Proiect Baze De Date

-GHEORGHE ANDREI-ALEXANDRU –

-331AA-

*Tema : Sistem de gestiune al unei unități de cazare*

În cadrul acestei teme am optat pentru realizarea unei aplicații desktop folosind framework-ul .NET și limbaj de programare, C#.

**Etapa 1: Proiectarea bazei de date:**

Pornim de la 5 tabele principale ce vor modela atât interacțiunea unității de cazare cu clientul , cât și managementul intern al unității de cazare , ce urmează a fi descrise :

*Tabela Clienti*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Client | int | NU | DA | - |
| CNP | nvarchar(13) | NU | NU | - |
| Nume | nvarchar(50) | NU | NU | - |
| Prenume | nvarchar(50) | NU | NU | - |
| Nationalitate | nvarchar(50) | NU | NU | - |
| Email | nvarchar(50) | NU | NU | UNIQUE |
| Parola | nvarchar(50) | NU | NU | UNIQUE |

Atributul unic de identificare este cuplul Email + Parola asupra cărora se impune constrângere de tip UNIQUE pentru a asigura faptul că un utilizator se va loga in sistem cu propriile credențiale și nimeni altcineva nu se poate loga în sistem cu aceste credențiale.

*Tabela Angajați :*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Angajat | int | NU | DA | - |
| Nume\_Job | nvarchar(13) | NU | NU | - |
| Nume | nvarchar(50) | NU | NU | - |
| Prenume | nvarchar(50) | NU | NU | - |
| Nationalitate | nvarchar(50) | NU | NU | - |
| Email | nvarchar(50) | NU | NU | UNIQUE |
| Password | nvarchar(50) | NU | NU | UNIQUE |
| Salariu | real | NU | NU | - |
| ID\_Departament | int | DA | NU | De tip FK |
| ID\_Manager | int | DA | NU | FK(autoreferinta) |

*Tabela Departamente*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Departament | int | NU | DA | - |
| Nume\_Departament | nvarchar(50) | NU | NU | - |
| Descriere\_Departament | text | NU | NU | - |

*Tabela Facturi*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| Cod\_Factura | int | NU | DA | - |
| Suma\_Plata | varchar(50) | NU | NU | - |
| ID\_Client | int | DA | NU | Tip FK |
| Data\_Facturare | date | NU | NU | - |
| ID\_Camera | int | NU | NU | - |

*Tabela Camere*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Camera | int | NU | DA | - |
| Tip\_Camera | nvarchar(50) | NU | NU | - |
| Descriere\_facilitati | text | NU | NU | - |
| Nr\_locuri | int | NU | NU | - |
| Disponibilitate | int | NU | NU | - |
| Pret\_camera | real | NU | NU | - |

În continuare vom detalia și calcula conform următorului tabel relațiile ce apar între tabelele descrise anterior:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Clienti | Angajati | Departam. | Facturi | Camere |
| Clienti | - | 1: N | - | 1: N | 1:N |
| Angajati | N:1 | 1:N | 1:1 | 1:N | 1:N |
| Departam. | - | 1:N | - | - | - |
| Facturi | 1:1 | 1:1 | - | - | N:1 |
| Camere | N:1 | N:1 | - | 1:N | - |

În continuare vom calcula conform tabelului, relațiile dintre tabele, în urma cărora vom decide dacă există nevoia introducerii unor tabele adiționale:

Clienți – Angajați : 1:N + N:1 = N:N (1)

Clienți – Facturi : 1: N + 1:1 = 1:N

Clienți – Camere : 1: N + N:1 = N:N (2)

Angajați – Angajați : 1: N

Angajați – Departamente : 1: N + 1:1 = 1:N

Angajați – Facturi : 1:N + 1:1 = 1:N

Facturi – Camere : N:1 + 1:N = N:N (3)

Camere – Angajați : N:1 + 1:N = N:N (4)

Observăm că obținem în baza noastră de date 4 relații 1:N , dar și 4 relații N:N (ceea ce în mod normal ar însemna 4 tabele adiționale ) . Totuși , putem minimiza numărul tabelelor la două ce vor fi definite după cum urmează:

*Tabela Menaj\_Camere*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Monitorizare | int | NU | DA | - |
| ID\_Camera | int | DA | NU | Tip FK |
| ID\_Angajat | int | DA | NU | Tip FK |
| Ora\_incepere | datetime | NU | NU | - |
| Ora\_finalizare | Datetime | NU | NU | - |

*Tabela Programări\_Clienți*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nume\_col | Tip\_data | Este FK | Este PK | Constrangeri |
| ID\_Programare | Int | NU | DA | - |
| ID\_Camera | Int | DA | NU | - |
| ID\_Client | Int | DA | NU | - |
| ID\_Factura | Int | DA | NU | - |
| Data\_cazare | Date | NU | NU | - |
| Data\_decazare | Date | NU | NU | - |
| ID\_Receptionist | Int | DA | NU | - |
| Validata | Int | NU | NU | - |

Obținem astfel în total 7 tabele , a căror schema relațională este următoarea :

//aici voi insera poza cu modelul relațional

**Etapa 2 : Stabilirea cerințelor de lucru**

Pe baza acestor tabele vom defini următoarele funcționalități ce vor fi implementate în cadrul aplicației:

1. Autentificare / Login Client
2. Autentificare / Login Angajat
3. Alegere data cazare / decazare pentru client , impreuna cu camerele dorite în limita disponibilității lor
4. Generare factură client ( în format pdf ) , împreună cu generarea unui mail automat către adresa de mail a client-ului în momentul în care programarea a fost validată de recepționist
5. Validarea sau ștergerea programărilor (perspectivă vizibilă pentru recepționeri)
6. Introducerea intervalului de lucru pentru menaj( perspectivă vizibilă menajerelor , bucătarilor)
7. Obținerea unor detalii într-o zonă de raportări privind performanțele angajaților dar și a vânzărilor (Perspectivă vizibilă pentru CEO și Manageri)
8. Posibilitatea adăugării de noi departamente și de asociere a unor angajați cu un anumit manager și un anumit departament ( Perspectivă vizibilă doar pentru CEO)
9. Posibilitatea concedierii angajaților (vizibilă doar pentru CEO)

**Etapa 3 : Realizarea operațiilor de tip UPDATE(U), DELETE(D), INSERT(I)**

Operațiile de acest tip permit gestiunea programărilor de către recepționiști dar și gestiunea utilizatorilor de către sistem fără a exista încurcături. În continuare vom da câteva exemple :

*I1 ) inserarea unui nou utilizator în sistem (angajat / client) :*

INSERT INTO Angajati (Nume, Prenume, Email, Password, Nationalitate, Nume\_Job, Salariu) VALUES('" + nume + "','" + prenume + "','" + email + "','" + parola + "','" + nationalitate + "','"+nume\_job+"','"+salariu+"');-- inserarea unui nou angajat

*I2) inserarea unei noi programări(și automat a unei facturi asociate )*

INSERT INTO Programari\_Clienti(ID\_Camera, ID\_Client, ID\_Factura, Data\_cazare, Data\_decazare, ID\_Receptionist)

VALUES('"+id\_camera+"', '"+id+"', '"+id\_factura+"', '"+zi\_incepere\_sejur+"', '"+zi\_finalizare\_sejur+"', NULL); -- inserarea unei noi programări

Am pus NULL în momentul de față deoarece câmpul ID\_recepționist va fi completat automat în momentul în care recepționistul va valida programarea respectivă.

