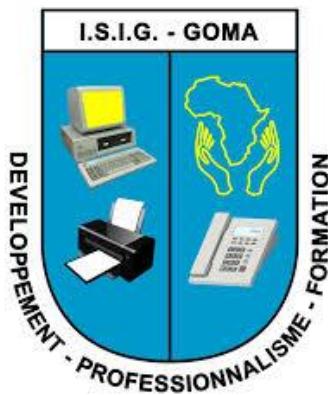


ENSEIGNEMENT SUPERIEUR ET UNIVERSITAIRE

« ESU »

INSTITUT SUPERIEUR D'INFORMATIQUE ET DE GESTION

I.S.I.G - GOMA



www.isig.ac.cd

EXAMEN DU COURS DE GENIE LOGICIEL

GROUPE N° 21 - LIC3 / LIAGE & LIC4

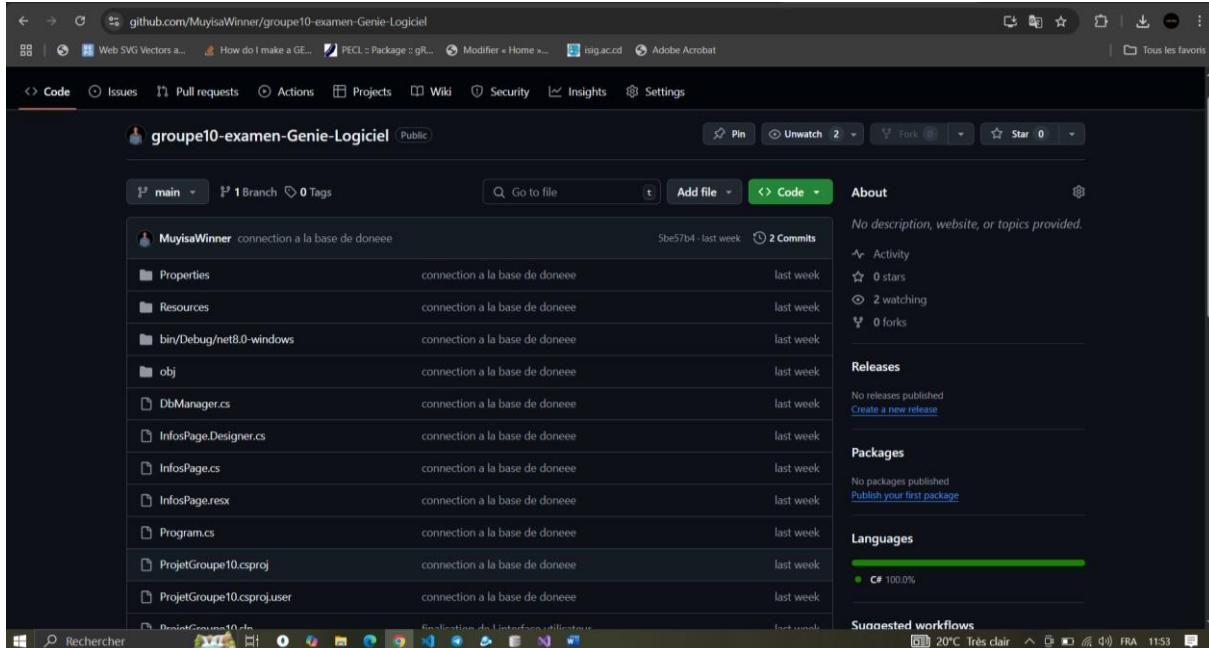
Année Académique : 2024-2025

Participation :

1. OLANGI SIKULI Samuel
2. ZIELE MPANGA JEDIDAH
3. ZAWADI SHAMAVU DIANE
4. YUSUFU SENDO JONAS
5. YALUWE BONHEUR ETIENNE
6. WAYI JEREMIE SALVADOR
7. WASINGYA NZANZU ELNATHAN
8. WABULAKOMBE IGUNZI JOSEPH
9. VUSIKE KAKULE KIMIMBI
10. USHINDI KASONGO PASCAL
11. USHINDI MUNYABA RUBEN
12. USHINDI KASIGWA VICTORINE
13. TUMUSIFU MUJARUGAMBA SCHOLASTIQUE
14. TUMAINI NDAHUGWA PAMELA
15. TUAMINI SEBABY DANIEL
16. TSONGO MUYISA WINNER
17. TSHIKAYA MULUMBA
18. TCHIKO KWINKA CYNTHIA
19. SUZANE UGWARO RIHA
20. SINZAHERA MANEGABE JOSUE
21. SILIMI KUMBA DEUS
22. SHUMBA BISIMWA EMMANUEL
23. SHEMA MUHOZA ALPHONSE
24. SHEKINAH MAPENDO BANGA

Étape 1 : Initialisation du projet

Le projet commence par la mise en place d'un **Référentiel GitHub** pour permettre le travail collaboratif. Ce choix permet à plusieurs membres de l'équipe de travailler en parallèle tout en suivant l'évolution du code grâce à l'historique des commits. Ensuite, un nouveau projet est créé dans **Visual Studio**, qui servira d'environnement principal pour le développement.



Repository GitHub

The screenshot shows the 'Manage access' section of a GitHub repository settings page. On the left, there's a sidebar with various security and integration options like Actions, Webhooks, Environments, Codespaces, Pages, Code Security, Deploy keys, Secrets and variables, GitHub Apps, and Email notifications. The main area is titled 'Manage access' and lists four users with pending invites:

- Josue Isamuna Nkembo (Collaborator)
- Pascal8924 (Awaiting Pascal8924's response)
- Shekinahmapendanobanga (Awaiting Shekinahmapendanobanga's response)
- Vusike-kimimbi (Awaiting Vusike-kimimbi's response)

At the bottom, there's a call-to-action button 'Create an organization'.

Collaborateur du projet groupe

Étape 2 : Conception de la base de données

L’application repose sur une base de données relationnelle contenant trois tables principales :

- **Personne** : enregistre les informations personnelles de l’utilisateur.
- **Adresse** : stocke l’adresse de chaque personne.
- **Téléphones** : gère les différents numéros associés à une même personne.

Chaque table est conçue pour permettre des opérations de création, lecture, mise à jour et suppression (CRUD). Les relations entre les tables sont bien définies afin d’assurer la cohérence des données.

