Proiect IP
Manager Parole

Proiect în cadrul cursului de Ingineria Programării, Facultatea de Automatică și Calculatoare, Universitatea Tehnică "Gheorghe Asachi" Iași

Autori,
Moscalu Sebastian, grupa 1310B
Balauca Sabin-Gabriel, grupa 1310B
Danalache Emanuel, grupa 1310B
Danalache Sebastian, grupa 1310B

Descriere generală

ManagerParole este o aplicație desktop dezvoltată în C# utilizând platforma Windows Forms, destinată gestionării parolelor într-un mod sigur și eficient. Aceasta a fost proiectată pentru a răspunde nevoii tot mai mari de protejare a datelor personale, în special a conturilor online. Aplicația funcționează local, fără a necesita conexiune la internet, și oferă o interfață prietenoasă, intuitivă pentru utilizatori obișnuiți, fără experiență tehnică avansată.

Scopul principal al aplicației este de a permite utilizatorilor să-și salveze parolele criptate într-o bază de date locală, protejate cu o parolă principală (master password). ManagerParole oferă funcționalități esențiale precum: adăugare, ștergere, copiere în clipboard, anulare a acțiunilor (Undo), generare automată de parole și un sistem de criptare AES implementat nativ.

Arhitectura aplicației

Aplicația este structurată modular, cu separarea clară a logicii de interfață și a manipulării datelor. Principalele componente sunt:

- **UI (User Interface)** – formulare Windows Forms care permit interacțiunea cu utilizatorul: autentificare, adăugare parole, listare, ștergere, undo.
- **Criptare** – realizată cu AES 256-bit, implementată direct în codul sursă folosind clasa Hasher.cs.
- **Persistență** – stocarea datelor se face într-o bază de date SQLite locală, gestionată de clasa Database.cs.
- **Undo** – implementate prin sablonul Command, cu interfețe precum ICommand, AddPasswordCommand, RemovePasswordCommand și gestionate printr-o stivă de comenzi în UpdatePasswordCommand.cs.
- **Observer** – implementat printr-un mecanism de notificare între componente, astfel încât modificările din model (de ex. adăugarea sau ștergerea unei parole) să actualizeze automat interfața utilizator. De exemplu, `PasswordRepository` poate notifica interfața atunci când baza de date este modificată, iar `Manager.cs` actualizează lista de parole afișate.

Baza de date

Aplicația folosește o bază de date locală SQLite numită passwords.db. Aceasta conține o tabelă principală:

Tabela: Parole

- id – INTEGER PRIMARY KEY
- url – TEXT NOT NULL
- userid – TEXT NOT NULL

- parolaCriptata – TEXT NOT NULL
- remarks – TEXT

COD:

```
if (isNewDb)
{
    using (var cmd = new SQLiteCommand(_connection))
    {
        cmd.CommandText = @"
        CREATE TABLE Passwords (
            Id INTEGER PRIMARY KEY AUTOINCREMENT,
            Url TEXT NOT NULL,
            UserId TEXT NOT NULL,
            EncryptedPassword TEXT NOT NULL,
            Remarks TEXT
        )";
        cmd.ExecuteNonQuery();
    }
}
```

Comenzile SQL sunt executate din cod prin metoda ExecuteNonQuery() a clasei Database.cs. Existența unei parole este verificată cu comenzi SELECT, iar adăugarea/ștergerea se face prin INSERT și DELETE corespunzător.

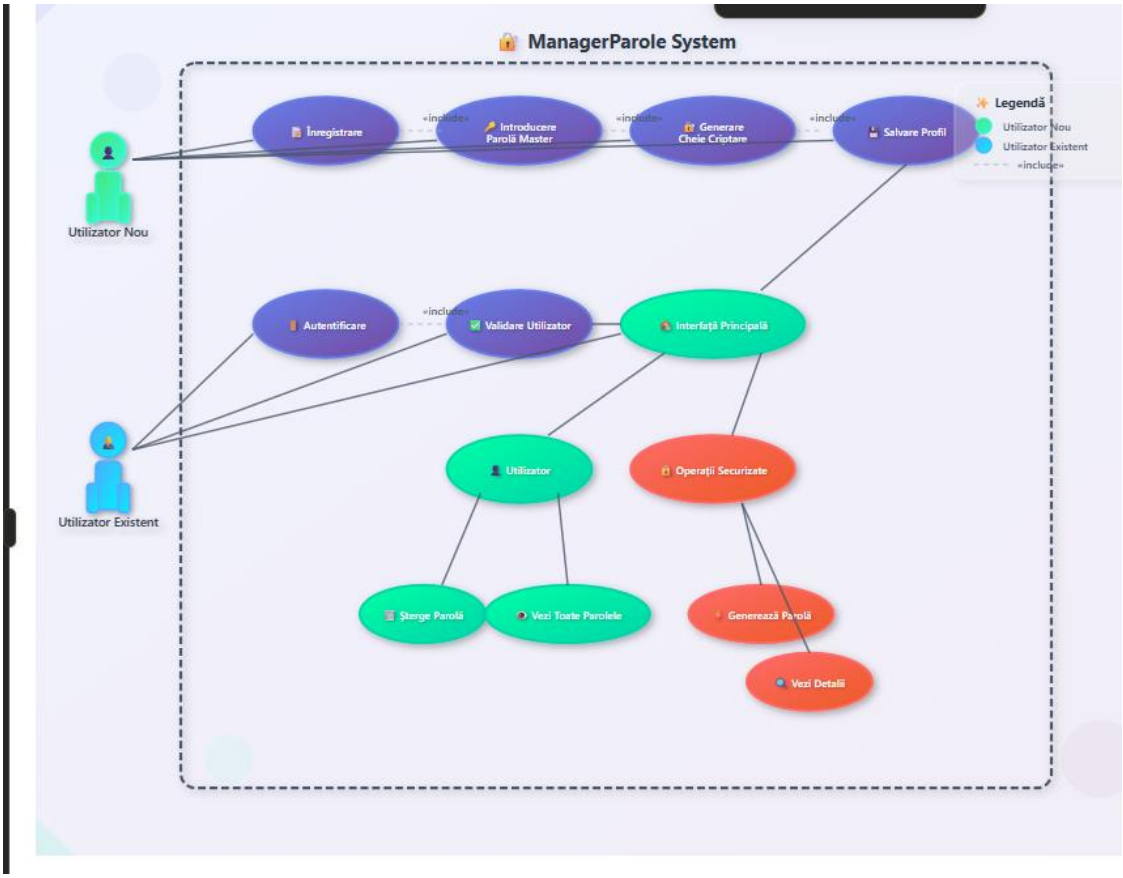
COD:

```
public int ExecuteNonQuery(string sql, Dictionary<string, object> parameters
= null)
{
    try
    {
        using (var cmd = new SQLiteCommand(sql, _connection))
        {
            if (parameters != null)
            {
                foreach (var p in parameters)
                {
                    cmd.Parameters.AddWithValue(p.Key, p.Value);
                }
            }
            return cmd.ExecuteNonQuery();
        }
    }
    catch (SQLiteException ex)
    {
        throw new PasswordManagerException("Database write failed", ex);
    }
}
```

