Student : Sveduneac Ioan

Grupa : 3123A

**Documentatie Proiect**

**PIU**

**Tema proiect : Aplicatie de gestionare service auto**

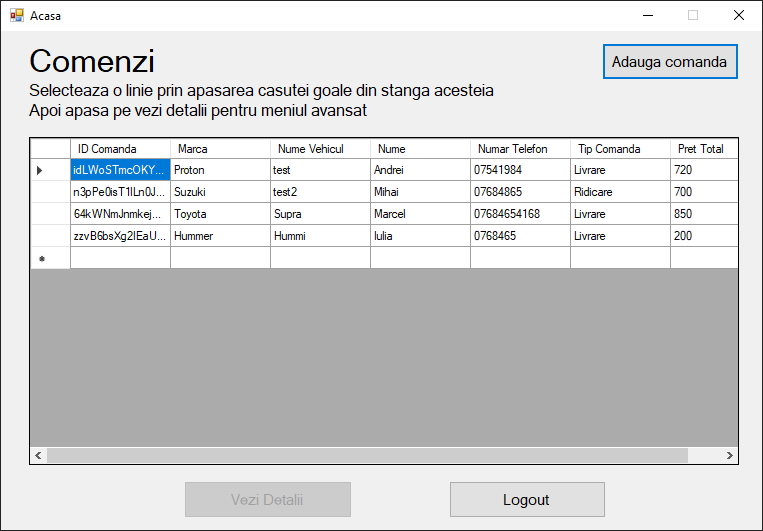
Aplicatia va fi folosita pentru a gestiona un service auto cu un set predefinit de servicii oferite, care sunt alese de client la adaugarea unei comenzi noi sau la editarea unei comenzi existente.

**Functionalitate din punct de vedere al utilizatorului**

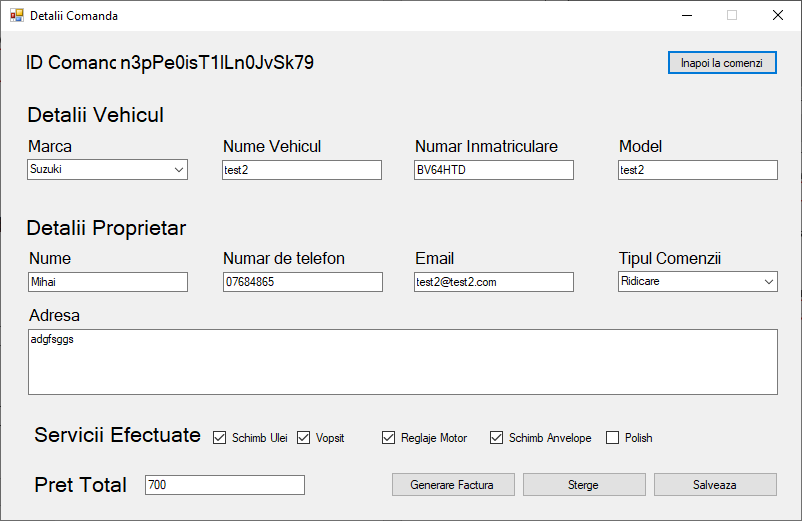
La lansarea aplicatiei utilizatorul va fi intampinat cu un meniu de logare cu ajutorul email-ului si a parolei, acesta fiind intampinat cu mesaje suggestive in cazul in care email-ul sau parola sunt gresite sau nu au fost completate.



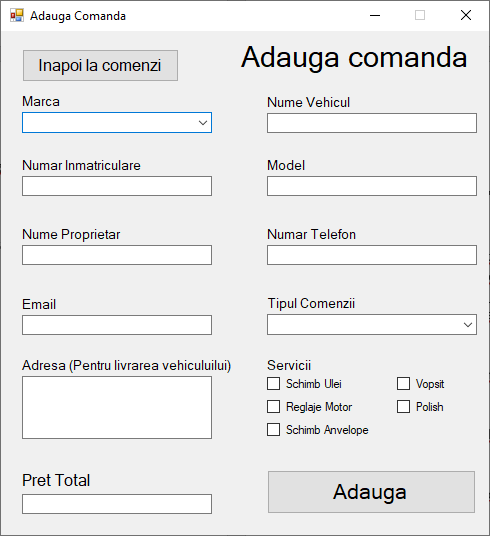
Dupa logare isi face aparitia un meniu care afiseaza comenzile deja existente, cu optiunea de a vedea detaliile unei comenzi specifice, de a adauga o comanda noua sau de logout.



