



**BONVALLET**

**Lucas**

**BTS SIO SLAM – 2024**

# **RAPPORT DE PROJET**

**2024**

# Sommaire

- 01.** Contexte
- 02.** Expression des besoins
- 03.** objectifs
- 04.** Analyse fonctionnelle
- 05.** Manuel utilisateur
- 06.** Conclusion

# Contexte

## Le laboratoire GSB

### Le secteur d'activité :

L'industrie pharmaceutique se distingue par sa rentabilité et son dynamisme en matière de fusion-acquisition. Les récentes consolidations entre laboratoires ont donné naissance à d'énormes conglomérats, qui persistent dans des structures de travail héritées du passé. Des incidents récents liés à des médicaments ou molécules ayant entraîné des complications médicales ont suscité des critiques concernant une pratique spécifique des laboratoires : la visite médicale. Cette dernière est souvent pointée du doigt pour son rôle présumé dans les arrangements entre l'industrie et les praticiens, créant ainsi un terrain d'influence aux contours opaques.

### L'entreprise :

Le laboratoire Galaxy Swiss Bourdin (GSB) résulte de la fusion entre le géant américain Galaxy, spécialisé dans le domaine des maladies virales telles que le SIDA et les hépatites, et le conglomérat européen Swiss Bourdin, qui se consacre à des médicaments plus conventionnels. Ce dernier est lui-même le fruit de l'union de trois petits laboratoires. En 2009, ces deux géants pharmaceutiques ont fusionné pour former un leader dans le secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris, tandis que le siège social de la multinationale est implanté à Philadelphie, en Pennsylvanie, aux États-Unis. La France a été sélectionnée comme référence pour l'amélioration du suivi de l'activité de visite.

# Les besoins

## **Identification des Utilisateurs :**

Chaque utilisateur, en l'occurrence les médecins, devra disposer d'un compte individuel doté d'une authentification sécurisée afin d'accéder à l'application.

## **Gestion des Patients**

Les profils des patients doivent comprendre des informations détaillées telles que le nom, l'âge, le sexe, les antécédents médicaux et les allergies connues.

## **Les Ordonnances**

Les praticiens doivent disposer de la fonctionnalité permettant de générer, ajuster et révoquer des prescriptions. Chaque prescription devra contenir des informations essentielles telles que le nom du médicament, la posologie, la durée du traitement, ainsi que des directives spécifiques si nécessaires. De plus, il devra être possible d'exporter chaque prescription dans des formats tels que .txt ou .pdf, offrant ainsi une flexibilité dans la gestion et la transmission des informations médicales.

## **Base de Données des Médicaments**

L'application doit intégrer une base de données complète des médicaments, englobant les détails sur les contre-indications et les interactions médicamenteuses. Elle devra également notifier le médecin en cas d'interactions potentiellement dangereuses ou de contre-indications, en se basant sur le profil spécifique du patient. Cela garantira une prise de décision éclairée et sécurisée lors de la prescription de médicaments.

# Les objectifs

Ce projet a pour but de faciliter la gestion des dossiers des patients par les médecins du cabinet du laboratoire GSB, en intégrant leurs antécédents médicaux et allergies dans la base de données. De même, il vise à centraliser la base de données des médicaments en y incluant les contre-indications et les incompatibilités. Le logiciel GeStionB s'orientera également vers la simplification du processus de création d'ordonnances en permettant la génération d'un fichier PDF correspondant à chaque prescription. En cours de création de l'ordonnance, le logiciel devra effectuer une vérification des éventuelles incompatibilités entre les médicaments prescrits et les caractéristiques propres au patient.

Pour atteindre ces objectifs, le logiciel devra offrir une interface conviviale et intuitive permettant d'ajouter, de modifier et de consulter les informations relatives aux patients et aux médicaments. La sécurité et la confidentialité des données médicales constitueront des aspects prioritaires, avec la mise en place de mesures adéquates telles que l'authentification des utilisateurs et le chiffrement des données sensibles comme les mots de passe.

