

Corentin **SANJUAN**
BTS SIO SLAM - 2024

RAPPORT DE PROJET



GeStionB

Sommaire

01



Contexte

02



Expression du
ou des besoins

03



Les objectifs

04



Analyse
fonctionnelle

05



Manuel
utilisateur

06



Conclusion

Contexte

Description du laboratoire GSB

Le secteur d'activité :

L'industrie pharmaceutique est un secteur très lucratif dans lequel le mouvement de fusion acquisition est très fort. Les regroupements de laboratoires ces dernières années ont donné naissance à des entités gigantesques au sein desquelles le travail est longtemps resté organisé selon les anciennes structures. Des déboires divers récents autour de médicaments ou molécules ayant entraîné des complications médicales ont fait s'élever des voix contre une partie de l'activité des laboratoires : la visite médicale, réputée être le lieu d'arrangements entre l'industrie et les praticiens, et tout du moins un terrain d'influence opaque.

L'entreprise :

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui même déjà union de trois petits laboratoires . En 2009, les deux géants pharmaceutiques ont uni leurs forces pour créer un leader de ce secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris. Le siège social de la multinationale est situé à Philadelphie, Pennsylvanie, aux Etats-Unis. La France a été choisie comme témoin pour l'amélioration du suivi de l'activité de visite.

Réorganisation :

Une conséquence de cette fusion, est la recherche d'une optimisation de l'activité du groupe ainsi constitué en réalisant des économies d'échelle dans la production et la distribution des médicaments (en passant par une nécessaire restructuration et vague de licenciement), tout en prenant le meilleur des deux laboratoires sur les produits concurrents. L'entreprise compte 480 visiteurs médicaux en France métropolitaine (Corse comprise), et 60 dans les départements et territoires d'outre-mer. Les territoires sont répartis en 6 secteurs géographiques (Paris-Centre, Sud, Nord, Ouest, Est, DTOM Caraïbes-Amériques, DTOM Asie-Afrique).

Expression des besoins

- **Identification des Utilisateurs :**

Chaque utilisateur (médecin) doit avoir un compte unique avec une authentification sécurisée pour accéder à l'application.

- **Gestion des Profils des Patients :**

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

- **Création et Gestion des Ordonnances :**

Les médecins doivent pouvoir créer, modifier et annuler des ordonnances.

Chaque ordonnance doit inclure le nom du médicament, la posologie, la durée du traitement, et les instructions spéciales si nécessaires

Chaque ordonnance doit pouvoir être exportée au format .txt ou .pdf

- **Base de Données des Médicaments :**

L'application doit avoir une base de données exhaustive des médicaments, y compris les informations sur les contre-indications et les interactions médicamenteuses.

L'application doit avertir le médecin en cas de potentielles interactions dangereuses ou contre-indications basées sur le profil du patient.

Les objectifs



Ce projet vise à permettre aux médecins du cabinet du laboratoire GSB de gérer la base de données des patients, en y incluant leurs antécédents médicaux et leurs allergies, ainsi que la base de données des médicaments, avec leurs contre-indications et incompatibilités. Le logiciel GeStionB aura également pour objectif de faciliter la création d'ordonnances en intégrant la possibilité de générer un fichier PDF correspondant à l'ordonnance. Lors de la création de l'ordonnance, le logiciel devra être capable de vérifier les éventuelles incompatibilités entre les médicaments prescrits et les caractéristiques du patient.

Pour réaliser ces fonctionnalités, le logiciel devra fournir une interface conviviale et intuitive permettant d'ajouter, modifier et consulter les informations des patients et des médicaments. Il sera important d'assurer la sécurité et la confidentialité des données médicales, en mettant en place des mesures de protection appropriées, telles que l'authentification des utilisateurs et le chiffrement des données sensibles tels que les mots de passe.

Analyse fonctionnelle

Ce logiciel présente plusieurs fonctionnalités

- Gestion des utilisateurs : création de comptes et authentification sécurisée.
- Profils de patients : création et mise à jour avec informations détaillées.
- Gestion des ordonnances : création, modification, annulation avec détails essentiels.
- Exportation des ordonnances : format .pdf .
- Base de données des médicaments : intégration exhaustive avec avertissements d'interactions et contre-indications.
- Sécurité des données : techniques de hachage et chiffrement.
- Interface utilisateur conviviale : navigation fluide et intuitive.

Langages, technologies et outils utilisés :

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

Lien du répertoire github :

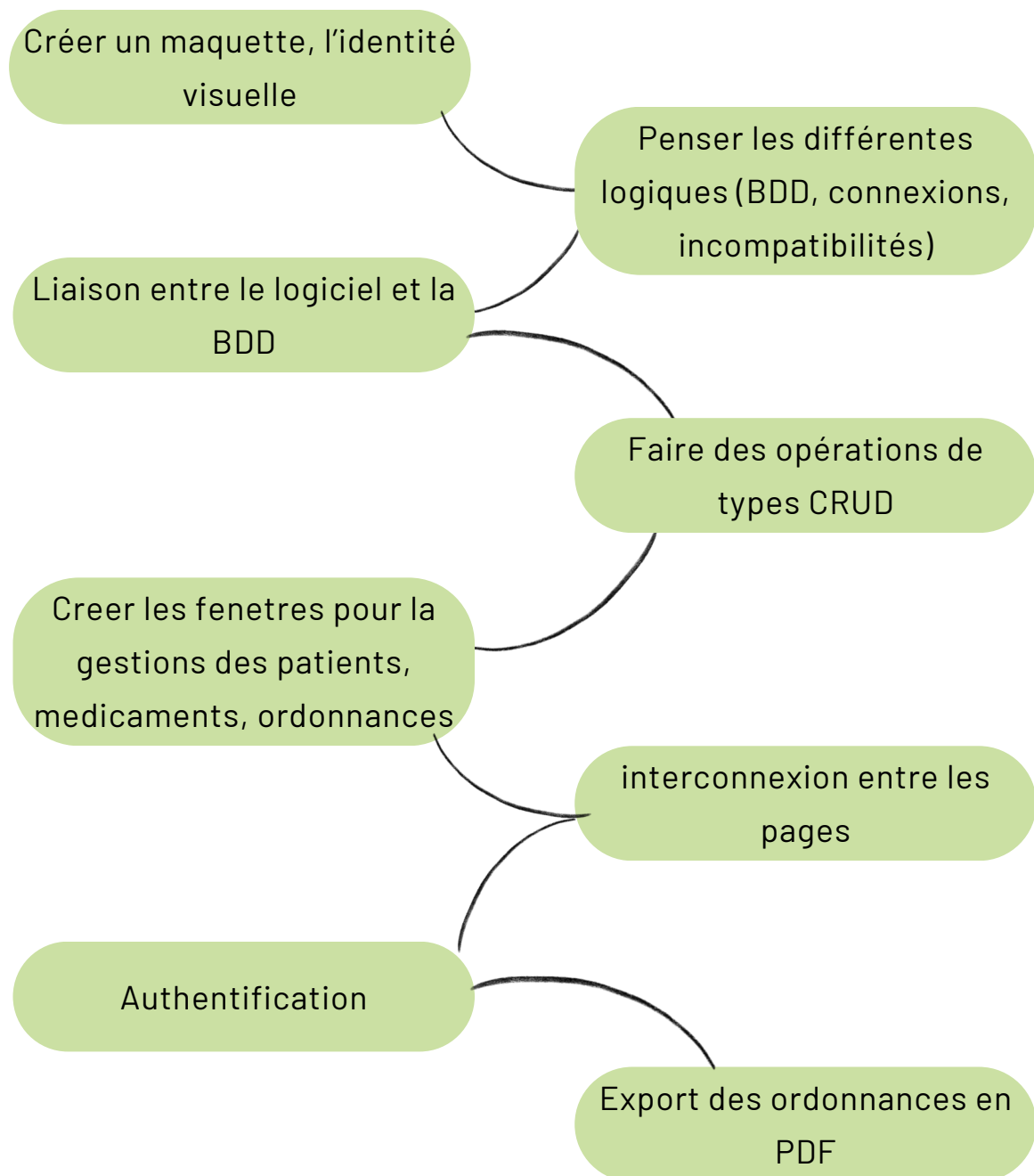
- Projet : <https://github.com/Crtnsj/GeStionB>
- Base de données : https://github.com/Crtnsj/BDD_GeStionB

Choix des technologies :

- C# : un langage de programmation polyvalent et orienté objet développé par Microsoft. Il est largement utilisé pour le développement d'applications Windows, de services web et d'applications mobiles.