SQLQuery1.sql - DESKTOP-209D8F6.groupe10 (sa (56)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery1.sql - DES...-groupe10 (sa (56))

```

USE [groupe10]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[adresseTable](
    [id] [varchar](255) NOT NULL,
    [numero] [varchar](20) NOT NULL,
    [rue] [varchar](255) NOT NULL,
    [quarter] [varchar](255) NOT NULL,
    [commune] [varchar](255) NOT NULL,
    [ville] [varchar](255) NOT NULL,
    [person] [varchar](255) NOT NULL,
    PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[adresseTable] WITH CHECK ADD CONSTRAINT [fk_person] FOREIGN KEY([person])
REFERENCES [dbo].[personnesTable] ([id])
GO
ALTER TABLE [dbo].[adresseTable] CHECK CONSTRAINT [fk_person]
GO

```

Properties

Aggregate Status

Connection failure:
Elapsed time:
Finish time:
Name: DESKTOP-209D8F6
Rows returned: 0
Start time:
State: Open

Connection

Connection name: DESKTOP-209D8F6 (sa)

Connection Details

Connection elapsed:
Connection encrypt: Not encrypted
Connection finish t:
Connection rows n: 0
Connection start ti:
Connection state: Open
Display name: DESKTOP-209D8F6
Login name: sa
Server name: DESKTOP-209D8F6
Server version: 13.0.1601
Session Tracing ID:
SPID: 56

Name: The name of the connection.

Ready

Rechercher

Connected. (1/1)

DESKTOP-209D8F6 (13.0 RTM) | sa (56) | groupe10 | 00:00:00 | 0 rows

Ln 35 Col 1 Ch 1 INS

20°C Très clair ⌂ FRA 12:08

Création table adresse

SQLQuery2.sql - DESKTOP-209D8F6.groupe10 (sa (57)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery2.sql - DES...-groupe10 (sa (57))

```

USE [groupe10]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[personnesTable](
    [id] [varchar](255) NOT NULL,
    [nom] [varchar](255) NOT NULL,
    [post] [varchar](255) NOT NULL,
    [prenom] [varchar](255) NOT NULL,
    [sexe] [varchar](255) NULL,
    PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

```

Properties

Aggregate Status

Connection failure:
Elapsed time:
Finish time:
Name: DESKTOP-209D8F6
Rows returned: 0
Start time:
State: Open

Connection

Connection name: DESKTOP-209D8F6 (sa)

Connection Details

Connection elapsed:
Connection encrypt: Not encrypted
Connection finish t:
Connection rows n: 0
Connection start ti:
Connection state: Open
Display name: DESKTOP-209D8F6
Login name: sa
Server name: DESKTOP-209D8F6
Server version: 13.0.1601
Session Tracing ID:
SPID: 57

Name: The name of the connection.

Ready

Rechercher

Connected. (1/1)

DESKTOP-209D8F6 (13.0 RTM) | sa (57) | groupe10 | 00:00:00 | 0 rows

Ln 26 Col 1 Ch 1 INS

20°C Très clair ⌂ FRA 12:08

Table personne

```

USE [group10]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[telephoneTable](
    [id] [varchar](255) NOT NULL,
    [code] [varchar](255) NOT NULL,
    [numero] [varchar](255) NOT NULL,
    [person] [varchar](255) NOT NULL,
    PRIMARY KEY CLUSTERED
    (
        [id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[telephoneTable] WITH CHECK ADD CONSTRAINT [fk_person_phone] FOREIGN KEY([person])
REFERENCES [dbo].[personnesTable] ([id])
GO

ALTER TABLE [dbo].[telephoneTable] CHECK CONSTRAINT [fk_person_phone]
GO

