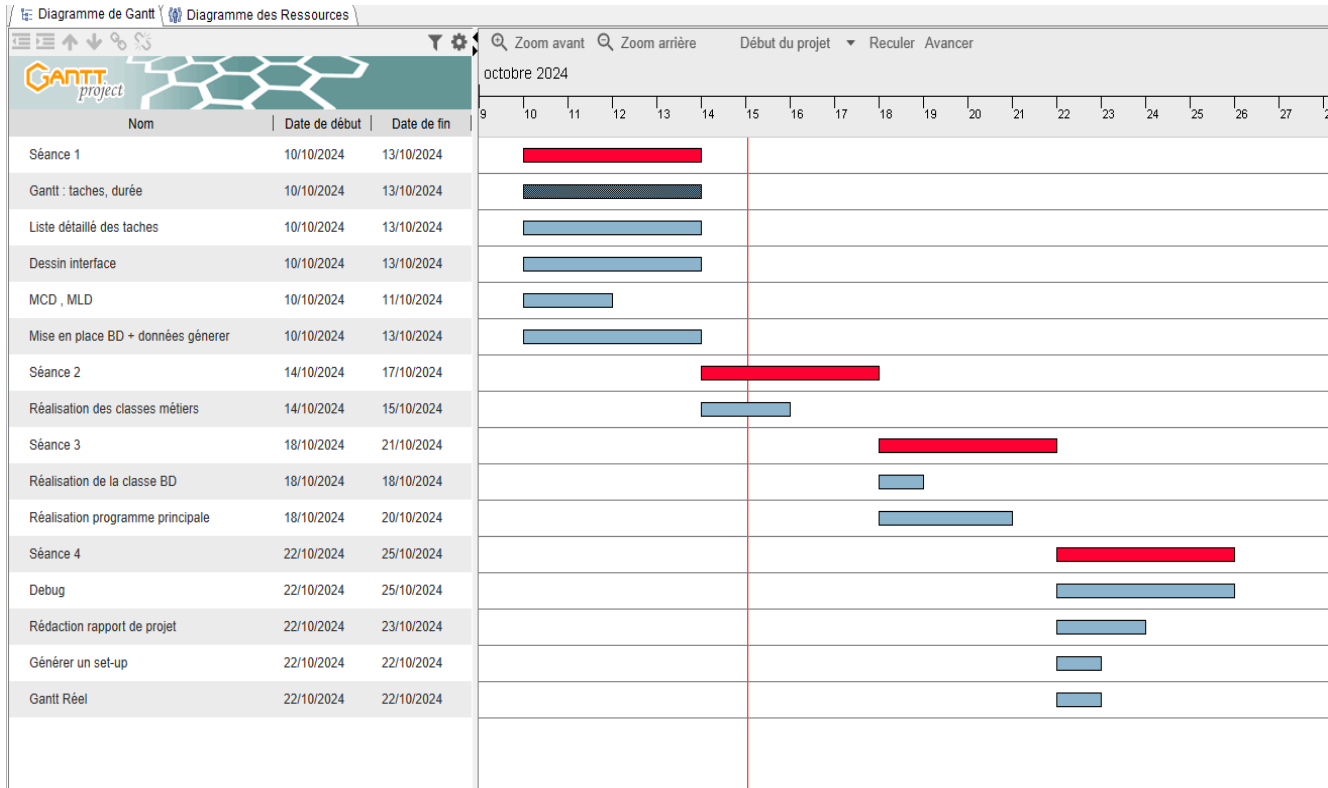


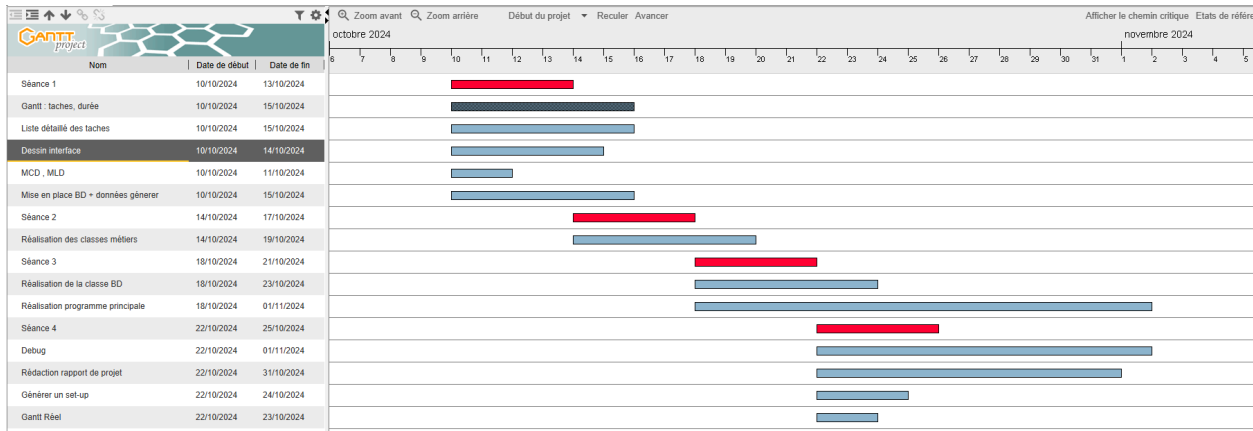
Documentation c# projet

Séance 1 : structure BD + prototype + gantt prévisionnel (tâches + ressources)