Daca este accesata fereastra de detalii a unei comenzi se pot vizualiza toate detaliile acelei comenzi si se poate edita oricare din campurile afisate apoi prin apasarea butonului de salvare aceste modificari sunt salvate. Tot din acest meniu de poate realiza stergerea comenzii sau afisarea facturii actuale a comenzii intr-un fisier pdf pentru a fi posibila printarea acesteia.



Prin accesarea meniului de adaugare comanda utilizatorul poate adauga o comanda noua prin completarea tuturor campurilor apoi prin apasarea butonului ‘Adauga’.



**Functionalitate din punct de vedere al programatorului**

**Formularul de login**

Acest cod se ocupă de funcționalitatea de autentificare a utilizatorilor în aplicație.

Principalele componente ale codului sunt următoarele:

1. Importarea bibliotecilor necesare:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

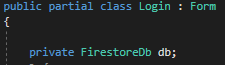
using Google.Cloud.Firestore;

Biblioteca Google.Cloud.Firestore este necesara pentru a lucra cu Google Cloud Firestore, care este un serviciu de bază de date în cloud pe care l-am folosit pentru salvarea datelor.

2. Declararea clasei Login:

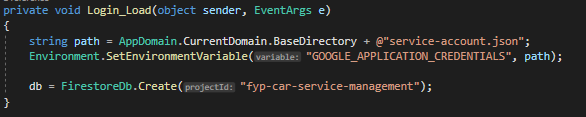
Aceasta este clasa formularului de autentificare. Este derivată din clasa `Form` și conține codul pentru evenimentele asociate cu formularul.

3. Inițializarea obiectului FirestoreDb:



Obiectul `db` este utilizat pentru a comunica cu baza de date Firestore. El va fi inițializat în evenimentul `Login\_Load` al formularului.

4. Inițializarea obiectului FirestoreDb în evenimentul `Login\_Load`:



Aceste linii de cod inițializează obiectul `db`. Calea către fișierul de configurare JSON pentru autentificarea în Firestore este specificată în variabila `path`. Apoi, variabila de mediu `GOOGLE\_APPLICATION\_CREDENTIALS` este setată cu calea către fișierul JSON. Apoi, obiectul `db` este creat utilizând numele proiectului Firestore.

5. Evenimentul `login\_btn\_Click`:

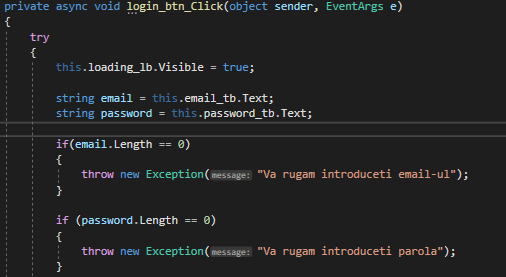
Acest eveniment este declanșat când utilizatorul apasă butonul de autentificare.

6. Autentificarea utilizatorului:

Se preiau valorile introduse de utilizator pentru email și parolă din câmpurile de text corespunzătoare.

7. Verificarea introducerii email-ului și parolei:

Aceste verificări asigură că email-ul și parola au fost introduse înainte de a continua procesul de autentificare. Dacă unul dintre acestea lipsește, se va arunca o excepție cu un mesaj corespunzător.

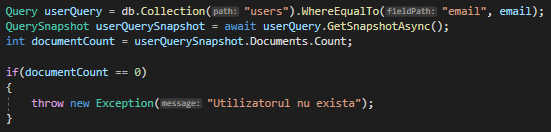


8. Interogarea bazei de date pentru utilizatorul specificat.

Se realizează o interogare în colecția "users" a bazei de date pentru a obține utilizatorul cu email-ul specificat. Rezultatul interogării este un `QuerySnapshot` care conține documentele rezultate. `documentCount` reprezintă numărul de documente returnate.

9. Verificarea existenței utilizatorului:

Se verifică dacă există vreun document returnat de interogare. Dacă nu există, se aruncă o excepție cu un mesaj corespunzător.



10. Verificarea corectitudinii parolei:



Se obține primul document returnat și se converteste într-un dicționar pentru a accesa valorile acestuia. Parola utilizatorului este comparată cu parola introdusă de utilizator. Dacă parolele nu se potrivesc, se aruncă o excepție cu un mesaj corespunzător.