```

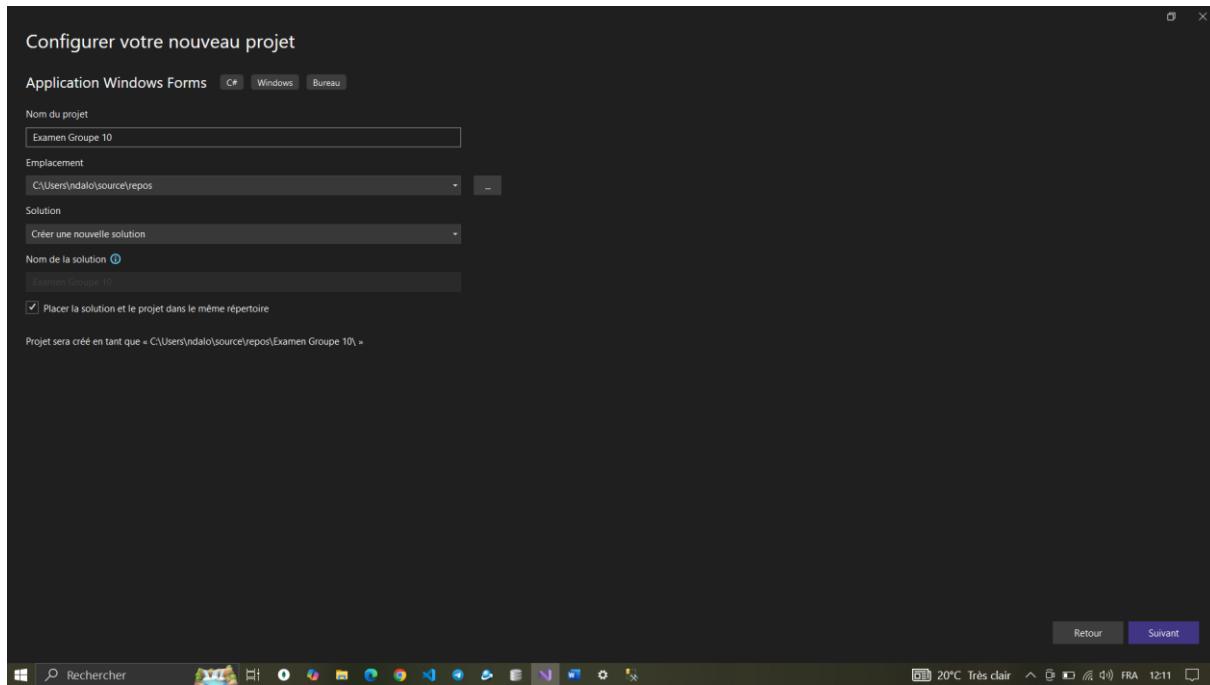
Table téléphones

Étape 3 : Conception des interfaces graphiques

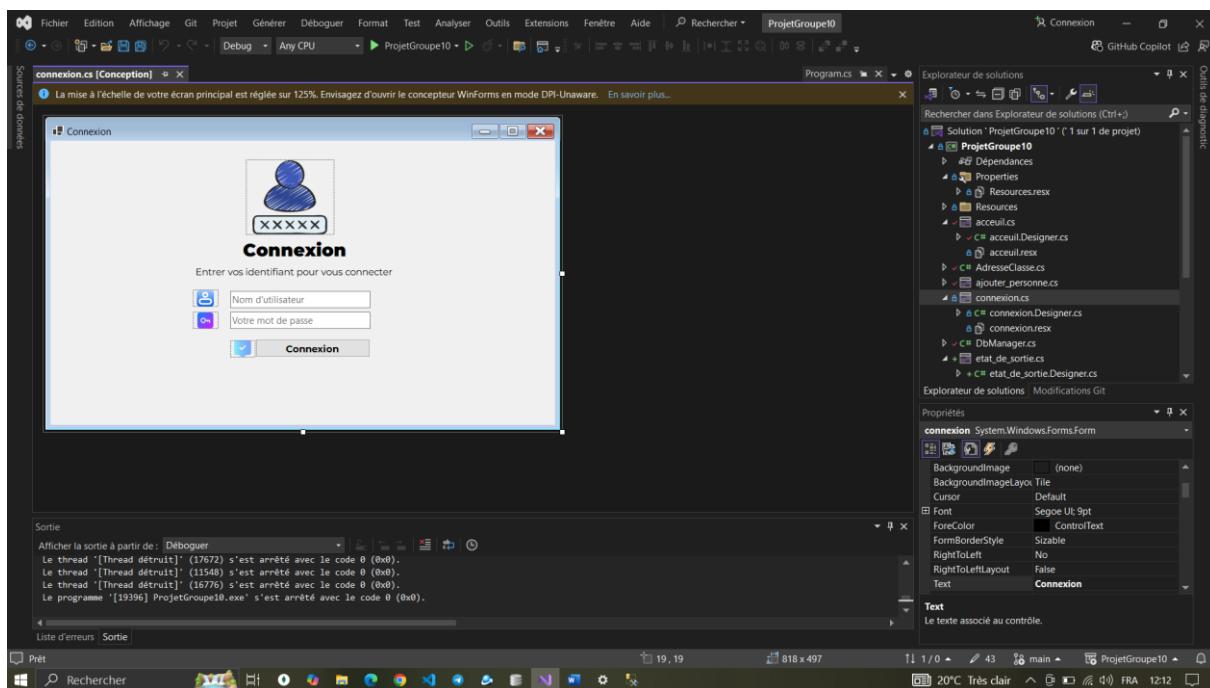
Des interfaces Windows Forms sont développées pour faciliter l’interaction avec l’utilisateur. On y retrouve :

- Une **interface d’accueil** qui présente les principales fonctionnalités.
- Un **formulaire d’ajout d’une nouvelle personne**, avec des champs pour les informations personnelles, l’adresse et les numéros.
- Une **interface d’affichage**, qui présente sous forme de tableau toutes les personnes enregistrées avec la possibilité de les modifier ou de les supprimer.

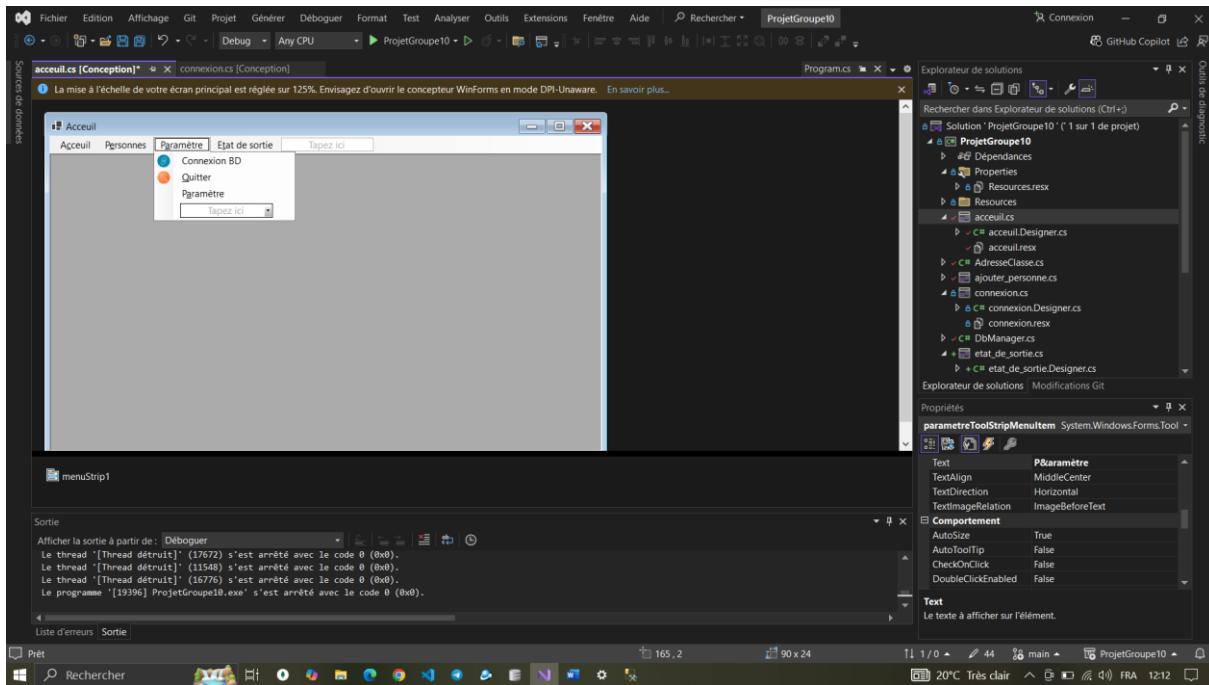
Ces interfaces sont pensées pour être simples à utiliser, tout en intégrant des contrôles nécessaires à la validation des données saisies.



