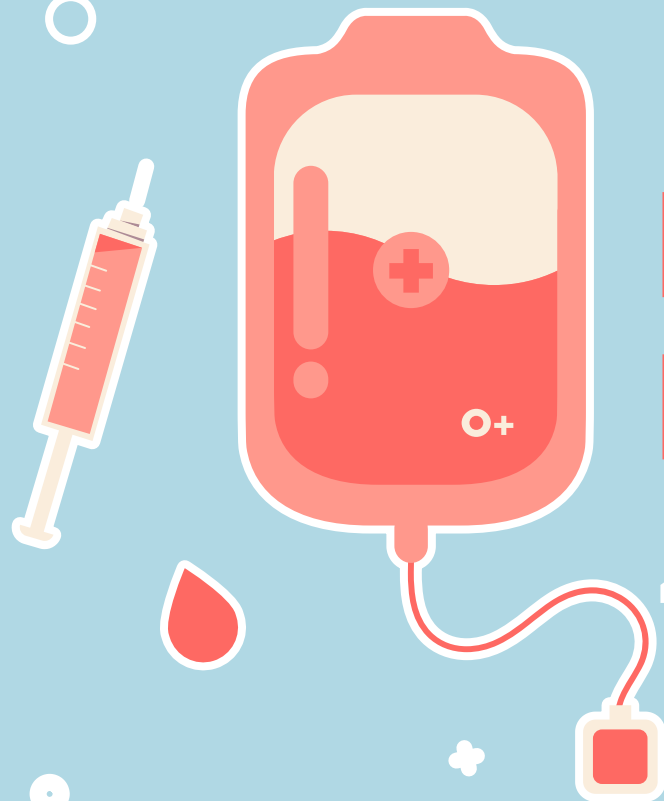




INSTITUT  
SUPÉRIEUR  
DE GESTION  
DE TUNIS  
Since 1969



# Projet Framework.NET 2023-2024

Nour Charfeddine - Nader Ben Ammar

2 LNIG BIS 1

# TABLE DES MATIERES



01

Introduction

02

Base de données

03

Interface “splash  
Screen”

04

Interfaces login

05

Interface employé

06

Interface Main

# TABLE DES MATIERES



07

Interface Donneurs

08

Interface voir Donneurs

09

Interface patients

10

Interface voir patients

11

Stock et transfert de  
sang

12

Conclusion





01

# Introduction

# Introduction :

Dans le cadre de notre projet pour la matière framework.NET, nous avons développé une application de gestion des dons de sang utilisant C#. Cette application vise à simplifier et à améliorer la gestion des processus de collecte, de stockage et de distribution du sang au sein des centres de dons. Dans cette présentation, nous explorerons les fonctionnalités clés de notre application ainsi que son architecture technique.





02

# Base de données

# TABLES UTILISEES

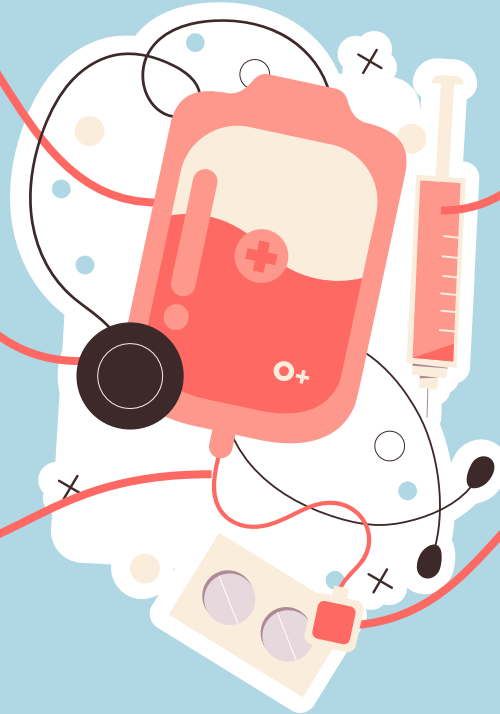
Table des  
donneurs

Table des  
patients

Table des  
employés

Table de stock

Table de  
transfert



# STRUCTURE DES TABLES

Table des donneurs :

```
CREATE TABLE [dbo].[DonorTbl] (  
    [DNum] INT IDENTITY (100, 1) NOT NULL,  
    [DName] VARCHAR (50) NOT NULL,  
    [DAge] INT NOT NULL,  
    [DGender] VARCHAR (10) NOT NULL,  
    [DPhone] VARCHAR (50) NOT NULL,  
    [DAddress] VARCHAR (50) NOT NULL,  
    [DBGroup] VARCHAR (5) NOT NULL,  
    PRIMARY KEY CLUSTERED ([DNum] ASC)  
);
```

	Nom	Type de données
PK	DNum	int
	DName	varchar(50)
	DAge	int
	DGender	varchar(10)
	DPhone	varchar(50)
	DAddress	varchar(50)
	DBGroup	varchar(5)