# Analyse fonctionnelle

## Ce logiciel présente plusieurs fonctionnalités

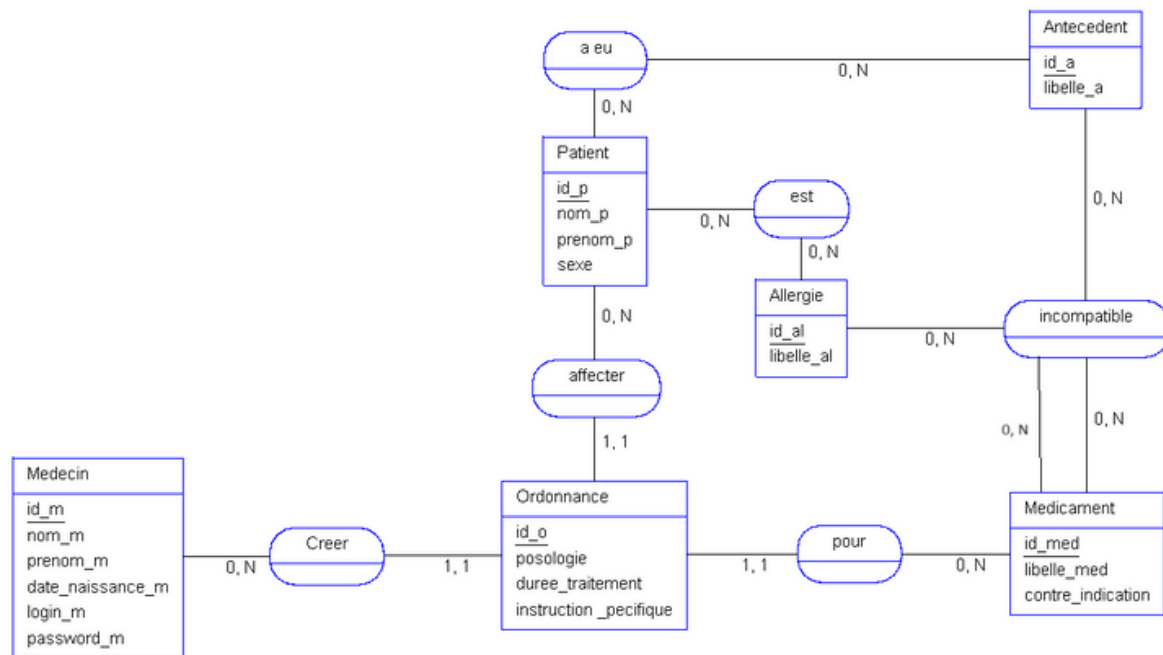
- Administration des utilisateurs : établissement de comptes et authentification sécurisée.
- Profils des patients : élaboration et mise à jour avec des informations détaillées.  
Gestion des prescriptions : création, modification, annulation avec inclusion de détails cruciaux.
- Profils des patients : élaboration et mise à jour avec des informations détaillées.  
Gestion des prescriptions : création, modification, annulation avec inclusion de détails cruciaux.
- Exportation des prescriptions : format .pdf.
- Base de données des médicaments : intégration exhaustive avec alertes pour interactions et contre-indications.
- Sécurité des données : utilisation de techniques de hachage et de chiffrement. Interface utilisateur conviviale : offre une navigation fluide et intuitive.

## Langages, technologies :

Git, GitHub Visual Studio C# .NET Windows Forms MAMP  
(Macintosh, Apache (serveur Web), MySQL (système de  
gestion de bases de données relationnelles), PHP) MySql  
Connector Bcrypt.Net-Next iText

# Réalisation du Projet

Avant de commencer la rédaction du code, j'ai élaboré des schémas pour planifier l'architecture de mon programme, la structure de ma base de données, ainsi que les différentes logiques à mettre en place. Je partage ici la première version du Modèle Conceptuel de Données que j'ai initialement utilisée au début du projet. Il est important de noter que cette version a subi des évolutions au fil du temps en raison de l'intégration progressive de nouvelles fonctionnalités.



Afin d'assurer un stockage sécurisé des mots de passe dans la base de données, j'ai utilisé la technique de "hashage" pour rendre ces derniers illisibles.

Dans la définition des exigences, il est spécifié que lors de la création d'une ordonnance, GeStionB doit empêcher la création de ladite ordonnance si le médicament prescrit est incompatible avec les antécédents médicaux ou les allergies du patient.

## Connexion entre le logiciel et la base de données

Pour établir la connexion avec ma base de données, j'ai opté pour l'utilisation du package NuGet MySQL Connector. Ce dernier offre une interface entre les applications et le serveur de base de données MySQL, permettant ainsi l'exécution de requêtes, l'insertion, la mise à jour, la suppression de données, ainsi que d'autres opérations de gestion de bases de données.