*U1) Se face operație de tip UPDATE pe tabela angajați în momentul în care li se asociază un manager și un departament.*

UPDATE Angajati SET ID\_Manager = '" + id\_manager + "' , ID\_Departament = '" + id\_departament + "'

WHERE Nume = '"+nume\_ang+"' AND Prenume = '" + prenume\_ang + "' AND Email = '" + email\_ang + "';

*U2) De asemenea se vor face operații de UPDATE pe tabela Camere (pentru setarea validității camerei pe 0 în momentul în care această a fost asociată cu o programare), dar și pe tabela Programări\_Clienți (pentru a asocial programarea cu un receptionist care a procesat comanda respectivă) :*

UPDATE Programari\_Clienti

SET ID\_Receptionist = '"+Form1.id+"'

WHERE ID\_Programare = ANY(SELECT ID\_Programare

FROM Programari\_Clienti

WHERE Data\_cazare = '" + data\_cazare+ "' AND Data\_decazare = '" + data\_decazare + "');

D1) *Se vor face operații de tip DELETE pe tabelele Facturi și Programări\_Clienți de către Recepționiști în momentul decazării clienților și automat se va face și o operație de tip UPDATE pe tabela Camera pentru a actualiza statusul de disponibilitate a camerei ce tocmai a fost eliberată.*

DELETE FROM Programari\_Clienti

WHERE ID\_Programare = (SELECT DISTINCT ID\_Programare FROM Programari\_clienti

WHERE Data\_cazare = '" + dt\_cazare + "' AND Data\_decazare = '" + dt\_decazare + "'AND ID\_Camera = '"+camera+"')

AND Data\_cazare = '" + dt\_cazare + "' AND Data\_decazare = '" + dt\_decazare + "';

D2) *Se va mai face operație de tip DELETE pe tabela Angajați în momentul în care CEO-ul decide să concedieze diverși angajați.*

DELETE FROM ANGAJATI WHERE

Nume = dataGridView1.Rows[e.RowIndex].Cells["Nume"].Value

AND

Prenume=dataGridView1.Rows[e.RowIndex].Cells["Prenume"].Value

AND

Email = dataGridView1.Rows[e.RowIndex].Cells["Email"].Value;

**Etapa 4: Realizarea select-urilor simple(S) și complexe(C).** Pentru ușurința explicării am evidențiat cu negru câmpurile variabile :

S1 . Selectarea tuturor datelor despre programările clientului făcute pe parcursul zilei de azi de către utilizatorul curent logat în sistem

SELECT C.Nume, C.Prenume, F.Suma\_plata, P.Data\_cazare, P.Data\_decazare, P.Validata FROM Clienti C

JOIN Facturi F ON C.ID\_Client = F.ID\_Client

JOIN Programari\_Clienti P ON F.Cod\_Factura = P.ID\_Factura

WHERE C.ID\_Client = id\_user AND F.Data\_Facturare = data

GROUP BY C.Nume, C.Prenume, F.Suma\_plata, P.Data\_cazare, P.Data\_decazare, P.Validata

//S2. Selectarea Departamentelor la care eu , utilizatorul curent sunt manager.

SELECT DISTINCT D.Nume\_Departament FROM Departamente D

JOIN Angajati A ON D.ID\_Departament = A.ID\_Departament

WHERE A.ID\_Manager = Form1.id ;

//S3. Vizualizarea istoricului de programări de către clientul logat curent în sistem

"SELECT C.Nume, C.Prenume, F.Suma\_plata, P.Data\_cazare,

P.Data\_decazare, P.Validata FROM Clienti C

JOIN Facturi F ON C.ID\_Client = F.ID\_Client

JOIN Programari\_Clienti P ON F.Cod\_Factura = P.ID\_Factura

WHERE (C.ID\_Client = id\_user AND F.Data\_Facturare <= data

GROUP BY C.Nume, C.Prenume, F.Suma\_plata, P.Data\_cazare, P.Data\_decazare, P.Validata;

//S4. Selectarea informațiilor despre toți angajații din sistem de la orice department ( un fel de vizualizare a competiției 😊)

SELECT DISTINCT A.Nume, A.Prenume, M.ID\_Camera AS Nr\_Camera FROM Angajati A

JOIN Menaj\_camere M ON A.ID\_Angajat = M.ID\_Angajat

UNION

SELECT DISTINCT A.Nume, A.Prenume, P.ID\_Camera AS Nr\_Camera FROM Angajati A

JOIN Programari\_clienti P ON A.ID\_Angajat = P.ID\_Receptionist

//S5. Nume,Prenume,Salariu crescut cu 20% pentru Menajere cu mai mult de 5 camere lucrate.

SELECT A.Nume, A.Prenume, 1.2\*A.Salariu AS Salariu\_Nou, COUNT(MC.ID\_Camera) AS Nr\_camere\_lucrate

FROM Angajati A

JOIN Menaj\_camere MC ON A.ID\_Angajat = MC.ID\_Angajat

WHERE Nume\_Job = 'Menajera'

GROUP BY A.Nume,A.Prenume,A.Salariu

HAVING COUNT(MC.ID\_Camera) > 5

//S6 . Angajatii care sunt si clienti ai unitatii de cazare la care lucreaza.

SELECT A.Nume,A.Prenume FROM Angajati A

JOIN Clienti C ON A.Email = C.Email

//C1. TOP 10 Angajati ai căror manager sunt eu

SELECT TOP 10 A.Nume,A.Prenume

FROM Angajati A

WHERE A.ID\_Manager = (SELECT A1.ID\_Angajat FROM Angajati A1 WHERE A1.ID\_Angajat = id\_manager)

ORDER BY A.Salariu,A.Nume,A.Prenume DESC

//C2. Angajații care câștigă cel mai mult la fiecare departament

SELECT A.Nume,A.Prenume,A.Salariu FROM Angajati A

WHERE EXISTS

(SELECT MAX(C.Salariu), D.ID\_Departament From Angajati C

JOIN Departamente D ON C.ID\_Departament = D.ID\_Departament

GROUP BY D.ID\_Departament

HAVING MAX(C.Salariu) = A.Salariu);

//C3. Receptionistul cu numarul cel mai mare de programari realizate

SELECT A.Nume,A.Prenume FROM Angajati A

WHERE ID\_Angajat =

(SELECT PC1.ID\_Receptionist

FROM Programari\_clienti PC1

GROUP BY PC1.ID\_Receptionist

HAVING COUNT(PC1.ID\_Programare) =

(SELECT TOP 1 COUNT(PC.ID\_Programare) AS Nr\_Programari

FROM Programari\_clienti PC

GROUP BY PC.ID\_Receptionist

ORDER BY COUNT(PC.ID\_Programare) DESC))

//C4. Clienții cu cel mai lung istoric de programari

SELECT C.Nume,C.Prenume

FROM Clienti C

WHERE C.ID\_Client = ANY(

SELECT TOP 10 PC.ID\_Client AS Nr\_Programari

FROM Programari\_clienti PC

GROUP BY PC.ID\_Client

ORDER BY COUNT(PC.ID\_Programare) DESC)

**Etapa 5 : Cererile enunțate mai sus au fost integrate în cadrul aplicației folosind C# .**

Desigur , operațiile prezentate mai sus sunt doar câteva din cele ce au fost integrate de-a lungul realizării aplicației , fiind necesare mult mai multe select-uri ușoare și/sau complexe , respectiv operații de tip UPDATE/INSERT/DELETE.

În continuare voi relata logica din spatele aplicației pentru a avea o priveliște de ansamblu mult mai bine conturată .

Am început prin crearea a 3 clase separate, clasa de bază fiind Users.cs din care derivă clasele Clienti.cs și Angajat.cs

Users.cs – aici se află definite metodele de clasificare a utilizatorilor ,de verificare a existenței utilizatorilor în sistem, dar și de clasificare a tipului de utilizator :