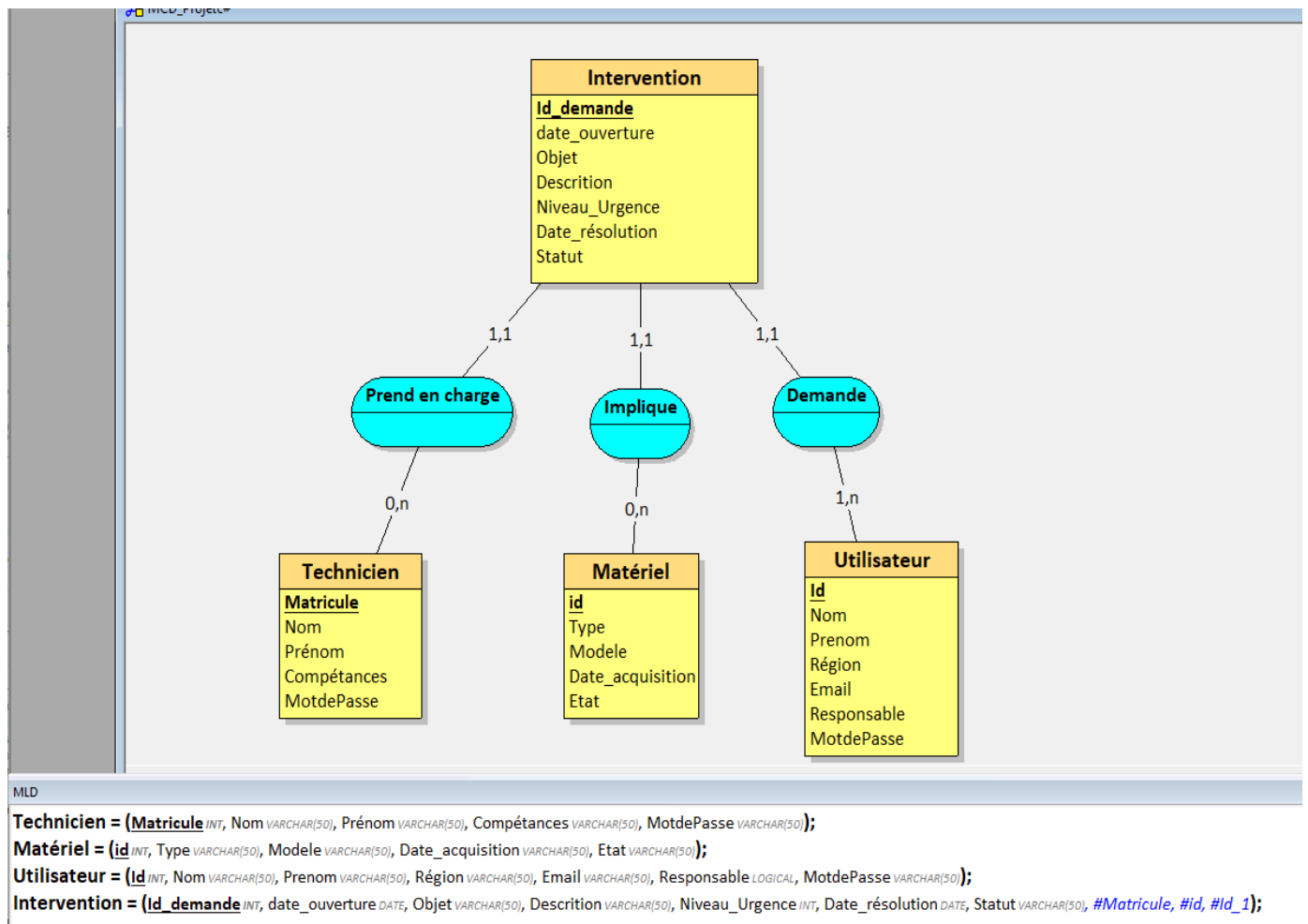
Gantt prévisionnel :



Gantt final:



Structure BD (MCD MLD) :



Prototype (interface):

Connexion Utilisateur Technicien Responsable

Identifiant :

Mot de passe :

séance 2 : mise en place BD + réalisation des classes (classes issues des tables en BD)

Technicien :

```
internal class Technicien
{
    private int matricule;
    private string nom;
    private string prenom;
    private string competences;
    private string motDePasse;

    /// <summary>Initialise une nouvelle instance de la classe Technicien.</summary>
    /// <param name="matricule">Le matricule du technicien.</param>
    /// <param name="nom">Le nom du technicien.</param>
    /// <param name="prenom">Le prénom du technicien.</param>
    /// <param name="competences">Les compétences du technicien.</param>
    /// <param name="motDePasse">Le mot de passe du technicien.</param>
    public Technicien(int matricule, string nom, string prenom, string competences, string
motDePasse)
    {
        this.matricule = matricule;
        this.nom = nom;
        this.prenom = prenom;
        this.competences = competences;
        this.motDePasse = motDePasse;
    }

    /// <summary>Obtient le matricule du technicien.</summary>
    public int getMatricule() { return matricule; }
    /// <summary>Obtient le nom du technicien.</summary>
    public string getNom() { return nom; }
    /// <summary>Obtient le prénom du technicien.</summary>
    public string getPrenom() { return prenom; }
    /// <summary>Obtient les compétences du technicien.</summary>
    public string getCompetences() { return competences; }
    /// <summary>Obtient le mot de passe du technicien.</summary>
    public string getMotDePasse() { return motDePasse; }
    /// <summary>Définit les compétences du technicien.</summary>
    public void setCompetences(string competences) { this.competences = competences; }
}
}
```