Clase de Utilizatori și Caracteristici

Utilizator standard: folosește aplicația pentru gestionarea parolelor

Experiență: medie, nu necesită cunoștințe de programare



Criptare și Securitate

Pentru securitatea parolelor, aplicația utilizează criptare AES în mod CBC. Cheia criptografică este derivată din parola master folosind un algoritm de tip PBKDF2, iar vectorul de inițializare (IV) este generat automat.
Parolele nu sunt niciodată stocate în clar, iar cheia nu este salvată pe disc. Parola master este salvată în registry-ul Windows în formă criptată, verificată ulterior la autentificare prin decriptare și comparare cu inputul utilizatorului.
Exemplu de utilizare a criptării în cod:

```
// Criptarea unei parole folosind parola master
string parolaUtilizator = "parola123";
string parolaCriptata = Hasher.Encrypt(parolaUtilizator, parolaMaster);

// Decriptarea parolei când utilizatorul accesează aplicația
string parolaDecriptata = Hasher.Decrypt(parolaCriptata, parolaMaster);
```

Implementarea Undo

Funcționalitatea Undo este implementată folosind sablonul de design Command. Fiecare acțiune (adăugare, ștergere) este înregistrată ca obiect ce implementează ICommand și este adăugată într-o stivă de Undo.
Clase implicate:

- ICommand – interfață cu metoda Execute() și Undo()
- AddCommand, DeleteCommand – implementări concrete
- UndoManager – gestionează stiva de comenzi și execută undo la cerere

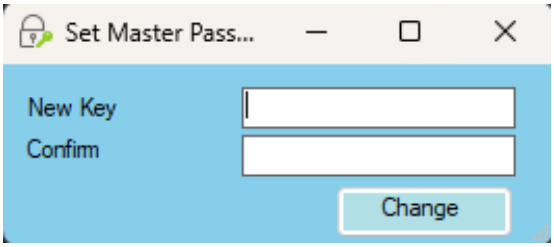
La apăsarea butonului „Undo”, se apelează:

```
private void buttonUndo_Click(object sender, EventArgs e)
{
   if (_commandHistory.Count > 0)
   {
      var last = _commandHistory.Pop();
      last.Undo();
      MessageBox.Show("The last action has been undone.", "Undo",
                  MessageBoxButtons.OK, MessageBoxIcon.Information);
   }
   else
   {
      MessageBox.Show("There are no actions to undo.", "Undo",
                  MessageBoxButtons.OK, MessageBoxIcon.Warning);
   }
}
```

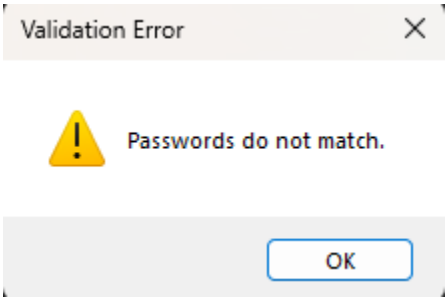
Aplicația reface automat starea anterioară a bazei de date și actualizează interfața

Gestiunea parolei master

La prima rulare a aplicației, utilizatorul este obligat să introducă o parolă master care va fi salvată criptat în Windows Registry. La pornirile ulterioare, parola introdusă este criptată și comparată cu cea existentă pentru validare.



Dacă parola introdusă este incorectă, aplicația notifică utilizatorul printr-un MessageBox și nu permite accesul mai departe.



Setarea parolei master:

```
string encryptedSecret = Hasher.Encrypt("MySecretData", newPass);
using (RegistryKey key = Registry.CurrentUser.CreateSubKey("Names"))
{
    key.SetValue("UserCP", encryptedSecret, RegistryValueKind.String);
}
```

Această valoare criptată este ulterior folosită pentru verificarea validității parolei master introduse de utilizator.

Modul de utilizare al programului

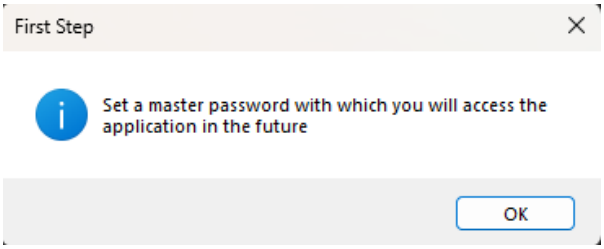
- Aplicația e împărțită în mai multe interfețe

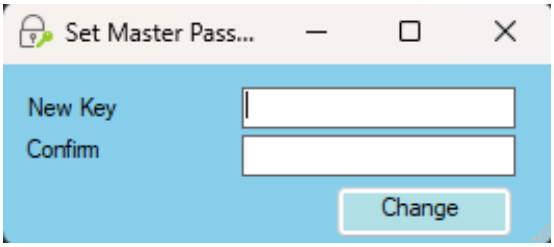
Pagina de autentificare(Login)

Aceasta este prima interfață afișată la deschiderea aplicației. Ea conține:

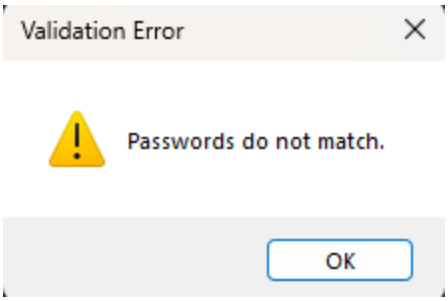
- un câmp pentru introducerea parolei master;
- un buton de autentificare;
- posibilitatea de a apăsa Enter pentru a trimite parola.

Dacă aplicația este lansată pentru prima dată (nu există o parolă salvată), se va deschide automat fereastra pentru setarea unei parole noi.





În cazul introducerii unei parole greșite, utilizatorul va primi un mesaj pop-up de avertizare.



După autentificare cu succes, utilizatorul este redirecționat către interfața principală.