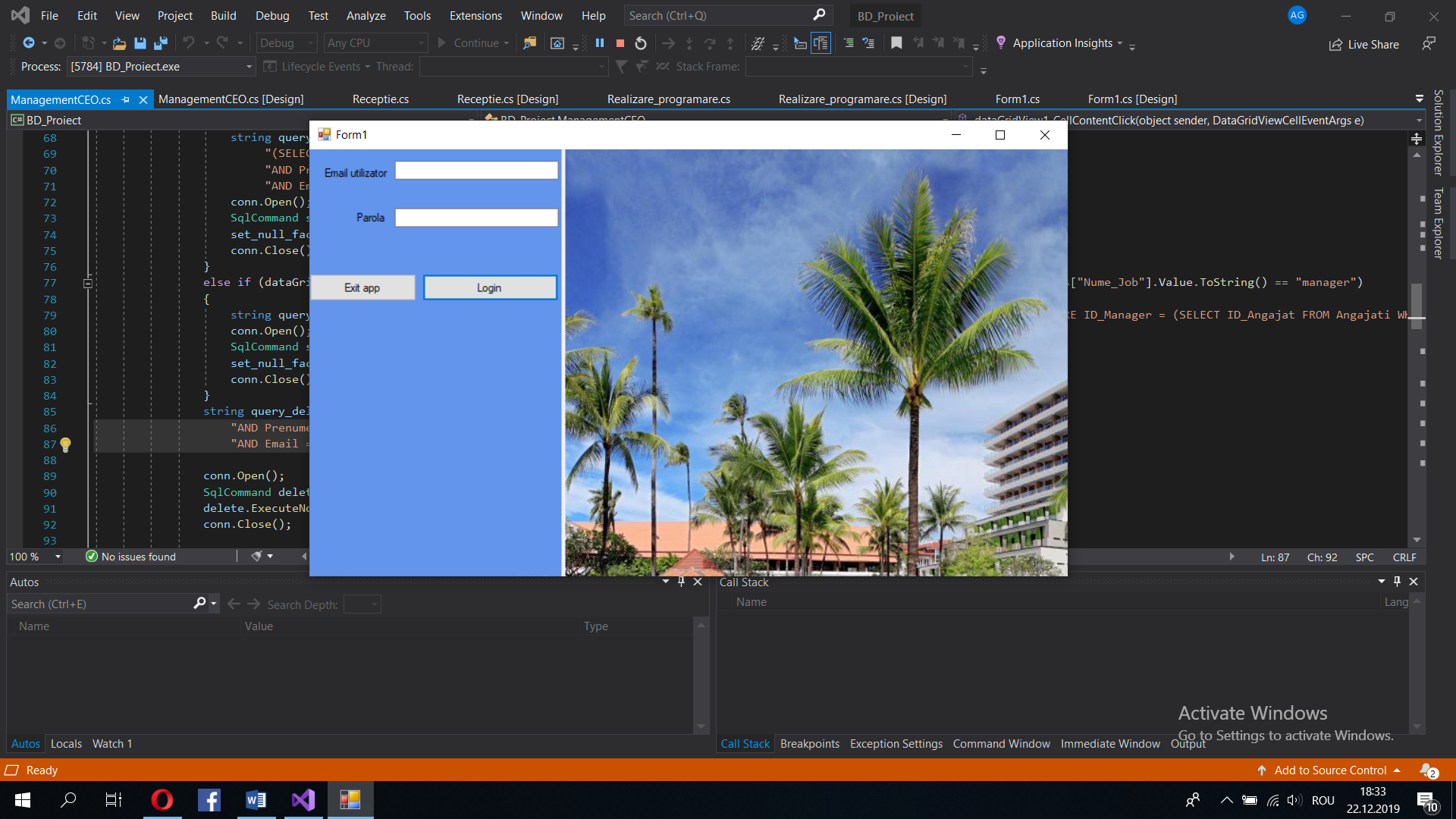
1. Utilizator privilegiat – CEO și Manager
2. Utilizator semi-privilegiat – Receptioner
3. Utilizator neprivilegiat – Menajeră și Bucătar
4. Utilizator special – Clientul

Clienti.cs – aici este definită metoda ce se ocupă de inserarea unui nou client în sistem

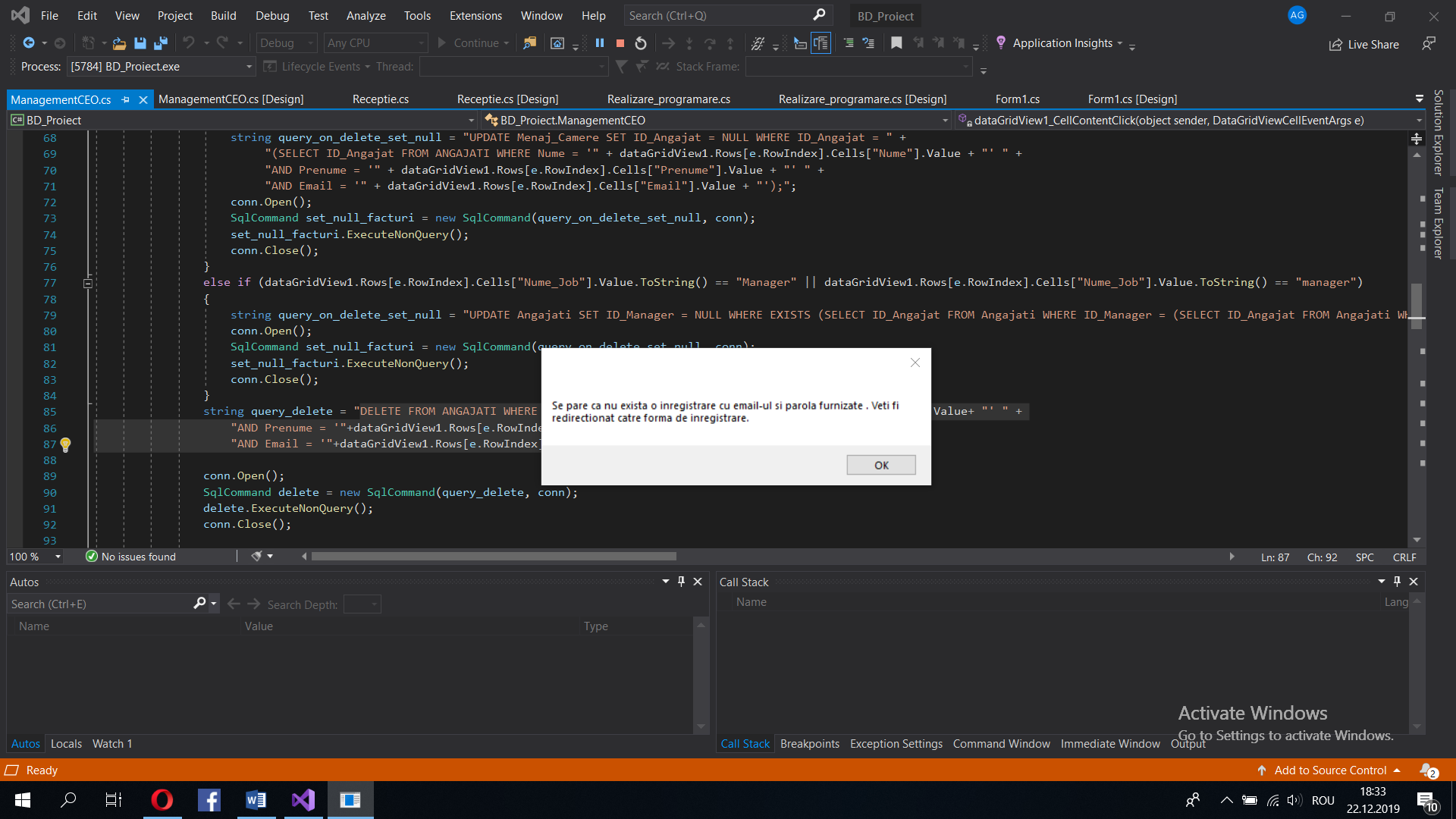
Angajat.cs – aici este definită metoda ce se ocupă de introducerea unui angajat în sistem

Primele forme cu care un utilizator (indiferent că este Client sau Angajat) se ”confruntă” sunt Form1.cs (forma de Login) și Inregistrare\_utilizator.cs(forma de Inregistrare) .