Matériel :

```
internal class Materiel
{
    private int id;
    private string type;
    private string modele;
    private DateTime dateAcquisition;
    private string etat;

    /// <summary>Initialise une nouvelle instance de la classe Materiel.</summary>
    /// <param name="id">L'ID du matériel.</param>
    /// <param name="type">Le type de matériel.</param>
    /// <param name="modele">Le modèle de matériel.</param>
    /// <param name="dateAcquisition">La date d'acquisition du matériel.</param>
    /// <param name="etat">L'état du matériel.</param>
    public Materiel(int id, string type, string modele, DateTime dateAcquisition, string etat)
    {
        this.id = id;
        this.type = type;
        this.modele = modele;
        this.dateAcquisition = dateAcquisition;
        this.etat = etat;
    }

    /// <summary>Obtient l'ID du matériel.</summary>
    public int getId() { return id; }
    /// <summary>Obtient le type de matériel.</summary>
    public string getType() { return type; }
    /// <summary>Obtient le modèle de matériel.</summary>
    public string getModele() { return modele; }
    /// <summary>Obtient la date d'acquisition du matériel.</summary>
    public DateTime getDateAcquisition() { return dateAcquisition; }
    /// <summary>Obtient l'état du matériel.</summary>
    public string getEtat() { return etat; }
    /// <summary>Définit l'état du matériel.</summary>
    public void setEtat(string etat) { this.etat = etat; }
}
```

Utilisateur :

```
internal class Utilisateur
{
    private int id;
    private string nom;
    private string prenom;
    private string region;
    private string email;
    private int responsable;
    private string motDePasse;

    /// <summary>Initialise une nouvelle instance de la classe Utilisateur.</summary>
    /// <param name="id">L'ID de l'utilisateur.</param>
    /// <param name="nom">Le nom de l'utilisateur.</param>
    /// <param name="prenom">Le prénom de l'utilisateur.</param>
    /// <param name="region">Région de l'utilisateur.</param>
    /// <param name="email">L'email de l'utilisateur.</param>
    /// <param name="responsable">Responsabilité de l'utilisateur.</param>
    /// <param name="motDePasse">Le mot de passe de l'utilisateur.</param>
    public Utilisateur(int id, string nom, string prenom, string region, string email, int responsable,
string motDePasse)
    {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
        this.region = region;
        this.email = email;
        this.responsable = responsable;
        this.motDePasse = motDePasse;
    }

    /// <summary>Obtient l'ID de l'utilisateur.</summary>
    public int getId() { return id; }
    /// <summary>Obtient le nom de l'utilisateur.</summary>
    public string getNom() { return nom; }
    /// <summary>Obtient le prénom de l'utilisateur.</summary>
    public string getPrenom() { return prenom; }
    /// <summary>Obtient l'email de l'utilisateur.</summary>
    public string getRegion() { return region; }
    /// <summary>Obtient la région de l'utilisateur.</summary>
    public string getEmail() { return email; }
    /// <summary>Obtient le mot de passe de l'utilisateur.</summary>
    public int getResponsable() { return responsable; }
    /// <summary>Obtient la reponsabilité de l'utilisateur.</summary>
```

```

    public string getMotDePasse() { return motDePasse; }
    /// <summary>Définit le mot de passe de l'utilisateur.</summary>
    public void setMotDePasse(string motDePasse) { this.motDePasse = motDePasse; }
}
}

```

Intervention :

```

internal class Intervention
{
    private int idDemande;
    private DateTime dateOuverture;
    private string objet;
    private string description;
    private int niveauUrgence;
    private DateTime dateResolution;
    private string statut;

    /// <summary>Initialise une nouvelle instance de la classe Intervention.</summary>
    /// <param name="idDemande">L'ID de la demande.</param>
    /// <param name="dateOuverture">La date d'ouverture de l'intervention.</param>
    /// <param name="objet">L'objet de l'intervention.</param>
    /// <param name="description">La description de l'intervention.</param>
    /// <param name="niveauUrgence">Le niveau d'urgence de l'intervention.</param>
    /// <param name="dateResolution">La date de résolution de l'intervention.</param>
    /// <param name="statut">Le statut de l'intervention.</param>
    public Intervention(int idDemande, DateTime dateOuverture, string objet, string description, int
niveauUrgence, DateTime dateResolution, string statut)
    {
        this.idDemande = idDemande;
        this.dateOuverture = dateOuverture;
        this.objet = objet;
        this.description = description;
        this.niveauUrgence = niveauUrgence;
        this.dateResolution = dateResolution;
        this.statut = statut;
    }
}

```

```

    /// <summary>Obtient l'ID de la demande.</summary>
    public int getIdDemande() { return idDemande; }
    /// <summary>Obtient la date d'ouverture de l'intervention.</summary>
    public DateTime getDateOuverture() { return dateOuverture; }
    /// <summary>Obtient l'objet de l'intervention.</summary>
    public string getObjet() { return objet; }
    /// <summary>Obtient la description de l'intervention.</summary>
    public string getDescription() { return description; }
    /// <summary>Obtient le niveau d'urgence de l'intervention.</summary>
    public int getNiveauUrgence() { return niveauUrgence; }
    /// <summary>Obtient la date de résolution de l'intervention.</summary>
    public DateTime getDateResolution() { return dateResolution; }
    /// <summary>Obtient le statut de l'intervention.</summary>
    public string getStatut() { return statut; }
    /// <summary>Définit le niveau d'urgence de l'intervention.</summary>
    public void setNiveauUrgence(int niveauUrgence) { this.niveauUrgence = niveauUrgence; }
    /// <summary>Définit le statut de l'intervention.</summary>
    public void setStatut(string statut) { this.statut = statut; }
    /// <summary>Définit la date d'ouverture de l'intervention.</summary>
    public void setDateOuverture(DateTime dateOuverture) { this.dateOuverture = dateOuverture; }
    /// <summary>Définit la date de résolution de l'intervention.</summary>
    public void setDateResolution(DateTime dateResolution) { this.dateResolution =
dateResolution; }
}

