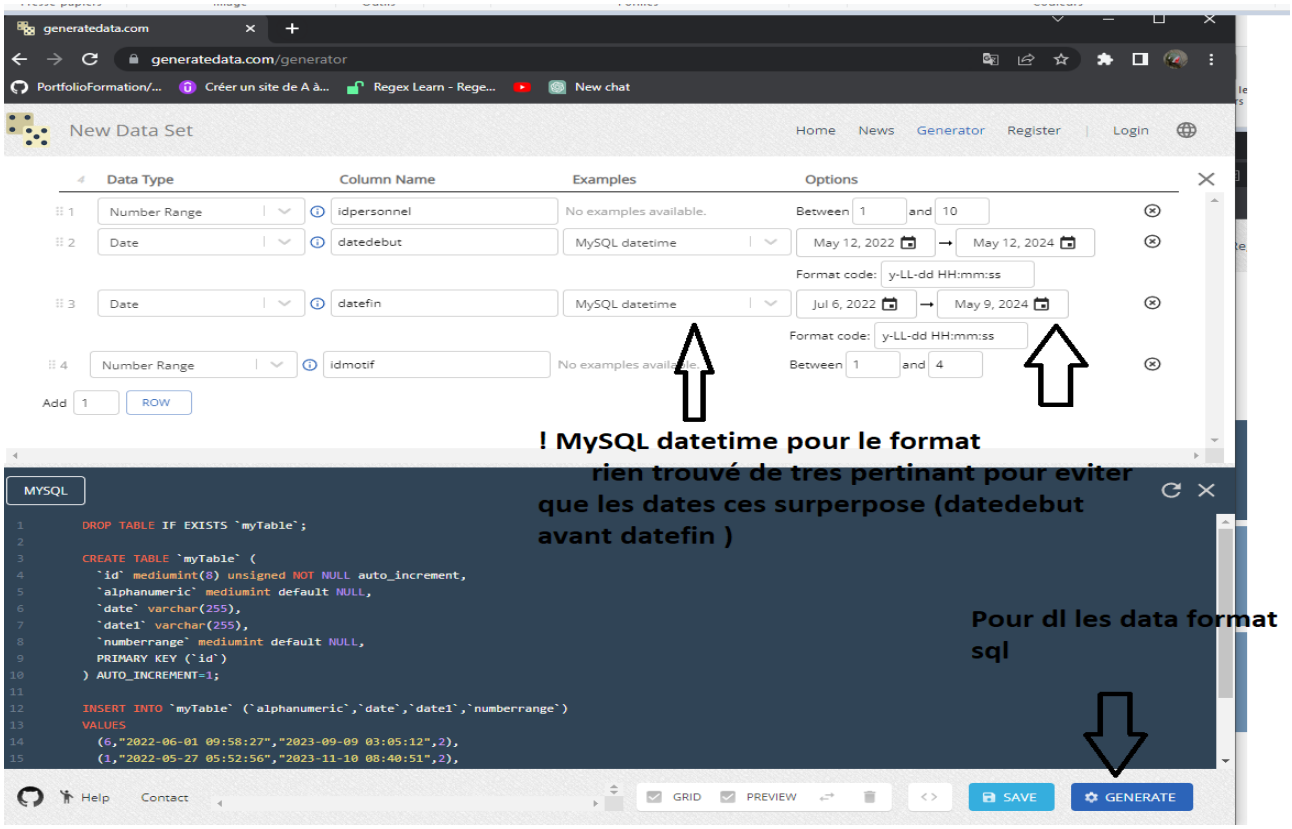
Etape #1 :

-Mise en place de la base de données :

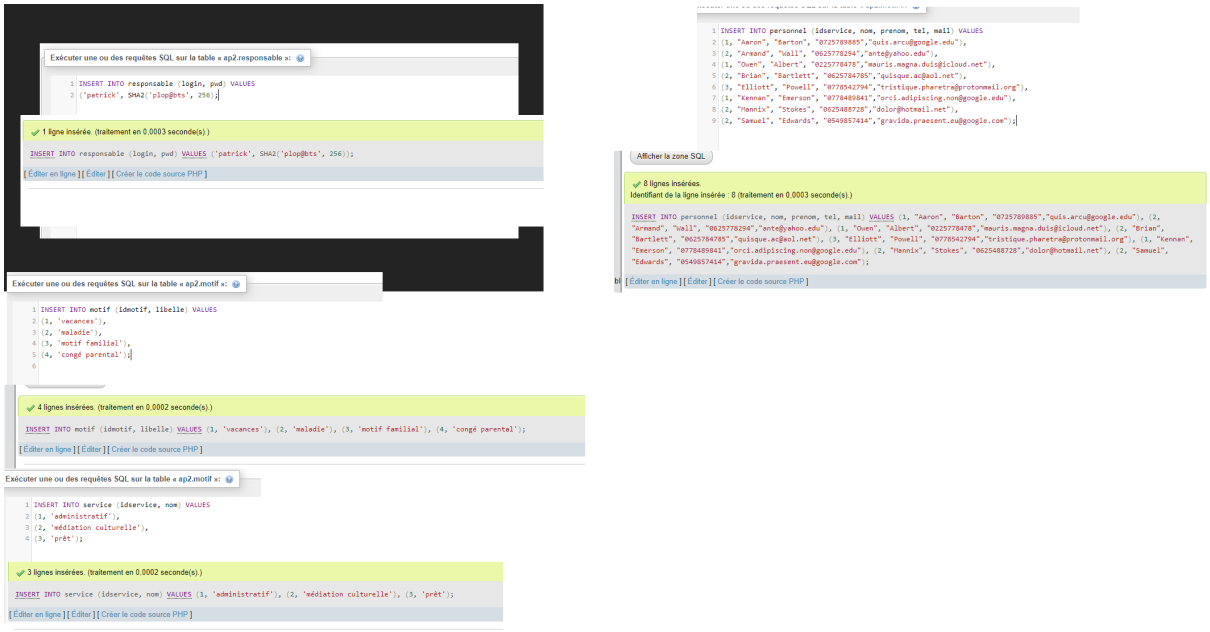
Pour mettre en place la base de données je commence par recupérer le schéma MCD puis je genere le modèle logique. Je créer une base sous Myphpadmin, j'importe le script créer avec windesign. J'ai le squelette de ma base de données.

Je dois génerer des données pour la remplir, pour cela j'utilise generatedata.com.