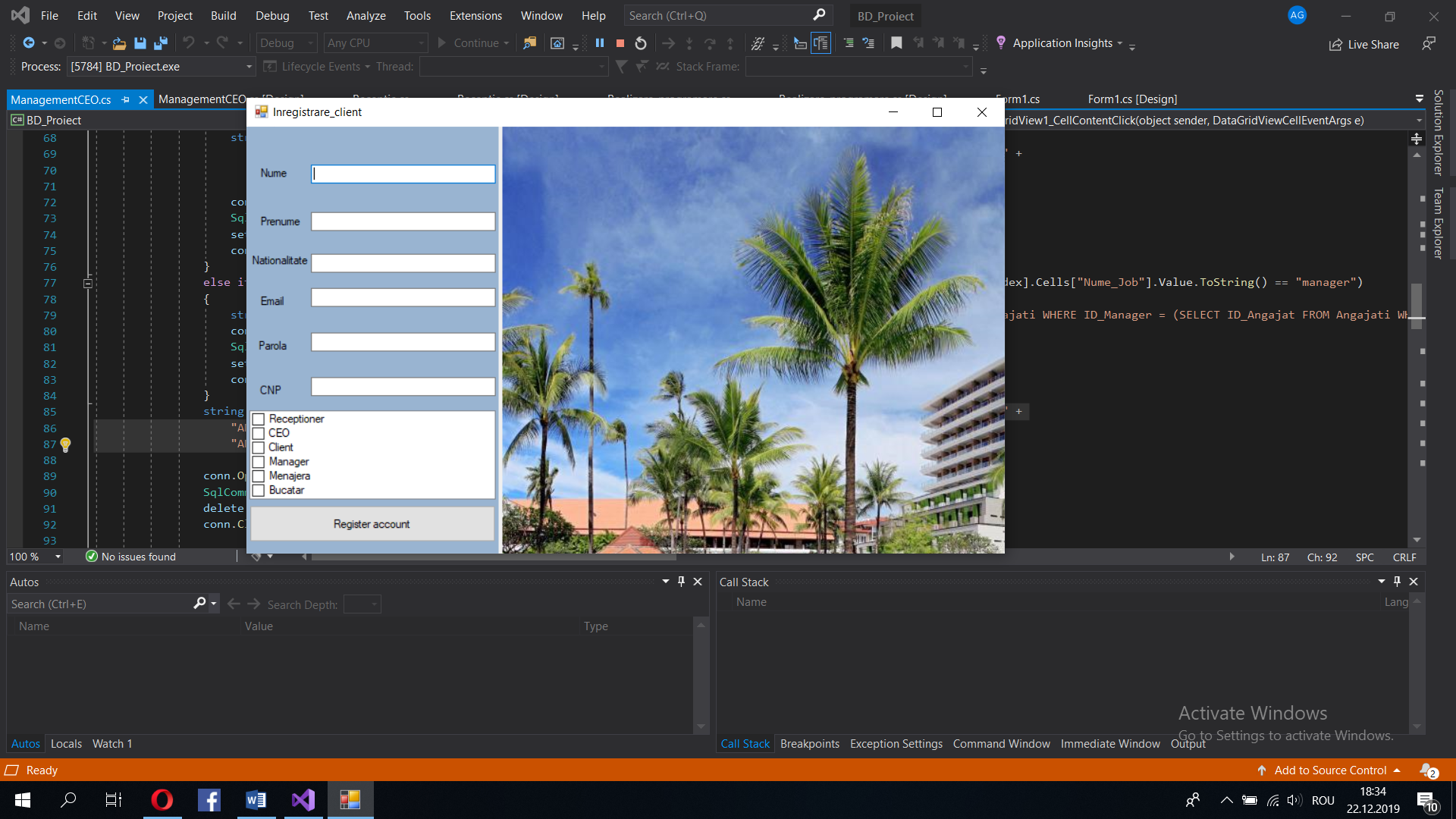
Astfel, un utilizator se poate loga în sistem , iar dacă acesta nu se regăsește în sistem este redirecționat către forma de înregistrare pentru a își crea un cont nou cu care se vor loga ulterior în sistem.



Forma de Login



Mesaj de redirecționare către forma de înregistrare

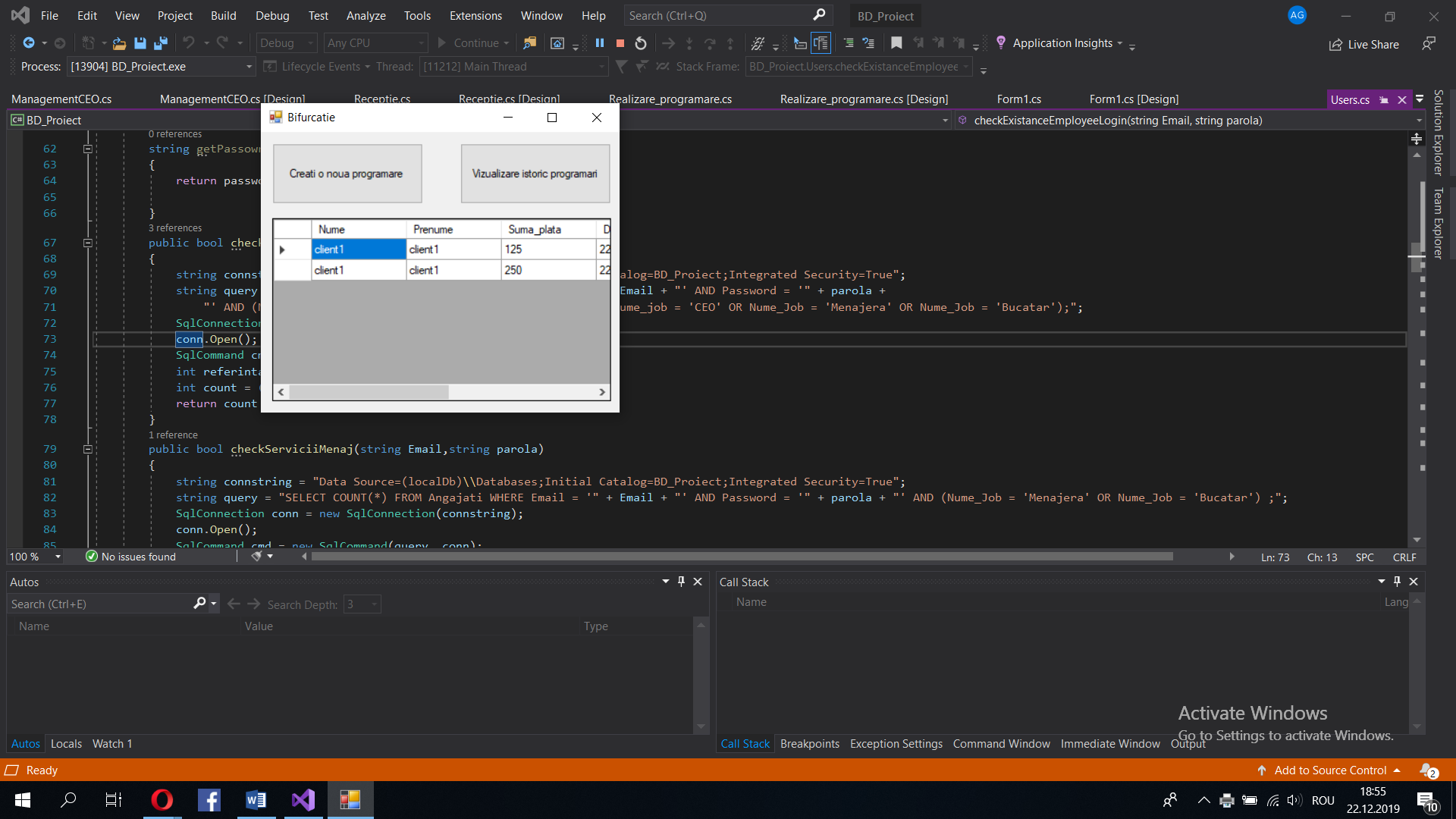


Forma de înregistrare

În funcție de tipul de utilizator, utilizatorul poate fi întâmpinat de următoarele forme :