```

séance 3 : réalisation de la classe BD + réalisation du programme principal

Classe BD :

```

/// <summary>
/// Classe BD qui permet d'interagir avec la base de données.
/// Contient des méthodes pour ajouter et récupérer des données concernant les techniciens,
/// utilisateurs, matériels et interventions.
/// </summary>
internal class BD
{
    // Chaîne de connexion à la base de données
    private static string connectionString = "Server=127.0.0.1 ; Database=laboratoire; Uid=root ;
Password=";

    /// <summary>
    /// Ajoute un technicien dans la base de données.
    /// </summary>

```

```

/// <param name="technicien">L'objet Technicien à ajouter.</param>
public static void AjouterTechnicien(Technicien technicien)
{
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "INSERT INTO TECHNICIEN VALUES (@matricule, @nom,
@prenom, @competences)";
    mySqlCommand.Parameters.AddWithValue("@matricule", technicien.getMatricule());
    mySqlCommand.Parameters.AddWithValue("@nom", technicien.getNom());
    mySqlCommand.Parameters.AddWithValue("@prenom", technicien.getPrenom());
    mySqlCommand.Parameters.AddWithValue("@competences", technicien.getCompetences());
    mySqlCommand.ExecuteNonQuery();
    con.Close();
}

/// <summary>
/// Récupère la liste de tous les techniciens dans la base de données.
/// </summary>
/// <returns>Liste de Technicien.</returns>
public static List<Technicien> getLesTechniciens()
{
    Technicien unTechnicien;
    List<Technicien> lesTechniciens = new List<Technicien>();
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM Technicien";
    MySqlDataReader reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {
        unTechnicien = new Technicien(Convert.ToInt32(reader["matricule"]),
reader["nom"].ToString(), reader["prenom"].ToString(), reader["competences"].ToString(),
reader["motDePasse"].ToString());
        lesTechniciens.Add(unTechnicien);
    }
    con.Close();
    return lesTechniciens;
}

/// <summary>
/// Récupère la liste de tous les utilisateurs dans la base de données.
/// </summary>
/// <returns>Liste d'Utilisateur.</returns>
public static List<Utilisateur> getLesUtilisateurs()
{

```



```

Utilisateur unUtilisateur;
List<Utilisateur> lesUtilisateurs = new List<Utilisateur>();
MySQLConnection con = new MySqlConnection(connectionString);
con.Open();
MySQLCommand mySqlCommand = con.CreateCommand();
mySqlCommand.CommandText = "SELECT * FROM utilisateur";
MySQLDataReader reader = mySqlCommand.ExecuteReader();
while (reader.Read())
{
    unUtilisateur = new Utilisateur(Convert.ToInt32(reader["id"]), reader["nom"].ToString(),
reader["prenom"].ToString(), reader["region"].ToString(), reader["email"].ToString(),
Convert.ToInt32(reader["responsable"]), reader["motDePasse"].ToString());
    lesUtilisateurs.Add(unUtilisateur);
}
con.Close();
return lesUtilisateurs;
}

```

```

/// <summary>
/// Récupère la liste de tous les matériels dans la base de données.
/// </summary>
/// <returns>Liste de Matériel.</returns>
public static List<Matériel> getLesMateriaux()
{
    Matériel unMatériel;
    List<Matériel> lesMateriaux = new List<Matériel>();
    MySQLConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySQLCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM Matériel";
    MySQLDataReader reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {
        unMatériel = new Matériel(Convert.ToInt32(reader["id"]), reader["type"].ToString(),
reader["modele"].ToString(), Convert.ToDateTime(reader["date_acquisition"]),
reader["etat"].ToString());
        lesMateriaux.Add(unMatériel);
    }
    con.Close();
    return lesMateriaux;
}

```

```

/// <summary>
/// Récupère la liste de toutes les interventions dans la base de données.
/// </summary>
/// <returns>Liste d'Intervention.</returns>

```

```

public static List<Intervention> getLesInterventions()
{
    Intervention uneIntervention;
    List<Intervention> lesInterventions = new List<Intervention>();
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM intervention";
    MySqlDataReader reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {
        uneIntervention = new Intervention(Convert.ToInt32(reader["id_demande"]),
        Convert.ToDateTime(reader["date_ouverture"]), reader["objet"].ToString(),
        reader["description"].ToString(), Convert.ToInt32(reader["niveau_urgence"]),
        Convert.ToDateTime(reader["date_resolution"]), reader["statut"].ToString());
        lesInterventions.Add(uneIntervention);
    }
    con.Close();
    return lesInterventions;
}

/// <summary>
/// Ajoute une intervention dans la base de données.
/// </summary>
/// <param name="intervention">L'objet Intervention à ajouter.</param>
public static void AjouterIntervention(Intervention intervention)
{
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "INSERT INTO INTERVENTION VALUES (@id_demande,
@date_Ouverture, @objet, @description, @Niveau_Urgence, @date_Resolution, @statut)";
    mySqlCommand.Parameters.AddWithValue("@id_demande", intervention.getIdDemande());
    mySqlCommand.Parameters.AddWithValue("@date_Ouverture",
intervention.getDateOuverture());
    mySqlCommand.Parameters.AddWithValue("@objet", intervention.getObjet());
    mySqlCommand.Parameters.AddWithValue("@description", intervention.getDescription());
    mySqlCommand.Parameters.AddWithValue("@Niveau_Urgence",
intervention.getNiveauUrgence());
    mySqlCommand.Parameters.AddWithValue("@date_Resolution",
intervention.getDateResolution());
    mySqlCommand.Parameters.AddWithValue("@statut", intervention.getStatut());
    mySqlCommand.ExecuteNonQuery();
    con.Close();
}

