

ITH KAGNANA
BTS SIO 2

eMeds

APPLICATION LOURDE C#
WINFORMS



OCTOBRE 2023

SOMMAIRE

01 Contexte “Laboratoire GSB”

- A.** Description du laboratoire
- B.** Domaine d'étude

02 Les objectifs et les besoins

- A.** Professionnel
- B.** Personnel

03 Analyse fonctionnelle

- A.** Conception
- B.** Tester
- C.** Déploiement
- D.** Présentation de l'application

04 Annexes

CONTEXTE GSB

Description du laboratoire

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.

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.

LES OBJECTIFS & BESOINS

Professionnels

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.

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

Conception – Architecture

Le projet comporte des fichiers avec tous des rôles différents.

Le C# est un langage de programmation objet. Par conséquent, il est important d'avoir des classes qui représente nos objets en base.

Les classes, qui servent d'objet de référence :

- Patient.cs
- ObjetPatient.cs (regroupe les antécédents et les allergies)
- Ordonnance.cs
- Medecin.cs
- Medicament.cs

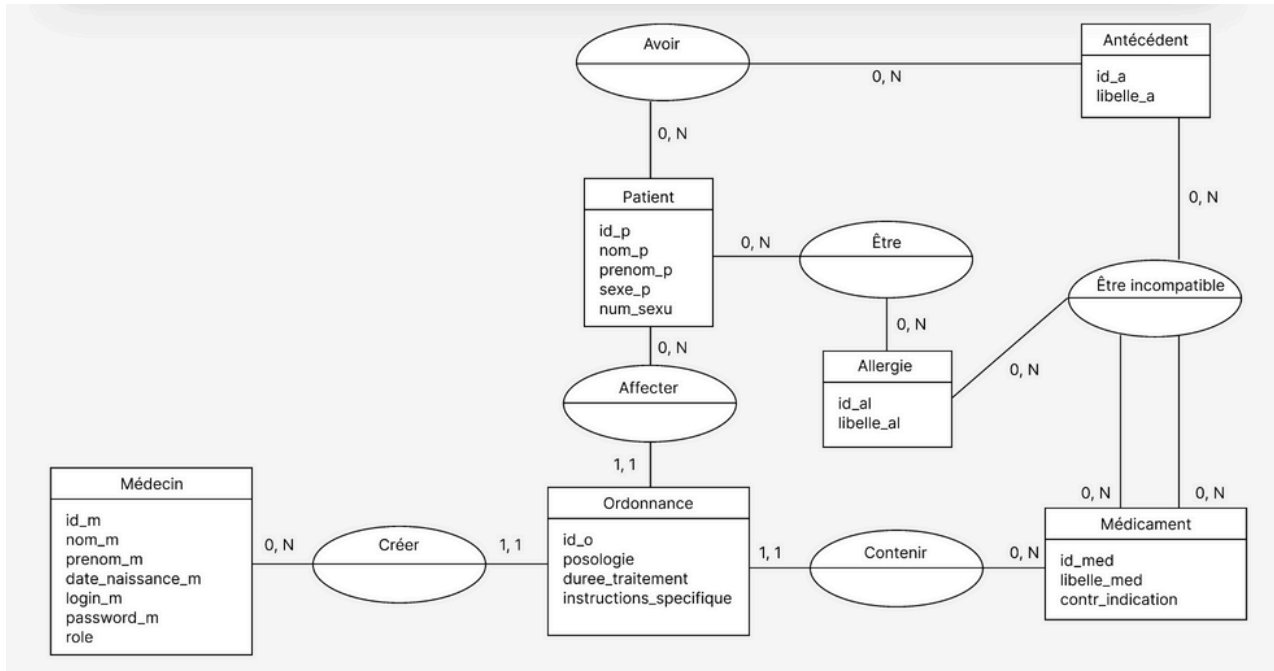
Nous avons également des controllers qui permettent de faire le lien entre le formulaire et la base de donnée.

Ce dernier contient toutes les fonctions relatives à une table en base de données.

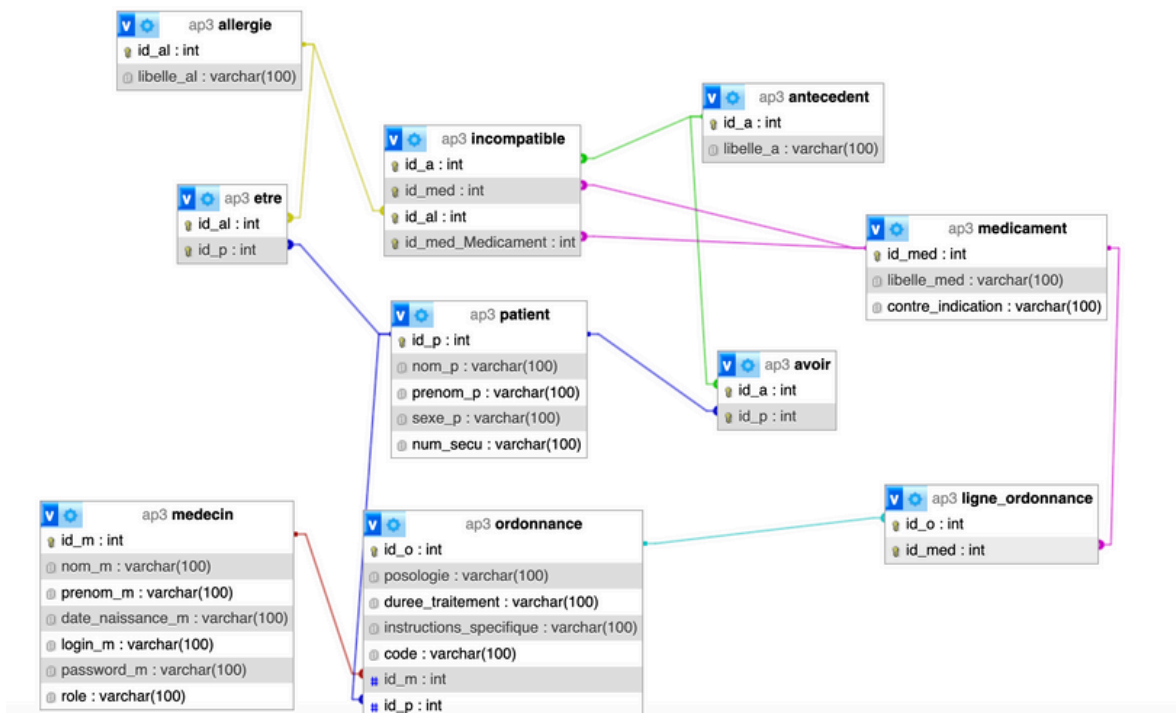
Par exemple, *OrdonnanceController.cs* qui permet a toutes les méthodes pour manipuler la table *ordonnance* dans notre base de données

Ensuite nous avons les formulaires qui s'occupent d'afficher l'information et de traiter les entrées utilisateurs. Il est lié au controller.

Conception – Base de donnée



MCD DE NOTRE APPLICATION



TABLES ET RELATIONS

Conception – Base de donnée

Modifications apportées à la base de données

Lors du développement de l'application, certains changements sont apportés à la base pour correspondre aux besoins du projet.

On ajoute le champ num_secu dans la table patient 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.

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

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

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

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

```
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)  
)
```

Conception – Sécurité

Hachage de mot de passe

Il est important de hacher le mot de passe pour éviter qu'il soit visible en clair dans la base de donnée.

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 NuGet : 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.

