



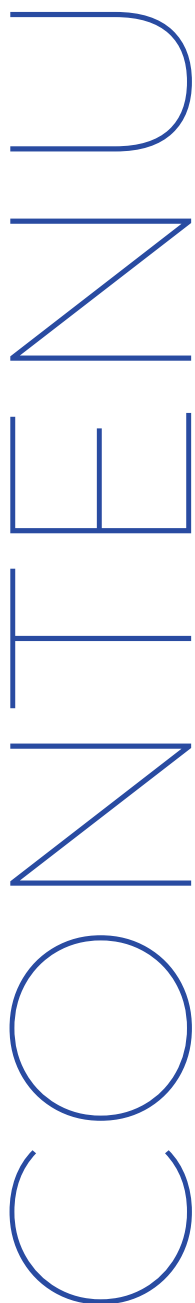
RICHER MARINE

---

GSB - AP 3

2024

# Sommaire



## 01.

Introduction

p.2-3

## 02.

Cahier des charges

p.4

## 03.

Environnement

p.5

## 04.

Base de donnée

p.6-7

## 05.

Les pages et fonctionnalités

p.8-37

## 06.

Manuel utilisateur

p.38-39

## 07.

Conclusion

p.40

# 01.

# NOUVEAU

Le laboratoire Galaxy Swiss Bourdin (GSB) est une entreprise pharmaceutique résultant de la fusion entre le géant américain Galaxy et le conglomerat européen Swiss Bourdin. Cette fusion a créé un leader dans le secteur pharmaceutique, spécialisé notamment dans les maladies virales telles que le SIDA et les hépatites. Le siège administratif de l'entité Galaxy Swiss Bourdin Europe est situé à Paris, en France, tandis que le siège social de la multinationale se trouve à Philadelphie, en Pennsylvanie, aux États-Unis.

Suite à la fusion, l'entreprise a entrepris une réorganisation interne, visant à optimiser son activité en réalisant des économies d'échelle dans la production et la distribution des médicaments. Cela a entraîné une restructuration et une vague de licenciements. GSB compte actuellement 480 visiteurs médicaux en France métropolitaine et 60 dans les départements et territoires d'outre-mer, répartis en 6 secteurs géographiques.

Le système informatique de GSB est bien développé et prend en charge différentes fonctions de l'entreprise. Les fonctions administratives, telles que la gestion des ressources humaines et la comptabilité, sont présentes sur le site parisien. Il y a également un service de recherche en laboratoire, un service juridique et un service communication. La salle serveur, située au 6ème étage du bâtiment, est sécurisée et restreinte aux accès. Les serveurs assurent les fonctions de base du réseau ainsi que les communications internes.

Le réseau informatique est segmenté en VLAN pour fluidifier le trafic. Les données de l'entreprise sont considérées comme stratégiques et sont répliquées quotidiennement aux États-Unis pour assurer une sauvegarde et une redondance maximales. La Direction des Services Informatiques (DSI) joue un rôle important dans la gestion informatique de GSB et participe aux choix stratégiques de l'entreprise.

# 01.

NOTIFICATION PROJET

Sur le site, chaque employé dispose d'un poste fixe relié au système central, et il y a également des stations de travail plus puissantes dans la partie labo-recherche. De plus, de nombreux ordinateurs portables sont utilisés par le personnel de direction, le service informatique et les services commerciaux. Les visiteurs médicaux reçoivent une indemnité ou une dotation en équipement informatique, selon les politiques de Swiss-Bourdin et Galaxy.

Chaque employé de l'entreprise a une adresse de messagerie de la forme "[nomUtilisateur@swiss-galaxy.com](mailto:nomUtilisateur@swiss-galaxy.com)". Les anciennes adresses des laboratoires fusionnés ont été définitivement fermées au 1er janvier 2011.

Responsabilité :

Le laboratoire désire mettre à disposition une application mobile leur permettant de suivre leur traitement médicamenteux. Les utilisateurs peuvent programmer des rappels pour prendre leur médicament à temps et consulter leur historique médical.

Les équipes du service développement auront notamment à produire puis à fournir les éléments applicatifs permettant :

Ici, nous allons créer une plateforme pour les médecins. Elle prend en compte les patients, les médicaments et les ordonnances.

Lien github du projet :



# 02. CAHIER DES CHARGES

01

## Identification des Utilisateurs

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

02

## 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.

03

## Création et Gestion des Ordonnances

- Les médecins doivent pouvoir créer 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
- Lors de la validation de l'ordonnance, si un médicament est incompatible avec un antécédent ou une allergie du patient, une alerte doit prévenir le médecin.

04

## 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.

## 03. ENVIRONNEMENT

Le projet est développé sur un système d'exploitation windows car on peut y développer des applications en Windows Form

### OUTILS UTILISÉS



Serveur web local qui permet de gérer la base en local et d'y avoir accès.



Logiciel permettant de gérer une base de données.



Langage de programmation orienté objet créé par microsoft



IDE qui permet la programmation en C#.



framework

# 04.BASE DE DONNÉE

Il s'agit d'une base en SQL.

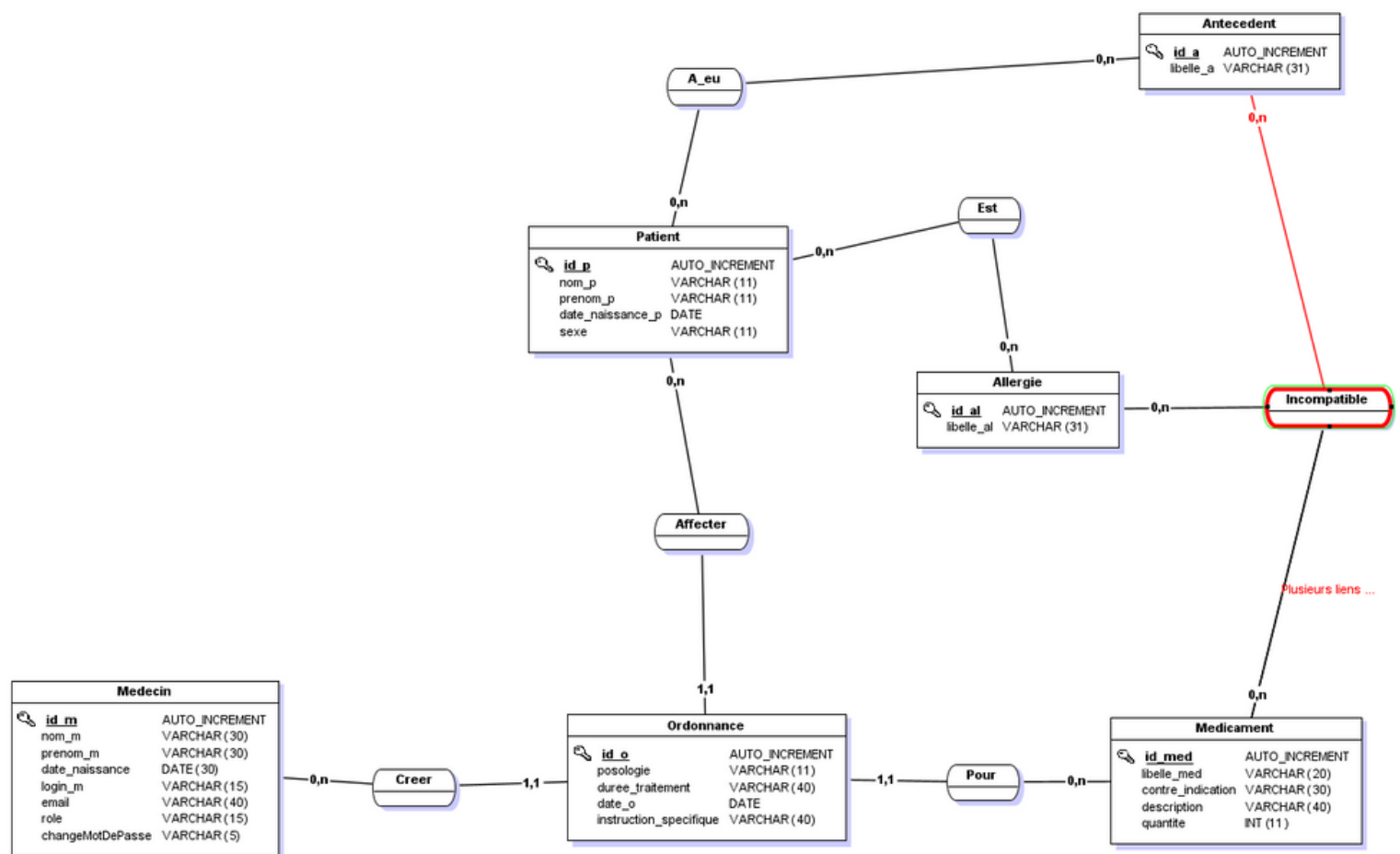
Le schéma de base de données est créé à partir du cahier des charges.

