FACULTATEA DE AUTOMATICĂ SI CALCULATOARE

DISCIPLINA INGINERIA PROGRAMĂRII

Specificațiile Cerințelor Software

APLICAȚIE DE GESTIUNE A BAZELOR DE DATE ÎN DIFERITE FORMATE

versiunea 1.0

Mai, 2022

Grupa 1309A

Studenți:

Bulboacă Monica-Andreea

Hociung Vlad-Petru

Mahu Petrișor

Tălmăcel Larisa-Maria

Profesor coordonator,

Tiberius Dumitriu

Cuprins

1. Introducere

1.1 Intenția documentului

Documentul este dedicat specificării cerințelor software (**SRS**) , unde vom oferi o descriere a funcțiilor produsului software „Aplicație de gestiune a bazelor de date în diferite formate”. In următoarele secțiuni se specifica toate cerințele non-funcționale ale produsului, pe care acesta ar trebui sa le respecte, precum: performanță, disponibilitate, securitate.

1.2 Scopul aplicației

Aplicația are scopul de a ușura și mări eficiența cu care o persoana necalificată poate modifica datele dintr-o bază de date provenită din diferite tipuri de fișiere. Astfel, productivitatea va fi sporită deoarece o persoana necalificata în domeniul programării, fără a fi nevoită sa cunoască date din spatele tipului de document și a bazei de date, poate face modificări asupra ei.

1.3 Definiții, acronime si abrevieri

Termen	Semnificație
Bază de date	Fișier ce conține o multitudine de date care sunt asemănătoare printr-o anumită structura
Utilizator	Persoana ce folosește aplicația
GUI	Interfața grafica cu utilizatorul
Aplicație	Produsul software din acest document
Offline	Fără o conexiune la Internet

1.4 Convențiile documentului

- Informațiile din acest document vor fi scrise cu fontul Times New Roman, dimensiune 12px.
- Titlurile vor fi scrise cu fontul Times New Roman, dimensiune de 16px .
- Subtitlurile vor fi scrise cu fontul Times New Roman, dimensiune de 14px.

1.5 Audiența țintă și sugestii de citire a documentului

Acest SRS are ca audiență țintă membrii echipei. Totodată, documentul poate fi folosit și de profesorii coordonatori pentru specificații detaliate legate de aplicație. Nu se recomandă utilizatorilor finali întrucât oferă specificații elaborate legate de modul de implementare al aplicației.

Documentul începe cu o introducere, urmată de o secțiune în care se va prezenta în ansamblu produsul. Ultima secțiune este destinată cerințelor non-functionale ale produsului.

1.6 Referințe

Implementarea aplicației este făcută folosind informații din cursurile si laboratoarele domnului profesor Florin Leon cât și din documentațiile disponibile online:

- http://florinleon.byethost24.com/lab_ip.html
- http://florinleon.byethost24.com/curs_ip.html
- IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society, 1998.

2. Descriere generală a aplicației

2.1 Perspectiva aplicației

Aplicația noastră este construită pentru pentru a ușura îndeplinirea obiectivului utilizatorului de a modifica anumite fișiere tocmai prin intermediul interfeței. Interfața grafica foarte ușor de folosit este perfectă pentru orice tip de utilizator, iar opțiunea de a salva într-un format diferit datele fata de cel inițial poate ajuta în diverse procese care presupun lucrul cu date de tip .XML, .CSV, .JSON.

2.2 Funcțiile aplicației

Pentru început, aplicația permite încărcarea unui fișier ales de către utilizator din spațiul de stocare al calculatorului, fișier al cărui date vor fi preluate, urmand apoi ca utilizatorul sa primească o vizualizare mult mai simplistă a acelor date.

După ce utilizatorul primește vizualizarea datelor, acesta va putea apoi sa faca diverse procesari pe aceste date, de tipul CRUD[1].

Astfel, putem prezenta fiecare functionalitate în parte după cum urmează:

- **Încărcare**
    - Utilizatorul poate încărca un fișier pe care acesta dorește să-l prelucreze
    - Aceasta functionalitate poate fi selectata atât din bara de meniu, categoria File, opțiunea Open, cât și prin butonul din partea dreapta a aplicației, „Load”
- **Salvare**
    - Utilizatorului îi este prezentată o fereastra unde acesta dorește sa salveze fișierul, prin intermediul butonului „Save”
    - Aici, in fereastra gestionata de sistemul de operare, utilizatorul poate alege formatul fișierului și numele acestuia
    - Salvarea fișierului se poate face și prin selectarea formatului în mod predefinit. Formatul predefinit poate fi editat prin intermediul butonului “Edit” -> “Preferences”, urmand sa fie apăsat butonul de “Save” din meniul File

Pentru operațiile de modificare asupra datelor din fișiere, avem doua functionalitati:

- **Modificare**
    - Utilizatorul poate selecta oricare înregistrare și prin intermediul butonului „Edit”, acesta poate modifica datele din înregistrarea selectata
- **Ștergere**
    - Utilizatorul poate selecta oricare înregistrare și prin intermediul butonului „Delete”, acesta poate șterge înregistrarea selectata

De asemenea, pentru utilizatorii care intampina probleme exista în meniu butonul „Help”, ce va deschide fișierul cu indicații referitoare la modul de utilizare al aplicației.

[1] Create, Read, Update, Delete – Crearea, Vizualizare, Actualizare, Ștergere

2.3 Caracteristicile utilizatorului

Produsul este conceput pentru utilizare individuala, consumatorii avand aceleași privilegii si functionalitati disponibile. Aplicația cere cunoștințe minime legate de funcționarea calculatorului intrucat utilizatorul este indrumat in orice acțiune prin intermediul interfeței. Astfel, aplicatia noastra este conceputa pentru persoanele ce au nevoie ca anumite date sa fie centralizate într-un anumit loc și într-un format specificat, oferind portabilitate sporită si accesare foarte ușor din orice loc.

2.4 Constrângeri generale

Aplicația a fost concepută astfel încât sa nu fie nevoie de o pregătire anterioară pentru folosirea programului, singurele cerințe din partea utilizatorului fiind cunoștințe minimale in operarea calculatorului. Pentru utilizare nu sunt necesare alte programe neobișnuite instalate anterior.