Voici la démarche pour établir la connexion avec la base de données :

```
<configuration>  
  <connectionStrings>  
    <add connectionString="Server=localhost;Database=gsb;UserID=root;Password=AjgZo@lp" name="localhost" providerName="MySql.Data.MySqlClient" />  
  </connectionStrings>  
</configuration>
```

Ce code établit la chaîne de connexion qui sera utilisée. Cette chaîne de connexion englobe le nom DNS du serveur, la base de données ciblée, le nom d'utilisateur et son mot de passe pour la connexion, ainsi que le nom de la connexion.



```
private string connectionString = ConfigurationManager.ConnectionStrings["localhost"].ConnectionString;
```

```
using (MySqlConnection conn = new MySqlConnection(connectionString))  
{  
    conn.Open();  
    string query = "Requete SQL";  
    using (MySqlCommand command = new MySqlCommand(query, conn))  
    {
```

Les extraits de code précédents sont inclus dans une classe dédiée à la gestion de la connexion avec la base de données. Dans le premier code, la chaîne de connexion est appelée et stockée dans la variable "connectionString". Le deuxième code crée une connexion en utilisant la classe "MySqlConnection" et en passant la chaîne de connexion en paramètre. Ensuite, il définit une requête SQL et l'utilise avec la classe "MySqlCommand" du package MySQL Connector.

## Les étapes CRUD

La réalisation d'opérations de type CRUD implique la capacité de créer, lire, mettre à jour et supprimer des éléments dans la base de données. Dans mon application, les opérations CRUD seront indispensables, par exemple, pour :

- Créer et lire les données d'un patient.
- Modifier les informations d'un médicament, y compris les incompatibilités.
- Supprimer une ordonnance.
- Récupérer les données nécessaires pour générer un fichier PDF correspondant.

## **Création des différentes vues**

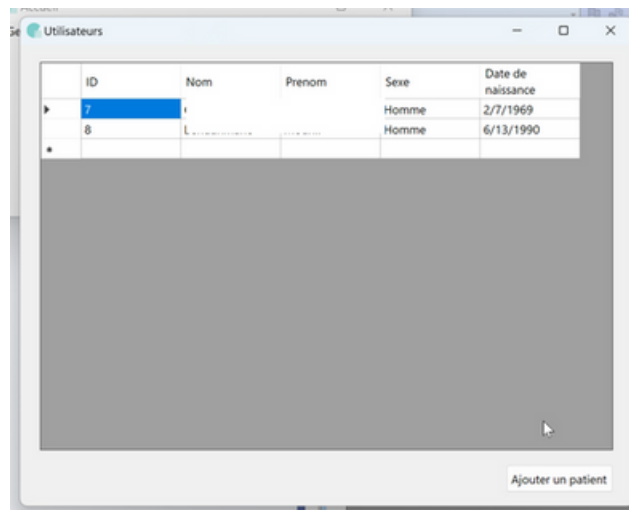
La création de vues avec Windows Forms est une tâche intuitive grâce aux outils mis à disposition par Visual Studio, qui permettent de concevoir une interface en utilisant simplement le glisser-déposer. Dans mon projet, j'ai utilisé plusieurs composants, dont la TextBox, qui permet de créer des champs d'entrée de texte.

Le paramètre le plus important à modifier est le nom du composant "(Name)", car il correspond au début du nom de la méthode qui gère l'événement souhaité, par exemple "Click" pour un bouton.

Pour établir des liens entre différentes pages, la méthode employée consiste à créer une nouvelle instance de la fenêtre envisagée, puis à utiliser la méthode ".Show()".

## **Connexion entre les pages**

Je vise à implémenter dans mon logiciel est la capacité d'échanger des données entre différents composants. Par exemple, sur la page de gestion des patients, lorsque l'on clique sur un élément du tableau pour ouvrir les détails du patient, il est crucial que les données soient transmises d'une fenêtre à l'autre de manière efficace.



