

Aplicație de Gestionare a Datelor Medicale cu MS SQL Server si .NET

Alecu Cedric-Ioachim
444A

CUPRINS

1	Introducere	4
1.1	Descrierea proiectului	4
1.2	Cerința temei.....	4
2	Tehnologii Folosite	5
2.1	MS SQL Server – Baza de date relațională	5
2.2	.NET Framework – Platforma de dezvoltare a aplicației	5
2.3	WPF (Windows Presentation Foundation) – Interfața grafică	5
2.4	ADO.NET – Conectarea aplicației cu baza de date.....	6
2.5	MS SQL Server Management Studio (SSMS) – Administrarea bazei de date	6
3	Arhitectura si Implementarea Aplicației	7
3.1	Structura Generală a Aplicației	7
3.2	Diagrama logica a bazei de date	7
3.3	Interfața Grafica a Utilizatorului (GUI)	8
3.3.1	Pagina de autentificare („Login Page”)	8
3.3.2	Meniul Principal „Admin”	9
3.3.3	Formular Medic.....	10
3.3.4	Formular Pacient.....	11
3.3.5	Formular Medicamente	12
3.3.6	Formular Consultatie.....	13
3.3.7	Meniu Principal “User normal”	14
3.3.8	Programările mele.....	15
3.3.9	Formular Actualizare Parola.....	16
3.4	Gestionarea conexiunii cu baza de date	17
3.4.1	Librăria utilizată pentru conexiune	17
3.4.2	Inițializarea conexiunii la baza de date	17
3.4.3	Utilizarea conexiunii în pagini individuale.....	17
3.4.4	Gestionarea conexiunilor în aplicație.....	18
3.5	Implementarea operațiunilor CRUD	18
3.5.1	Introducere	18
3.5.2	Citirea datelor (Read)	19
3.5.3	Adăugarea unui medic nou (Create)	18

3.5.4	Actualizarea datelor (Update)	20
3.5.5	Ștergerea unui medic (Delete)	21
3.6	Gestionarea securității utilizatorului	21
3.6.1	Introducere	21
3.6.2	Hasharea Parolelor si Utilizarea HASHBYTES('SHA2_256') in SQL Server	22
3.6.3	Autentificarea utilizatorului	22
3.6.4	Schimbarea parolei	23
3.6.5	Gestionarea rolurilor utilizatorilor	24
3.6.6	Protecția împotriva atacurilor SQL Injection.....	25
4	Concluzii	26
5	Bibliografie	27

1 Introducere

1.1 Descrierea proiectului

În era digitalizării accelerate, gestionarea eficientă a datelor medicale reprezintă o provocare esențială pentru instituțiile de sănătate. Proiectul de față își propune dezvoltarea unei aplicații informatice care să faciliteze managementul pacienților, consultațiilor și medicamentelor într-o unitate medicală, utilizând tehnologiile **MS SQL Server** pentru stocarea datelor și **.NET (WPF)** pentru interfața grafică a utilizatorului.

1.2 Cerința temei

Creați o aplicație care să conțină o baza de date creată în MS SQL SERVER și o interfață pentru aceasta. La crearea interfeței se va folosi tehnologia .NET.

Baza de date va fi compusă din următoarele tabele:

- Medic(MedicID, NumeMedic, prenumeMedic, Specializare);
- Pacient(PacientID, CNP, NumePacient, PrenumePacient, Adresa, Asigurare);
- Medicamente(MedicamentID, Denumire);

Asocierile între tabele sunt următoarele:

- Între tabela Medic și tabela Pacient – raport de cardinalitate M:N.
- Între tabela Pacient și tabela Medicament – raport de cardinalitate M:N. În acest caz, tabela de legătură se va numi Consultatie; astfel, pe lângă atributele necesare realizării multiplicității M:N, vor mai fi adăugate următoarele: Data, Diagnostic, DozaMedicament.

Interfața va trebui să permită utilizatorului să facă următoarele operații pe toate tabelele: vizualizare, adăugare, modificare, ștergere. Vizualizarea tabelelor de legătură va presupune vizualizarea datelor referite din celelalte tabele.

2 Tehnologii Folosite

2.1 MS SQL Server – Baza de date relațională

Microsoft SQL Server este soluția aleasă pentru gestionarea și stocarea datelor aplicației. Această bază de date relațională oferă:

- Fiabilitate și scalabilitate: Permite gestionarea eficientă a volumelor mari de date, fiind ideală pentru aplicațiile medicale.
- Integritate și securitate: Asigură protecția datelor pacienților prin mecanisme de autentificare și autorizare.
- Funcționalități avansate: Suportă proceduri stocate, vizualizări și trigger-e, facilitând operațiunile complexe asupra datelor.
- Compatibilitate cu .NET: Permite integrarea ușoară cu interfața aplicației prin conexiuni standard utilizând ADO.NET.

Rol în proiect:

- Stocarea datelor despre pacienți, medici, medicamente și consultații.
- Interogarea și manipularea eficientă a datelor.
- Asigurarea consistenței relațiilor prin utilizarea cheilor primare și străine.

2.2 .NET Framework – Platforma de dezvoltare a aplicației

.NET Framework este utilizat pentru a dezvolta aplicația de tip desktop, oferind o infrastructură robustă pentru crearea de aplicații performante și scalabile.

- Suport extins pentru interfețe grafice: Permite dezvoltarea interfeței grafice cu ajutorul WPF (Windows Presentation Foundation).
- Integrare ușoară cu SQL Server: Prin folosirea bibliotecilor integrate precum System.Data.SqlClient.
- Modularitate și mentenabilitate: Permite separarea logicii de business de interfața grafică.

Rol în proiect:

- Crearea aplicației desktop care permite utilizatorilor să vizualizeze, editeze și gestioneze datele din baza de date.
- Implementarea funcționalităților de login diferențiat (Admin și User).
- Comunicarea securizată cu baza de date pentru efectuarea operațiunilor CRUD (Create, Read, Update, Delete).

2.3 WPF (Windows Presentation Foundation) – Interfața grafică

WPF este tehnologia utilizată pentru dezvoltarea interfeței grafice a aplicației, oferind:

- Design modern și atractiv: Utilizarea XAML pentru crearea unei interfețe responsive și intuitive.
- Separarea logicii de UI: Utilizarea modelului MVVM (Model-View-ViewModel) pentru o mai bună organizare a codului.
- Personalizare extinsă: Suport pentru stiluri, template-uri și efecte vizuale.
- Binding avansat: Permite conectarea automată a elementelor UI la datele din baza de date.

Rol în proiect:

- Crearea unei interfețe prietenoase pentru utilizatorii finali (admin și utilizatori standard).
- Navigare intuitivă între diferite secțiuni (consultații, pacienți, medicamente).
- Validarea și gestionarea datelor în timp real.

2.4 ADO.NET – Conectarea aplicației cu baza de date

ADO.NET este folosit pentru comunicarea între aplicația WPF și baza de date SQL Server. Această tehnologie permite:

- Eficiență în manipularea datelor: Prin utilizarea obiectelor precum SqlConnection, SqlCommand, SqlDataAdapter.
- Execuția de interogări securizate: Utilizarea parametrilor pentru a preveni atacurile de tip SQL Injection.
- Gestionarea conexiunilor: Control optim al conexiunilor deschise și închise pentru performanță maximă.