11. Deschiderea formularului Acasa:



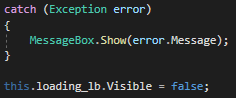
Dacă autentificarea a fost reușită, se deschide formularul Acasa și se ascunde formularul de autentificare.

12. Gestionarea erorilor:

În cazul în care apare o excepție în timpul procesului de autentificare, aceasta este prinsă și un mesaj de eroare este afișat utilizatorului într-o casetă de dialog.

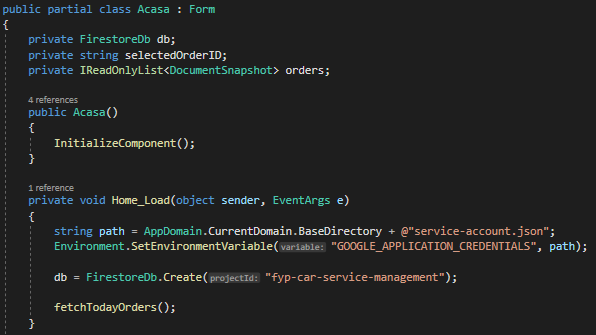
13. Vizibilitatea etichetei `loading\_lb`:

La sfârșitul evenimentului, se setează vizibilitatea etichetei `loading\_lb` la `false`. Aceasta poate fi utilizată pentru a indica utilizatorului că se încarcă ceva în timpul procesului de autentificare.

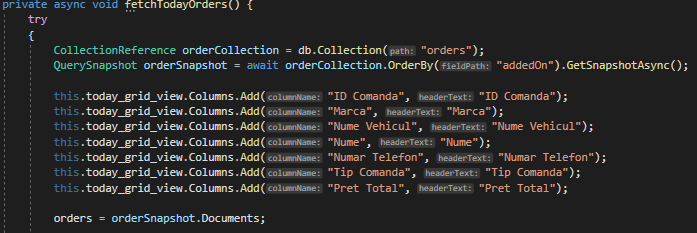


**Pagina de ‘Acasa’**

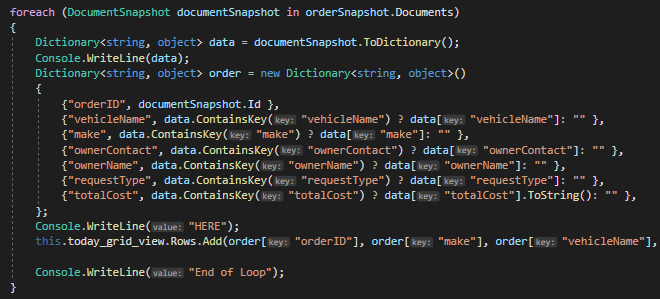
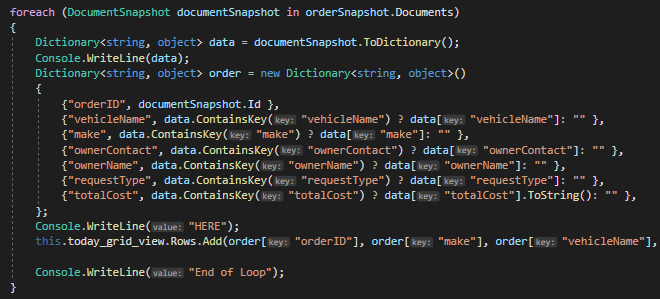
1. La încărcarea paginii (Home\_Load), se realizează inițializarea bazei de date Firestore și se apelează metoda fetchTodayOrders pentru a obține comenzile de astăzi din colecția "orders".



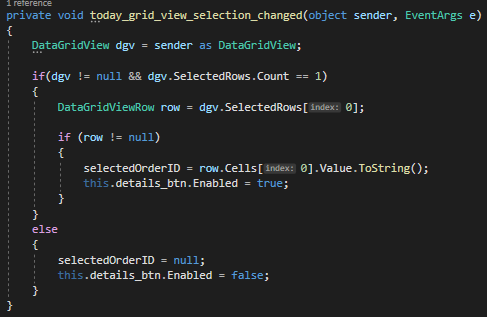
2. Metoda fetchTodayOrders face o interogare în baza de date pentru a obține comenzile de astăzi și le afișează într-un DataGridView (today\_grid\_view).