Exemple de code (cf. MedecinController.cs):

```
91 // create connection to the db to make query
92 using (MySQLConnection conn = new MySqlConnection(connectionString))
93 {
94     conn.Open();
95     string query = "INSERT INTO medecin (nom_m, prenom_m, login_m, password_m, date_naissance_m, role) " +
96         "VALUES (@lastname, @firstname, @login, @password, @birthdate, @role)";
97
98     using (MySQLCommand command = new MySQLCommand(query, conn))
99     {
100         // hash password
101         string hashedPassword = BCrypt.Net.BCrypt.HashPassword(medecin.Password);
102         command.Parameters.AddWithValue("@lastname", medecin.Nom);
103         command.Parameters.AddWithValue("@firstname", medecin.Prenom);
104         command.Parameters.AddWithValue("@login", medecin.Email);
105         command.Parameters.AddWithValue("@password", hashedPassword);
106         command.Parameters.AddWithValue("@birthdate", medecin.DateNaissance);
107         command.Parameters.AddWithValue("@role", medecin.Role);
108         int result = command.ExecuteNonQuery();
109         conn.Close();
110         return status.GetRequestStatusNoError(result);
111     }
112 }
```

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.

Exemple de code (cf. MedecinController.cs méthode Login):

```
if (!hashedPassword.Equals("") && BCrypt.Net.BCrypt.Verify(medecin.Password, hashedPassword))
{
    Global.UserId = idMedecin;
    Global.UserRole = role;
    return status.GetRequestStatusNoError(1);
}
```


Conception – Sécurité

Gestion d'erreur

Il est important de gérer les erreurs que peuvent nous renvoyer notre SGBD lorsqu'une requête a échoué. Ainsi nous pouvons prévenir les utilisateurs que l'opération n'a pas abouti.

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.

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`.

```
10      // enum with type error
11      public enum typeError
12      {
13          NoConnection,
14          InvalidCredentials,
15          UnknownError,
16          CannotDelete,
17          OnlyGroupBy,
18          NoError
19      }
```

RequestStatus va me permettre de stocker le résultat de ma requête afin de pouvoir le traiter lorsque l'utilisateur va interagir avec l'application. Ses attributs sont **success** et **typeError**.

- **success** : booléen qui me permet de savoir si l'opération a bien été effectué
- **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`):

```
268     public RequestStatus DeleteMedecin(int id)
269     {
270         RequestStatus status = new RequestStatus();
271         try
272         {
273             using (MySqlConnection conn = new MySqlConnection(connectionString))
274             {
275                 conn.Open();
276                 string query = "DELETE FROM medecin WHERE id_m = @id";
277
278                 using (MySqlCommand command = new MySqlCommand(query, conn))
279                 {
280                     command.Parameters.AddWithValue("@id", id);
281
282                     int result = command.ExecuteNonQuery();
283                     conn.Close();
284                     return status.GetRequestStatusNoError(result);
285                 }
286             }
287         } catch (MySqlException e)
288         {
289             ErrorHandler handler = new ErrorHandler(e);
290             Console.WriteLine(handler.GetMessageError());
291             return status.GetRequestStatusError(handler.type);
292         }
293     }
294 }
295 }
```

Utilisation dans le formulaire du détail du médecin (cf DetailsMedecin.cs)

On utilise la variable **status** pour gérer les cas de base.

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

Mais on peut également utiliser le type dans la variable status pour gérer d'autres cas spécifiques.

Conception – Sécurité

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)

```
// create connection to the db to make query
using (MySQLConnection conn = new MySQLConnection(connectionString))
{
    conn.Open();
    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.

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;
```

Récupération des informations dans un controller :

```
// get value for localhost in App.config  
private string connectionString = ConfigurationManager.ConnectionStrings["localhost"].ConnectionString;
```

Utilisation dans une méthode du controller :

```
using (MySQLConnection conn = new MySQLConnection(connectionString))  
{  
    conn.Open();  
    string query = "DELETE FROM medecin WHERE id_m = @id";  
  
    using (MySQLCommand command = new MySQLCommand(@query, conn))  
    {  
        command.Parameters.AddWithValue("@id", id);  
  
        int result = command.ExecuteNonQuery();  
        conn.Close();  
        return status.GetRequestStatusNoError(result);  
    }  
}
```

Gestions des rôles et des utilisateurs

On a différents rôles :

- "ADMIN"
- "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 :

- consulter les médicaments mais 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

Pour pouvoir tester dans l'application le rôle de l'utilisateur et ainsi lui permettre d'accéder à certaines fonctionnalités ou non, on va instancier une classe Global.cs qui aura des variables statiques.

On pourra ensuite accéder à ces données dans les autres classes

```
86 // open the new form Details
87 private void OpenForm(object sender, DataGridViewCellEventArgs e)
88 {
89     string role = Global.UserRole;
90     if (role == "ADMIN")
91     {
92         if (e.RowIndex >= 0)
93         {
94             DataGridViewRow selectedRow = dataGridView1.Rows[e.RowIndex];
95             if (selectedRow != null)
96             {
97                 Medicament selected = selectedRow.DataBoundItem as Medicament;
98                 DetailsMedicament details = new DetailsMedicament(selected);
99                 // add closing event to the form
100                 details.FormClosing += new FormClosingEventHandler(this.DetailsClosing);
101                 details.Show();
102             }
103         }
104     } else
105     {
106         MessageBox.Show("Vous n'avez pas les droits pour modifier le médicament");
107     }
108 }
109
110 }
```

Ici on veut uniquement permettre à l'utilisateur avec le rôle admin de voir les détails d'un médicament. Les détails des médicaments

Tester

Comment l'installer

Pour installer le projet, dans le terminal à l'emplacement voulu :
git clone <https://github.com/iKagnana/AP3-eMEDS.git>

Ensuite, dans le gestionnaire de base de données de votre choix, importez la base de données du projet grâce au script : ap3-emeds_2024-01-19.sql

Ne pas oublier de changer dans l'App.config la connectionString.

```
<configuration>
  <connectionStrings>
    <add
      connectionString="Server=localhost;Database=db;UserID=user;Password=
      user" name="localhost" providerName="MySQL.Data.MySqlClient" />
    </connectionStrings>
  </configuration>
```

