

ITH KAGNANA
BTS SIO 2



eMeds

APPLICATION LOURDE C#
WINFORMS



OCTOBRE 2023

Sommaire

1. Contexte “Laboratoire GSB”
2. Les objectifs et les besoins
3. Analyse fonctionnelle

Contexte “Laboratoire GSB”

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 unissent 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.

Domaine d’étude

L’entreprise souhaite porter une attention nouvelle à sa force commerciale dans un double objectif : obtenir une vision plus régulière et efficace de l’activité menée sur le terrain auprès des praticiens, mais aussi redonner confiance aux équipes malmenées par les fusions récentes.

Les visiteurs

La force commerciale d’un laboratoire pharmaceutique est assurée par un travail de conseil et d’information auprès des prescripteurs. Les visiteurs médicaux (ou délégués) démarchent les médecins, pharmaciens, infirmières et autres métiers de santé susceptibles de prescrire aux patients les produits du laboratoire.

L’objectif d’une visite est d’actualiser et rafraîchir la connaissance des professionnels de santé sur les produits de l’entreprise. Les visiteurs ne font pas de vente, mais leurs interventions ont un impact certain sur la prescription de la pharmacopée du laboratoire.

Pour donner une organisation commune aux délégués médicaux, l’entreprise a adopté l’organisation de la flotte de visiteurs existant chez Galaxy, selon un système hiérarchique

par région et, à un niveau supérieur, par secteur géographique (Sud, Nord, Paris-Centre, etc).

Il n'y a pas eu d'harmonisation de la relation entre les personnels de terrain (Visiteurs et Délégués régionaux) et les responsables de secteur. Les habitudes en cours avant la fusion ont été adaptées sans que soient données des directives au niveau local.

Les objectifs et les besoins

1. Professionnel

Le but de ce client lourd est de permettre à un médecin de gérer les ordonnances de ses patients. On prend également en compte les allergies et les antécédents du patient qui peuvent avoir une influence sur les médicaments qu'il peut prendre ainsi que les incompatibilités entre les médicaments. Dans ces cas-là, nous allons avertir le médecin qu'il peut mettre en danger le patient mais nous allons lui laisser la décision finale lors de la création de l'ordonnance.

2. Personnel

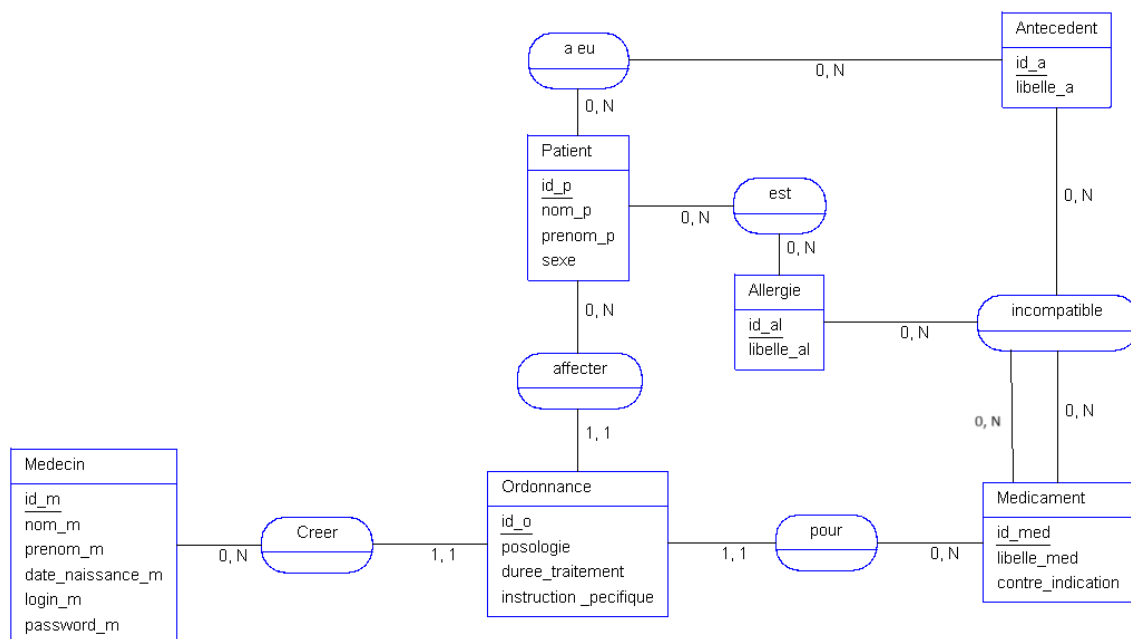
Personnellement, ce projet a aussi pour but de :

- D'apprendre dans un contexte professionnel le langage de programmation C#
- D'approfondir mes connaissances en base de données relationnel (MySQL)
- Découvrir la création d'une application lourde

Analyse fonctionnelle

lien github : <https://github.com/iKagnana/AP3-eMEDS>

Base de donnée - MySQL



MCD de notre application

Le Modèle de Conception de Données nous permet de concevoir et visualiser les données que nous allons mettre dans notre base de données et les différentes interactions que les tables vont avoir entre elles.

Package NutGet : MySQL

Le script de la base de donnée se situe dans le projet et se nomme **ap3-emeds_2024-01-19.sql**

Modification des tables pour s'adapter aux besoins

Les médecins

```
ALTER TABLE medecin ADD role VARCHAR(100)
```

On ajoute le champ role afin de pouvoir bloquer l'utilisateur dans certaines actions comme par exemple l'ajout de médecin ou d'allergies et antécédents. Il existe deux rôles : "ADMIN" et "USER".

Le rôle "ADMIN" permet au médecin administrateur d'avoir tous les droits. Tandis que le médecin avec le rôle "USER" aura uniquement certains droits.

Il peut :

- Uniquement consulter les médicaments, il ne peut pas les modifier
- Créer des ordonnances
- Gérer des patients

Il ne peut pas :

- Consulter ni gérer les médecins de l'application
- Consulter ni gérer les allergies et antécédents

Les patients

```
ALTER TABLE patient ADD num_secu VARCHAR(100)
```

On ajoute le champ num_secu afin d'avoir un moyen unique d'identifier le patient autre que l'id. Le numéro de sécurité sociale est utilisé pour récupérer l'id lors de la deuxième partie de l'ajout des allergies et antécédents du patient.

Les ordonnances

```
ALTER TABLE ordonnance ADD code VARCHAR(100)
```

On ajoute le champ code qui sera un UUID afin de pouvoir ajouter à l'ordonnance plusieurs médicaments.

```
CREATE TABLE ligne_ordonnance (  
  id_o int NOT NULL,  
  id_med int NOT NULL,  
  PRIMARY KEY (id_o, id_med),  
  KEY ligne_ordonnance_Medicament_FK (id_med),  
  CONSTRAINT ligne_ordonnance_Medicament_FK FOREIGN KEY (id_med)  
  REFERENCES medicament ( id_med ),  
  CONSTRAINT ligne_ordonnance_Ordonnance_FK FOREIGN KEY (id_o)  
  REFERENCES ordonnance (id_o)  
)
```

Cette table va me permettre de relier un médicament avec une ordonnance.

