

Ministerul Educaţiei și Cercetării al Republicii Moldova

Centrul de Excelență în Informatică și Tehnologii Informaționale

Catedra Informatică I

Raport

pentru practica tehnologică

**Disciplina**: Practica Tehnologică: administrarea aplicațiilor vizuale prin C#

Tema: BENEFICIARI

**Realizat** Oprea Constantin P-2131

(Numele, prenumele elevului, grupa)

**Verificat** Nebunelea Diana

(Numele, prenumele proesorului)

Nota\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

  Chişinău 2024

Cuprins

[Introducere 2](#_Toc167790391)

[Conținutul activitătii 3](#_Toc167790392)

[Observații generale 4](#_Toc167790393)

[Code Listening 4](#_Toc167790394)

[Concluzie: 12](#_Toc167790395)

[Webografie 13](#_Toc167790396)

### Introducere

În cadrul acestei sarcini, am dezvoltat o aplicație vizuală utilizând limbajul de programare C# și tehnologia Windows Presentation Foundation (WPF) pentru a gestiona informațiile despre beneficiarii unei societăți de distribuție. Baza de date utilizată pentru stocarea informațiilor despre beneficiari este administrată prin intermediul unui sistem de gestionare a bazelor de date relaționale (RDBMS), iar conexiunea dintre aplicație și baza de date a fost realizată folosind entity framework. Pentru a facilita interacțiunea utilizatorilor cu datele, am implementat o serie de funcționalități utilizând diverse controale și formulare din WPF.

Baza de date a fost proiectată pentru a asigura integritatea datelor prin utilizarea constrângerilor precum chei primare, chei externe și restricții de unicitate. Obiectele bazei de date, inclusiv tabelele, procedurile stocate și declanșatoarele, sunt administrate pentru a asigura consistența și acuratețea datelor. Planul de întreținere a bazei de date include operațiuni regulate de backup, indexare și actualizare a statisticilor pentru a optimiza performanța interogărilor.

Aplicația include următoarele funcționalități principale: înregistrarea unui nou beneficiar, utilizatorul poate introduce datele unui nou beneficiar, care sunt apoi salvate în baza de date; excluderea unui beneficiar, utilizatorul poate șterge un beneficiar specificat prin codul său, iar modificările corespunzătoare sunt reflectate în tabele; eliminarea înregistrărilor cu NrBen zero, aplicația verifică și șterge automat toate înregistrările unde atributul NrBen are valoarea zero; afișarea listei de beneficiari, lista tuturor beneficiarilor este afișată pe ecran, ordonată alfabetic după nume; calculul raportului urban/rural, aplicația determină și afișează raportul dintre numărul de beneficiari din mediul urban și cei din mediul rural; afișarea beneficiarilor dintr-o localitate specifică, utilizatorul poate introduce codul unei localități și vizualiza toți beneficiarii din acea localitate; exportul datelor, informațiile despre localitățile cu cel mai mare număr de beneficiari pot fi exportate într-un fișier MS Word sau MS Excel; crearea unei noi tabele pentru localitățile rurale, o nouă tabelă este creată și populată doar cu atributele beneficiarilor din localitățile rurale; afișarea detaliilor de contact, aplicația afișează adresa de e-mail, numărul de telefon și adresa fiecărui beneficiar.

Pentru a implementa aceste funcționalități, am utilizat framework-uri și biblioteci precum Entity Framework pentru maparea obiect-relatională și LiveCharts pentru generarea de grafice și vizualizarea datelor într-un mod interactiv și intuitiv. Interfața utilizator a fost dezvoltată folosind controalele și funcționalitățile oferite de WPF, asigurând astfel o experiență de utilizare fluidă și modernă. Securitatea datelor a fost asigurată prin implementarea de politici stricte de acces și autentificare a utilizatorilor, criptarea datelor sensibile și utilizarea unor mecanisme de backup și restaurare pentru a preveni pierderea datelor în caz de incidente neprevăzute. Prin implementarea acestei aplicații, am reușit să oferim un instrument eficient și ușor de utilizat pentru gestionarea informațiilor despre beneficiarii societății de distribuție, asigurând totodată integritatea și securitatea datelor.

### Conținutul activitătii

* Elaborarea produsului program a început cu crearea unei pagini de login, care permite utilizatorului să se autentifice și să verifice datele direct în baza de date. Am inclus și funcționalitatea de înregistrare a unui utilizator nou.
* Pagina de management a datelor conține un dashboard cu tabelul principal și diverse informații relevante. Compartimentul dedicat beneficiarilor utilizează un DataGrid pentru combinarea datelor din mai multe tabele și oferă un design custom pentru a permite sortarea, editarea, ștergerea și adăugarea de noi date.
* Pe pagina de localizare sunt prezentate datele despre beneficiarii proveniți din mediile rurale, iar utilizatorul poate interacționa cu aceste date într-un mod intuitiv datorită designului custom.
* Pagina de diagrame conține rapoarte vizuale despre codurile localităților și mediile în care se află beneficiarii, prezentate sub formă de grafice ușor de înțeles.
* Un buton de logout este disponibil pentru a permite utilizatorului să se deconecteze și să se întoarcă la pagina de login pentru autentificarea cu un alt cont.
* Toate aceste sarcini sunt gestionate printr-o instanță de bază, asigurând consistența și corectitudinea operațiilor efectuate în cadrul aplicației.

##### Pentru testarea produsului program, am folosit următoarele proceduri:

* Datele de intrare: Am introdus în mod deliberat diverse combinații de date și situații pentru a verifica funcționalitatea și robustețea aplicației. Aceste date au inclus diferite tipuri de date pentru autentificare și informații despre beneficiari.
* Datele de ieșire: Am verificat corectitudinea datelor afișate în interfața utilizatorului în funcție de acțiunile efectuate. Am asigurat că datele prezentate sunt actualizate corespunzător și reflectă modificările făcute în baza de date.
* Funcționalitatea produsului program: Am testat fiecare funcționalitate a aplicației, inclusiv autentificarea, înregistrarea utilizatorilor noi, gestionarea datelor, afișarea corectă a informațiilor și generarea de rapoarte vizuale. Am verificat și interacțiunea cu diverse elemente ale interfeței utilizatorului pentru a asigura o experiență fluidă și intuitivă.
* Am efectuat teste de încărcare pentru a evalua performanța aplicației în condiții de utilizare intensivă și am testat și în diverse medii și configurații pentru a asigura compatibilitatea și stabilitatea produsului. Toate aceste teste au fost documentate și raportate pentru a identifica și remedia eventualele probleme sau erori.