Bien faire attention de remplacer avec les bonnes données

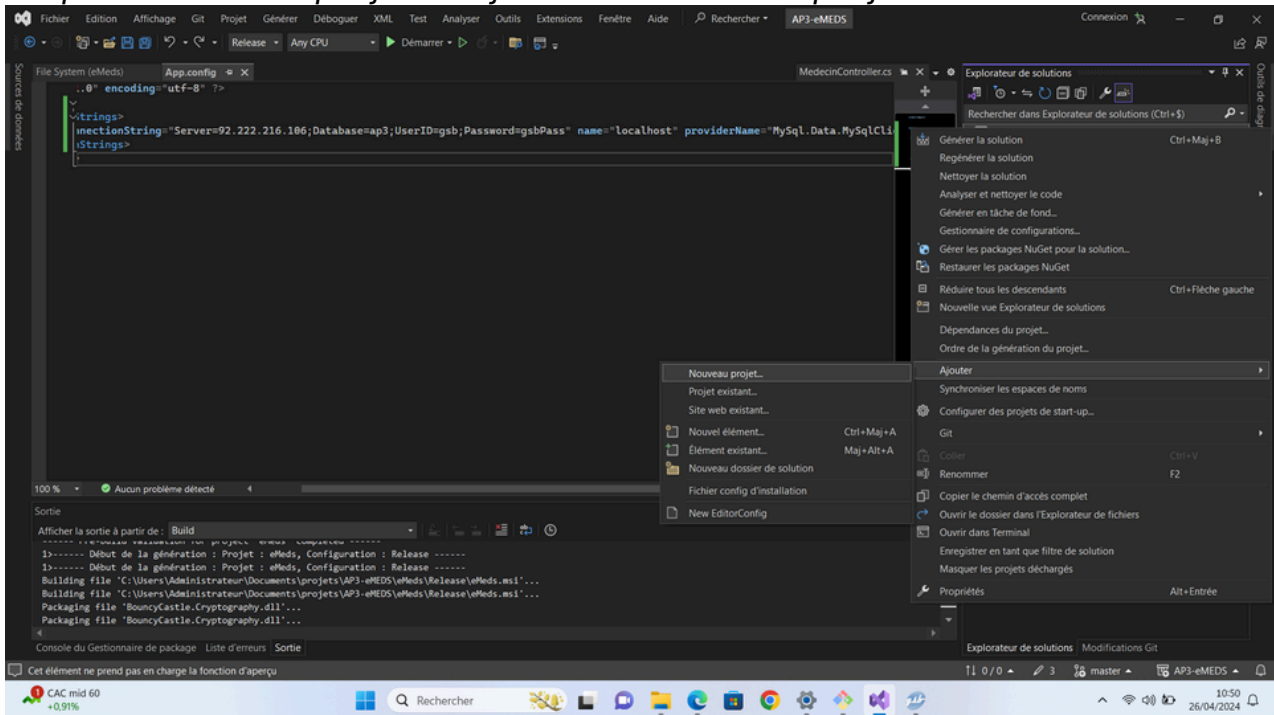
Différents comptes pour tester

	Administrateur	Employé
Email	ikagnana@gmail.com	simpleuser@gmail.com
Mot de passe	admin	simpleUser

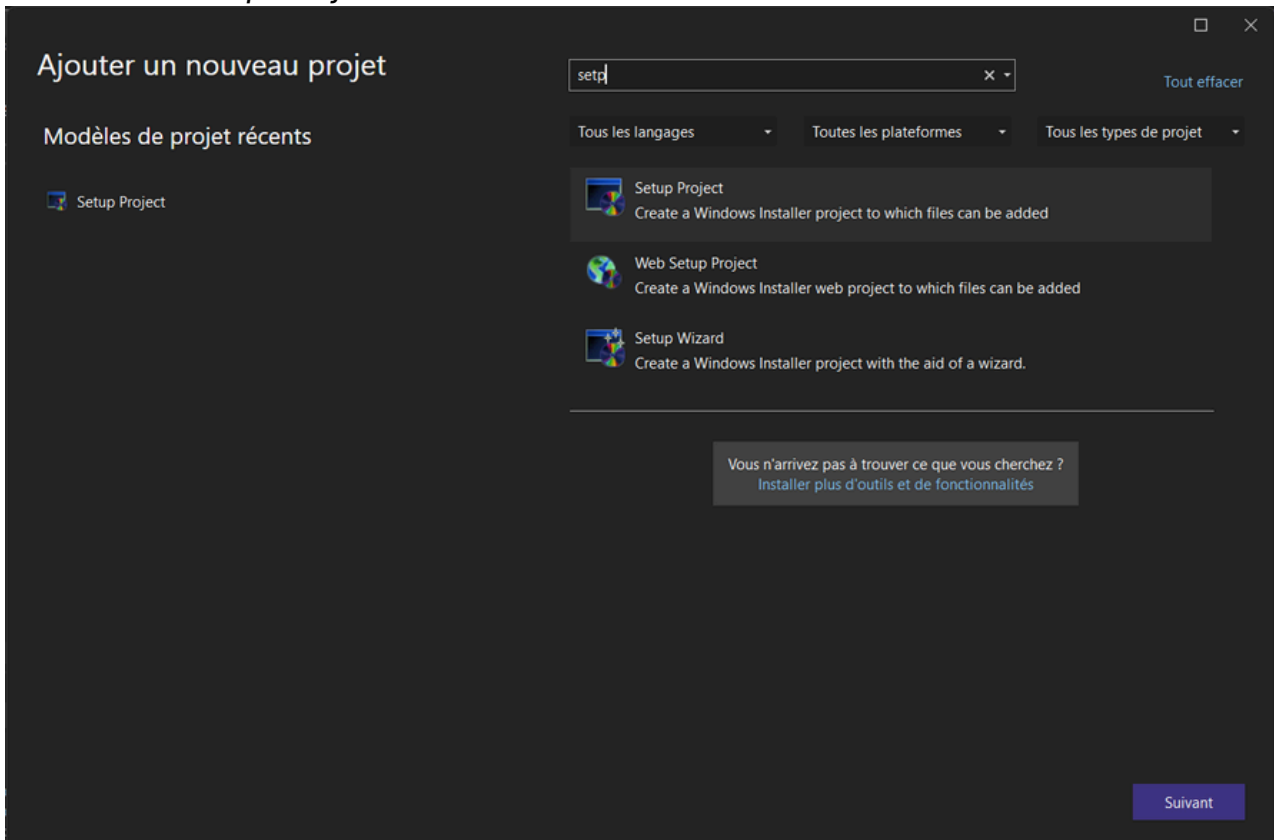
Déploiement

Pour déployer le projet, il faut installer les extensions **Microsoft Visual Studio Installer** et **Package Installer**. Il faut ensuite suivre ces étapes :

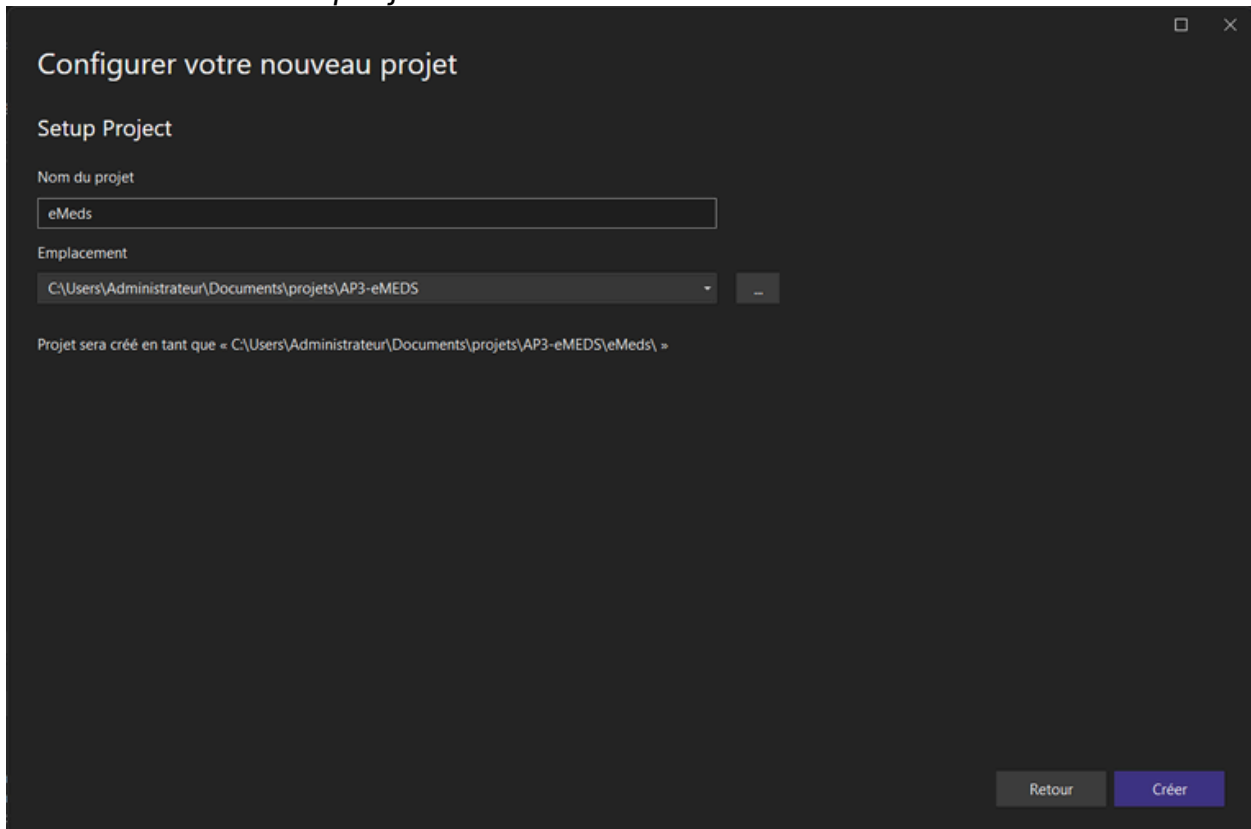
Clique droit sur le projet et ajouter un nouveau projet.