Conception

[Lien figma vers le schéma de conception de l'AP3](#)

On y retrouve notamment les requêtes utilisées ainsi que le schéma fonctionnel de notre application.

Médicament

Gestion des médicaments

La gestion des médicaments se fait grâce au formulaire AddMedicamentForm.cs.

On y retrouve plusieurs fonctionnalités :

- ajout de médicament
- affichage de la liste des médicaments
- recherche de médicament dans la liste

Ajout de médicament - GSB

Rechercher

Ajout de médicament

Libellé

Contre-indications

Ajouter

	Id	Libellé	Contre-indications
▶	1	doliprane	ne pas prendre plus de 6 par jours
	2	Aspirine	Allergie à l'aspirine
	3	Paracétamol	Allergie au paracétamol
	4	Ibuprofène	Ulcères gastriques actifs
	5	Amoxicilline	Réaction allergique aux pénicillines
	6	Ciprofloxacine	Tendinite
	7	Metformine	Insuffisance rénale
	8	Omeprazole	Grossesse
	9	Losartan	Hyperkaliémie
	10	Simvastatine	Maladie du foie active
	11	Amlodipine	Hypotension sévère
	12	Atorvastatine	Maladie du foie active
	13	Sertraline	Syndrome sérotoninergique
	14	Fluoxetine	Syndrome sérotoninergique
	15	Diazepam	Dépendance aux substances
	16	Prednisone	Infections fongiques systémiques
	17	Lisinopril	Angioedème
	18	Albuterol	Tachycardie
	19	Gabapentine	Dépendance

Pour modifier un médicament, il suffit de **cliquer sur la cellule correspondant dans le dataGridView**. Cette action ouvrira une fenêtre qui permettra à l'utilisateur de modifier, supprimer et gérer les incompatibilités.

Détails du médicament - GSB

Détails du médicament :

Libellé

Aspirine

Contre-indications

Allergie à l'aspirine

Valider les modifications

Supprimer le médicament

Allergies incompatibles

Allergie au pollen

	Id	Libelle
▶	4	Allergie à l'aspirine

Antécédents incompatibles

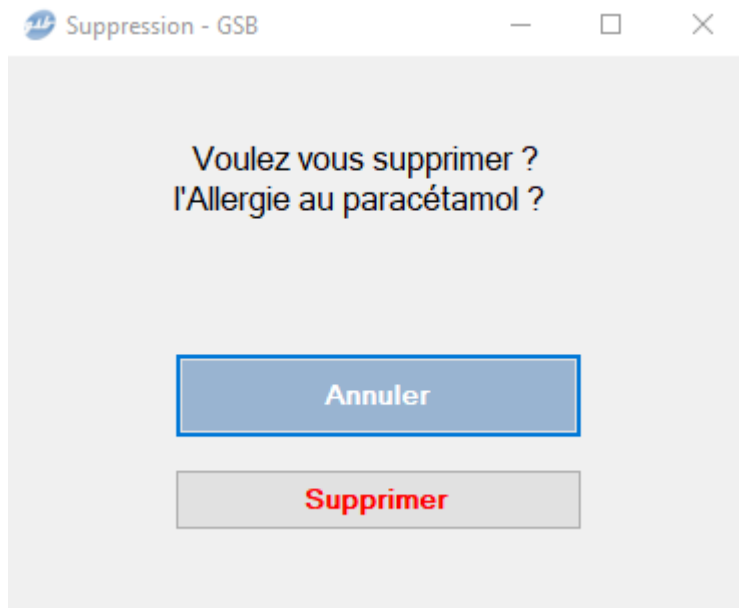
Ulcères gastriques actifs

Médicaments incompatibles

doliprane

	Id	Libelle
▶	8	Omeprazole
	6	Ciprofloxacine

Pour supprimer les incompatibilités, il faut **cliquer sur l'allergie, l'antécédent ou le médicament** que l'on souhaite retirer de la liste. Une fenêtre va s'ouvrir demandant la confirmation à l'utilisateur.



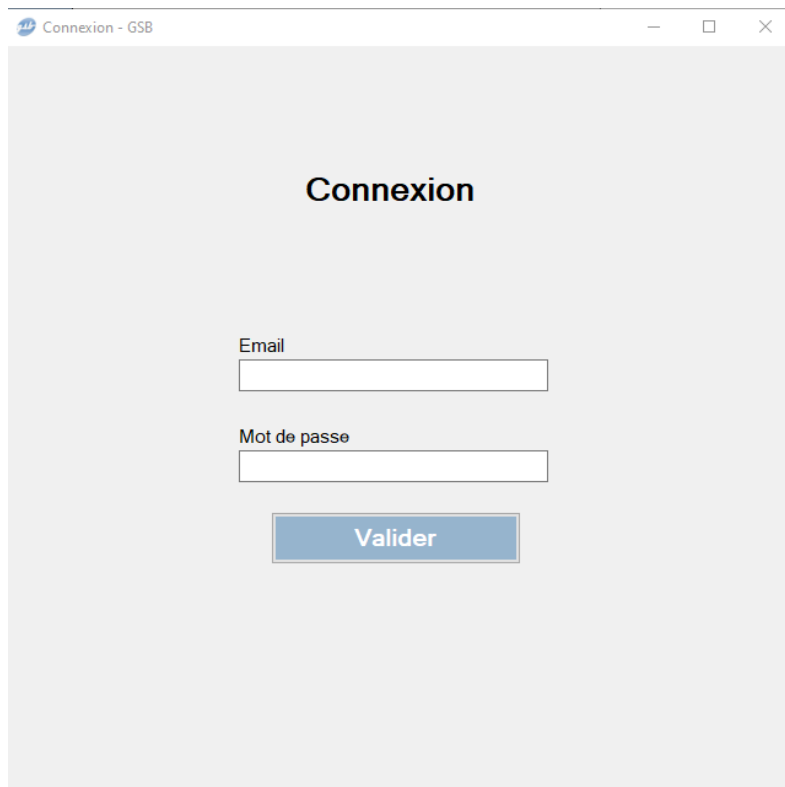
Seul un administrateur a les droits de modifier un médicament.

Médecin

Utilisateurs

Les médecins sont les utilisateurs de notre application. Ils peuvent alors se connecter et utiliser l'application pour gérer les ordonnances des patients.