```

```

/// <summary>
/// Ajoute un matériel dans la base de données.
/// </summary>
/// <param name="matériel">L'objet Matériel à ajouter.</param>
public static void AjouterMatériel(Matériel matériel)
{
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "INSERT INTO matériel VALUES (@type, @modele,
@date_acquisition, @etat)";
    mySqlCommand.Parameters.AddWithValue("@type", matériel.GetType());
    mySqlCommand.Parameters.AddWithValue("@modele", matériel.getModele());
    mySqlCommand.Parameters.AddWithValue("@date_acquisition",
matériel.getDateAcquisition());
    mySqlCommand.Parameters.AddWithValue("@etat", matériel.getEtat());
    mySqlCommand.ExecuteNonQuery();
    con.Close();
}

/// <summary>
/// Ajoute un utilisateur dans la base de données.
/// </summary>
/// <param name="utilisateur">L'objet Utilisateur à ajouter.</param>
public static void AjouterUtilisateur(Utilisateur utilisateur)
{
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "INSERT INTO utilisateur VALUES (@nom, @prenom,
@région, @email, @responsable)";
    mySqlCommand.Parameters.AddWithValue("@nom", utilisateur.getNom());
    mySqlCommand.Parameters.AddWithValue("@prenom", utilisateur.getPrenom());
    mySqlCommand.Parameters.AddWithValue("@région", utilisateur.getRegion());
    mySqlCommand.Parameters.AddWithValue("@email", utilisateur.getEmail());
    mySqlCommand.Parameters.AddWithValue("@responsable", utilisateur.getResponsable());
    mySqlCommand.ExecuteNonQuery();
    con.Close();
}
}

```

séance 4 : debug du programme + rédaction rapport de projet(voir doc attendue) + générer un set-up + gant réel

Programme principal:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace Projet_C__1
{
    public class BD
    {
        private static string connectionString = "Server=127.0.0.1 ; Database=laboratoire; Uid=root
; Password=";

        public static void AjouterTechnicien(Technicien technicien)
        {
            MySqlConnection con = new MySqlConnection(connectionString);
            con.Open();
            MySqlCommand mySqlCommand = con.CreateCommand();
            mySqlCommand.CommandText="INSERT INTO TECHNICIEN VALUES (@matricule,
@nom, @prenom, @competences";
            mySqlCommand.Parameters.AddWithValue("@matricule", technicien.getMatricule());
            mySqlCommand.Parameters.AddWithValue("@nom", technicien.getNom());
            mySqlCommand.Parameters.AddWithValue("@prenom", technicien.getPrenom());
            mySqlCommand.Parameters.AddWithValue("@competences",
technicien.getCompetences());
            mySqlCommand.ExecuteNonQuery();
            con.Close();
        }

        public static List<Technicien> getLesTechniciens()
        {
            Technicien unTechnicien;
            List<Technicien> lesTechniciens = new List<Technicien>();
            MySqlConnection con = new MySqlConnection(connectionString);
            con.Open();
            MySqlCommand mySqlCommand = con.CreateCommand();
            mySqlCommand.CommandText = "SELECT * FROM Technicien";
            MySqlDataReader reader = mySqlCommand.ExecuteReader();
            reader=mySqlCommand.ExecuteReader();
            while (reader.Read())
            {

```

```

        unTechnicien = new Technicien(Convert.ToInt32(reader["matricule"]),
reader["nom"].ToString(), reader["prenom"].ToString(), reader["competences"].ToString(),
reader["motDePasse"].ToString());
        lesTechniciens.Add(unTechnicien);
    }
    con.Close();
    return lesTechniciens;
}

```

```

public static List<Utilisateur> getLesUtilisateurs()
{
    Utilisateur unUtilisateur;
    List<Utilisateur> lesUtilisateurs = new List<Utilisateur>();
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM utilisateur";
    MySqlDataReader reader = mySqlCommand.ExecuteReader();
    reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {
        unUtilisateur = new Utilisateur(Convert.ToInt32(reader["id"]), reader["nom"].ToString(),
reader["prenom"].ToString(), reader["region"].ToString(), reader["email"].ToString(),
Convert.ToInt32(reader["responsable"]), reader["motDePasse"].ToString());
        lesUtilisateurs.Add(unUtilisateur);
    }
    con.Close();
    return lesUtilisateurs;
}

```

```

public static List<Materiel> getLesMateriaux()
{
    Materiel unMateriel;
    List<Materiel> lesMateriaux = new List<Materiel>();
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM Materiel";
    MySqlDataReader reader = mySqlCommand.ExecuteReader();
    reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {

```

```

        unMateriel = new Materiel(Convert.ToInt32(reader["id"]), reader["type"].ToString(),
reader["modele"].ToString(), Convert.ToDateTime(reader["date_acquisition"]),
reader["etat"].ToString());
        lesMateriaux.Add(unMateriel);
    }
    con.Close();
    return lesMateriaux;
}

```

```

public static List<Intervention> getLesInterventions()
{
    Intervention uneIntervention;
    List<Intervention> lesInterventions = new List<Intervention>();
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "SELECT * FROM intervention";
    MySqlDataReader reader = mySqlCommand.ExecuteReader();
    reader = mySqlCommand.ExecuteReader();
    while (reader.Read())
    {
        uneIntervention = new Intervention(Convert.ToInt32(reader["id_demande"]),
Convert.ToDateTime(reader["date_ouverture"]), reader["objet"].ToString(),
reader["description"].ToString(), Convert.ToInt32(reader["niveau_urgence"]),
Convert.ToDateTime(reader["date_resolution"]), reader["statut"].ToString());
        lesInterventions.Add(uneIntervention);
    }
    con.Close();
    return lesInterventions;
}

```