- .NET : un framework logiciel développé par Microsoft. Il fournit un environnement d'exécution et une bibliothèque de classes pour le développement d'applications. .NET prend en charge plusieurs langages de programmation, dont C#, et offre des fonctionnalités avancées pour le développement orienté objet.
- Windows Forms : une technologie de développement d'interfaces utilisateur (UI) pour les applications de bureau basées sur Windows. Elle permet de créer des interfaces graphiques interactives en utilisant une variété de contrôles prêts à l'emploi, tels que les boutons, les fenêtres modales, les grilles de données, etc.
- MAMP (Macintosh, Apache, MySQL, PHP) : une suite de logiciels utilisée pour créer un environnement de développement web sur un système d'exploitation Macintosh. Elle comprend un serveur Apache pour le traitement des requêtes HTTP, un système de gestion de bases de données MySQL pour le stockage des données, et le langage de programmation côté serveur PHP pour le développement d'applications web dynamiques.
- MySql Connector : une bibliothèque spécifique à C# qui permet de se connecter à une base de données MySQL depuis une application C#. Elle fournit des fonctionnalités pour exécuter des requêtes SQL, récupérer des données et effectuer des opérations de mise à jour sur la base de données.
- Bcrypt.Net-Next : une bibliothèque utilisée pour le hachage sécurisé des mots de passe. Elle implémente l'algorithme Bcrypt, qui est considéré comme sûr et résistant aux attaques de force brute.
- Git, GitHub : Git est un système de contrôle de version distribué largement utilisé pour le suivi des modifications du code source d'un projet. GitHub est une plateforme d'hébergement de code source basée sur Git, qui facilite la collaboration entre les développeurs et offre des fonctionnalités telles que le suivi des problèmes, les demandes de tirage (pull requests) et le déploiement continu.
- iText : une bibliothèque Java (également disponible pour d'autres langages, dont C#) utilisée pour la génération et la manipulation de documents PDF. Elle permet de créer des fichiers PDF, d'ajouter du contenu, de générer des rapports et d'effectuer d'autres opérations liées aux documents.
- Visual Studio : un environnement de développement intégré (IDE) fourni par Microsoft. Il fournit des outils puissants pour la création, le débogage et le déploiement d'applications basées sur .NET et C#. Visual Studio offre une interface conviviale, des fonctionnalités de productivité avancées et une intégration étroite avec les autres outils de développement Microsoft.

Les grandes étapes à réaliser :



Réalisation des grandes étapes

Création de la maquettes et de l'identité visuelle :

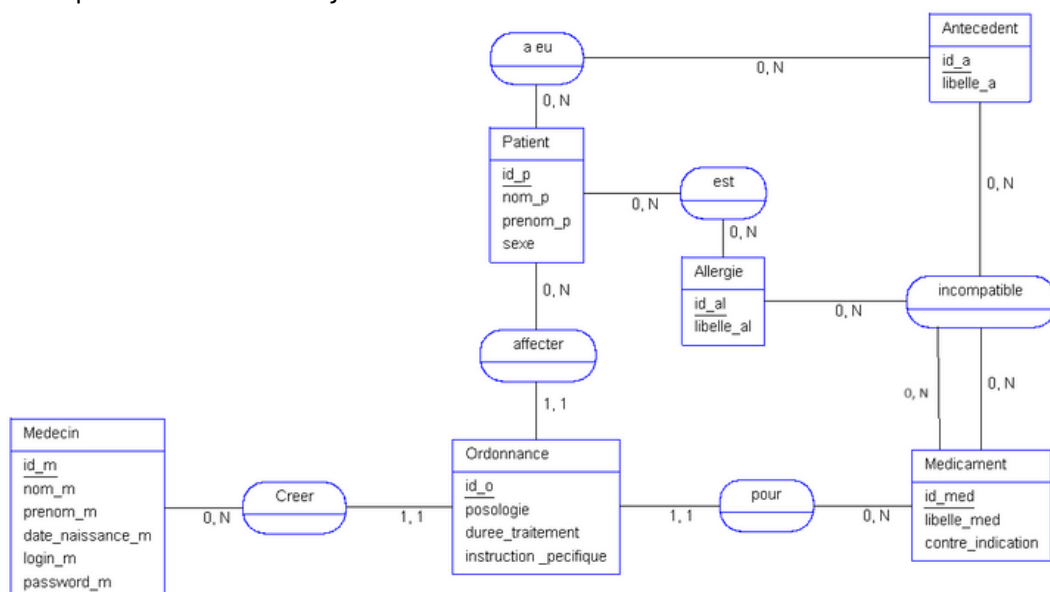
Afin de réaliser ma maquette de l'application et de créer une identité visuelle, j'ai utilisé Excalidraw. Excalidraw est un outil en ligne de dessin et de création de diagrammes. Il permet aux utilisateurs de créer facilement des illustrations simples, des schémas, des diagrammes de flux, des wireframes et d'autres types de dessins collaboratifs grâce à son interface intuitive et conviviale.

Excalidraw se distingue par sa simplicité et sa facilité d'utilisation. Il propose une variété de formes de base, comme des rectangles, des cercles, des flèches, des lignes, etc., que les utilisateurs peuvent dessiner et organiser sur une toile virtuelle. Il offre également la possibilité d'ajouter du texte, de modifier les couleurs, de redimensionner les objets et d'effectuer d'autres opérations de base pour personnaliser les dessins.

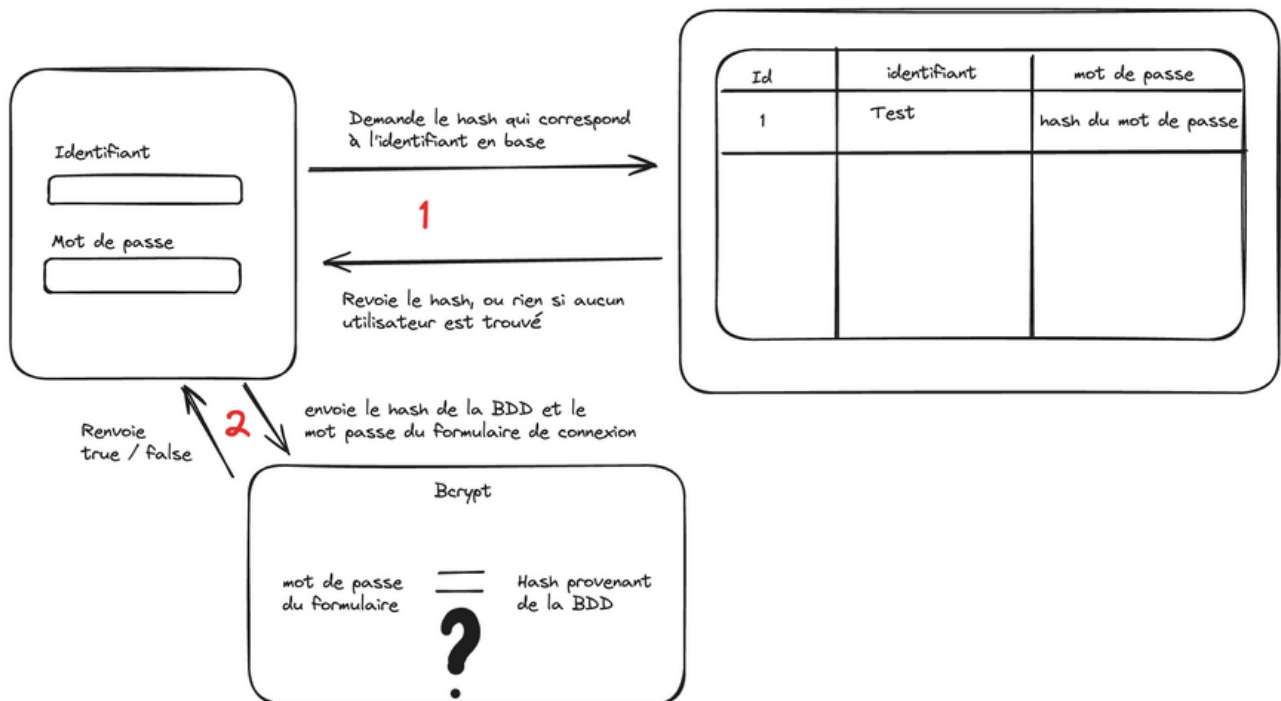
Je vous invite à consulter la première annexe pour comprendre l'architecture des pages.