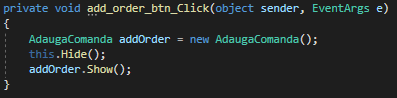
3. Pentru fiecare comandă obținută, se creează un obiect order sub formă de dicționar cu informațiile relevante, care este apoi adăugat ca rând în DataGridView.

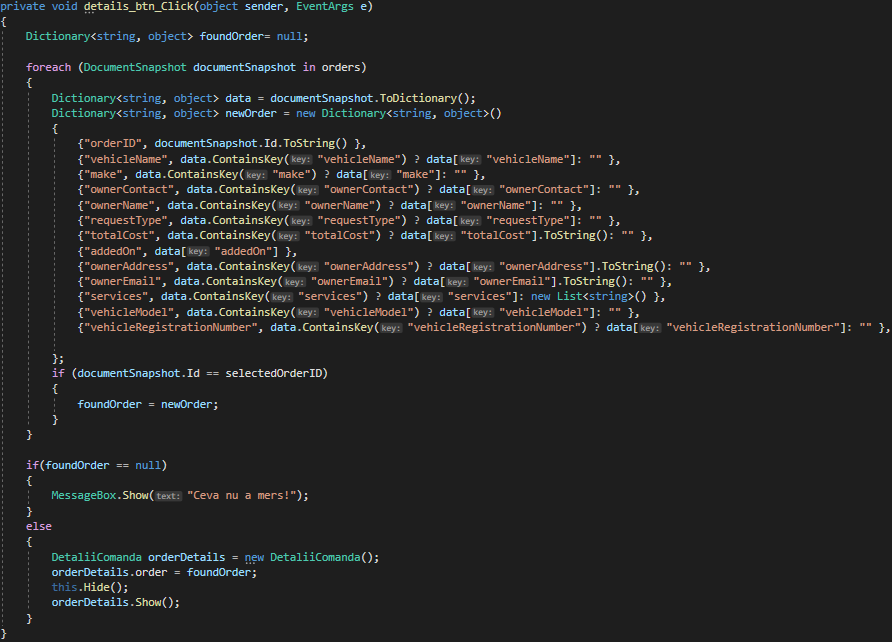
4. În metoda today\_grid\_view\_selection\_changed, se gestionează evenimentul de selectare a unei comenzi în DataGridView. Se obține ID-ul comenzii selectate și se activează butonul ‘Detalii’.



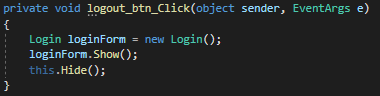
5. La apăsarea butonului "Adaugă comandă" (add\_order\_btn\_Click), se deschide fereastra AdaugaComanda și se ascunde fereastra curentă (Acasa).



6. La apăsarea butonului "Detalii" (details\_btn\_Click), se caută comanda selectată în lista de comenzi și se obține un obiect foundOrder care conține informațiile detaliate despre comandă. Apoi, se deschide fereastra DetaliiComanda și se furnizează informațiile despre comandă acesteia.



7. La apăsarea butonului "Deconectare" (logout\_btn\_Click), se deschide fereastra Login și se ascunde fereastra curentă (Acasa).



**Pagina de ‘Detalii Comanda’**

Codul include operații precum afișarea detaliilor unei comenzi, salvarea modificărilor, ștergerea comenzii și generarea unei facturi în format PDF.

Aplicația se bazează pe Google Cloud Firestore pentru stocarea și gestionarea datelor. Folosește și bibliotecile iTextSharp pentru generarea de documente PDF.

1. În secțiunea de început a codului sunt importate mai multe module și biblioteci necesare pentru funcționarea aplicației.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

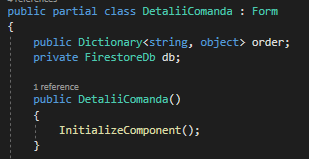
using Google.Cloud.Firestore;

using iTextSharp.text;

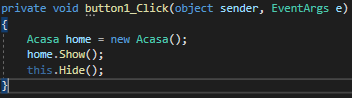
using iTextSharp.text.pdf;

2. Urmează declarația și definirea clasei `DetaliiComanda`, care extinde clasa `Form`. Această clasă reprezintă fereastra aplicației și conține toate componentele vizuale și logica necesară pentru gestionarea comenzilor de service auto.