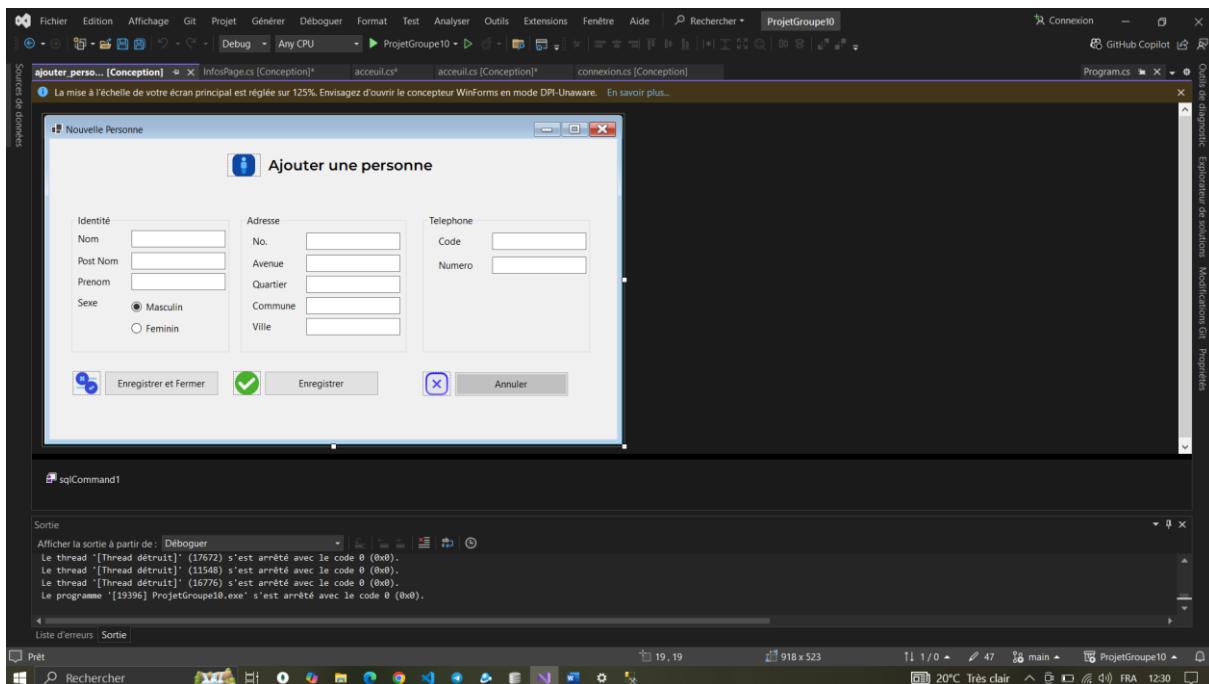
Création du nouveau projet



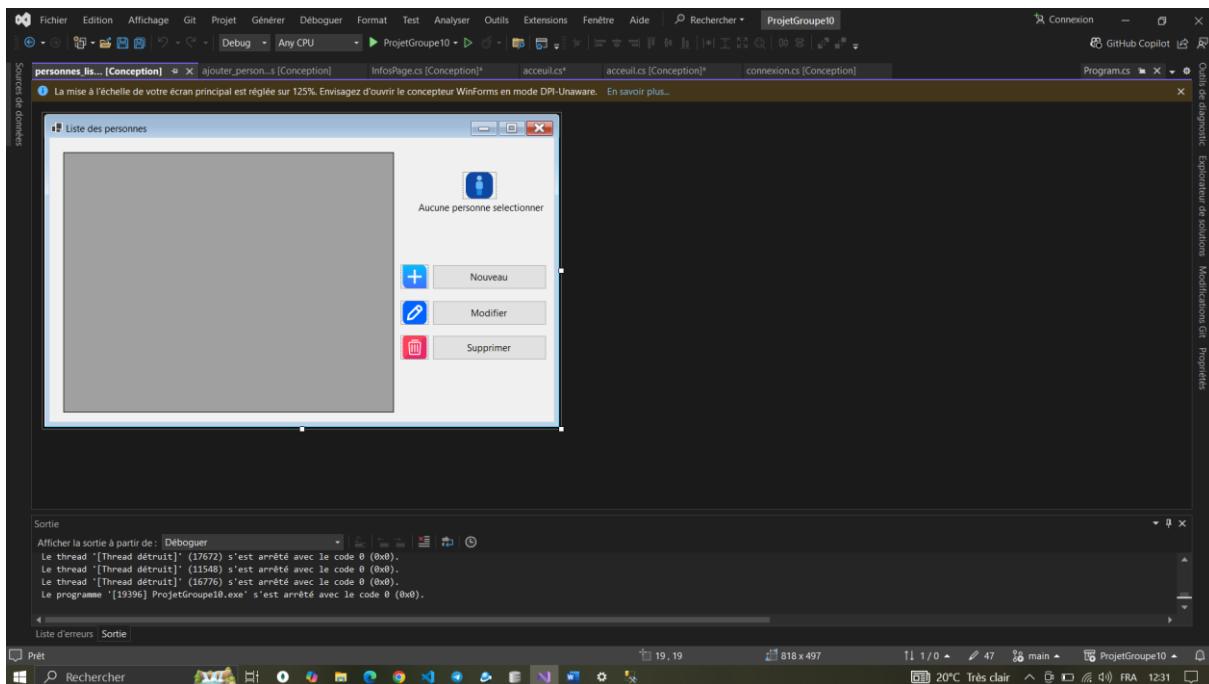
Création d'interface



Interface de la partie d'accueil



Interface nouvelle personne



interface d'affichage des personnes

Étape 4 : Connexion à la base de données

Un **code de connexion** est implémenté pour lier l'application à la base de données, avec des fonctions pour l'ouverture et la fermeture de la connexion. Un formulaire est également créé pour **se connecter manuellement à la base** via l'application, ce qui permet de tester différents paramètres de connexion.

The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** Ficher, Edition, Affichage, Git, Projet, Générer, Déboguer, Test, Analyser, Outils, Extensions, Fenêtre, Aide.
- Toolbar:** Standard icons for Open, Save, Build, Run, etc.
- Solution Explorer:** Shows the solution "ProjetGroupe10" with files like "DbManager.cs", "etat_de_sortie.cs", "personnes_list.cs", "ajouter_person...cs", "InfosPage.cs", "accueil.cs", and "connectionString(string user, string password)".
- Code Editor:** Displays the "DbManager.cs" code. The code implements a singleton pattern for a database manager. It includes properties for user and password, a private constructor, and a public static instance method. It also contains a connection string definition and a selectedPerson variable.
- Status Bar:** Shows the current line (Lig. : 21), character (Car. : 197), and mode (SPC CR LF).
- Task List:** Shows the "Sortie" (Output) task with the message: "Afficher la sortie à partir de : Déboguer".
- Bottom Status:** Shows the operating system status bar with icons for battery, signal, and language (FR). The temperature is listed as 20°C.