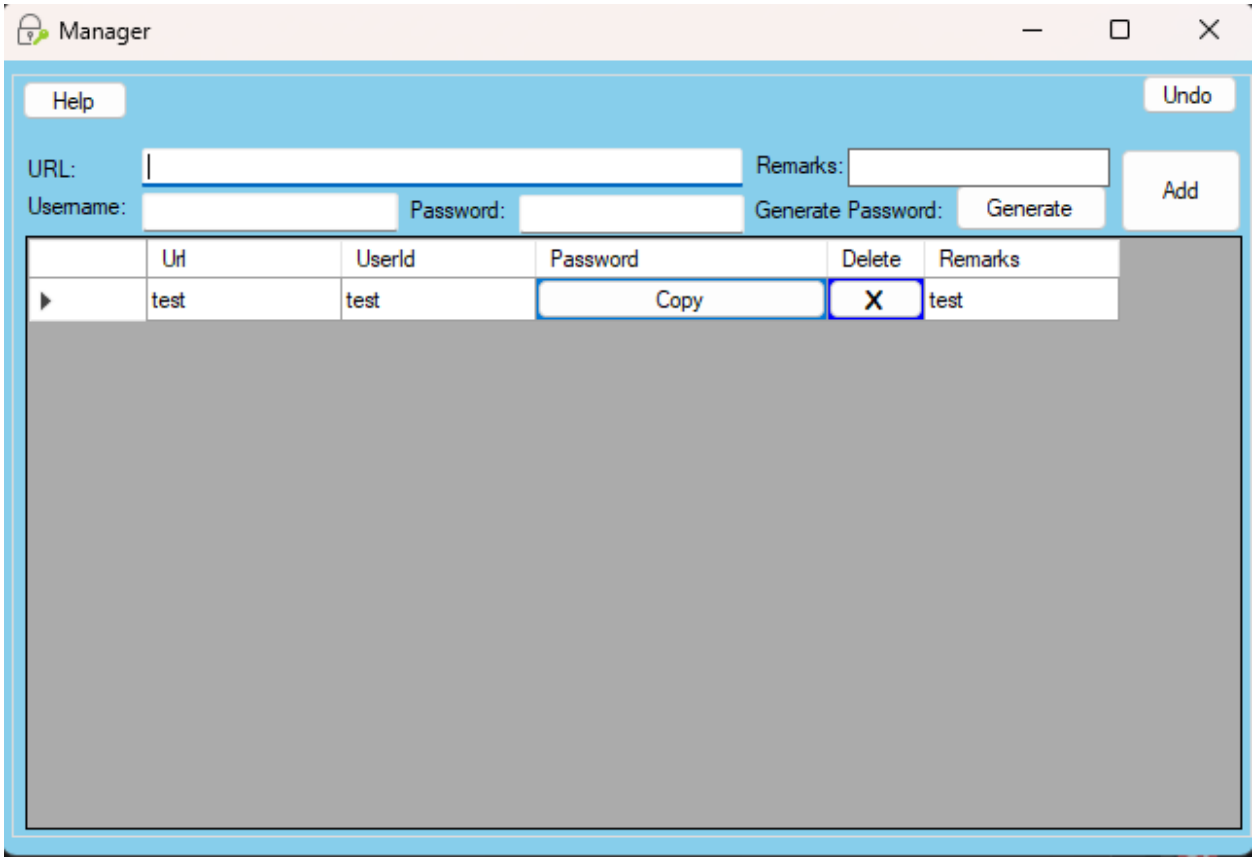
Pagina principala(Manager)

Aceasta este interfața centrală a aplicației și conține:

- câmpuri de introducere pentru URL, UserID, Parolă și Observații;
- butoane: Add, Generate, Undo, Help;
- un tabel (DataGridView) ce afișează toate parolele salvate, în următoarea ordine:
    1. URL
    2. UserID
    3. Password (buton „Copy”)
    4. Delete (buton „X”)
    5. Remarks

Funcționalități:

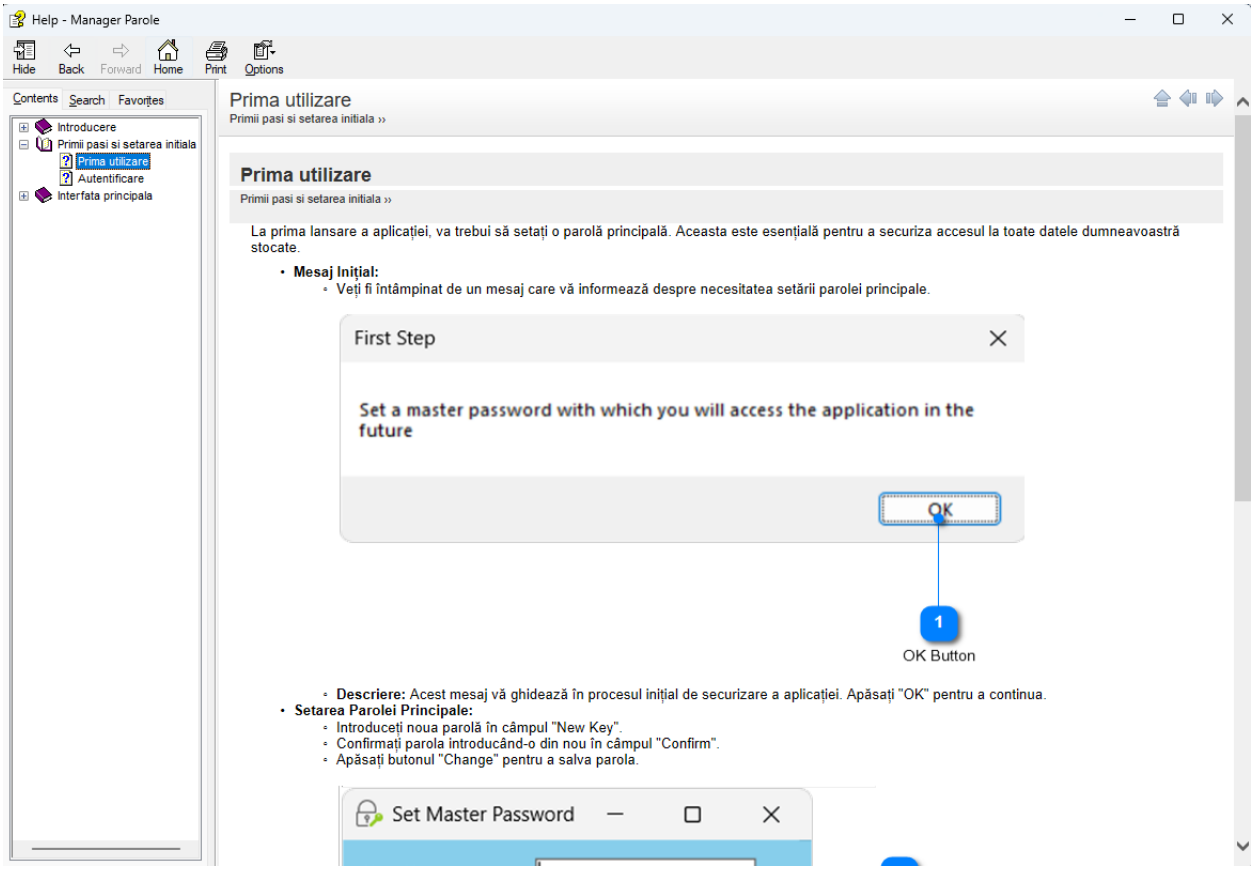
- utilizatorul introduce datele și apasă butonul Add pentru a salva o nouă parolă (criptată automat);
- câmpurile sunt resetate după salvare;
- parola poate fi generată automat prin butonul Generate;
- parolele pot fi copiate în clipboard prin apăsarea butonului Copy;
- parolele pot fi șterse definitiv cu butonul Delete (cu confirmare);
- se poate reveni asupra ultimei acțiuni (Add/Delete) cu butonul Undo.