Page de connection :



Connexion - GSB

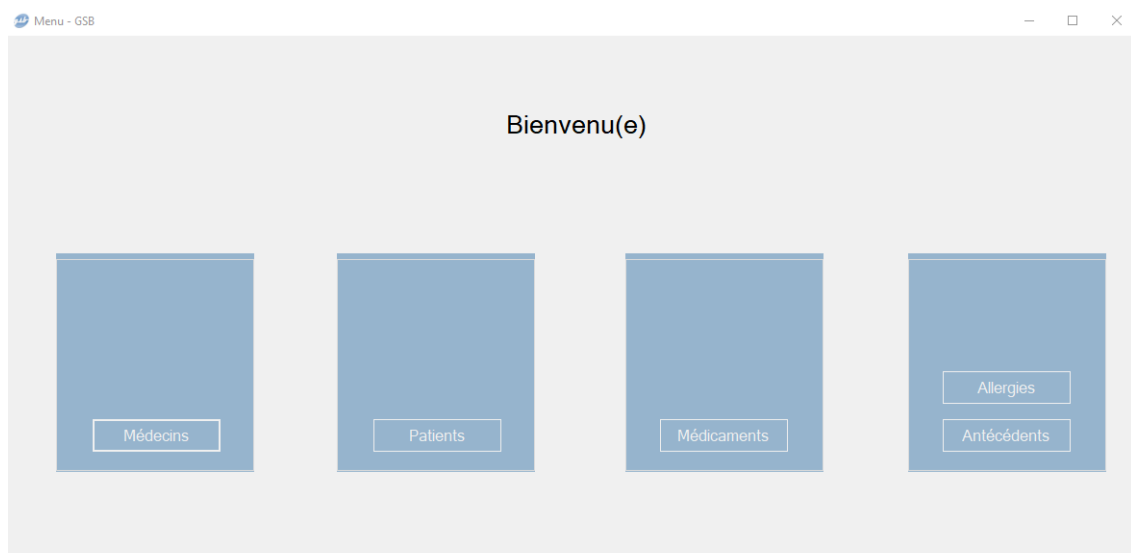
Connexion

Email

Mot de passe

Valider

Une fois connecté, les médecins administrateurs auront accès à ce menu :



Menu - GSB

Bienvenu(e)

Médecins

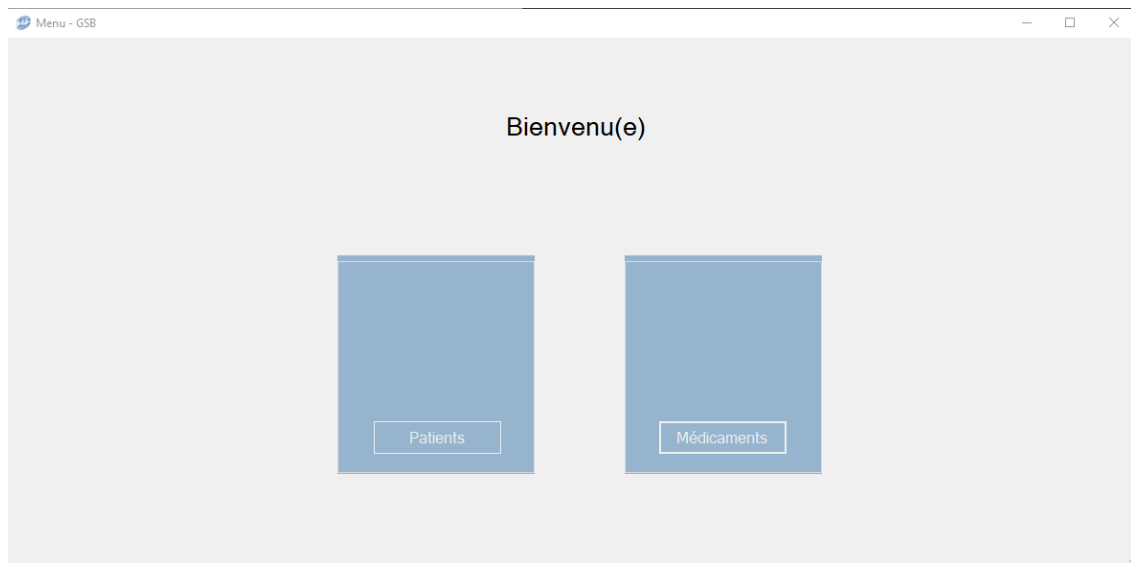
Patients

Médicaments

Allergies

Antécédents

Tandis que les médecins non administrateurs auront accès à celui-ci :

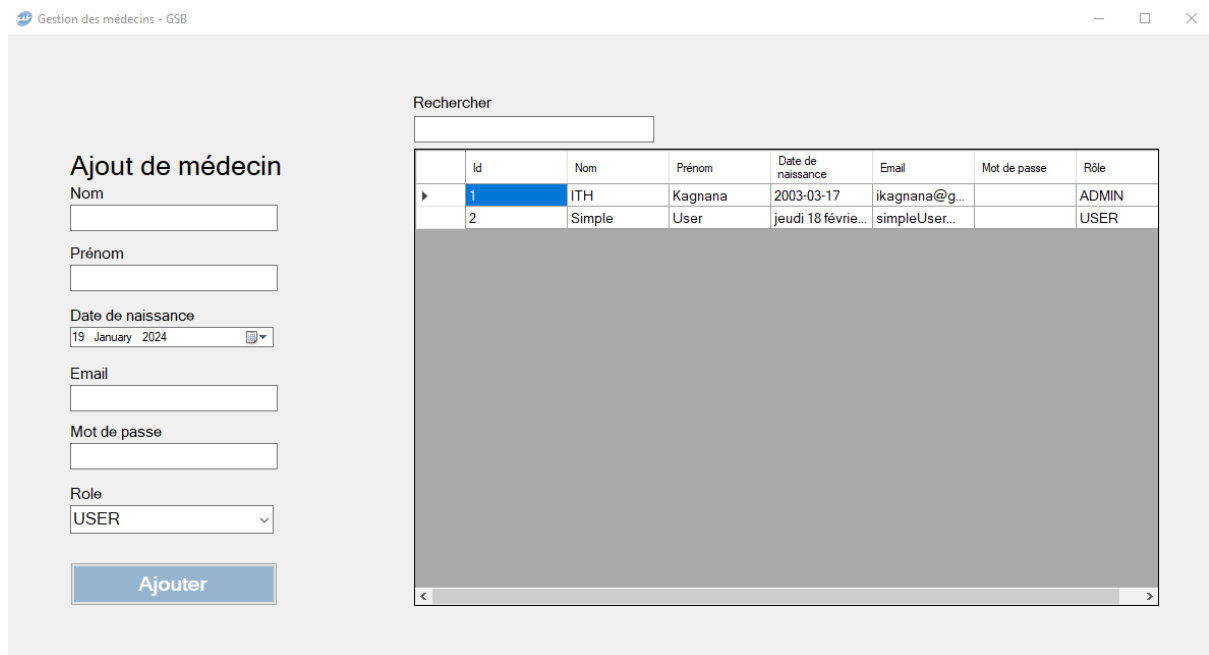


Gestion des médecins

La gestion des médecins ne peut se faire uniquement par un administrateur via le formulaire AddMedecin.cs. Ce formulaire est accessible en **cliquant sur le bouton “Médecin” du Menu.**

On peut à partir de ce formulaire :

- ajouter un médecin
- avoir la liste des médecins
- rechercher dans la liste
- modifier le médecin
- supprimer le médecin



Lors de l'ajout d'un médecin on va venir vérifier si l'email existe déjà, si l'email existe déjà alors le médecin ne sera pas ajouté et un MessageBox apparaîtra pour prévenir que l'email est déjà utilisé.

Gestion des médecins - GSB

Ajout de médecin

Nom
test

Prénom
ajout

Date de naissance
19 January 2024

Email
ikagnana@gmail.com

Mot de passe
•••••

Rôle
USER

Ajouter

Rechercher

	Id	Nom	Prénom	Date de naissance	Email	Mot de passe	Rôle
▶	1	ITH	Kagnana	2003-03-17	ikagnana@g...		ADMIN
	2	Simple	User	jeudi 18 févrie...	simpleUser...		USER

Cet email a déjà été attribué.