Code source de connexion de la base de donnée

```
// Connexion à la base
public bool Connect()
{
    if (connection.State == System.Data.ConnectionState.Closed)
    {
        try
        {
            connection.Open();
            Console.WriteLine("Connexion à la base de données réussie.");
            return true;
        }
        catch (SqlException ex)
        {
            Console.WriteLine($"Erreur lors de la connexion : {ex.Message}");
            MessageBox.Show(ex.Message
                , "Alerte", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Clipboard.SetText(ex.Message);
            return false;
        }
    }
    else
    {
        return true;
    }
}
```

Sortie

Afficher la sortie à partir de : Déboguer

Le programme '[3260] ProjetGroupe10.exe' s'est arrêté avec le code 0 (0x0).

code de connexion base de donnée

```

    // Déconnexion
    public bool Disconnect()
    {
        if (connection.State == System.Data.ConnectionState.Open)
        {
            try
            {
                connection.Close();
                Console.WriteLine("Connexion fermée.");
                return true;
            }
            catch (SqlException ex)
            {
                connection.Close();
                Console.WriteLine("Connexion fermée.");
                return true;
            }
        }
        else
        {
            return true;
        }
    }

    // Vérifie si la connexion est ouverte
    public bool IsConnected()

```

The screenshot shows the Visual Studio IDE interface with the code editor open. The code is written in C# and defines a `Disconnect` method that closes a database connection if it is open. It also contains a commented-out section for handling SQL exceptions. Below the code editor, the status bar shows the current line (Lig. 95) and column (Car. 14). The bottom of the screen shows the Windows taskbar with various icons.

Déconnexion base de données

Étape 5 : Programmation de la logique métier

Des classes sont créées pour gérer les entités :

- La classe Personne, avec les méthodes pour ajouter, mettre à jour, lire et supprimer.
- La classe Adresse, qui fonctionne de manière similaire.
- La classe Téléphone, pour gérer les numéros associés à chaque personne.

Chaque opération dans l'interface déclenche des méthodes de ces classes pour manipuler les données dans la base.

```

119     return false;
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }

public static string randomID(int taille = 14)
{
    const string caracteres = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    var random = new Random();
    var sb = new StringBuilder(taille);

    for (int i = 0; i < taille; i++)
    {
        int index = random.Next(caracteres.Length);
        sb.Append(caracteres[index]);
    }

    return sb.ToString();
}

```

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes Fichier, Edition, Affichage, Git, Projet, Générer, Déboguer, Test, Analyser, Outils, Extensions, Fenêtre, Aide, Rechercher, and Connexion. The title bar says "ProjetGroupe10". The main window displays the "Sources de données" (DbManager.cs) file. The code implements a static method "randomID" that generates a random string of a specified length (14 by default) using a random character set from uppercase letters and digits. The bottom status bar shows "Sortie" (Output), "Afficher la sortie à partir de : Déboguer", and "Le programme '[3268] ProjetGroupe10.exe' s'est arrêté avec le code 0 (0x0)". The taskbar at the bottom shows various application icons.

Générer Les Id Aléatoires

```

26     {
27         if(txtUserName.Text.Length > 0 && txtPassword.Text.Length > 0)
28         {
29             try {
30                 var con = DbManager.Instance;
31                 con.user = txtUserName.Text;
32                 con.password = txtPassword.Text;
33                 con.initialise();
34
35                 var result = con.Connect();
36                 if(result)
37                 {
38                     this.Close();
39                     MessageBox.Show("Connexion réussie", "Alerte", MessageBoxButtons.OK, MessageBoxIcon.None);
40                 }
41                 else
42                 {
43                     MessageBox.Show("Connexion échouée", "Alerte", MessageBoxButtons.OK, MessageBoxIcon.Warning);
44                 }
45             }
46             catch (Exception) {
47                 MessageBox.Show("Une erreur s'est produite ", "Alerte", MessageBoxButtons.OK, MessageBoxIcon.Warning);
48             }
49         }
50     }
51 }
52 }


```

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes Fichier, Edition, Affichage, Git, Projet, Générer, Déboguer, Test, Analyser, Outils, Extensions, Fenêtre, Aide, Rechercher, and Connexion. The title bar says "ProjetGroupe10". The main window displays the "Sources de données" (connexion.cs) file. The code handles user connection logic, specifically connecting to a database using the DbManager class. It checks if both the username and password fields are filled and then attempts to connect. If successful, it closes the current form and shows a success message box. If unsuccessful, it shows an error message box. The bottom status bar shows "Sortie" (Output), "Afficher la sortie à partir de : Déboguer", and "Le programme '[20328] ProjetGroupe10.exe' s'est arrêté avec le code 4294967295 (0xffffffff)". The taskbar at the bottom shows various application icons.

Formulaire de connexion a la base de données

Classe Personne

```

4     using System.Windows.Forms;
5
6     namespace ProjetGroupe10
7     {
8         11 références
9         internal class PersonneClasse
10        {
11             13 références
12             public string id { get; set; }
13             7 références
14             public string nom { get; set; }
15             7 références
16             public string post { get; set; }
17             3 références
18             public string prenom { get; set; }
19             7 références
20             public string sexe { get; set; }
21
22             17 références
23             public AdresseClasse adresse { get; set; }
24             11 références
25             public TelephoneClasse telephone { get; set; }
26
27             3 références
28             SqlConnection con => DbManager.Instance.Connection;
29             5 références
30             5 références
31             string tableName => "personnesTable";
32
33             1 référence
34             public bool AddOrUpdate()
35             {
36                 try
37                 {
38                     string query = @""
39                     IF EXISTS (SELECT 1 FROM " + tableName + " WHERE id = @id)
40                     : UPDATE " + tableName + @"
41                     : SET nom = @nom, post = @post, prenom = @prenom, sexe = @sexe
42                     WHERE id = @id
43                     ELSE
44                     INSERT INTO " + tableName + @" (id, nom, post, prenom, sexe)
45                     VALUES (@id, @nom, @post, @prenom, @sexe)";
46
47                     using (var cmd = new SqlCommand(query, con))
48                     {
49                         cmd.Parameters.AddWithValue("@id", id);
50                         cmd.Parameters.AddWithValue("@nom", nom);
51                         cmd.Parameters.AddWithValue("@post", post);
52                         cmd.Parameters.AddWithValue("@prenom", prenom);
53                         cmd.Parameters.AddWithValue("@sexe", sexe);
54
55                         int result = cmd.ExecuteNonQuery();
56                         return result != 0;
57                     }
58                 }
59                 catch (Exception ex)
60                 {
61                     MessageBox.Show("Erreur d'ajout ou mise à jour : " + ex.Message + ex.StackTrace.ToString(), "Erreur", MessageBoxButtons.OK);
62                     return false;
63                 }
64             }
65         }
66     }
67 
```