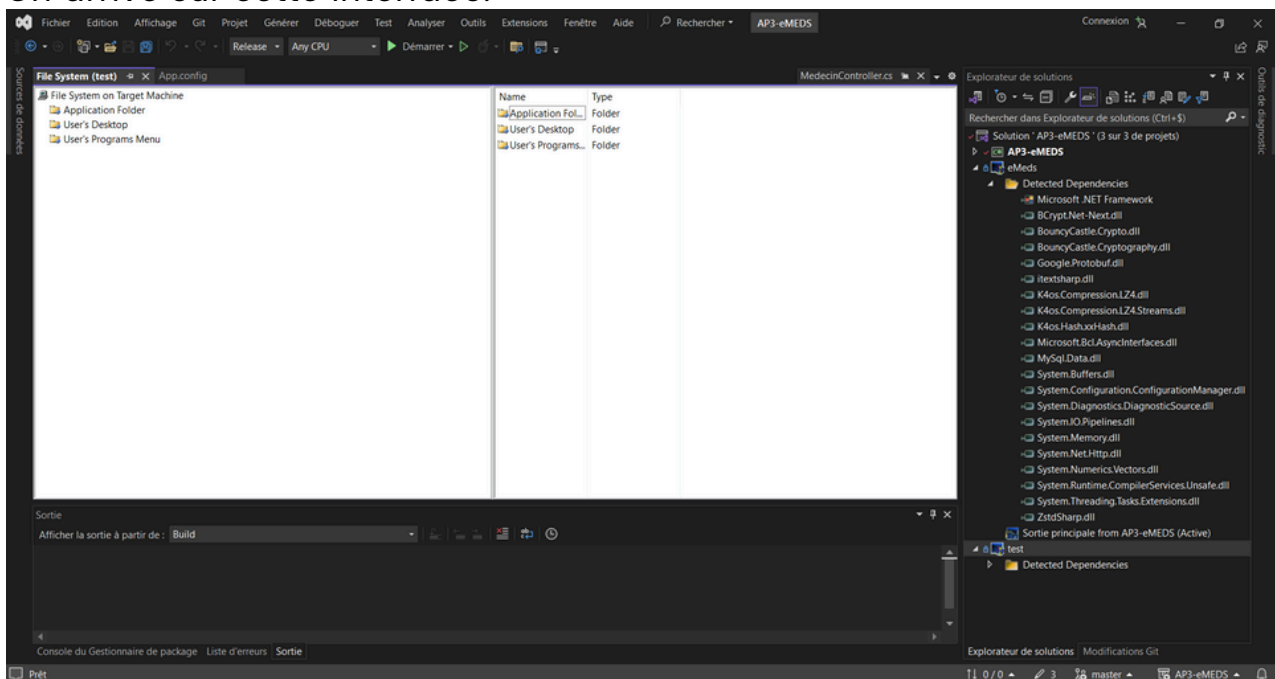
Chercher Setup Project



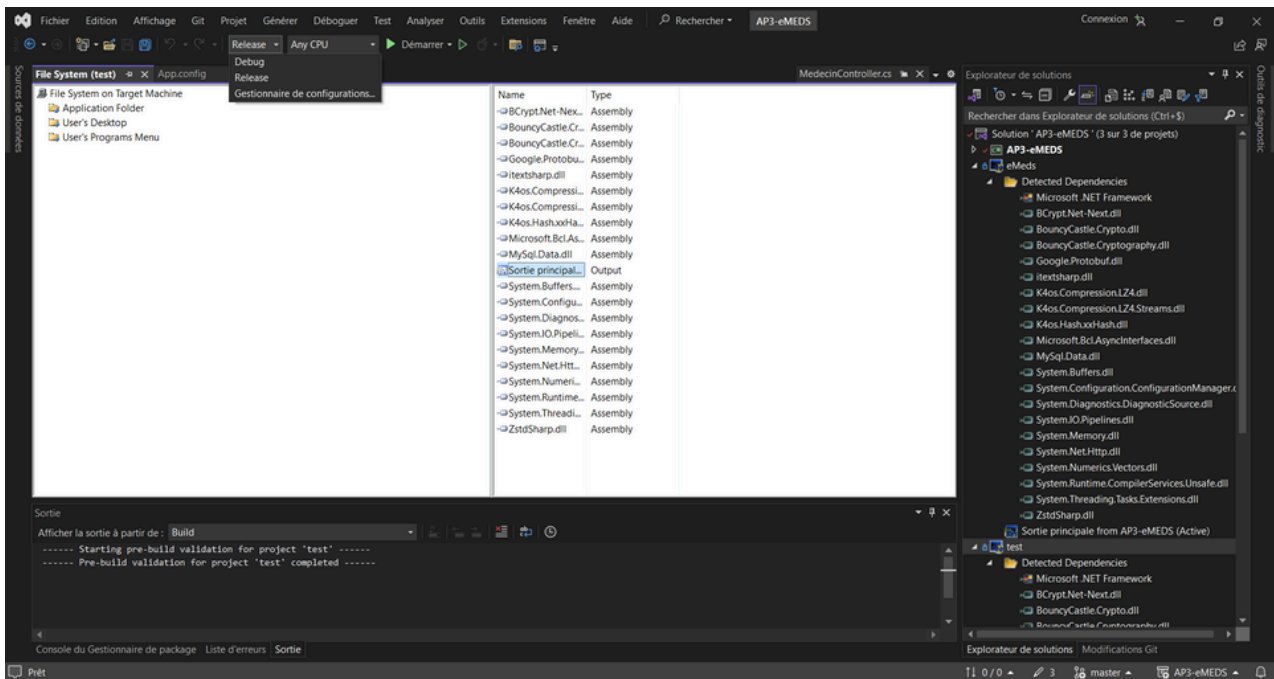
Donner un nom au projet.