Rol în proiect:

- Realizarea conexiunii între aplicație și baza de date.
- Executarea operațiunilor de inserare, actualizare și ștergere a datelor.
- Gestionarea și afișarea datelor în interfața utilizatorului.

2.5 MS SQL Server Management Studio (SSMS) – Administrarea bazei de date

SSMS este utilizat pentru gestionarea bazei de date pe durata dezvoltării proiectului, oferind:

- Interfață grafică intuitivă pentru administrarea și explorarea bazei de date.
- Scriere și testare a interogărilor SQL înainte de integrarea în aplicație.
- Monitorizare și optimizare a performanței bazei de date.

Rol în proiect:

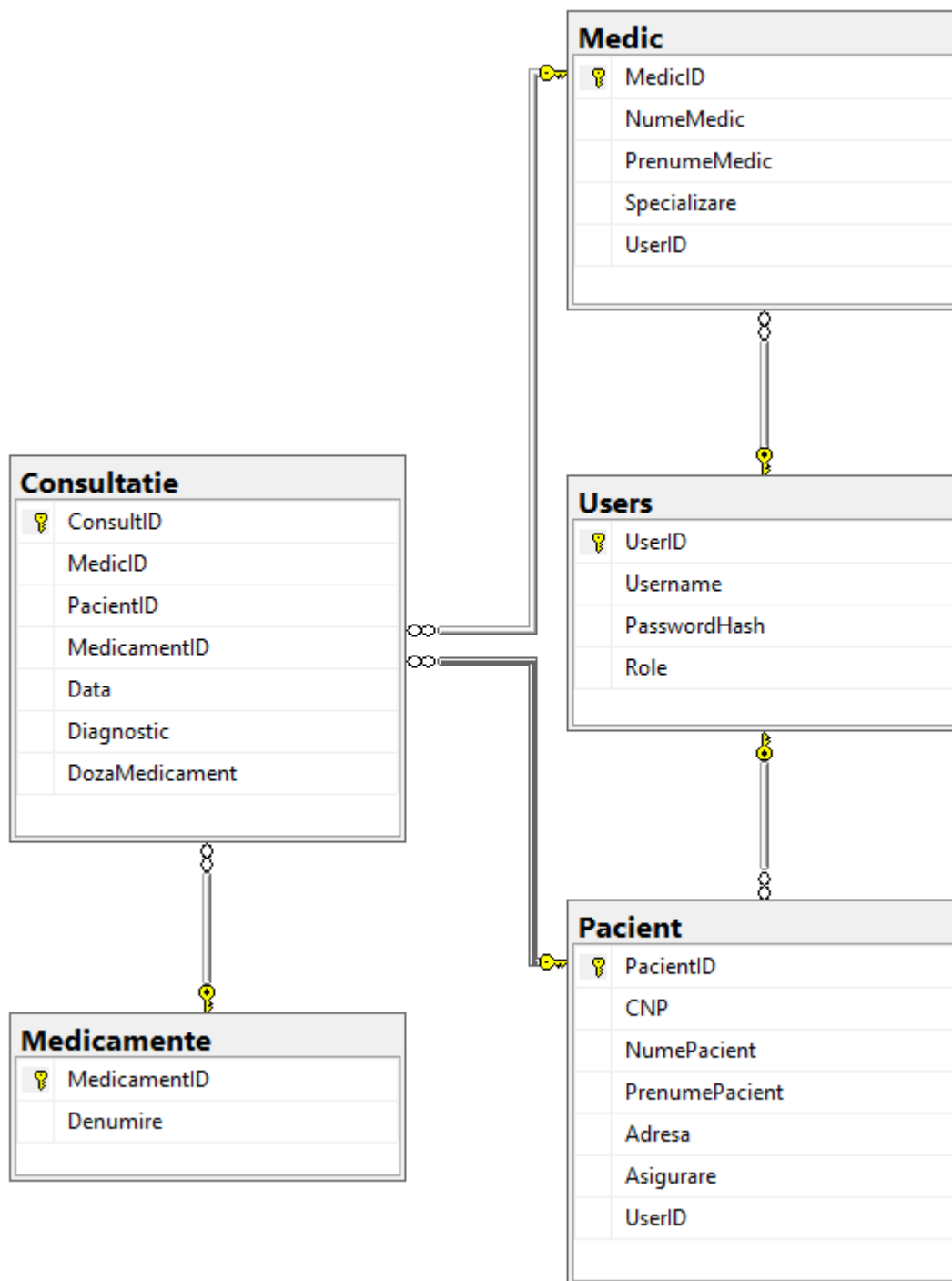
- Crearea și gestionarea tabelelor necesare aplicației.
- Verificarea consistenței datelor și a relațiilor dintre entități.
- Exportul și backup-ul bazei de date pentru siguranța datelor.

3 Arhitectura si Implementarea Aplicației

3.1 Structura Generală a Aplicației

Arhitectura aplicației este concepută pentru a asigura o gestionare eficientă a datelor medicale, oferind o interfață intuitivă pentru utilizatori și un mecanism robust de interacțiune cu baza de date. Soluția este structurată conform unui model client-server, unde interfața grafică dezvoltată în **.NET WPF** comunică cu baza de date **MS SQL Server** prin intermediul tehnologiei **ADO.NET**, facilitând astfel schimbul rapid și securizat de informații.

3.2 Diagrama logica a bazei de date



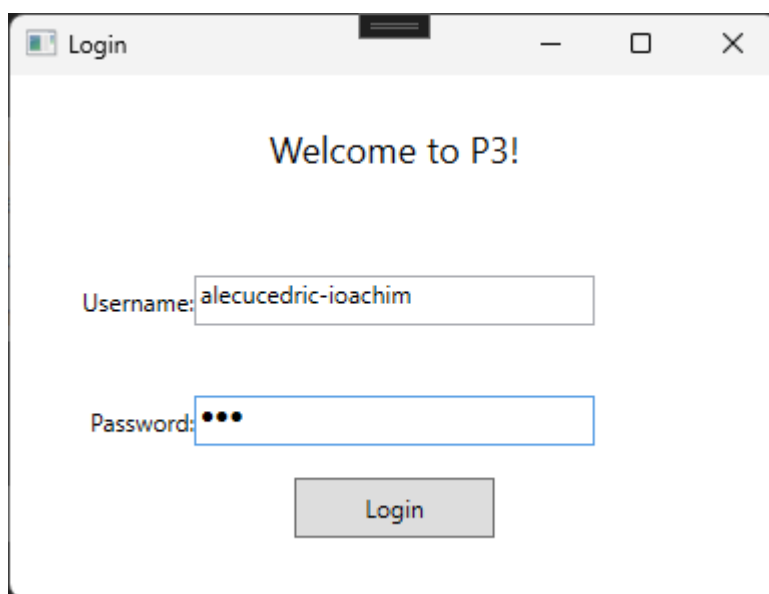
Figură 3.1 Diagrama bazei de date

Diagrama logică a bazei de date, generată în MySQL Server, oferă o reprezentare vizuală a structurii bazei de date utilizate în cadrul aplicației, denumită spitaldb. Aceasta include tabelele principale necesare pentru gestionarea eficientă a informațiilor medicale, și anume: Medic, Consultatie, Users, Medicamente și Pacient. Fiecare tabel este proiectat pentru a reflecta entitățile cheie din sistem, iar relațiile dintre ele sunt stabilite astfel încât să asigure integritatea și consistența datelor. În diagrama prezentată, sunt evidențiate conexiunile dintre tabele prin utilizarea cheilor primare și cheilor străine, facilitând astfel implementarea operațiunilor CRUD (Create, Read, Update, Delete) în cadrul aplicației. Această structură logică permite o organizare clară a datelor și o gestionare optimă a relațiilor dintre entitățile medicale, oferind suport pentru funcționalitățile aplicației.