2.5 Ipoteze si dependențe

Ipoteza: O companie in urma unor procesări pe un anumit set de date are nevoie ca cineva sa gestioneze acele date prin intervenția umană, astfel pentru orice fel de cerere pentru a modifica datele respective, obținute într-un anumit format, se poate folosi aplicația prezentată pentru a ușura procesarea acestora, fara a fi nevoie de alte cunoștințe tehnice.

Dependențe: Aceasta aplicatie poate fi folosit de un sistem de calcul ce rulează sistemul de operare Microsoft Windows, versiunea minima suportată Windows 10 și are instalat pachetul .NET 4.8 Runtime.

3. Cerințe specifice

3.1 Cerințe specifice interfeței externe

3.1.1 Interfața cu utilizatorul

Odată cu deschiderea aplicației, utilizatorului îi va fi prezentată o interfața, unde prin intermediul butoanelor, acesta va putea alege fișierul pe care dorește să-l vizualizeze.

Figură 1 Interfața fără fișier încărcat

Astfel, după ce utilizatorul alege fișierul ce va fi încărcat, interfața ar trebui să îi prezinte datele din fișierul ales, avand primul rand cu denumirile câmpurilor iar următoarele rânduri reprezentand înregistrările din fișier.

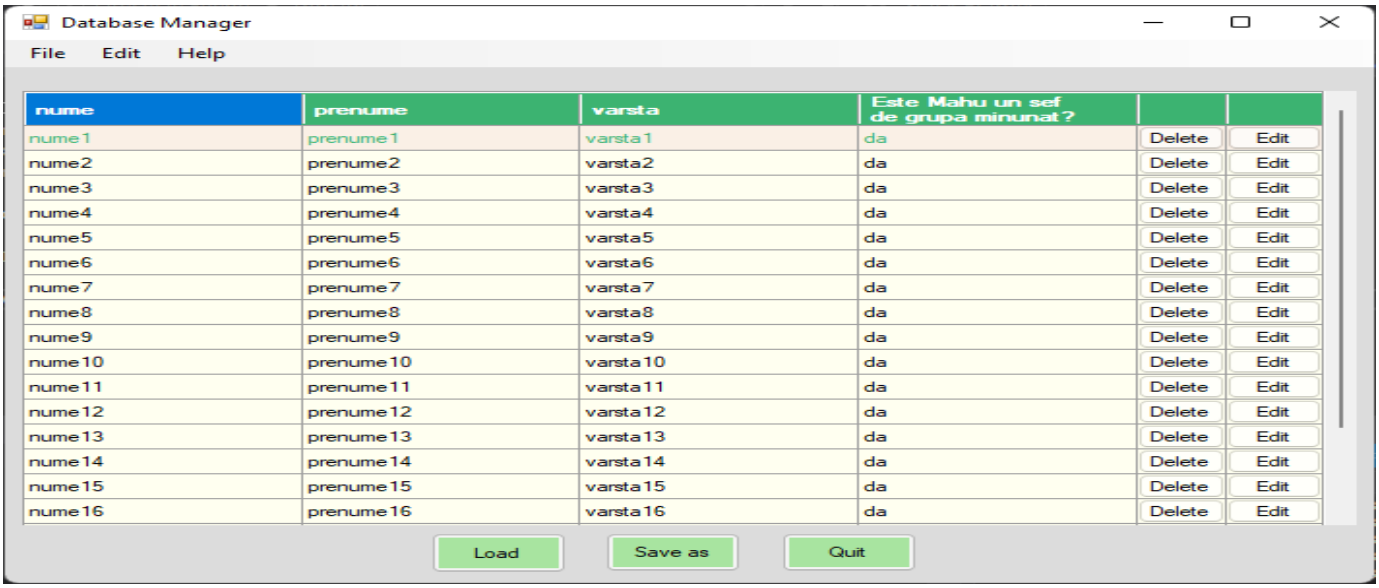


Figura 2 Interfața cu fișierul încărcat

3.1.2 Interfața hardware

Pentru ca aplicația să funcționeze corespunzător, este necesar un sistem de calcul ce poate rula sistemul de operare Windows 10 sau ce îl poate simula. De asemenea, este nevoie de o tastatura si un mouse, pentru a selecta si face modificări asupra inregistrarilor din fișierul selectat cât si pentru a selecta fișierul ce va fi încărcat/salvat.

3.1.3 Interfața software

Pentru ca aplicația să funcționeze corespunzător, sistemul de calcul trebuie sa ruleze Windows 10 și să aibă instalat pachetul .NET 4.8 Runtime. Acest pachet poate fi descarcat oficial de pe site-ul celor de la Microsoft.

Pentru ca funcția de ajutor sa functioneze, este necesar ca aplicatia sa se afle intr-o locatie de nu contine caracterul ‚#'.

3.1.4 Interfețe de comunicare

Aceasta aplicația nu depinde de niciun fel de conexiune cu mediul înconjurător, aplicația putând fi rulata offline.

3.2 Cerințe funcționale

3.2.1 Funcționalitatea de încărcare

Aceasta funcționalitate se refera la fișierul pe care utilizatorul îl alege pentru a-l încărca in aplicație. Pentru ca fisierul sa fie valid, acesta trebuie sa respecte standardele formatului respectiv si să aiba o structura asemanatoare pentru o baza de date, insemnand sa fie compus din mai multe randuri cu o structura repetitiva.

3.2.2 Funcționalitatea de salvare

Aceasta functionalitate este disponibilă abia după ce utilizatorul a încărcat un fisier valid, urmând sa fie întrebat la ce locație dorește sa se faca salvarea fișierului, indiferent dacă acesta a făcut modificări sau nu asupra datelor din fișiere.

3.2.3 Funcționalitatea de modificare

Pentru ca aceasta sa fie disponibilă, utilizatorul trebuie sa incarce un fisier valid, iar după ce acest fișier a fost încărcat, modificările se pot obține în urma folosirii celor doua butoane puse la dispoziție prin intermediul interfeței grafice.