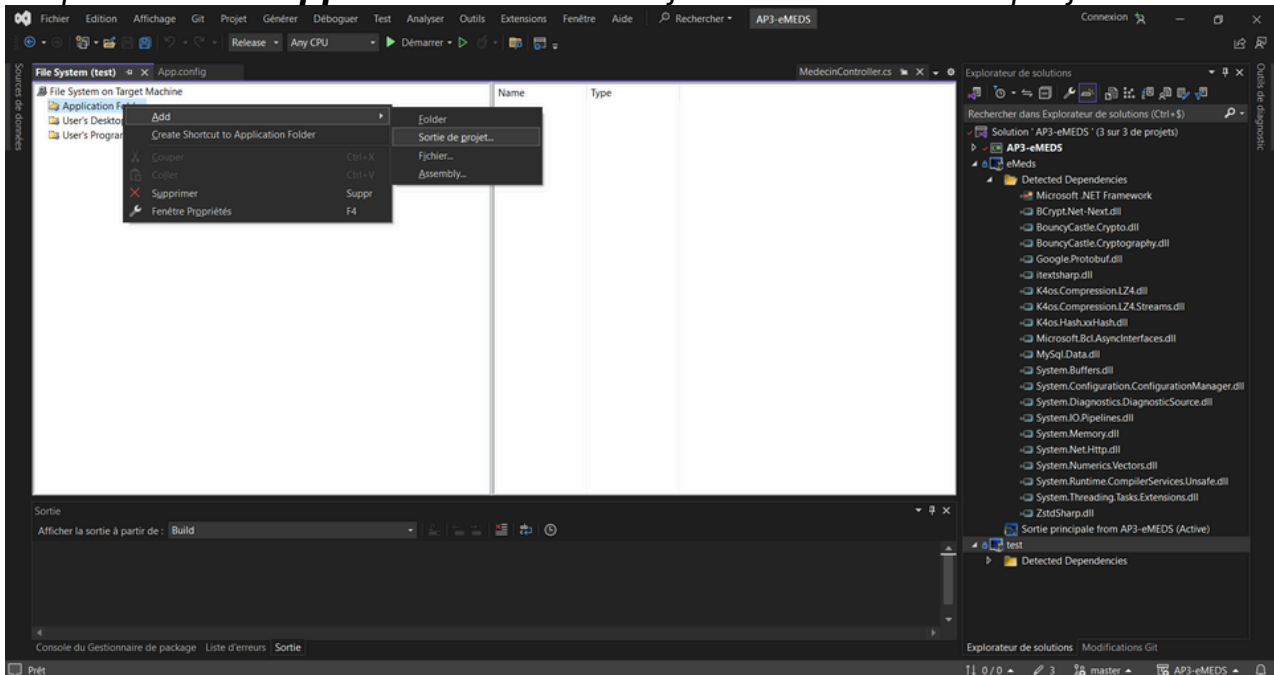
On arrive sur cette interface.



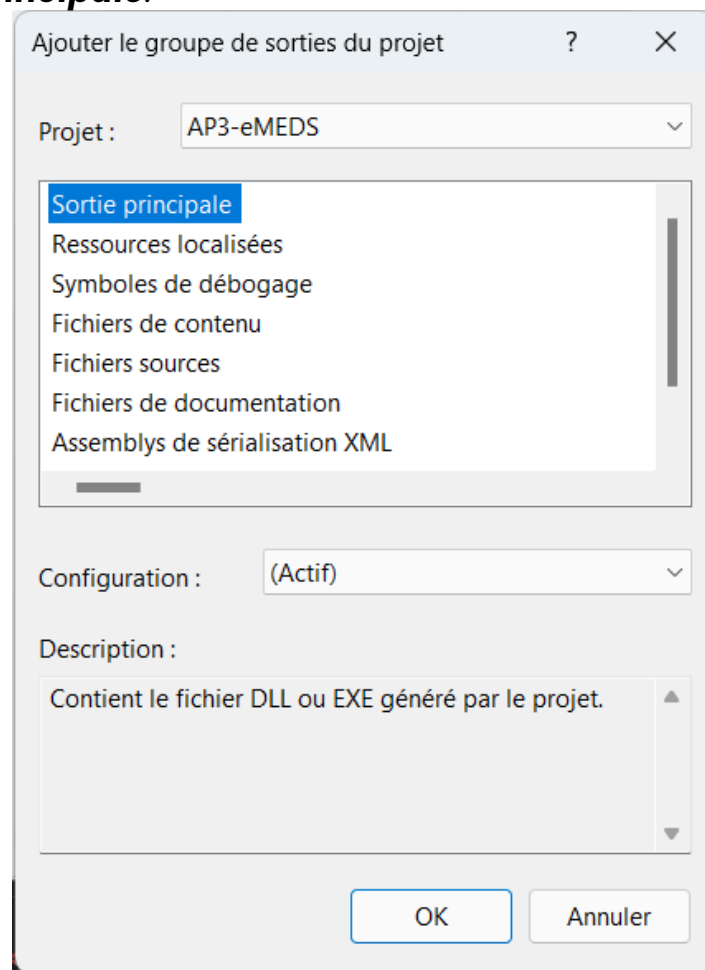
Il faut bien attention à choisir la version Release.



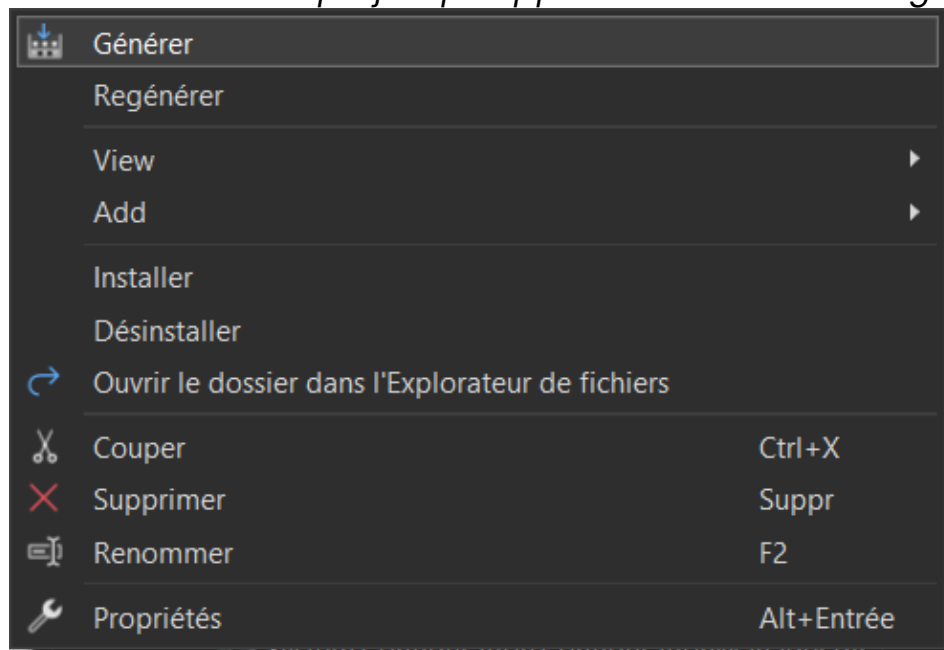
*Clique droit sur **Application Folder** et ajouter une sortie de projet.*



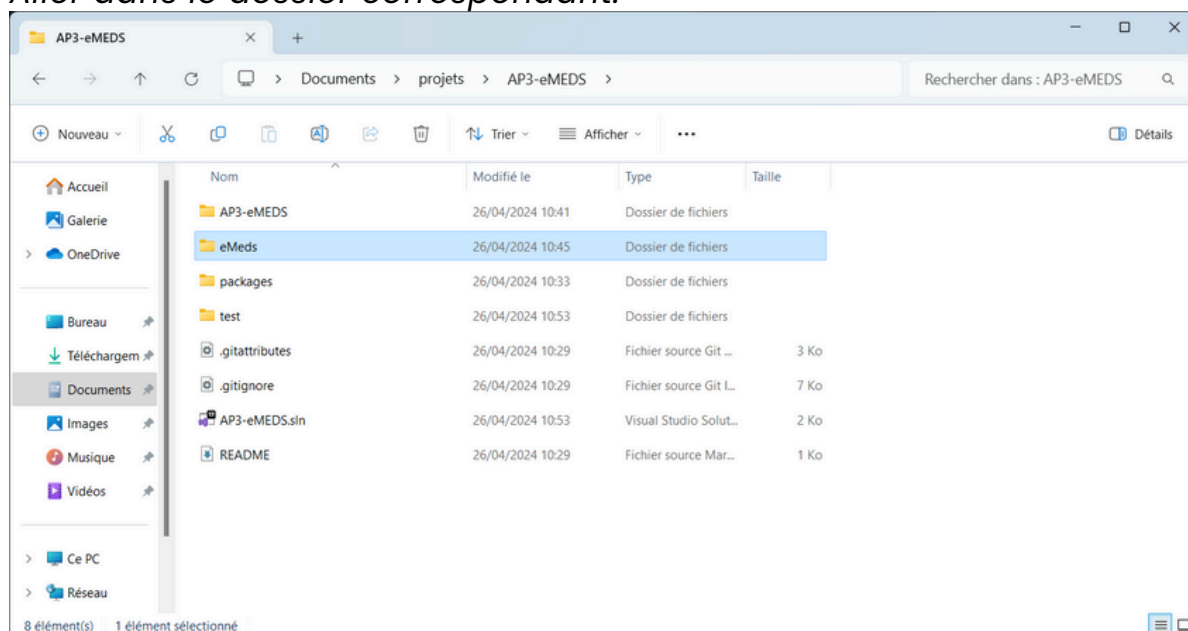
Choisir **Sortie principale**.