OK

```

57 private void btnAjouter_Click(object sender, EventArgs e)
58 {
59     Medecin newMedecin = new Medecin(this.lastnameTxt.Text,
60                                     this.firstnameTxt.Text,
61                                     this.birthDatePicker.Text,
62                                     this.emailTxt.Text,
63                                     this.pwTxt.Text,
64                                     this.selectedRole);
65     // test if email already attributed
66     RequestStatus statusUnique = controller.UniqueEmail(this.emailTxt.Text);
67     if (!statusUnique.success)
68     {
69         MessageBox.Show("Cet email a déjà été attribué.");
70         // exit func
71         return;
72     }
73     RequestStatus status = controller.AddMedecin(newMedecin);
74     if (status.success)
75     {
76         MessageBox.Show("Médecin ajouté.");
77         UpdateDataGrid();
78         ResetForm();
79     } else
80     {
81         MessageBox.Show("Erreur lors de l'ajout du médecin.");
82     }
83 }

```

Pour accéder à la fenêtre de modification du médecin, il suffit de **cliquer sur le médecin** que l'on souhaite modifier. On accède alors à cette fenêtre.

The screenshot shows a web application window titled "Détails du médecin - GSB". The main heading is "Détails du médecin". The form contains the following fields and controls:

- Nom:** Text input with the value "ITH".
- Prénom:** Text input with the value "Kagnana".
- Date de naissance:** Date picker showing "17 March 2003".
- Email:** Text input with the value "ikagnana@gmail.com".
- Mot de passe:** Password input field (currently empty).
- Role:** Dropdown menu with "ADMIN" selected.
- Buttons:**
 - A blue button labeled "Modifier le mot de passe" located below the password field.
 - A blue button labeled "Modifier" centered below the other fields.
 - A red button labeled "Supprimer le médecin" centered below the "Modifier" button.

Tous les champs sont modifiables sauf le champ du mot de passe qui requiert d'**appuyer sur le bouton "Modifier le mot de passe"**. Cette action ouvrira une fenêtre qui permet à l'utilisateur de modifier le mot de passe du médecin choisi.

The screenshot shows a dialog box titled "Changement de mot de passe - GSB". It contains the following elements:

- Nouveau mot de passe:** Text input field.
- Button:** A blue button labeled "Modifier le mot de passe" located below the input field.

On va également utiliser BCrypt pour hasher le nouveau mot de passe de l'utilisateur grâce à la fonction `UpdateMedecinPassword()` dans `MedecinController.cs`

Allergies et antécédents

Gestion des allergies et des antécédents

La gestion des allergies et des antécédents n'est possible que par l'administrateur via le formulaire AddObjetPatient.cs

On peut à partir de ce formulaire :

- ajouter une allergie ou un antécédent
- afficher toutes les allergies ou les antécédents
- modifier une allergie ou un antécédent
- supprimer une allergie ou un antécédent

	Id	Libelle
▶	1	Allergie au pollen
	3	Allergie aux poils de chat
	4	Allergie à l'aspirine
	5	Allergie au paracétamol
	6	Réaction allergique aux pénicillines
	7	Allergie aux achiariens

Pour pouvoir modifier ou supprimer une allergie ou un antécédent, il suffit de **cliquer sur l'allergie ou l'antécédent que l'on souhaite modifier**. La fenêtre DetailsObjetPatient.cs s'ouvrira.

Libellé

Allergie au pollen

Modifier

Supprimer le médecin

Patient

Gestion des patients

La gestion des patients se fait grâce au formulaire PatientList.

On y retrouve plusieurs fonctionnalités :

- liste des patients
- recherche dans la liste
- ajout d'un patient
- modification des détails du patient
- suppression d'un patient
- gérer les allergies et les antécédents du patient

	Accès ordonnance	Id	Nom	Prénom	Sexe	Numéro de Sécu	Allergies	Antécédents
▶		1	Doe	John	M	123456789123412	4	4
		2	Martin	Thomas	M	154392345434566	0	0
		3	Laroche	Julie	F	243234523124323	0	0
		4	Delacourt	Francis	F	243243532341243	0	0
		5	Rabbit	Roger	M	123442342512314	0	0
		6	test	ajout	F	241231453121435	1	1
		7	test	test	F	1232341234353	0	0
		8	test	reload	M	123124123134325	1	0

L'ajout du patient se fait en 2 parties. La première permet de créer le patient avec des informations de la table patient.

Ajout du patient étape 1 - GSB

Ajout de patient

Nom

Prénom

Numéro de Sécurité Sociale

☒ Homme
☐ Femme

Pour le numéro de sécurité sociale, on va vérifier si il est bien composé de 15 chiffres et s'il est unique.

Avec le numéro de sécurité sociale, on va pouvoir récupérer l'id de l'utilisateur pour la deuxième partie de l'ajout du patient. Dans cette deuxième partie, on pourra lui ajouter des allergies et des antécédents.

Ajout d'un patient étape 2 - GSB

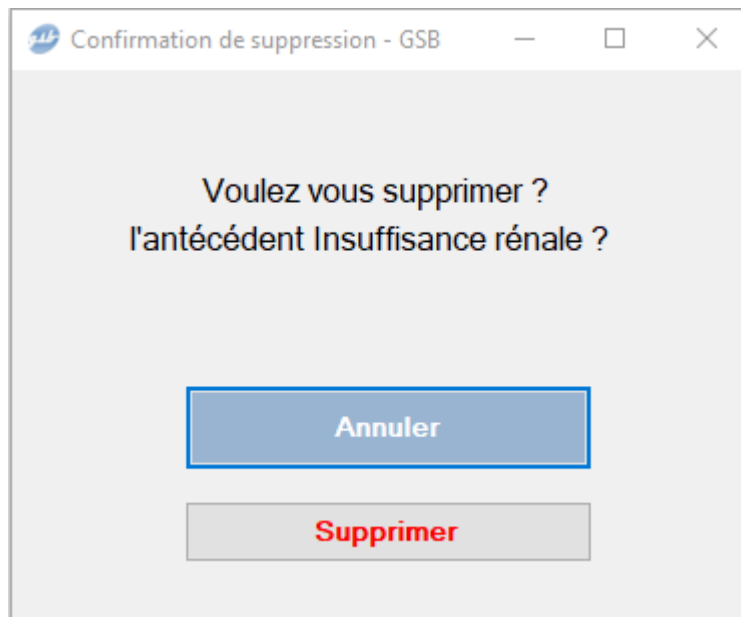
Allergie(s)

	Id	Libelle
▶	5	Allergie au parac...