1. ***Pentru utilizator special ( tip Client)***

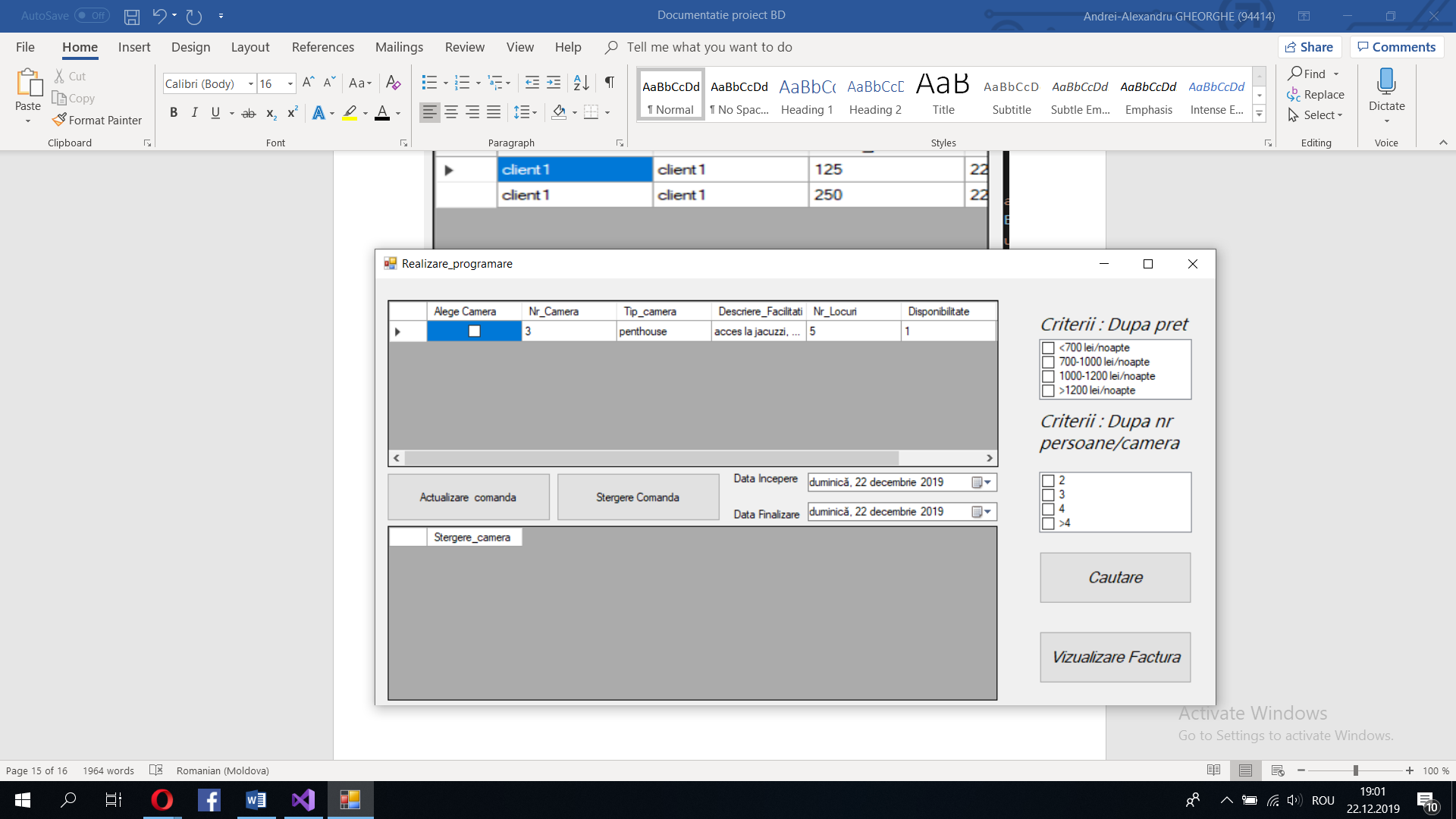
Bifurcatie.cs – specifică utilizatorului client deoarece acesta poate fie să vizualizeze comenzi trecute (împreună cu statusul lor – validate / nevalidate) sau să realizeze o nouă comandă avansând căre forma Realizare\_Programare.cs .

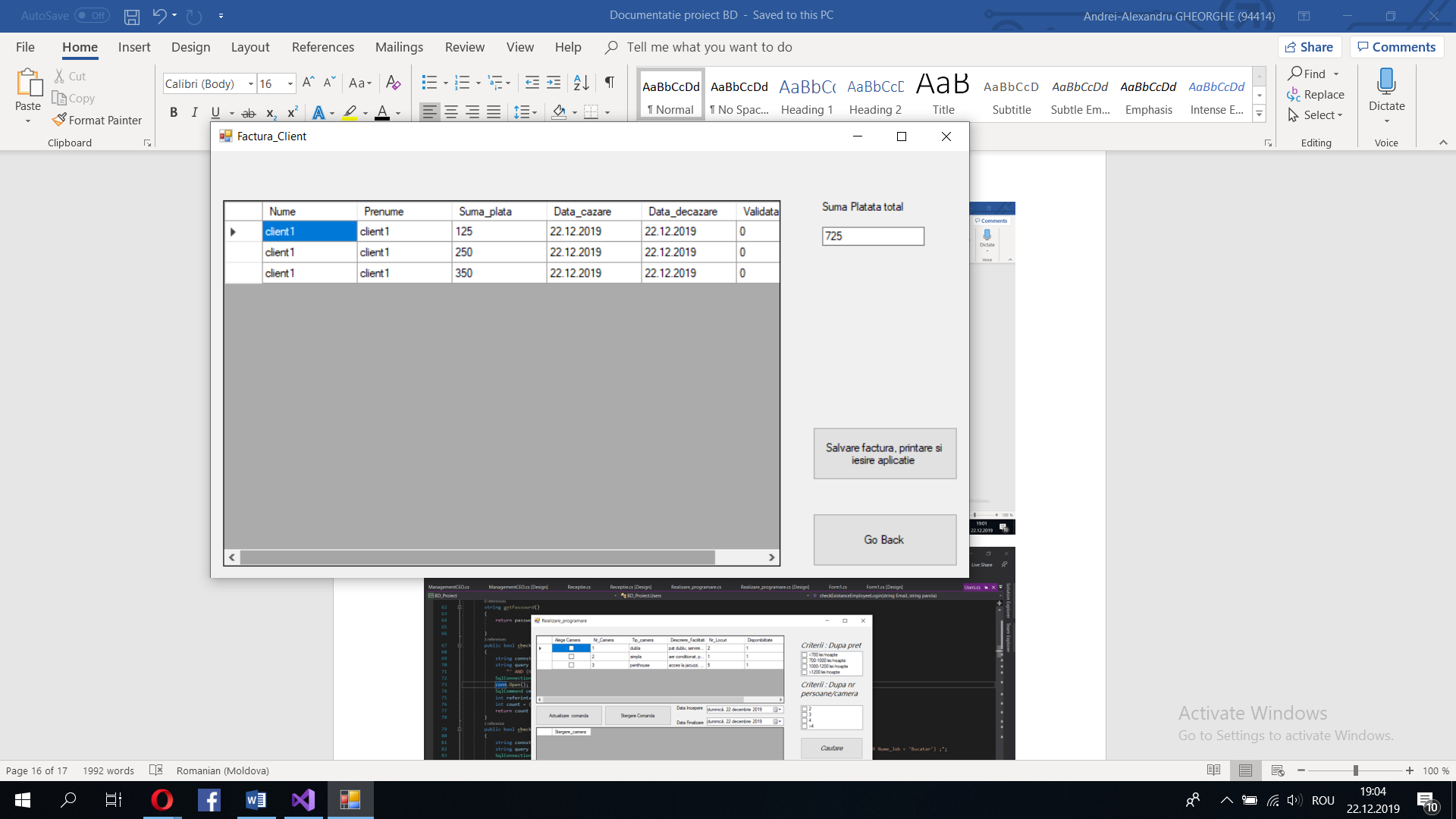


Realizare\_programare.cs – formă în cadrul căreia utilizatorul poate aplica asupra camerelor disponibile filtre de tip preț / nr. Persoane ce pot exista în camera respectivă selectând filtrul dorit și apoi , apăsând butonul ”Căutare”.