### Observații generale

* Produsul program dezvoltat reușește să îndeplinească cu succes cerințele și funcționalitățile specificate în cadrul sarcinii.
* Interfața utilizatorului este intuitivă și prietenoasă, facilitând navigarea și interacțiunea cu diferitele module și funcționalități.
* Implementarea unui sistem de login și înregistrare asigură securitatea și controlul accesului la date, contribuind la protejarea informațiilor sensibile.
* Utilizarea designului custom pentru elementele de interfață adaugă un plus de estetică și îmbunătățește experiența utilizatorului.

### Code Listening

###### Metodele utilizate în toate clasele

*Metodele de MouseEnter și MouseLeave au fost implementate în fiecare clasă pentru a crea animația de hover în momentul accesării butoanelor.*

private void Label\_MouseEnter(object sender, MouseEventArgs e)

{

Mouse.OverrideCursor = Cursors.Hand;

}

private void Label\_MouseLeave(object sender, MouseEventArgs e)

{

Mouse.OverrideCursor = null;

}

*Aceste metode sunt responsabile pentru schimbarea cursorului la trecerea peste butoane, indicând utilizatorului că acțiunea de click este posibilă. Când cursorul este pe buton, acesta se transformă într-o mână (Cursors.Hand), iar când cursorul părăsește butonul, se revine la cursorul implicit al sistemului (null). Această funcționalitate îmbunătățește experiența utilizatorului și oferă un feedback vizual în timp real.*

###### Verificarea pentru login

*Metoda btn\_SignIn\_Click este responsabilă pentru verificarea credențialelor introduse în interfața de autentificare. Aceasta este declanșată atunci când utilizatorul apasă butonul de "Sign In".*

private void btn\_SignUp\_Click(object sender, RoutedEventArgs e)

{

string username = textBox\_username.Text;

string password = textBox\_pass.Password;

using (var context = new BeneficiariContext()) {

LoginUser user = new LoginUser();

user.Username = username;

user.Pass = password;

bool isUserPresent = context.LoginUsers.Any(u => u.Username == username && u.Pass == password);

if (isUserPresent)

{

MessageBox.Show("User already exists");

}

else {

context.LoginUsers.Add(user);

context.SaveChanges();

}

}

lbl\_signin\_Click(sender, e);

}

*Această metodă extrage numele de utilizator și parola introduse în interfața grafică și le compară cu datele stocate în baza de date. Dacă sunt găsite credențiale valide, se afișează un mesaj de succes și se deschide pagina principală a aplicației. În caz contrar, se afișează un mesaj de eroare că autentificarea a eșuat din cauza credențialelor invalide.*

###### Activarea paginilor

*Metoda loadFrame() este responsabilă pentru activarea și încărcarea conținutului în cadrul cadrelor (frames) din interfața grafică. Această metodă poate fi utilizată pentru a încărca și afișa diferite pagini sau module în cadrul unei aplicații.*

public void loadFrame()

{

// Încărcarea conținutului în cadrul fiecărui frame

Buttons.Content = new BeneficiariButtons();

DashBoard.Content = new Dashboard();

Chart.Content = new Chart();

}

*Această metodă poate fi apelată în momentul în care este necesară activarea sau actualizarea conținutului din cadrul cadrelor. De exemplu, în momentul inițializării aplicației sau după efectuarea unor acțiuni specifice care necesită afișarea sau modificarea unor componente ale interfeței grafice.*

###### Logout

*Metoda LogoutCase este responsabilă pentru deconectarea utilizatorului și revenirea la pagina de autentificare.*

private void LogoutCase(object sender, RoutedEventArgs e)

{

// Închiderea ferestrei curente (pagina curentă)

Window.GetWindow(this).Close();

// Crearea unei noi instanțe a paginii de autentificare (MainWindow) și afișarea acesteia

MainWindow login = new MainWindow();

login.Show();

}

*Atunci când utilizatorul dorește să se deconecteze, această metodă închide fereastra curentă (pagina curentă) și deschide pagina de autentificare (MainWindow). Astfel, utilizatorul este redirecționat la pagina de login pentru a se autentifica cu alt cont sau pentru a ieși din aplicație.*

###### Creare pop-up pentru edit/add

*Aici este un markup pentru un Popup care ar putea fi folosit pentru actualizarea informațiilor despre un beneficiar într-o aplicație WPF. Acest Popup conține o serie de câmpuri de introducere pentru diferite detalii despre beneficiar și două butoane pentru a trimite actualizările sau a închide fereastra de dialog.*

<Popup Placement="Center" x:Name="updateBeneficiary" Style="{StaticResource popup\_width}">

<Border Style="{StaticResource popup\_border}">

<StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Nume" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_name" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Prenume" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_surname" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Adresa" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_address" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Document" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_document" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Telefon" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_phone" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Mediul" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_mediul" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Cod Localitate" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_codLocalitate" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Card Number" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_cardNumber" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Suma" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_sum" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Email" Style="{StaticResource popup\_label\_width}"/>

<TextBox x:Name="update\_email" Style="{StaticResource popup\_textbox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<Button Content="Enter" Click="update\_EnterButton\_Click" Margin="10" Width="50"/>

<Button Content="Close" Click="update\_CloseButton\_Click" Margin="10" Width="50"/>

</StackPanel>

</StackPanel>

</Border>

</Popup>

*Acest markup definește un Popup cu un stil (popup\_width) și un border (popup\_border) pentru a încadra conținutul. În interior, avem un StackPanel care conține mai multe StackPanel-uri orizontale, fiecare conținând un Label pentru a descrie câmpul și un TextBox pentru a introduce sau edita informațiile.*

*La sfârșit, avem un StackPanel orizontal care conține două butoane: unul pentru a trimite datele actualizate (Enter) și unul pentru a închide fereastra de dialog (Close). Ambele butoane sunt conectate la evenimente care vor fi definite în cod-behind-ul paginii.*