Penser les différentes logiques (BDD, connexions, incompatibilités)

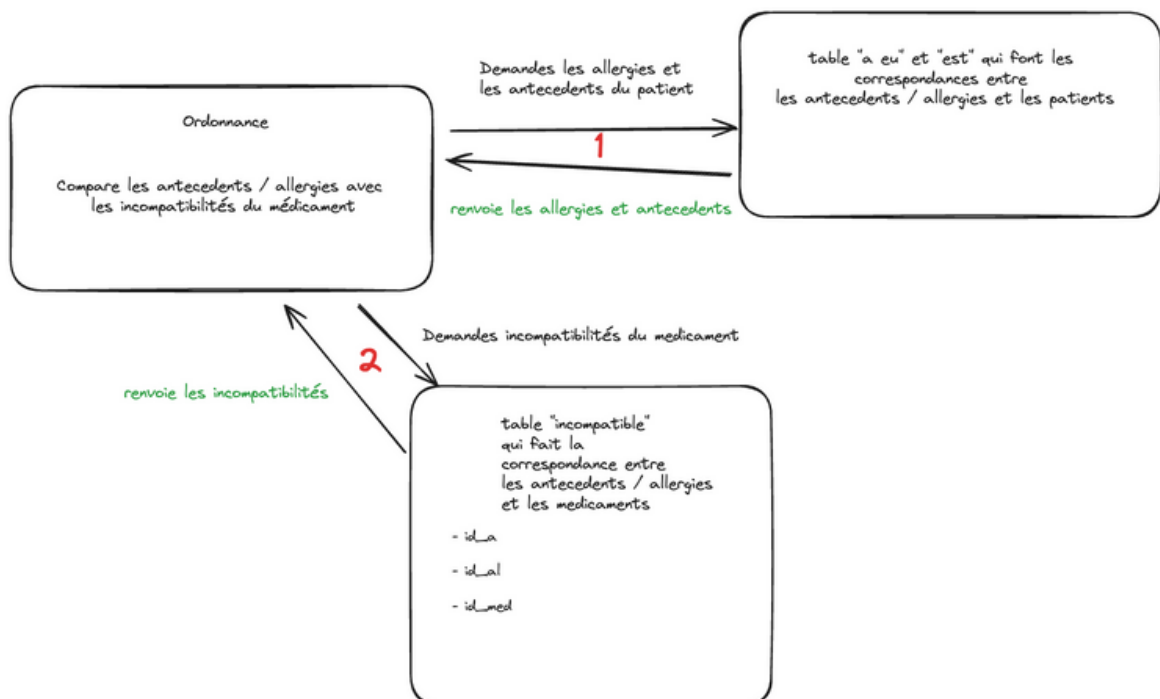
Avant d'écrire le code, j'ai créé des schémas pour déterminer l'organisation de mon code, de ma base de données et des différentes logiques. Voici le Modèle Conceptuel de Données que j'ai utilisé au début du projet. Veuillez noter que cette version a évolué au fil du temps en raison de l'ajout de fonctionnalités.



Pour avoir un moyen sécurisé de stocker les mots de passe en base de données, j'ai "hashé" celui-ci afin de le rendre illisible. Ma logique pour la connexion se présente donc ainsi :



Dans l'expression des besoins, il est également mentionné que lors de la création de l'ordonnance, GeStionB doit interdire la création de l'ordonnance si le médicament prescrit présente des incompatibilités avec les antécédents ou les allergies du patient. Voici donc mon schéma pour traiter ce problème :



Liaison entre le logiciel et la base de données

Pour lier ma base de données, j'utilise le package NuGet MySQL Connector. MySQL Connector fournit une interface entre les applications et le serveur de base de données MySQL, ce qui permet d'exécuter des requêtes, d'insérer, de mettre à jour et de supprimer des données, ainsi que d'effectuer d'autres opérations de gestion de bases de données.

Voici comment 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 définit la chaîne de connexion qui sera utilisée. Cette chaîne de connexion comprend le nom DNS du serveur, la base de données ciblée, l'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))
    {
        //ce l'on veut faire avec le resultat
    }
    conn.Close();
}
```

Les extraits de code ci-dessus sont écrits dans une classe qui gère la connexion avec la base de données. Le premier code fait appel à la chaîne de connexion et la stocke 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.

Réalisation des opérations de types CRUD

Faire des opérations de type CRUD signifie être capable de créer, lire, modifier et supprimer un élément dans la base de données. Dans mon application, les opérations de type CRUD seront utiles pour, par exemple :

- 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.

Afin d'expliquer chaque étape d'une opération CRUD, voici un schéma qui explique le fonctionnement de la visualisation des allergies d'un patient. Ce schéma correspond à une lecture de la base de données (BDD) :

```
//Evenement qui correspond au bouton pour gérer les allergies
1 reference
private void btn_PatientsDetails_Allergies_Click(object sender, EventArgs e)
{
    // Créer une nouvelle fenetre de viewAllergies en passe en parametre l'ID du patient
    ViewAllergies viewAntecedents = new ViewAllergies(Id);
    //Afficher la fenetre
    viewAntecedents.Show();
}
```

Page de
visualisation des
données d'un
patient

Page de
visualisation des
allergies d'un
patient

```
private int Id_p { get; set; }
//Initialise la variable qui contient l'ID du patient
1 reference
public ViewAllergies(int id_p)
{
    //recupère l'ID passé en parametre et definit la variable Id_p afin
    //de le rendre accessible à toute la classe
    Id_p = id_p;
    //initialise la fenetre
    InitializeComponent();
    //appelle la méthode UpdateDataGridView et lui passe en parametre l'id du patient
    UpdateDataGridView(id_p);
    //Evenement qui permet d'actualiser la GridView lorsque la fenetre est active
    this.Activated += ViewAllergies_Activated;
}

1 reference
private void ViewAllergies_Activated(object sender, EventArgs e)
{
    //appelle la méthode UpdateDataGridView et lui passe en parametre l'id du patient
    UpdateDataGridView(Id_p);
}

2 references
private void UpdateDataGridView(int id_p)
{
    //Créer un nouvel objet, dataAccess afin de rendre accessible les methodes de AllergiesDataAccess
    AllergiesDataAccess dataAccess = new AllergiesDataAccess();
    //initialise la grid view à null
    this.Grid_Allergies.DataSource = null;
    //Complete la dataGridView avec les allergies du patient renvoyé par
    //la méthode GetAllergieListFromDB de dataAccess
    this.Grid_Allergies.DataSource = dataAccess.GetAllergieListFromDB(id_p);
}
```

Classe de gestion
des accès vers la
BDD pour les
allergies

```
public DataTable GetAllergieListFromDB(int id_p)
{
    //initialise un nouvel objet de type DataTable
    DataTable dataTable = new DataTable();