On va retrouver le schéma MCD et MLD ci-dessous. Pour les voir en détail, ils se trouvent dans le dossier du projet.

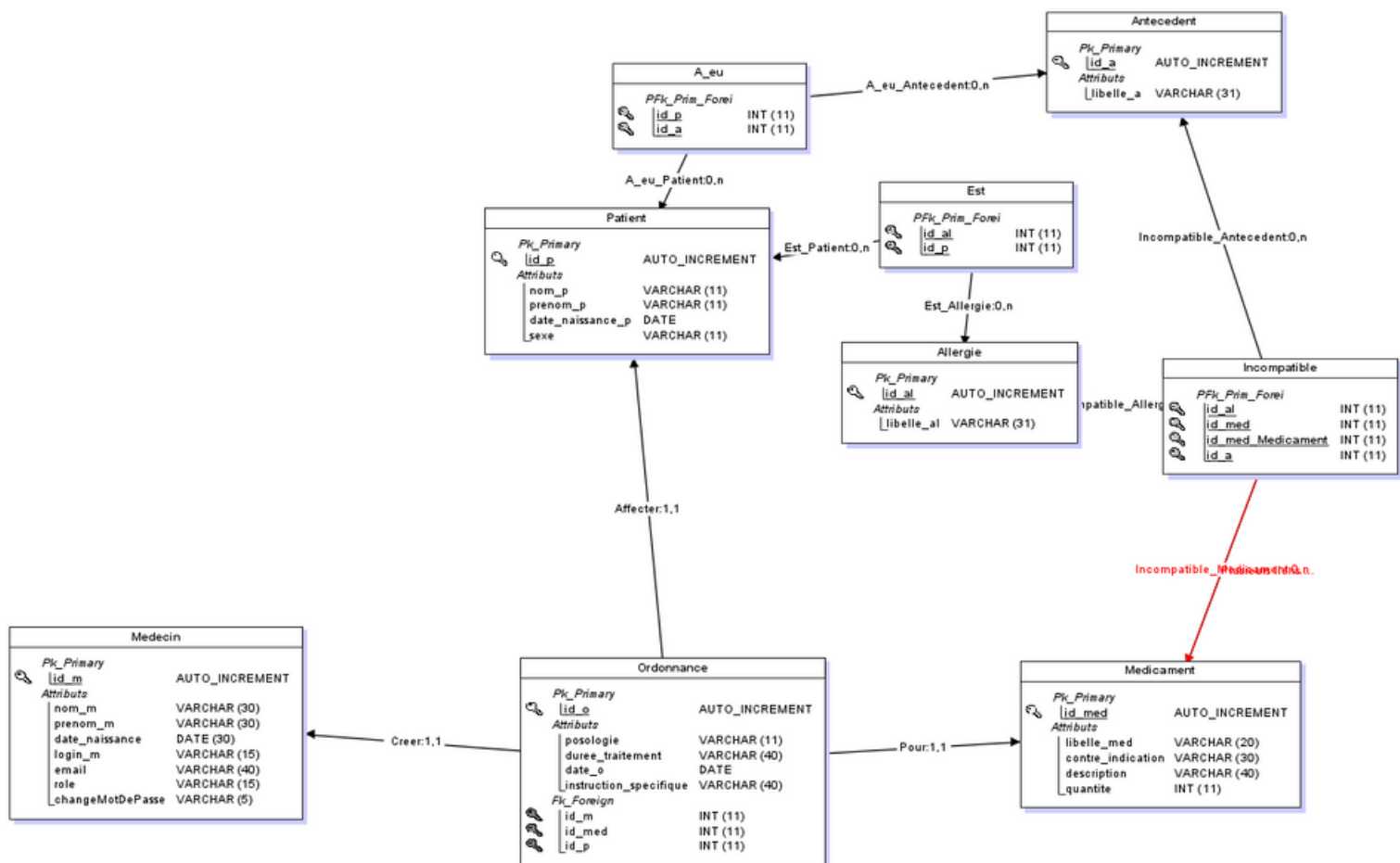
Afin d'avoir accès à une base, il faut avoir WampServer et lancer phpMyAdmin.

Les identifiants de connexions sont dans le fichier APP.config.