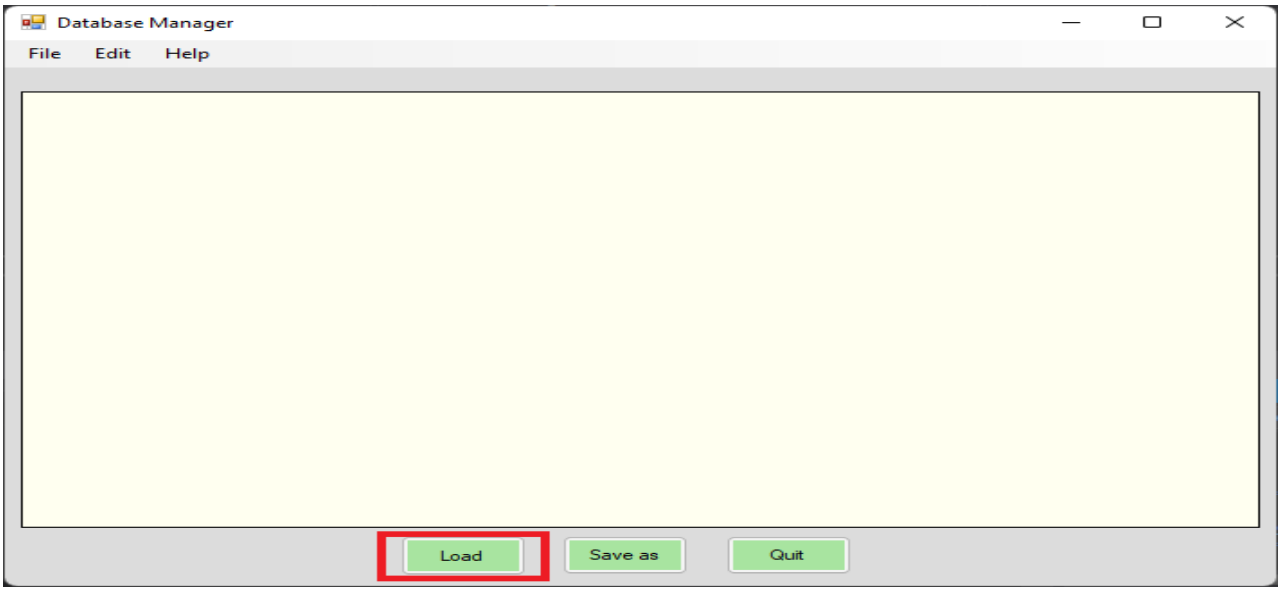
3.2.4 Functionalitatea de ajutor

Pentru ca acesta sa fie disponibilă, utilizatorul trebuie sa ruleze aplicația dintr-o locație a carei cale nu conține caracterul ‚#' iar acesta sa apese pe butonul „Help" din bara de meniu.

4 Modul de utilizare a programului

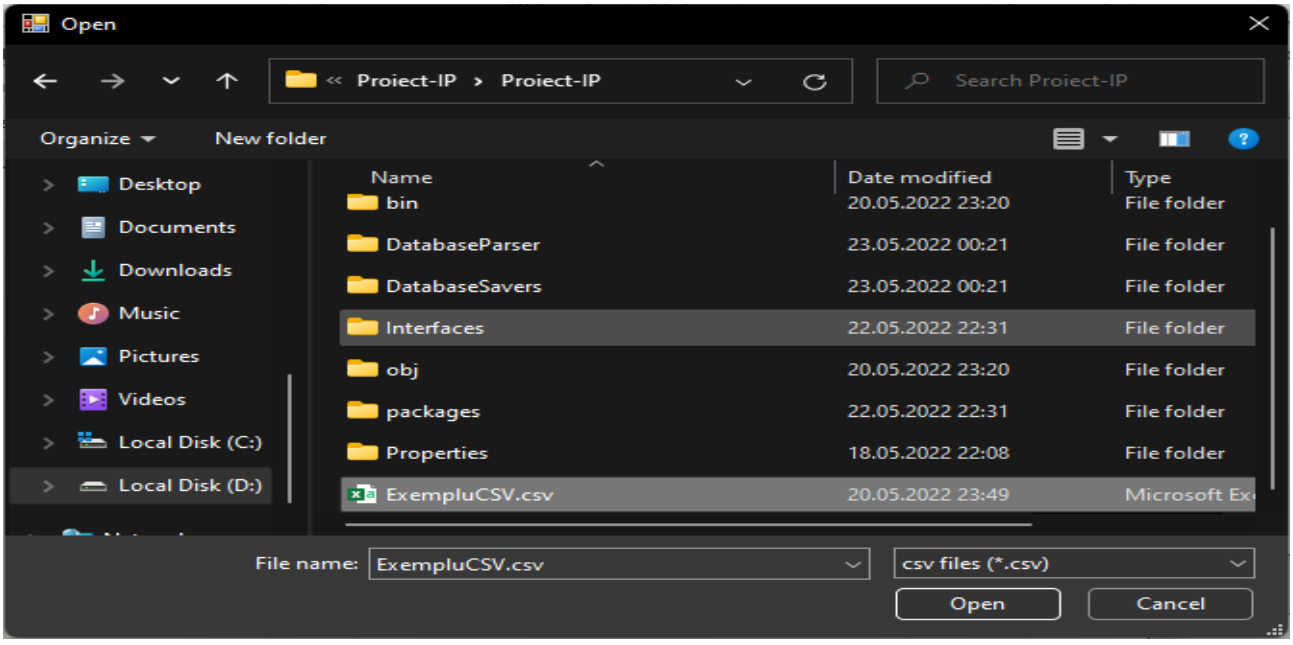
4.1 Încărcarea unui fișier

Pentru a incarca un fișier, se apăsa butonul „Load" după conform indicatiilor din figura.