###### Crearea chart

*Aici este un exemplu de markup pentru crearea unui grafic de tip tort (pie chart) într-o aplicație WPF, folosind biblioteca LiveCharts:*

<Wpf:PieChart Name="Doughnut"

Series="{Binding SeriesCollection}"

LegendLocation="Right"

InnerRadius="90"

Margin="621,114,228,242"

Panel.ZIndex="999"

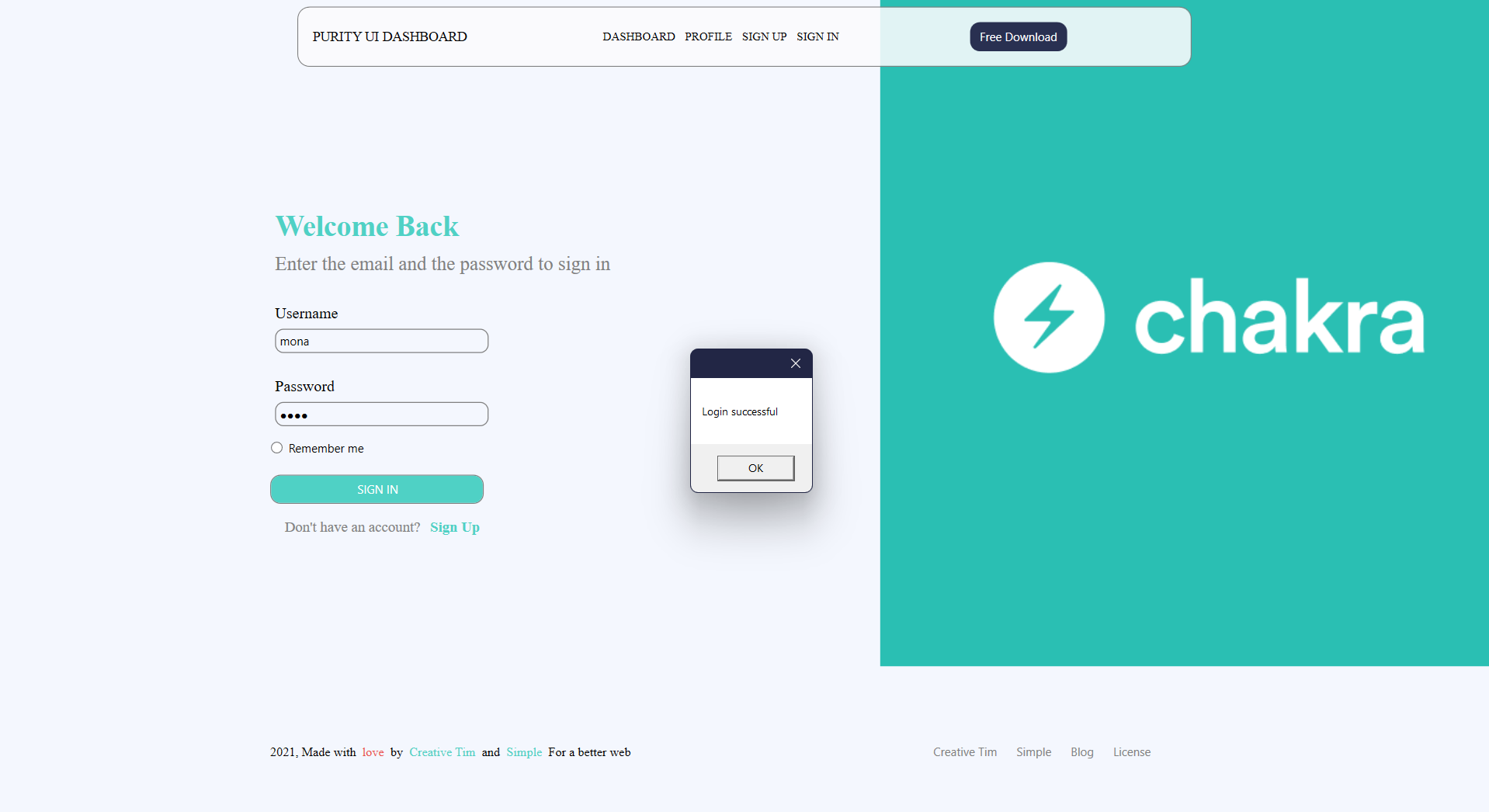
/>

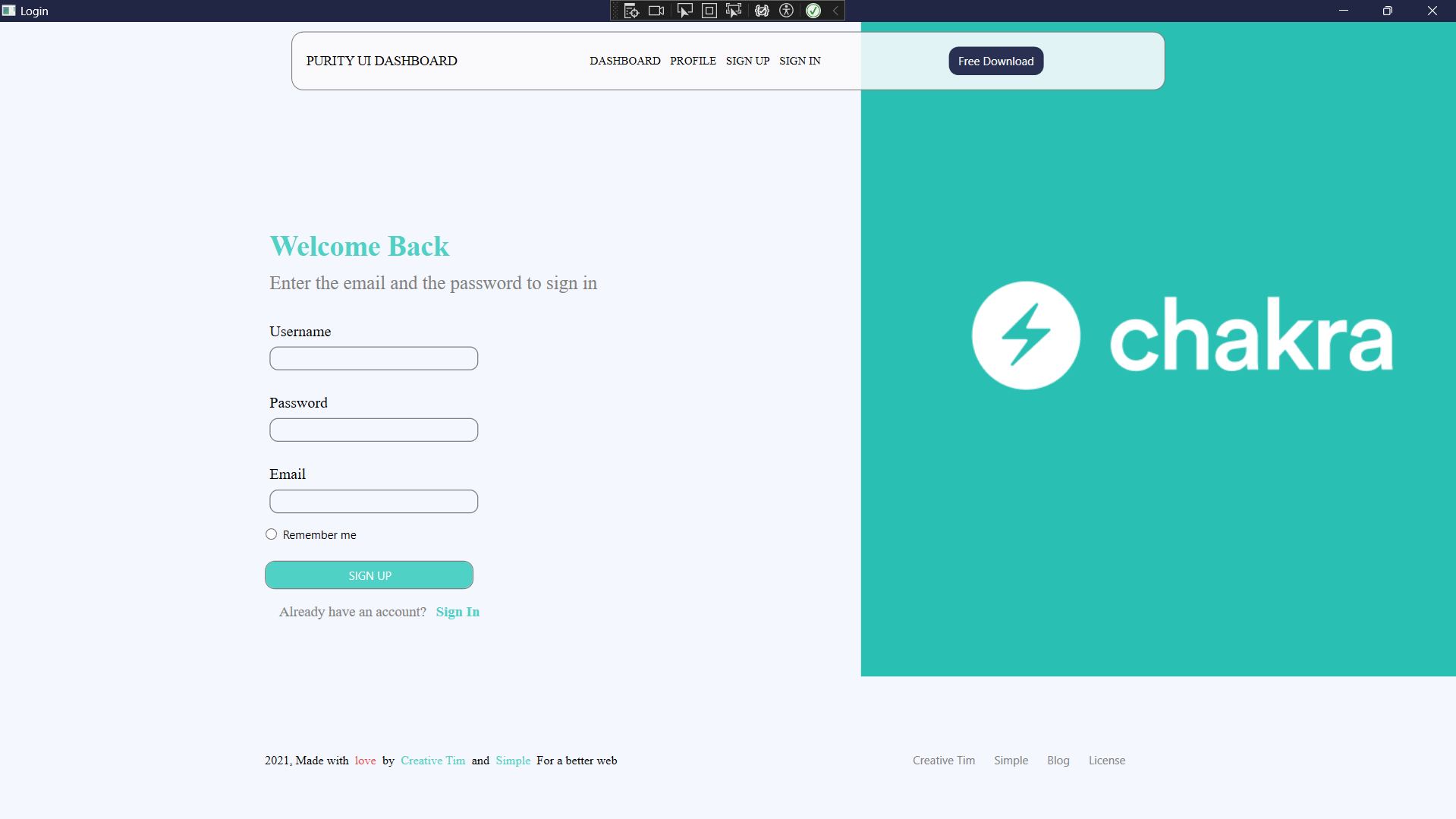
*Acest cod definește un obiect de tip PieChart cu numele "Doughnut". Atributul Series este legat la o colecție de serii de date (SeriesCollection), care trebuie să fie definită în codul din spatele paginii sau în modelul de date. LegendLocation specifică poziția legendei graficului (în