Pour gérer cela, ma méthode consiste à transmettre les données en tant que paramètres au composant cible (PatientsDetails). Voici comment cela fonctionne concrètement :

```
// Au clic sur une cellule de la grille (DataGridView) ->
1 reference
private void PatientGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    // Vérifie si l'index de ligne est valide (supérieur ou égal à zéro)
    if (e.RowIndex >= 0)
    {
        // Récupère la ligne sélectionnée à partir de l'index de ligne
        DataGridViewRow selectedRow = this.PatientGridView.Rows[e.RowIndex];

        // Récupère les valeurs des cellules de la ligne sélectionnée
        int id = Convert.ToInt32(selectedRow.Cells["ID"].Value);
        string nom = selectedRow.Cells["nom"].Value.ToString();
        string prenom = selectedRow.Cells["prenom"].Value.ToString();
        string sexe = selectedRow.Cells["sexe"].Value.ToString();
        string birthday = selectedRow.Cells["Date de naissance"].Value.ToString();

        // Crée une instance de PatientsDetails avec les valeurs récupérées
        PatientsDetails patientDetail = new PatientsDetails(id, nom, prenom, sexe, birthday);

        // Affiche la fenêtre PatientsDetails
        patientDetail.Show();
    }
}
```

```

public int Id { get; set; }
// Propriété publique Id de type entier, utilisée pour stocker l'ID du patient

1 reference
public PatientsDetails(int id, string nom, string prenom, string sexe, string birthday)
{
    InitializeComponent();
    this.Id = id;
    // Affecte la valeur de l'ID passé en paramètre à la propriété Id de l'objet PatientsDetails
    this.Box_change_nom.Text = nom;
    // Affecte la valeur du nom passé en paramètre à la propriété Text de la boîte de texte Box_change_nom
    this.Box_change_prenom.Text = prenom;
    // Affecte la valeur du prénom passé en paramètre à la propriété Text de la boîte de texte Box_change_prenom
    this.combo_change_sexe.Text = sexe;
    // Affecte la valeur du sexe passé en paramètre à la propriété Text de la liste déroulante combo_change_sexe
    this.date_PatientDetails.Text = birthday;
    // Affecte la valeur de la date de naissance passée en paramètre à la propriété Text de date_PatientDetails
}

```

## listes déroulantes à partir de la base de données.

Les listes déroulantes constituent d'excellents moyens pour prévenir les erreurs de correspondance entre le texte saisi dans les différents formulaires et la base de données. Par exemple, lorsqu'on se trouve sur la page de gestion des incompatibilités,

La sélection s'effectue via une liste déroulante dans le but de prévenir les erreurs de frappe et, par conséquent, d'éviter les erreurs d'attribution d'incompatibilité. L'attribution d'incompatibilité repose sur les noms des allergies, antécédents et médicaments, et cela sera expliqué plus en détail par la suite.

```

1 reference
int Id { get; set; }
IncompatibiliteDataAccess dataAccessIncompatibilite = new IncompatibiliteDataAccess();

1 reference
public ManageIncompatibilite(int id)
{
    InitializeComponent();
    this.Id = id;
    // Affecte la valeur de l'ID passé en paramètre à la propriété Id de l'objet ManageIncompatibilite
    this.Activated += ManageMedicament_Activated;
    // Lorsque la fenetre est active -> executer ManageMedicament_Activated
}

1 reference
private void ManageMedicament_Activated(object sender, EventArgs e)
{
    //appelle les methodes pour remplir les combobox
    FillComboBoxAntecedents();
    FillComboBoxAllergies();
    FillComboBoxMedicaments();
}

1 reference
public void FillComboBoxMedicaments()
{
    MedicamentsDataAccess dataAccess = new MedicamentsDataAccess();
    // Instance de la classe MedicamentsDataAccess utilisée pour accéder aux données de médicaments
    dataAccess.FillComboBox(combo_Medicaments);
    // Remplit la liste déroulante combo_Medicaments avec les données de médicaments via la méthode
    // FillComboBox de l'objet dataAccess
    this.combo_Medicaments.Text = dataAccessIncompatibilite.FillDefaultValueComboBoxMedicaments(Id);
    // Affecte la valeur par défaut de la liste déroulante combo_Medicaments en fonction
    // de l'ID de gestion de l'incompatibilité
}
1 reference

```

```

public string FillDefaultValueComboBoxMedicaments(int id_med)
{
    string medicament = "";
    // Variable utilisée pour stocker le nom du médicament par défaut
    using (MySQLConnection conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        // Ouverture de la connexion à la base de données
        string query = "SELECT libelle_med FROM medicament WHERE medicament.id_med = " +
            "(SELECT id_med_Medicament FROM incompatible WHERE incompatible.id_med = @id_med);";
        // Requête SQL pour sélectionner le nom du médicament en fonction de l'ID fourni
        using (MySQLCommand command = new MySqlCommand(query, conn))
        {
            command.Parameters.AddWithValue("@id_med", id_med);
            // Ajout du paramètre @id_med à la requête SQL pour éviter les injections SQL
            using (MySQLDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    medicament = reader.GetString(0);
                    // Lecture du résultat de la requête et affectation du nom du médicament à la variable medicament
                }
            }
        }
        conn.Close();
        // Fermeture de la connexion à la base de données
    }
    return medicament;
    // Retourne le nom du médicament par défaut
}