3.3 Interfața Grafică a Utilizatorului (GUI)

Interfața grafică a utilizatorului reprezintă componenta principală prin care utilizatorii interacționează cu aplicația. Aceasta a fost proiectată pentru a fi intuitivă și ușor de utilizat, oferind acces rapid la funcționalitățile de gestionare a datelor medicale. În această secțiune vor fi prezentate principalele elemente de interfață, paginile aplicației și modul de navigare între acestea.

3.3.1 Pagina de autentificare („Login Page”)



Figură 3.2 Pagina de autentificare

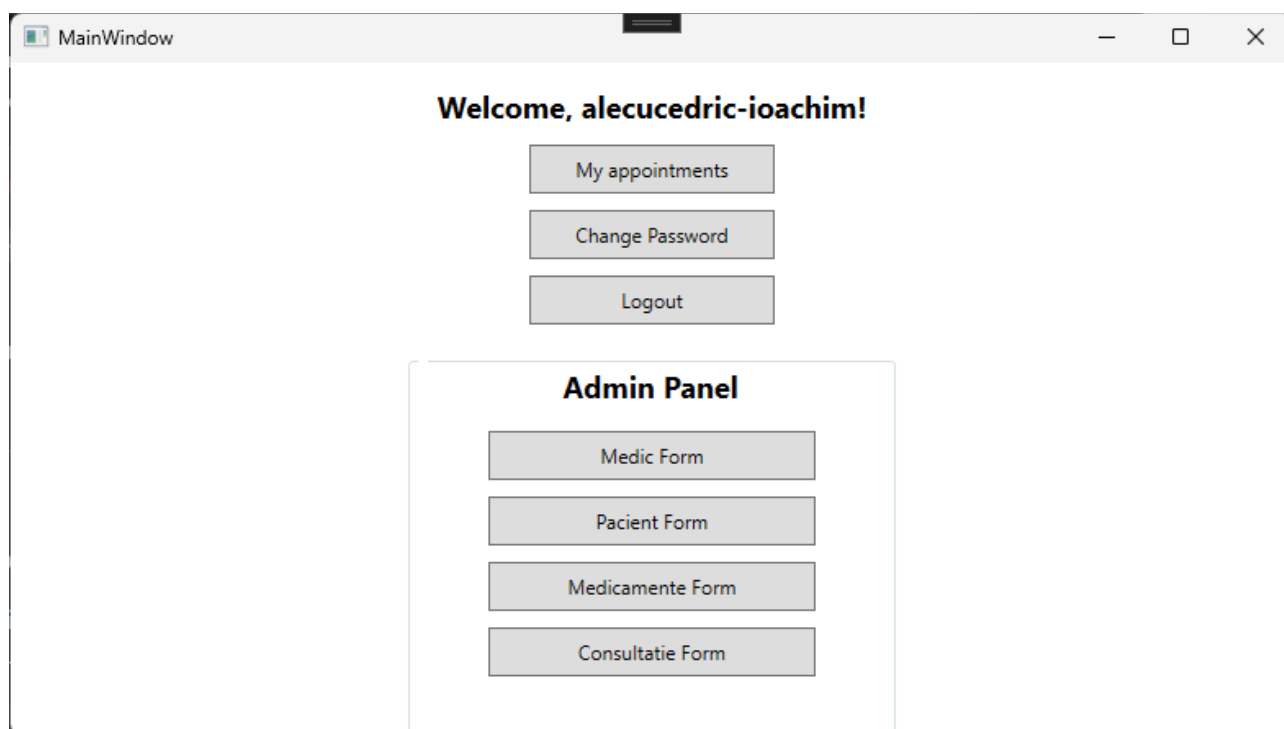
Pagina de autentificare reprezintă punctul de acces în aplicație și asigură securitatea datelor prin validarea utilizatorilor. Aceasta conține două câmpuri de introducere a datelor: **Username** și **Password**, unde utilizatorii trebuie să introducă datele lor de acces. Procesul de autentificare este declanșat prin apăsarea butonului **Login**, care verifică informațiile introduse și permite accesul în funcție de rolul utilizatorului.

Aplicația definește două tipuri de utilizatori:

- **Admin**, care are drepturi complete de gestionare a datelor, inclusiv adăugarea, editarea și ștergerea înregistrărilor.
- **User**, care are acces limitat, putând vizualiza doar informațiile relevante pentru propriul profil, cum ar fi programările la consultații.

Această pagină este esențială pentru menținerea confidențialității și securității datelor din sistem, asigurând acces diferențiat în funcție de permisiuni.

3.3.2 Meniul Principal „Admin”



Figură 3.3 Meniul Principal "Admin"

Meniul principal destinat utilizatorilor cu rol de Admin oferă acces rapid și intuitiv la funcționalitățile esențiale pentru gestionarea aplicației. Pagina conține mai multe butoane organizate în două secțiuni principale:

1. Funcționalități generale:

- **My Appointments:** permite administratorului să vizualizeze programările disponibile în sistem.
- **Change Password:** oferă posibilitatea de a actualiza parola contului pentru securitate sporită.
- **Logout:** deconectează utilizatorul din aplicație și revine la pagina de autentificare.

2. Panoul de administrare (Admin Panel):

Această secțiune permite gestionarea datelor esențiale ale aplicației prin următoarele formulare:

- **Medic Form:** gestionarea informațiilor despre medici (adăugare, editare, ștergere).
- **Pacient Form:** administrarea datelor pacienților, inclusiv detalii personale și istoricul medical.
- **Medicamente Form:** gestionarea bazei de date a medicamentelor disponibile.
- **Consultatie Form:** organizarea și administrarea consultațiilor pentru pacienți.

Meniul principal este conceput pentru a oferi administratorului un control complet asupra datelor din aplicație, asigurând un flux de lucru eficient și acces rapid la toate modulele importante.

3.3.3 Formular Medic

MedicID	NumeMedic	PrenumeMedic	Specializare	UserID
1	Alecu	Cedric-loachim	Doctor in Inginerie	1
2	Ionescu	Catalina	Doctor in laborator de METC	10
3	Nedelcu	Antonia	Catan	11

Figură 3.4 Formular Medic

Formularul Medic permite gestionarea eficientă a informațiilor despre personalul medical din cadrul aplicației. În această pagină, utilizatorii pot introduce și actualiza datele medicilor prin completarea următoarelor câmpuri:

- **Nume** – permite introducerea numelui medicului.
- **Prenume** – câmp destinat introducerii prenumelui medicului.
- **Specializare** – specifică domeniul de specialitate al medicului.