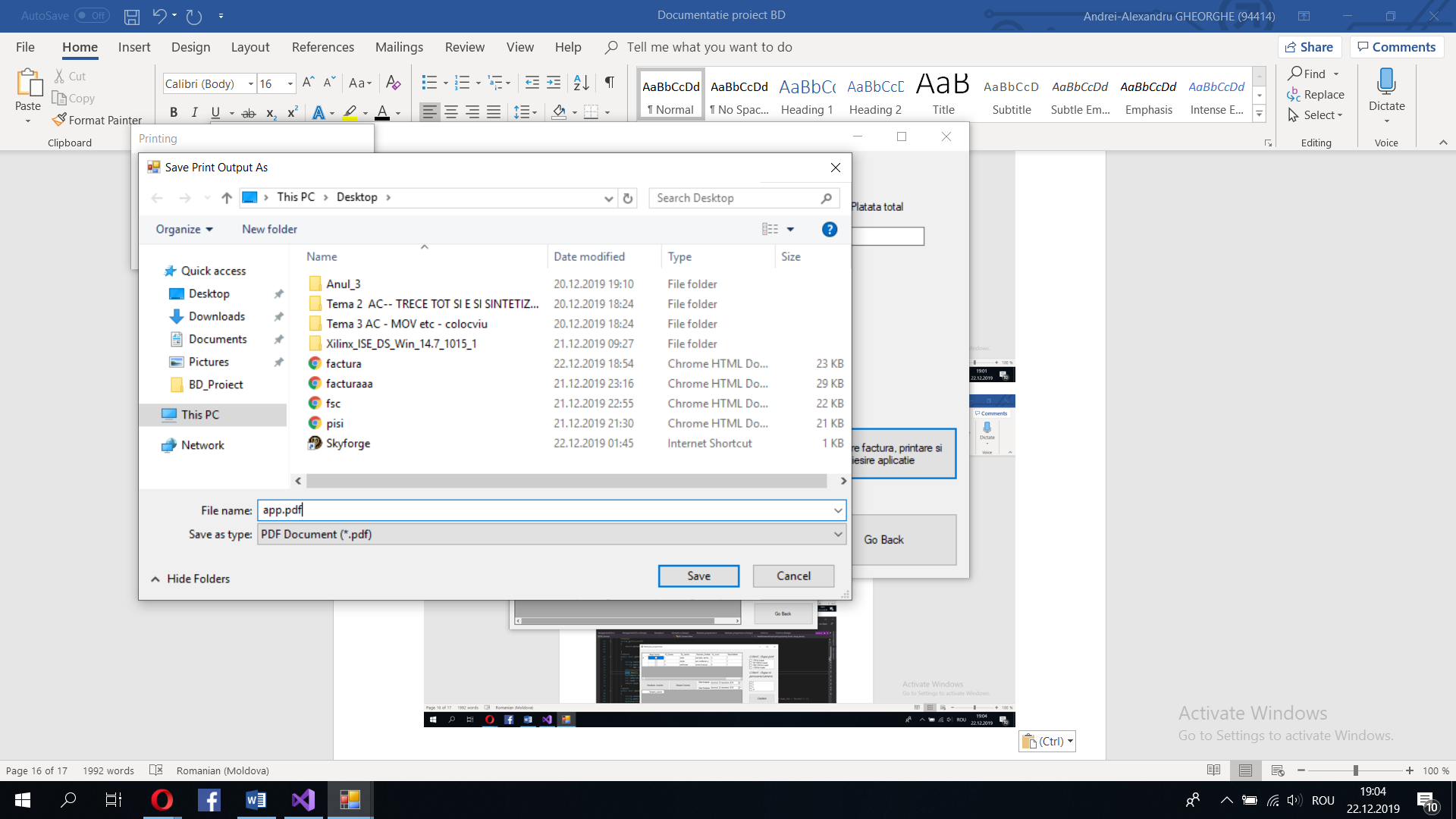
Apoi utilizatorul bifează căsuța în dreptul camerei la care vrea să se programeze , își alege dățile de cazare și decazare , urmând să apese butoanele ”Actualizare Comandă ” sau ”Stergere Comandă” pentru modificarea comenzii , respectiv ”Finalizare Comanda” pentru a salva modificările făcute.

Utilizatorul va fi redirecționat către forma Factura\_Client.cs , unde utilizatorul poate observa suma de plată și detaliile despre factura sa, cu posibilitatea salvării unei copii de factură în format pdf.

Forma Realizare\_Programare.cs unde utilizatorul poate alege camerele pe care dorește să le rezerve .



O factură generată după actualizarea comenzii

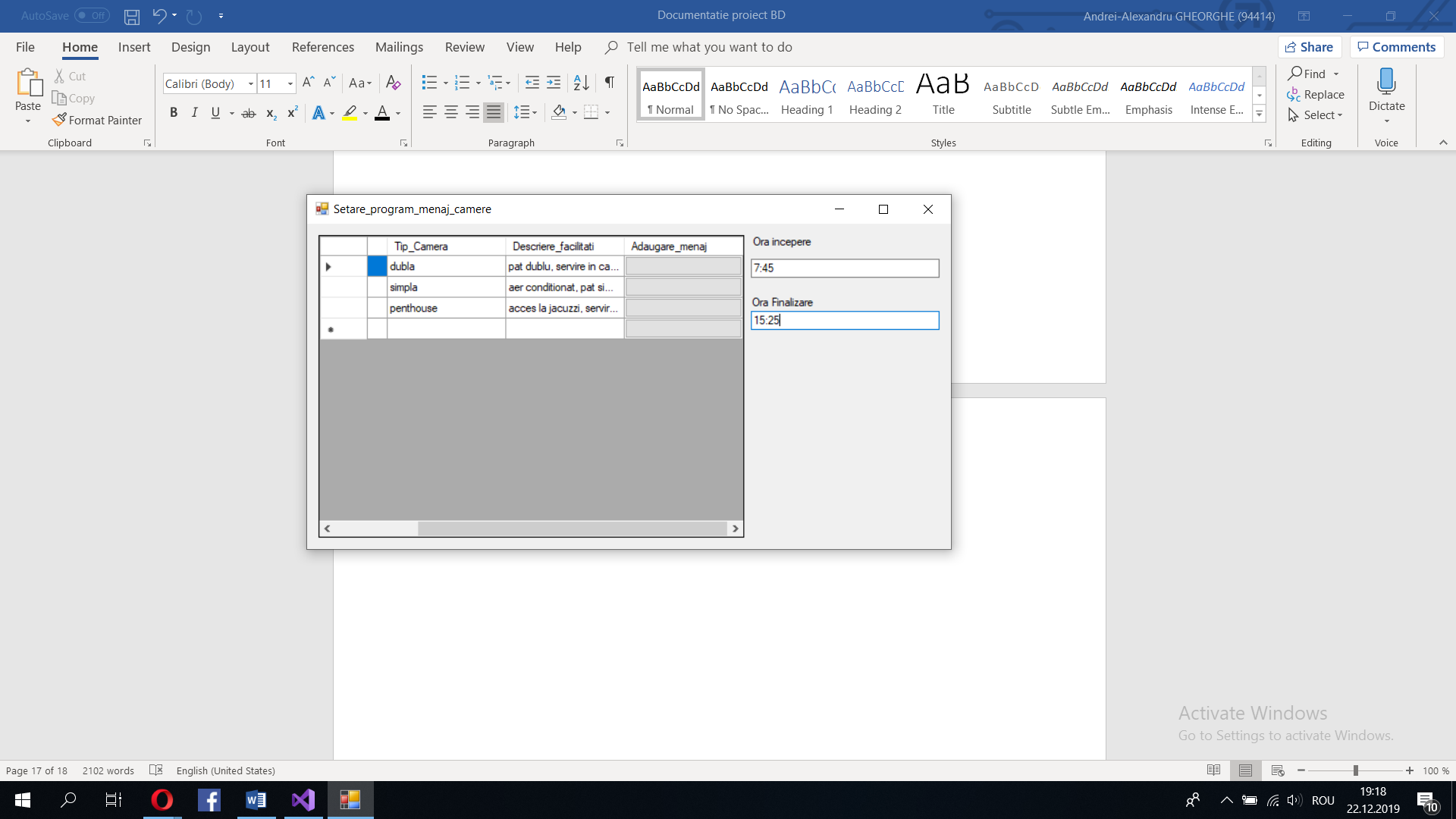


Salvarea unei facturi în format .pdf

După salvarea facturii, aplicația se va închide automat și sesiunea se va distruge.