    //Créer la connexion à la BDD grace à la chaine de connexion
    using (MySQLConnection conn = new MySQLConnection(connectionString))
    {
        //ouvre la connexion
        conn.Open();
        //definit la requete qui sera utilisee
        string query = "SELECT * FROM allergie WHERE id_al IN (SELECT id_al FROM est WHERE id_p = @id_patient)";
        using (MySQLCommand command = new MySQLCommand(query, conn))
        {
            //créer un parametre de requete, @id_apatient sera remplace au fonctionnement par l'ID du patient
            command.Parameters.AddWithValue("@id_patient", id_p);
            //complete la dataTable avec le resultat de la requete
            using (MySQLDataAdapter adapter = new MySQLDataAdapter(command))
            {
                adapter.Fill(dataTable);
            }
        }
        //ferme la connexion
        conn.Close();
    }
    // renvoie la dataTable
    return dataTable;
}
```

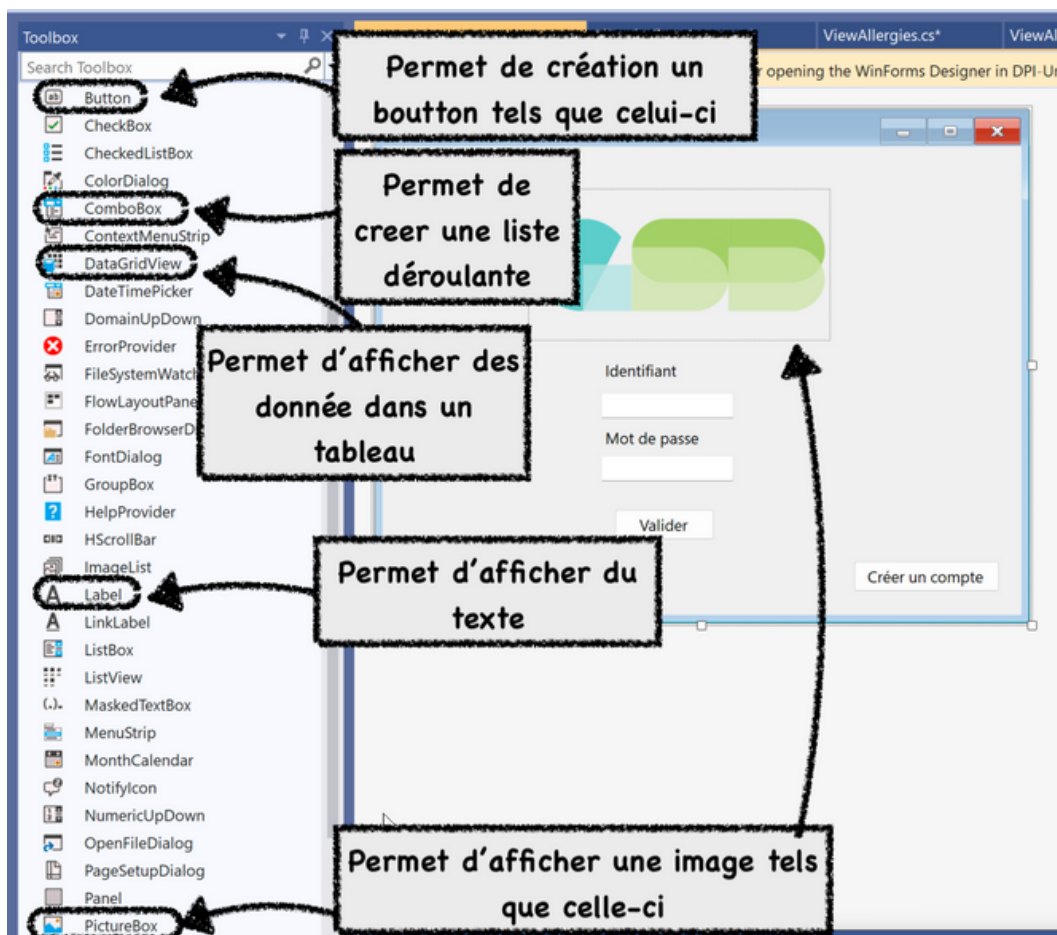
Dans l'exemple ci-dessus, j'ai utilisé une requête imbriquée afin de récupérer les bonnes données. Voici comment on peut lire cette requête :

Sélectionner toutes les colonnes de la table 'allergie' lorsque la colonne 'id_al' est présente dans les résultats de la sous-requête suivante : Sélectionner la colonne 'id_al' de la table 'est' lorsque la colonne 'id_p' est égale à @id_patient.

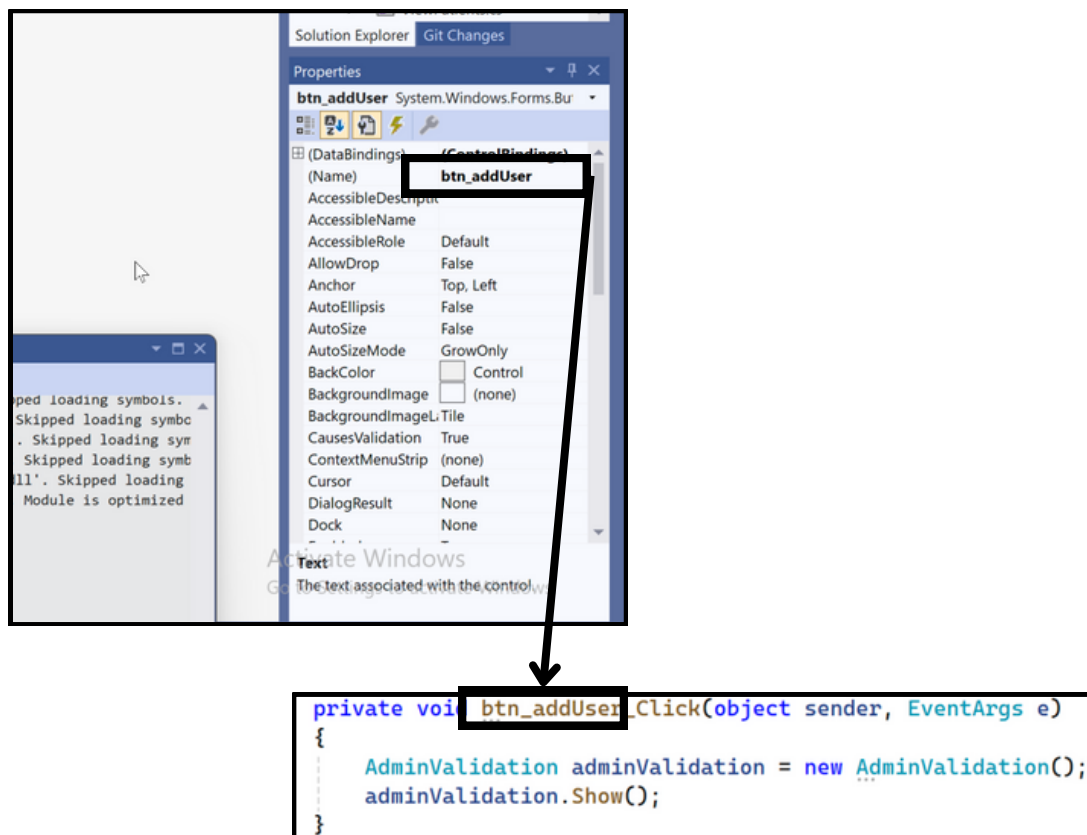
Ce principe de requête imbriquée est largement utilisé dans mon application.

Création des différentes vues

La création de vues avec Windows Forms se fait assez intuitivement. Visual Studio met à disposition des outils qui permettent de créer une interface simplement en utilisant le glisser-déposer. Voici les composants que j'ai utilisés dans mon projet, seule la TextBox, qui permet de créer des entrées de texte, manque sur la capture.



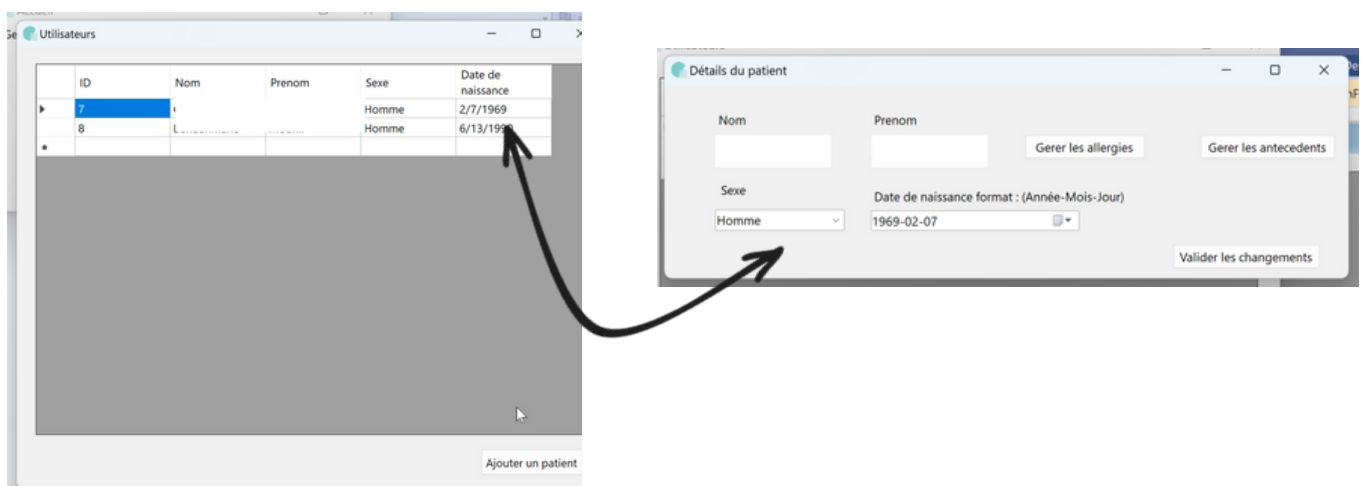
Le paramètre le plus important à modifier est le nom du composant "(Name)", car il correspond au début du nom de la méthode qui gère l'événement souhaité, par exemple "Click" pour un bouton.



Pour interconnecter différentes pages, la méthode utilisée consiste à créer un nouvel objet de la future fenêtre, puis à utiliser la méthode ".Show()" comme indiqué ci-dessus.

Interconnexion entre les pages

L'un des concepts importants à réaliser aussi dans mon logiciel est la possibilité d'échanger des données d'un composant à un autre. 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 faut que les données passent d'une fenêtre à l'autre.



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

```
// 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();
    }
}
```

Page de
visualisation des
patients

```
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 sex, 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 = sex;
    // 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
}
```

Page de
visualisation des
détails d'un
patients

Remplissage des listes déroulantes à partir de la base de données

Les listes déroulantes sont de bons moyens pour éviter les erreurs de correspondance entre le texte saisi dans les différents formulaires et la base de données. Par exemple, lorsque l'on se trouve sur la page de gestion des incompatibilités,

Gérer les incompatibilités

Ce médicament est incompatible avec :

Medicament	Allergie	Antecedent
Aspirine	Allergie au cacao	Ulcères gastriques actifs
		Tendinite
		Insuffisance rénale
		Grossesse

Valider

La sélection se fait grâce à une liste déroulante afin 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, ce qui sera expliqué plus en détail par la suite.

Concrètement le remplissage de "comboBox" se fait de cette façon :