Il suffit ensuite d'importer le fichier gsb.SQL



# 04.BASE DE DONNÉE

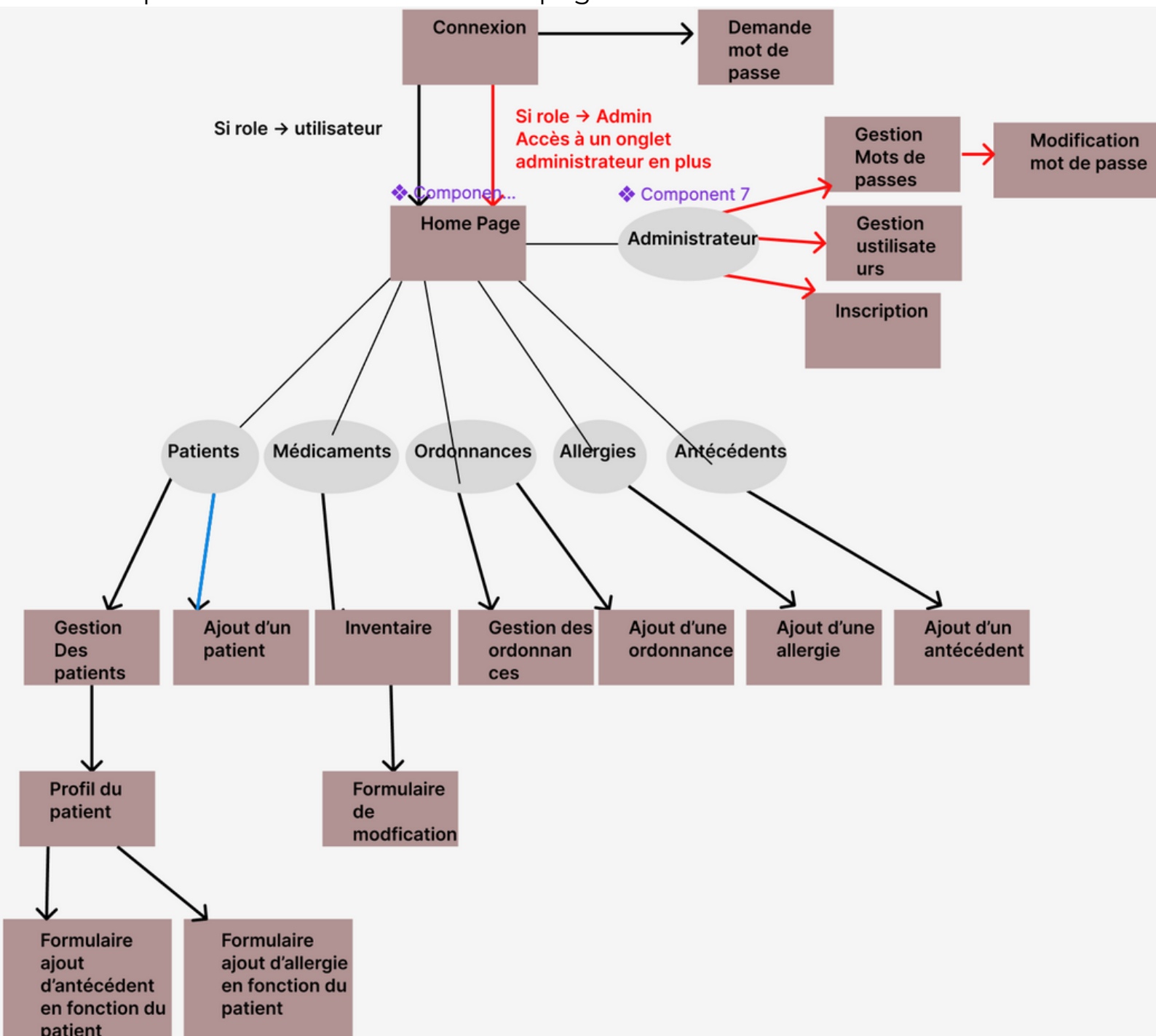




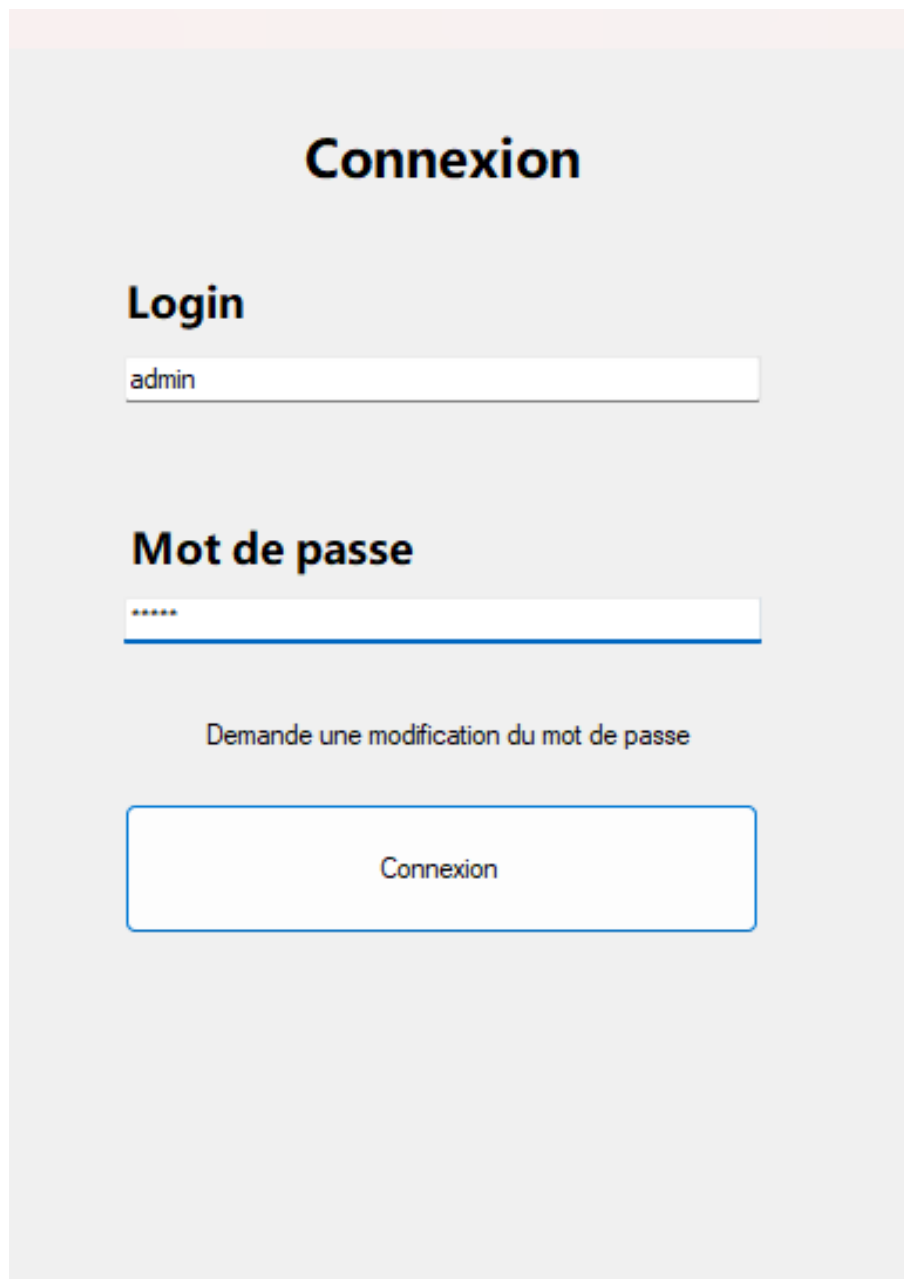
# 05. LES PAGES ET FONCTIONNALITÉS

## SCHEMA DE REPRÉSENTATION DES PAGES ET LEURS LIENS

Chaque rectangle comporte une page et chaque forme ovale comporte un lien dans le menu vers la page. Les flèches représentent les liens entre les pages.



# PAGE DE CONNEXION



The image shows a login page with a light gray background and a white header bar. The title 'Connexion' is centered at the top in a bold, black font. Below it, the word 'Login' is also centered in a bold, black font. There are two input fields: the first contains the text 'admin', and the second contains six asterisks '\*\*\*\*\*'. Below the password field, there is a link that says 'Demande une modification du mot de passe'. At the bottom, there is a large, rounded rectangular button with a blue border and the text 'Connexion' centered inside it.

**Connexion**

**Login**

admin

**Mot de passe**

\*\*\*\*\*

[Demande une modification du mot de passe](#)

Connexion

# PAGE DE CONNEXION


Elle permet d'accéder à la page 'HomePage', on y retrouve également le lien vers un formulaire de demande de changement de mot de passe.

L'utilisateur doit entrer un login et mot de passe.

Si les identifiants ne sont pas corrects ou les cases sont vides, un message apparaît.

Une condition vérifie dans un premier temps si les champs de texte sont vides.

## Vérification des cases :




```
if
((string.IsNullOrEmpty(inputSignInLogin.Text)||
string.IsNullOrEmpty(inputSignInPassword.Text
))){
    MessageBox.Show("Entrer votre login
et mot de passe");}
```

# PAGE DE CONNEXION

Ensuite, on a la vérification en base, qui se fait par une requête SQL

## Requête SQL pour la connexion




```
SELECT login_m, role,
prenom_m, nom_m, id_m FROM
medecin WHERE
login_m=@login AND
password_m=@password;
```

La requête me permet d'avoir toutes ses informations, afin de les stocker dans ma classe user, en tant que currentUser.

Dans un premier temps, on va comparer le nombre de ligne en retour à 0 en retour, si on retrouve 0 ligne, un message est affiché.

## Comparaison de la sortie de la requête et 0



```
if (users.Count == 0) {
    MessageBox.Show("Merci de
vous inscrire");
}
```

En revanche, si les identifiants sont corrects, l'affichage de la page suivante s'affiche en fonction du rôle utilisateur ou admin.

Afin de contrôler cet afficher, on compare la colonne "rôle" de la requête SQL avec utilisateur.

Si c'est égal à "utilisateur", on rend tous les liens présents dans le menu dépliant "Administrateur" invisibles et in-activable avec 'enable' et 'visible'.

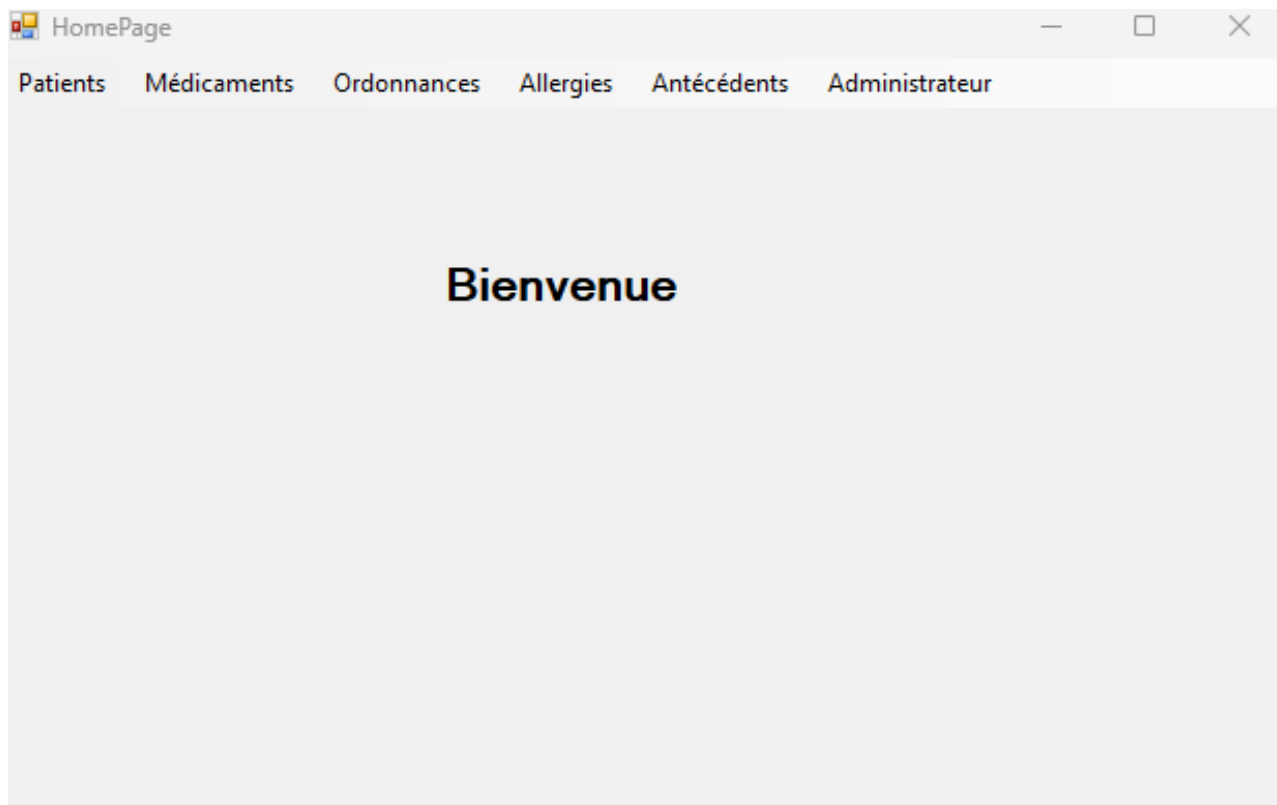
# PAGE DE CONNEXION



```
elseif(role=='utilisateur')
{
    foreach (ToolStripItem
item in
home.menuStrip1.Items)
    {
        switch (item.Name)
        {case
"administrateurToolStrip
MenuItem":
            item.Visible = false;
            item.Enabled = false;
            break;}}
```