1. ***Pentru utilizator neprivilegiat(Bucătar/Menajeră)***

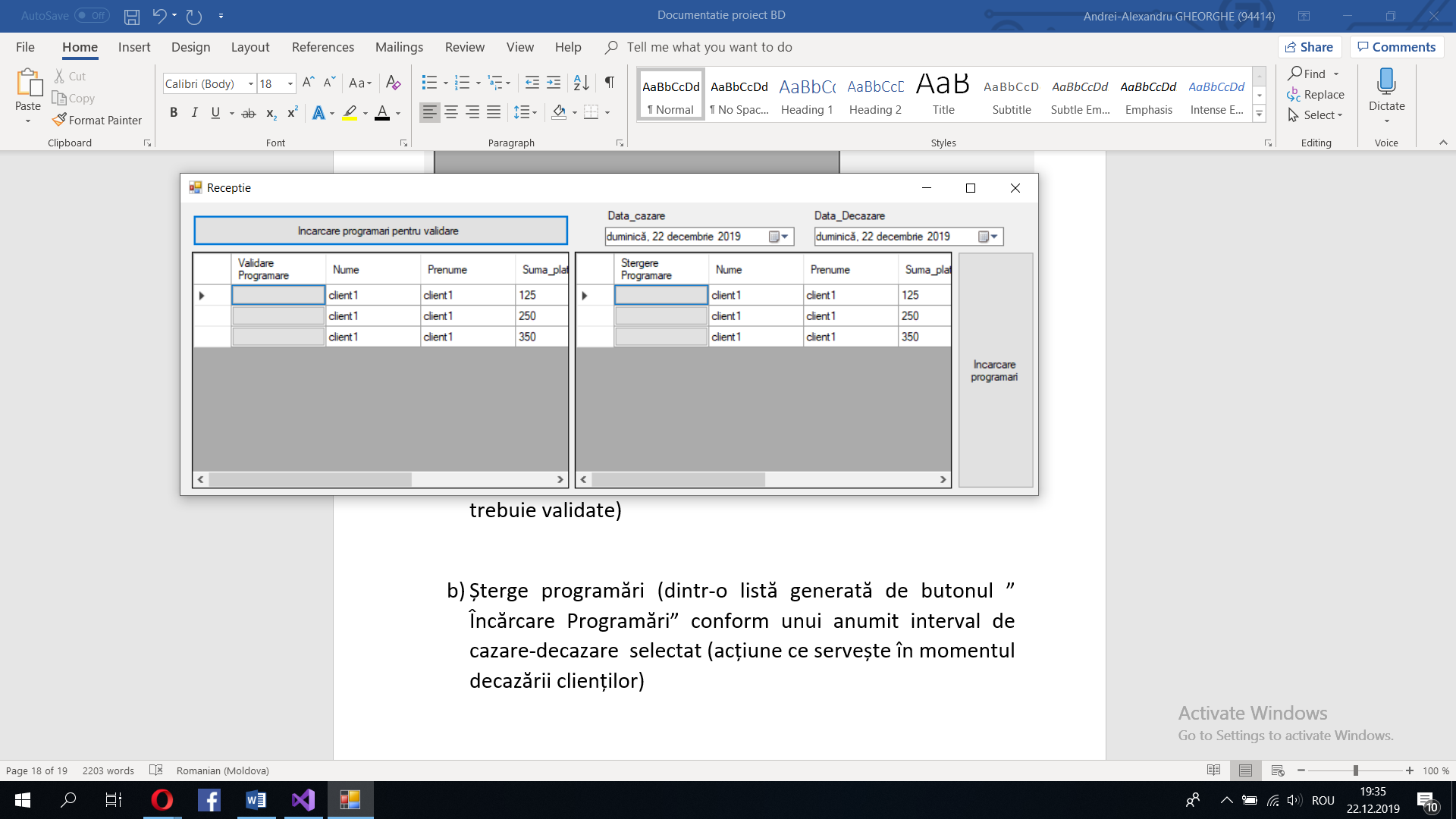
Acești utilizatori au cea mai puțină interacțiune cu sistemul , astfel că după ce ei trec de etapa Înregistrare-Login despre care am discutat mai înainte sunt întâmpinați de forma ”Setare\_Program\_Menaj\_Camere”, în cadrul căreia , utilizatorul își alege camera(camerele de care se ocupă și notează intervalul orar în care se vor ocupa de acele camere) . După apăsarea butonului ”Adăugare\_Menaj” , se vor asocia camerele selectate cu intervalul orar propus de utilizator și utilizator poate ieși din sistem apăsând butonul X din dreapta – sus.



1. ***Pentru utilizator semi-privilegiat (Recepționer)***

Acest tip de utilizator este întâmpinat după logare de forma ”Receptie.cs” în care un recepționer poate :

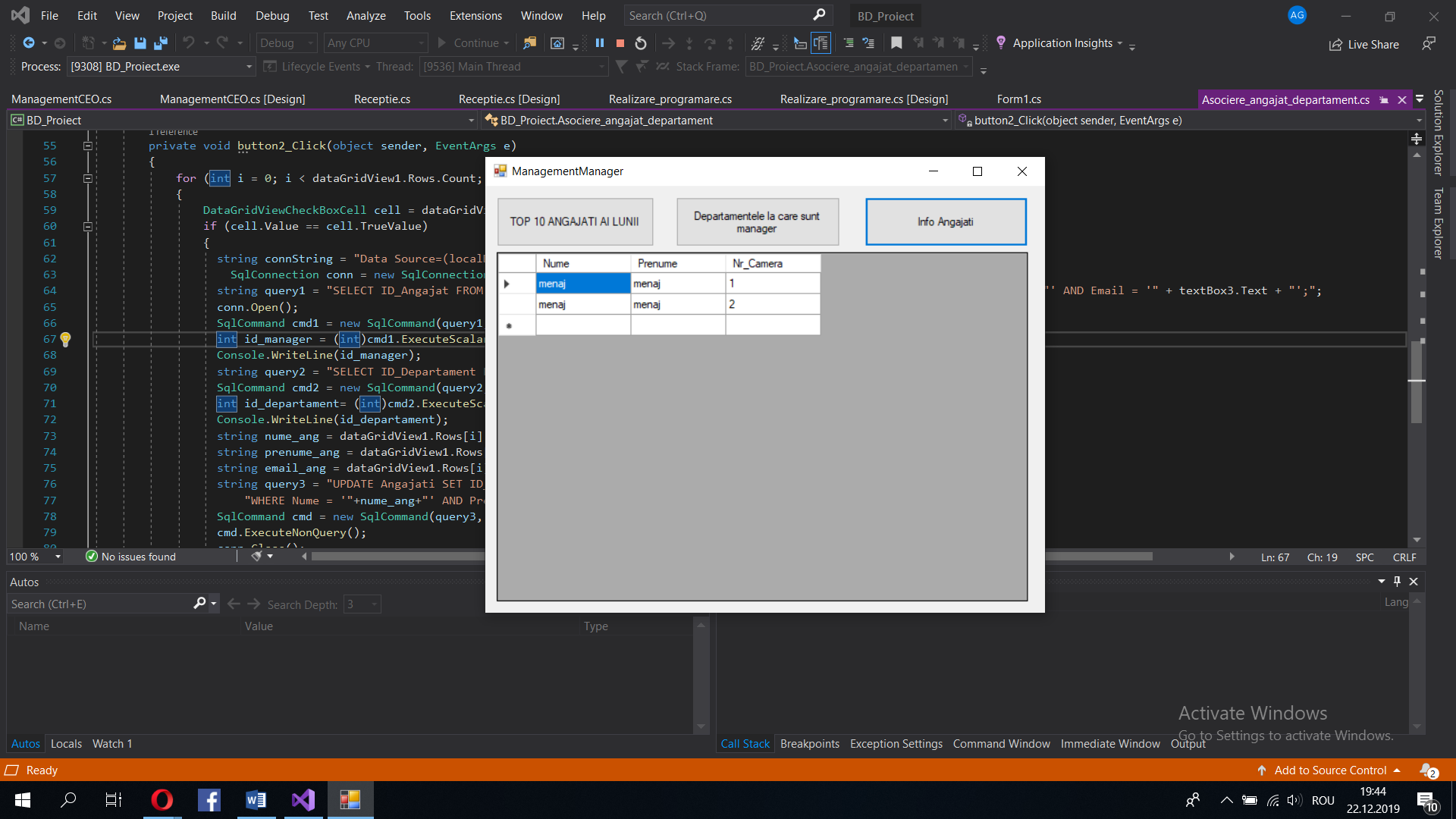
1. Valida programări ce nu au fost validate încă (Se apasă butonul de încarcare programări pentru validare și apoi butonul Validare\_programare în dreptul programării ce trebuie validate)
2. Șterge programări (dintr-o listă generată de butonul ” Încărcare Programări” conform unui anumit interval de cazare-decazare selectat (acțiune ce servește în momentul decazării clienților)