Table des patients :

```
CREATE TABLE [dbo].[PatientTbl] (  
    [PNum] INT IDENTITY (500, 1) NOT NULL,  
    [PName] VARCHAR (50) NOT NULL,  
    [PAge] INT NOT NULL,  
    [PPhone] VARCHAR (50) NOT NULL,  
    [PGender] VARCHAR (50) NOT NULL,  
    [PBGGroup] VARCHAR (5) NOT NULL,  
    [PAddress] VARCHAR (50) NOT NULL,  
    PRIMARY KEY CLUSTERED ([PNum] ASC)  
);
```

	Nom	Type de données
PK	PNum	int
	PName	varchar(50)
	PAge	int
	PPhone	varchar(50)
	PGender	varchar(50)
	PBGGroup	varchar(5)
	PAddress	varchar(50)



# STRUCTURE DES TABLES

Table des employés :

```
CREATE TABLE [dbo].[EmployeeTbl] (  
    [EmpNum] INT IDENTITY (1, 1) NOT NULL,  
    [EmpId] VARCHAR (50) NOT NULL,  
    [EmpPass] VARCHAR (50) NOT NULL,  
    PRIMARY KEY CLUSTERED ([EmpNum] ASC)  
);
```

	Nom	Type de données
PK	EmpNum	int
	EmpId	varchar(50)
	EmpPass	varchar(50)

Table de Stock:

```
CREATE TABLE [dbo].[BloodTbl] (  
    [BGroup] VARCHAR (5) NOT NULL,  
    [BStock] INT NOT NULL,  
    PRIMARY KEY CLUSTERED ([BGroup] ASC)  
);
```

	Nom	Type de données
PK	BGroup	varchar(5)
	BStock	int

Table de Transfert:

```
CREATE TABLE [dbo].[TransferTbl] (  
    [TNum] INT IDENTITY (1, 1) NOT NULL,  
    [PName] VARCHAR (50) NOT NULL,  
    [BGroup] VARCHAR (50) NOT NULL,  
    PRIMARY KEY CLUSTERED ([TNum] ASC)  
);
```

	Nom	Type de données
PK	TNum	int
	PName	varchar(50)
	BGroup	varchar(50)

03



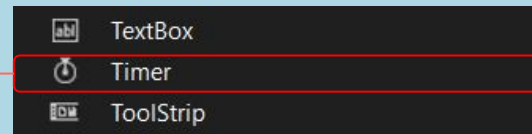
## Interface “SPLASH SCREEN”

# Interface Splash Screen

Il s'agit de la première vue pour notre application.

Pour cette interface on à utiliser une barre de progression de type ("circular progress bar") et un compteur ("Timer") allant de 0 à 100 en incrémentant la valeur de la variable "startpos" pendant cet interval en poussant la progression vers sa fin dans le but de simuler le chargement de l'application.

## Aperçu de l'exécution



```
int startpos = 0;
1 référence
private void timer1_Tick(object sender, EventArgs e)
{
    startpos += 1;
    Myprogress.Value = startpos;
    if (Myprogress.Value == 30)
    {
        Myprogress.Value = 0;
        timer1.Stop();
        Login log = new Login();
        log.Show();
        this.Hide();
    }
}

1 référence
private void splashScreen_Load(object sender, EventArgs e)
{
    timer1.Start();
}
```

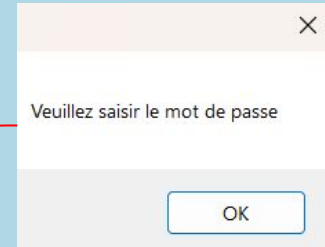
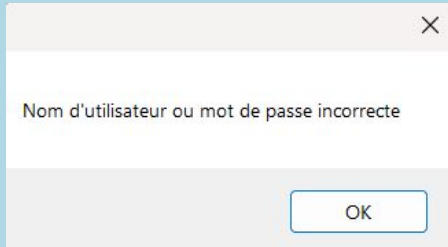
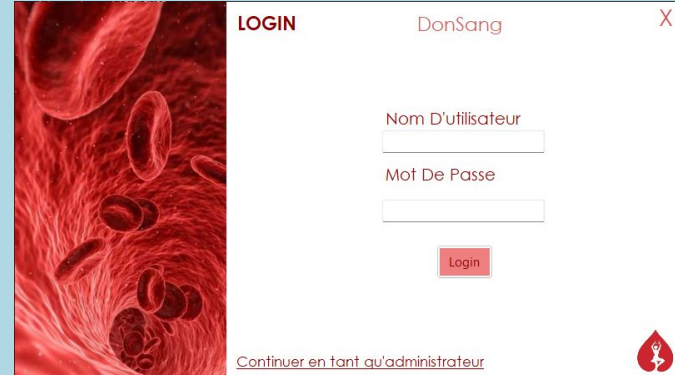
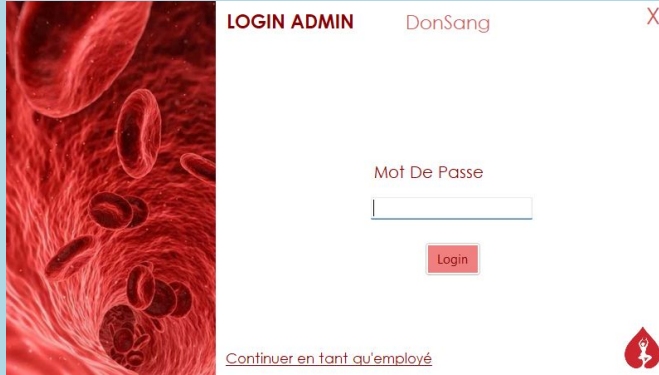