Formularul include o serie de butoane care facilitează gestionarea datelor:

- **Add** (Adaugă): permite introducerea unui nou medic în baza de date.
- **Edit** (Editează): oferă posibilitatea de a modifica informațiile unui medic existent.
- **Delete** (Șterge): elimină un medic selectat din listă.
- **Refresh** (Reîmprospătare): actualizează lista afișată pentru a reflecta ultimele modificări.
- **Done** (Gata): finalizează interacțiunea cu acest formular și permite revenirea la meniul anterior.

În partea principală a formularului sunt afișate datele existente din tabela Medic, oferind o vizualizare clară și ușor de utilizat a tuturor înregistrărilor disponibile. Această secțiune permite utilizatorilor să selecteze rapid un medic pentru a efectua modificări sau ștergeri.

3.3.4 Formular Pacient

Pacient

CNP: 6250110018217 Adresa: Bucuresti

Nume: Popescu Asigurare: BCR Asigurari de Viata

Prenume: Pop

PacientID	CNP	NumePacient	PrenumePacient	Adresa	Asigurare	UserID
2	5764802357023	Nicolae	Stefan-Alex	Ploiesti	Rugaciuni	5
4	4654364356435	pac	ient1	acasa	nema	14
5	6250110018217	Popescu	Pop	Bucuresti	BCR Asigurari de Viata	15

Add Edit Delete Refresh Done

Figură 3.5 Formular Pacient

Formularul Pacient permite gestionarea detaliată a informațiilor despre pacienți, oferind o interfață intuitivă pentru adăugarea și actualizarea datelor personale. În cadrul acestui formular, utilizatorii pot introduce și edita următoarele câmpuri esențiale:

- **CNP** (Cod Numeric Personal): identificator unic al pacientului, utilizat pentru evidența medicală.
- **Nume**: permite introducerea numelui de familie al pacientului.
- **Prenume**: câmp destinat introducerii prenumelui pacientului.
- **Adresa**: secțiune pentru completarea adresei de domiciliu a pacientului.
- **Asigurare**: tipul de asigurare medicală deținută de pacient (privată/stat).

Formularul include aceleași butoane funcționale pentru gestionarea datelor pacienților, și anume:

- **Add** (Adaugă): permite înregistrarea unui nou pacient în baza de date.
- **Edit** (Editează): oferă posibilitatea de a modifica datele unui pacient existent.
- **Delete** (Șterge): elimină pacientul selectat din listă.
- **Refresh** (Reîmprospătare): actualizează lista afișată pentru a reflecta ultimele modificări.
- **Done** (Gata): închide formularul și revine la meniul principal al aplicației.

În cadrul paginii sunt afișate toate înregistrările existente din tabela Pacient, permițând utilizatorilor să vizualizeze și să gestioneze rapid datele pacienților.

3.3.5 Formular Medicamente

MedicamentID	Denumire
8	Inginerocalmin
9	Licentocalmin
10	Ceas destept
11	Paracetamol
12	Ibuprofen
13	Aspirina
14	Diclofenac

Figură 3.6 Formular Medicamente

Formularul Medicamente permite gestionarea informațiilor referitoare la medicamentele disponibile în sistem. Acesta oferă o interfață simplă și intuitivă, permițând utilizatorilor să introducă și să administreze doar **numele medicamentului**, fără alte detalii suplimentare.

Prin intermediul acestui formular, utilizatorii pot efectua următoarele acțiuni:

- **Add** (Adaugă): permite introducerea unui nou medicament în baza de date.
- **Edit** (Editează): oferă posibilitatea de a modifica numele unui medicament existent.
- **Delete** (Șterge): elimină un medicament selectat din listă.
- **Refresh** (Reîmprospătare): actualizează lista afișată pentru a reflecta cele mai recente modificări.
- **Done** (Gata): finalizează interacțiunea cu acest formular și permite revenirea la meniul principal.

În cadrul formularului, datele sunt afișate într-o listă, facilitând identificarea și selectarea rapidă a înregistrărilor existente.

3.3.6 Formular Consultatie

Consultatie

Data: 15/01/2025 15 Medic: Ionescu Ion

Diagnostic: Raceala si gripa Pacient: Popescu Pop

DozaMedicament: 500 mg de 3 ori pe zi, timp de 5 zile Medicamente: Paracetamol

	Pacient	Data	Diagnostic
ic-loachim	Nicolae Stefan-Alex	1/17/2025 12:00:00 AM	Nota mica la Microun
ic-loachim	Nicolae Stefan-Alex	1/24/2025 12:00:00 AM	t43t43
ic-loachim	Nicolae Stefan-Alex	1/17/2025 12:00:00 AM	fews
n	Popescu Pop	1/15/2025 12:00:00 AM	Raceala si gripa

< >

Add Edit Delete Refresh Done

Figură 3.7 Formular Consultatie

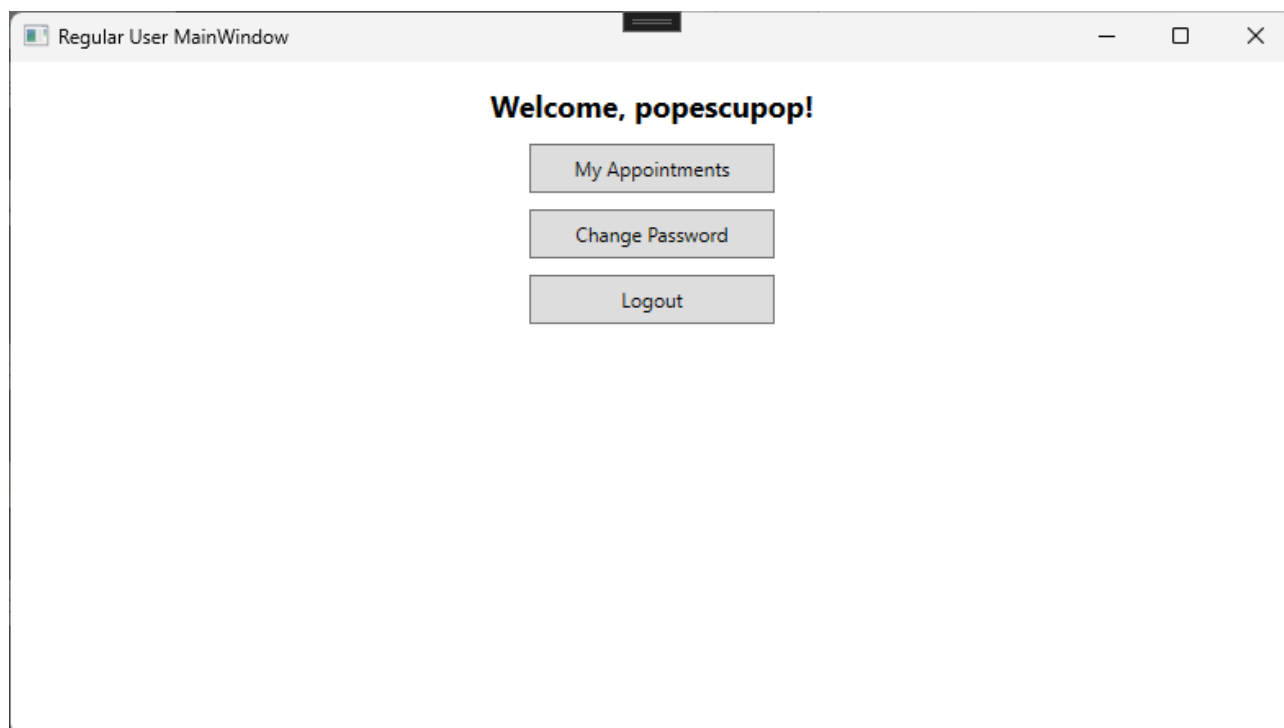
Formularul Consultație permite vizualizarea și editarea datelor din tabelul de legătură dintre pacienți, medici și medicamente. Acesta oferă o interfață intuitivă pentru gestionarea consultațiilor medicale, permițând utilizatorilor să introducă și să actualizeze informații esențiale.