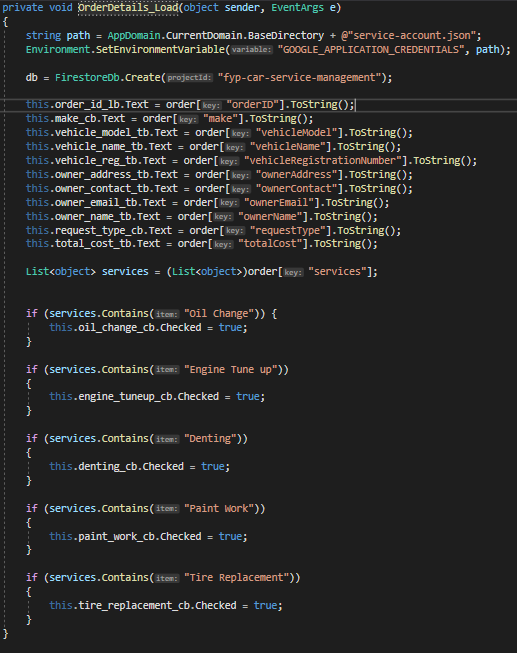
3. În constructorul clasei `DetaliiComanda`, se inițializează componentele ferestrei aplicației prin apelul metodei `InitializeComponent()`.



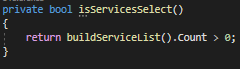
4. Metoda `button1\_Click` este un handler pentru evenimentul "Click" al butonului "Acasa". Acesta deschide fereastra "Acasa" și ascunde fereastra curentă.



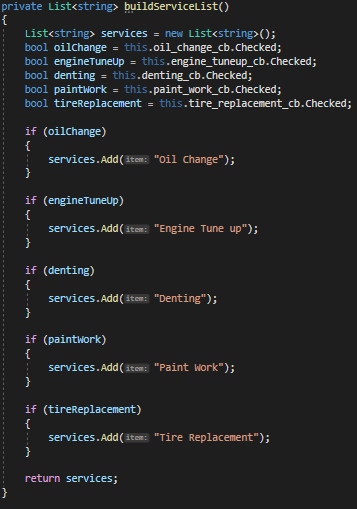
5. Metoda `OrderDetails\_Load` este un handler pentru evenimentul "Load" al ferestrei. Aici se realizează inițializarea și afișarea detaliilor comenzii curente. Detaliile comenzii sunt extrase din dicționarul `order` și sunt afișate în controalele corespunzătoare ale ferestrei.



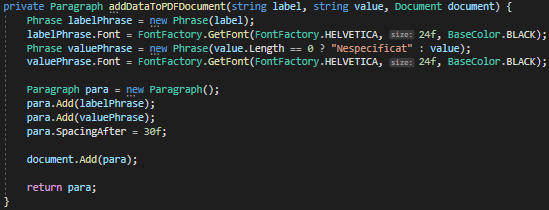
6. Metoda `isServicesSelect` verifică dacă cel puțin un serviciu a fost selectat. Aceasta verifică starea casetelor de selectare corespunzătoare serviciilor și returnează `true` dacă cel puțin unul dintre acestea este selectat.



7. Metoda `buildServiceList` construiește și returnează o listă de servicii selectate. Aceasta verifică starea casetelor de selectare corespunzătoare serviciilor și adaugă serviciile selectate în lista rezultată.



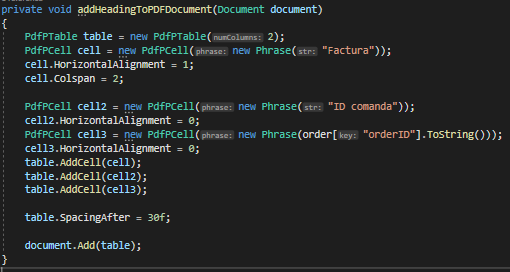
8. Metoda `addDataToPDFDocument` adaugă o etichetă și o valoare într-un document PDF. Aceasta primește o etichetă și o valoare, creează elemente de tip `Phrase` pentru acestea și le adaugă într-un paragraf. Paragraful este apoi adăugat în documentul PDF.