04

# Interface Login

# Interfaces Login

Au niveau d'accès à l'application nous avons implémenté deux interfaces login pour les types d'utilisateurs (Admin user / Employé) .



Messages  
d'erreurs

# Code “Login”

```
SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;  
AttachDbFilename=C:\Users\LENOVO\OneDrive\Documents\BloodBankDb.mdf;Integrated Security=True;Connect Timeout=30;Encrypt=False");
```

Le code débute par l’instanciation d’un objet “SqlConnection” qui portera le chemin de notre base de donnée

```
private void button1_Click(object sender, EventArgs e)  
{  
    Con.Open();  
    SqlDataAdapter sda = new SqlDataAdapter("select count(*) from EmployeeTbl where EmpId = '"+EmpIdTb.Text+"' and EmpPass = '"+EmpPassTb.Text+"'", Con);  
    DataTable dt = new DataTable();  
    sda.Fill(dt);  
    if (dt.Rows[0][0].ToString()=="1")  
    {  
        MainForm Main = new MainForm();  
        Main.Show();  
        this.Hide();  
        Con.Close();  
    }  
    else  
    {  
        MessageBox.Show("Nom d'utilisateur ou mot de passe incorrecte");  
    }  
    Con.Close();  
}
```

Ensuite, la **requête SQL** et l'objet de connexion ("**Con**") sont passés en paramètres à l'objet `SqlDataAdapter` (`**sda**`) pour récupérer et comparer les données (identifiant et mot de passe) de la base de données avec les entrées fournies par l'utilisateur. Les résultats de la requête sont stockés dans l'objet "DataTable" ("**dt**") grâce à la méthode "**Fill()**". Si ces données correspondent aux informations de l'utilisateur, l'application affichera l'interface principale ("**MainForm**"). Sinon, un message d'erreur sera affiché.

Même démarche implémenté pour  
l'utilisateur (**ADMIN**).

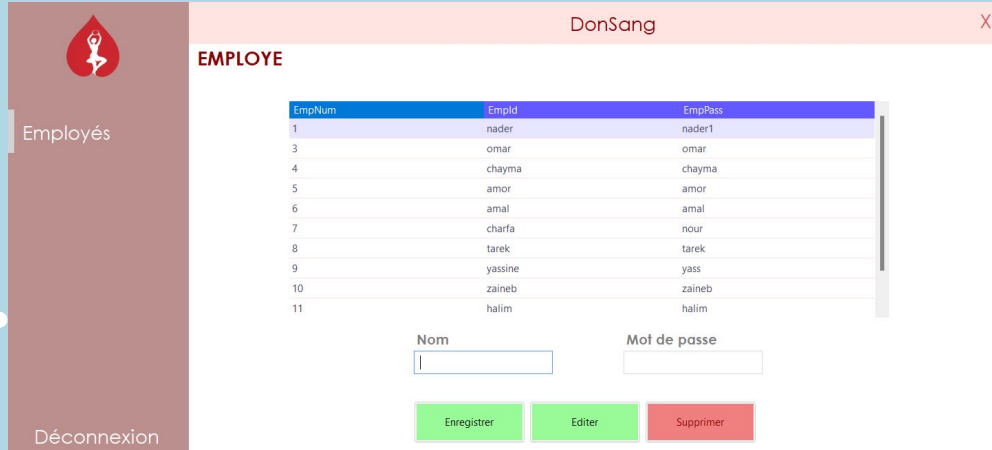


05

Interface  
employé

# Interface Employé

- Si l'utilisateur qui se connecte à l'application est un administrateur, il sera redirigé vers l'interface **"employés"**, où il pourra effectuer des opérations telles que l'ajout, la suppression ou la modification des employés.
- L'utilisateur peut sélectionner un employé dans le **"DataGridView"** pour remplir automatiquement les champs (**nom/prénom**).