Figură 3 Adăugare fișier in aplicație

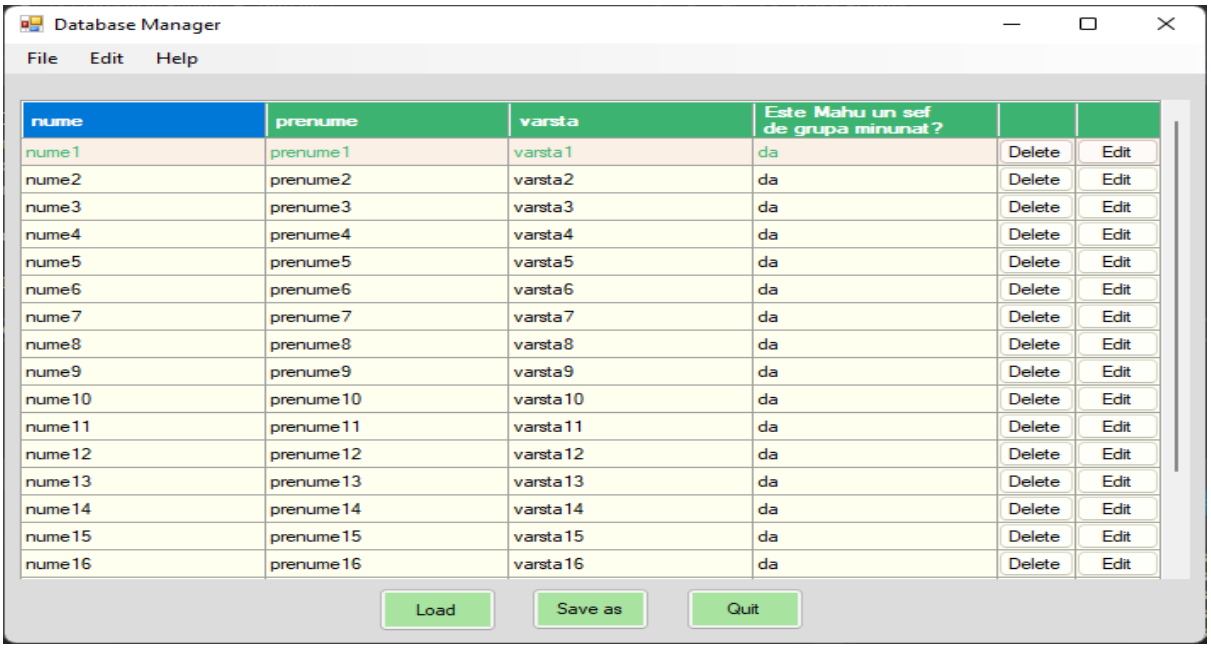
Ulterior, se navighează către calea fișierului ce urmează a fi încărcat.



Figură 4 Navigare către fișier

4.2 Vizualizarea datelor

Pentru a vizualiza datele dintr-un fișier, se încarcă un fișier conform pasului precedent iar în interfata vor fi afișate datele, conform Fig. 2

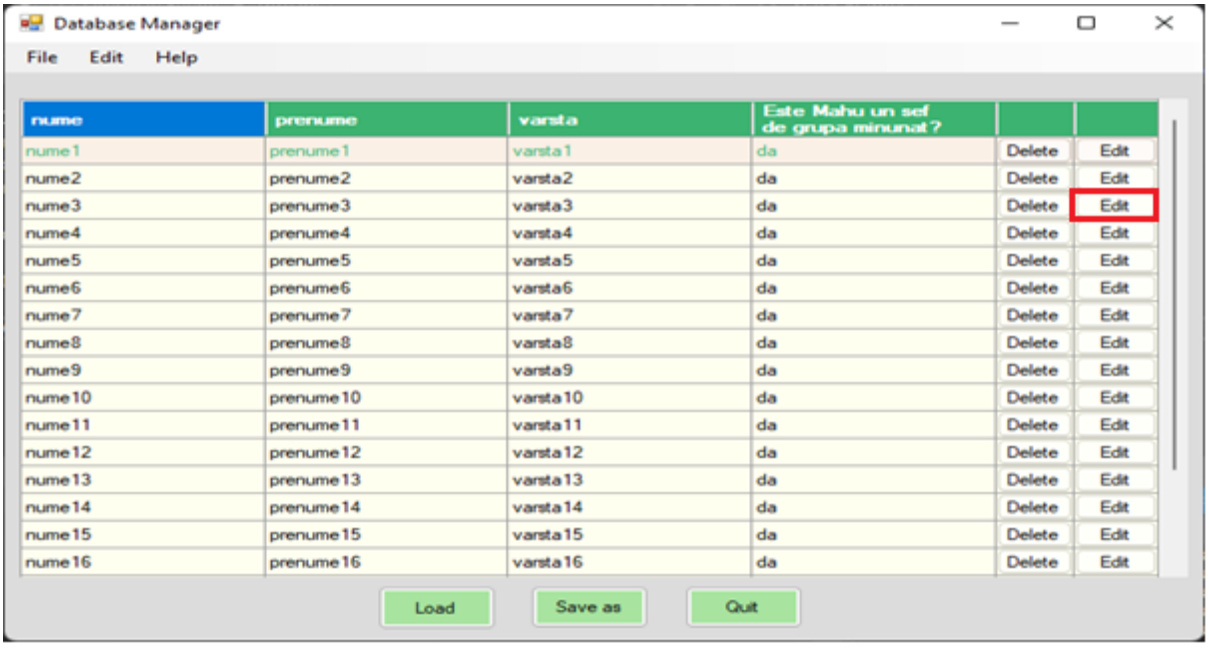


Figură 5 Interfața cu fișierul încărcat

4.3 Modificarea datelor

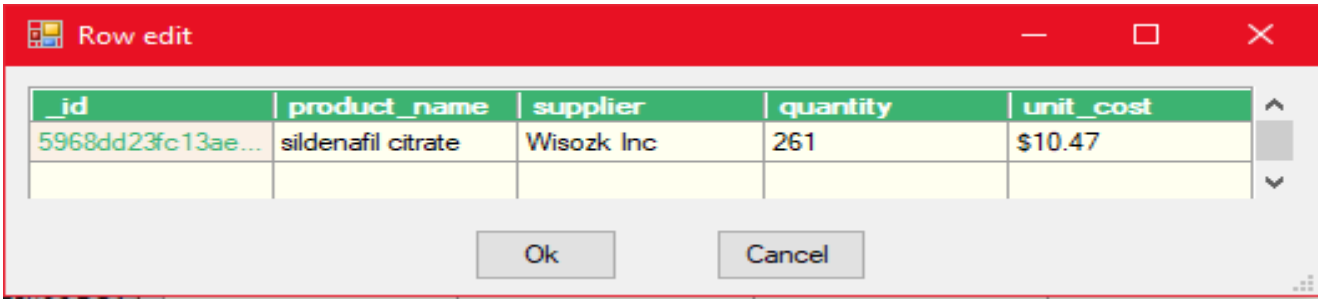
4.3.1 Editarea unei înregistrări

Pentru a putea edita o înregistrare, se urmează pașii precedenti pentru a vizualiza datele, urmand apoi a se apăsa butonul „Edit”, după cum se poate observa în figura următoare:



Figură 6 Modificarea inregistrarilor – Editare

După ce butonul a fost apăsat, se va deschide o fereastra unde înregistrarea va putea fi modificata:

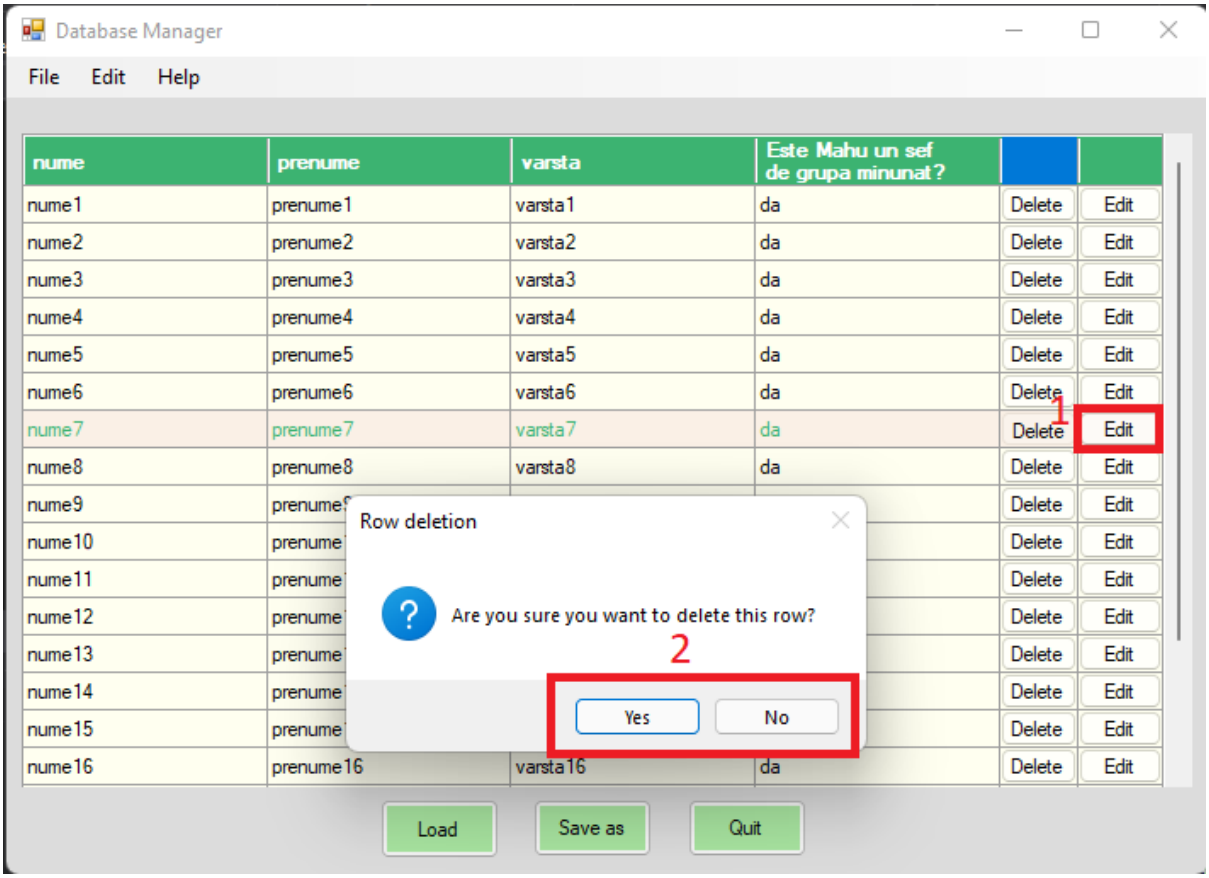


Figură 7 Modificarea inregistrarilor – Editarea

Aici, utilizatorul poate modifica după bunul plac fiecare camp din înregistrarea respectivă. În cazul în care acesta este mulțumit cu modificările aduse, poate apăsa butonul „Ok” iar acestea vor fi salvate, iar în caz contrar, poate apăsa butonul „Cancel”, pentru a anula modificările.

4.3.2 Ștergerea unei înregistrări

Pentru a șterge o înregistrare din fișier, utilizatorul trebuie doar sa apese pe butonul „Delete” înregistrării corespunzătoare. După ce apăsa pe butonul respectiv, acesta este intampinat de un mesaj de confirmare pentru a șterge înregistrarea respectivă, după cum poate fi observat în figura următoare:

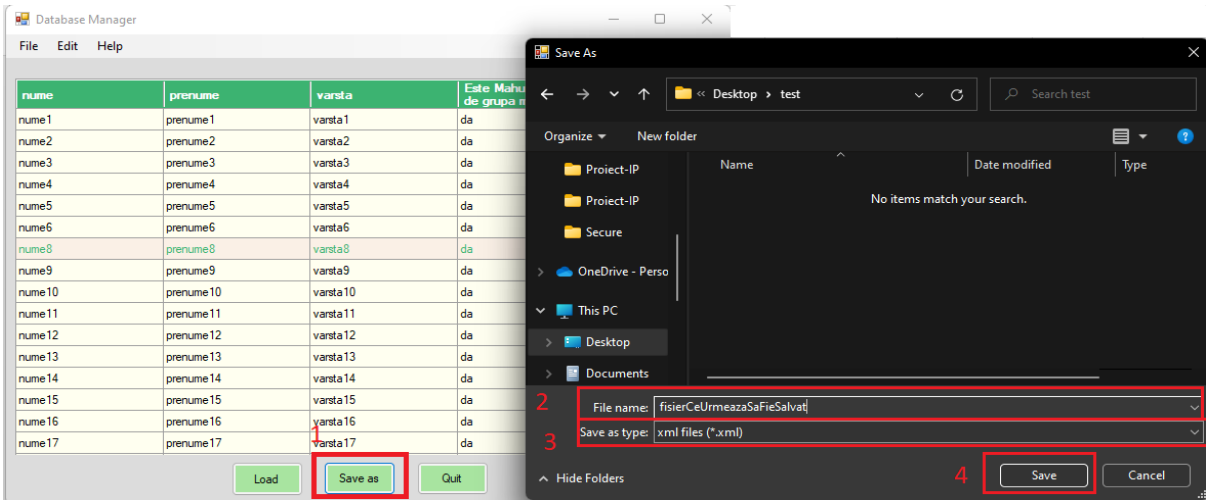


Figură 8 Modificare inregistrarilor – Stergerea

4.4 Salvarea fișierului

După ce utilizatorul a terminat de adus modificări în fișierul respectiv, acesta poate apăsa pe butonul „Save as” pentru a alege sa salveze fișierul respectiv cu posibilele modificări aduse.

