Aplicație pentru Managementul Costurilor unui Depou Feroviar

DepoHelper

Autor:
Badea Cătălin-Gabriel

Octombrie 2025

Cuprins

1 Introducere

Proiectul de față are ca scop dezvoltarea unei aplicații informatice pentru gestionarea activităților și cheltuielilor unui depou feroviar. În cadrul acestui depou, locomotivelor și vagoanelor li se asigură întreținerea și reparațiile necesare pentru a funcționa în condiții optime. Aplicația urmărește înregistrarea achizițiilor de piese și a serviciilor prestate, asociindu-le corect fiecărei locomotive sau fiecărui vagon, precum și evidența facturilor și a furnizorilor implicați.

Un aspect important al proiectului este gestionarea centrelor de cost, care permit departajarea cheltuielilor pe întreținerea locomotivelor, întreținerea vagoanelor, activități de manevre sau costuri centrale. Prin această structurare, se pot obține rapoarte detaliate privind cheltuielile lunare și anuale pentru fiecare punct de lucru, fiecare locomotivă și fiecare vagon, oferind astfel un instrument util pentru monitorizarea și planificarea resurselor.

În esență, proiectul combină evidența logistică a trenurilor cu analiza financiară, permițând nu doar urmărirea pieselor și serviciilor, ci și evaluarea costurilor într-un mod clar și structurat. Această aplicație poate fi astfel folosită atât pentru administrarea operațională a depoului, cât și pentru luarea deciziilor strategice privind întreținerea și optimizarea resurselor feroviare.

2 Cerințe trasate pe baza discuției cu clientul

Pe baza discuțiilor ipotetice cu clientul, am tras următoarele concluzii referitoare la cerințele și funcționalitățile care trebuie implementate în aplicația pentru gestiunea depoului feroviar:

- Depoul colaborează cu mai multe firme, unele având statutul de furnizor, altele de client.
- În vederea efectuării activității sale, depoul utilizează numeroase tipuri de piese (și consumabile).
- O factură este asociată unui furnizor și poate conține mai multe piese, fiecare cu cantitatea și prețul unitar (care poate varia de la o factură la alta, în funcție de furnizor și situația economică actuală).
- Piesele pot fi alocate unei locomotive specifice sau depozitului, iar o factură poate include piese pentru mai multe locomotive.
- Depoul cuprinde mai multe puncte de lucru, fiecare cu locomotivele sale.
- Pentru fiecare locomotivă se achiziționează piese diferite la momente diferite și la prețuri diferite, prin intermediul facturilor.
- Firmele implicate au o persoană de contact.
- Toate vagoanele au proprietari (firme-client) și pentru acestea se achiziționează piese pentru reparații, în cantități și prețuri variabile, similar cu locomotivele.
- Pentru vagoane nu există servicii prestate.
- Serviciile sunt prestate de firme și pot fi asociate unor locomotive sau, pentru centrul de cost „Central”, să nu fie asociate niciunei locomotive.
- Aplicația trebuie să permită calcularea și urmărirea costurilor pe centre de cost, pentru fiecare locomotivă și punct de lucru, atât lunar, cât și anual.

- Centrele de cost sunt structurate astfel:
  - „Întreținere locomotive” – piese și servicii pentru locomotive
  - „Întreținere vagoane” – doar piese pentru vagoane
  - „Central” – servicii nelegate de întreținere locomotive sau vagoane, ci pentru activitățile firmei
  - „Manevre” – piese (în special consumabile) pentru depozit și anumite servicii fără asociere la locomotivă.

3 Etapa de proiectare

În etapa de proiectare a bazei de date, am pornit de la analiza cerințelor clientului și a activităților depoului feroviar.

Am identificat entitățile principale: firme, locomotive, vagoane, piese și servicii, precum și relațiile dintre acestea.

Am creat tabelele corespunzătoare fiecărei entități și am definit tabelele de legătură pentru relațiile de tip N:N, astfel încât să pot urmări achizițiile de piese și serviciile prestate pentru fiecare locomotivă și vagon.

De asemenea, am stabilit centrele de cost pentru a organiza corect facturile și a putea genera rapoarte lunare și anuale pe puncte de lucru, locomotive și vagoane.

Fiecare tabel a fost proiectat astfel încât să respecte integritatea referențială și să permită validarea datelor, evitând introducerea de informații incomplete sau inconsistente. Prin această structurare, am reușit să realizez un model de date clar, flexibil și adaptat nevoilor clientului, facilitând atât urmărirea logistică, cât și analiza financiară a depoului.