În cadrul acestui formular, utilizatorii pot completa următoarele câmpuri:

- **Data consultației:** Este introdusă printr-un selector de dată specific WPF, care permite selectarea facilă a datei consultației.
- **Diagnostic:** Câmp text destinat introducerii diagnosticului stabilit de medic în urma consultației.
- **Doza medicament:** Specificarea dozei recomandate pentru tratamentul pacientului.
- **Medic:** Selectarea medicului responsabil dintr-un **meniu drop-down**, care conține lista medicilor disponibili.
- **Pacient:** Alegerea pacientului dintr-un **meniu drop-down**, populat cu datele pacienților înregistrați în sistem.
- **Medicamente:** Selectarea medicamentelor prescrise printr-un **meniu drop-down**, care permite alegerea din lista de medicamente disponibile.

Formularul include și butoanele standard pentru gestionarea datelor.

3.3.7 Meniu Principal “User normal”



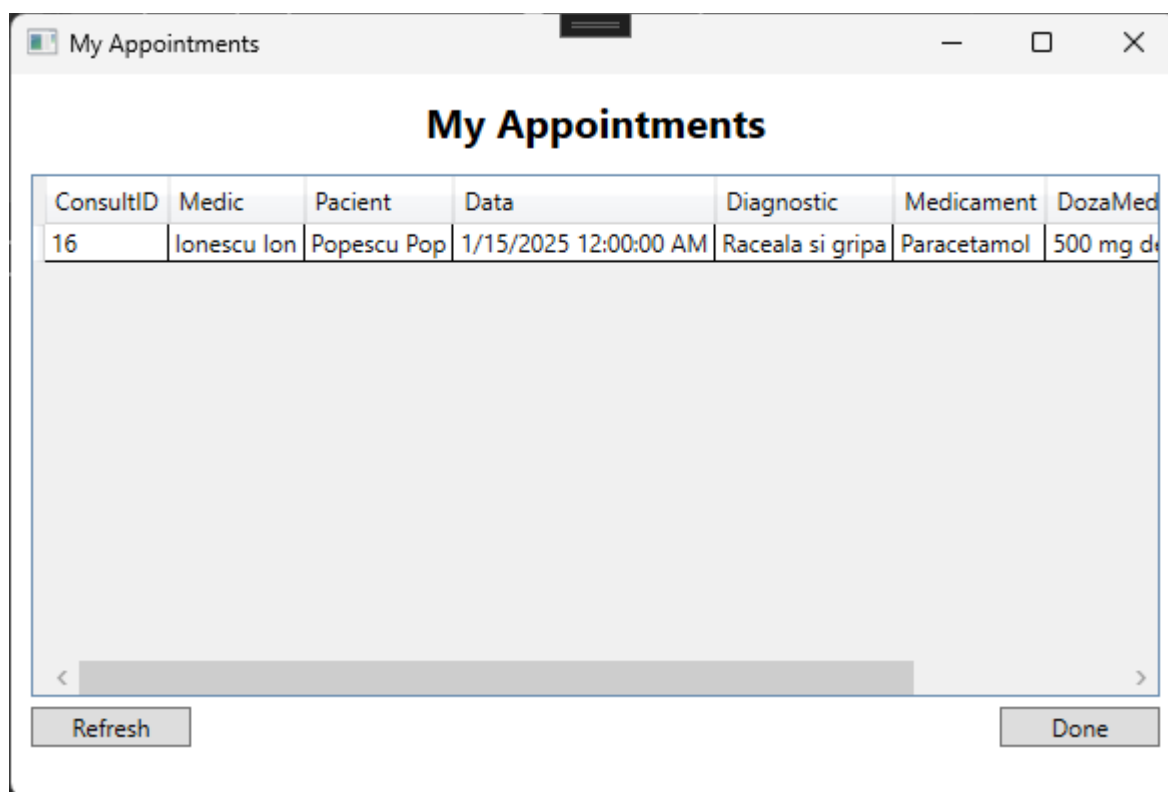
Figură 3.8 Meniu Principal "User"

Meniul principal destinat utilizatorilor de tip User normal oferă o interfață simplificată, concepută pentru a permite accesul la funcționalitățile esențiale pentru pacienți. Spre deosebire de meniul dedicat administratorilor, acesta conține doar opțiunile necesare pentru gestionarea datelor personale și a programărilor.

Pagina include următoarele butoane:

- **My Appointments** (Programările mele): Permite utilizatorului să vizualizeze programările existente, inclusiv detalii precum data, medicul și diagnosticul aferent. Această secțiune oferă pacienților o imagine clară asupra istoricului și a programărilor viitoare.
- **Change Password** (Schimbare Parolă): Oferă utilizatorului posibilitatea de a-și modifica parola contului pentru a menține securitatea accesului la datele personale.
- **Logout** (Deconectare): Permite utilizatorului să se deconecteze în siguranță din aplicație și să revină la pagina de autentificare.

3.3.8 Programările mele



Figură 3.9 Programările mele

Formularul „Programările Mele” oferă utilizatorilor autentificați, fie că sunt medici sau pacienți, posibilitatea de a vizualiza programările asociate contului lor. Acesta asigură un acces rapid și eficient la informațiile relevante, fără a permite modificarea datelor, garantând astfel integritatea și securitatea acestora.