```

Dans le code ci-dessus, la requête pourrait se lire comme suit : "Sélectionner le libellé du médicament à partir de la table 'medicament' où l'ID du médicament est égal à l'ID du médicament récupéré à partir de la table 'incompatible' où l'ID du médicament est égal à la valeur du paramètre 'id\_med'".

## Authentification

Lorsque l'utilisateur ne possède pas de compte, un bouton pour créer un compte est disponible sur la page de connexion. Ce bouton ouvre initialement une première fenêtre où le mot de passe administrateur est requis pour restreindre la création de comptes aux personnes autorisées uniquement. Ensuite, la page de création de compte s'ouvre, permettant à l'administrateur de renseigner les informations du nouveau médecin.

Lorsque le formulaire est validé, voici ce qui se produit :

Actions :

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace GeStionB.Medecin
8  {
9      internal class Bcrypt
10     {
11     public string Encryption (string pswd)
12     {
13         string hash = BCrypt.Net.BCrypt.EnhancedHashPassword(pswd,13);
14         return hash;
15     }
16     public bool Description(string pswd, string hash)
17     {
18         bool result = BCrypt.Net.BCrypt.EnhancedVerify(pswd, hash);
19         return result;
20     }
21     }
22 }

```

---

En revanche, lorsque l'utilisateur a déjà un compte, il lui suffit de saisir son identifiant et son mot de passe. Voici ce qui se passe après la validation du formulaire :

```

MedecinDataAccess dataAccess = new MedecinDataAccess();
string hash = dataAccess.GetHashForAuthentication("administrateur");

if (hash != null)
{
    Bcrypt bcrypt = new Bcrypt();
    bool result = bcrypt.Description(this.Box_mdpAdmin.Text, hash);
    if (result)
    {
        AddMedecin addMedecin = new AddMedecin();
        this.Hide();
        addMedecin.Show();
    }
    else
    {
        MessageBox.Show("Mauvais identifiant/mot de passe");
    }
}
else
{
    await Task.Delay(800);
    MessageBox.Show("Mauvais identifiant/mot de passe");
}
}
}
}
}

```

---

Il est crucial de mentionner qu'avant le clic, j'ai introduit un délai de réponse de 800 ms si aucun hash n'est renvoyé par la base de données (BDD). En effet, sans ce délai, un utilisateur malveillant pourrait déduire quels sont les identifiants valides ou non. Actuellement, si un utilisateur tente de se connecter avec un nom d'utilisateur invalide, la réponse sera instantanée. En revanche, si un nom d'utilisateur valide est utilisé, la réponse ne sera pas immédiate, car le logiciel s'efforcera de comparer le mot de passe de la BDD avec celui du formulaire. Cette comparaison prendra plus de temps, rendant ainsi plus difficile la divulgation d'un identifiant valide.

## Création d'une ordonnance

Afin de respecter les spécifications du cahier des charges, il est impératif que lors de la création d'une ordonnance, GeStionB empêche la création de celle-ci en cas de détection d'une incompatibilité entre le patient et le médicament. Voici le code qui correspond à la comparaison entre les allergies et antécédents du patient, ainsi que les incompatibilités du médicament :

```
public void CreateOrdonnance(string posologie, int duree, string instructions, string nom_m, string nom_prenom_p, string libelle_med)
{
    List<string> listeAllergies = GetAllergieListFromDB(nom_prenom_p);
    List<string> listeAntecedents = GetAntecedentListFromDB(nom_prenom_p);
    List<Incompatibilite> listeIncompatibilites = GetIncompatibiliteFromDB(libelle_med);
    bool isIncompatible = false;
    foreach (string allergie in listeAllergies)
    {
        if (listeIncompatibilites.Any(incompatibilite => incompatibilite.Id_al == allergie || incompatibilite.Id_a == allergie))
        {
            isIncompatible = true;
            MessageBox.Show("Le médicament est incompatible avec une allergie du patient ");
            return;
        }
    }
    foreach (string antecedent in listeAntecedents)
    {
        if (listeIncompatibilites.Any(incompatibilite => incompatibilite.Id_al == antecedent || incompatibilite.Id_a == antecedent))
        {
            isIncompatible = true;
            MessageBox.Show("Le médicament est incompatible avec un antécédent du patient ");
            return;
        }
    }
}
```

# Génération d'une ordonnance en PDF

Afin de générer un PDF, j'ai utilisé la dépendance iText. La dépendance iText fonctionne de la manière suivante