DonSang

EMPLOYEE

EmpNum	EmpId	EmpPass
1	nader	nader1
3	omar	omar
4	chayma	chayma
5	amor	amor
6	amal	amal
7	charfa	nour
8	tarek	tarek
9	yassine	yass
10	zaineb	zaineb
11	halim	halim

Nom

Mot de passe

Enregistrer Editer Supprimer

Déconnexion

- Une fois que l'administrateur a terminé la gestion des employés, il peut utiliser le bouton **"Déconnexion"** pour fermer sa session administrative et revenir à l'interface de connexion (**"Login"**) réservée aux employés.



# Code “Employé”



## L'Evenement CellContentClick (Sélectionner un élément depuis le DGV)

```
int key = 0;
1 référence
private void EmpDGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    EmpNameTb.Text = EmpDGV.SelectedRows[0].Cells[1].Value.ToString();
    EmpPassTb.Text = EmpDGV.SelectedRows[0].Cells[2].Value.ToString();

    if (EmpNameTb.Text == "")
    {
        key = 0;
    }
    else
    {
        key = Convert.ToInt32(EmpDGV.SelectedRows[0].Cells[0].Value.ToString());
    }
}
```

Cette ligne récupère la valeur de la deuxième colonne (**index 1**) de la ligne sélectionnée dans le DataGridView (**EmpDGV**) et l'assigne au champ texte (**TextBox**) EmpNameTb.

Cette condition vérifie si le champ “EmpNameTb” est vide

Si EmpNameTb n'est pas vide, l'identifiant de l'employé associé à cette ligne est extrait de la première colonne (**index 0**) et stocké dans la variable key sous forme entière.


# Code “Employé”

## Bouton Enregistrer

Ce bouton permet d'insérer les valeurs saisies (nom/prénom) dans la table des employés après avoir rempli les champs correspondants dans l'interface.

```
private void populate()
{
    Con.Open();
    String Query = "select * from EmployeeTbl";
    SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
    SqlCommandBuilder builder = new SqlCommandBuilder(sda);
    var ds = new DataSet();
    sda.Fill(ds);
    EmpDGV.DataSource = ds.Tables[0];
    Con.Close();
}
```

Après avoir inséré les valeurs dans la table des employés, la méthode populate permet de rafraîchir les données affichées dans le DataGridView en récupérant les informations sur les employés depuis la base de données et en mettant à jour le contenu du DataGridView (DGV).



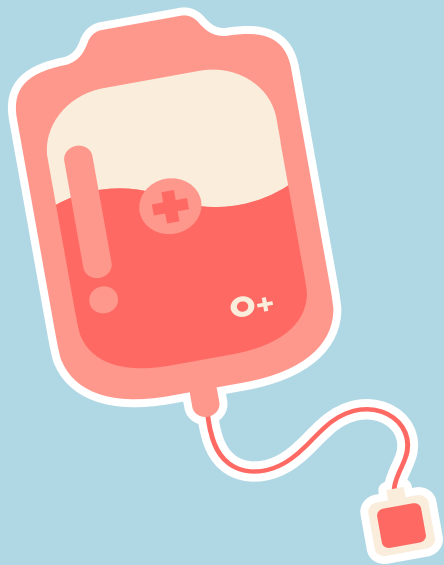
```
private void button1_Click(object sender, EventArgs e)
{
    if (EmpNameTb.Text == "" || EmpPassTb.Text == "")
    {
        MessageBox.Show("Informations Manquantes");
    }
    else
    {
        try
        {
            String query = "insert into EmployeeTbl values ('" + EmpNameTb.Text + "','" + EmpPassTb.Text + "')";
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Employé enregistré avec succès");
            Con.Close();
            reset();
            populate();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```

Base de données

DataAdapter

DataSet

DataGridView

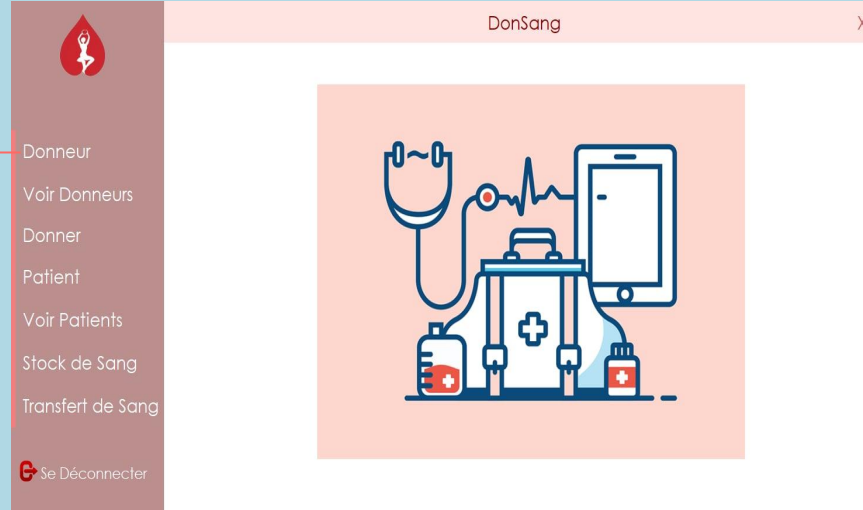


# Interface Main

# Interface Main