După plasarea acestuia, o fereastra apare cu locația unde dorește fișierul sa fie salvat si cu formatul pe care acesta dorește sa il foloseasca.

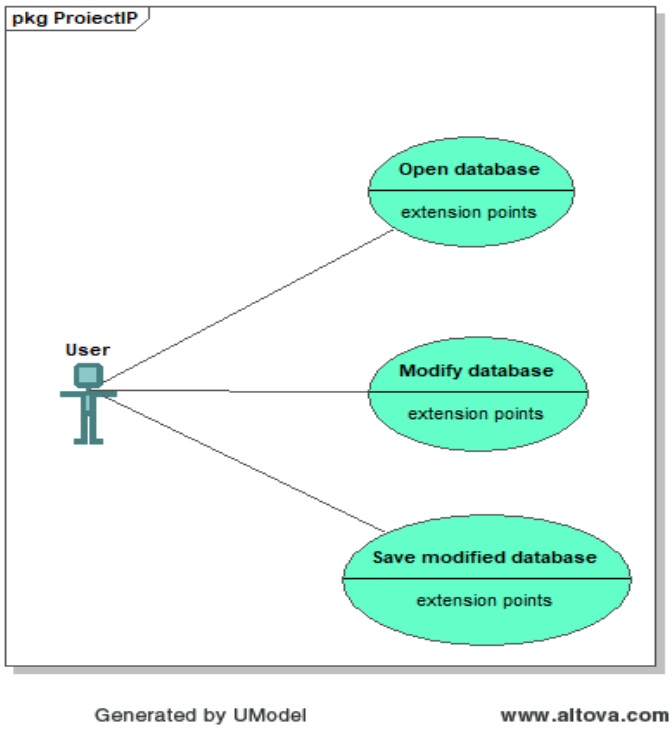


Figură 9 Salvarea fișierului

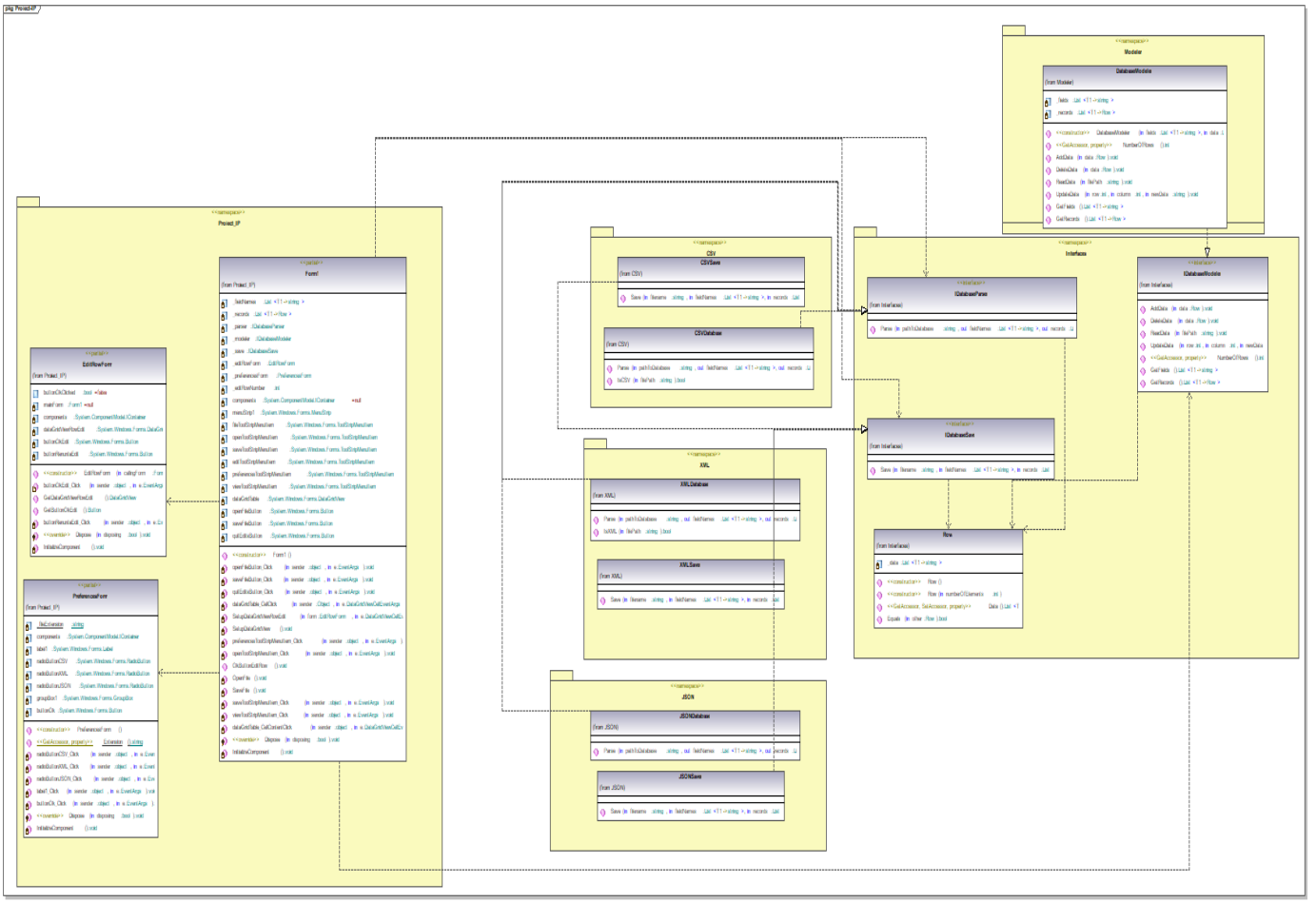
5. Anexe

5.1 Diagrame UML

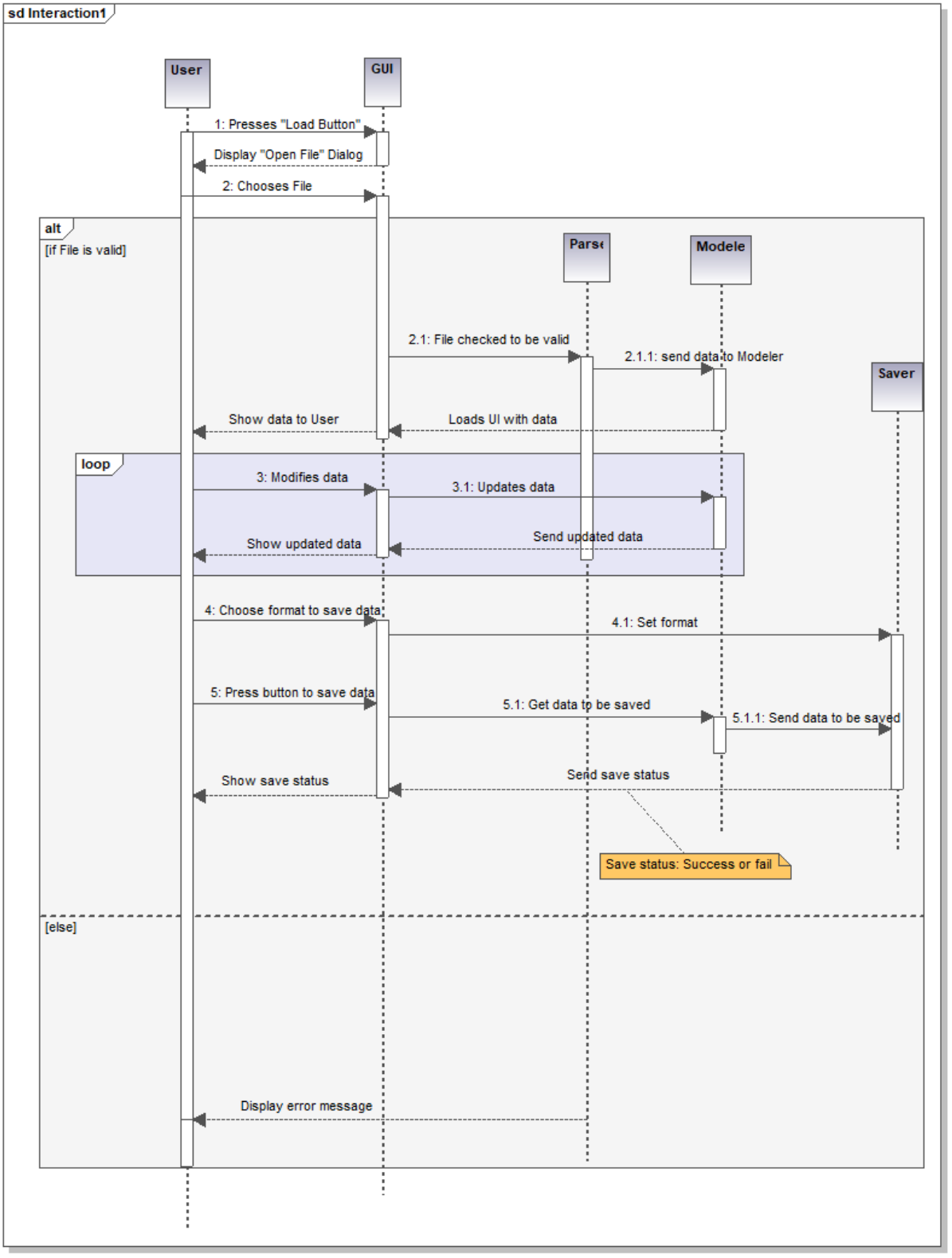
5.1.1 Diagrama de use-case



5.1.2 Diagrama de clase

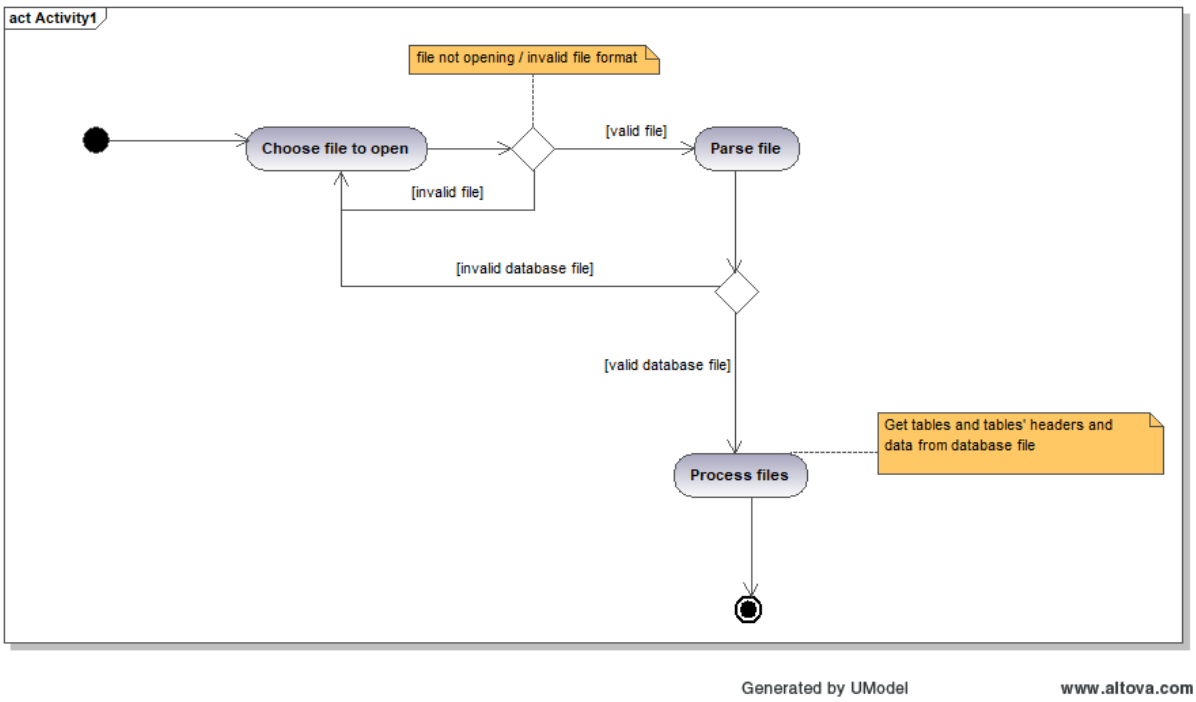


5.1.3 Diagrama de secvență



Generated by UModel www.altova.com

5.1.4 Diagrama de activități



5.2 Părți semnificative ale codului sursă:

5.2.1 Metoda de încărcare a unui fișier în interfață:

```
        /// <summary>
        /// Method that loads a file and parses its content.
        /// </summary>
        private void OpenFile()
        {
            string filePath = string.Empty;
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = "c:\\";
            openFileDialog.Filter = "csv files (*.csv)|*.csv|xml files (*.xml)|*.xml|json
files (*.json)|*.json";
            openFileDialog.FilterIndex = 1;
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                filePath = openFileDialog.FileName;

                // check extension
                string fileExtension = filePath.Split('.')[1];
                switch (fileExtension)
                {
                    case "csv":
```

```
                    _parser = new CSV.CSVDatabase();
                    try
                    {
                        _parser.Parse(filePath, out _fieldNames, out _records);
                        _modeler = new Modeler.DatabaseModeler(_fieldNames, _records);
                        SetupDataGridView();
                    }
                    catch (Exception ex)
                    {
                        string title = "File error!";
                        MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
                        DialogResult result = MessageBox.Show(ex.Message, title, buttons,
MessageBoxIcon.Exclamation);
                    }
                    break;
                case "xml":
                    _parser = new XML.XMLDatabase();
                    try
                    {
                        _parser.Parse(filePath, out _fieldNames, out _records);
                        _modeler = new Modeler.DatabaseModeler(_fieldNames, _records);
                        SetupDataGridView();
                    }
                    catch (Exception ex)
                    {
                        string title = "File error";
                        MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