```
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.FillDefaultValueComboxBoxMedicaments(Id);
    // Affecte la valeur par défaut de la liste déroulante combo_Medicaments en fonction
    // de l'ID de gestion de l'incompatibilité
}
```

Page de gestion des
incompatibilités

Classe qui
complète la
comboBox des
médicament

```
public void FillComboBox(ComboBox comboBox)
{
    using (MySQLConnection conn = new MySQLConnection(connectionString))
    {
        conn.Open();
        string query = "SELECT libelle_med FROM medicament ORDER BY libelle_med ASC;";
        // Requete qui ressort tous les libelle de la table medicament et les trie par ordre alphanique
        using (MySQLCommand command = new MySQLCommand(query, conn))
        {
            using (MySQLDataReader reader = command.ExecuteReader())
            {
                comboBox.Items.Clear();
                while (reader.Read())
                {
                    string value = reader.GetString(0);
                    comboBox.Items.Add(value);
                    //complete la comboxBox avec chaque ligne du resultat le requete
                }
            }
        }
        conn.Close();
    }
}
```



```

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
}

```

Méthode qui initialise la valeur par défaut de la comboBox

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 n'a pas de compte, un bouton pour créer un compte est disponible sur la page de connexion. Ce bouton ouvre d'abord une première fenêtre où le mot de passe administrateur est demandé afin d'interdire la création de compte aux personnes non autorisées. Ensuite, la page de création de compte s'ouvre et l'administrateur renseigne les informations du nouveau médecin. Lorsque le formulaire est validé, voici ce qui se passe :

```

private void btn_addMedecin_Valid_Click(object sender, EventArgs e)
{
    Bcrypt bCrypt = new Bcrypt();
    //créer un nouvel objet de la classe Bcrypt
    string hash = bCrypt.Encryption(this.box_AddMedecin_MDP.Text);
    //Utilise la méthode Encryption de la classe Bcrypt et recupere le resultat qui sera le hash du mot de passe
    MedecinDataAccess dataAccess = new MedecinDataAccess();
    dataAccess.AddMedecin(this.box_AddMedecin_nom.Text, this.box_AddMedecin_prenom.Text, this.date_AddMedecin.Text);
    //créer le medecin en BDD avec le mot de passe hasher
    MessageBox.Show("Utilisateur crée");
}

```

Actions réalisées après le clic

Méthode Encryption de ma classe Bcrypt

```

public string Encryption (string pswd)
{
    string hash = Bcrypt.Net.BCrypt.EnhancedHashPassword(pswd,13);
    //hash le mot de passe avec 13 "tours de boucle", ce qui prends environ 800ms
    return hash;
    //renvoie le hash
}

```

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 :

```
private async void Btn_Login_valid_Click(object sender, EventArgs e)
{
    MedecinDataAccess dataAccess = new MedecinDataAccess();
    string hash = dataAccess.GetHashForAuthentication(this.Box_Login_Username.Text);
    //récupere le hash de la BDD correspondant à l'utilisateur
    string nom_m = dataAccess.GetNameOfMedecin(this.Box_Login_Username.Text);
    //recupere le nom
    if (hash != null)
        //si un hash est trouver dans la BDD -> comparer les mots de passe
    {
        Bcrypt bcrypt = new Bcrypt();
        bool result = bcrypt.Description(this.Box_Login_Password.Text, hash);
        //utilise la methode description de la classe Bcrypt
        if (result)
        {
            //si le resultat est true -> ouvrir l'accueil et cacher la page de connexion
            Accueil accueil = new Accueil(nom_m);
            this.Hide();
            accueil.Show();
        }
        else
            //sinon refuser la connexion
        {
            MessageBox.Show("Mauvais identifiant/mot de passe");
        }
    }
    else
        //sinon attendre 800ms pour simuler la description du mot de passe
        //et ensuite interdire la connexion
    {
        await Task.Delay(800);
        MessageBox.Show("Mauvais identifiant/mot de passe");
    }
}
```

Actions réalisées
après le clic

Méthode
Description de
ma classe Bcrypt

```
public bool Description(string pswd, string hash)
{
    bool result = BCrypt.Net.BCrypt.EnhancedVerify(pswd, hash);
    //compare le mot de passe du formulaire au hash de la BDD
    return result;
    //renvoie true ou false
}
```

Il est important de préciser qu'avant le clic, j'ai ajouté 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 deviner quels sont les identifiants valides ou non. Actuellement, si un utilisateur essaie de se connecter avec un nom d'utilisateur invalide, la réponse sera immédiate. En revanche, si un nom d'utilisateur valide est utilisé, la réponse ne sera pas immédiate, car le logiciel va tenter de comparer le mot de passe de la BDD avec celui du formulaire. Cette comparaison prendra plus de temps et pourrait donc révéler un identifiant valide.

Création d'un ordonnance

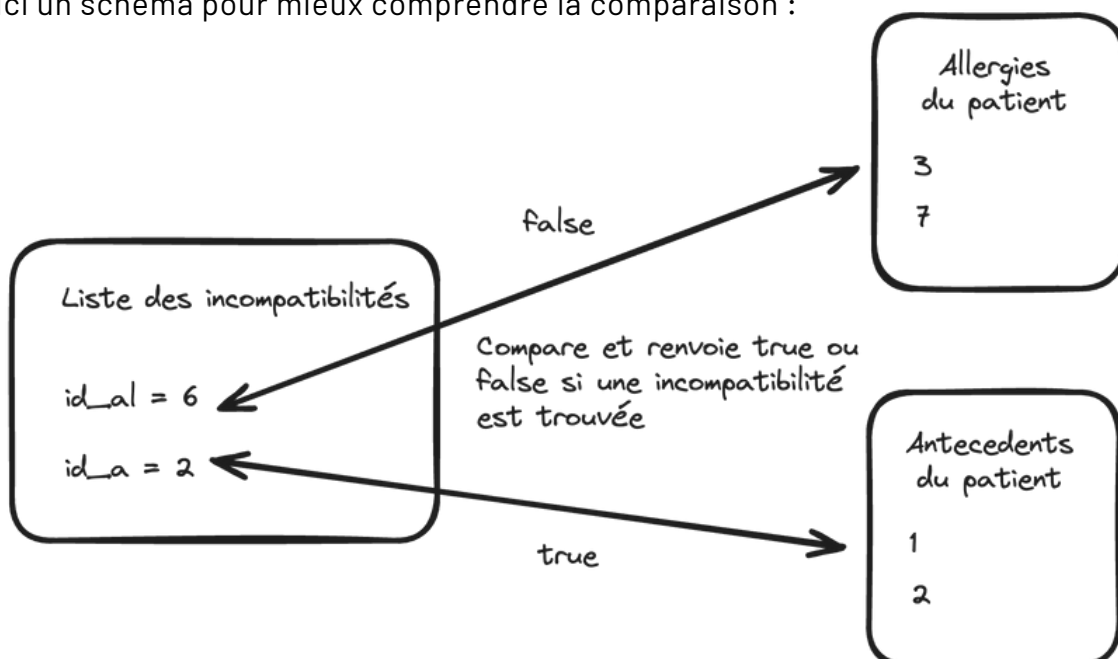
Pour correspondre au cahier des charges, il est nécessaire que lors de la création d'une ordonnance, GeStionB se doit d'interdire la création de celle-ci si une incompatibilité entre le patient et le médicament est trouvée.

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);
    // Récupération de la liste des allergies du patient à partir de la base de données
    List<string> listeAntecedents = GetAntecedentListFromDB(nom_prenom_p);
    // Récupération de la liste des antécédents du patient à partir de la base de données
    List<Incompatibilite> listeIncompatibilites = GetIncompatibiliteFromDB(libelle_med);
    // Récupération de la liste des incompatibilités du médicament à partir de la base de données
    bool isIncompatible = false;
    // Variable utilisée pour indiquer si le médicament est incompatible
    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;
        }
    }
    // Pour chaque allergie dans la liste des allergies du patient,
    // vérifier si elle est présente dans la liste des incompatibilités du médicament.
    // Si l'incompatibilité est trouvée -> sortir de la méthode

    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;
        }
    }
    // de même pour els antecedes
    if (!isIncompatible)
    {
        // si la compatibilité est validée -> création de l'ordonnance
        string date = DateTime.Now.ToString("yyyy-MM-dd");
        using (MySQLConnection conn = new MySQLConnection(connectionString))
        {
            conn.Open();
            string query = "INSERT INTO ordonnance (id_o, posologie, date, duree_traitement, instruction_specifique, id_m, id_p) VALUES (0, @posologie, @date, @duree, @instruction, @id_m, @id_p)";
            using (MySQLCommand command = new MySQLCommand(query, conn))
            {
                command.Parameters.AddWithValue("@posologie", posologie);
                command.Parameters.AddWithValue("@date", date);
                command.Parameters.AddWithValue("@duree", duree);
                command.Parameters.AddWithValue("@instruction", instructions);
                command.Parameters.AddWithValue("@id_m", libelle_med);
                command.Parameters.AddWithValue("@id_p", nom_prenom_p);
                command.ExecuteNonQuery();
            }
        }
    }
}
```