Utilizatorii pot accesa detalii despre programările efectuate, cum ar fi:

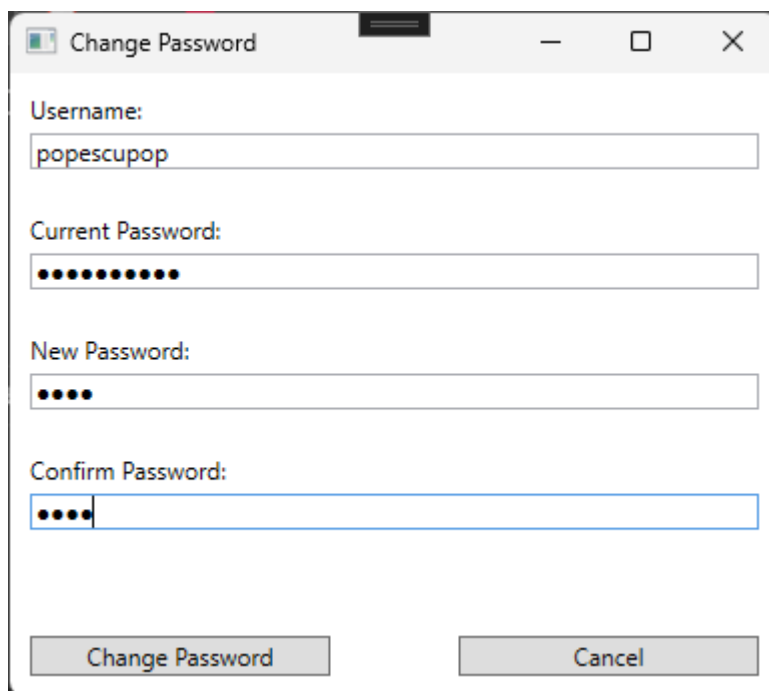
- Data și ora consultației;
- Numele medicului sau pacientului asociat;
- Diagnosticul (dacă este cazul);
- Medicația prescrisă.

Sistemul afișează automat doar programările aferente utilizatorului autentificat, eliminând astfel accesul la informații neautorizate.

Butoane disponibile:

- **Refresh** (Reîmprospătare): Actualizează lista programărilor pentru a reflecta eventualele modificări recente din baza de date.
- **Done** (Gata): Permite utilizatorului să închidă formularul și să revină la meniul principal.

3.3.9 Formular Actualizare Parola



Figură 3.10 Formular Actualizare Parola

Formularul „Schimbare Parolă” permite utilizatorilor autentificați să își modifice parola contului pentru a asigura securitatea și confidențialitatea accesului la aplicație. Acesta oferă o interfață simplă și intuitivă, prin care utilizatorii pot introduce datele necesare pentru actualizarea parolei.

Câmpurile disponibile:

- **Username** (Nume utilizator): Este un câmp afișat automat pentru identificarea utilizatorului autentificat. Acesta nu poate fi modificat, fiind utilizat pentru confirmarea identității.
- **Current Password** (Parola curentă): Utilizatorul trebuie să introducă parola existentă pentru verificarea identității înainte de schimbarea acesteia.
- **New Password** (Parola nouă): Câmp destinat introducerii unei noi parole care trebuie să respecte cerințele de securitate ale aplicației (ex: lungime minimă, caractere speciale etc.).
- **Confirm Password** (Confirmare parolă): Utilizatorul trebuie să reintroducă noua parolă pentru a evita eventualele greșeli de tastare. Parola introdusă trebuie să coincidă cu cea din câmpul „New Password”.

Butoanele disponibile:

- **Change Password** (Schimbă Parola): Validează datele introduse și, dacă sunt corecte, actualizează parola utilizatorului în baza de date. În cazul unor erori (parolă curentă incorectă, neconcordanță între parole), se afișează un mesaj corespunzător.
- **Cancel** (Anulare): Permite utilizatorului să renunțe la schimbarea parolei și să revină la meniul principal fără a efectua modificări.

Formularul „Schimbare Parolă” contribuie la securizarea conturilor utilizatorilor și oferă o modalitate simplă și eficientă de actualizare a datelor de autentificare, prevenind accesul neautorizat la aplicație.

3.4 Gestionarea conexiunii cu baza de date

Aplicația utilizează tehnologia **.NET (WPF)** și baza de date **MS SQL Server** pentru stocarea și gestionarea informațiilor. Conexiunea la baza de date este realizată folosind biblioteca **ADO.NET**, care permite comunicarea directă cu baza de date prin obiecte specifice, cum ar fi **SqlConnection** și **SqlCommand**.

3.4.1 Librăria utilizată pentru conexiune

Microsoft.Data.SqlClient, care este o versiune modernă și recomandată a provider-ului pentru conexiunile SQL.

```
using Microsoft.Data.SqlClient;
```

Figură 3.11 Cod exemplu din Medic.xaml.cs

Funcționalități principale oferite:

- Conexiunea la baze de date SQL Server prin clasa **SqlConnection**.
- Executarea interogărilor SQL folosind **SqlCommand**.
- Manipularea datelor prin **SqlDataAdapter** și **DataReader**.
- Securitate îmbunătățită prin suport pentru criptare și autentificare modernă.

3.4.2 Inițializarea conexiunii la baza de date

Conexiunea cu baza de date este gestionată prin intermediul clasei **DatabaseHelper**, care conține metoda **GetConnection()** pentru furnizarea unui obiect **SqlConnection** configurat cu un **connection string** corespunzător.

```
internal class DatabaseHelper
{
    private string connectionString = "Server=G713RS;Database=spitaldb;Trusted_Connection=True;";

    0 references
    public SqlConnection GetConnection() {
        return new SqlConnection(connectionString);
    }
}
```

Figură 3.12 Cod exemplu din DatabaseHelper.cs

Acest cod creează o conexiune nouă la baza de date folosind un connection string, care conține informațiile necesare pentru a stabili conexiunea cu serverul:

- **Server = G713RS** - Numele serverului de baze de date.
- **Database = spitaldb** - Numele bazei de date utilizate.
- **Trusted_Connection = True** - Folosește autentificarea Windows.

3.4.3 Utilizarea conexiunii în pagini individuale

În cadrul aplicației, fiecare fereastră WPF, precum **Medic.xaml.cs**, gestionează propria conexiune prin inițializarea unui obiect **SqlConnection**. De exemplu, în metoda **LoadData()**, conexiunea este deschisă pentru a prelua datele din tabelul **Medic**, utilizând un **SqlDataAdapter** pentru încărcarea datelor într-un **DataTable**.

```

private void LoadData()
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        try
        {
            conn.Open();
            SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM Pacient", conn);
            DataTable dataTable = new DataTable();
            dataAdapter.Fill(dataTable);
            dataGridPacient.ItemsSource = dataTable.DefaultView;
            conn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
        finally
        {
            conn.Close();
        }
    }
}

```

Figură 3.13 Cod exemplu din Pacient.xaml.cs

În acest fragment de cod:

- Se deschide conexiunea cu baza de date folosind `conn.Open()`.
- Se folosește un `SqlDataAdapter` pentru a executa o interogare SQL (`SELECT * FROM Medic`).
- Datele sunt populate într-un `DataTable` și afișate în interfața grafică.
- Conexiunea este închisă în blocul `finally` pentru a preveni scurgeri de resurse.

3.4.4 Gestionarea conexiunilor în aplicație

Pentru a asigura o bună practică în utilizarea conexiunilor la bazele de date, aplicația folosește instrucțiunea `using`, care garantează eliberarea conexiunii odată ce operațiunea este finalizată.

Beneficiile acestei abordări:

- Prevenirea scurgerilor de conexiuni.
- Îmbunătățirea performanței prin eliberarea resurselor la timp.
- Asigurarea unui cod mai clar și ușor de întreținut.