```
1 référence  
private void label2_Click(object sender, EventArgs e)  
{  
    Donneur D = new Donneur();  
    D.Show();  
    this.Hide();  
}
```

Lorsque l'utilisateur clique sur le label2(**Donneur**), cet événement déclenche la création d'une nouvelle instance de la classe **Donneur**, affiche sa fenêtre, puis masque la fenêtre actuelle.

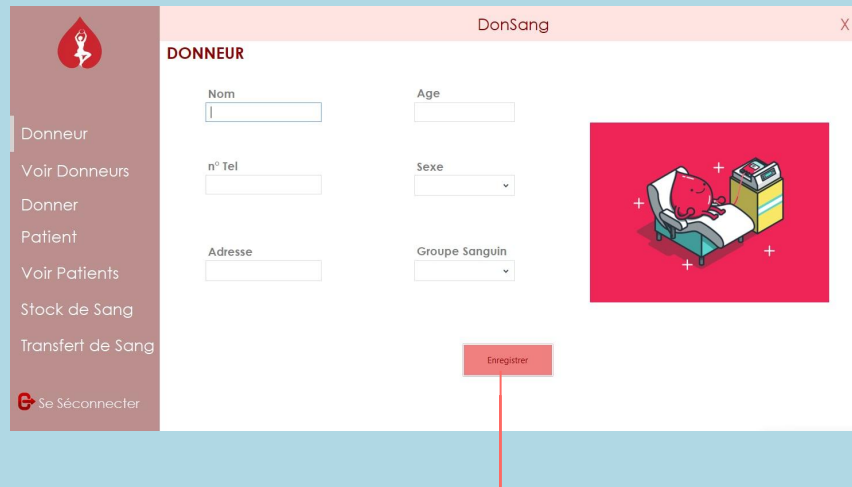


Cette interface Main permet d'afficher les fonctionnalités de notre Application DanSang. Elle nous permet de naviguer entre les différentes sections.



# Interface Donneur

# Interface Donneur



The screenshot shows a web application window titled 'DonSang'. On the left is a sidebar with a red header containing a logo and a list of menu items: 'Donneur', 'Voir Donneurs', 'Donner', 'Patient', 'Voir Patients', 'Stock de Sang', 'Transfert de Sang', and 'Se Déconnecter'. The main content area is titled 'DONNEUR' and contains a form with the following fields: 'Nom' (text input), 'Age' (text input), 'n° Tel' (text input), 'Sexe' (dropdown menu), 'Adresse' (text input), and 'Groupe Sanguin' (dropdown menu). A red button labeled 'Enregistrer' is positioned below the form. To the right of the form is a red square containing a cartoon illustration of a red heart character lying in a hospital bed, connected to a medical machine. A red line points from the 'Enregistrer' button to the text below.

Ce bouton permet d'enregistrer les informations saisi par l'employé concernant le donneur

Ce code utilise **SqlCommand** pour insérer des données dans la base de données et affiche des messages à l'utilisateur avec **MessageBox**. Les contrôles d'interface utilisateur fournissent les données pour **l'insertion**, tandis qu'un objet de connexion et une **méthode reset()** sont utilisés pour gérer la communication avec la base de données et réinitialiser les contrôles.

```
private void button1_Click(object sender, EventArgs e)
{
    if (DNameTb.Text == "" || DPhoneTb.Text == "" || DAddressTb.Text == "" || DAgeTb.Text == "" || DGenCb.SelectedIndex == -1 || DBGroupCb.SelectedIndex == -1)
    {
        MessageBox.Show("Informations Manquantes");
    }
    else
    {
        try
        {
            String query = "insert into DonorTbl values ('" + DNameTb.Text + "'," + DAgeTb.Text + "," + DGenCb.SelectedItem.ToString() + "','" + DPhoneTb.Text + "';";
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Donneur enregistré avec succès");
            reset();
            Con.Close();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```



**Interface**  
**Voir Donneur**

# Interface Voir Donneur



Cette interface permet à l'utilisateur (**Employé**) de lister toutes les informations sur les donneurs existants (**pre-inscrits**) sous forme d'un tableau (**DataGridView**).