Le rôle administrateur permet de créer des nouveaux comptes, de gérer les utilisateurs et de modifier le mot de passe des utilisateur ayant demandés une modification.

# PAGE D'ACCUEIL

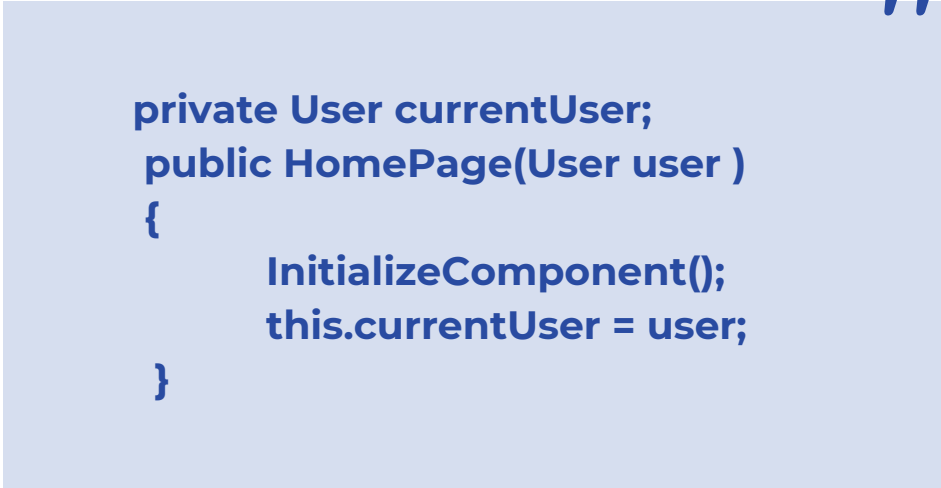


Dans le cas où ce serait un rôle utilisateur, il ne verra pas le menu "administrateur".

# PAGE D'ACCUEIL

La page d'accueil est composée de nombreux menus déroulant qui sont classés par catégories.

Pour commencer, on récupère les informations de l'utilisateur connecté dans 'currentUser'.

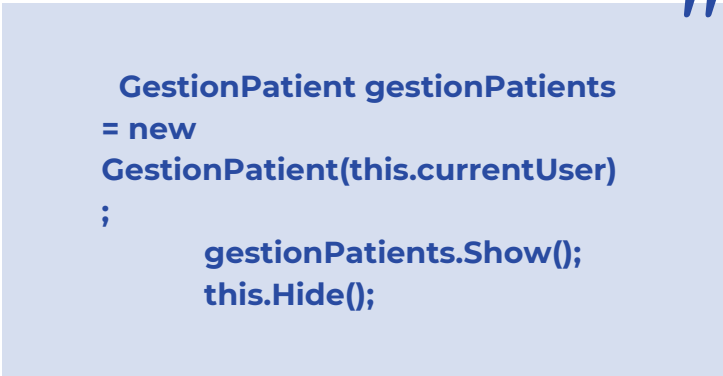


```
private User currentUser;  
public HomePage(User user )  
{  
    InitializeComponent();  
    this.currentUser = user;  
}
```

Ensuite, pour ouvrir les pages associées, on crée un nouvel objet et utilise la fonction Show() afin d'ouvrir la page.

De plus, on transmet bien 'currentUser', afin de garder les informations de l'utilisateur de page en page.

Ici nous prenons  
l'exemple de la page  
Gestion Patients.



```
GestionPatient gestionPatients  
= new  
GestionPatient(this.currentUser)  
;  
    gestionPatients.Show();  
    this.Hide();
```

## GESTION PATIENTS

## Gestion des patients

	Prénom	Nom de famille	Date de naissance	Sexe
►	Jean	Dupont	18/01/2024	M
	karine	dupont	11/01/2024	F

### Profil pacient

[Ajouter un patient](#)

[Supprimer ce patient](#)



# GESTION PATIENTS

La page de gestion des patients comprend un tableau qui représente les noms et prénoms des patients associés à l'utilisateur connecté.

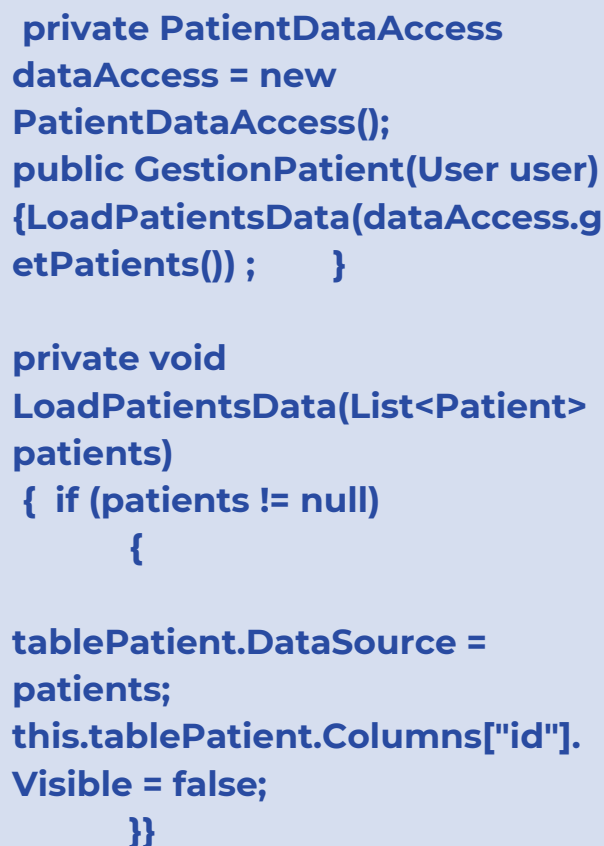
On peut supprimer un patient, accéder à son profil ou accéder au formulaire d'ajout d'un patient.

Pour afficher les patients, on utilise 'DataGridView', qui va afficher les colonnes issues de la requête SQL dans le tableau.

'dataAccess ' permet d'utiliser les fonctions faisant les requêtes SQL concernant les patients.

'DataSource' contient les données à afficher dans 'DataGridView'.

Une fonction 'LoadPatientData' est créée afin d'afficher dans le tableau le résultat de la requête SQL et de paramétrer en fonction des besoins. Comme dans ce cas, on rend invisible la colonne 'id'.




```
private PatientDataAccess  
dataAccess = new  
PatientDataAccess();  
public GestionPatient(User user)  
{LoadPatientsData(dataAccess.g  
etPatients());    }  
  
private void  
LoadPatientsData(List<Patient>  
patients)  
{ if (patients != null)  
  {  
  
tablePatient.DataSource =  
patients;  
this.tablePatient.Columns["id"].  
Visible = false;  
  }  
}}
```

# GESTION PATIENTS


La requête SQL qui sélectionne les patients, les sélectionne en fonction du médecin connecté.