3.5 Implementarea operațiunilor CRUD

3.5.1 Introducere

Operațiunile **CRUD** (**Create**, **Read**, **Update**, **Delete**) sunt esențiale pentru gestionarea datelor în aplicație. În cadrul acestei secțiuni, vom analiza implementarea acestor operațiuni asupra tabelului `Medic`, care permite gestionarea informațiilor despre medici, cum ar fi numele, prenumele și specializarea.

3.5.2 Adăugarea unui medic nou (Create)

Pentru a adăuga un nou medic în baza de date, aplicația utilizează metoda `AddButton_Click()`. Aceasta inserează datele în tabelul `Users`, obține ID-ul nou generat și apoi inserează în tabelul `Medic`.

```

// Step 1: Insert into Users table
string userQuery = "INSERT INTO Users (Username, PasswordHash, Role) " +
    "VALUES (@Username, HASHBYTES('SHA2_256', @Password), @Role); SELECT SCOPE_IDENTITY()";
using (SqlCommand cmdUser = new SqlCommand(userQuery, conn))
{
    // Generate username and password
    string username = (txtNumeMedic.Text + txtPrenumeMedic.Text).ToLower();
    string password = username; // Initial password is the same as the username

    cmdUser.Parameters.AddWithValue("@Username", username);
    cmdUser.Parameters.AddWithValue("@Password", password);
    cmdUser.Parameters.AddWithValue("@Role", "Admin");

    try
    {
        // Execute and get the new UserID
        int userId = Convert.ToInt32(cmdUser.ExecuteScalar());

        // Step 2: Insert into Medic table
        string medicQuery = "INSERT INTO Medic (UserID, NumeMedic, PrenumeMedic, Specializare) " +
            "VALUES (@UserID, @NumeMedic, @PrenumeMedic, @Specializare)";
        using (SqlCommand cmdMedic = new SqlCommand(medicQuery, conn))
        {
            cmdMedic.Parameters.AddWithValue("@UserID", userId);
            cmdMedic.Parameters.AddWithValue("@NumeMedic", txtNumeMedic.Text);
            cmdMedic.Parameters.AddWithValue("@PrenumeMedic", txtPrenumeMedic.Text);
            cmdMedic.Parameters.AddWithValue("@Specializare", txtSpecializare.Text);

            cmdMedic.ExecuteNonQuery();
        }

        MessageBox.Show("Medic added successfully!");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
}

```

Figură 3.14 Cod exemplu Adăugare din Medic.xaml.cs

Explicație cod:

- Se inserează un utilizator în tabelul Users, apoi se folosește SCOPE_IDENTITY() pentru a prelua ID-ul generat.
- Se inserează apoi medicul în tabelul Medic cu ID-ul utilizatorului.
- Se utilizează parametri pentru a preveni atacurile de tip SQL injection.

3.5.3 Citirea datelor (Read)

Operațiunea de citire permite încărcarea și afișarea datelor existente în tabelul Medic. Aceasta este realizată în metoda LoadData(), care utilizează un SqlDataAdapter pentru preluarea și afișarea datelor într-un DataGridView.

```

private void LoadData()
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        try
        {
            conn.Open();
            SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM Medic", conn);
            DataTable dataTable = new DataTable();
            dataAdapter.Fill(dataTable);
            dataGridMedic.ItemsSource = dataTable.DefaultView;
            conn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
        finally
        {
            conn.Close();
        }
    }
}

```

Figură 3.15 Cod exemplu Citire din Medic.xaml.cs

Explicație cod:

- Se creează un obiect SqlConnection pentru a stabili conexiunea.
- SqlDataAdapter execută interogarea SELECT * FROM Medic pentru a prelua toate datele.
- Datele sunt încărcate într-un obiect DataTable și sunt afișate în interfață prin dataGridMedic.ItemsSource.

3.5.4 Actualizarea datelor (Update)

Modificarea unui medic existent în baza de date se realizează în metoda EditButton_Click(), care preia valorile din câmpurile formularului și le actualizează în baza de date.

```

private void EditButton_Click(object sender, RoutedEventArgs e)
{
    // Check if a row is selected in the DataGrid
    if (dataGridMedic.SelectedItem != null)
    {
        DataRowView row = (DataRowView)dataGridMedic.SelectedItem;

        // Get the MedicID (primary key) of the selected record
        long medicID = (long)row["MedicID"];

        //SQL query for updating the Medic table
        string query = "UPDATE Medic SET NumeMedic = @NumeMedic, " +
            "PrenumeMedic = @PrenumeMedic, Specializare = @Specializare WHERE MedicID = @MedicID";

        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand(query, conn);

            // Add parameters for the SQL query
            cmd.Parameters.AddWithValue("@MedicID", medicID);
            cmd.Parameters.AddWithValue("@NumeMedic", txtNumeMedic.Text);
            cmd.Parameters.AddWithValue("@PrenumeMedic", txtPrenumeMedic.Text);
            cmd.Parameters.AddWithValue("@Specializare", txtSpecializare.Text);
        }
    }
}

```

Figură 3.16 Cod exemplu Actualizarea datelor din Medic.xaml.cs

Explicație cod:

- Se selectează rândul dorit din DataGrid.
- Se preia MedicID pentru identificarea înregistrării.
- Se utilizează o interogare parametrizată pentru actualizarea datelor.

3.5.5 Ștergerea unui medic (Delete)

Ștergerea unei înregistrări este gestionată în metoda DeleteButton_Click(), care permite eliminarea unui medic selectat.

```
private void DeleteButton_Click(object sender, RoutedEventArgs e)
{
    // Check if a row is selected in the DataGrid
    if (dataGridMedic.SelectedItem != null)
    {
        DataRowView row = (DataRowView)dataGridMedic.SelectedItem;

        // Get the MedicID (primary key) of the selected record
        long medicID = (long)row["MedicID"];

        // Ask for confirmation before deleting the record
        MessageBoxResult result = MessageBox.Show("Are you sure you want to delete this record?"
            , "Delete Confirmation", MessageBoxButton.YesNo);

        if (result == MessageBoxResult.Yes)
        {
            // Create SQL query for deleting the Medic record
            string query = "DELETE FROM Medic WHERE MedicID = @MedicID";

            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                SqlCommand cmd = new SqlCommand(query, conn);

                cmd.Parameters.AddWithValue("@MedicID", medicID);
            }
        }
    }
}
```

Figură 3.17 Cod exemplu Ștergere din Medic.xaml.cs

Explicație cod:

- Se verifică dacă un rând este selectat în DataGrid.
- Se preia MedicID din rândul selectat pentru identificarea înregistrării ce urmează a fi ștearsă.
- Se afișează o fereastră de confirmare pentru utilizator înainte de ștergere.
- Se creează o interogare SQL parametrizată pentru ștergerea înregistrării din baza de date.
- Se deschide conexiunea la baza de date și se execută comanda de ștergere.
- Se verifică dacă ștergerea a avut succes și se reîncarcă datele în DataGrid.
- În cazul unei erori, aceasta este capturată și afișată utilizatorului.