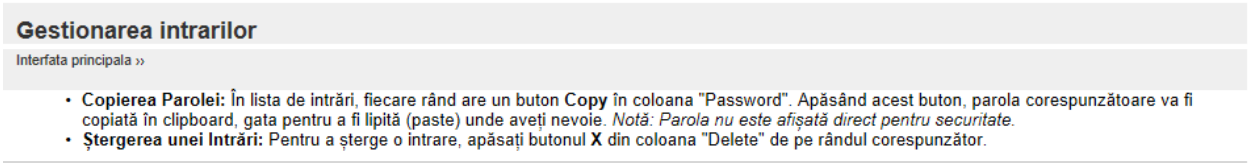
Pagina de ajutor(Help)

Apăsând butonul Help, utilizatorul deschide o pagină informativă care conține:

- Ghid pas cu pas pentru utilizarea aplicației;



- Descrierea funcționalităților principale;



Gestionarea intrarilor

Interfata principala ››

- **Copierea Parolei:** În lista de intrări, fiecare rând are un buton **Copy** în coloana "Password". Apăsând acest buton, parola corespunzătoare va fi copiată în clipboard, gata pentru a fi lipită (paste) unde aveți nevoie. *Notă: Parola nu este afișată direct pentru securitate.*
- **Ștergerea unei Intrări:** Pentru a șterge o intrare, apăsați butonul **X** din coloana "Delete" de pe rândul corespunzător.

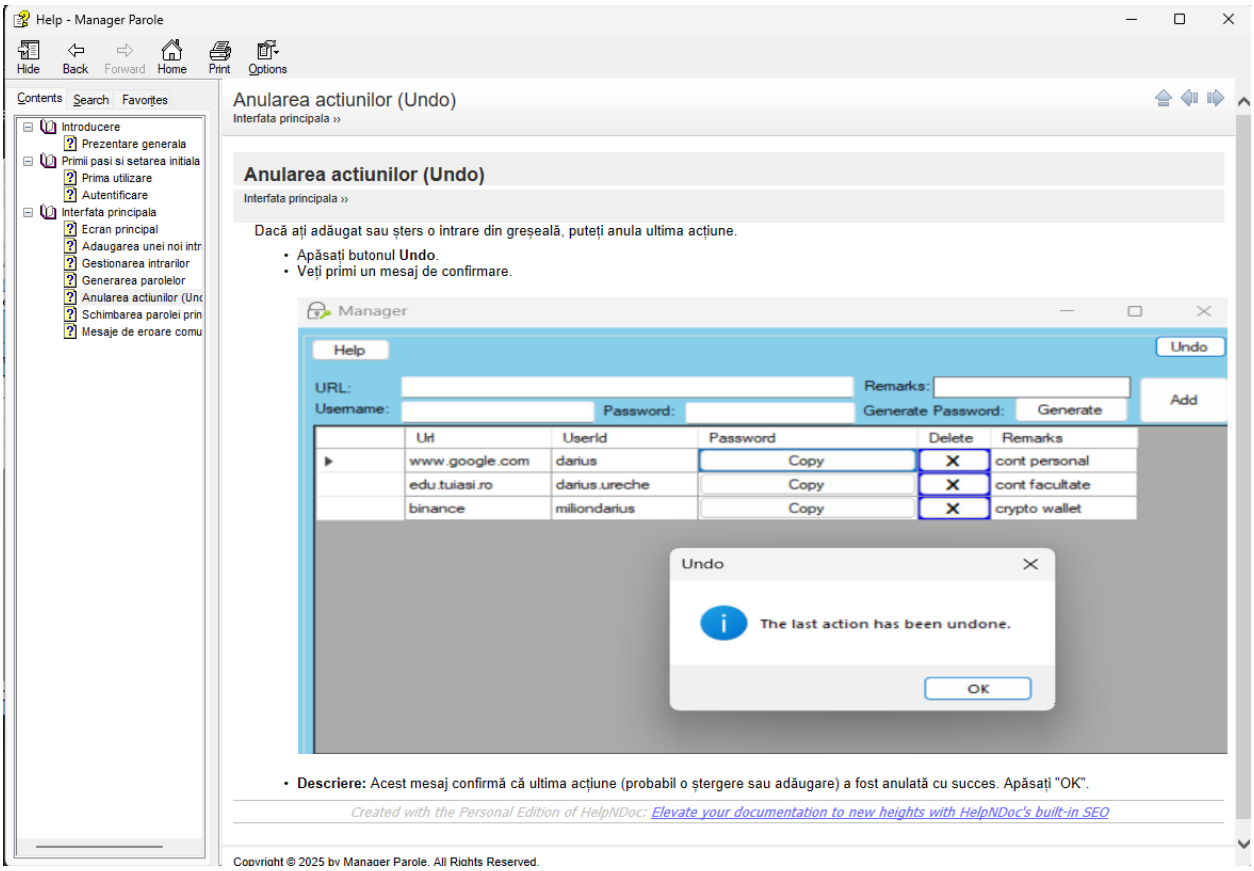
Generarea parolelor

Interfata principala ››

Dacă doriți să creați o parolă nouă și sigură pentru un cont:

1. Poziționați-vă pe câmpul **Password**.
2. Apăsați butonul **Generate Password** (etichetat "Generate" lângă câmpul "Password").
3. O parolă complexă va fi generată și introdusă automat în câmpul **Password**.

- Explicații despre criptare, Undo și modificarea parolei;



Structura fișierelor

Structura aplicației ManagerParole este organizată pe următoarele fișiere sursă, fiecare având o responsabilitate clară în cadrul arhitecturii aplicației:

- App.config – configurația aplicației
- Program.cs – punctul de intrare al aplicației

Interfață grafică (UI):