Ajout et mise à jour

```

21
22             1 référence
23             public bool AddOrUpdate()
24             {
25                 try
26                 {
27                     string query = @""
28                     IF EXISTS (SELECT 1 FROM " + tableName + " WHERE id = @id)
29                     : UPDATE " + tableName + @"
30                     : SET nom = @nom, post = @post, prenom = @prenom, sexe = @sexe
31                     WHERE id = @id
32                     ELSE
33                     INSERT INTO " + tableName + @" (id, nom, post, prenom, sexe)
34                     VALUES (@id, @nom, @post, @prenom, @sexe)";
35
36                     using (var cmd = new SqlCommand(query, con))
37                     {
38                         cmd.Parameters.AddWithValue("@id", id);
39                         cmd.Parameters.AddWithValue("@nom", nom);
40                         cmd.Parameters.AddWithValue("@post", post);
41                         cmd.Parameters.AddWithValue("@prenom", prenom);
42                         cmd.Parameters.AddWithValue("@sexe", sexe);
43
44                         int result = cmd.ExecuteNonQuery();
45                         return result != 0;
46                     }
47                 }
48                 catch (Exception ex)
49                 {
50                     MessageBox.Show("Erreur d'ajout ou mise à jour : " + ex.Message + ex.StackTrace.ToString(), "Erreur", MessageBoxButtons.OK);
51                     return false;
52                 }
53             }
54         }
55     }
56 
```

Lecture de toute la liste

The screenshot shows the Visual Studio IDE interface with the code editor open to the `PersonneClasse.cs` file. The code implements the `GetAll()` method, which retrieves all records from the database and returns them as a list of `PersonneClasse` objects. The code uses a `SqlDataReader` to read the results and map them to `PersonneClasse` instances.

```

56     public List<PersonneClasse> GetAll()
57     {
58         var personnes = new List<PersonneClasse>();
59         try
60         {
61             var query = "SELECT * FROM " + tableName;
62             using (var cmd = new SqlCommand(query, con))
63             using (var reader = cmd.ExecuteReader())
64             {
65                 while (reader.Read())
66                 {
67                     var p = new PersonneClasse()
68                     {
69                         id = reader.GetString(0),
70                         nom = reader.GetString(1),
71                         post = reader.GetString(2),
72                         prenom = reader.GetString(3),
73                         sexe = reader.GetString(4)
74                     };
75                     p.adresse = new AdresseClasse().GetByPerson(p.id);
76                     p.telephone = new TelephoneClasse().GetByPerson(p.id);
77                     personnes.Add(p);
78                 }
79             }
80         }
81         catch (Exception ex)
82         {
83             MessageBox.Show("Erreur de récupération : " + ex.Message, "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Warning);
84         }
85     }
86 }
87 
```

Suppression

The screenshot shows the Visual Studio IDE interface with the code editor open to the `PersonneClasse.cs` file. The code implements the `Delete()` method, which takes an ID as input and performs a delete operation on the database. It uses a `SqlCommand` with a parameterized query to ensure data integrity.

```

56     public bool Delete()
57     {
58         if(id.Length == 0)
59         {
60             return true;
61         }
62         try
63         {
64             var query = "DELETE FROM " + tableName + " WHERE id = @id";
65             using (var cmd = new SqlCommand(query, con))
66             {
67                 cmd.Parameters.AddWithValue("@id", id);
68                 cmd.ExecuteNonQuery();
69                 return true;
70             }
71         }
72         catch (Exception ex)
73         {
74             MessageBox.Show("Erreur de suppression : " + ex.Message, "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Warning);
75             return false;
76         }
77     }
78 }
79 
```

Classe Adresse : a les même fonctionnalités que la classe personne seulement le model change

```

15     public string ville { get; set; }
16     public string person { get; set; }
17
18     protected SqlConnection con => DbManager.Instance.Connection;
19     protected string tableName => "adresseTable";
20
21     public bool AddOrUpdate()
22     {
23         try
24         {
25             string query = @"IF EXISTS (SELECT 1 FROM " + tableName + " WHERE id = @id)
26             UPDATE " + tableName + @"
27             SET numero = @numero, avenue = @avenue, quartier = @quartier,
28                 commune = @commune, ville = @ville, person = @person
29             WHERE id = @id
30         ELSE
31             INSERT INTO " + tableName + @"
32             (@id, numero, avenue, quartier, commune, ville, person)
33             VALUES (@id, @numero, @avenue, @quartier, @commune, @ville, @person)";
34
35             using (var cmd = new SqlCommand(query, con))
36             {
37                 cmd.Parameters.AddWithValue("@id", id);
38                 cmd.Parameters.AddWithValue("@numero", numero);
39                 cmd.Parameters.AddWithValue("@avenue", avenue);
40                 cmd.Parameters.AddWithValue("@quartier", quartier);
41                 cmd.Parameters.AddWithValue("@commune", commune);
42                 cmd.Parameters.AddWithValue("@ville", ville);
43                 cmd.Parameters.AddWithValue("@person", person);
44
45                 int result = cmd.ExecuteNonQuery();
46                 return result != 0;
47             }
48         catch (Exception ex)
49         {
50             MessageBox.Show("Erreur d'ajout ou modification : " + ex.Message, "Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error);
51             return false;
52         }
53     }

```

Sortie

Afficher la sortie à partir de : Déboguer

Le programme '[23728] ProjetGroupe10.exe' s'est arrêté avec le code 0 (0x0).