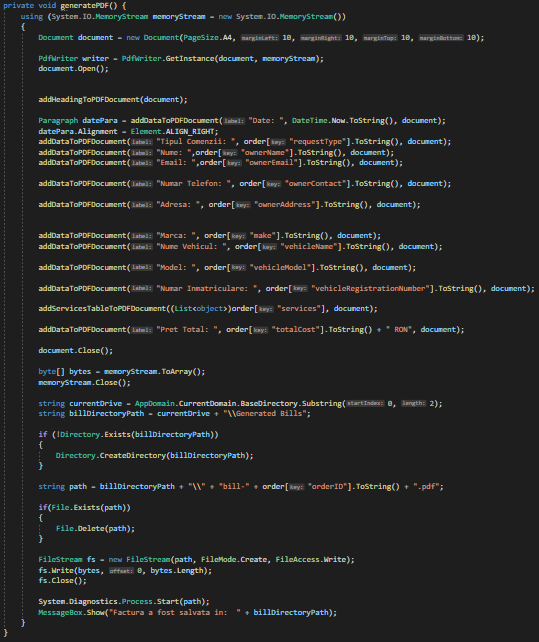
9. Metoda `addServicesTableToPDFDocument` adaugă o tabelă cu serviciile utilizate într-un document PDF. Aceasta primește o listă de servicii, creează un obiect `PdfPTable` și adaugă celule corespunzătoare serviciilor și stării acestora în tabel. Tabelul este apoi adăugat în documentul PDF.



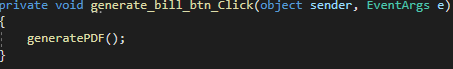
10. Metoda `addHeadingToPDFDocument` adaugă un antet într-un document PDF. Aceasta primește un obiect `Document`, creează un obiect `PdfPTable` și adaugă o celulă corespunzăto are antetului în tabel. Tabelul este apoi adăugat în documentul PDF.



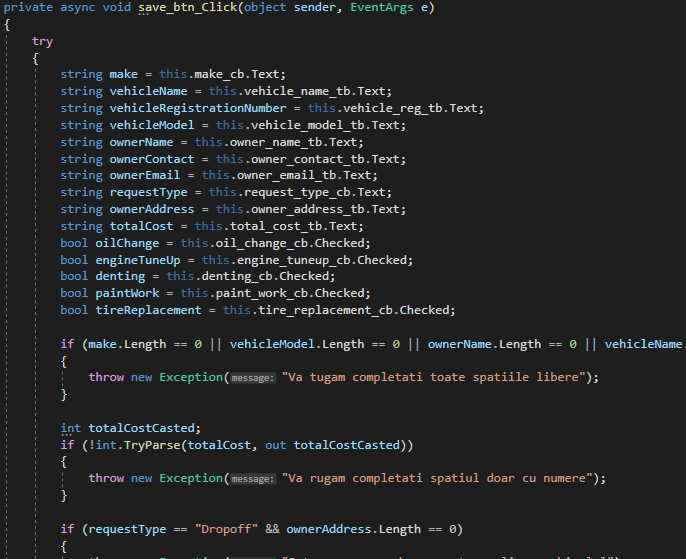
11. Metoda `generateInvoicePDF` generează o factură în format PDF. Aceasta primește un nume de fișier și o listă de servicii utilizate. În interiorul metodei, se creează un document PDF utilizând obiectul `Document` și se adaugă în el antetul, detaliile comenzii și tabelul cu servicii. La final, documentul este închis și salvat cu numele specificat.



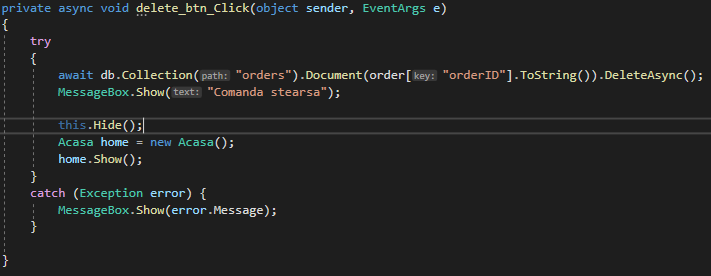
12. Metoda `generate\_bill\_btn\_Click` este un handler pentru evenimentul "Click" al butonului "Generare factură". Aceasta verifică dacă cel puțin un serviciu a fost selectat, iar apoi generează o factură în format PDF utilizând metoda `generateInvoicePDF`.



13. Metoda `save\_btn\_Click` este un handler pentru evenimentul "Click" al butonului "Salvare". Aceasta salvează detaliile comenzii în baza de date Firestore.



14. Metoda `delete\_btn\_Click` este un handler pentru evenimentul "Click" al butonului "Ștergere". Aceasta șterge detaliile comenzii din baza de date Firestore.