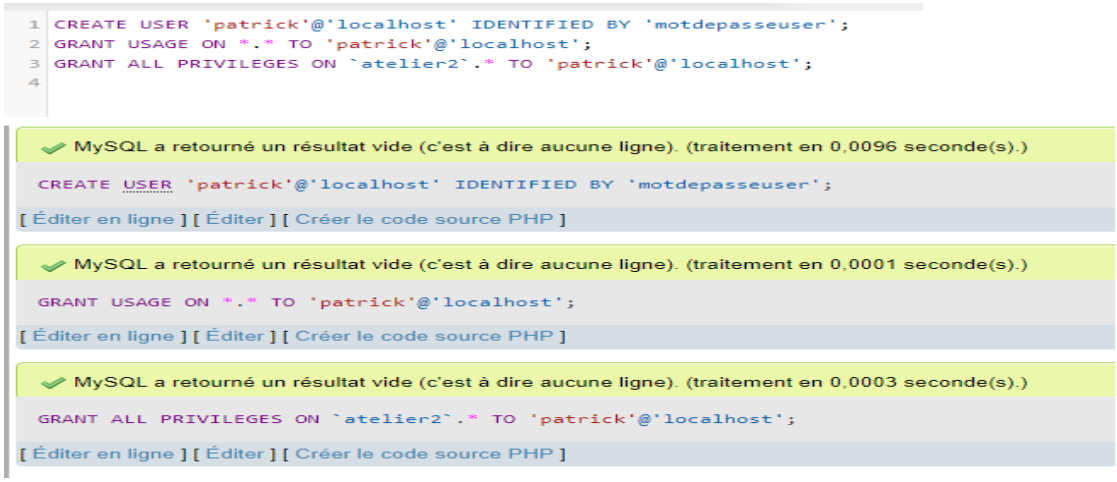




Puis avec les données générer et celles fournit, je remplis ma base de données



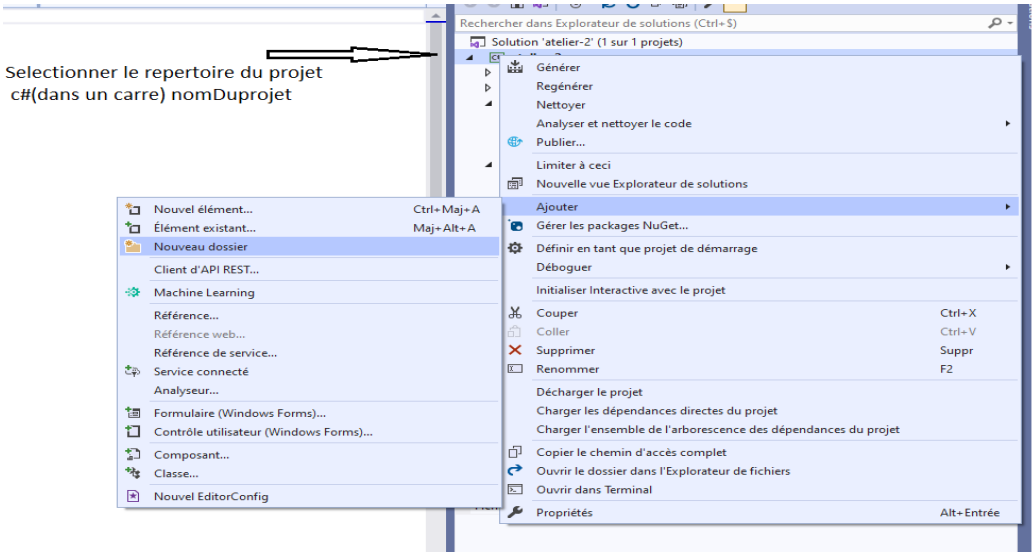
Je crée aussi la table responsable qui me permettra de filtrer l'accès au logiciel de gestion et dans laquelle je créais un responsable.
Et un utilisateur à qui j'élève les droits sur base de données du projet que j'utiliserai pour m'y connecter.



Etape #2 :

-Creation du projet et de ces dossiers

Création d'un nouveau projet windows form avec Visual Studio .
Je crée les différents dossiers du projet, la view , le controlleur, le dal et le model



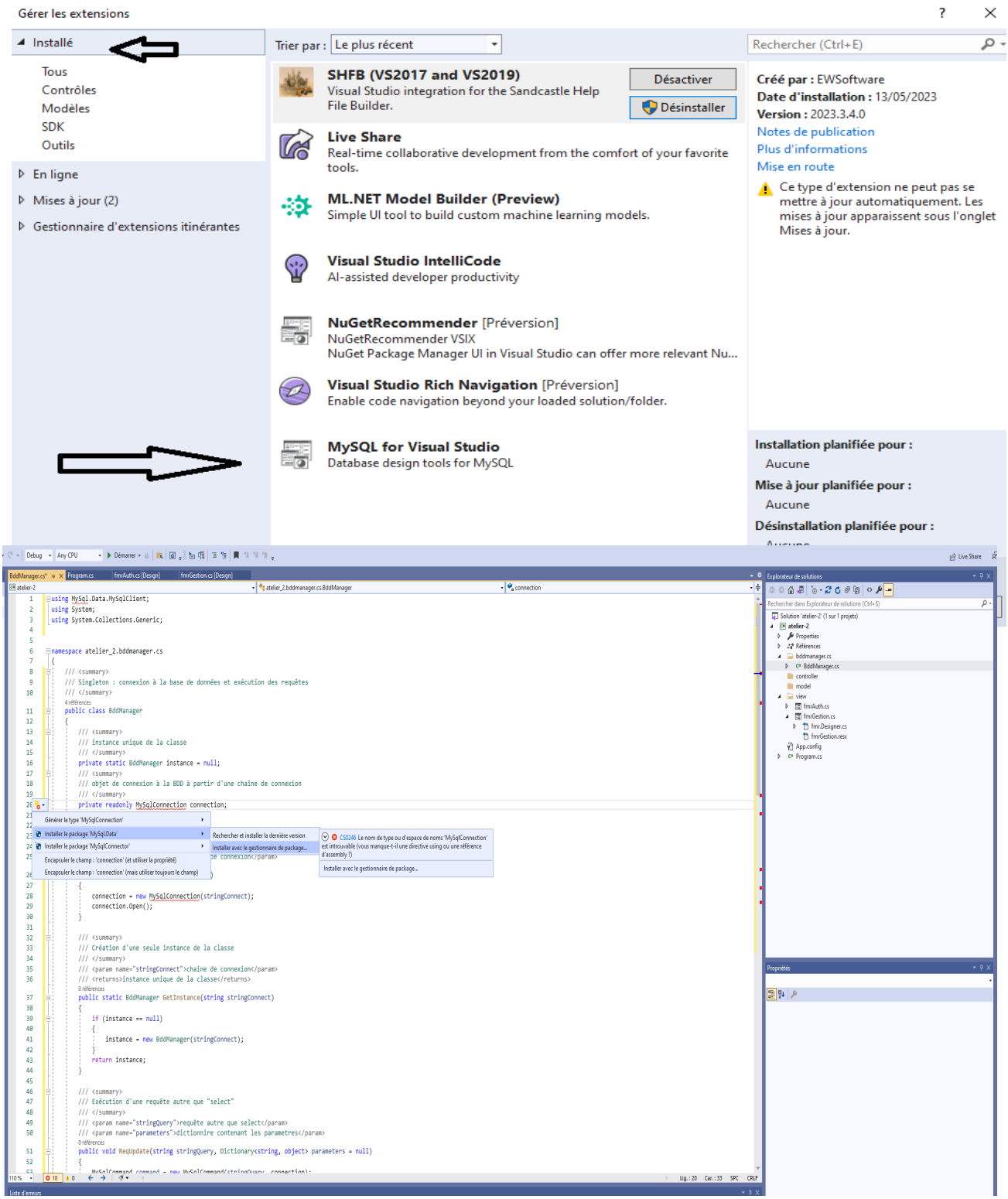
-Creation du kanban et du repos sur github

Je crée le répertoire github nécessaire au projet et le kanban qui lui est associé. N'ayant encore jamais utilisé Kanban, je lui fais afficher "hello world!" . Le test étants concluant je passe à la suite.