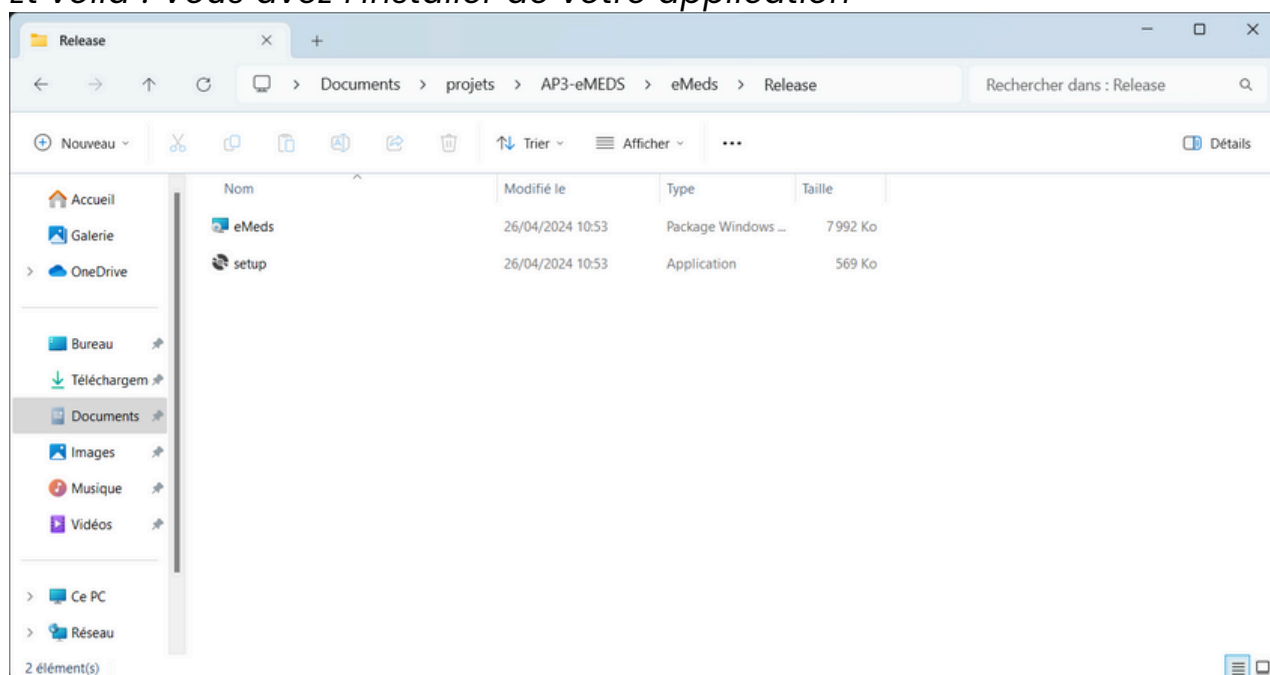
Clique droit sur le nouveau projet qui apparaît dans la liste et générer.



Aller dans le dossier correspondant.



Et voilà ! Vous avez l'installer de votre application



Dans le cadre de mon projet, ma version déployée se trouve sur un ordinateur de prêt de l'école dans :

C:\Users\Administrateur\Documents\projets

La base de données se situe elle sur un VPS personnel.

Présentation de l'application

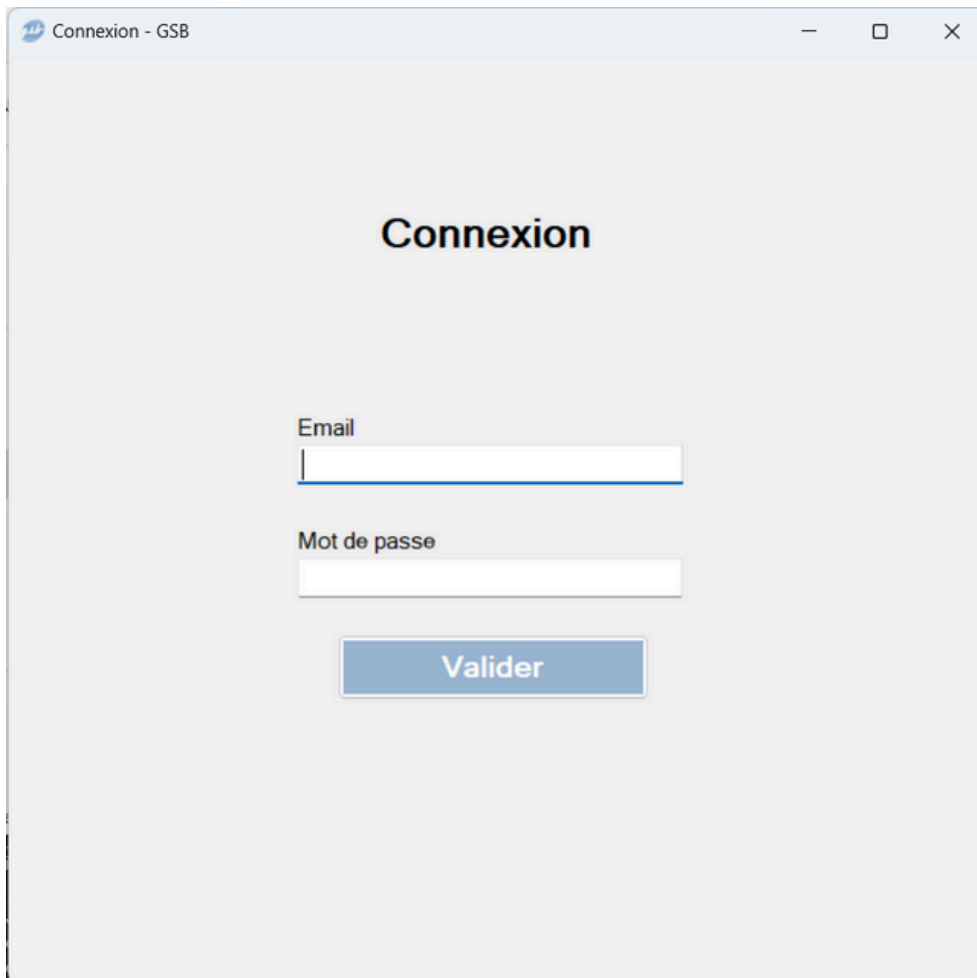
L'affichage des vues peut différer en fonction du rôle de l'utilisateur.

Par exemple, un simple médecin ne peut pas accéder au menu des médecin et des allergies et antécédents.

Il ne peut donc pas consulter les médecins ni les gérer. Il ne peut également pas consulter les différentes allergies et antécédents ni les gérer.

Nous allons nous mettre à la place de l'administrateur pour montrer toutes les fonctionnalités et pages de l'application.

Page de connexion



The screenshot shows a web browser window titled "Connexion - GSB". The page has a light gray background and the word "Connexion" centered at the top. Below it, there are two input fields: "Email" and "Mot de passe". The "Email" field has a blue underline. Below the "Mot de passe" field is a blue button labeled "Valider".