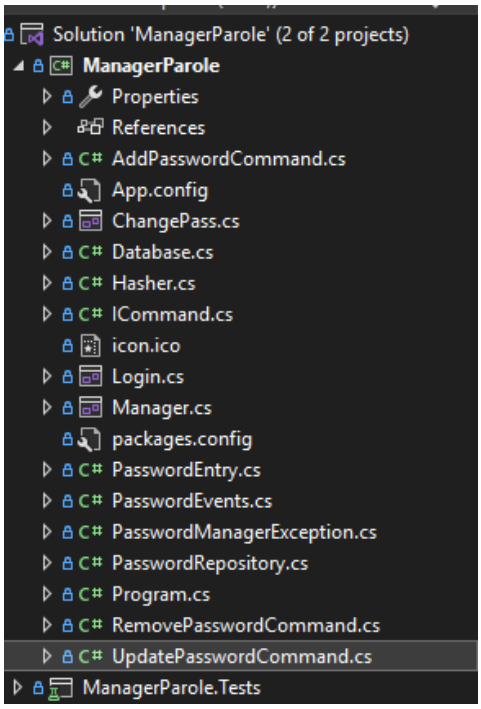
- Login.cs – logica ferestrei de autentificare
- ChangePass.cs – interfața de schimbare a parolei master
- Manager.cs – fereastra principală a aplicației (listă parole, acțiuni)

Persistență și criptare:

- Database.cs – interfața cu baza de date SQLite (inserare, ștergere, actualizare)
- PasswordEntry.cs – clasa model pentru o parolă salvată
- PasswordRepository.cs – interfață între UI și baza de date pentru obiecte PasswordEntry
- Hasher.cs – criptare AES-256 și decriptare cu PBKDF2

Undo (Command Pattern):

- ICommand.cs – interfață pentru comenzi
- AddPasswordCommand.cs – comandă de adăugare parolă
- RemovePasswordCommand.cs – comandă de ștergere parolă
- UpdatePasswordCommand.cs – comandă de modificare parolă



Alte componente:

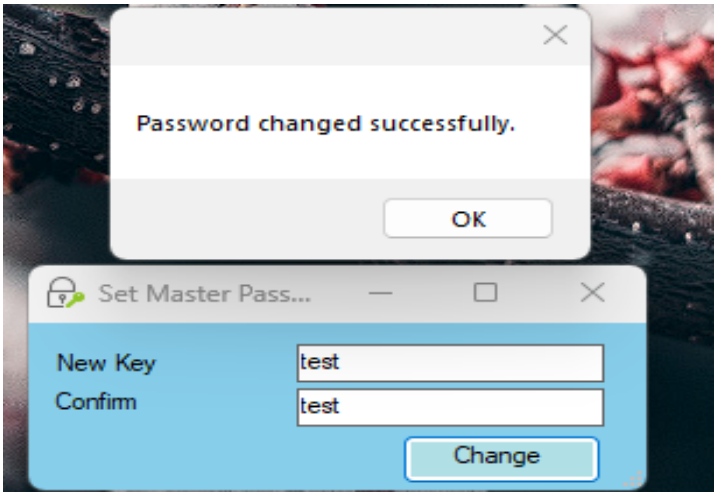
- PasswordManagerException.cs – clasă personalizată de excepții pentru operații critice
- icon.ico – pictograma aplicației
- packages.config – gestionarea dependențelor în proiect

Această structură modulară permite extinderea aplicației cu funcționalități noi fără a compromite organizarea codului existent.

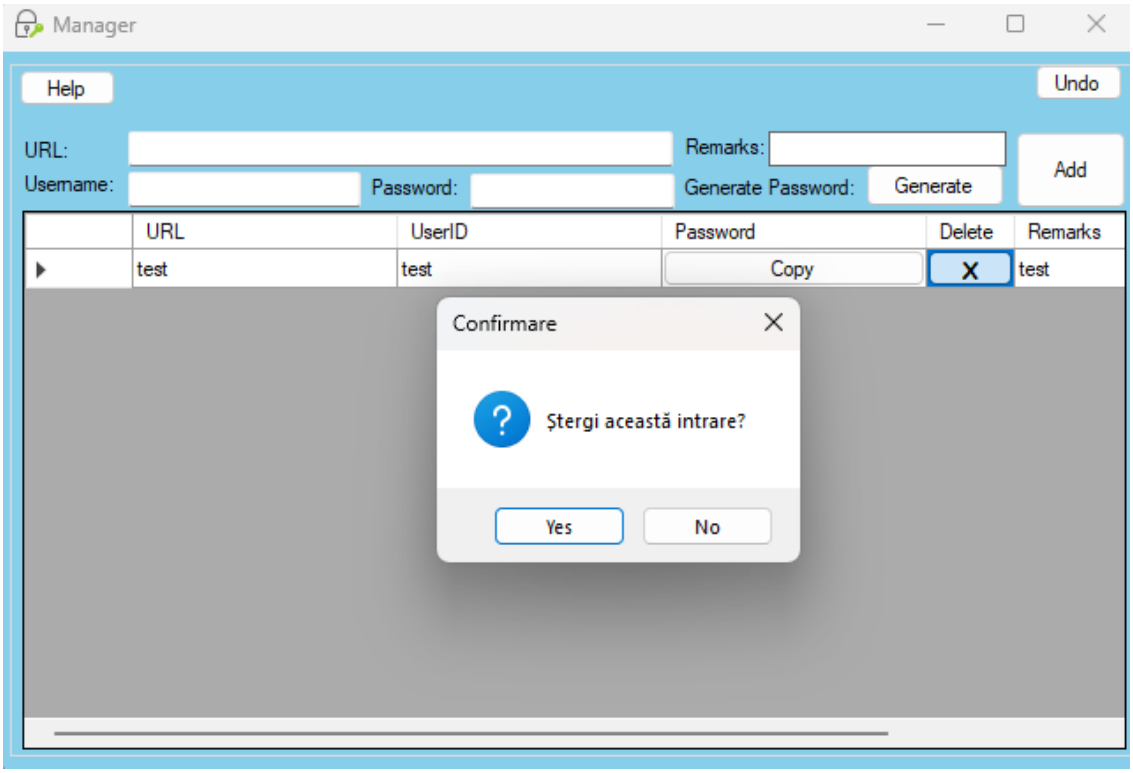
Capturi ecran diferite situații de utilizare