```

public static void AjouterIntervention(Intervention intervention)
{
    MySqlConnection con = new MySqlConnection(connectionString);
    con.Open();
    MySqlCommand mySqlCommand = con.CreateCommand();
    mySqlCommand.CommandText = "INSERT INTO INTERVENTION VALUES
(@id_demande, @date_Ouverture, @objet, @description, @Niveau_Urgence,
@date_Resolution, @statut";
    mySqlCommand.Parameters.AddWithValue("@id_demande",
intervention.getIdDemande());
}

```

```

        mySqlCommand.Parameters.AddWithValue("@date_Ouverture",
intervention.getDateOuverture());
        mySqlCommand.Parameters.AddWithValue("@objet", intervention.getObjet());
        mySqlCommand.Parameters.AddWithValue("@description",
intervention.getDescription());
        mySqlCommand.Parameters.AddWithValue("@Niveau_Urgence",
intervention.getNiveauUrgence());
        mySqlCommand.Parameters.AddWithValue("@date_Resolution",
intervention.getDateResolution());
        mySqlCommand.Parameters.AddWithValue("@statut", intervention.getStatut());
        mySqlCommand.ExecuteNonQuery();
        con.Close();
    }

    public static void AjouterMateriel(Materiel materiel)
    {
        MySqlConnection con = new MySqlConnection(connectionString);
        con.Open();
        MySqlCommand mySqlCommand = con.CreateCommand();
        mySqlCommand.CommandText = "INSERT INTO matériel VALUES (@type, @modele,
@date_acquisition, @etat";
        mySqlCommand.Parameters.AddWithValue("@type", materiel.GetType());
        mySqlCommand.Parameters.AddWithValue("@modele", materiel.getModele());
        mySqlCommand.Parameters.AddWithValue("@date_acquisition",
materiel.getDateAcquisition());
        mySqlCommand.Parameters.AddWithValue("@etat", materiel.getEtat());
        mySqlCommand.ExecuteNonQuery();
        con.Close();
    }

    public static void AjouterUtilisateur(Utilisateur utilisateur)
    {
        MySqlConnection con = new MySqlConnection(connectionString);
        con.Open();
        MySqlCommand mySqlCommand = con.CreateCommand();
        mySqlCommand.CommandText = "INSERT INTO utilisateur VALUES (@nom,
@prenom, @région, @email, @responsable";
        mySqlCommand.Parameters.AddWithValue("@type", utilisateur.getNom());
        mySqlCommand.Parameters.AddWithValue("@modele", utilisateur.getPrenom());
        mySqlCommand.Parameters.AddWithValue("@date_acquisition",
utilisateur.getRegion());
        mySqlCommand.Parameters.AddWithValue("@etat", utilisateur.getEmail());
        mySqlCommand.Parameters.AddWithValue("@etat", utilisateur.getResponsable());
    }

```

```

        mySqlCommand.ExecuteNonQuery();
        con.Close();
    }
}
}

```

Programme Principale page Technicien :

```

private void btnValiderIncident_Click(object sender, EventArgs e)
{
    Intervention intervention = new Intervention(tbDéclatIncident.Text, "",
textBoxIdPerso.Text, dateTimePickerDateDecla.Value, Convert.ToInt32(textBoxIdMateriel.Text),
textBoxIdTech.Text);
    BD.AjouterIntervention(intervention);
    MessageBox.Show("Exécuté");
}

```

/// <summary>

/// Événement déclenché lorsqu'un technicien ajoute un matériel.

/// </summary>

```

private void btnConfirmerMatériel_Click(object sender, EventArgs e)
{

```

```

    bool garantie = rbGarantieTrue.Checked;

```

// Correction : les contrôles de type TextBox contiennent des textes, donc utilisez leurs valeurs Text.

```

    Matériel nouveauMatériel = new Matériel(textBoxIdPerso.Text, textBoxIdTech.Text,

```

```

        Convert.ToDateTime(textBoxIdMateriel.Text), tbEtatInci.Text);

```

BD.AjouterMatériel(nouveauMatériel); // Méthode renommée pour correspondre à celle de la classe BD.

```

    ActualiserMatériel();
}

```

/// <summary>

/// Fonction pour actualiser les ComboBox de matériel.

/// </summary>

```

public void ActualiserMatériel()
{

```

```

    // Effacez d'abord tous les éléments de la ComboBox

```

```

    cbConsultMatériel.Items.Clear();

```

```

    cbSuppMater.Items.Clear();

```

// Correction : il est préférable de vérifier la validité de la liste obtenue.

```

    List<int> idMatériel = BD.getLesMateriaux();

```



```

        if (idMateriel != null)
        {
            foreach (int id in idMateriel)
            {
                cbConsultMatériel.Items.Add(id);
                cbSuppMater.Items.Add(id);
            }
        }
    }

    /// <summary>
    /// Fonction pour actualiser les ComboBox des techniciens.
    /// </summary>
    public void ActualiserTechnicien()
    {
        cbSuppTech.Items.Clear();

        List<string> matriculesTechnicien = BD.GetMatriculeTechnicien();
        if (matriculesTechnicien != null)
        {
            foreach (string matricule in matriculesTechnicien)
            {
                cbSuppTech.Items.Add(matricule);
            }
        }
    }

    /// <summary>
    /// Fonction pour actualiser les ComboBox des utilisateurs.
    /// </summary>
    public void ActualiserUtilisateur()
    {
        cbSuppU.Items.Clear();

        List<string> idUtilisateur = BD.GetIdUtilisateur();
        if (idUtilisateur != null)
        {
            foreach (string id in idUtilisateur)
            {
                cbSuppU.Items.Add(id);
            }
        }
    }
}

```