```
private void btnCreatePDF_Click(object sender, EventArgs e)
{
    using (FolderBrowserDialog folderBrowserDialog = new FolderBrowserDialog())
    {
        //ouvre l'explorateur de fichier afin de choisir le dossier de destination
        if (folderBrowserDialog.ShowDialog() == DialogResult.OK)
        {
            string selectedFolder = folderBrowserDialog.SelectedPath;
            //initialise la variable selectedFolder avec la valeur du dossier sélectionné
            PdfCreator pdfCreator = new PdfCreator();
            //créer une nouvelle instance de PdfCreator
            pdfCreator.CreatePDF(selectedFolder, this.boxId.Text, this.boxMedecin.Text, this.boxDate.Text, this.comboPatient.Text, this.comboMedicament.Text);
            //Utilise la méthode CreatePDF de pdfCreator et le passe les dossier de destinations et les informations
            //nécessaire à l'ordonnance
        }
    }
}
//evenement qui correspond au click du bouton de création du PDF
```

```
public void CreatePDF(string filePath, string id_p, string nom_m, string date_p, string nom_p, string libelle_med, string posologie, string duree, string instructions) {
    doc.Open();

    BaseColor noir = new BaseColor(0, 0, 0);

    Font policeTitre = new Font(iTextSharp.text.Font.FontFamily.HELVETICA, 18, iTextSharp.text.Font.BOLD, noir);
    Font policeParagraphe = new Font(iTextSharp.text.Font.FontFamily.HELVETICA, 14, iTextSharp.text.Font.NORMAL, noir);

    Paragraph infoMedecin = new Paragraph("DR. " + nom_m + " \n21 rue des Lisettes, 92228 Bagneux", policeTitre);
    infoMedecin.Alignment = Element.ALIGN_LEFT;
    doc.Add(infoMedecin);

    Paragraph separateur = new Paragraph("=====");
    separateur.Alignment = Element.ALIGN_CENTER;
    doc.Add(separateur);

    Paragraph date = new Paragraph("Fait le " + date_p, policeParagraphe);
    date.Alignment = Element.ALIGN_RIGHT;
    doc.Add(date);
    doc.Add(separateur);

    Paragraph Patient = new Paragraph("Ordonnance médicale pour : Mr/Mme " + nom_p);
    Patient.Alignment = Element.ALIGN_LEFT;
    doc.Add(Patient);
    doc.Add(separateur);

    Paragraph Contenu = new Paragraph("Médicament prescrit: " + libelle_med + " \n" + posologie + " pendant " + duree + " jours");
    Contenu.Alignment = Element.ALIGN_CENTER;
    doc.Add(Contenu);
    doc.Add(separateur);

    Paragraph Instructions = new Paragraph(instructions);
    Instructions.Alignment = Element.ALIGN_CENTER;
    doc.Add(Instructions);
    doc.Add(separateur);