3.6 Gestionarea securității utilizatorului

3.6.1 Introducere

Securitatea aplicației este esențială pentru protejarea datelor utilizatorilor și prevenirea accesului neautorizat. Aplicația implementează mecanisme de autentificare și autorizare folosind MS SQL Server, cu hasharea parolelor utilizând algoritmul SHA2_256.


```

using (SqlConnection conn = new SqlConnection(connectionString))
{
    string query = "SELECT Role FROM Users WHERE " +
        "Username = @Username AND PasswordHash = HASHBYTES('SHA2_256', @Password)";
    SqlCommand cmd = new SqlCommand(query, conn);
    cmd.Parameters.AddWithValue("@Username", username);
    cmd.Parameters.AddWithValue("@Password", password);

    try
    {
        conn.Open();
        object result = cmd.ExecuteScalar();
        if (result != null)
        {
            string role = result.ToString();
            if (role == "Admin")
            {
                MainWindow mainWindow = new MainWindow(username);
                mainWindow.Show();
            }
            else if (role == "User")
            {
                RegularUserMainWindow RegularUserMainWindow = new RegularUserMainWindow(username);
                RegularUserMainWindow.Show();
            }
            this.Close();
        }
        else
        {
            MessageBox.Show("Invalid username or password.",
                "Login Failed", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error: {ex.Message}", "Database Error",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

Figură 3.20 Cod autentificare din LoginWindow.xaml.cs

Măsurile de securitate implementate:

- Folosirea parametrilor SQL (@Username, @Password) pentru a preveni atacurile de tip SQL Injection.
- Parola nu este transmisă în clar către baza de date, ci este hashată folosind HASHBYTES('SHA2_256', @Password).
- Se returnează doar rolul utilizatorului pentru a decide accesul.

3.6.4 Schimbarea parolei

Utilizatorii au opțiunea de a-și schimba parola prin intermediul ferestrei ChangePasswordWindow.xaml.cs.

Fluxul de schimbare a parolei:

- Utilizatorul introduce parola curentă, noua parolă și confirmarea acesteia.
- Aplicația verifică dacă parola curentă este corectă folosind interogarea SQL parametrizată.
- Dacă parola este corectă și confirmarea noii parole coincide, aceasta este hashată și actualizată în baza de date.

```

// Query to check if the current username and password hash exist in the database
string checkQuery = "SELECT COUNT(*) FROM Users WHERE Username = @Username AND" +
    " PasswordHash = HASHBYTES('SHA2_256', @CurrentPassword)";
using (SqlCommand checkCmd = new SqlCommand(checkQuery, conn))
{
    checkCmd.Parameters.AddWithValue("@Username", currentUsername);
    checkCmd.Parameters.AddWithValue("@CurrentPassword", currentPassword);

    int count = (int)checkCmd.ExecuteScalar();
    if (count == 0)
    {
        MessageBox.Show("Current password is incorrect.");
        return;
    }
}

// If valid, update the password
string updateQuery = "UPDATE Users SET PasswordHash = " +
    "HASHBYTES('SHA2_256', @NewPassword) WHERE Username = @Username";
using (SqlCommand updateCmd = new SqlCommand(updateQuery, conn))
{
    updateCmd.Parameters.AddWithValue("@Username", currentUsername);
    updateCmd.Parameters.AddWithValue("@NewPassword", newPassword);

    int rowsAffected = updateCmd.ExecuteNonQuery();
    if (rowsAffected > 0)
    {
        MessageBox.Show("Password updated successfully!");

        // Open the LoginWindow
        LoginWindow loginWindow = new LoginWindow();
        loginWindow.Show();

        // Close the current ChangePasswordWindow
        this.Close();
    }
}

```

Figură 3.21 Cod Schimbare parola din ChangePasswordWindow.xaml.cs

Măsuri de securitate implementate:

- Verificarea parolei curente înainte de schimbare.
- Stocarea noii parole în format hash, pentru a preveni accesul neautorizat la datele utilizatorilor.
- Prevenirea schimbării parolei cu o valoare identică cu cea anterioară.

3.6.5 Gestionarea rolurilor utilizatorilor

În baza de date, tabelul Users gestionează rolurile utilizatorilor prin coloana Role, care poate avea valori precum "Admin" și "User". Acest mecanism permite restricționarea accesului la diferite secțiuni ale aplicației.


```
CREATE TABLE Users (
    UserID BIGINT IDENTITY(1,1) NOT NULL PRIMARY KEY,
    Username NVARCHAR(50) NOT NULL UNIQUE,
    PasswordHash NVARCHAR(256) NOT NULL,
    Role NVARCHAR(20) NOT NULL CHECK (Role IN ('Admin', 'User'))
);
```

Figură 3.22 Structura tabelului Users

Gestionarea accesului în aplicație:

- Administratorii au acces complet la gestionarea utilizatorilor și datelor.
- Utilizatorii normali au acces doar la propriile date și pot vizualiza programările personale.

3.6.6 Protecția împotriva atacurilor SQL Injection

Aplicația utilizează interogări parametrizate în toate interacțiunile cu baza de date pentru a preveni atacurile de tip SQL Injection.

```
cmd.Parameters.AddWithValue("@MedicID", medicID);
cmd.Parameters.AddWithValue("@NumeMedic", txtNumeMedic.Text);
cmd.Parameters.AddWithValue("@PrenumeMedic", txtPrenumeMedic.Text);
cmd.Parameters.AddWithValue("@Specializare", txtSpecializare.Text);
```

Figură 3.23 Exemplu interogare parametrizata din Medic.xaml.cs

4 Concluzii

În cadrul acestui proiect, s-au realizat următoarele:

- Arhitectura bine definită a aplicației, bazată pe un model client-server, care permite o interacțiune fluentă între interfața utilizatorului și baza de date.
- Gestionarea eficientă a datelor prin implementarea operațiunilor CRUD (Create, Read, Update, Delete) asupra entităților principale: Medic, Pacient, Consultatie și Medicamente.
- Securizarea aplicației, asigurând protecția datelor utilizatorilor prin:
- Hasharea parolelor cu algoritmul SHA2_256 pentru a preveni stocarea parolelor în format clar.
- Utilizarea interogărilor parametrizate pentru a preveni atacurile de tip SQL Injection.
- Implementarea unei diferențieri clare a accesului utilizatorilor în funcție de rolurile definite în sistem (Admin și User).
- Interfață grafică intuitivă, realizată cu ajutorul tehnologiei WPF, oferind o experiență ușor de utilizat și accesibilă pentru utilizatori.

Posibile îmbunătățiri:

Procedură stocată pentru validarea datelor la inserare/actualizare: această procedură poate fi apelată înainte de inserarea sau actualizarea datelor în tabelul Users, verificând dacă parola respectă cerințele minime de securitate și dacă numele de utilizator conține doar caractere permise.

5 Bibliografie

SQL Server <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>

.NET <https://learn.microsoft.com/en-us/dotnet/>

WPF <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-9.0>

HASHBYTES <https://learn.microsoft.com/en-us/sql/t-sql/functions/hashbytes-transact-sql?view=sql-server-ver16>