"Date format" permet de renvoyer la date de naissance au bon format, car dans la base SQL elle est stockée sous la forme YYYY-MM-DD.



```
select id_p, prenom_p, nom_p,  
DATE_FORMAT(birth,  
'%d/%m/%Y') as birth, sexe from  
patient where id_m=@id
```

Pour supprimer un patient, il suffit de sélectionner la ligne puis de cliquer sur le bouton supprimé afin de lancer la requête.



```
if  
(tablePatient.SelectedRows.Count > 0) {  
    DataGridViewRow  
    selectedRow =  
    tablePatient.SelectedRows[0];  
    int id =  
(int)selectedRow.Cells["id"].Value;  
    Patient patient = new  
    Patient(id);  
    int result =  
    dataAccess.deletePatient(patient);  
    updateData();  
}
```

# GESTION PATIENTS

La requête SQL qui supprime un patient, se base sur l'id du patient. En supprimant un patient, les id contenu dans les allergies et les antécédents que ce patient a eu.

```
delete from gsb.a_eu where
    id_p=@id;
delete from gsb.est where
    id_p=@id;
delete from patient where
    id_p=@id;
```

Afin d'ajouter un patient, j'accède au bouton "AddPatientPage" et le formulaire d'ajout va s'ouvrir.

Puis en sélectionnant la ligne, on peut accéder au profil du patient.

## FORM PATIENT

### Ajouter un patient

Prénom	Antécédents
<input type="text"/>	Ulcères gastriques actifs ▾ +
Nom de famille	Allergie
<input type="text"/>	Allergie à l'aspirine ▾ +
Genre	Date de naissance
<input type="checkbox"/> femme <input type="checkbox"/> homme	samedi 20 janvier 2024 📅 ▾
<input type="button" value="Ajouter ce patient"/>	

# FORM PATIENT

Il s'agit d'un formulaire d'ajout d'un patient en fonction de l'utilisateur qui est connecté. Il comprend le nom, prénom, sexe, la date de naissance et ses allergies et antécédents.

Le patient va être ajouté dans la base, avec pour médecin, celui qui a créé le patient.

Ensuite, les tables "a\_eu" et est vont permettre de stocker des allergies et antécédents avec l'id de l'antécédent ou de l'allergie et du patient.

La fonction récupère les données et les intègre dans les requêtes SQL, en fonction des arguments dont a besoin.

//

```
private void buttonAddPatient_Click(object  
sender, EventArgs e)  
//récupération des données du patients dans  
les textBox  
Patient patient = new Patient( name,  
lastName, birth, sexe);  
  
int result =  
dataAccess.addPatientToDB(patient,  
id);dataAccess.inscriptionPatientAllergies(  
patient, selectedAllergy);  
  
dataAccess.inscriptionPatientAntecedent(p  
atient, selectedAntecedent);  
gestionPatientForm.updateData();  
this.Close();  
  
}
```

# FORM PATIENT

Les requêtes sont donc lancées en 3 parties, pour intégrer une allergie ou un antécédent, on sélectionne l'id en fonction du nom de l'allergie ou de l'antécédent.

```
INSERT INTO patient (prenom_p,  
nom_p, birth, sexe, id_m)  
VALUES (@name, @lastName,  
@birth, @sexe, @idmedecin);
```

```
insert into est (id_al, id_p) values  
((select id_al from allergie where  
libelle_al=@name), @id)
```

```
insert into a_eu (id_a, id_p)  
values ((select id_a from  
antecedent where  
libelle_a=@name), @id)
```

## PROFIL PATIENT

Le profil du patient contient donc les informations relatives à celui-ci qui sont : le prénom, le nom, la date de naissance, les allergies et antécédents.

Jean Dupont

Date de naissance ✎  
18/01/2024

Sexe  
M

Allergies

	Nom de l'antécédent	
▶	Allergie à l'aspirine	
	Réaction allergiq...	

AjouterSupprimer

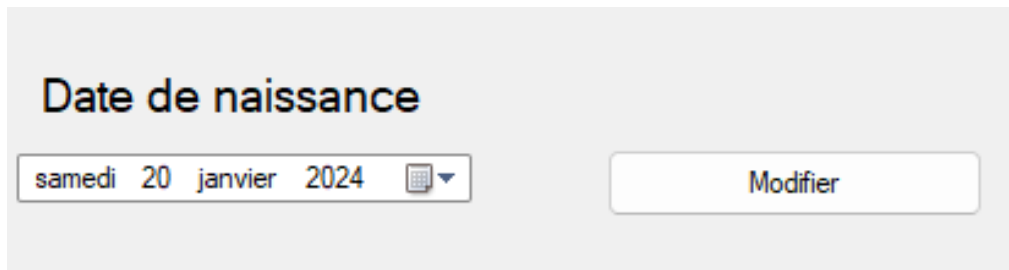
Antecedents

	Name	
▶	Maladie du foie a...	
	Tachycardie	

AjouterSupprimer

# PROFIL PATIENT

La date de naissance peut être modifiée en cliquant sur le crayon.



The screenshot shows a form titled "Date de naissance". Below the title is a date picker displaying "samedi 20 janvier 2024" with a calendar icon. To the right of the date picker is a button labeled "Modifier".

Afin que la date soit au bon format, elle est convertie dans celui que l'on veut avant de l'insérer dans la base de données.

```
DateTimePicker editBirthPatient = new  
DateTimePicker();  
DateTime selectedDate =  
BirthTimePicker.Value;  
string newBirth =  
selectedDate.ToString("yyyy-MM-dd");
```

Pour afficher les tableaux, on fait une requête SQL, qui va nous permettre de mettre le lien les tables qui stockent les données liées aux allergies/antécédents/patients.

# PROFIL PATIENT

"Join" permet de faire le lien entre deux tables à partir d'un index, qui ici sont les clés primaires.

```
SELECT distinct
  antecedent.libelle_a AS Nom
FROM (antecedent, a_eu) INNER
  JOIN est ON antecedent.id_a =
  a_eu.id_a WHERE a_eu.id_p = @id
```

```
SELECT allergie.libelle_al AS
  Nom FROM allergie INNER JOIN
  est ON allergie.id_al = est.id_al
  WHERE est.id_p = @id
```

## AJOUT D'UNE ALLERGIE OU UN ANTÉCÉDENT AU PATIENT

Les deux pages fonctionnent sur des tables différentes, mais on retrouve les mêmes fonctions.

Une 'checkedListBox' permet de pouvoir sélectionner plusieurs données en même temps, qui seront intégrées une par une grâce à une condition foreach().

```
foreach(AntecedentsItem item in
  antecedents) {
  dataAccess.addAntecedentToPatient(item.
  Name, idPatient);
}
```



# AJOUT D'UNE ALLERGIE OU UN ANTÉCÉDENT AU PATIENT

Afin de générer la liste, on utilise la requête `getAntecedents()`. Une expression booléenne permet de savoir si les cases sont cochées ou non et les contenus associés à celle-ci.

//

```
checkedListBoxAntecedent.DataSource =  
antecedents;  
private void  
AddAntecedentPatient_Load(object sender,  
EventArgs e)  
{  
  
    var antecedents =  
dataAccess.getAntecedents()  
    .Select(a => new AntecedentsItem { Name  
= a, IsChecked = false })  
    .ToList();  
  
    checkedListBoxAntecedent.DataSource =  
antecedents;  
    checkedListBoxAntecedent.DisplayMember  
= "Name";  
    checkedListBoxAntecedent.ValueMember =  
"IsChecked";  
}
```

# INVENTAIRE MÉDICAMENT

	Prénom	Description	Contre-Indication	Quantité
►	Aspirine	maux de tête	Allergie à l'aspirine	6
	Paracétamol	maux de tête	Allergie au parac...	6
	Ibuprofène	maux de tête	Ulcères gastrique...	6
	Amoxicilline	maux de tête	Réaction allergiq...	6
	Ciprofloxacine	maux de tête	Tendinite	6
	Metformine	maux de tête	Insuffisance rénale	6
	Omeprazole	maux de tête	Grossesse	6
	Losartan	maux de tête	Hyperkaliémie	6
	Simvastatine	maux de tête	Maladie du foie a...	6
	Amlodipine	maux de tête	Hypotension sév...	4
	Atorvastatine	maux de tête	Maladie du foie a...	6
	Sertraline	maux de tête	Syndrome séroto...	6

Supprimer le médicament

Nom médicament

Description médicament

Contre indication

Quantité

0

Ajouter le médicament

# INVENTAIRE MÉDICAMENT

FormDrug() regroupe l'inventaire des médicaments, l'ajout et la suppression.