Voici un schéma pour mieux comprendre la comparaison :



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();
            //crée une nouvelle instance de PdfCreator
            pdfCreator.CreatePDF(selectedFolder, this.boxId.Text, this.boxMedecin.Text, this.boxDate.Text, this.comboPatient);
            //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
```

Actions réalisées
après le clic

Méthode qui crée
le PDF

```
1 reference
public void CreatePDF(string filePath, string id_o, string nom_m, string date_o, string nom_p, string libelle_med, string libelle_patient)
{
    String OutFile = filePath + "\\ordonnance " + id_o + ".pdf";
    //initialise OutFile avec le fichier de destination + le nom du PDF avec l'ID de l'ordonnance pour le rendre unique
    Document doc = new Document();
    // Crée un nouvel objet Document qui représente un document PDF vide.
    PdfWriter.GetInstance(doc, new FileStream(OutFile, FileMode.Create));
    // Obtient une instance de PdfWriter et lie le document à un flux de sortie.
    // Dans cet exemple, le flux de sortie est créé en utilisant un objet FileStream
    // qui écrit dans un fichier spécifié par la variable OutFile.
    // FileMode.Create indique que si le fichier existe déjà, il sera écrasé.
    doc.Open();
    // Ouvre le document pour commencer à écrire son contenu.

    BaseColor noir = new BaseColor(0, 0, 0);
    //créer la couleur noir

    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);
    //définit les différentes polices

    Paragraph InfoMedecin = new Paragraph("DR." + nom_m + " \nLaboratoires GSB \n21 rue des Lisette, 92220 Bagneux", policeTitre);
    InfoMedecin.Alignment = Element.ALIGN_LEFT;
    doc.Add(InfoMedecin);
    //créer un nouveau paragraphe et l'ajoute au document en l'alignant à gauche