**Formularul de ‘Adauga comanda’**

Codul reprezintă o clasă numită "AdaugaComanda" care extinde clasa "Form" din Windows Forms. Această clasă reprezintă fereastra de adăugare a unei noi comenzi în aplicația de gestionare a serviciilor auto.

1. În secțiunea de început a codului sunt importate modulele și bibliotecile necesare pentru funcționarea aplicației.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

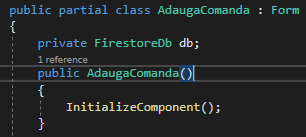
using System.Text;

using System.Threading.Tasks;

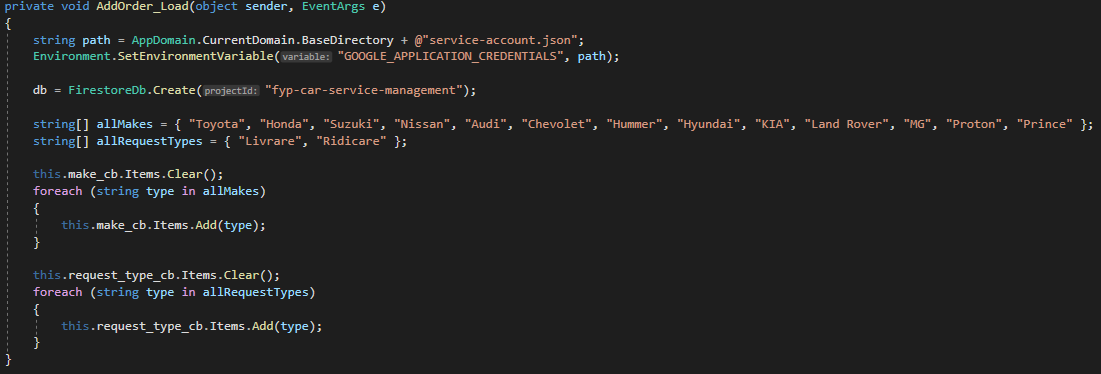
using System.Windows.Forms;

using Google.Cloud.Firestore;

2. În constructorul clasei "AdaugaComanda", se inițializează componentele ferestrei aplicației prin apelul metodei "InitializeComponent()".



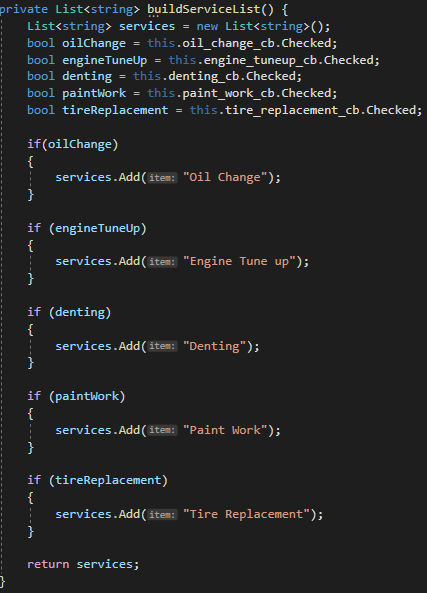
3. Metoda "AddOrder\_Load" este un handler pentru evenimentul "Load" al ferestrei. Aici se realizează setarea inițială a combobox-urilor "make\_cb" și "request\_type\_cb" cu valorile disponibile pentru selecție.



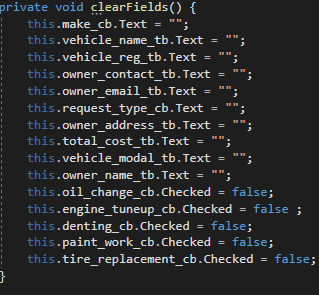
4. Metoda "isServicesSelect" verifică dacă cel puțin un serviciu a fost selectat. Aceasta verifică starea casetelor de selectare corespunzătoare serviciilor și returnează "true" dacă cel puțin unul dintre acestea este selectat.