Pour ajouter un médicament, on est sur le même procédé que FormPatient.

Les tableaux "DataGridView" comportent une fonction updateData(), elle permet d'être appelée après une fonction comme l'ajout ou la suppression d'un patient, médicament, utilisateur.

Voici l'exemple pour formDrug():

```
public void updateData()
{
    tableDrug.Refresh();
    this.tableDrug.DataSource = null;
    this.tableDrug.DataSource =
dataAccess.selectDrug();

    this.tableDrug.Columns["Id"].Visible =
false;
}
```

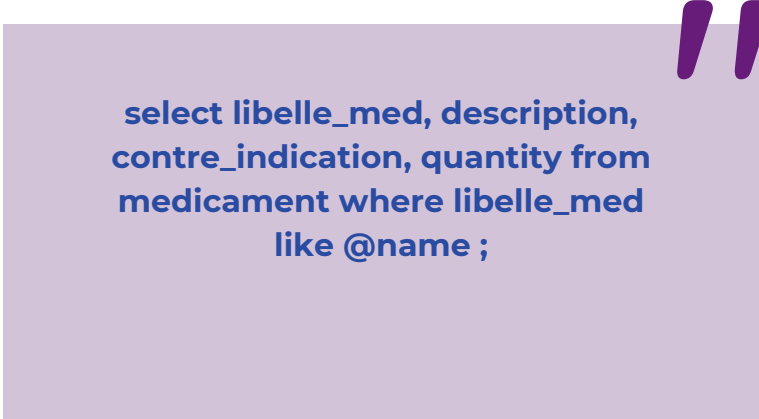
La requête faites initialement à l'ouverture de la page et à nouveau faite en l'attribuant au tableau associé.

Ensuite, on a un champ pour rechercher les médicaments en fonction du nom entré, le tableau va alors se mettre à jour.

```
string name = searchDrug.Text;
name = name + "%";
dataAccess.searchDrug(name);
List<Drug> searchedDrugs =
dataAccess.searchDrug(name);
LoadDrugData(searchedDrugs);
```

# INVENTAIRE MÉDICAMENT

Cette requête se lance quand le texte change, avec en paramètre le nom entré, ses données sont mises à jour avec la fonction LoadDrugData() qui va nous permettre de mettre à jour les données du tableau.



```
select libelle_med, description,  
contre_indication, quantity from  
medicament where libelle_med  
like @name ;
```

Un ajout de médicament se fait directement à gauche en entrant les champs libellés du médicament, les contre-indications associées, sa description et sa quantité en stock en récupérant les valeurs pour l'insérer dans la base.

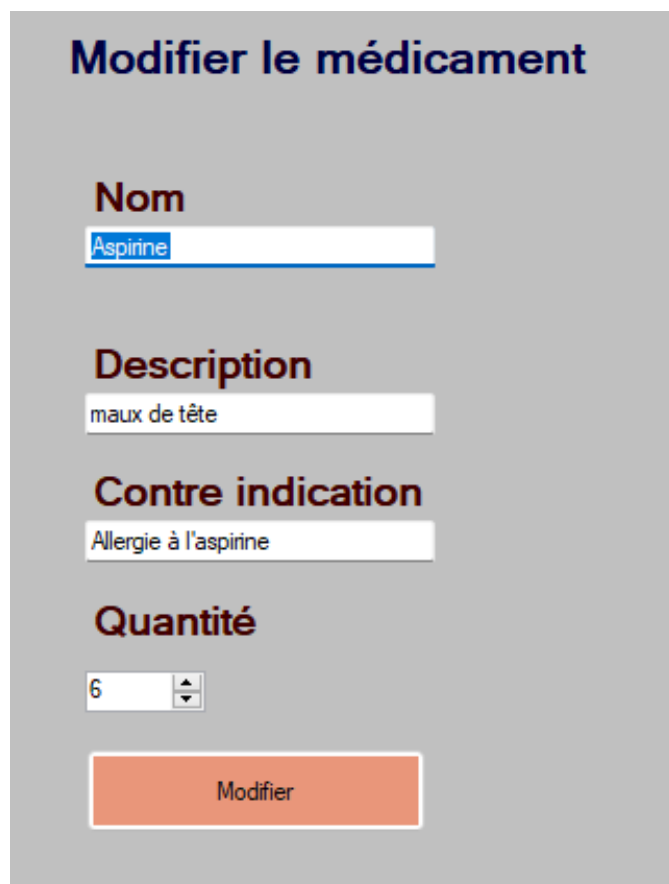
## FORM DRUG

En cliquant deux fois sur le contenu d'une cellule, un formulaire de modification du médicament ;

Afin de modifier les données, je récupère les anciennes pour les retrouver en paramètres dans ma requête et j'insère les nouvelles.

Après la mise à jour du tableau, l'inventaire est mis à jour, on peut chercher les nouvelles informations entrées.

## FORM DRUG



The screenshot shows a web form titled "Modifier le médicament" on a light gray background. The form contains four labeled input fields and a submit button. The labels are in a bold, dark red font. The first field, labeled "Nom", contains the text "Aspirine". The second field, labeled "Description", contains "maux de tête". The third field, labeled "Contre indication", contains "Allergie à l'aspirine". The fourth field, labeled "Quantité", is a spinner box showing the number "6". At the bottom of the form is an orange rectangular button with the text "Modifier" in black.

**Modifier le médicament**

**Nom**  
Aspirine

**Description**  
maux de tête

**Contre indication**  
Allergie à l'aspirine

**Quantité**  
6

Modifier

## GESTION DES ORDONNANCES

La page de gestion des ordonnances affiche les ordonnances associées au médecin connecté, elles incluent la posologie, la durée du traitement, instructions, la date de création, nom du médicament et nom et prénom du patient concerné.

On peut accéder au formulaire d'ajout supprimé l'ordonnance, mais pas la modifier.

# GESTION DES ORDONNANCES

Ordonnances de jean dupont

	Posologie	Durée du traitement	Instruction	Date	Nom Patient	Prénom	Nom du médicament
▶	qwe	0	wre	16/01/2024	Dupont	Jean	Aspirine

Ajouter une ordonnance

Supprimer

Les données sont importé avec selectOrdonnances() qui contient la requête qui va faire le lien avec le nom et prénom du patient par son ID, de même pour médicament

```
select a.id_o, a.posologie, a.duree_traitement,
a.instruction_specifique, DATE_FORMAT(a.date_o,
'%d/%m/%Y') as date_o, p.nom_p as Nom_Patient,
p.prenom_p as Prenom_Patient, m.libelle_med as
Nom_medicament from gsb.ordonnance a join
gsb.patient p on p.id_p = a.id_p join gsb.medicament m
on m.id_med= a.id_med where a.id_m=@idMed;
```

# GESTION DES ORDONNANCES

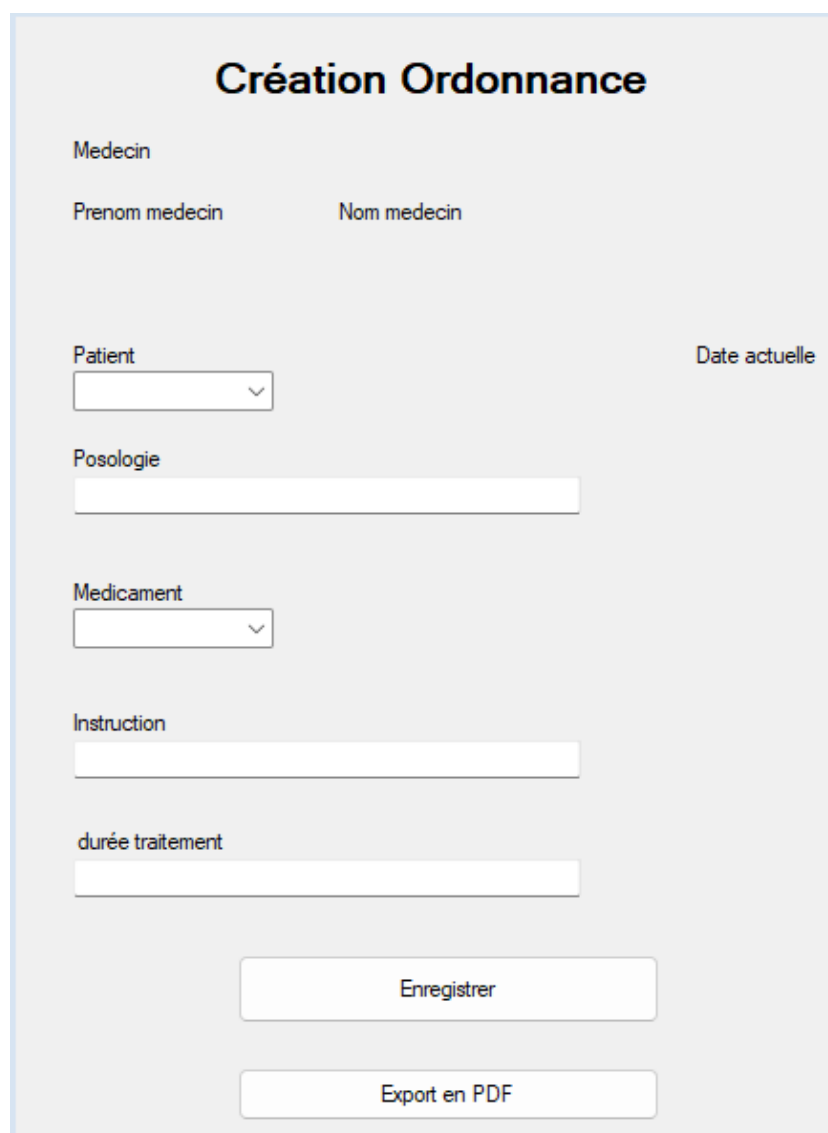
Le tableau est mis à jour de la même manière que pour les autres “DataGridView”.