acest caz, în partea dreaptă a graficului). InnerRadius controlează dimensiunea găurii centrale a graficului, permițând crearea unui efect de inel (în acest caz, valoarea 90). Margin specifică marginile graficului în cadrul containerului părinte, iar Panel.ZIndex controlează nivelul de stivă al graficului (în acest caz, un număr mare pentru a asigura afișarea deasupra altor elemente din interfața grafică).*

###### Utilizarea paginilor

###### Pagina de login

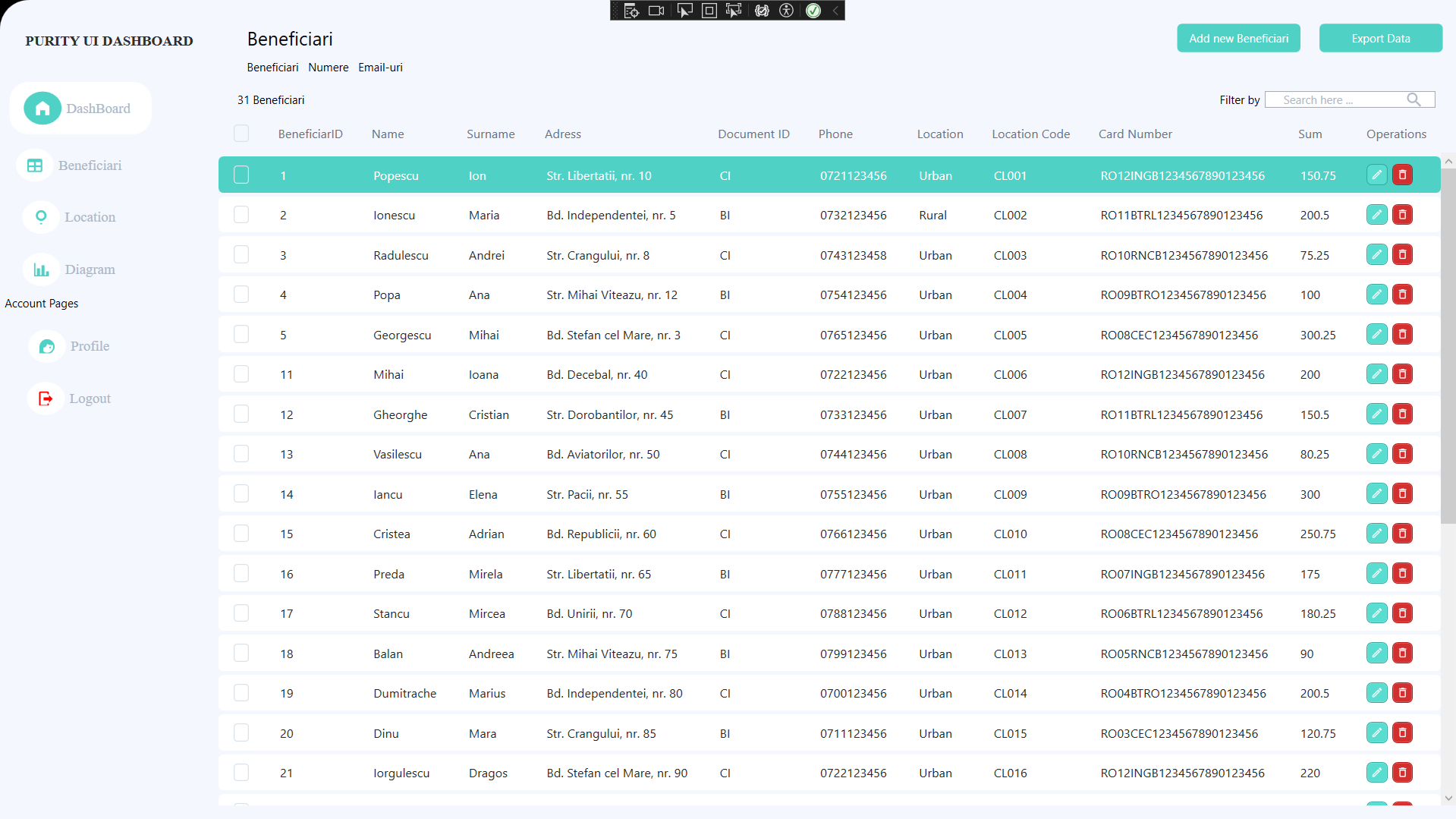
Avem un design și o interfață simplificată pentru pagina de login, care permite atât autentificarea, cât și înregistrarea utilizatorilor. Acest lucru este implementat în C# folosind WPF.