-Test de la connection a la base de données

Je récupère le package bddManager que j'incorpore à mon programme
et instale le packages MySql.Data. J'ai pu rencontrer des soucis en fonctionnent comme dans les cours. J'avais dû installer mySql directement depuis Visual Studio 2019 dans les extensions et a chaque nouveau programme installer le package pour que cela marche.

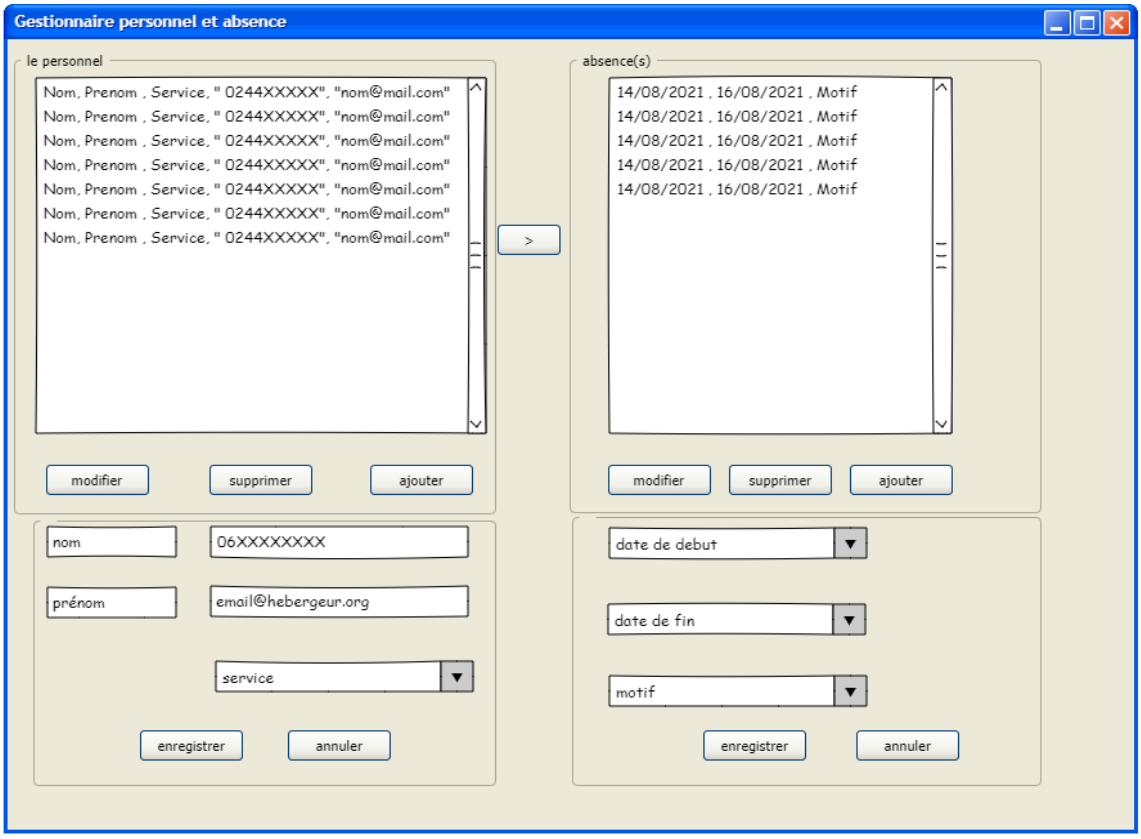
```
12 références
public class Access
{
    /// <summary>
    /// chaine de connexion à la bdd
    /// </summary>
    private static readonly string connectionString = "server = 127.0.0.1; user id = patrick ; Pwd = motdepasseuser ;  persistsecurityinfo=False;database=atelier2";
    /// <summary>
    /// instance unique de la classe
    /// </summary>
    private static Access instance = null;
```

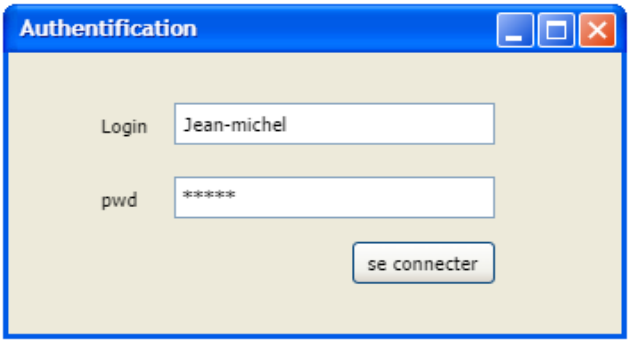


Je vérifie la connexion à ma base de données dans le menu de l'IDE, cela fonctionne, je passe à la suite