```

/// <summary>
/// Événement déclenché lors de l'enregistrement d'un utilisateur.
/// </summary>
private void btnEnregU_Click(object sender, EventArgs e)
{
    // Correction : Assurez-vous que le constructeur Utilisateur correspond bien aux
    paramètres fournis.
    Utilisateur unUtilisateur = new Utilisateur(tbMatriculeU.Text, tbNomU.Text,
    dateTimePicker1.Value, tbRegion.Text, 0, Convert.ToInt32(numericUpDown1.Value),
    Convert.ToInt32(numericUpDown2.Value), tbAvan.Text,
    Convert.ToInt32(numericUpDown3.Value));

    BD.AjouterUtilisateur(unUtilisateur); // Méthode renommée pour correspondre à celle de
    la classe BD.
    BD.AjouterUtilisateur(unUtilisateur); // Méthode renommée pour correspondre à celle de
    la classe BD.
    ActualiserUtilisateur();
}

/// <summary>
/// Événement de chargement de la fenêtre principale.
/// </summary>
private void Form1_Load(object sender, EventArgs e)
{
    ActualiserMateriel();
    ActualiserTechnicien();
    ActualiserUtilisateur();

    // Correction : Vérifiez que la méthode affichIncident renvoie bien une liste d'objets
    Incident.
    List<Intervention> interventions = BD.affichIntervention();
    if (interventions != null)
    {
        cbConsInci.Items.Clear();