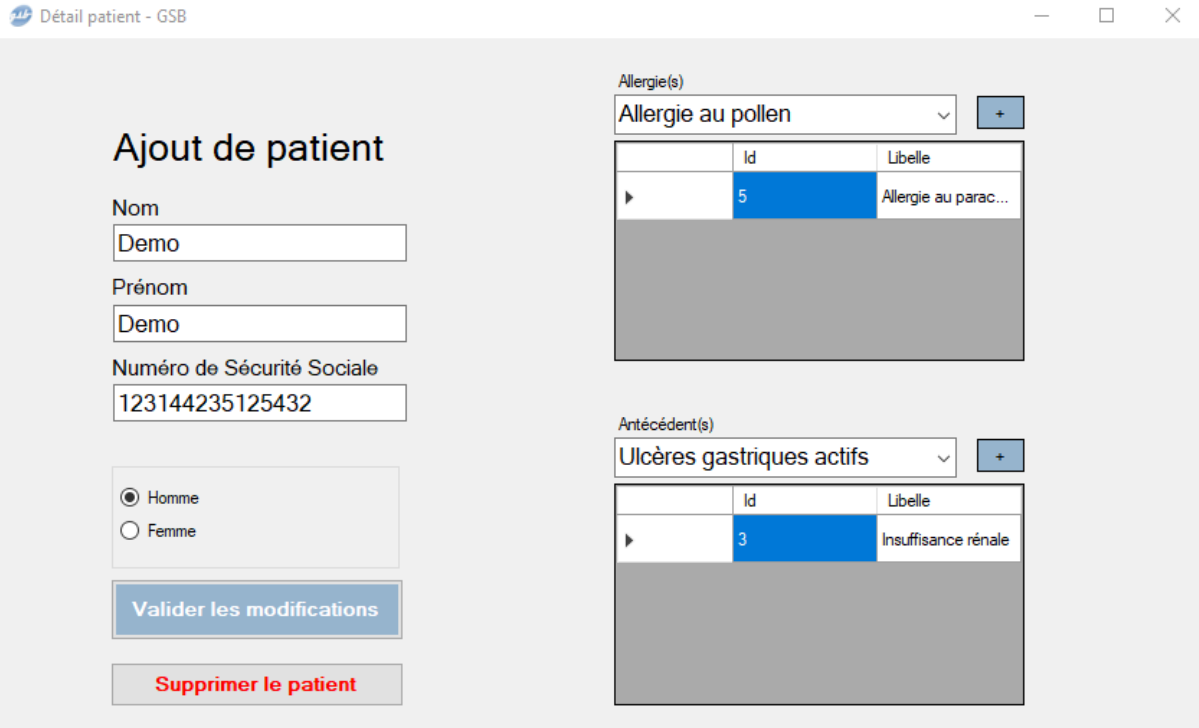
Antécédent(s)

	Id	Libelle
▶	3	Insuffisance rénale

Pour supprimer une allergie ou un antécédent ajouté dans l'étape 2 du formulaire, il suffit de **cliquer sur l'élément que l'on souhaite supprimer**.



La modification, suppression d'un patient et la gestion de ses allergies et antécédents se fait dans le formulaire DetailsPatient. On y accède en **cliquant sur le patient que l'on souhaite modifier**.

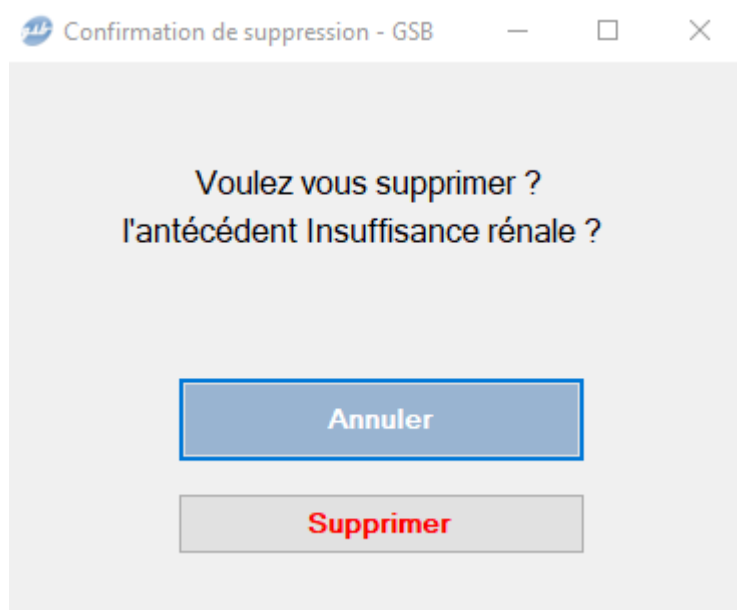


A screenshot of the "Détail patient - GSB" form. The form is divided into two main sections. The left section, titled "Ajout de patient", contains input fields for "Nom" (Demo), "Prénom" (Demo), and "Numéro de Sécurité Sociale" (123144235125432). Below these are radio buttons for "Homme" (selected) and "Femme". At the bottom of this section are two buttons: "Valider les modifications" and "Supprimer le patient". The right section contains two tables. The first table, titled "Allergie(s)", has a dropdown menu showing "Allergie au pollen" and a "+" button. Below it is a table with columns "Id" and "Libelle". The second table, titled "Antécédent(s)", has a dropdown menu showing "Ulcères gastriques actifs" and a "+" button. Below it is a table with columns "Id" and "Libelle".

	Id	Libelle
▶	5	Allergie au parac...

	Id	Libelle
▶	3	Insuffisance rénale

Pour supprimer des antécédents ou des allergies du patient, il **suffit de cliquer sur la ligne** et une fenêtre s'ouvrira pour demander confirmation à l'utilisateur.

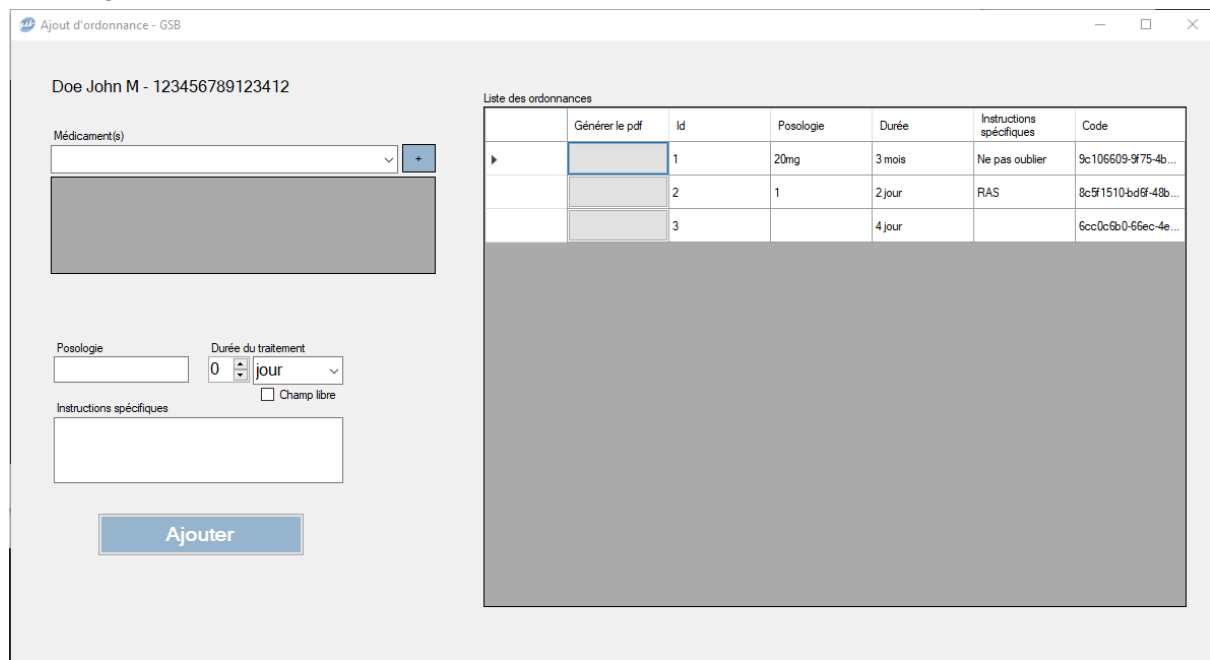


Gestion des ordonnances

La gestion des ordonnances est une des fonctionnalités principales du projet.