## AJOUTER UNE ORDONNANCE

Pour ajouter une ordonnance, on récupère le nom et le prénom du médecin connecté, une liste déroulante permet de récupérer les patients associés à ce médecin.

Ensuite, il y a les champs pour entrer la posologie, les instructions, la durée du traitement. Une liste permet de choisir un médicament.

L'ordonnance peut être exportée en PDF.



The screenshot shows a web form titled "Création Ordonnance". It contains several input fields and two buttons. The fields are: "Medecin" (with sub-fields "Prenom medecin" and "Nom medecin"), "Patient" (a dropdown menu), "Date actuelle" (a text field), "Posologie" (a text field), "Medicament" (a dropdown menu), "Instruction" (a text field), and "durée traitement" (a text field). At the bottom, there are two buttons: "Enregistrer" and "Export en PDF".

# AJOUTER UNE ORDONNANCE

Afin de trouver les patients associée, une requête est faite en prenant pour paramètre l'id du médecin connecte.

Avec DataSource, le résultat apparaît dans une liste déroulante. Puis on garde la valeur sélectionnée.

```
List<Patient> patients =
dataAccessPatient.getPatientByMed(idMed);

if (comboBoxPatient != null &&
comboBoxPatient.SelectedItem != null)
{
    selectedPatient =
comboBoxPatient.SelectedItem.ToString();
}
```

Pour créer un PDF du formulaire, il faut avoir le dépendance iText qui va permettre de le générer. On commence par le créer avec les valeurs que l'on veut insérer dedans, il écrit dans ce fichier, ensuite, il est généré.

```
{string filePath = saveFileDialog.FileName;

using (PdfWriter writer = new
PdfWriter(filePath))
{
    using (PdfDocument pdf = new
PdfDocument(writer))
    {
        Document document = new
Document(pdf);
        document.Add(new
Paragraph("Ordonnance du patient " +
comboBoxPatient.Text));
    }}
}}
```




# AJOUTER UNE ORDONNANCE

Lors de l'ajout, plusieurs requêtes sont effectuées à partir de la table incompatible. Elle va vérifier si les allergies et antécédents du patient ne sont pas incompatibles avec le médicament choisi.


Dans ces cas, une alerte est affichée, si le médecin répond OK, l'ordonnance est quand même créée et s'il appuie sur Annuler, celle-ci ne sera pas créée.

Les requêtes retournent le nom des antécédents ou allergies concernés si c'est le cas, puis on compte le nombre de ligne en retour afin de savoir si l'on met un message.

On fait le lien entre les tables avec l'id du patient et médicament afin de récupérer les valeurs dont on a besoin.



```
SELECT a.libelle_a, m.libelle_med FROM a_eu e JOIN  
incompatible i ON e.id_a = i.id_a JOIN antecedent a ON  
i.id_a = a.id_a JOIN médicament m ON i.id_med =  
m.id_med WHERE e.id_p = @id_p AND i.id_med =  
@id_med;
```



```
SELECT a.libelle_al, m.libelle_med FROM est e JOIN  
incompatible i ON e.id_al = i.id_al JOIN allergie a ON  
i.id_al = a.id_al JOIN médicament m ON i.id_med =  
m.id_med WHERE e.id_p = @id_p AND i.id_med =  
@id_med
```

# AJOUTER UNE ALLERGIE OU UN ANTECEDENT

Si une allergie ou un antécédent n'est pas présent(e) dans la base de données, on peut en ajouter directement, on récupère simplement le nom dans la textBox, puis on l'insère dans la base de données qui va lui attribuer automatiquement un ID,

The image displays two overlapping light gray panels, each containing a form for adding data to a database.

**Left Panel: Ajouter un antécédent**

- Title: **Ajouter un antécédent**
- Input field: A single-line text box.
- Button: A button labeled "Ajouter".

**Right Panel: Ajouter une allergie**

- Title: **Ajouter une allergie**
- Label: "Nom de l'allergie"
- Input field: A single-line text box.
- Button: A button labeled "Ajouter".

# GESTION UTILISATEURS

Pour rappel, le menu administrateur n'est accessible qu'aux comptes ayant le rôle administrateur.

Dans la gestion des utilisateurs, on y affiche leurs informations excepte le mot de passe, puis on peut ajouter un nouvel utilisateur ou le supprimer.

Il s'agit du même système que les autres pages de gestion.

**Gestion des utilisateurs**

	Prénom	Nom de famille	Email	Nom d'utilisateur	Rôle
▶	jean	dupont	admin@gmail.com	admin	admin
	Marine	patry	utilisateur@gmail....	utilisateur	utilisateur
	sylvain	luiset	sylvain.luiset@ya...	syluiset	utilisateur