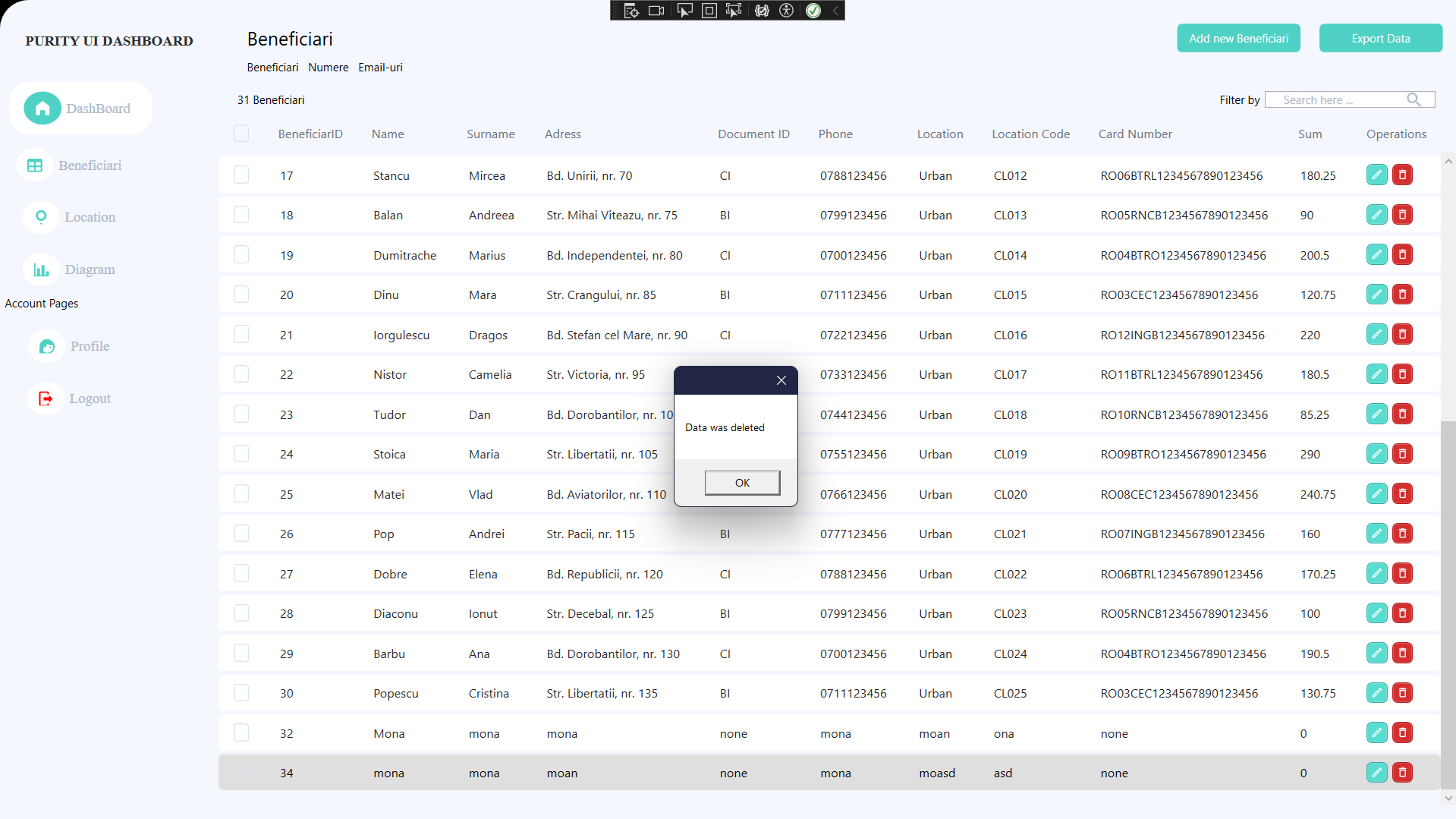


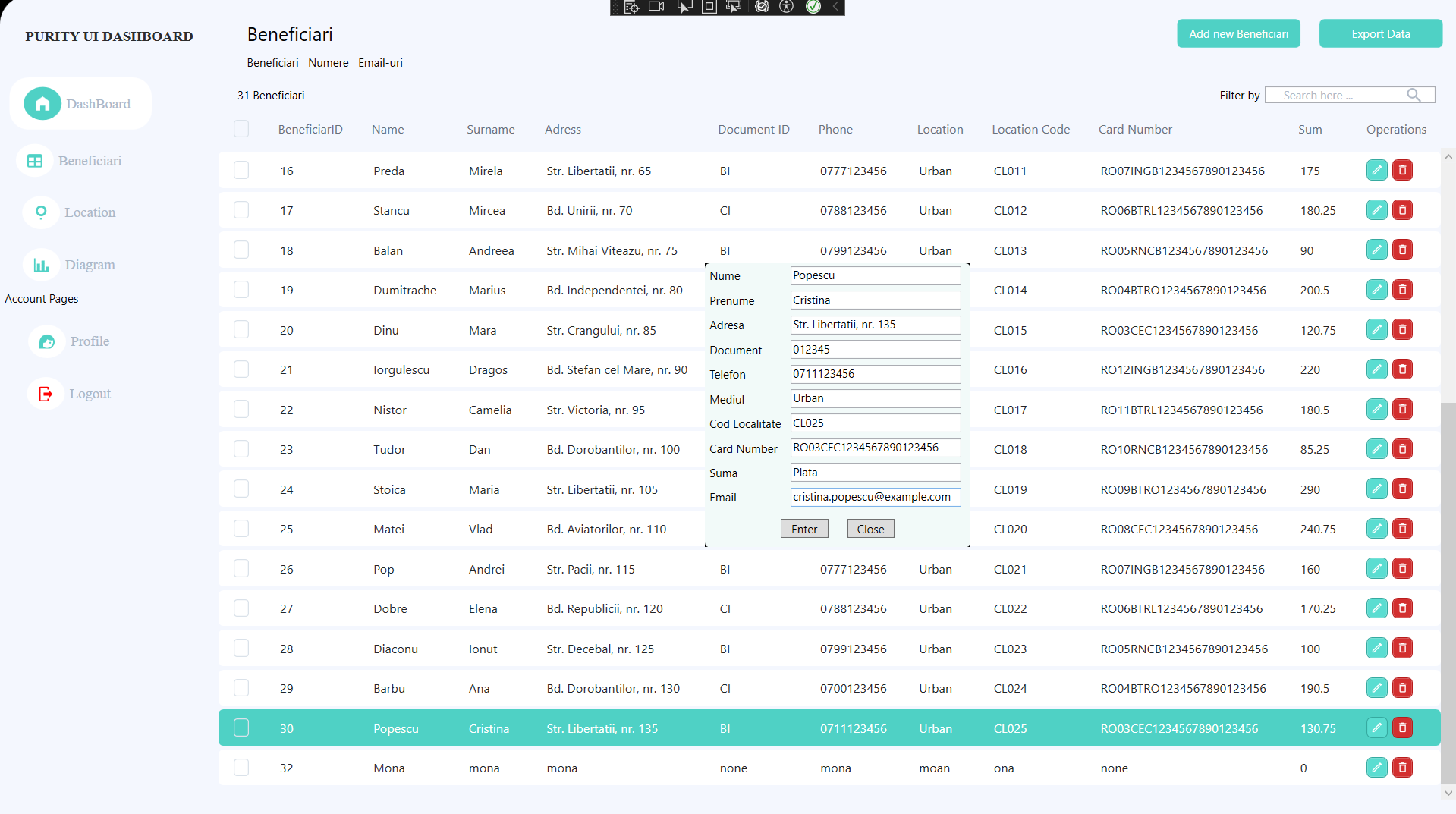