Donneur

Voir Donneurs

Donner

Patient

Voir Patients

Stock de Sang

Transfert de Sang

 Se Séconnecter

DonSang

LISTE DES DONNEUR

Nom

DNum	DName	DAge	DGender	DPhone	DAddress	DGroup
100	nader	21	Homme	99315886	ennasr	A+
101	najet	40	Femme	99315613	ariana	A+
102	ridha	45	Homme	20366510	ariana	AB+
103	omar sex	1	Femme	sex	ddd	O+
104	charfa	21	Homme	23568974	azert	O+
105	halim	21	Femme	123456789	m9	AB-
106	omarr	21	Femme	27277990	zbrit	O+
107	nader	3	Homme	99315886	vzzezea	B-
108	nour	20	Femme	45454545	m1	B+
109	omarr	30	Homme	27277990	ennasr	A-

```
SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\M
1 référence
private void populate()
{
    Con.Open();
    String Query = "select * from DonorTbl";
    SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
    SqlCommandBuilder builder = new SqlCommandBuilder(sda);
    var ds = new DataSet();
    sda.Fill(ds);
    DonorDGV.DataSource = ds.Tables[0];
    Con.Close();
}
```

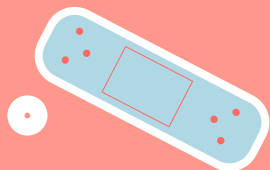
Le code commence par l'établissement de la connexion à la base de données. Ensuite la fonction "populate" permet de capturer tous les donneurs à travers l'exécution d'une requête "SELECT" et le stockage de sa résultat dans l'objet "SqlDataAdapter". Finalement ces données seront transférés vers une "DataSet" et du dataSet vers le DGV qui représente le tableau d'affichage.



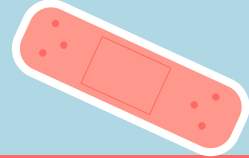




# Interface Patients



# Interface Patients



DonSang

PATIENTS

Nom

Age

n°Tel

Adresse

Sexe

Groupe Sanguin

Enregistrer

```
private void button1_Click(object sender, EventArgs e)
{
    if (PNameTb.Text == "" || PPhoneTb.Text == "" || PAdressTb.Text == "" || PAgeTb.Text == "")
    {
        MessageBox.Show("Informations Manquantes");
    }
    else
    {
        try
        {
            String query = "insert into PatientTbl values ('" + PNameTb.Text + "', " + PAgeTb.Te
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Patient enregistré avec succès");
            reset();
            Con.Close();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```

## Méthodes (btnEnregistrer/reset())


- Le constructeur patient() initialise les composants de l'interface utilisateur.
- La méthode reset() réinitialise les champs de saisie dans l'interface.
- L'événement (clic sur le bouton enregistrer) vérifie si tous les champs sont remplis, puis insère les données du nouveau patient dans la table PatientTbl de la base de données en utilisant une requête SQL INSERT INTO.
- Après l'insertion réussie, il affiche un message de confirmation, réinitialise les champs et ferme la connexion à la base de données. Les exceptions sont gérées pour afficher les erreurs éventuelles.



# Interface Voir Patients

# Interface Voir Patients

Cette interface est composée de 6 champs de texte , 2 boutons (Modifier/Supprimer) et un tableau (DGV) pour l'affichage des informations sur les patients.



Donneur

Voir Donneurs


Donner

Patient

Voir Patients

Stock de Sang

Transfert de Sang

 Se Séconnecter

DonSang

X

LISTE DES PATIENTS

Nom

Age

n° Tel

Sexe

Adresse

Groupe Sanguin

Modifier

Supprimer

PNum	PName	PAge	PPhone	PGender	PBGroup	PAddress
502	med	25	99315889	Homme	A-	zerty
503	charfa	3	98798415	Homme	B+	m1
504	opos	21	33214569	Homme	O+	azertyu
505	tarek	20	85285285	Homme	A+	ennasr
506	halim2	3	12457854	Femme	AB+	m9
507	wassim	3	21548798	Femme	O+	ben aarous
508	feriel	3	78457898	Homme	O-	manar

L'événement du clic sur un élément du DataGridView déclenche la sélection de la ligne entière contenant les informations du patient, et le remplissage des champs associés. ensuite , l'utilisateur peut supprimer le patient sélectionné en cliquant sur le bouton de suppression, ou bien le modifier en cliquant sur "modifier" après avoir appliquer des changements aux champs en question.

# Code Voir Patients

```
private void populate()
{
    Con.Open();
    String Query = "select * from PatientTbl";
    SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
    SqlCommandBuilder builder = new SqlCommandBuilder(sda);
    var ds = new DataSet();
    sda.Fill(ds);
    PatientsDGV.DataSource = ds.Tables[0];
    Con.Close();
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    if (PNameTb.Text == "" || PPhoneTb.Text == "" || PAddressTb.Text == "")
    {
        MessageBox.Show("Informations introuvables");
    }
    else
    {
        try
        {
            String query = "update PatientTbl set PName = '" + PNameTb.Text + "' where PNum = " + PNumTb.Text;
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Patient modifié avec succès");
            Con.Close();
            reset();
            populate();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    if (key == 0)
    {
        MessageBox.Show("veuillez selectionner un patient à supprimer");
    }
    else
    {
        try
        {
            String query = "Delete from PatientTbl where PNum = " + key + " ";
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Patient supprimé avec succès");
            Con.Close();
            reset();
            populate();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}