Inscription

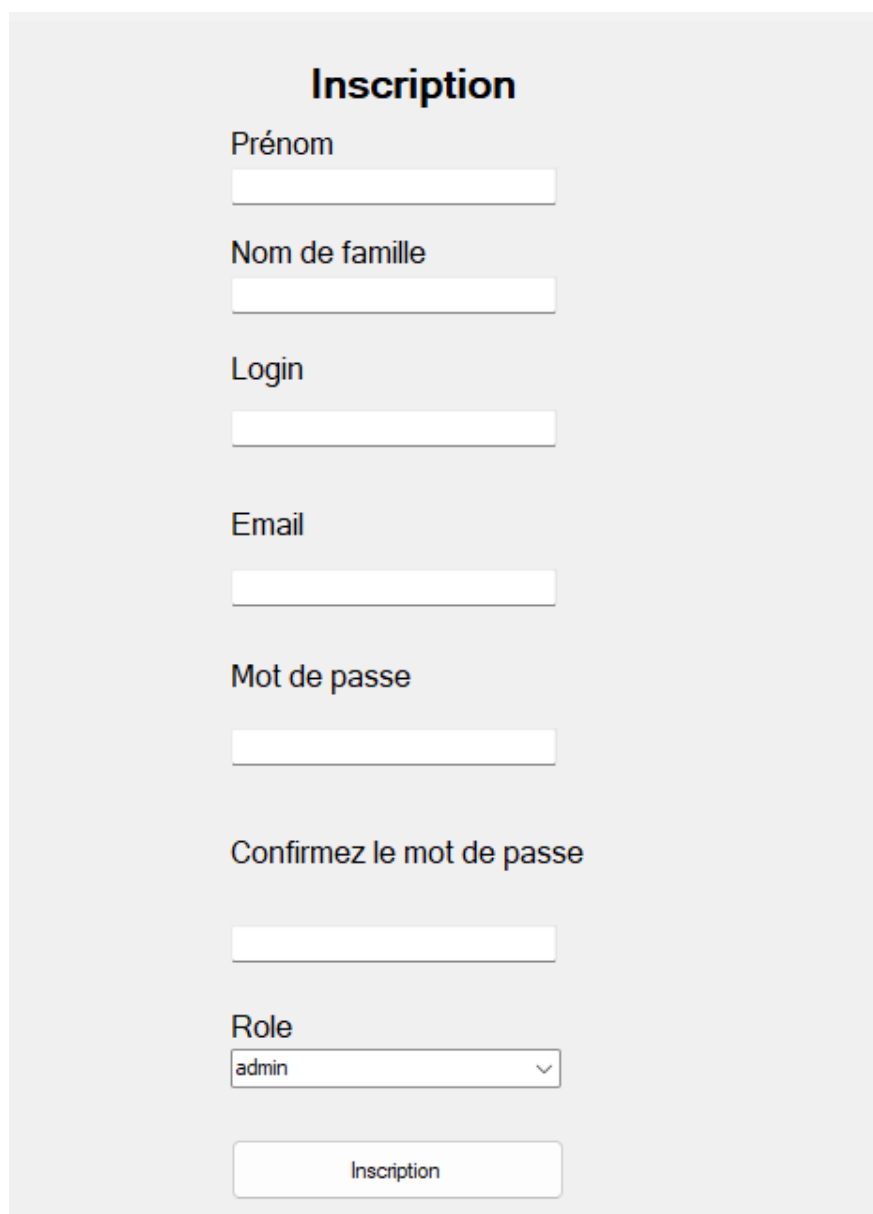
Supprimer

## INSCRIPTION

Afin d'inscrire un utilisateur, on doit entrer tous les éléments dont on a besoin, le prénom, nom, Email, Login, Rôle et le mot de passe.

Il est préférable que ce soit un responsable du service informatique qui fasse l'inscription afin que les utilisateurs aient un mot de passe assez sécurisé.

# INSCRIPTION



The image shows a registration form titled "Inscription" on a light gray background. The form contains the following fields and elements:

- Inscription**: The title of the form.
- Prénom**: A text input field for the first name.
- Nom de famille**: A text input field for the last name.
- Login**: A text input field for the login name.
- Email**: A text input field for the email address.
- Mot de passe**: A text input field for the password.
- Confirmez le mot de passe**: A text input field for confirming the password.
- Role**: A dropdown menu with "admin" selected and a downward arrow.
- Inscription**: A button to submit the registration form.

## GESTION DES MOTS DE PASSES

Pour changer son mot de passe, il faut faire la demande pour commencer dans un formulaire que l'on peut ouvrir depuis l'écran de connexion. On y entre les informations nous concernant, et dans la colonne "ChangeMotDePasse" on y ajoute "yes", afin qu'il soit affiché dans la gestion de mot de passe.

# GESTION DES MOTS DE PASSES

Demandes de changement de mot de passe

	Prénom	Nom de famille
▶	Marine	patry
	sylvain	luiset

## Demande de modification de mot de passe

Prénom

Nom de famille

Email

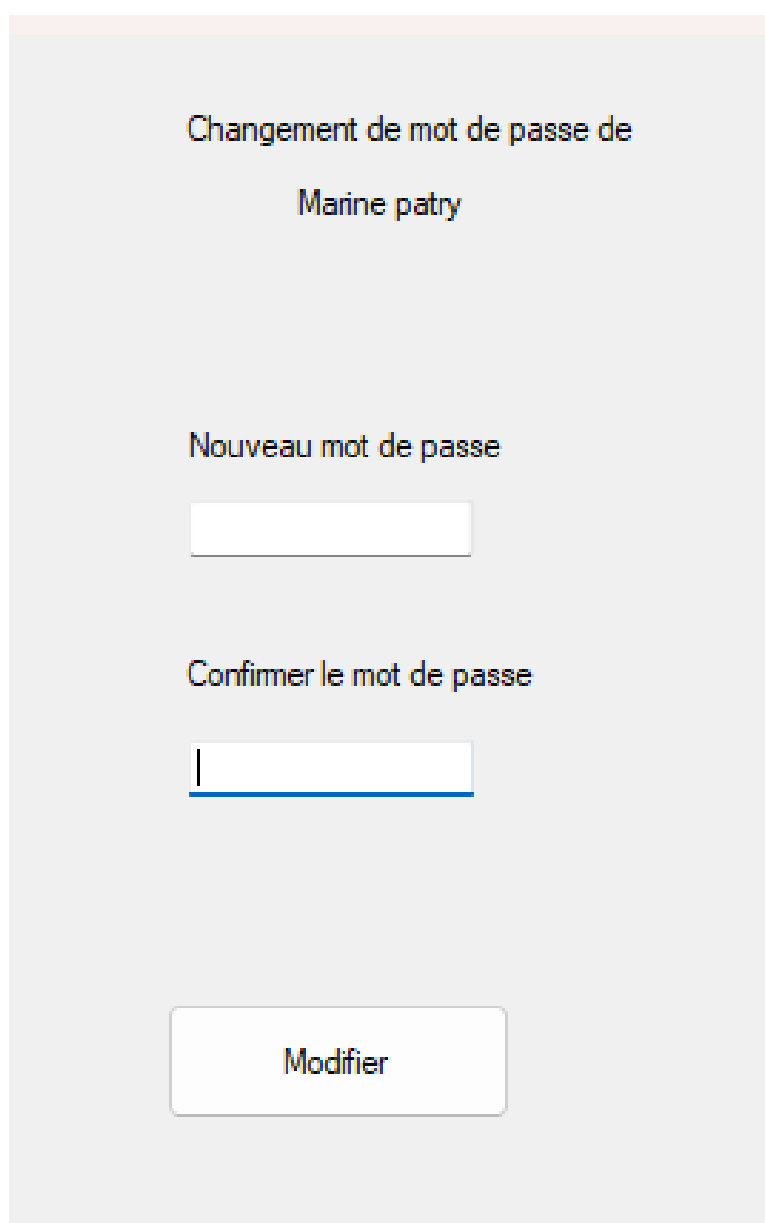
Login

Enregistrer

# GESTION DES MOTS DE PASSES

Pour changer le mot de passe, on clique sur la personne concernée, afin d'ouvrir un formulaire d'édition du mot de passe.

Lorsque celui-ci est changé, la colonne "ChangeMotDePasse" redevient nul.



The image shows a light gray modal window for changing a password. At the top, it says 'Changement de mot de passe de' followed by 'Marine patry'. Below this, there are two input fields: 'Nouveau mot de passe' and 'Confirmer le mot de passe'. The second field has a blue underline. At the bottom is a button labeled 'Modifier'.

Changement de mot de passe de  
Marine patry

Nouveau mot de passe

Confirmer le mot de passe

Modifier

# 06. MANUEL UTILISATEUR

## CONNEXION

Pour se connecter, il faut demander à un administrateur, afin qu'ils nous créaient ce compte.

Ici, il y a le compte avec ses identifiants pour se connecter : [admin/admin](#).

## PATIENT

Pour voir les patients du compte connecter, il faut aller dans la page "Gestion des patients", on peut voir le profil en détail en sélectionnant la ligne et en cliquant sur le bouton "Profil patient".

Pour le supprimer, il faut également sélectionner la ligne et cliquer sur le bouton "supprimé".

Enfin, pour ajouter un patient, on clique simplement sur le bouton "ajout patient".

Sur le profil du patient, on peut ajouter plusieurs antécédents et allergies au patient en cliquant sur les boutons "Ajouter".

La date de naissance peut être modifiée en cliquant sur le crayon.

Afin d'ajouter un patient, on peut ajouter seulement une allergie ou un antécédent.

## MEDICAMENTS

Sur la même page, on retrouve l'ajout de médicament, les médicaments et la rechercher.

Il suffit de cliquer sur le texte d'une cellule que l'on veut modifier.

## ORDONNANCE

On peut ajouter et supprimer une ordonnance, mais pas la modifier.

Lors de l'ajout, une alerte apparaît en cas d'incompatibilités, mais on peut quand même valider ou annuler.

L'export en PDF se fait avant de soumettre l'ordonnance.

## ALLERGIES ET ANTECEDENTS

On peut ajouter et supprimer une ordonnance, mais pas la modifier.

Lors de l'ajout, une alerte apparaît en cas d'incompatibilités, mais on peut quand même valider ou annuler.

L'export en PDF se fait avant de soumettre l'ordonnance.



# CONCLUSION

Ici, on est sur la première version, mais des ajouts de fonctionnalités peuvent être envisagés pour différents acteurs, tels que les chercheurs, les patients...

Voici ci-dessous quelques idées possibles :

**Médicaments plus détaillés avec les compositions chimiques.**  
**Gestion des incompatibilités par les chercheurs.**

**Compte patients pour consulter ses ordonnances et prendre rendez-vous en ligne avec leur médecin.**

**Renforcement de la sécurité de l'application.**