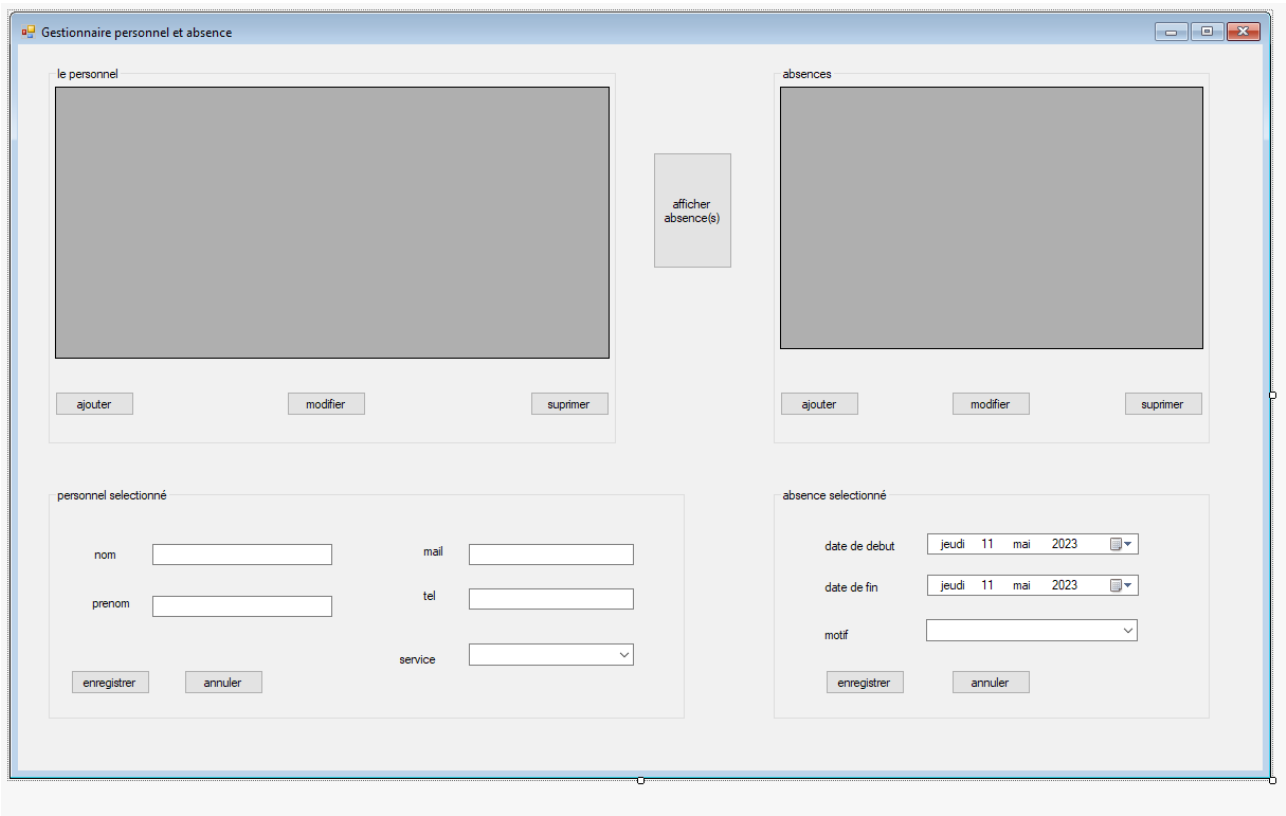
-Prototypage des vus

Après études des cas d'utilisation et du diagramme de la base de données, je dessine un prototype avec Pensil qui me servira de modèle a la création de mon interface graphique





-Mise en places des vus



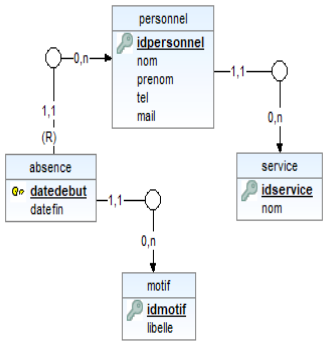
Comme dans le programme étudié en cours "Habilitation" je fais le choix d'utiliser des DataGridView pour les affichages principaux. Les DateTimePicker seront aussi très pratique pour afficher les dates de début et date de fin de mes absences (la vue sera modifiée plus tard pour plus de praticités.)

Je clone mon répertoire avec gitbash et je fais mon premier commit .

Etape #3 :

-Code de la parti modèle, classes metiers

L'affichage se fessant avec des objets de type BindingSource qui permet les contenant d'une classe ou leur méthode ToString(), je fais le choix de créer une classe pour chaque table de ma base. À la lecture du modèle conceptuel de données, on peut constater que chaque personnel fait partie d'un service et chaque absence a un motif. Je conçois les classes en conséquence.



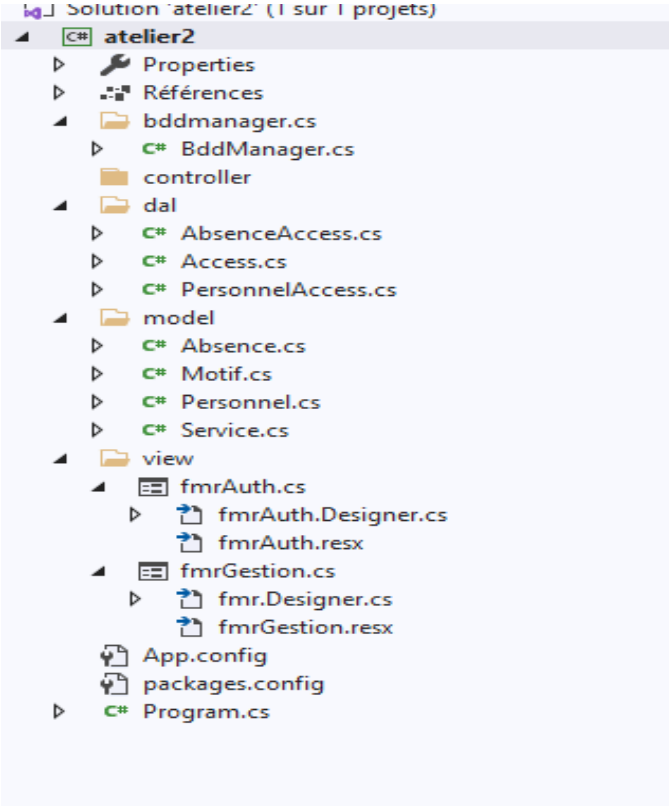

```
namespace atelier2.model
{
    /// <summary>
    ///  Classe métier liée à la table Personnel
    /// </summary>
    1 référence
    class Personnel
    {
        /// <summary>
        /// Valorise les propriétés
        /// </summary>
        /// <param name="idpersonnel"></param>
        /// <param name="nom"></param>
        /// <param name="prenom"></param>
        /// <param name="tel"></param>
        /// <param name="mail"></param>
        /// <param name="service"></param>
        0 références
        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;
        }
        1 référence
        public int Idpersonnel { get; }
        1 référence
        public string Nom { get; set; }
        1 référence
        public string Prenom { get; set; }
        1 référence
        public string Tel { get; set; }
        1 référence
        public string Mail { get; set; }

        1 référence
        public Service Service { get; set; }
    }
}
```