Connexion - GSB

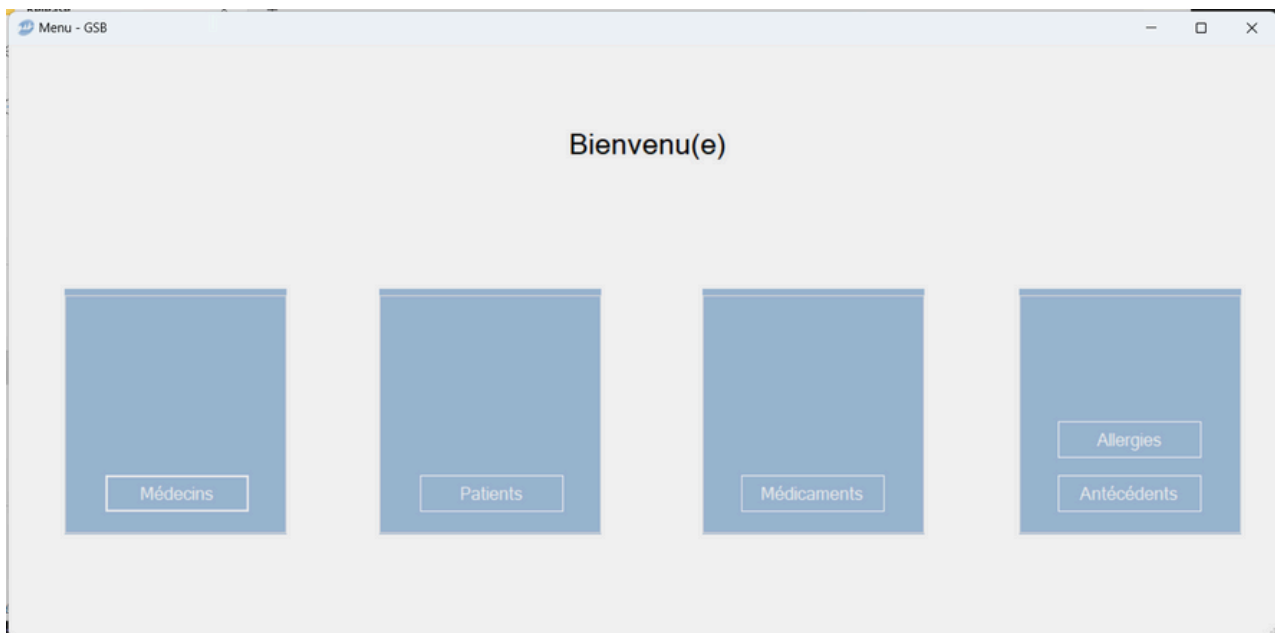
Connexion

Email

Mot de passe

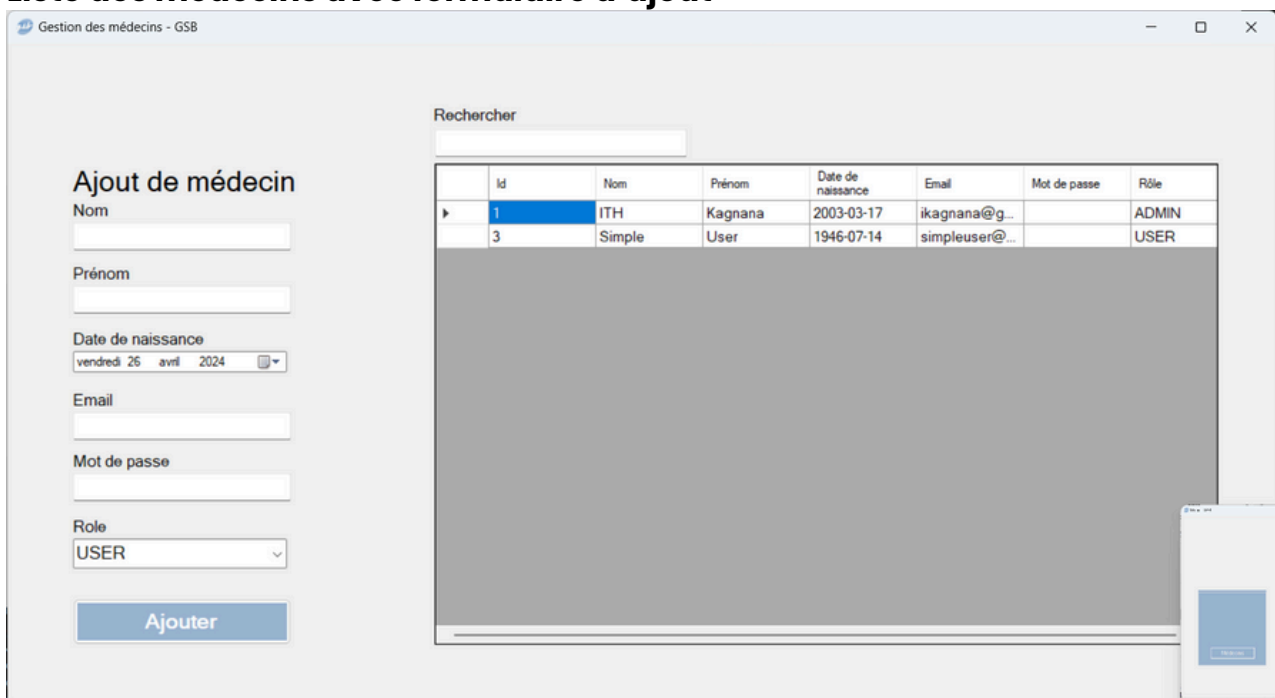
Valider

Page d'accueil de l'application



LA PARTIE MÉDECIN

Liste des médecins avec formulaire d'ajout



Si on double clique sur un médecin, on obtient la fenêtre avec les détails du médecin.

Formulaire avec les détails du médecin

Détails du médecin - GSB

Détails du médecin

Nom	ITH	Prénom	Kagnana
Date de naissance	lundi 17 mars 2003	Email	ikagnana@gmail.com
Mot de passe	Role: ADMIN		

Modifier le mot de passe

Modifier

Supprimer le médecin

Pour modifier le mot de passe de l'utilisateur, il suffit de cliquer sur le bouton **"Modifier le mot de passe"**.

Formulaire pour changer le mot de passe

Changement de mot de passe - GSB

Nouveau mot de passe

Modifier le mot de passe

LA PARTIE PATIENT

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

Liste avec tous les patients de la base de données

Rechercher

Liste des patients

Accès ordonnance	Id	Nom	Prénom	Sexe	Numéro de Sécu	Allergies	Antécédents
<input type="text"/>	1	Doe	John	M	123456789123412	4	4
	2	Martin	Thomas	M	154392345434566	0	0
	3	Laroché	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
	9	Demo	Demo	M	123144235125432	1	1

Possibilité de faire une recherche par nom et prénom

Pour ajouter un nouveau patient, il suffit de cliquer sur le petit bouton avec le signe “+” dessus.

L'ajout se fait en 2 étapes.

La première étape permet d'enregistrer les informations essentielles du patient.

Détails d'une commande

Ajout du patient étape 1 - GSB

Ajout de patient

Nom

Prénom

Numéro de Sécurité Sociale

☐ Homme
☐ Femme

Deuxième partie de l'ajout du patient

Ajout d'un patient étape 2 - GSB

Allergie(s)

Allergie au pollen

+

Id	Libelle
----	---------

Antécédent(s)

Ulcères gastriques actifs

+

Id	Libelle
----	---------

Passer

Pour supprimer une allergie ou un antécédent, il suffit de cliquer sur l'élément présent dans la liste et une fenêtre va s'ouvrir pour demander confirmation à l'utilisateur.

Confirmation de suppression - GSB

Voulez vous supprimer ?
l'antécédent Insuffisance rénale ?