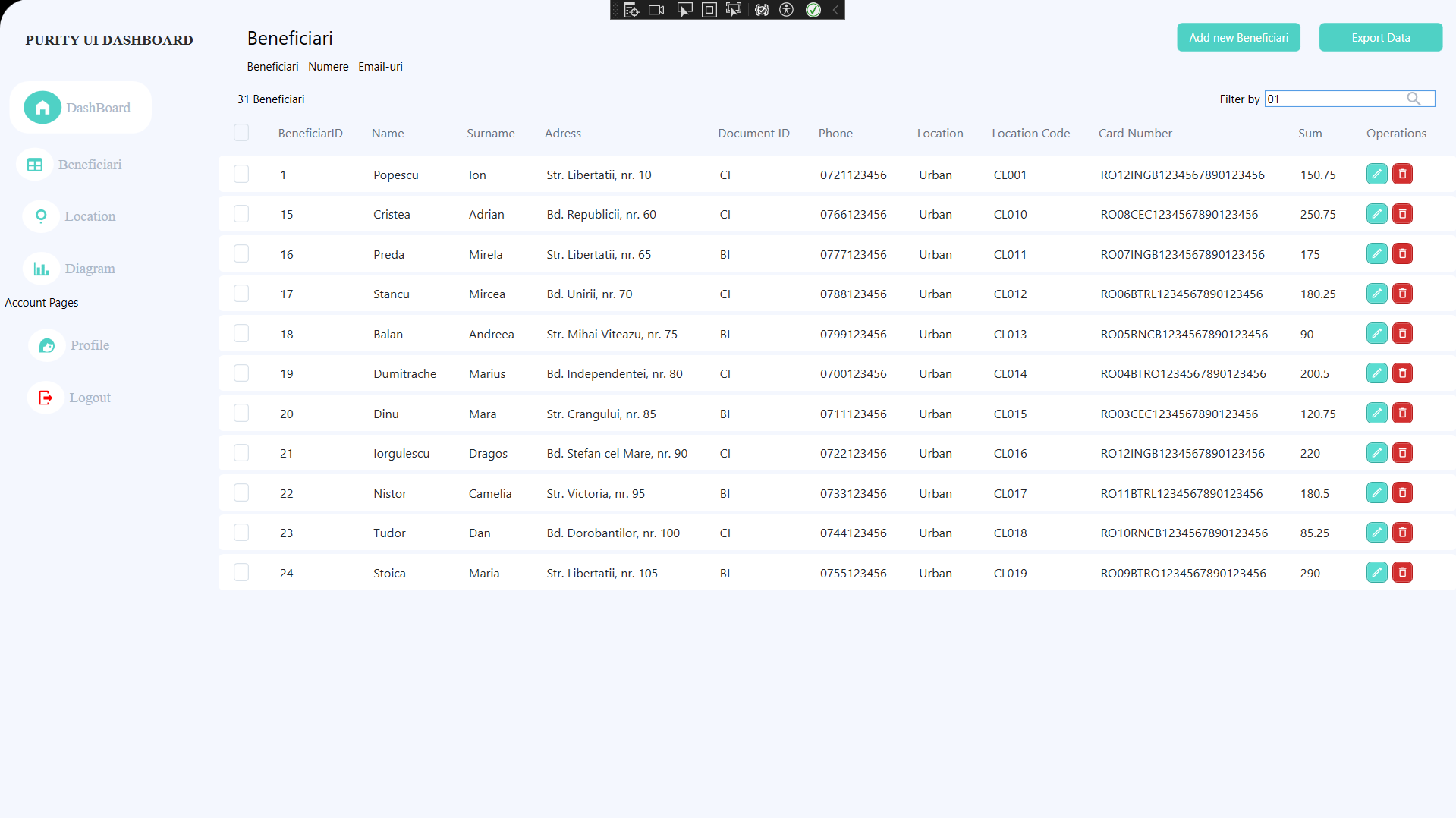
###### Management (dataGrid, beneficiari)