                        DialogResult result = MessageBox.Show(ex.Message, title, buttons,
MessageBoxIcon.Exclamation);
                    }
                    break;
                case "json":
                    _parser = new JSON.JSONDatabase();
                    try
                    {
                        _parser.Parse(filePath, out _fieldNames, out _records);
                        _modeler = new Modeler.DatabaseModeler(_fieldNames, _records);
                        SetupDataGridView();
                    }
                    //invalid json exception
                    catch (Newtonsoft.Json.JsonReaderException ex)
                    {
                        string title = "Invalid file!";
                        MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
                        DialogResult result = MessageBox.Show(ex.Message, title, buttons,
MessageBoxIcon.Exclamation);
                    }
                    //general exception
                    catch (Exception ex)
                    {
```

```
                        string title = "File error!";
                        MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
                        DialogResult result = MessageBox.Show(ex.Message, title, buttons,
MessageBoxIcon.Exclamation);
                    }
                    break;
                }
            }
        }
```

5.2.2 Metoda de salvare a unui fișier cu un anumit format:

```
        /// <summary>
        /// Method that saves a file with certain extension.
        /// </summary>
        private void SaveFile()
        {
            string filePath = string.Empty;
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.InitialDirectory = "c:\\";
            saveFileDialog.Filter = "csv files (*.csv)|*.csv|xml files
(*.xml)|*.xml|json files (*.json)|*.json";
            string defaultExtension = PreferencesForm.Extension;
            saveFileDialog.FilterIndex = defaultExtension == "csv" ? 1 :
(defaultExtension == "xml" ? 2 : (defaultExtension == "json" ? 3 : 1));
            try
            {
                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {
                    filePath = saveFileDialog.FileName;
                    string fileExtension = filePath.Split('.')[1];
                    switch (fileExtension)
                    {
                        case "csv":
                            _save =new CSV.CSVSaver();
                            _save.Save(filePath, _modeler.GetFields(),
_modeler.GetRecords());
                            break;
                        case "json":