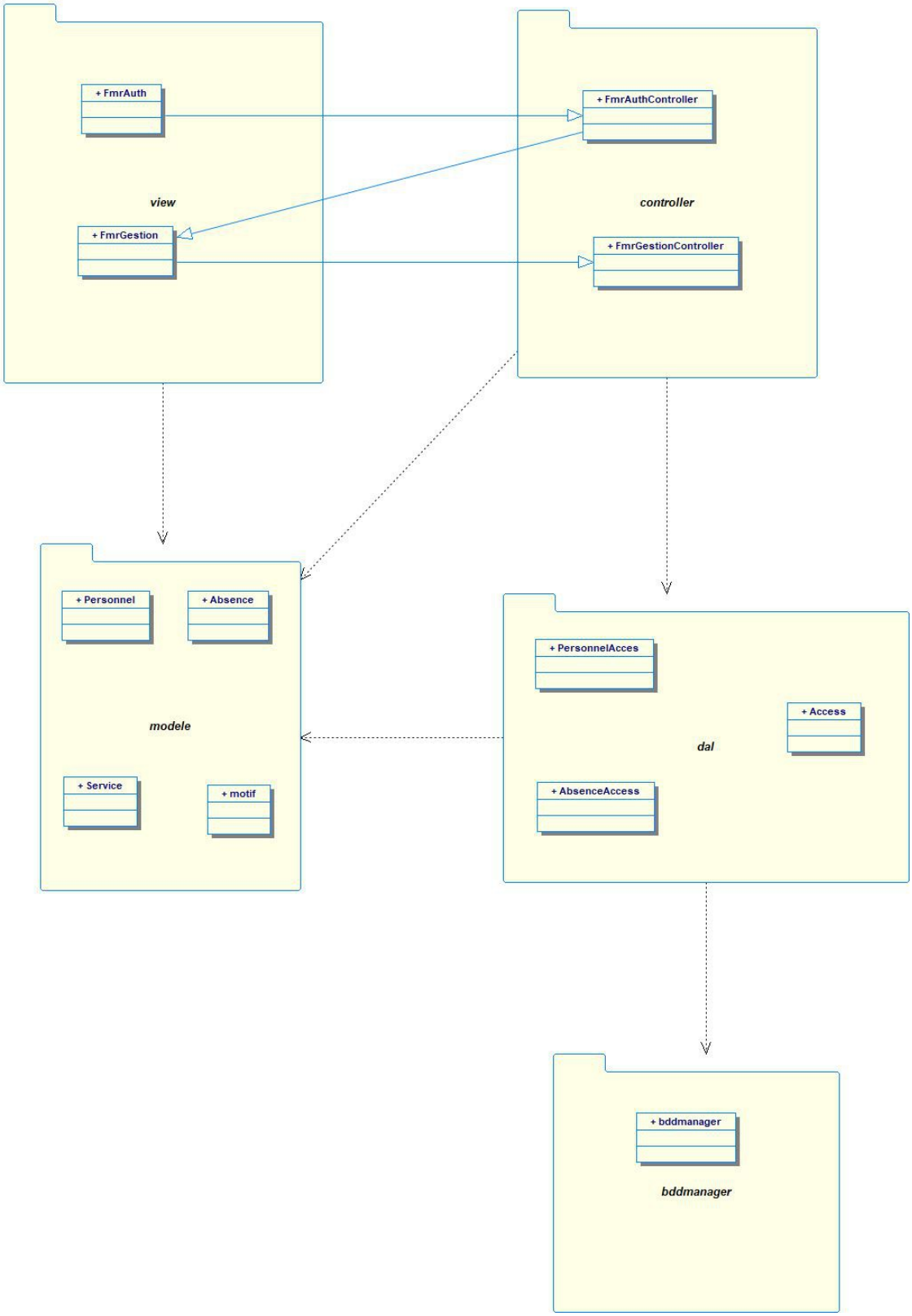
Les variables de classe sont en public, majuscule avec setter et getter pour pouvoir être instancié ou modifié depuis le dal correspondant, ici personnelAcces . Pour rester cohérent avec la base de données les variables ''clef primaire n'ont pas de setter.
Je crée une méthode override ToString() pour que les objets de type service et motif ne m'affiche que leur nom/libelle dans mes dataGridView .

-Implementation du dal

Je crée mes différentes classes qui me permettront par la suite d'accéder a ma base de données. Access classe en sigleton me permettant de centraliser la seule et unique instance de Bddmanager (reprise comme BddaManager du td "Habilitations") .
Pour la gestion du personnel et celle des absences, je crée PersonnelAccess et AbsencesAccess que je compléterai plus tard.

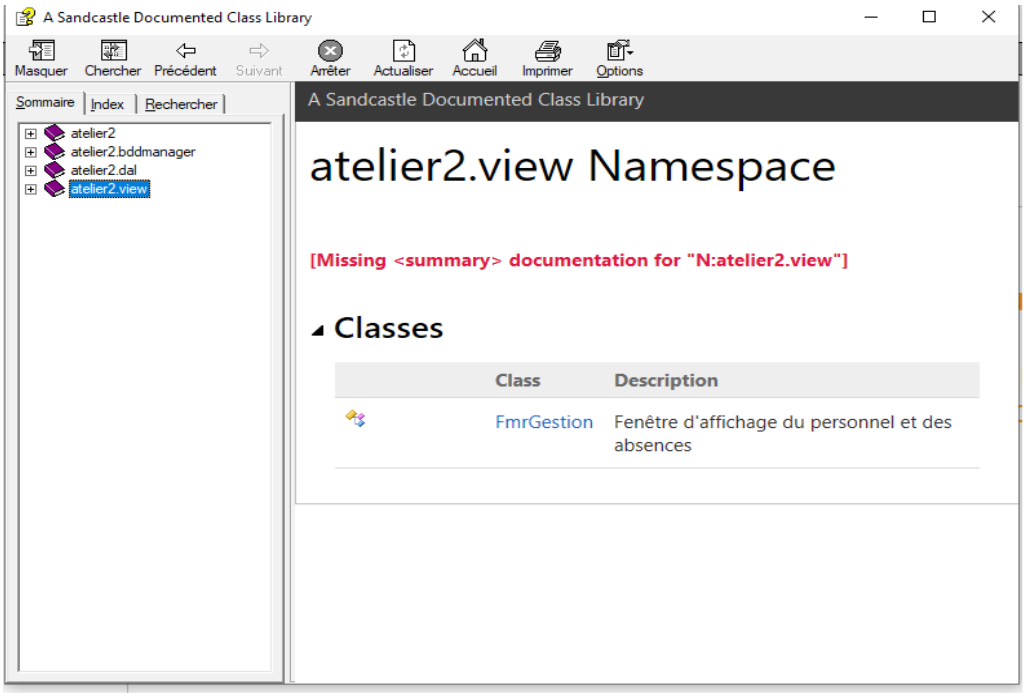


Je dessine un brouillon de l'architecture de mon projet pour pouvoir développer plus rapidement.


+ FmrAuth
view
+ FmrGestion
+ FmrAuthController
controller
+ FmrGestionController
+ Personnel
+ Absence
modele
+ Service
+ motif
+ PersonnelAcces
+ Access
dal
+ AbsenceAccess
+ bddmanager
bddmanager

-Génération de la premiere documentation technique au format chm

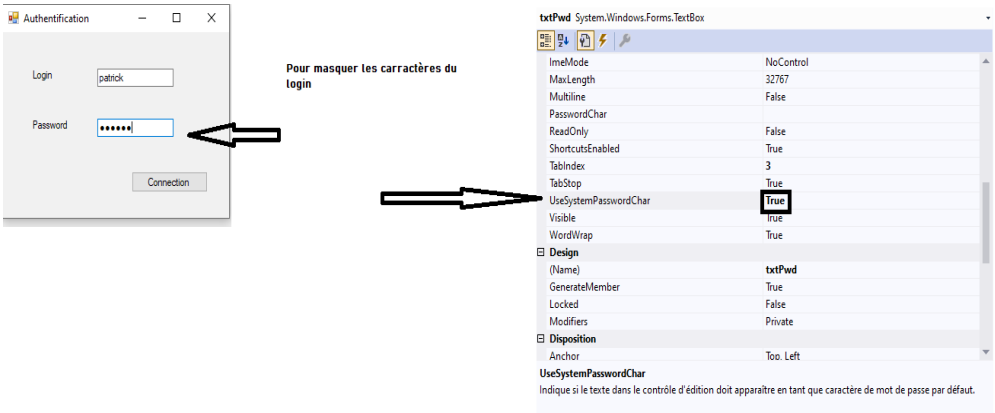
Après quelques difficultés a comprendre le fonctionnement du logiciel Sandcastle
je génère la première documentation technique du projet en format chm