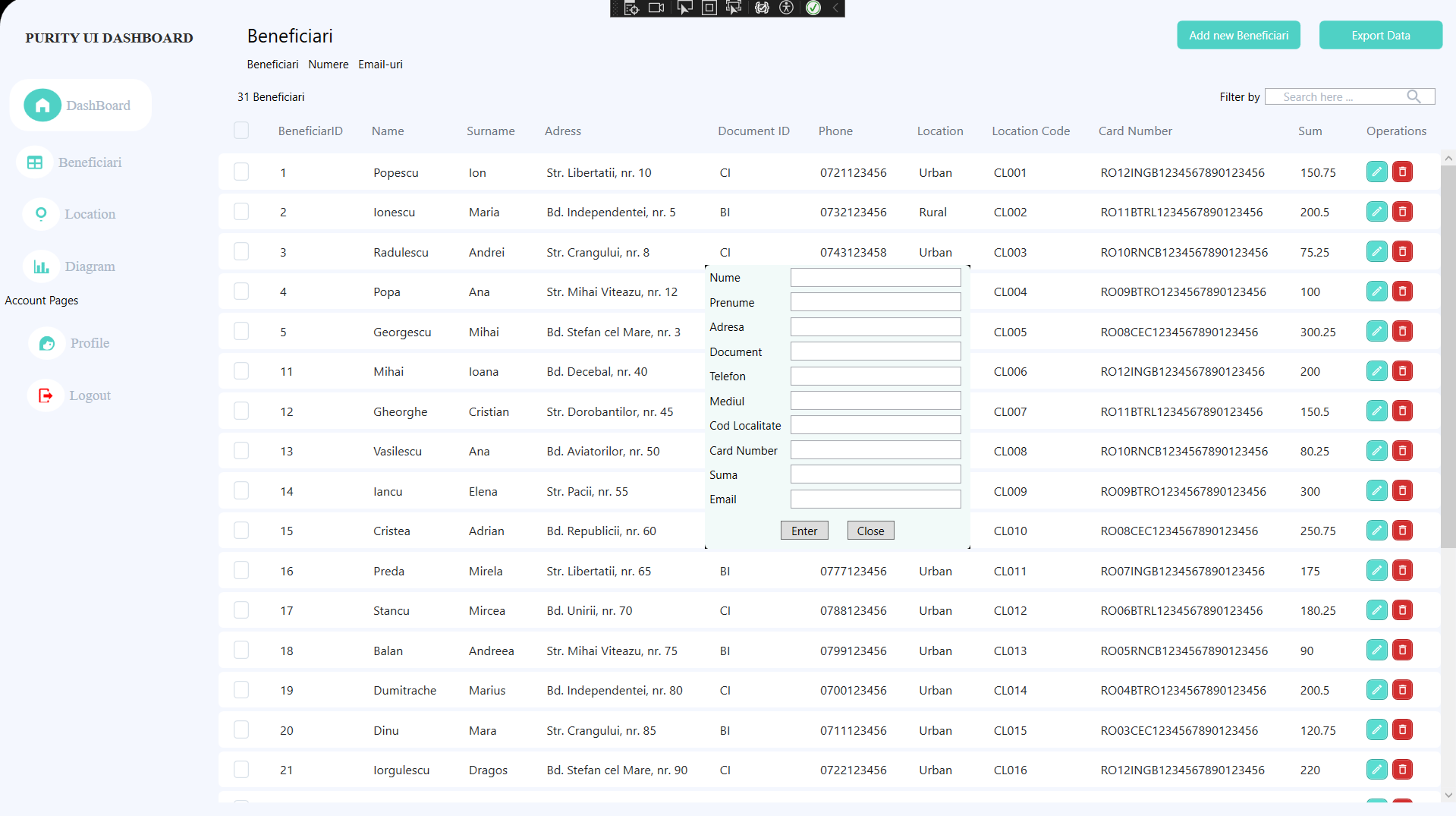
Pentru pagina de management, am implementat funcționalități care permit adăugarea beneficiarilor, exportarea datelor în format Excel, editarea informațiilor și ștergerea beneficiarilor.