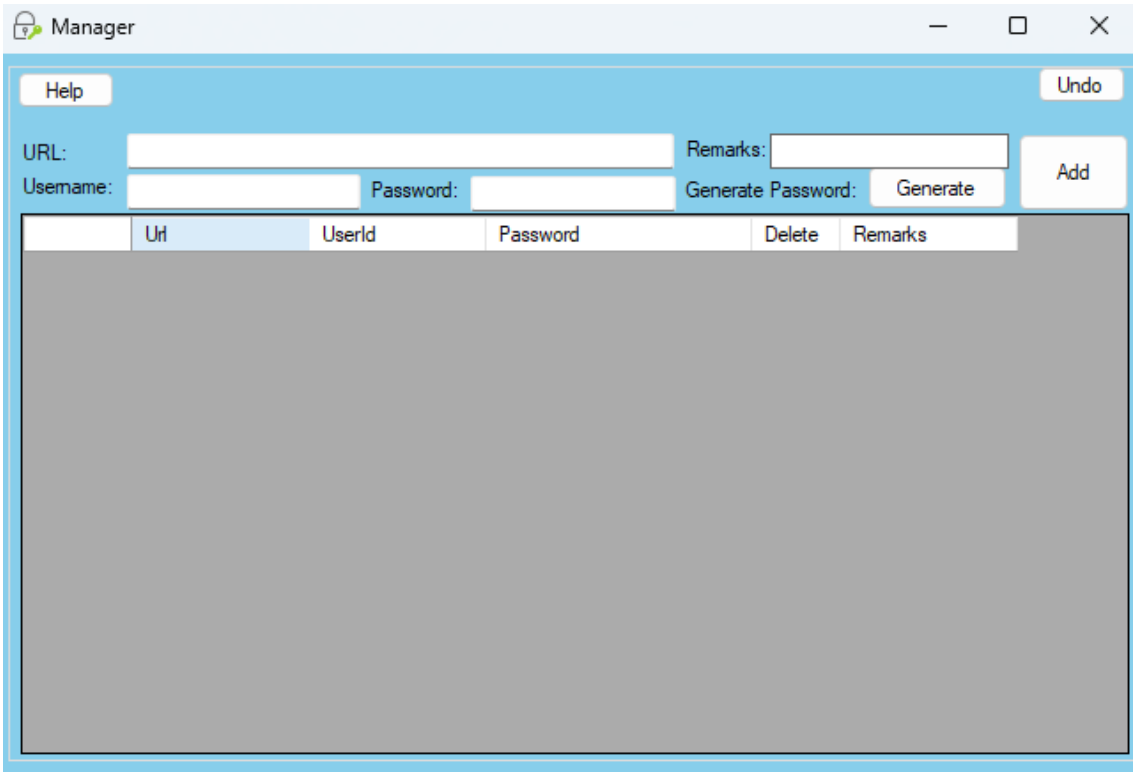
Aplicația a fost testată prin scenarii funcționale și testare manuală. Scenariile acoperă:

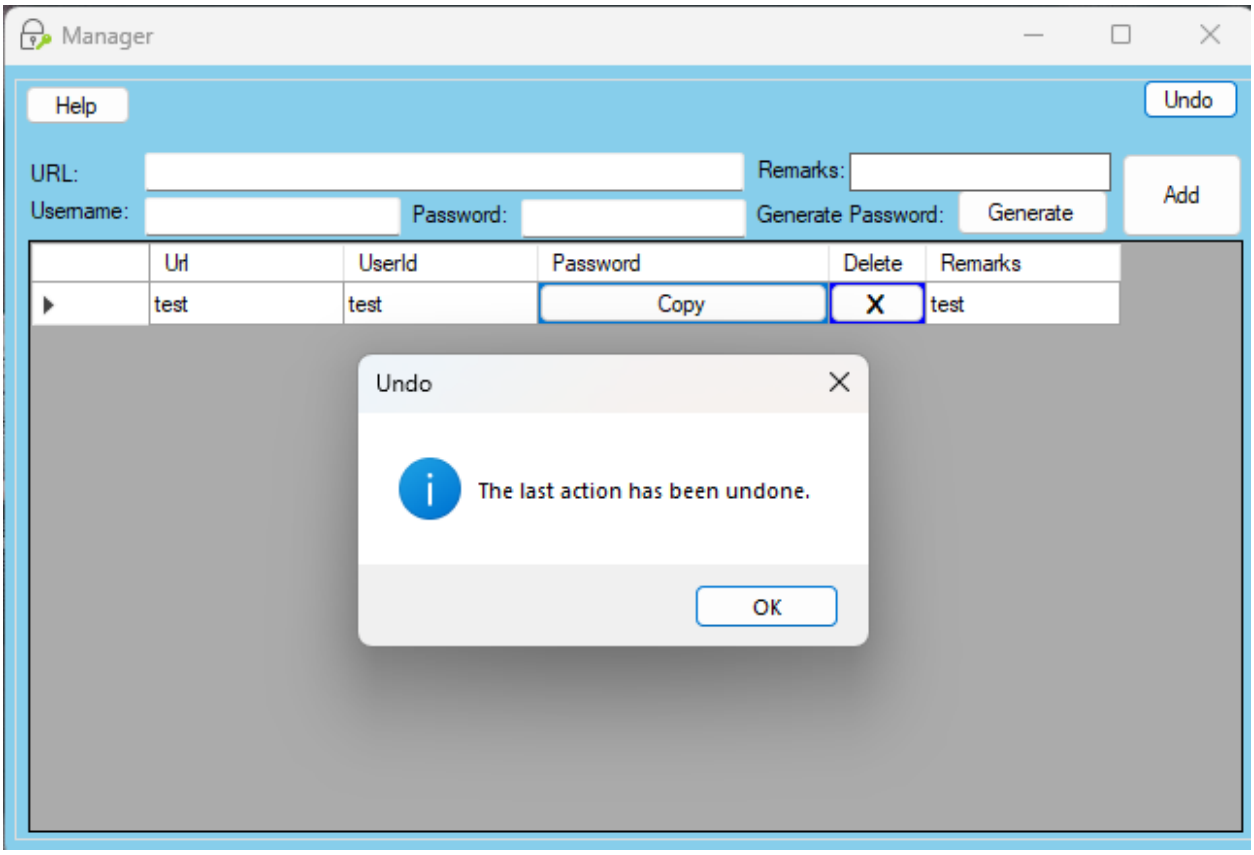
- Introducere date valide → salvare reușită



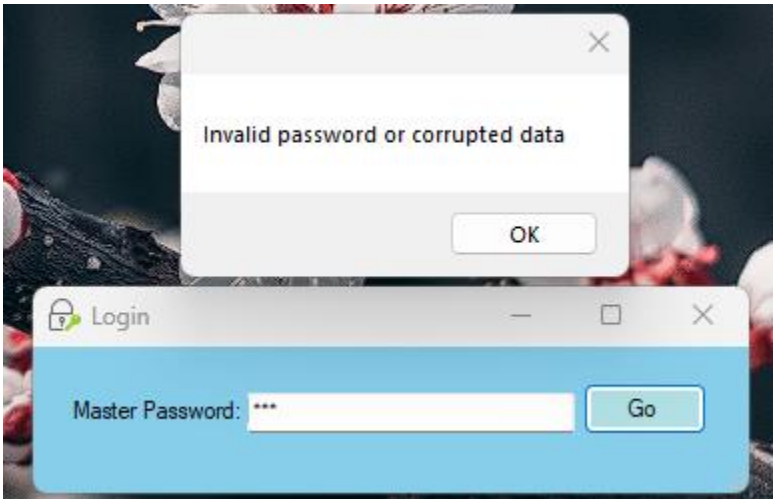
- Ștergere înregistrare → confirmare și dispariție







- Undo după adăugare/ștergere → stare restaurată
- Parolă master greșită → respingere access