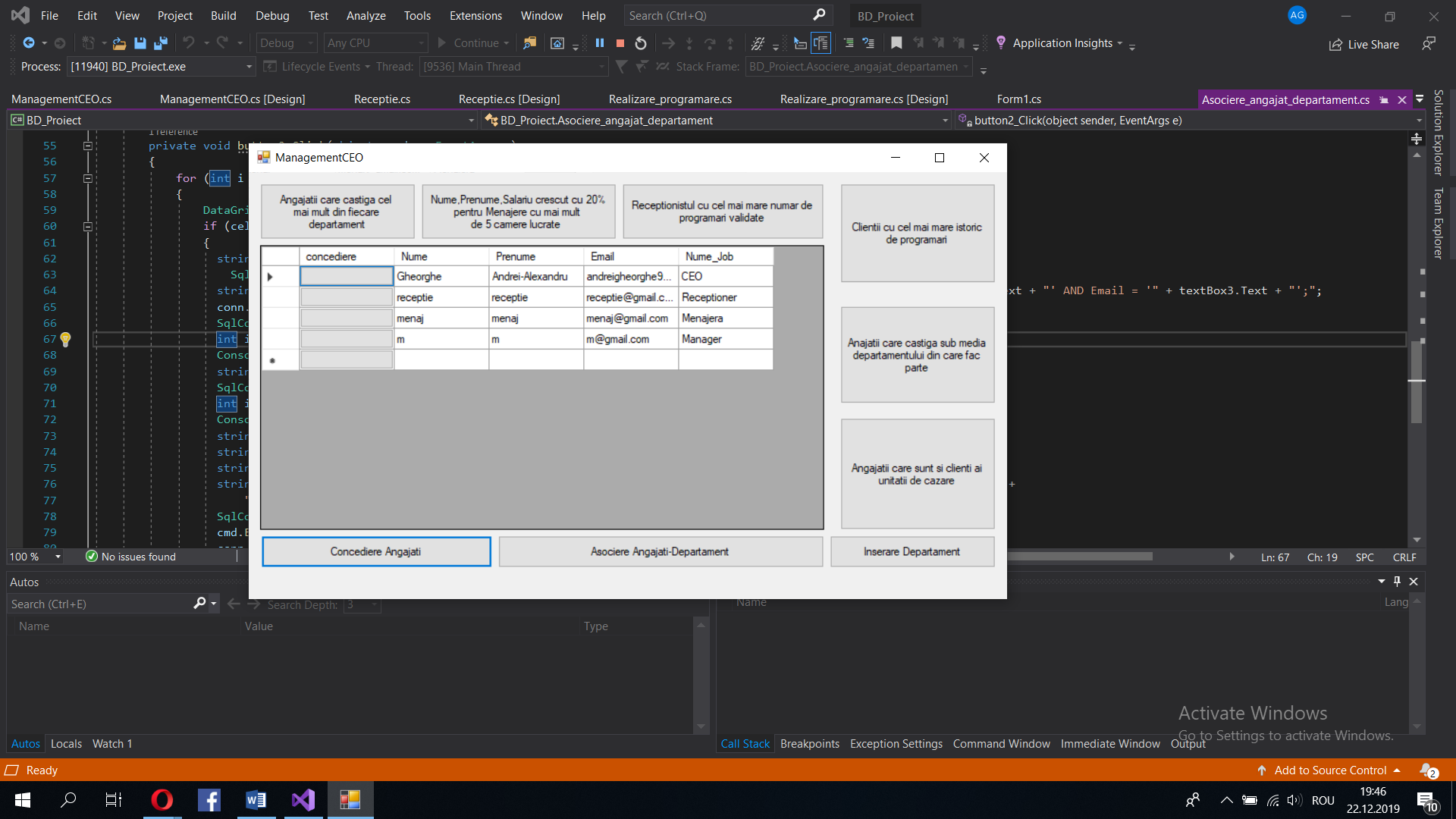
Forma Receptie.cs

1. ***Utilizator privilegiat – Manager și CEO***

Pentru acest tip de utilizator s-au creat formele Management\_Manager , respectiv Management\_CEO ce constituie o zonă de raportări în care au fost incluse majoritatea cererilor simple și complexe ce au trebuit realizate în cadrul proiectului. Pe lângă aceste funcționalități am adăugat posibilitatea CEO-ului de a adăuga departamente ( în cadrul unei forme numite AddDepartament.cs ) , de concediere a angajaților ,respectiv de asociere a unor angajați cu un departament și manager .

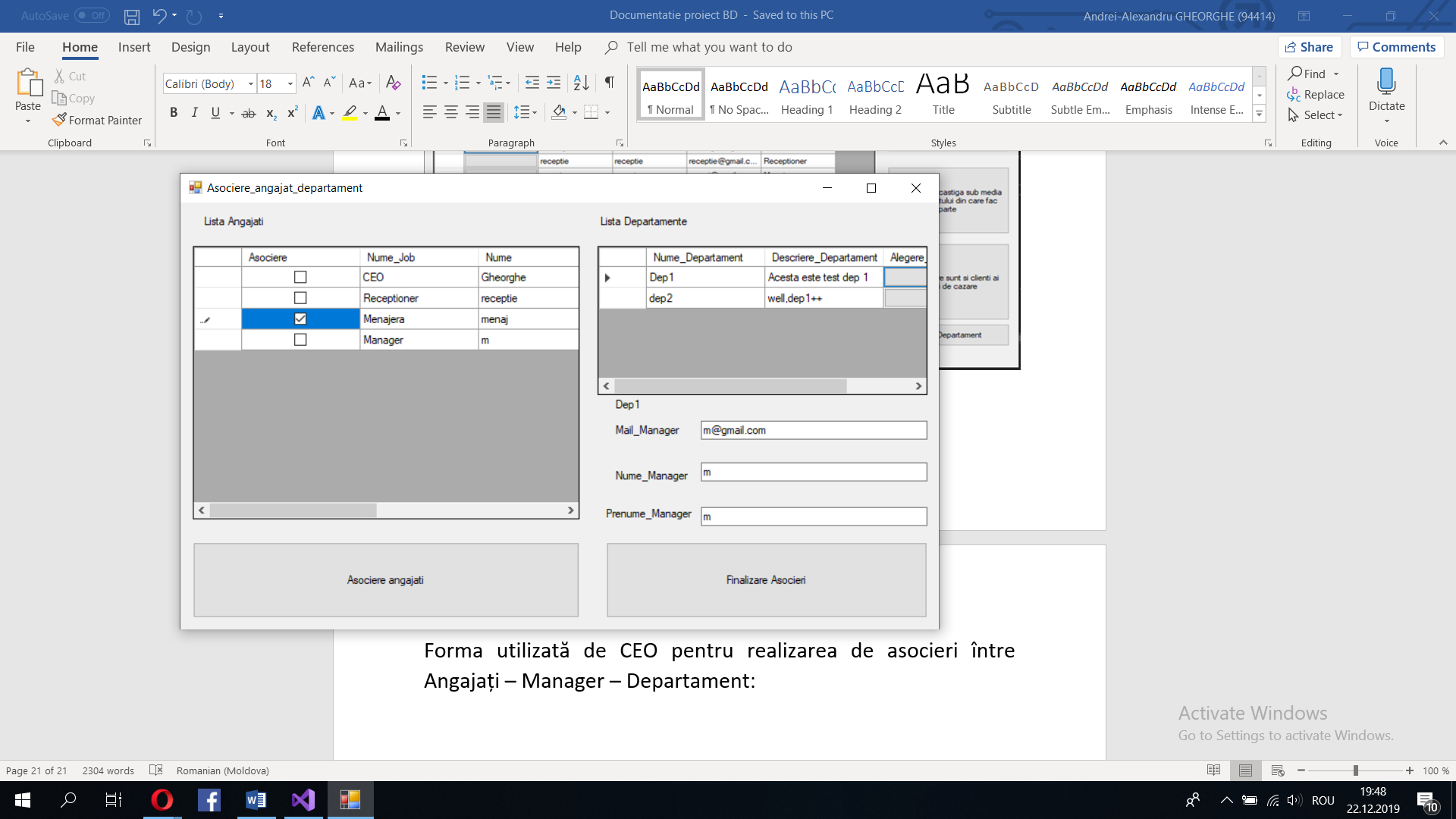


Forma specifică utilizatorului de tip Manager



Forma specifică CEO-ului

Forma utilizată de CEO pentru realizarea de asocieri între Angajați – Manager – Departament:



Forma prin intermediul căreia se realizează inserarea departamentlor în sistem :