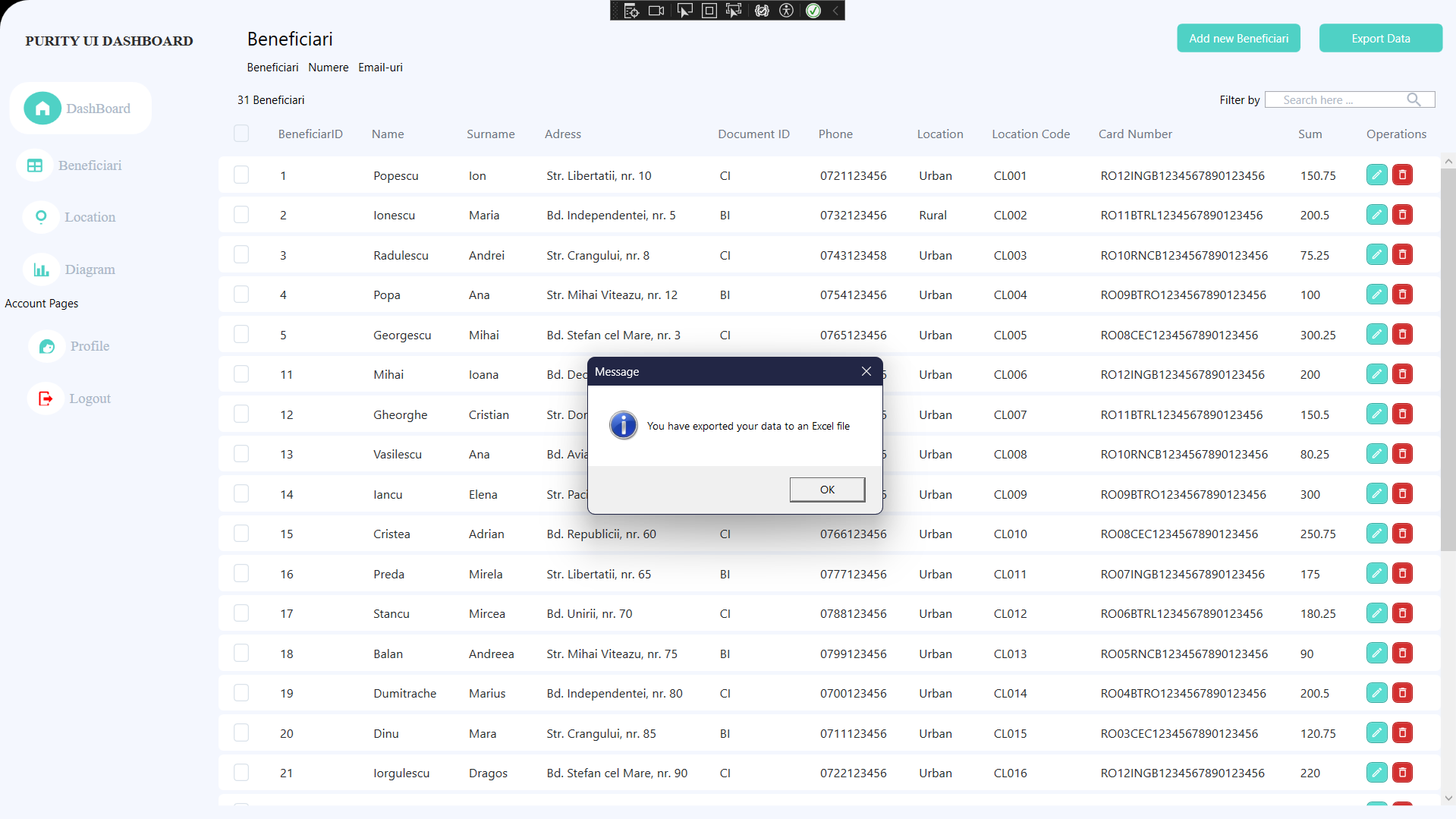


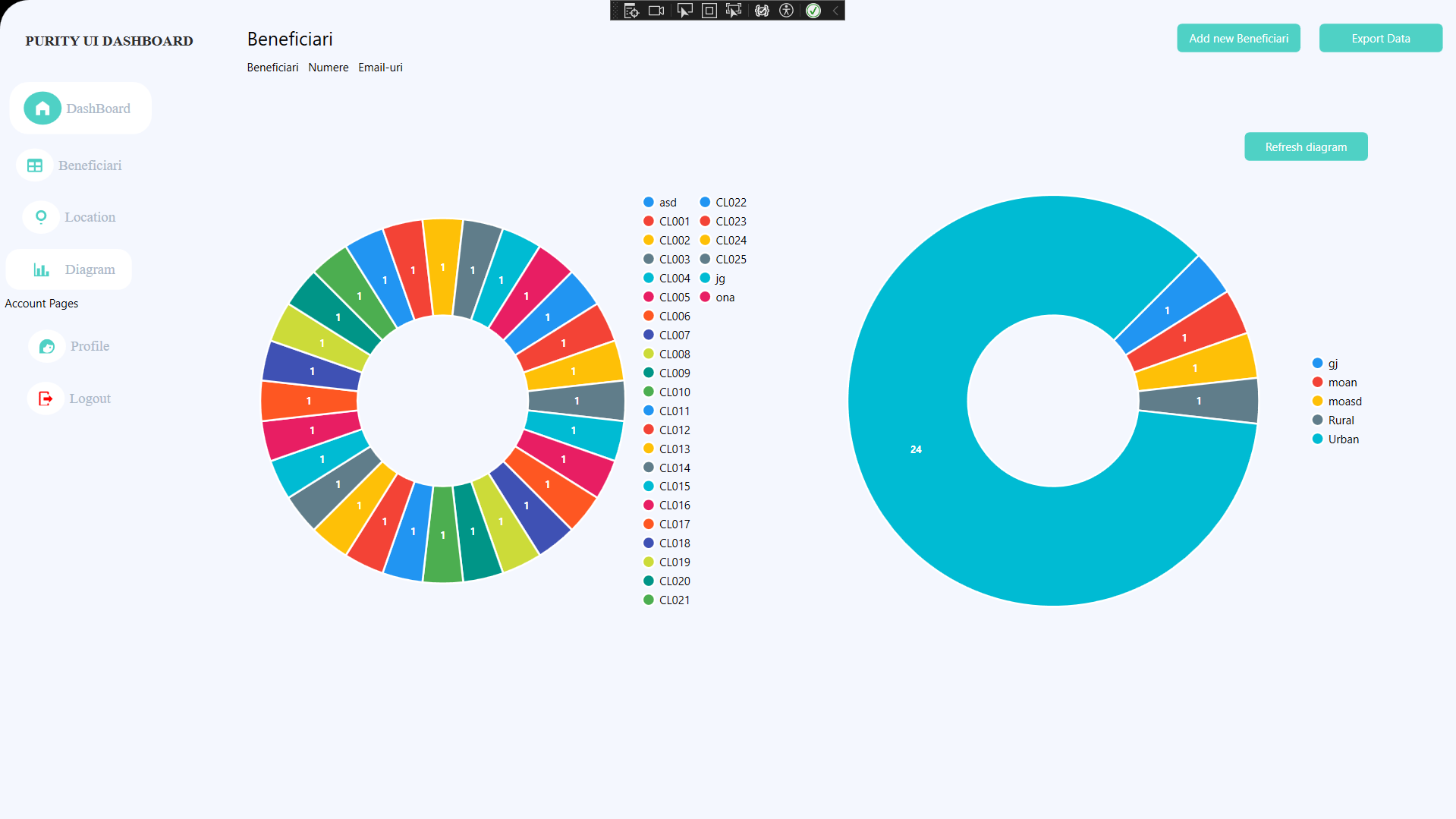












### Concluzie:

În final, produsul program reprezintă o soluție eficientă și robustă pentru gestionarea informațiilor despre beneficiarii unei societăți de distribuție. Cu ajutorul tehnologiilor moderne și a framework-urilor adecvate, am reușit să creăm o aplicație care simplifică procesul de gestionare a datelor și oferă utilizatorilor o experiență plăcută și eficientă. Prin implementarea acestei soluții, am demonstrat capacitatea noastră de a proiecta și dezvolta aplicații complexe, satisfăcând nevoile și cerințele clienților noștri.

### Webografie

1. <https://www.youtube.com/>
2. <https://stackoverflow.com/>