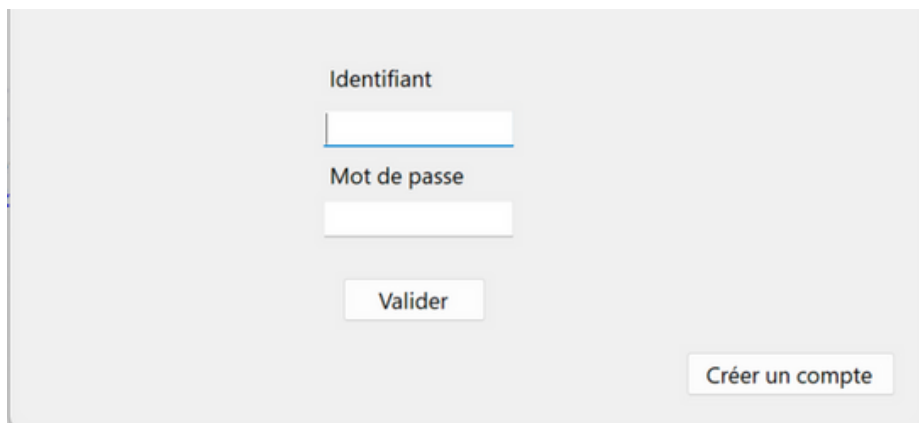
    doc.Close();
    //ferme le document
    Process.Start(new ProcessStartInfo
    {
        FileName = OutFile,
        UseShellExecute = true
    });
}
```



# Manuel utilisateur

Pour tester l'application, il est indispensable d'avoir le service MySQL installé. Ensuite, l'exécution du script SQL présent dans le répertoire GitHub associé à la base de données est nécessaire. Une fois la base de données importée, ouvrez le projet correspondant dans Visual Studio à partir du répertoire GitHub.

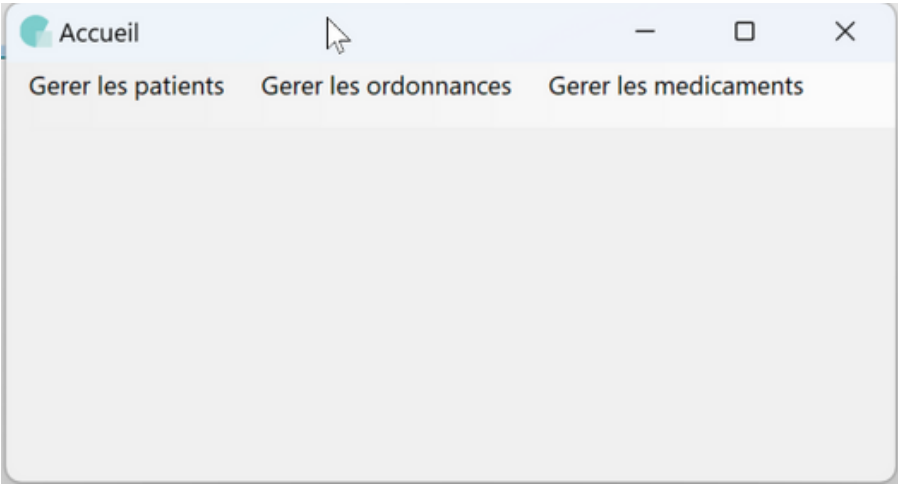
Une fois le projet ouvert et le service MySQL actif, vous pouvez lancer l'application en cliquant sur le bouton suivant :

A screenshot of a web-based login form. It contains two input fields: 'Identifiant' and 'Mot de passe'. Below these fields is a 'Valider' button. In the bottom right corner, there is a link that says 'Créer un compte'.

Identifiant : testeur

Mot de passe : test

Une fois la connexion établie avec succès, la page d'accueil s'affiche comme suit. Vous y trouverez les fonctionnalités principales de l'application, notamment la gestion des patients, des ordonnances et des médicaments.



Lorsque vous cliquez sur "Gérer les patients", cette fenêtre s'ouvre et affiche la liste complète des patients ainsi que leurs informations associées.

Pour ajouter un patient, il vous suffit de cliquer sur "Ajouter un patient", de saisir les informations nécessaires, puis de valider.

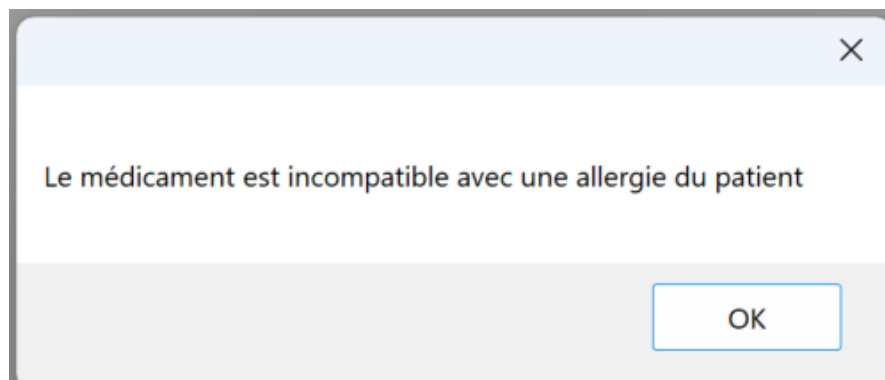
En cas d'erreur lors de la création, vous avez la possibilité de revenir à la page précédente et de cliquer sur le patient à modifier. C'est également de cette manière que l'on renseigne les allergies et les antécédents.

Si aucune allergie correspondante n'est trouvée, vous pouvez en créer une nouvelle en utilisant le bouton "Créer une nouvelle allergie".

Il vous suffit simplement de saisir l'intitulé de la future allergie et de valider.

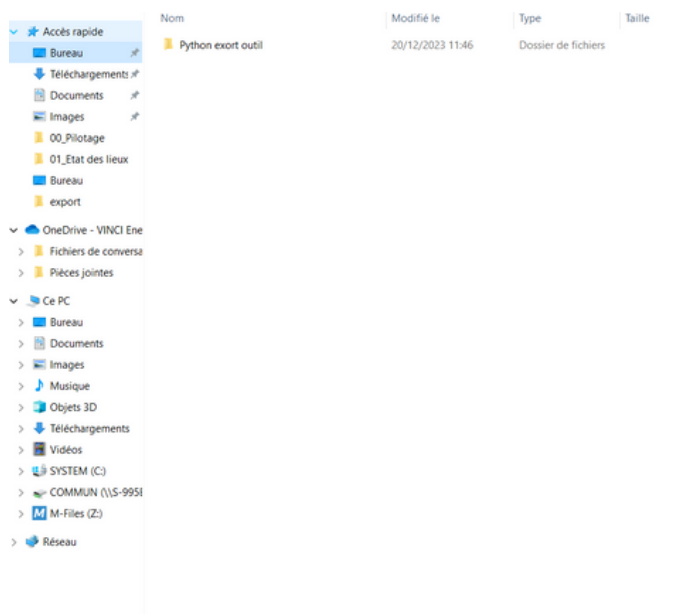
La gestion des médicaments fonctionne de manière similaire à la gestion des patients. La gestion des incompatibilités se fait sur la page des détails du médicament que l'on souhaite gérer.