Les fonctionnalités sont :

- l'ajout d'ordonnance
- consultation d'ordonnance
- génération de pdf



Lors de l'ajout d'un médicament dans l'ordonnance, on va vérifier plusieurs choses :

- incompatibilité avec une allergie du patient
- incompatibilité avec un antécédent du patient
- incompatibilité avec un autre médicament déjà ajouté

Cette vérification se fait avec la fonction `TextIncompatibility()` dans `AddOrdonnanceForm.cs`

Par exemple pour les antécédents, on va vérifier ainsi :

```
117 // test if incompatibility with patient's antecedent
118 foreach (ObjetPatient antecedent in patientAntecedents)
119 {
120     RequestStatus status = medController.GetIncompatibilityAntecedent(antecedent.Id, med.Id);
121     if (status.success)
122     {
123         incompatibilities.Add(new Incompatibility(typeItem.Antecedent, med, antecedent));
124         this.warningText.Visible = true;
125         break;
126     } else if (!status.success && (status.typeError == typeError.NoConnection || status.typeError == typeError.InvalidCredentials ))
127     {
128         MessageBox.Show("Attention aucune possibilité de vérifier s'il y a une incompatibilité avec le médicament sélectionné.");
129         break;
130     }
131 }
```

Ici on va gérer si les erreurs sont dues à un problème de connexion à la base car si le médecin n'est pas capable de savoir qu'il y a une incompatibilité avec l'antécédent du patient, on peut être tenu responsable si le patient est en danger.

Si jamais il y a des incompatibilités, il y aura des textes d'avertissement qui vont apparaître dans le formulaire de création de l'ordonnance. Un dernier avertissement sera donné quand l'utilisateur voudra valider l'ordonnance.

- Ici dans le cas où un médicament est incompatible avec une allergie ou un antécédent du patient :

Ajout d'ordonnance - GSB

Doe John M - 123456789123412

Médicament(s)

Aspirine

	Id	Libelle
▶	2	Aspirine

Posologie

Durée du traitement

0 jour

Instructions spécifiques

☐ Champ libre

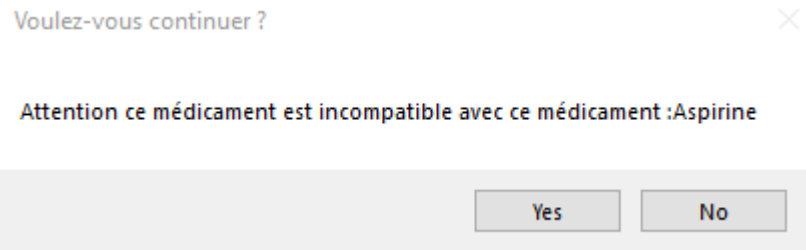
Attention certains médicaments sont incompatible avec le patient

Ajouter

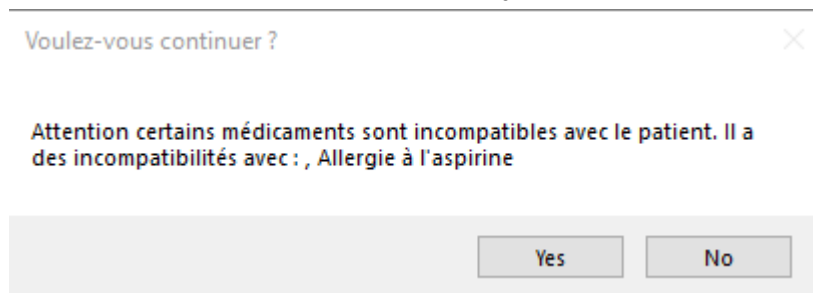
Liste des ordonnances

	Générer le pdf	Id	Posologie	Durée	Instructions spécifiques	Code
▶		1	20mg	3 mois	Ne pas oublier	9c106609-9f75-4...
		2	1	2 jour	RAS	8c5f1510-bd6f-4...
		3		4 jour		6cc0c6b0-66ec...

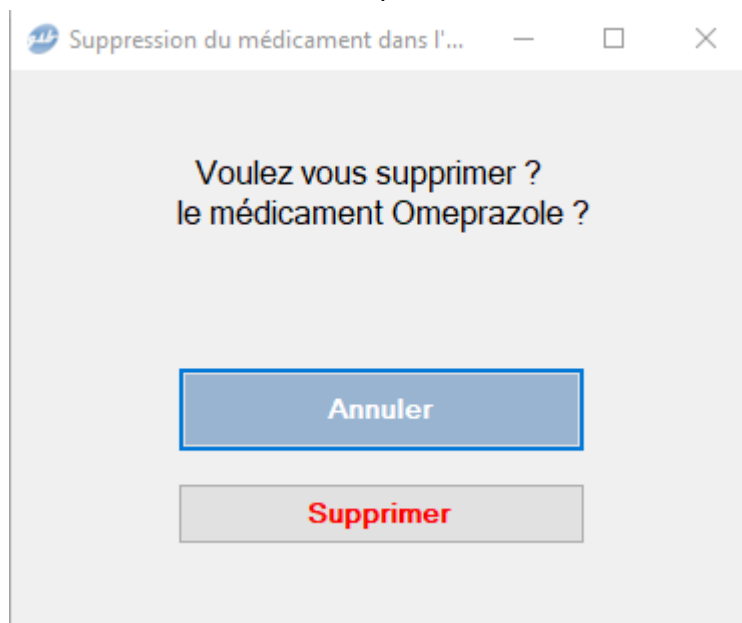
- MessageBox qui s'affiche si on ajoute un médicament incompatible avec un autre médicament.