Et la classe Téléphone

```

16     public string person { get; set; }
17     protected SqlConnection con => DbManager.Instance.Connection;
18     protected string tableName => "telephoneTable";
19     protected TelephoneClasse()
20     {
21         try
22         {
23             string query = @"IF EXISTS (SELECT 1 FROM " + tableName + " WHERE id = @id)
24             UPDATE " + tableName + @"
25             SET code = @code, numero = @numero, person = @person
26             WHERE id = @id
27         ELSE
28             INSERT INTO " + tableName + @" (@id, code, numero, person)
29             VALUES (@id, @code, @numero, @person)";
30
31             using (var cmd = new SqlCommand(query, con))
32             {
33                 cmd.Parameters.AddWithValue("@id", id);
34                 cmd.Parameters.AddWithValue("@code", code);
35                 cmd.Parameters.AddWithValue("@numero", numero);
36                 cmd.Parameters.AddWithValue("@person", person);
37
38                 int result = cmd.ExecuteNonQuery();
39                 return result != 0;
40             }
41         catch (Exception ex)
42         {
43             MessageBox.Show("Une erreur s'est produite : " + ex.Message, "Alerte", MessageBoxButtons.OK, MessageBoxIcon.Error);
44             return false;
45         }
46     }
47
48     public List<TelephoneClasse> getAll()
49     {
50         var telephones = new List<TelephoneClasse>();
51
52         return telephones;
53     }

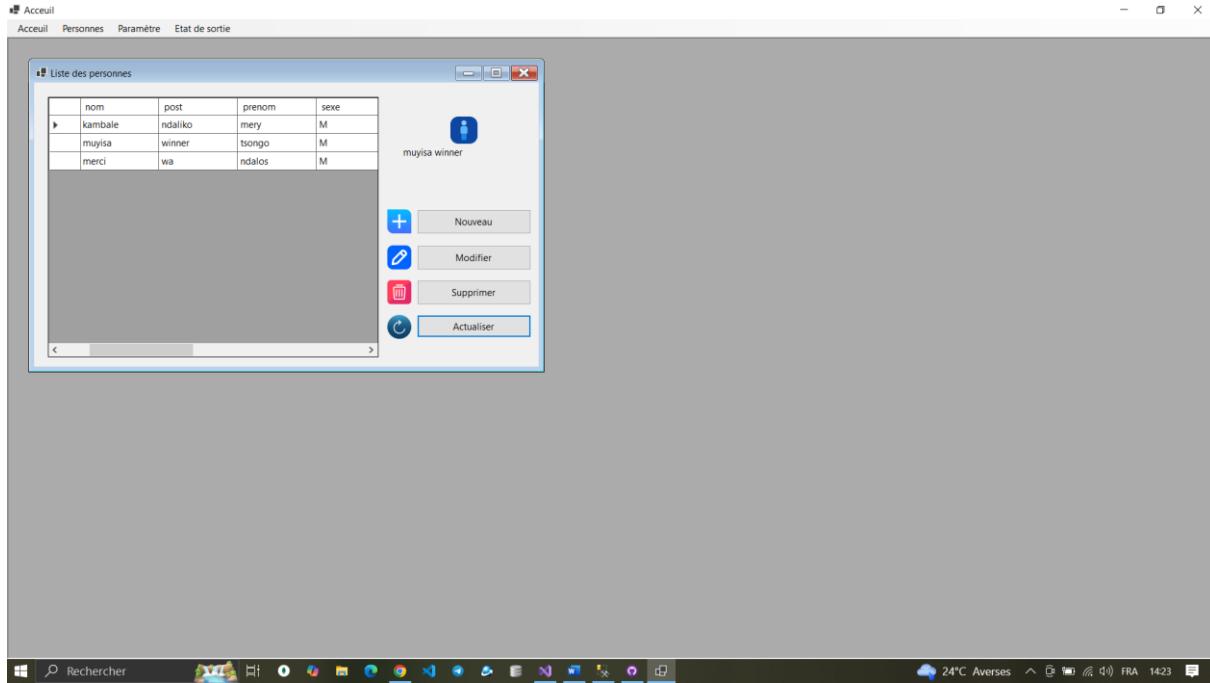
```

Sortie

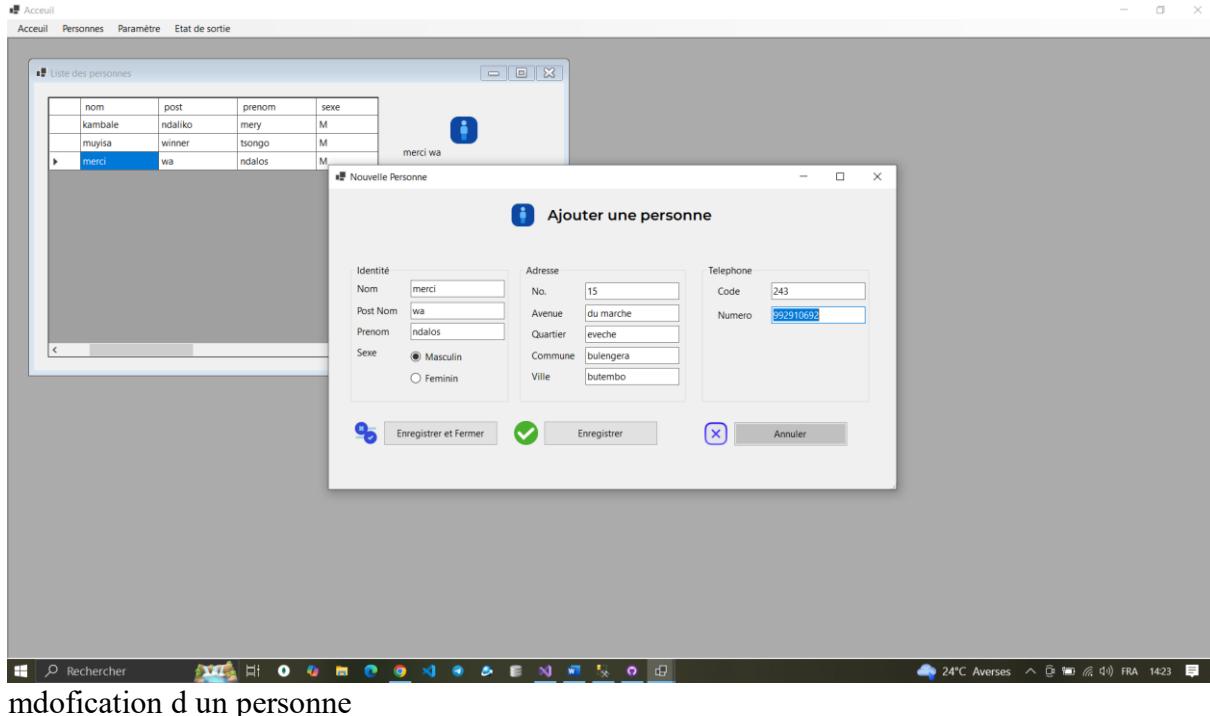
Afficher la sortie à partir de : Déboguer

Le programme '[23728] ProjetGroupe10.exe' s'est arrêté avec le code 0 (0x0).