int key = 0;
private void PatientsDGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    PNameTb.Text = PatientsDGV.SelectedRows[0].Cells[1].Value.ToString();
    PAgeTb.Text = PatientsDGV.SelectedRows[0].Cells[2].Value.ToString();
    PPhoneTb.Text = PatientsDGV.SelectedRows[0].Cells[3].Value.ToString();
    PGenCh.SelectedText = PatientsDGV.SelectedRows[0].Cells[4].Value.ToString();
    PGroupCh.SelectedText = PatientsDGV.SelectedRows[0].Cells[5].Value.ToString();
    PAddressTb.Text = PatientsDGV.SelectedRows[0].Cells[6].Value.ToString();

    if (PNameTb.Text == "")
    {
        key = 0;
    }
    else
    {
        key = Convert.ToInt32(PatientsDGV.SelectedRows[0].Cells[0].Value.ToString());
    }
}
```

Coté technique, la méthode "populate" au début permet de collecter les informations sur les patients et les affichés dans un DataGridView.

Ensuite, après avoir identifié le patient sur lequel l'utilisateur souhaite appliquer des actions telles que la modification ou la suppression, les champs de saisie de l'interface sont automatiquement remplis avec les informations spécifiques du patient sélectionné. Cette interaction directe avec les données du patient facilite la gestion personnalisée des enregistrements. En parallèle, l'exécution des requêtes SQL pour les opérations de modification et de suppression garantit une intégrité et une cohérence des données dans la base de données sous-jacente.



11

# Stock et transfert de sang

# Stock de sang

Cette interface aide l'utilisateur à savoir le niveau de stock pour chaque type de sang.



[Donneur](#)  
[Voir Donneurs](#)  
[Donner](#)  
[Patient](#)  
[Voir Patients](#)  
[Stock de Sang](#)  
[Transfert de Sang](#)  
[Se Séconnecter](#)

DonSang X

STOCK DE SANG

Filtrer AB+

BGroup	BStock
A+	1
A-	1
AB+	1
AB-	0
B+	1
B-	0
O+	0
O-	2



# Transactions de sang

Le stock est automatiquement mise à jour après toute opération qui modifie son niveau.

```
int oldstock;
// référence
private void GetStock(String Bgroup)
{
    Con.Open();
    string query = "select * from BloodTbl where BGroup = '" + Bgroup + "'";
    SqlCommand cmd = new SqlCommand(query, Con);
    DataTable dt = new DataTable();
    SqlDataAdapter sda = new SqlDataAdapter(cmd);
    sda.Fill(dt);
    foreach (DataRow dr in dt.Rows)
    {
        oldstock = Convert.ToInt32(dr["BStock"].ToString());
    }
    Con.Close();
}
```

**DONNER**

**Stock de sang**

Bgroup	BStock
A+	1
A-	1
AB+	1
AB-	0
B+	1
B-	0
O+	0
O-	2

**Liste des donneurs**

DNum	DName	DAge	DGender	DPhone	DAddress	DGroup
100	nader	21	Homme	99315886	ennasr	A+
101	najet	40	Femme	99315613	ariana	A+
102	ridha	45	Homme	20366510	ariana	AB+
104	charfa	21	Homme	23568974	azert	O+
105	halim	21	Femme	123456789	m9	AB-
106	omarr	21	Femme	27277990	esprit	O+
108	nour	20	Femme	45454545	m1	B+
109	omarr	30	Homme	27277990	ennasr	A-
110	amor	21	Homme	87878787	azerty	O+
112	nour	21	Femme	20366510	m1	O+

Nom

Groupe Sanguin

Donner

```
private void DonorDGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    DNameTb.Text = DonorDGV.SelectedRows[0].Cells[1].Value.ToString();
    BGroupTb.Text = DonorDGV.SelectedRows[0].Cells[6].Value.ToString();
    GetStock(BGroupTb.Text);
}
```

A Travers ces deux tableaux (DGV) on peut sélectionner le donneur qui a effectué le don et voir le changement du stock en temps réel.

Le bouton "donner" permet d'effectuer le don après avoir sélectionné le donneur en question.



# Code bouton “donner”

```
private void button1_Click(object sender, EventArgs e)
{
    if (DNameTb.Text == "")
    {
        MessageBox.Show("Veuillez selectionner un donneur");
    }
    else
    {
        try
        {
            int stock = oldstock + 1;
            String query = "update BloodTbl set BStock = " + stock + " where BGroup = '" + BGroupTb.Text + "'";
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Don effectué avec succès");
            Con.Close();
            reset();
            bloodStock();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```

```
private void bloodStock()
{
    Con.Open();
    String Query = "select * from BloodTbl";
    SqlDataAdapter sda = new SqlDataAdapter(Query, Con);
    SqlCommandBuilder builder = new SqlCommandBuilder(sda);
    var ds = new DataSet();
    sda.Fill(ds);
    BloodStockDGV.DataSource = ds.Tables[0];
    Con.Close();
}
```

Le code commence par vérifier que l'utilisateur a sélectionné un donneur, si oui, il change la variable contenant le dernier niveau de stock en incrémentant sa valeur de 1. Ensuite il insère la nouvelle valeur obtenue pour cette variable dans la table 'blood Tbl' En utilisant :

- Une requete SQL
- Un objet SqlCommand pour l'exécution

Ensuite, on fait appel à la méthode reset() réinitialiser les champs(Nom/Groupe Sanguin) Et finalement on fait appel à la méthode "bloodStock()" pour afficher le niveau du stock dans le DataGridView.