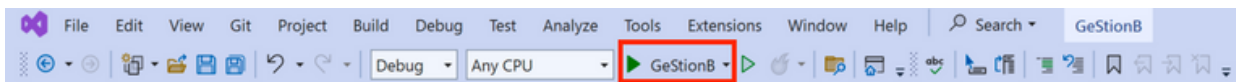
    doc.Close();
    //Ferme le document
    Process.Start(new ProcessStartInfo
    {
        FileName = OutFile,
        UseShellExecute = true
    });
    //Ouvre avec le lecteur pdf par défaut le PDF créé
}
```

Manuel utilisateur

Pour pouvoir tester l'application, il est nécessaire d'avoir le service MySQL. Ensuite, il faut exécuter le script SQL présent dans le répertoire GitHub correspondant à la base de données.

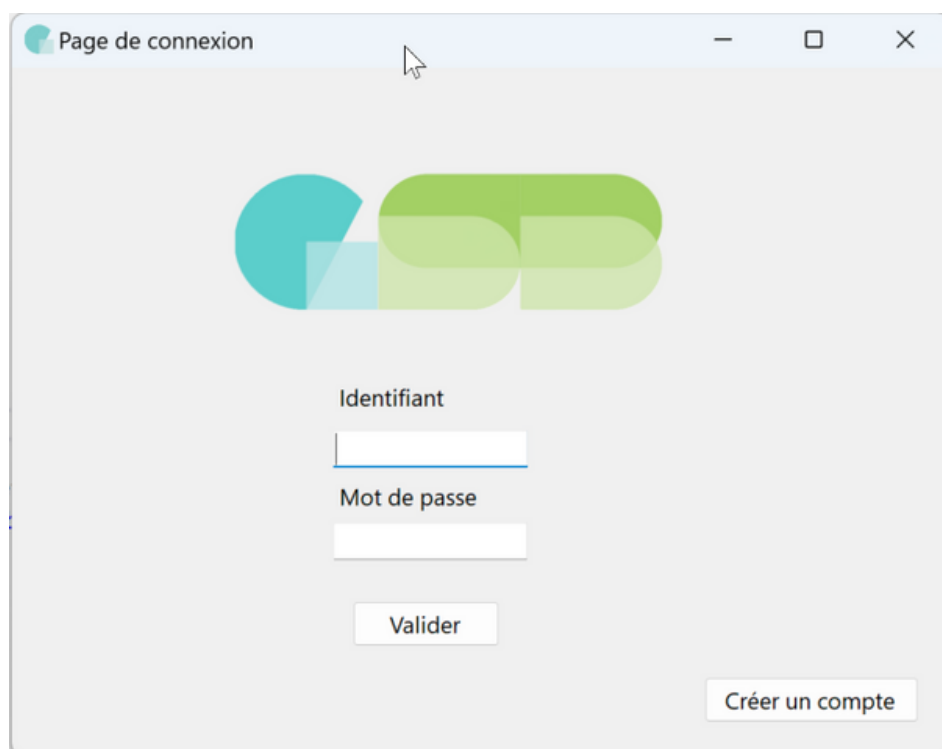
Une fois la base de données importée, il faut ouvrir le projet correspondant au projet dans Visual Studio, à partir du répertoire GitHub.

Lorsque le projet est ouvert et que le service MySQL est en cours d'activité, vous pouvez lancer l'application en cliquant sur le bouton suivant :

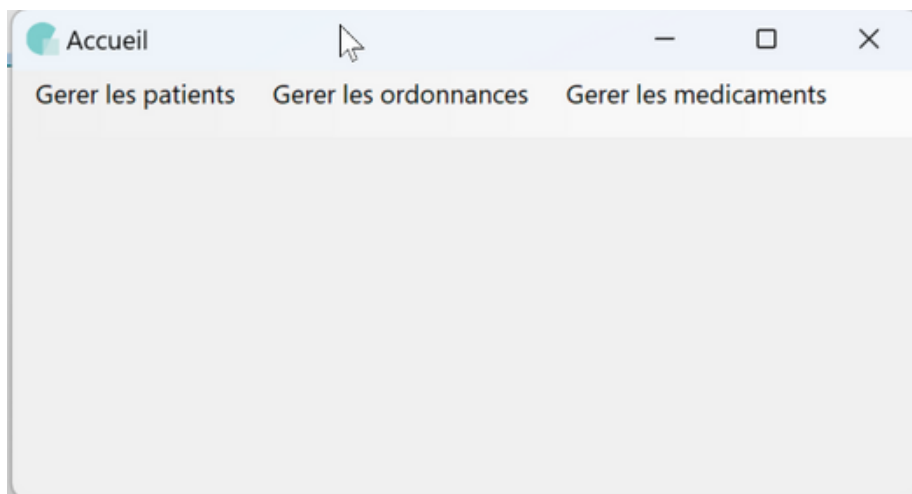


Lorsque l'application est lancée, la page de connexion apparaîtra. Les identifiants de test sont les suivants :

- Identifiant : testeur
- Mot de passe : test

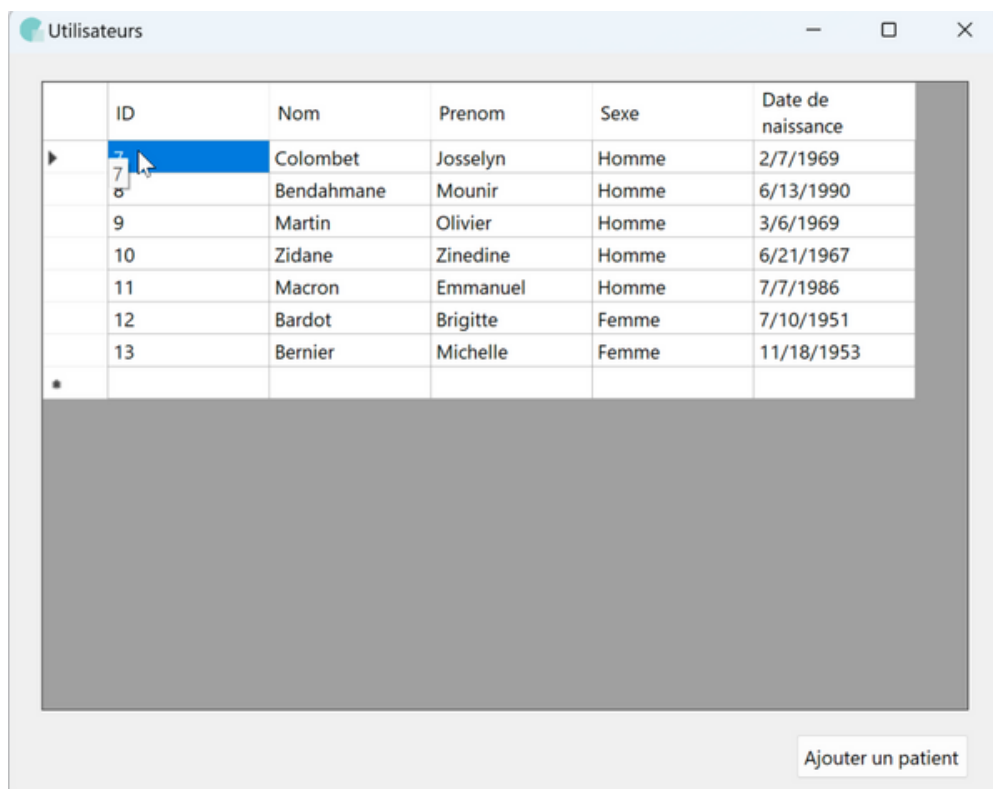


Une fois la connexion réussie, voici comment se présente la page d'accueil. Vous pourrez retrouver les fonctionnalités principales de l'application : gérer les patients, les ordonnances et les médicaments.

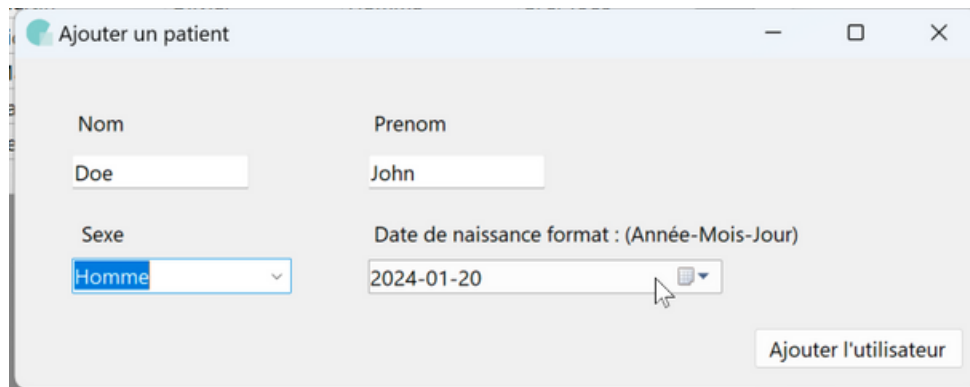


Gérer les patients

Lorsque vous cliquez sur "Gérer les patients", cette fenêtre s'ouvre et affiche tous les patients ainsi que leurs informations.

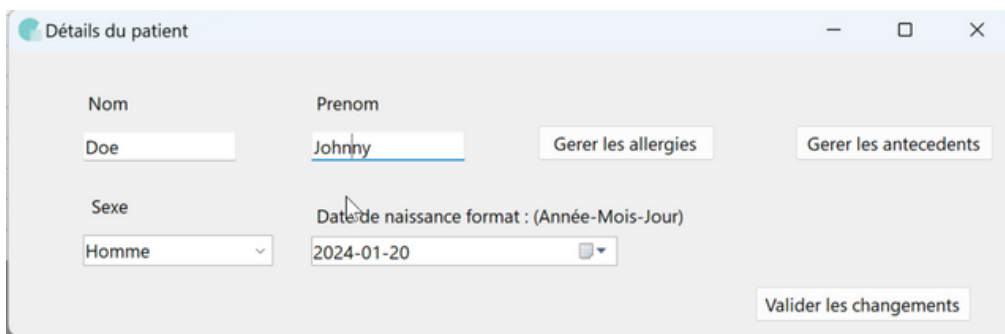


Pour ajouter un patient, il suffit de cliquer sur "Ajouter un patient", de lui saisir les informations nécessaires, puis de valider.



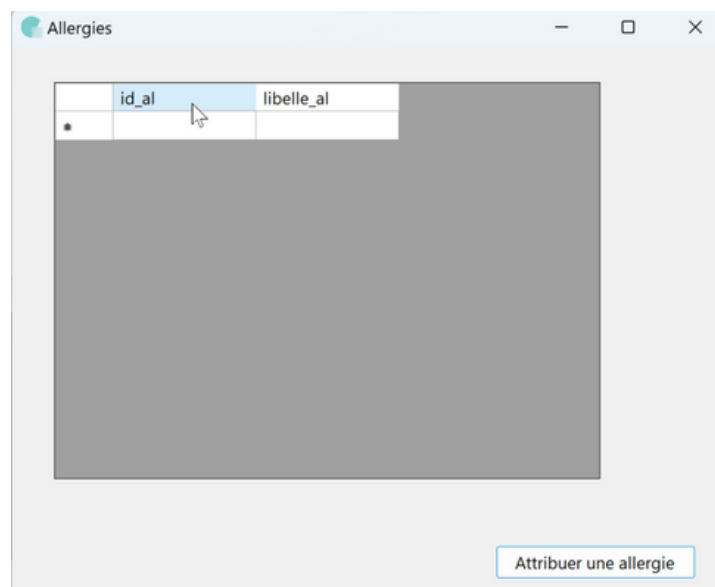
The screenshot shows a window titled "Ajouter un patient". It contains four input fields: "Nom" with the value "Doe", "Prenom" with the value "John", "Sexe" with a dropdown menu showing "Homme", and "Date de naissance format : (Année-Mois-Jour)" with the value "2024-01-20". A button labeled "Ajouter l'utilisateur" is located at the bottom right.

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



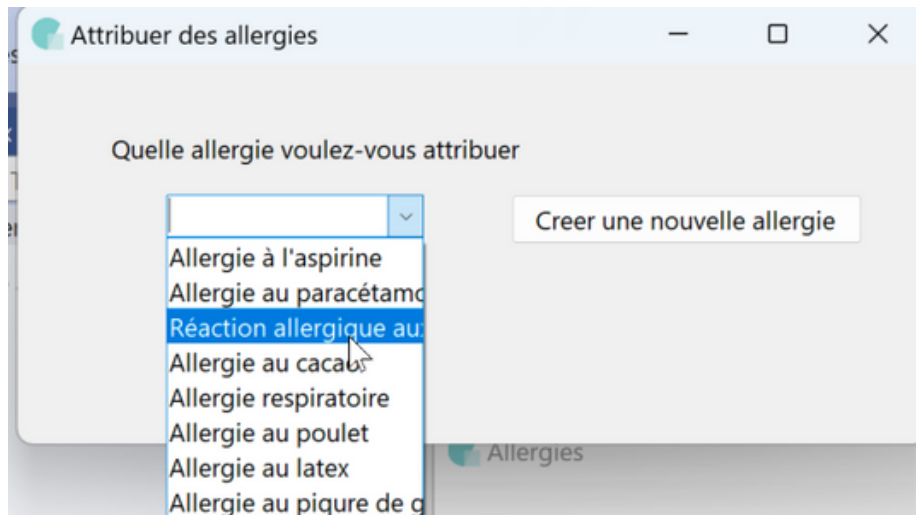
The screenshot shows a window titled "Détails du patient". It contains the same four input fields as the previous window, but the "Prenom" field now has the value "Johnny". There are two buttons: "Gerer les allergies" and "Gerer les antecedents". A button labeled "Valider les changements" is located at the bottom right.

Pour la gestion des antécédents ou des allergies, le fonctionnement est le même. Après avoir cliqué sur le bouton pour gérer les allergies/antécédents du patient, cette fenêtre s'ouvre (par exemple pour les allergies) :



The screenshot shows a window titled "Allergies". It contains a table with two columns: "id_al" and "libelle_al". The table is currently empty. A button labeled "Attribuer une allergie" is located at the bottom right.

Nous constatons par exemple qu'aucune allergie ne lui est attribuée. Attribuons-lui une allergie en utilisant le bouton "Attribuer une allergie".



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

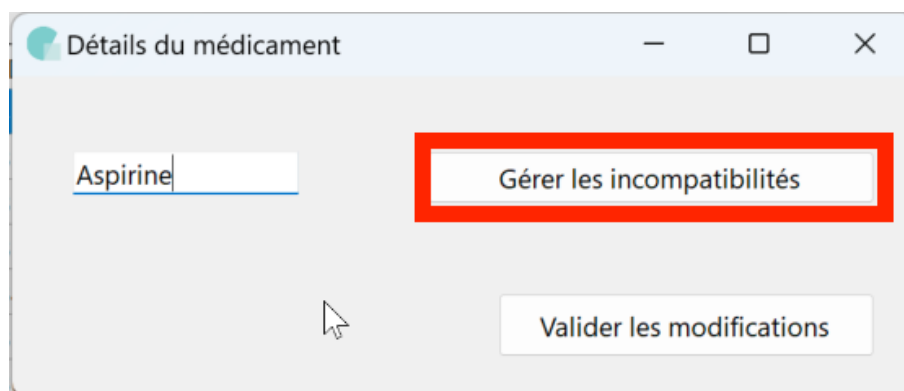


Il suffit simplement de saisir l'intitulé de la future allergie et de valider.

Gérer les médicaments et les incompatibilités

Le fonctionnement pour gérer les médicaments est le même que pour la gestion des patients.

La gestion des incompatibilités se fait sur la page des détails du médicament que l'on souhaite gérer.



À partir de cette fenêtre, nous pouvons facilement gérer les incompatibilités d'un médicament.