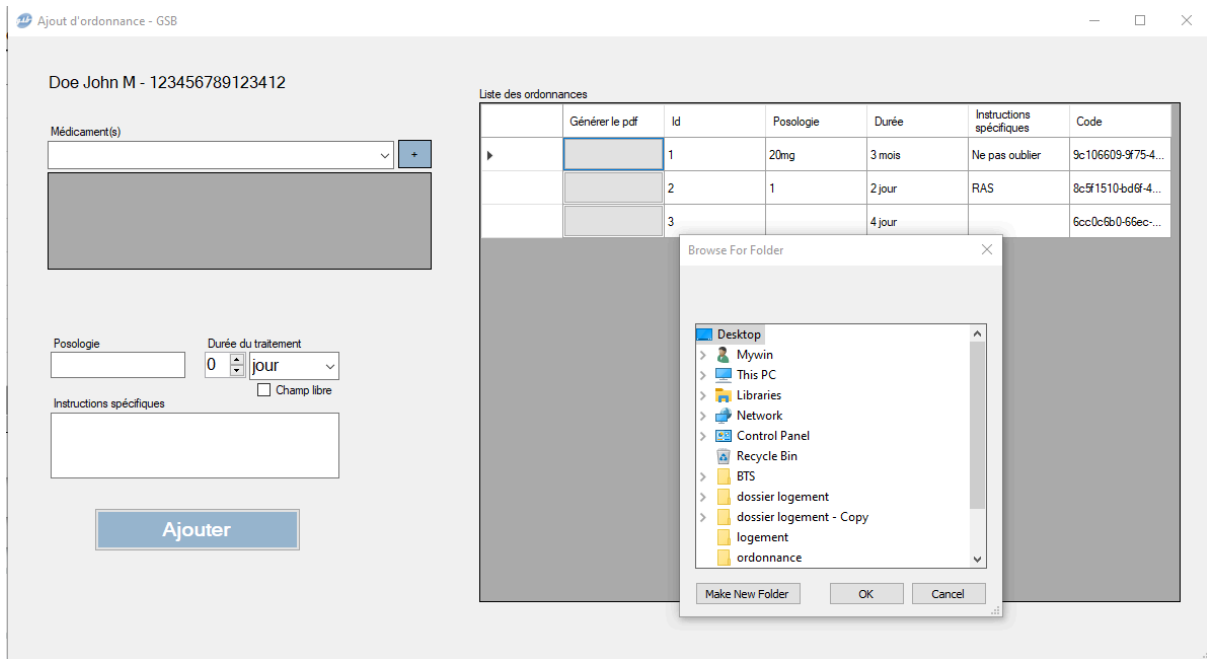
- Dernier avertissement avant l'ajout d'une ordonnance



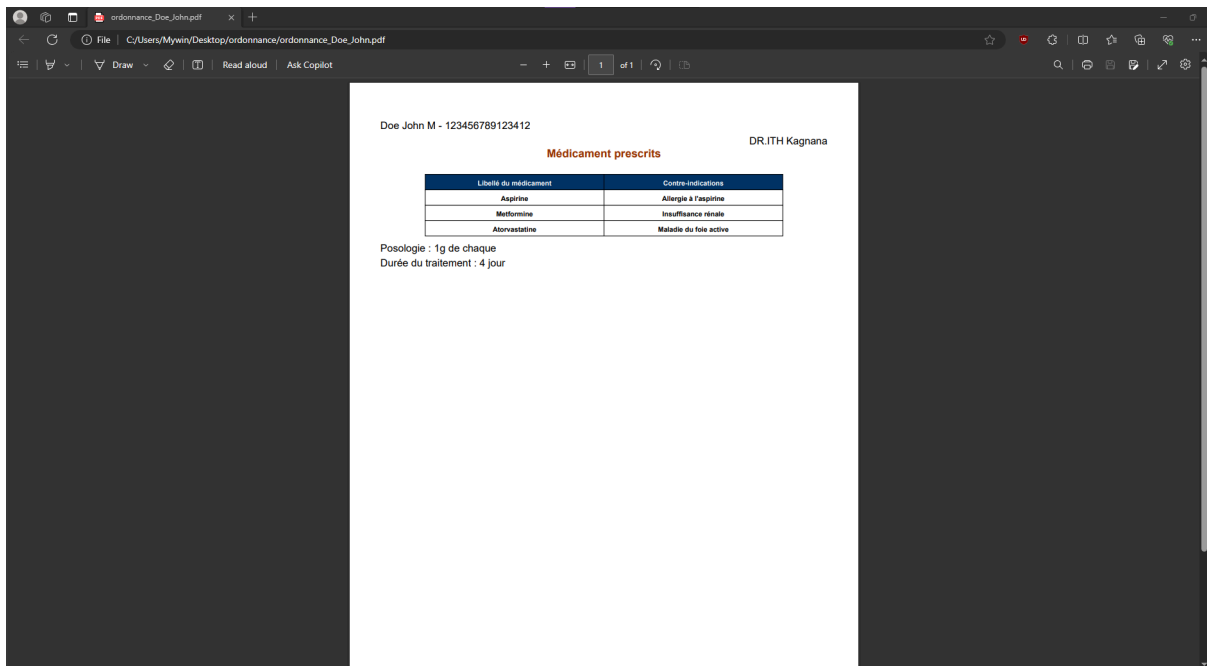
Pour supprimer un médicament de la liste d'ordonnance, il suffit de **cliquer dessus dans la liste** et une fenêtre s'ouvrira pour demander confirmation à l'utilisateur.



On peut également générer le pdf lié à l'ordonnance grâce au bouton "Générer le pdf" depuis le dataGridView qui liste toutes les ordonnances du patient.



Cette action va nous ouvrir un FolderBrowserDialog qui va nous permettre de choisir là où on veut enregistrer notre pdf. Ensuite le fichier s'ouvrira de lui même.



On peut également consulter l'ordonnance en **cliquant sur l'ordonnance souhaitée dans le dataGridView** lisant les ordonnances du patient.

Doe John M - 123456789123412

Posologie

1g de chaque

Durée du traitement

4 jour

Instructions spécifiques

Si douleurs persistante, coupler avec
du doliprane

Générer le pdf

Liste de médicaments

	Id	Libelle	ContreIndication
▶	0	Aspirine	Allergie à l'aspirine
	0	Metformine	Insuffisance rénale
	0	Atorvastatine	Maladie du foie active

On peut également générer le pdf depuis cette page.

La génération de pdf se fait grâce au package NutGet iTextSharp. On va venir implémenter la fonction GeneratePDF dans la classe Ordonnance et ainsi chaque objet de la classe pourra gérer un PDF.

Sécurité

Cryptage de mot de passe

Grâce à BCrypt, on va pouvoir crypter les mots de passe afin de ne pas les avoir en clair dans la base de données.

Package NutGet : BCrypt.Net-Next

Dans MedecinController.cs, on va utiliser la fonction **HashPassword** de **BCrypt.Net.BCrypt** pour crypter le mot de passe de l'utilisateur avant de l'envoyer en base. Ainsi, le mot de passe ne sera pas affiché en clair.

```
52 // method create user type medecin
53 public RequestStatus AddMedecin(Medecin medecin)
54 {
55     RequestStatus status = new RequestStatus();
56     try
57     {
58         // create connection to the db to make query
59         using (MySqlConnection conn = new MySqlConnection(connectionString))
60         {
61             conn.Open();
62             string query = "INSERT INTO medecin (nom_m, prenom_m, login_m, password_m, date_naissance_m, role) " +
63                 "VALUES (@lastname, @firstname, @login, @password, @birthdate, @role)";
64
65             using (MySqlCommand command = new MySqlCommand(query, conn))
66             {
67                 // hash password
68                 string hashedPassword = BCrypt.Net.BCrypt.HashPassword(medecin.Password);
69                 command.Parameters.AddWithValue("@lastname", medecin.Lastname);
70                 command.Parameters.AddWithValue("@firstname", medecin.FirstName);
71                 command.Parameters.AddWithValue("@login", medecin.Username);
72                 command.Parameters.AddWithValue("@password", hashedPassword);
73                 command.Parameters.AddWithValue("@birthdate", medecin.BirthDate);
74                 command.Parameters.AddWithValue("@role", medecin.Role);
75                 int result = command.ExecuteNonQuery();
76                 conn.Close();
77                 return status.GetRequestStatusNoError(result);
78             }
79         }
80     } catch (MySqlException e)
81     {
82         ErrorHandler handler = new ErrorHandler(e);
83         Console.WriteLine(handler.GetMessageError());
84         return status.GetRequestStatusError(handler.type);
85     }
86 }
87
```

Ensuite pour permettre à l'utilisateur de se connecter, on va récupérer le mot de passe de l'utilisateur grâce à l'email qu'il va renseigner. Étant donné que l'email est forcément unique à chaque utilisateur, on peut récupérer le mot de passe grâce à l'email.