Etape #4 :

-Code de la connection (de la form jusqu'au dal):

creation et code de dal.connectionAcces, code des fonctionnalités de la vue et de son controleur



Je crée un controller pour la forme, FmrAuthController(instancier par la vue (FmrAuth)) et ConnectionsAcces dans le dal avec qui je ferais la requette de verification de mot de passe.

```
/// <summary>
/// retourne vrai si la combinaison mdp pwd existe dans la base de données
/// </summary>
/// <param name="login"></param>
/// <param name="pwd"></param>
/// <returns>vrai ou faux</returns>
1 référence
public bool GetLaConnection(string login, string pwd)
{
    string req = "SELECT * from responsable WHERE login = @login AND pwd = SHA2(@pwd, 256);";
    Dictionary<string, object> parameters = new Dictionary<string, object>();
    parameters.Add("@login",(Object)login);
    parameters.Add("@pwd", (Object)pwd);
    try
    {
        List<Object[]> records = access.Manager.ReqSelect(req, parameters);
        if (records.Count == 1)
        {
            return true;
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        Environment.Exit(0);
    }
    return false;
}
```

-Code de la parti personnel (de la form jusqu'au dal) :

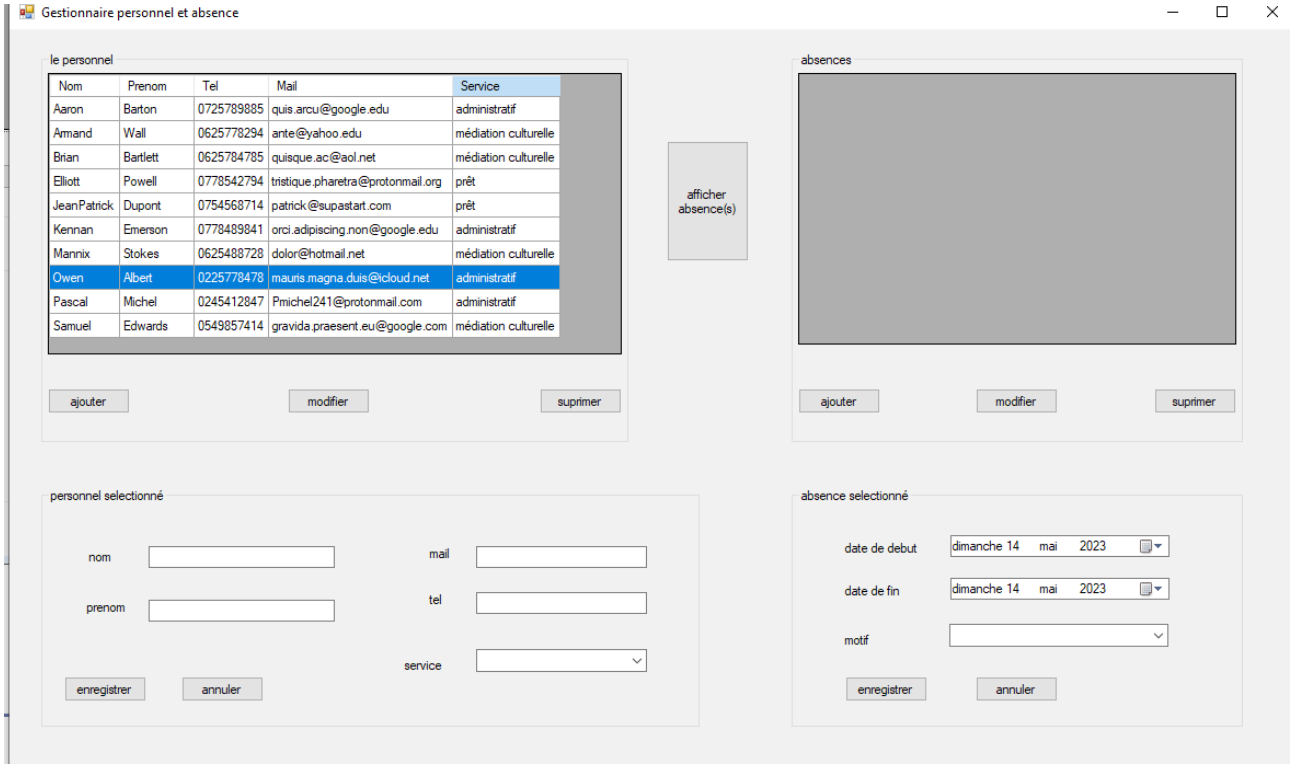
creation d'un deuxiéme controlleur, code de la parti personnel de la vue, du controlleur, creation et code de dal.personnelAcces

Si le retour est positif, mon contrôleur instancie ma deuxième vue (celle du logiciel de gestion) qui instanciera son contrôler.

Je crée donc la classe FmrGestionController qui me servira à traiter les demandes de la vue à la base de données (tout ce qui est dans le dal) et qui lui renverra à la vue les informations nécessaires si besoin.

Je créais la méthode GetLesPersonnel dans PersonnelAccess et dans le controller pour pouvoir tester l'affichage de ma première DataGridView dans la vue.
Je code rapidement un affichage en dur pour remplir les comboBox.
Je fais le choix conscient a ce moment du développement d'utiliser le pattern SOLID, malgré la lourdeur d'avoir un code peu factorisé et long, le fait d'avoir des responsabilités uniques et

une ségrégation des interfaces me permettra une meilleure maintenance et une meilleure lisibilité en cas de modification.



Le code étant fonctionnel, je passe à la suite et je code les différentes options pour gérer le personnel. Pour le moment je n'ai pas besoin de réfléchir car cela a déjà été fait de manière très similaire dans le td "Habilitations" .
Je code donc rapidement les différentes méthodes nécessaire dans le controlleur personnelAcces et la form en adaptant à la situation . Je supprime mon bouton ajouter, les labels des groupebox me permettant de guider l'utilisateur (ici, ils indiquent si un personnel est en cours de modification, ou si il est possible d'en créer un nouveau, il n'est pas possible de faire les deux en même temps)

-Code de la parti absences (de la form jusqu'au dal) :

code de la parti absence de la vue, du controlleur, creation et code de dal.absenceAcces