Annuler

Supprimer

Ensuite, si on retourne sur la page avec la liste des patients et que l'on clique sur la cellule "Accès ordonnance". On arrive sur une fenêtre qui regroupe toutes les ordonnances du patient et un formulaire pour lui prescrire

Partie ordonnance du patient

Ajout d'ordonnance - GSB

Martin Thomas M - 154392345434566

Médicament(s)

Posologie

Durée du traitement

Instructions spécifiques

Ajouter

Liste des ordonnances

Générer le pdf	Id	Posologie	Durée	Durée	Instructions spécifiques
----------------	----	-----------	-------	-------	--------------------------

On peut générer un pdf pour chaque ordonnance de la liste

Un warning apparait si un médicament sélectionné est incompatible

Ajout d'ordonnance - GSB

Doe John M - 123456789123412

Médicament(s)

Aspirine

Id

Libelle

Aspirine

Posologie

Durée du traitement

Instructions spécifiques

Ajouter

Attention certains médicaments sont incompatible avec le patient

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	8c9f1510-bd9f-4...
	3		4 jour		6cc0c9b0-66ec-...

On estime que le médecin est celui qui prends la décision finale. Lorsqu'il y a une incompatibilité, on choisi de ne pas le bloquer, uniquement de l'avertir.

Messages d'avertissement

Voulez-vous continuer ?

Attention ce médicament est incompatible avec ce médicament :Aspirine

YesNo

Voulez-vous continuer ?

Attention certains médicaments sont incompatibles avec le patient. Il a des incompatibilités avec : , Allergie à l'aspirine

YesNo

Il est possible de consulter le détail d'une ordonnance en cliquant dessus dans la liste à droite présente dans la partie ordonnance du patient.

Détails d'une ordonnance

Doe John M - 123456789123412

Posologie
1g de chaque

Durée du traitement
4 jour

Instructions spécifiques
Si douleurs persistante, coupler avec du doliprane

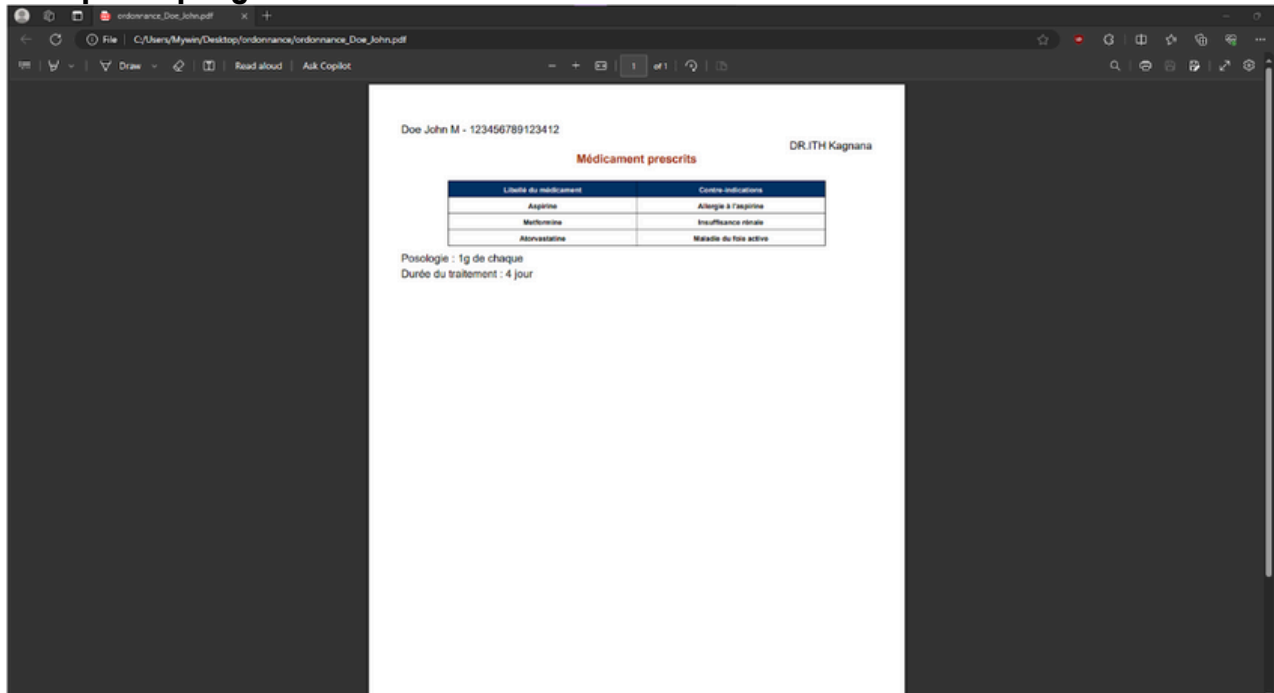
Générer le pdf

Liste de médicaments

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

On a uniquement la possibilité de consulter les informations de l'ordonnance et de gérer le pdf.

Exemple de pdf généré



Pour accéder à la partie détail du patient, il suffit de double cliquer sur les autres cellules correspondant au patient que l'on souhaite consulter.

Partie détails du patient

Détail patient - GSB

Ajout de patient

Nom

Prénom

Numéro de Sécurité Sociale

☒ Homme
☐ Femme

[Valider les modifications](#)

[Supprimer le patient](#)

Allergie(s)
 +

Id	Libelle
----	---------

Antécédent(s)
 +

Id	Libelle
----	---------

Pour supprimer une allergie ou un antécédent, il suffit de cliquer sur l'élément souhaité

LA PARTIE MÉDICAMENT

Formulaire et liste des médicaments

The screenshot shows a web application window titled 'Gestion des médicaments - GSB'. On the left is a form titled 'Ajout de médicament' with fields for 'Libellé' and 'Contre-indications', and an 'Ajouter' button. On the right is a search bar labeled 'Rechercher' and a table of medications.

	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	Fluoxétine	Syndrome sérotoninergique
	15	Diazepam	Dépendance aux substances
	16	Prednisone	Infections fongiques systémiques
	17	Lisinopril	Angioedème
	18	Albuterol	Tachycardie
	19	Gabapentine	Néonatal

A callout box on the right states: 'Possibilité de rechercher par nom'.

Pour accéder au détail du médicament, il suffit de double cliquer sur le médicament souhaité.

Détails d'un médicament

The screenshot shows a web application window titled 'Détails du médicament - GSB'. On the left is a form titled 'Détails du médicament :'. It has fields for 'Libellé' (containing 'Aspirine') and 'Contre-indications' (containing 'Allergie à l'aspirine'). Below these are buttons for 'Valider les modifications' and 'Supprimer le médicament'. On the right are three tables showing incompatibilities.

Allergies incompatibles

	Id	Libellé
▶	4	Allergie à l'aspirine

Antécédents incompatibles

	Id	Libellé
--	----	---------

Médicaments incompatibles

	Id	Libellé
▶	8	Omeprazole
	6	Ciprofloxacine

Ici on peut ajouter les différentes incompatibilités que peut avoir un médicament. Pour supprimer les incompatibilités, il suffit de cliquer sur l'élément que l'on souhaite supprimer.

LA PARTIE ALLERGIES ET ANTÉCÉDENTS

Formulaire d'ajout d'une allergie et liste des différents allergies

	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 achariens

Pour accéder au détail d'une allergie, il suffit de double cliquer sur l'allergie souhaitée.

Détails d'une allergie

Libellé

Allergie à l'aspirine

Modifier

Supprimer le médecin

Fermer

L'affichage et les fonctionnalités pour les allergies et les antécédents sont les mêmes.

Annexes

Lien GITHUB vers le projet

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

Lien FIGMA MDC + Conception + Navigation

<https://www.figma.com/file/w84gh5wpUHJIGi2LYkkUNp/schema-AP3?type=design&node-id=142-2&mode=design&t=hNHidW5WN0mQPaer-0>