# Transactions de sang

Cette interface permet de vérifier l'existence en stock du type de sang pour un patient sélectionné, si c'est le cas alors on peut effectuer le transfert en cliquant sur le bouton transférer.



DonSang

X

Donneur

Voir Donneurs

Donner

Patient

Voir Patients

Stock de Sang

Transfert de Sang

TRANSFERT DE SANG

ID Patient

508

Nom

feriel

Groupe Sanguin


O-


Valable en stock

Transfert effectué avec succès

OK

Transférer

 Se Déconnecter



DonSang

X

Donneur

Voir Donneurs

Donner

Patient

Voir Patients

Stock de Sang

Transfert de Sang

TRANSFERT DE SANG

ID Patient

507

Nom


wassim

Groupe Sanguin

O+

Hors Stock

Transférer

 Se Déconnecter

# Code bouton “transferer”

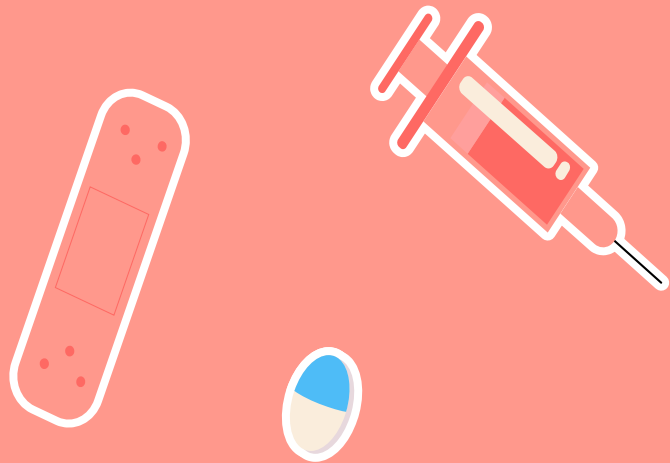
```
private void TransferBtn_Click(object sender, EventArgs e)
{
    if (PatNameTb.Text == "")
    {
        MessageBox.Show("Informations Manquantes");
    }
    else
    {
        try
        {
            String query = "insert into TransferTbl values ('" + PatNameTb.Text + "','" + BloodGroup.Text + "')";
            Con.Open();
            SqlCommand cmd = new SqlCommand(query, Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Transfert effectué avec succès");

            Con.Close();
            GetStock(BloodGroup.Text);
            updateStock();
            reset();
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
    }
}
```

```
private void PatientIdCb_SelectionChangeCommitted(object sender, EventArgs e)
{
    GetData();
    GetStock(BloodGroup.Text);
    if (stock > 0)
    {
        TransferBtn.Visible = true;
        AvailableLbl.Text = "Valable en stock";
        AvailableLbl.Visible = true;
    }
    else
    {
        AvailableLbl.Text = "Hors Stock";
        AvailableLbl.Visible = true;
    }
}
```

Ce code gère la sélection d'un patient dans un ComboBox en déclenchant l'événement PatientIdCb\_SelectionChangeCommitted. Il utilise les méthodes GetData() et GetStock(BloodGroup.Text) pour obtenir des informations sur le patient et le stock de sang du groupe sélectionné. En fonction du stock disponible, il rend visible le bouton de transfert (TransferBtn) et affiche l'état du stock dans une étiquette (AvailableLbl). Lorsque l'utilisateur clique sur ce bouton, le système vérifie le champ du nom du patient (PatNameTb) et enregistre le transfert dans une table (TransferTbl) via une requête SQL. Ensuite, le stock est mis à jour et les contrôles de l'interface sont réinitialisés (reset()) pour une nouvelle saisie. Les erreurs éventuelles sont gérées par des exceptions affichant des messages d'erreur appropriés.





**Conclusion**

# CONCLUSION



Les apprentissages tirés de ce projet sont applicables à de nombreux contextes. Du point de vue technique, ce projet a nécessité la mise en place d'une connexion à une base de données pour interagir avec elle et récupérer des informations spécifiques sur les donneurs, les patients et le niveau de stock. De plus, cette expérience a offert une excellente occasion d'approfondir la compréhension de la navigation entre les différentes interfaces d'une application, contribuant ainsi à développer des compétences clés en développement logiciel.