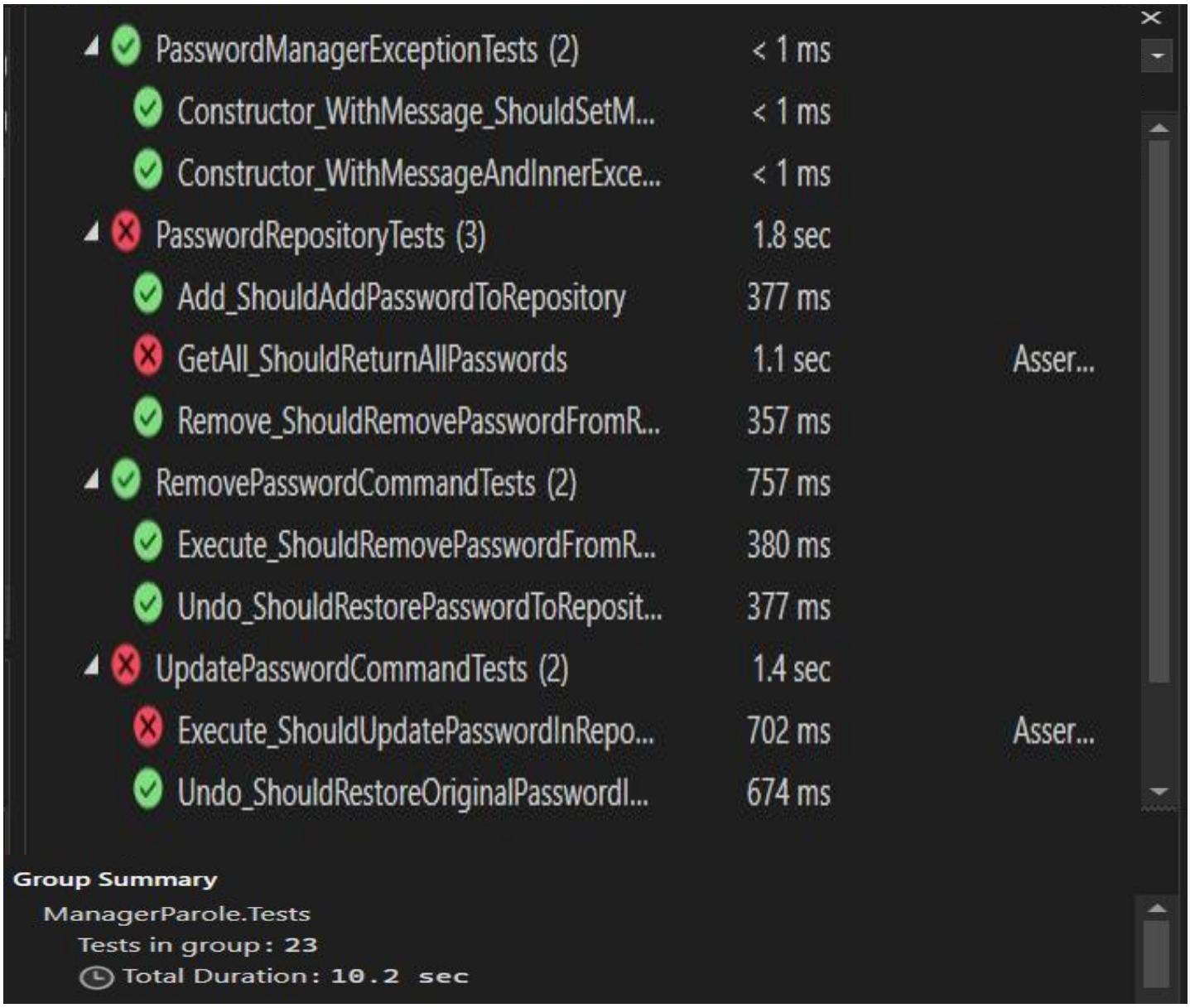
Testare unitară a aplicației (Unit Testing)

Pe lângă testarea manuală și funcțională, s-au implementat și **teste unitare** pentru a asigura corectitudinea și robustețea componentelor individuale ale aplicației. Testele unitare sunt esențiale pentru a verifica dacă fiecare "unitate" de cod funcționează conform așteptărilor în mod izolat. Acest lucru ajută la identificarea timpurie a problemelor, facilitează refactorizarea și menține calitatea codului pe termen lung.

Aplicația a fost testată prin scenarii funcționale și testare manuală. Scenariile acoperă:

- Introducere date valide → salvare reușită
- Ștergere înregistrare → confirmare și dispariție
- Undo după adăugare/ștergere → stare restaurată