The screenshot shows a window titled "Gérer les incompatibilités". It contains three columns: "Médicament", "Allergie", and "Antécédent". Under "Médicament", there is a dropdown menu with "Aspirine" selected. Under "Allergie", there is a dropdown menu with "Allergie au cacao" selected. Under "Antécédent", there is a dropdown menu with a list of conditions: "Ulcères gastriques actifs", "Tendinite", "Insuffisance rénale", and "Grossesse". A mouse cursor is pointing at the "Antécédent" dropdown menu.

Gérer les ordonnances

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.

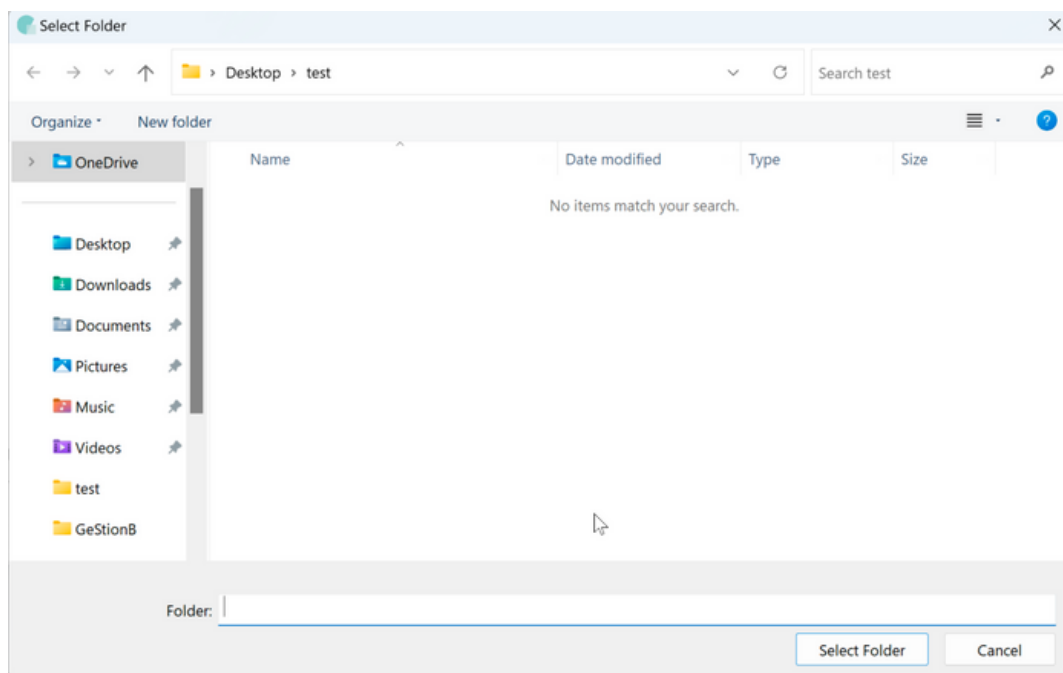
Si le médicament ne peut pas être prescrit au patient en raison d'incompatibilités, ce message s'affichera :

The screenshot shows a dialog box with a close button (X) in the top right corner. The text inside the dialog box reads: "Le médicament est incompatible avec une allergie du patient". At the bottom right of the dialog box, there is an "OK" button.

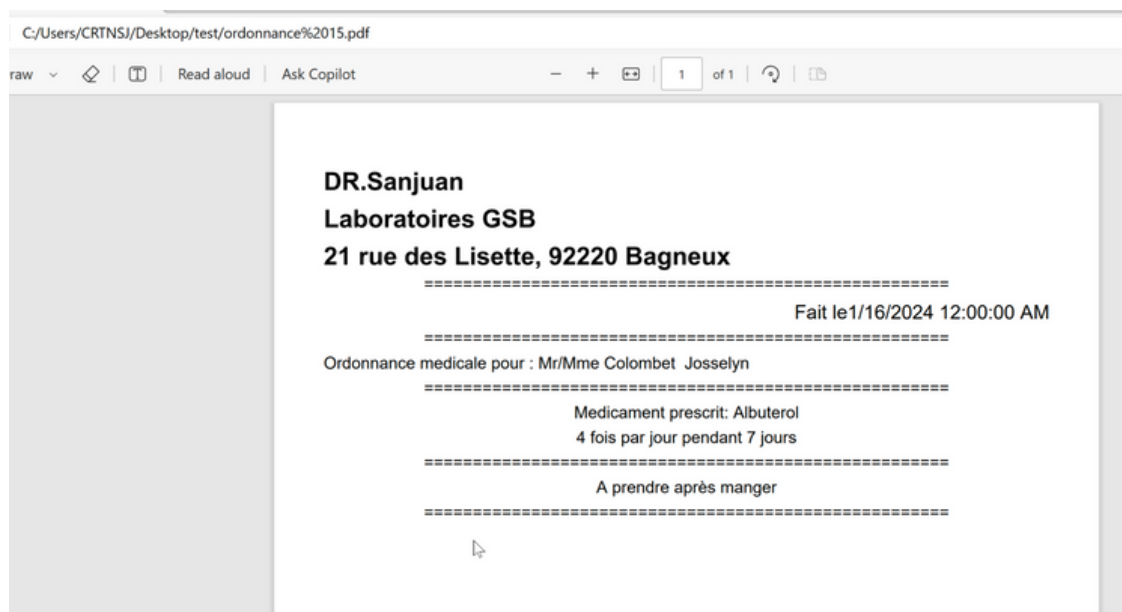
Pour générer le PDF d'une ordonnance, il faut cliquer sur l'ordonnance souhaitée dans la liste des ordonnances, puis cliquer sur le bouton "Générer le PDF".

The screenshot shows a window titled "Détails de l'ordonnance". It contains several fields for creating an order. The fields are arranged in a grid-like structure. The first row contains "Posologie" (4 fois par jour), "Date de création" (1/16/2024 12:00:0), and "Durée du traitement en jours" (7). The second row contains "Instruction spécifique" (A prendre après m) and "Identifiant" (15). The third row contains "Médicament prescrit" (Albuterol), "Patient" (Colombet Josselyn), and "Medecin" (Sanjuan). At the bottom left, there is a button "Supprimer l'ordonnance". At the bottom right, there is a button "Générer le PDF" which is highlighted with a red rectangle.

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.



Fermer l'application

Afin de fermer l'application il suffit de fermer la page d'accueil

En cas de problèmes :

Vous pouvez me contacter à l'adresse suivante : crtnsj.pro@gmail.com

Conclusion




Pour conclure, ce projet a été extrêmement intéressant à réaliser, car il m'a permis de découvrir de nouvelles technologies telles que C#, .NET, Windows Forms, Visual Studio, iText ainsi que d'approfondir mes connaissances avec Bcrypt, MySQL.

J'ai dû me documenter et rechercher des réponses à mes questions pour résoudre les problèmes rencontrés. Cette expérience a été enrichissante et formatrice, car c'était la deuxième fois que je réalisais un projet de développement de cette envergure, ce qui m'a obligé à adopter une rigueur dans l'écriture du code pour éviter de me perdre au fil du temps.

Cependant, je regrette de ne pas avoir réussi à rendre le logiciel fluide et agréable à utiliser. Pour moi il manque aussi une touche de personnalisation graphique, l'application a encore trop la touche "Windows Forms".

Malgré tout, je suis très fier du résultat obtenu après environ 50 heures de travail, de découvertes, d'organisation, de réflexion et de documentation. J'ai hâte de proposer des versions futures de ce projet, avec pour objectifs une plus grande clarté dans le code, une meilleure sécurité et de nouvelles fonctionnalités. J'aimerais également rendre cette application accessible, c'est-à-dire permettre à tous de l'utiliser dans les meilleures conditions possibles.



Annexes

Maquette:

La gestion des antécédents et des allergies se fait sur des fenêtres différentes, mais je les ai rassemblées dans le schéma car elles présentent la même architecture. Les pages de gestion des ordonnances et des médicaments, bien que non représentées, suivent la même logique que celle de la gestion des patients.