                            _save = new JSON.JSONSaver();
                            _save.Save(filePath, _modeler.GetFields(),
_modeler.GetRecords());
                            break;
                        case "xml":
                            _save = new XML.XMLSaver();
                            _save.Save(filePath, _modeler.GetFields(),
_modeler.GetRecords());
                            break;
                        default:
                            throw new Exception("Invalid extension! Please try .csv,
```

```
.xml or .json!");
                    }
                    string title = "Save";
                    MessageBoxButtons buttons = MessageBoxButtons.OK;
                    DialogResult result = MessageBox.Show("File saved successfully!",
title, buttons, MessageBoxIcon.Asterisk);
                }
            }
            catch (Exception ex)
            {
                string title = "Invalid extension!";
                MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
                DialogResult result = MessageBox.Show(ex.Message, title, buttons,
MessageBoxIcon.Exclamation);
            }
        }
```

5.2.3 Metoda de parsare a unui fișier csv:

```
        /// <summary>
        /// Extract field names and records from a csv file
        /// </summary>
        /// <param name="pathToDatabase">Filepath to .csv file</param>
        /// <param name="fieldNames">Output parameter for fieldnames</param>
        /// <param name="records">Output parameter for records</param>
        public void Parse(string pathToDatabase, out List<string> fieldNames, out
List<Row>      records)
        {
            if (IsCSV(pathToDatabase))
            {
                fieldNames = new List<string>();
                records = new List<Row>();
                StreamReader file = new StreamReader(pathToDatabase);
                string firstLine = file.ReadLine();
                string[] headers = firstLine.Split(',');
                foreach (string header in headers)
                {
                    fieldNames.Add(header);
                }

                string line;
                while ((line = file.ReadLine()) != null)
                {
                    Row row = new Row();

                    string[] fields = line.Split(',');
                    foreach (string field in fields)
                    {
                        row.Data.Add(field);
```

```
            }
            records.Add(row);
        }
        file.Close();
    }
    else
    {
        throw new Exception("The csv file is not valid!");
    }
}
```