- Parolă master greșită → respingere access



Diagrame

Diagrama Use-Case: Login, Adaugă, Șterge, Undo

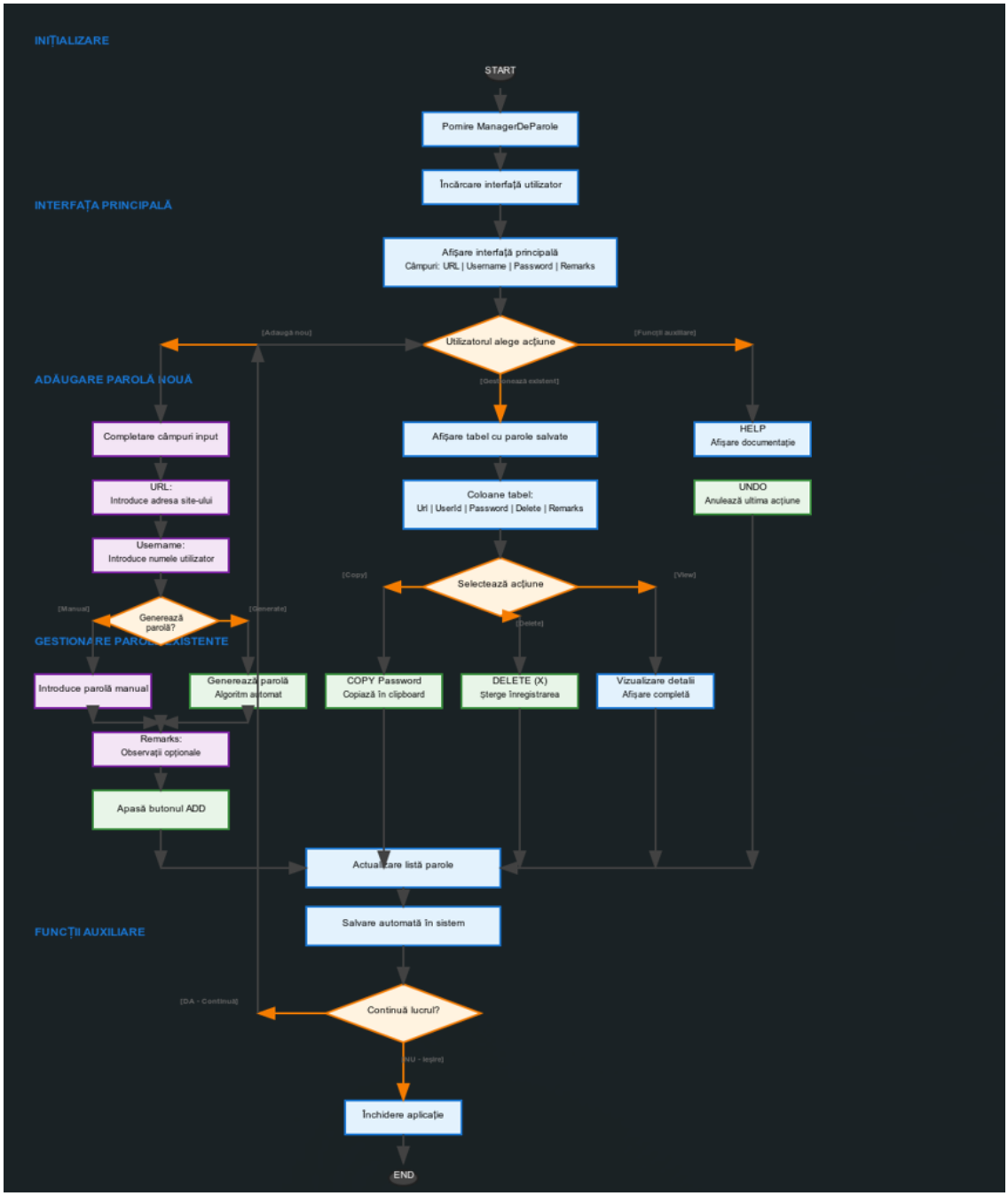


Diagrama de clase: Manager, PasswordEntry, Database, ICommand …

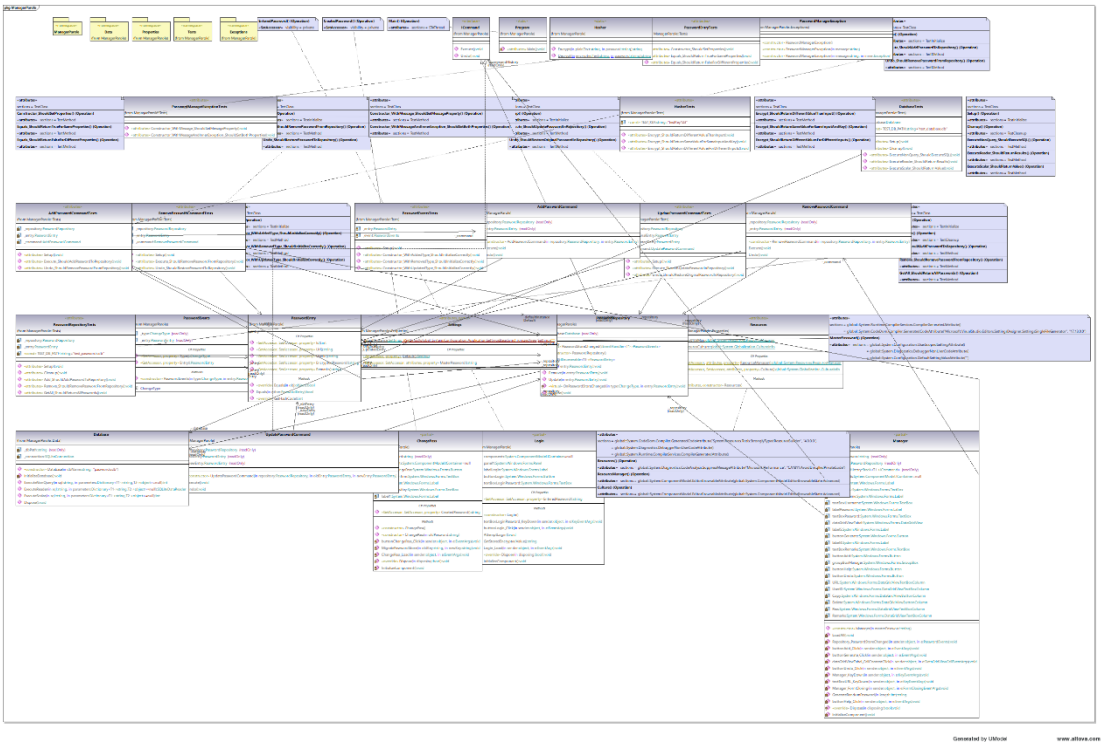
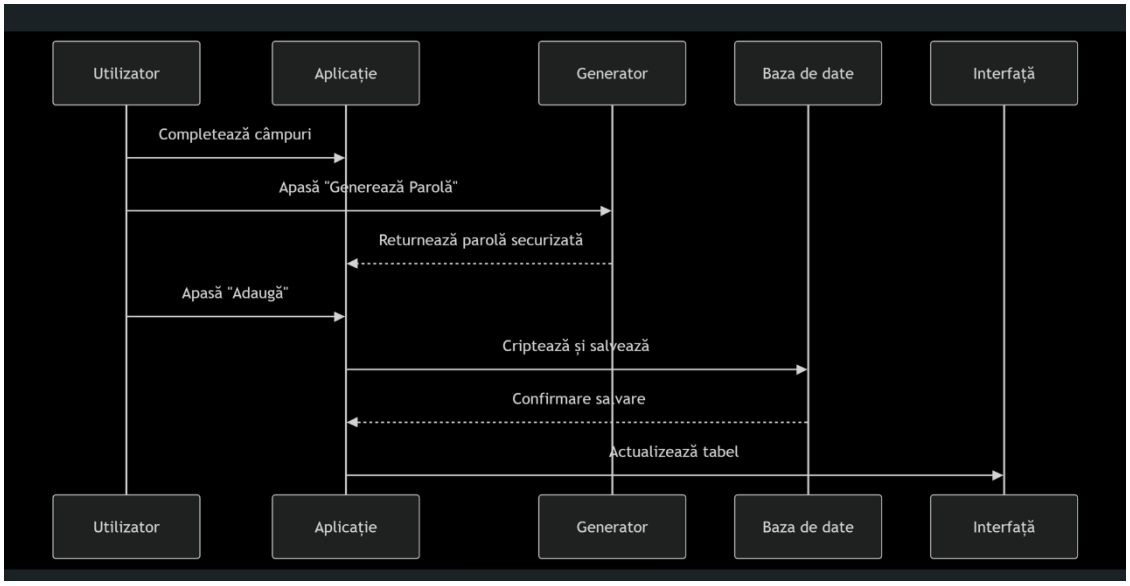


Diagrama de secvență: Login → Add → Undo



Limitări și îmbunătățiri viitoare

Limitări:

- Nu suportă utilizatori multipli

Îmbunătățiri posibile:

- Integrare cloud (Google Drive, Dropbox)
- Implementare funcționalitate Redo