Ensuite grâce à la fonction `Verify()` de `BCrypt.Net.BCrypt`, on va pouvoir comparer le mot de passe hashé en base et le mot de passe donné par l'utilisateur. Si la fonction `Verify()` return **true**, alors la connexion a aboutie.

```

130 string query = "SELECT id_m, role, password_m FROM medecin WHERE login_m = @login";
131
132 int idMedecin = 0;
133 string role = "";
134 string hashedPassword = "";
135
136 using (MySQLCommand command = new MySQLCommand(query, conn))
137 {
138     command.Parameters.AddWithValue("@login", medecin.Username.ToLower());
139
140     idMedecin = Convert.ToInt32(command.ExecuteScalar());
141     MySQLDataReader reader = command.ExecuteReader();
142     while (reader.Read())
143     {
144         idMedecin = reader.GetInt32(0);
145         role = reader.GetString(1);
146         hashedPassword = reader.GetString(2);
147     }
148     conn.Close();
149     if (!hashedPassword.Equals("") && BCrypt.Net.BCrypt.Verify(medecin.Password, hashedPassword))
150     {
151         Global.UserId = idMedecin;
152         Global.UserRole = role;
153         return status.GetRequestStatusNoError(1);
154     }
155     else
156     {
157         return status.GetRequestStatusNoError(0);
158     }

```

Ajout de bloc try/catch

Les blocs try/catch permettent une meilleure gestion des erreurs dans notre application. Ils permettent d'empêcher que notre application crash dès le lancement mais également de gérer les erreurs liées aux requêtes SQL.

<https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html>

Création des classes ErrorHandler et RequestStatus

ErrorHandler va me permettre de gérer l'erreur capturer par le catch lorsque l'on va faire une requête SQL.

Son principal attribut est **type**.

- type : de type **typeError** et va stocker le type d'erreur que l'on souhaite gérer
- **typeError** est un enum qui va nous permettre de gérer les messages que l'on souhaite afficher dans un Console.WriteLine() ou dans un MessageBox.

```

public enum typeError
{
    NoConnection,
    InvalidCredentials,
    UnknownError,
    CannotDelete,
    OnlyGroupBy,
    NoError
}

```


RequestStatus va me permettre de stocker le résultat de ma requête afin de pouvoir le traiter lorsque l'utilisateur va interagir avec mon application.

Ses attributs sont **success** et **typeError**.

- success : booléen qui me permet de savoir si l'opération a bien été effectuée
- typeError : de type `typeError`, il va me permettre d'identifier si l'erreur a été récupérée dans un catch.

`typeError` est utilisé lorsque je veux savoir si l'opération n'a pas abouti à cause d'un problème de connexion à la base ou si c'est parce que `MySQL` m'empêche de faire des opérations comme supprimer des médecins qui sont assimilés à des ordonnances par exemple.

Exemple de code (cf. `MedecinController.cs`) :

```
234 public RequestStatus DeleteMedecin(int id)
235 {
236     RequestStatus status = new RequestStatus();
237     try
238     {
239         using (MySQLConnection conn = new MySQLConnection(connectionString))
240         {
241             conn.Open();
242             string query = "DELETE FROM medecin WHERE id_m = @id";
243
244             using (MySQLCommand command = new MySQLCommand(query, conn))
245             {
246                 command.Parameters.AddWithValue("@id", id);
247
248                 int result = command.ExecuteNonQuery();
249                 conn.Close();
250                 return status.GetRequestStatusNoError(result);
251             }
252         }
253     } catch (MySQLException e)
254     {
255         ErrorHandler handler = new ErrorHandler(e);
256         Console.WriteLine(handler.GetMessageError());
257         return status.GetRequestStatusError(handler.type);
258     }
259 }
260 }
```

Utilisation dans le formulaire du détail du médecin :

```
46 1 référence
47 private void supprBtn_Click(object sender, EventArgs e)
48 {
49     Console.WriteLine("Supress Médecin ", selectedMedecin.Id);
50     RequestStatus status = controller.DeleteMedecin(selectedMedecin.Id);
51
52     if (status.success)
53     {
54         MessageBox.Show("Suppression du médecin réussi");
55     } else if (!status.success && status.typeError == typeError.CannotDelete)
56     {
57         MessageBox.Show("Vous ne pouvez pas supprimer ce médecin");
58     } else
59     {
60         MessageBox.Show("Erreur lors de la suppression");
61     }
62     this.Close();
63 }
64 }
```

On utilise la variable *status* pour gérer les cas de base, si l'opération a été exécutée ou non, mais on peut également utiliser le *type* dans la variable *status* pour gérer d'autres cas spécifiques et qu'on souhaite prévenir l'utilisateur de pourquoi l'action n'a pas pu aboutir.

Requêtes paramétrées

Les requêtes paramétrées permettent d'éviter les injections SQL.
Pour faire des requêtes paramétrées avec C#, il est simple.

Exemple de requête SQL paramétrée (cf.MedicamentController.cs)

```
string query = "INSERT INTO medicament (libelle_med, contre_indication) " +
    "VALUES (@libelle, @contre_indication)";

using (MySQLCommand command = new MySQLCommand(query, conn))
{
    command.Parameters.AddWithValue("@libelle", med.Libelle);
    command.Parameters.AddWithValue("@contre_indication", med.ContreIndication);
    int result = command.ExecuteNonQuery();
    conn.Close();
    return status.GetRequestStatusNoError(result);
}
```

Ici on remarque que @libelle ou encore @contre_indication sert de placeholder dans le string query pour les informations données par l'utilisateur.

Sanitarisation des entrées utilisateurs

Après chaque formulaire d'ajout remplis, on va réinitialiser les champs.

On va également faire attention à ce que l'utilisateur n'envoie pas des données incomplètes en bases. Si c'est le cas, on prévient l'utilisateur avec des textes de warning.

Comme par exemple, si on ajoute un médecin mais qu'il manque des informations, le texte d'avertissement apparaîtra.

Gestion des médecins - GSB

Ajout de médecin

Nom

Prénom

Date de naissance

19 January 2024

Email

Mot de passe

Role

USER

Ajouter

Rechercher

	Id	Nom	Prénom	Date de naissance	Email	Mot de passe	Rôle
►	1	ITH	Kagnana	2003-03-17	ikagnana@g...		ADMIN
	3	Simple	User	1946-07-14	simpleuser@...		USER

Faites attention, certains champs sont mal remplis

Stockage des informations de connexion

Dans app.config, on va ajouter la connectionString qui va permettre à notre application d'être connectée à notre base de données. Cette connectionString pourra être utilisée dans toute l'application.

```
string connectionString = Configuration.ConnectionString["localhost"].ConnectionString;
```