**La création d'une ordonnance se fait de la même manière que la création d'un patient. Veuillez noter que dans le champ "Durée", un nombre est attendu. Si le médicament ne peut pas être prescrit au patient en raison d'incompatibilités, un message explicatif s'affichera :**



**Pour générer le PDF d'une ordonnance, il faut cliquer sur l'ordonnance souhaitée dans la liste des ordonnances, puis cliquer sur le bouton "Générer le PDF".**

**Une fenêtre de l'explorateur de fichiers va alors s'ouvrir, vous permettant de sélectionner le dossier de destination.**



**Une fois sélectionné, le fichier PDF va se générer et le lecteur PDF par défaut de votre ordinateur va ouvrir le fichier généré.**

# Conclusion

En conclusion, ce projet a représenté une expérience extrêmement enrichissante et stimulante, me permettant de découvrir de nouvelles technologies telles que C#, .NET, Windows Forms, Visual Studio, iText, ainsi que de consolider mes compétences avec Bcrypt et MySQL. La nécessité de rechercher des réponses à mes questions pour résoudre les problèmes rencontrés a renforcé ma capacité à résoudre des défis techniques.

Je tiens à souligner le soutien précieux que j'ai reçu de la part des membres de ma classe, leur contribution a été une source d'inspiration et d'entraide tout au long du projet. Les contraintes liées à un arrêt maladie ont impacté la gestion du temps et m'ont mis dans une situation délicate pour respecter les délais de rendu.

Néanmoins, je reste fier du résultat obtenu, de découvertes, d'organisation, de réflexion et de documentation. Cette expérience m'a motivé à envisager des versions futures du projet, visant une clarté accrue dans le code, une meilleure sécurité et l'intégration de nouvelles fonctionnalités. Mon objectif ultime est de rendre cette application accessible à tous, en améliorant son ergonomie et en offrant une expérience utilisateur optimale.

En dépit des défis rencontrés, je demeure fier des résultats obtenus au cours de ce projet. Cette aventure a été riche en découvertes, en apprentissages organisationnels, en réflexions approfondies et en documentation. Les enseignements tirés de cette expérience m'ont stimulé à envisager des versions futures du projet, avec pour ambition une clarté accrue dans le code, une sécurité renforcée, et l'intégration de nouvelles fonctionnalités.

Cette expérience a été un catalyseur pour mes ambitions, m'incitant à poursuivre le développement de l'application. Mon objectif ultime demeure de rendre cette application accessible à tous, en mettant l'accent sur l'amélioration de son ergonomie et en offrant une expérience utilisateur optimale. Cette conclusion représente une étape clé dans mon parcours, et j'envisage avec enthousiasme les opportunités d'amélioration et d'innovation qui se présenteront dans les versions futures du projet.