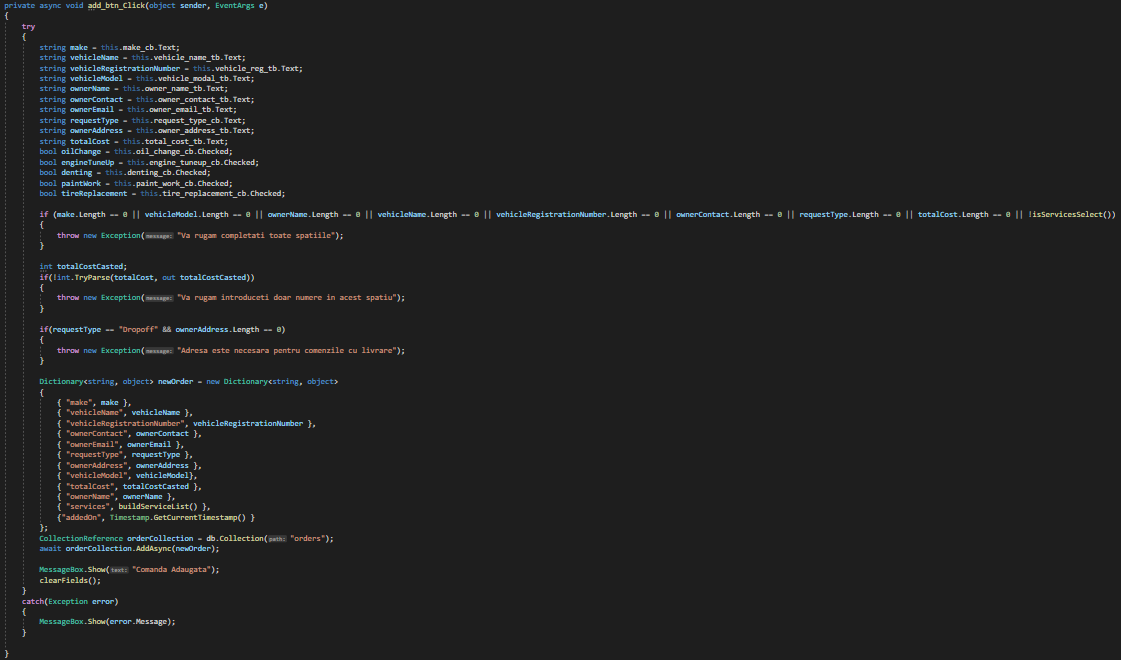
5. Metoda "buildServiceList" construiește și returnează o listă de servicii selectate. Aceasta verifică starea casetelor de selectare corespunzătoare serviciilor și adaugă serviciile selectate în lista rezultată.



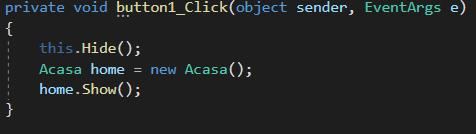
6. Metoda "clearFields" curăță valorile introduse în câmpurile de introducere ale ferestrei.



7. Metoda "add\_btn\_Click" este un handler pentru evenimentul "Click" al butonului "Adauga". Aici se preiau valorile introduse în câmpurile ferestrei, se validează și se adaugă o nouă comandă în baza de date Firestore utilizând obiectul "FirestoreDb" și metoda "AddAsync". În caz de eroare, se afișează un mesaj corespunzător.



8. Metoda "button1\_Click" este un handler pentru evenimentul "Click" al butonului "Acasa". Acesta deschide fereastra "Acasa" și ascunde fereastra curentă.



**Biblioteci speciale folosite**

1. System.Windows.Forms:

Această bibliotecă face parte din .NET Framework și oferă funcționalități pentru crearea interfețelor grafice utilizator (GUI) în aplicațiile Windows. Ea conține clase și metode pentru a crea și manipula elemente precum ferestre, butoane, casete de text, imagini etc.

2. Google.Cloud.Firestore:

Acesta este un pachet de biblioteci oferit de Google pentru a interacționa cu serviciul lor de bază de date Firestore. Firestore este o bază de date NoSQL în timp real, oferind funcționalități de stocare și sincronizare a datelor în cloud. Biblioteca Google.Cloud.Firestore furnizează metode și clase pentru a efectua operațiuni de citire, scriere și interogare în baza de date Firestore.

3. iTextSharp.text:

Aceasta este o bibliotecă open-source pentru manipularea și generarea documentelor PDF în C#. Ea oferă clase și metode pentru a crea, edita și formata documente PDF, inclusiv adăugarea de text, imagini, tabele, formate speciale și multe altele.

4. iTextSharp.text.pdf:

Acesta este un subpachet al bibliotecii iTextSharp care se concentrează pe funcționalitățile specifice PDF. El conține clase și metode pentru manipularea directă a conținutului PDF, precum adăugarea de pagini, adăugarea de conținut grafic, manipularea textului și a formelor etc.