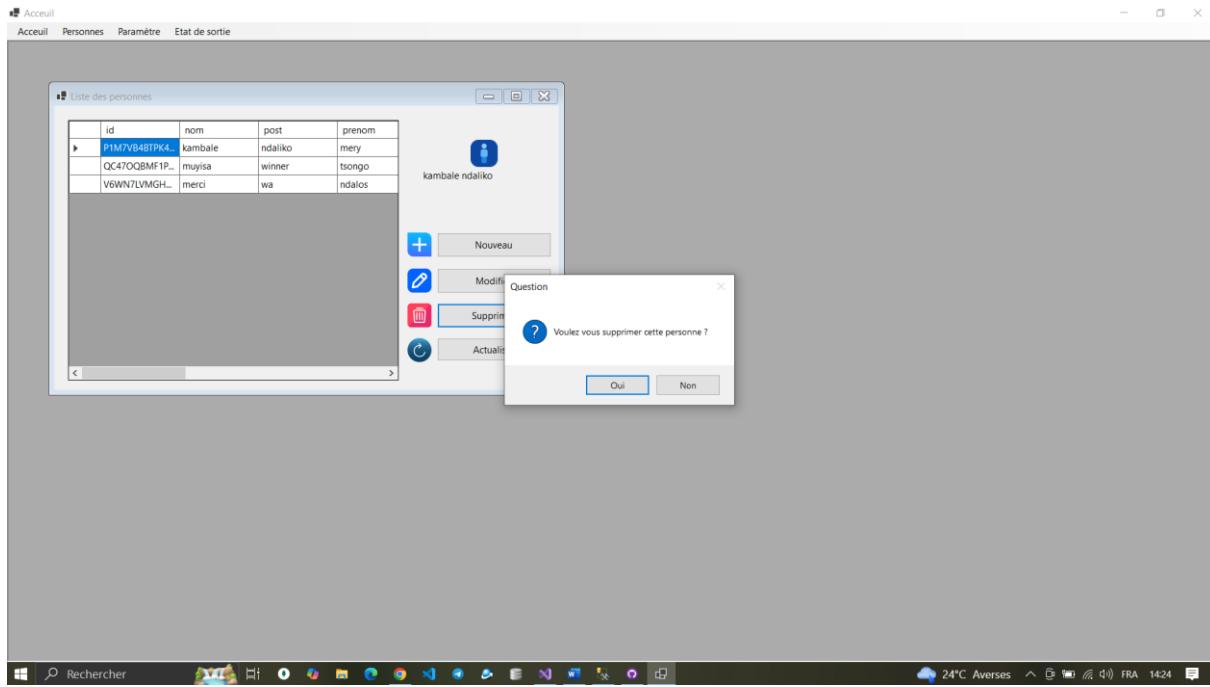
Lectures des personnes



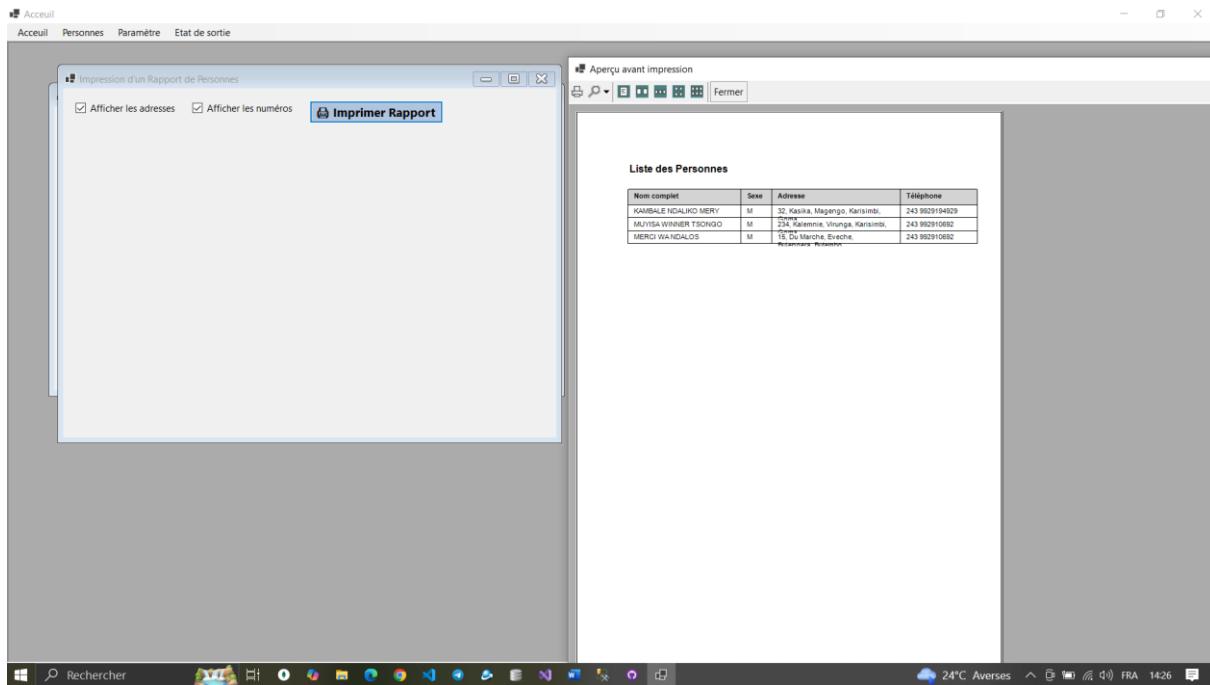
Modification d'une personne



mdofication d un personne



Etat de sortie : avec filtre d'affichage des personnes et des numéros de téléphone



Historique des Commit GitHub

Note : cet historique résume les grandes lignes ou les grandes étapes de la conception du logiciel.

The screenshot shows a GitHub pull request interface with the message "36 files changed +939 -93 lines changed". The main view is focused on the `DbManager.cs` file. The code editor displays several lines of C# code, with specific changes highlighted in red and green. The changes involve modifying the connection string logic to include a `TrustServerCertificate=True` parameter. The commit message at the bottom of the code editor reads: "@@ -102,7 +102,22 @@ public bool Disconnect()". The GitHub interface also shows other files like `AdresseClasse.cs`, `DbManager.cs`, and `PersonneClasse.cs` in the sidebar.

Fonctionnalités supplémentaires

Plusieurs fonctions avancées sont intégrées :

- **Génération automatique d'identifiants** aléatoires pour chaque personne.
- **Confirmation de suppression** via une boîte de dialogue.
- Lors de la suppression d'une personne, toutes ses données associées (adresse et numéros) sont également supprimées pour éviter les orphelins dans la base.
- **État de sortie filtré** : une fonctionnalité permettant d'afficher les personnes et leurs numéros de téléphone en fonction de certains critères.

Étape 7 : Suivi de développement

Tout au long du développement, l'historique des commits sur GitHub permet de suivre les grandes étapes de l'évolution du logiciel, documentant ainsi le processus de construction du projet.

Difficultés rencontrées

- **Problèmes de coordination** entre les membres du groupe, notamment pour la gestion du code partagé via GitHub.
- **Conception initiale de la base de données**, notamment pour établir des relations correctes entre les tables tout en assurant leur intégrité.
- **Gestion des erreurs de connexion à la base de données**, surtout en cas de mauvaises configurations.

- **Synchronisation entre l'interface graphique et la base de données**, nécessitant une bonne gestion des événements.
- **Suppression en cascade** délicate à mettre en œuvre pour éviter la perte involontaire de données.
- **Tests** : certaines fonctions ont dû être ajustées après des tests réels pour corriger des bugs liés à la logique métier ou à l'interface.