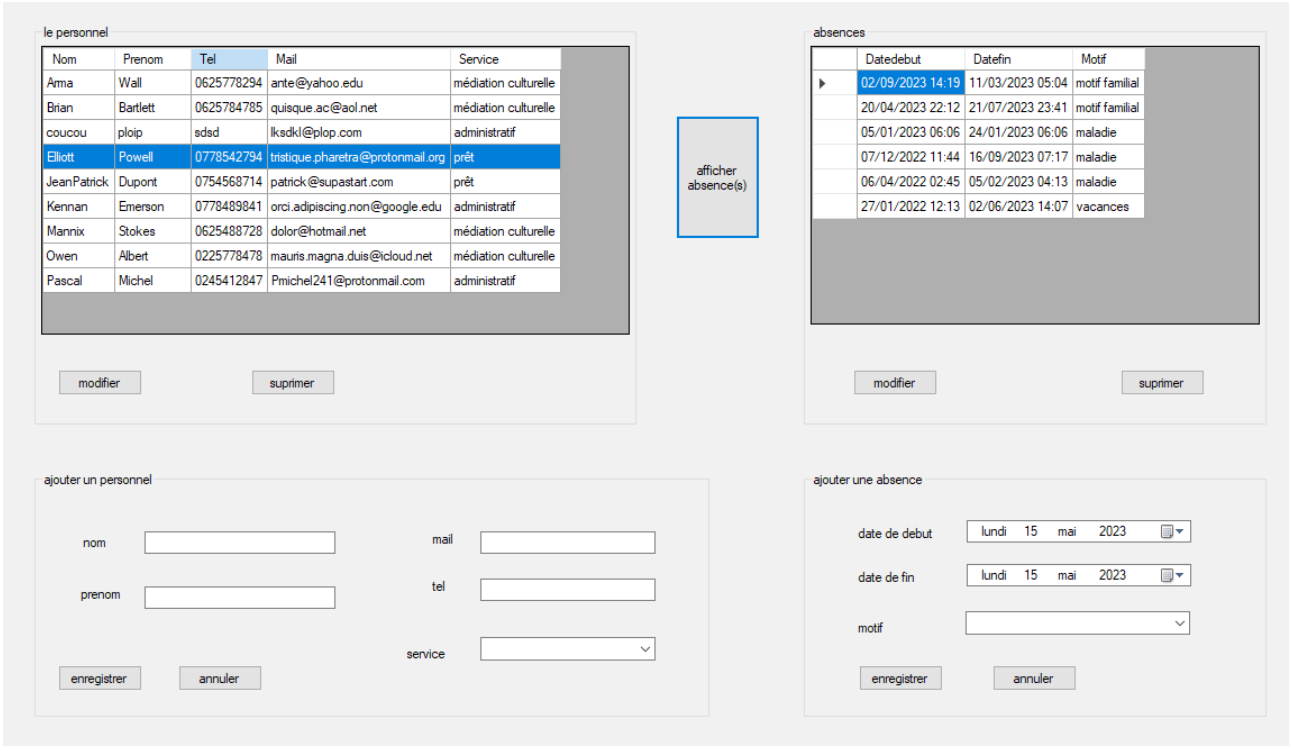
Sur le même schéma, je commence par coder les éléments me permettant de charger les absences d'un personnel dans le controller et AbsencesAccess. Si aucune ligne n'est sélectionné dans la dataGridView où est affiché le personnel, je code le bouton "afficher les absences" pour qu'il avertisse l'utilisateur qu'il doit sélectionner une ligne et il ne ce passe rien.
Dans le cas contraire le personnel sélectionné est assigné a une variable de classe de ma form pour faciliter la gestion. En cas d'action sur la liste de personnel, remplirListePersonnel() s'actualise et actualise l'assignation du personnel sélectionné .
Je code remplirListeAbsence() sur le même modèle que remplirListepersonnel().

```
/// <summary>
/// Affiche le personnel
/// </summary>
3 références
private void RemplirListePersonnel()
{
    List<Personnel> lesPersonnels = controller.GetLesPersonnels();
    bdgPersonnel.DataSource = lesPersonnels;
    dgvPerso.DataSource = bdgPersonnel;
    dgvPerso.Columns["idpersonnel"].Visible = false;
    dgvPerso.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    dgvPerso.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
}

/// <summary>
/// Affiche les absences du personnel selectionné
/// </summary>
/// <param name="personnel"></param>
4 références
private void RemplirListeAbsence(Personnel personnel)
{
    List<Absence> LesAbsences = controller.GetLesAbsences(personnel);
    bdgAbsence.DataSource = LesAbsences;
    dgvAbs.DataSource = bdgAbsence;
    dgvAbs.Columns["idpersonnel"].Visible = false;
    dgvAbs.Columns["datedebut"].DefaultCellStyle.Format = "d/M/yyyy";
    dgvAbs.Columns["datefin"].DefaultCellStyle.Format = "d/M/yyyy";
    dgvAbs.Columns["datedebut"].HeaderText = "Date de debut";
    dgvAbs.Columns["datefin"].HeaderText = "Date de fin";
    dgvAbs.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    dgvAbs.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    gbxAbsences.Text = "absence de " + personnel.Nom + " " + personnel.Prenom;
    gbxAbs.Enabled = true;
}
```

```
1 reference
public List<Absence> GetLesAbsences(Personnel personnel)
{
    List<Absence> lesAbsences = new List<Absence>();
    if (access.Manager != null)
    {
        string req = "select a.idpersonnel as idpersonnel, a.datedebut as datedebut, a.datefin as datefin, a.idmotif as idmotif, m.libelle as libelle ";
        req += "from absence a join motif m on (a.idmotif = m.idmotif) ";
        req += "where idpersonnel = @idpersonnel ";
        req += "order by datedebut desc ";
        try
        {
            Dictionary<string, object> parameters = new Dictionary<string, object>();
            parameters.Add("@idpersonnel",personnel.Idpersonnel);
            List<Object[]> records = access.Manager.ReqSelect(req, parameters);
            if (records != null)
            {
                Console.WriteLine(records.Count);

                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;
}
```



L'affichage fonctionnant je passe aux différentes options de la partie absences. Je recopie les fonctions très proche de PersonnelAccess dans AbsencesAccess et change les noms. Les dateTimePicker en C# étant très simple d'utilisation avec les objets de type dateTime la difficulté a été de convertir les MySqlDateTime en objet de type DateTime et inversement. Après une rapide recherche j'arrive à obtenir quelque chose de fonctionnel . Le souci se situe au niveau des heures et des secondes. J'ai fait le choix arbitraire sur mon programme de ne pouvoir paramétrer que les jours dans l'affichage alors que mon format de données s'arrête a l'échelle des secondes.

```
1 référence
public void AddAbsence(Absence absence)
{
    if (access.Manager != null)
    {
        Console.WriteLine("AddAbsence");

        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.ToString("yyyy-MM-dd HH:mm"));
        parameters.Add("@datefin", absence.Datefin.ToString("yyyy-MM-dd HH:mm"));
        parameters.Add("@idmotif", absence.Motif.Idmotif);
        try
        {
            access.Manager.ReqUpdate(req, parameters);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            Environment.Exit(0);
        }
    }
}
```