        }
    }
}
}

```

Screen Interface :

Page connexion :

The screenshot shows a web application window with a title bar and four tabs: 'Connexion', 'Utilisateur', 'Technicien', and 'Responsable'. The 'Connexion' tab is selected. The main content area contains the following elements:

- A label 'Identifiant :' followed by a text input field.
- A label 'Mot de passe :' followed by a text input field.
- A button labeled 'Connexion' centered below the input fields.

Page Utilisateur :

The screenshot shows a web application window with a title bar and four tabs: 'Connexion', 'Utilisateur', 'Technicien', and 'Responsable'. The 'Utilisateur' tab is selected. The main content area is divided into two sections:

- Déclarer un incident :** This section contains a large text area for reporting an incident. Below it, there is a date selection field showing 'jeudi 21 novembre 2024' with a calendar icon. Further down are four input fields labeled 'Matricule Personnel', 'Matricule Technicien', 'Id Matériel', and 'Etat'. A 'Valider' button is positioned at the bottom of this section.
- Afficher les incidents :** This section features an 'Afficher' button and a list box labeled 'lbUtilisateur' which is currently empty.

Page Technicien :

Connexion Utilisateur Technicien Responsable

Matériel

Type :

Modele :

Date d'achat :

Etat :

Consulter infos matériels :

Supprimer un matériel :

Incident

Consulter des incidents :

Mise à jour

Etat

Page Responsable :

Connexion Utilisateur Technicien Responsable

Technicien - Ajouter

Matricule :

Mot de passe :

Region carriere :

Formation :

Compétence :

Niveau intervention :

Technicien - Modifier

Selectionner :

Matricule :

Mot de passe :

Region carriere :

Formation :

Compétence :

Niveau intervention :

Utilisateur - Ajouter

Matricule :

Mot de passe :

Région carriere :

Objectif :

Prime :

Avantage :

Budget :

Supprimer

Technicien :

Utilisateur :

Description de chaque clic :

ClasseTechnicien :

Le bouton confirmer dans la section matériel permet d'enregistrer un matériel dans la base de données.

Le bouton consulter permet d'afficher dans la listBox les informations d'un matériel. sélectionné dans la comboBox.

Le bouton supprimer permet lui de supprimer le matériel sélectionné dans une autre comboBox.

Le bouton consulter dans la section incident permet de voir les incidents déclarés par les utilisateurs.

Le bouton enregistrer permet à lui de prendre en charge un incident sélectionné dans une comboBox et mettre à jour son état

classeUtilisateur :

Ajouter Utilisateur : Permet d'ajouter un nouvel utilisateur en remplissant les champs pour le matricule, mot de passe, région carrière, objectif, prime, avantage et budget. Le bouton Enregistrer permet de sauvegarder ces informations.

Modifier Utilisateur: Modifier les informations d'un utilisateur existant. L'utilisateur peut être sélectionné, et des champs pour le matricule, mot de passe, région carrière, objectif, prime, avantage, et budget sont proposés. Le bouton Enregistrer permet de valider ces modifications.

Supprimer Utilisateur: Permet de supprimer un utilisateur de la base de données. L'utilisateur est sélectionné via un menu déroulant, et un bouton Supprimer permet d'effectuer cette action.

classeResponsable :

Ajouter Responsable : Permet d'ajouter un responsable au système en remplissant les informations suivantes :

- Matricule : Le matricule du responsable.
- Mot de passe : Le mot de passe de connexion pour le responsable.
- Région carrière : La région où le responsable est affecté.
- Formation : La formation ou le parcours du responsable.
- Compétence : Les compétences professionnelles du responsable.
- Niveau intervention : Le niveau d'intervention ou d'expertise du responsable.
- Le bouton Enregistrer permet de sauvegarder ces informations et d'ajouter le responsable à la base de données.

Modifier Responsable: Permet de modifier un responsable existant dans le système. Il faut d'abord sélectionner un responsable existant via un menu déroulant. Après sélection, les champs de modification sont activés pour la mise à jour des informations suivantes :

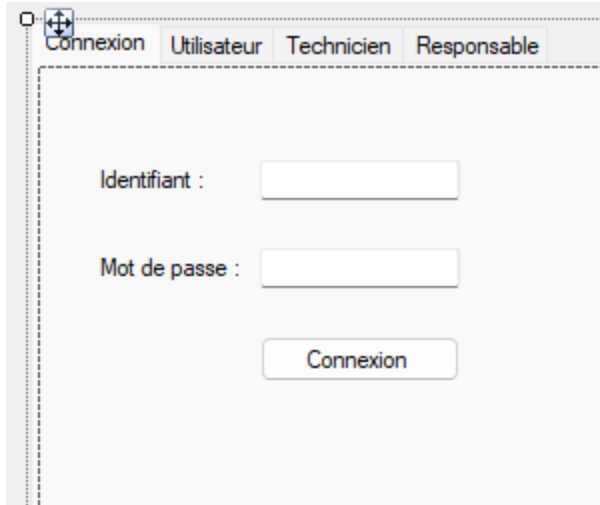
- Matricule
- Mot de passe
- Région carrière
- Formation
- Compétence
- Niveau d'intervention
- Le bouton Enregistrer permet de valider et d'enregistrer les modifications au responsable.

Supprimer Responsable: Permet de supprimer un responsable du système. Pour ce faire, le responsable à supprimer est sélectionné via un menu déroulant. Le bouton Supprimer valide la suppression du responsable sélectionné.

Manuel Utilisateur :

Se connecter :

Pour cela, il faudra renseigner son matricule ainsi que son mot de passe et confirmer sur le bouton "Connexion"



Connexion Utilisateur Technicien Responsable

Identifiant :

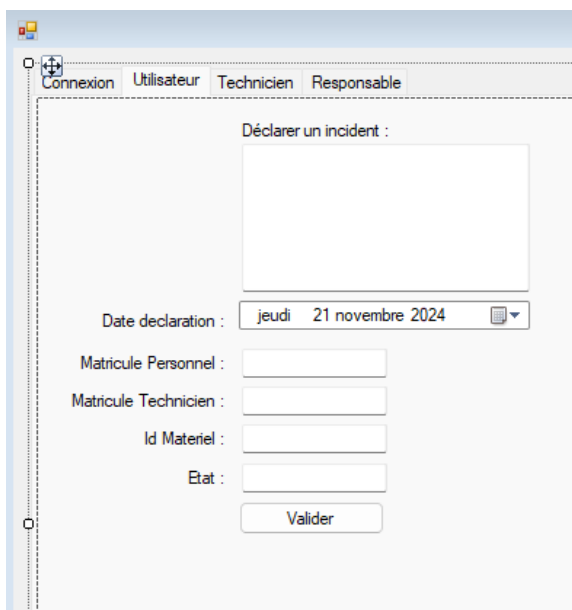
Mot de passe :

Connexion

Vous serez ensuite redirigé sur la page “Utilisateur”

Déclarer un incident :

Pour déclarer un incident, vous devrez remplir le formulaire avec les informations demandés (label à gauche pour vous aider quoi remplir ou) et cliquer sur le bouton “Valider”



Connexion Utilisateur Technicien Responsable

Déclarer un incident :

Date declaration : jeudi 21 novembre 2024

Matricule Personnel :

Matricule Technicien :

Id Matériel :

Etat :

Valider

Visualiser les incidents

Pour visualiser les incidents déclarés vous n'avez juste qu'à cliquer sur le bouton "Afficher" et toutes les informations ressortiront.

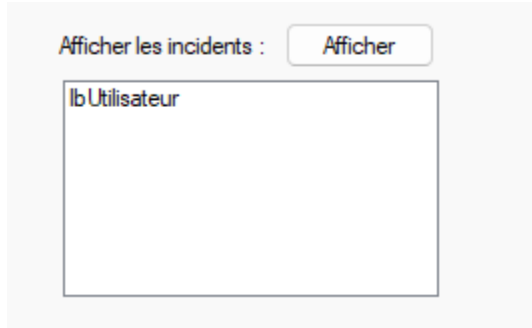
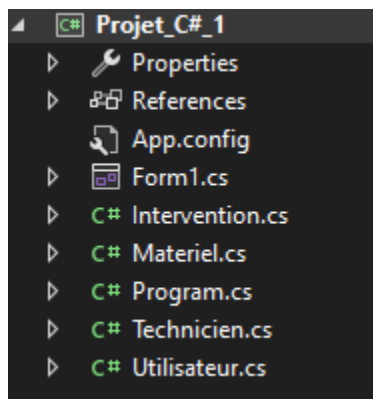


Schéma de Classes :

La documentation des classes (BD, Personnel, Utilisateur, Technicien, Incident et Matériel) est intégrée directement dans le code. Il est donc nécessaire de consulter le projet "laboGSB" pour y accéder.



Script sauvegarde BD + tuto généraOn set-up

Projet_C#_1Form1.csForm1.cs [Design]BD.csIntervention.csTechnicien.csMateriel.csUtilisateur.csForm1.Designer.csProgram.cs

ApplicationBuildBuild EventsDebugResourcesServicesSettingsReference PathsSigningSecurityPublishCode Analysis

Configuration: N/APlatform: N/A

Publish Location

Publishing Folder Location (ftp server or file path):D:\Projet_C#\Installation Folder URL (if different than above):[Learn how to test your application in Azure DevTest Labs](#)

Install Mode and Settings

☐ The application is available online only

☒ The application is available offline as well (launchable from Start menu)

Application Files...

Prerequisites...

Updates...

Options...

Publish Version

Major:1Minor:0Build:0Revision:0

☒ Automatically increment revision with each publish

Publish Wizard...

Publish Now

| Nom | Modifié le | Type | Taille |
|-------------------|------------------|-----------------------|--------|
| .vs | 07/11/2024 10:38 | Dossier de fichiers | |
| Application Files | 21/11/2024 10:28 | Dossier de fichiers | |
| Projet_C#_1 | 21/11/2024 10:28 | Dossier de fichiers | |
| Projet_C#_1 | 21/11/2024 10:28 | Application Manif... | 6 Ko |
| Projet_C#_1.sln | 07/11/2024 10:38 | Visual Studio Solu... | 2 Ko |
| setup | 21/11/2024 10:28 | Application | 538 Ko |