DateTime.ToString pour etre au format datetime MysQl

```
    /// <summary>
    /// Demande de supression d'une absence
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 référence
    private void BtnAbsSuprimer_Click(object sender, EventArgs e)
    {
        if (dgvAbs.SelectedRows.Count > 0)
        {
            Absence absence = (Absence)bdgAbsence.List[bdgAbsence.Position];
            if (MessageBox.Show("Voulez-vous vraiment supprimer l'" + gbxAbsences.Text + " du "  + absence.Datedebut.ToString("d/M/yyyy")
                + " au " + absence.Datefin.ToString("d/M/yyyy") + " ?", "Confirmation de suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                controller.DelAbsence(absence);
                Console.WriteLine("suprimer");
                RemplirListeAbsence(personnelSelect);
            }
        }
        else
        {
            MessageBox.Show("Une ligne doit être sélectionnée.", "Information");
        }
    }
```

Je bricole pour que cela soit cohérent : a l'utilisation quand on modifie ou ajoute une absence mon programme vérifie si le jour n'a pas déjà été renseigné pour ce personnel.

```
/// <summary>
/// Verifie qu'une absence n'a pas deja été assigné le même jour
/// </summary>
/// <param name="absence"></param>
/// <returns></returns>
2 références
private bool VerifAbs(Absence absence)
{
    int compt = 0;
    if (enCoursDeModifAbsence)
        compt--;
    foreach (Absence absenceAVerifier in bdgAbsence.List)
    {
        if (absence.Datedebut.ToString("yyyy-MM-dd") == absenceAVerifier.Datedebut.ToString("yyyy-MM-dd"))
        {
            compt++;
            if (compt == 1)
            {
                MessageBox.Show("Cette date est deja renseigné", "Information");
                return false;
            }
        }
    }
    return true;
}
```

La faite de vérifier si une absence n'empiète ou n'est pas comprise dans une autre n'étant pas dans le cahier des charges, je ne l'ai pas rajouté, mais en cas de demande cela peu être intégré très rapidement au programme.
On ne peut pas modifier une absence quand une ou plusieurs son renseigné le même jour. Si on veut changer la date de début d'une absence, il faut la supprimer et la recréer.

L'etat de mon controlleur a ce moment du devellopement (test avant creation des commentaires)

```
3 références
public class FmrGestionController
{
    /// <summary>
    /// objet d'accès aux opérations possibles sur Personnel
    /// </summary>
    private readonly PersonnelAccess personnelAccess;
    /// <summary>
    /// objet d'accès aux opérations possibles sur Absence
    /// </summary>
    private readonly AbsenceAccess absenceAccess;

    /// <summary>
    ///
    /// </summary>
    1 référence
    public FmrGestionController()
    {
        personnelAccess = new PersonnelAccess();
        absenceAccess = new AbsenceAccess();
    }

    1 référence
    public List<Personnel> GetLesPersonnels()
    {
        return personnelAccess.GetLesPersonnels();
    }
```

```
1 reference
public List<Absence> GetLesAbsences(Personnel personnel)
{
    return absenceAccess.GetLesAbsences(personnel);
}

1 référence
public void AddPersonnel(Personnel personnel)
{
    personnelAccess.AddPersonnel(personnel);
}

1 référence
public void AddAbsence(Absence absence)
{
    absenceAccess.AddAbsence(absence);
}

1 référence
public void UpdatePersonnel(Personnel personnel)
{
    personnelAccess.UpdatePersonnel(personnel);
}

1 référence
public void UpdateAbsence(Absence absence)
{
    absenceAccess.UpdateAbsence(absence);
}

1 référence
public void DelPersonnel(Personnel personnel)
{
    personnelAccess.DelPersonnel(personnel);
}

1 référence
public void DelAbsence(Absence absence)
{
    absenceAccess.DelAbsence(absence);
}
```

-Code de l'initialisation des parti graphique pour les serives et les motifs
code du dal initAcces et du code nécessaire a son fonctionnement dans la vue et le contrôleur

Le remplissage de mes comboBox se fessait pour le moment en dur, ce qui n'est pas très évolutif (en cas d'ajout de table dans la base de données le programme ne marcherait plus). Sur les conseils de ma professeure, je décide donc de les charger depuis ma base de données.
Pour ne pas saturer de demande le SGBD, la liste de motifs et de services ne seront chargés qu'à l'initialisation de fmrGestion pour etre assigné au comboBox.
Pour conserver le style de conception du programme ou chaque fonction a une tache unique, je crée une classe InitAccess contenant les méthodes getLesServices() et getLesMotifs().
L'architecture du programme me permet d'effectuer très rapidement ces modifications, même si le code peu ce "répéter", il est très simple d'utilisation et organisé.

```
/// <summary>
/// Initialisation:
/// Creation du controleur et remplissage des listes
/// </summary>
1 référence
private void Init()
{
    controller = new FmrGestionController();
    RemplirListePersonnel();
    gbxAbs.Enabled = false;
    EnCourseDeModifPersonnel(false);
    EnCourseDeModifAbsence(false);
    RemplirListeService();
    RemplirListeMotif();
}
```

Dans FmrGestion (la vue)

```
/// <summary>
/// Affiche les services
/// </summary>
1 référence
private void RemplirListeService()
{
    List<Service> lesServices = controller.GetLesServices();
    bdgServices.DataSource = lesServices;
    cbxService.DataSource = bdgServices;
}

/// <summary>
/// Affiche les motifs
/// </summary>
1 référence
private void RemplirListeMotif()
{
    List<Motif> lesMotifs = controller.GetLesMotifs();
    bdgMotifs.DataSource = lesMotifs;
    cbxMotif.DataSource = bdgMotifs;
}
```

-Finalisation :

modifications, testes et verifications des diverses fonctionnalités

Le programme ne supprime pas les absences d'un personnel, quand le personnel est supprimé, je code cette fonctionnalité. Je n'ai qu'à rajouter un module DelLesAbsences dans AbsencesAcces et dans le controller, à qui je fournis avant de supprimer un personnel, l'objet de ce personnel.

```
/// <summary>
/// Demande de supression d'un personnel
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 référence
private void BtnProSuprimer_Click(object sender, EventArgs e)
{
    if (dgvPerso.SelectedRows.Count > 0)
    {
        Personnel personnel = (Personnel)bdgPersonnel.List[bdgPersonnel.Position];
        if (MessageBox.Show("Voulez-vous vraiment supprimer " + personnel.Nom + " " + personnel.Prenom + " ?", "Confirmation de suppression", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            controller.DelLesAbsences(personnel);
            controller.DelPersonnel(personnel);
            RemplirListePersonnel();
            if (dgvAbs.RowCount > 0)
            {
                Personnel personnelSelect = (Personnel)bdgPersonnel.List[bdgPersonnel.Position];
                this.personnelSelect = personnelSelect;
                RemplirListeAbsence(personnelSelect);
            }
        }
    }
    else
    {
        MessageBox.Show("Une ligne doit être sélectionnée.", "Information");
    }
}
```

Dans FmrGestion (vue)

```
/// <summary>
/// Demande de suppression des absences d'une personnel
/// </summary>
/// <param name="personnel">le personnel dont on effasse les absences</param>
1 référence
public void DelLesAbsences(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);
        }
    }
}
```

Après débogages les testes sont concluants le programme est fonctionnel.

FIN