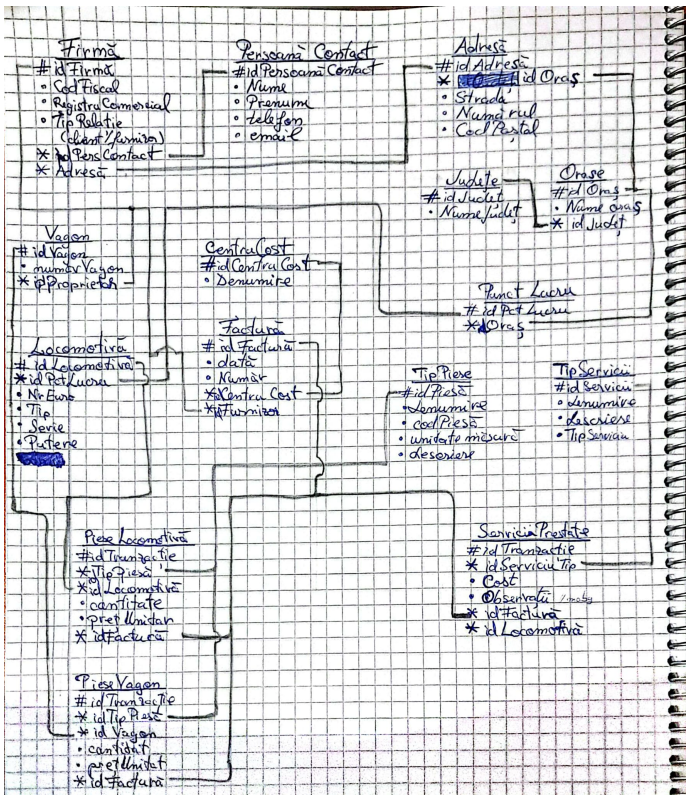

Figura 1: Schema Tabelelo pe Hârtie

4 Descrierea tabelelor și a câmpurilor

4.1 Tabele principale

Firme: conține informații despre firmele implicate (furnizori sau clienți).
PK: idFirma
FK: idPersoanaContact (PersoaneContact:idPersoanaContact, 1:1), idAdresa (Adrese:idAdresa, 1:1)
Alte Atribute: numeFirma, codFiscal, registruComercial, tipRelatie
Atribute Opționale: –

PersoaneContact: stochează datele de contact ale persoanelor asociate firmelor.
PK: idPersoanaContact
FK: –
Alte Atribute: nume, prenume, telefon, email
Atribute Opționale: –

Adrese: stochează adresele firmelor sau punctelor de lucru.
PK: idAdresa
FK: idOras (Orase:idOras, 1:N)
Alte Atribute: strada, numar
Atribute Opționale: codPostal

Orase: stochează orașele în care există puncte de lucru sau firme.
PK: idOras
FK: idJudet (Judete:idJudet, 1:N)
Alte Atribute: numeOras
Atribute Opționale: –

Judete: stochează județele corespunzătoare orașelor.
PK: idJudet
FK: –
Alte Atribute: numeJudet
Atribute Opționale: –

PuncteLucru: reprezintă punctele de lucru ale depoului, fiecare cu locomotivele sale.
PK: idPunctLucru
FK: idOras (Orase:idOras, 1:N)
Alte Atribute: –
Atribute Opționale: –

CentreCost: stochează centrele de cost folosite în aplicație.
PK: idCentruCost
FK: –
Alte Atribute: denumireCentruCost
Atribute Opționale: –

Vagoane: reprezintă vagoanele gestionate de depou, asociate firmelor-clienți.
PK: idVagon
FK: idProprietar (Firme:idFirma, 1:N)
Alte Atribute: numarVagon
Atribute Opționale: –

Locomotive: stochează informațiile despre locomotivele depoului.
PK: idLocomotiva
FK: idPunctLucru (PuncteLucru:idPunctLucru, 1:N)
Alte Atribute: tip, putere
Atribute Opționale: nrEuro, serie

Facturi: reprezintă facturile emise pentru piese și servicii.
PK: idFactura
FK: idCentruCost (CentreCost:idCentruCost, 1:N), idFurnizor (Firme:idFirma, 1:N)
Alte Atribute: dataFactura, numarFactura, sumaTotal
Atribute Opționale: –

Piese: stochează tipurile de piese disponibile în depou.
PK: idPiesa
FK: –
Alte Atribute: denumirePiesa
Atribute Opționale: codPiesa, unitateMasuraPiesa, descrierePiesa

Servicii: stochează tipurile de servicii prestate de firme.
PK: idServiciu
FK: –
Alte Atribute: denumireServiciu
Atribute Opționale: descriereServiciu, tipServicu

4.2 Tabele de legătură N:N

PieseLocomotive: asociază piesele achiziționate cu locomotivele și facturile.
PK: idTranzactiePL
FK: idPiesa (Piese:idPiesa, 1:N), idLocomotiva (Locomotive:idLocomotiva, 1:N), idFactura (Facturi:idFactura, 1:N)
Alte Atribute: cantitate, pretUnitar
Atribute Opționale: –

PieseVagoane: asociază piesele achiziționate cu vagoanele și facturile.
PK: idTranzactie
FK: idPiesa (Piese:idPiesa, 1:N), idVagon (Vagoane:idVagon, 1:N), idFactura (Facturi:idFactura, 1:N)
Alte Atribute: cantitate, pretUnitar
Atribute Opționale: –

ServiciiPrestate: asociază serviciile prestate cu facturile și, opțional, cu locomotivele.
PK: idTranzactieSP
FK: idServiciu (Servicii:idServiciu, 1:N), idFactura (Facturi:idFactura, 1:N), idLocomotiva (Locomotive:idLocomotiva, 0:N)
Alte Atribute: cost
Atribute Opționale: observatii

5 Relații între tabele

Firme – PersoaneContact: fiecare firmă are asociată o singură persoană de contact.
Tip relație: 1:1
Cheie străină: Firme.idPersoanaContact → PersoaneContact.idPersoanaContact

Firme – Adrese: fiecare firmă are o adresă unică.
Tip relație: 1:1
Cheie străină: Firme.idAdresa → Adrese.idAdresa

Adrese – Orase: fiecare adresă aparține unui singur oraș, dar un oraș poate avea mai multe adrese.
Tip relație: N:1
Cheie străină: Adrese.idOras → Orase.idOras

Orase – Judete: fiecare oraș aparține unui județ, dar un județ poate avea mai multe orașe.
Tip relație: N:1
Cheie străină: Orase.idJudet → Judete.idJudet

PuncteLucru – Orase: fiecare punct de lucru este situat într-un singur oraș, dar un oraș poate avea mai multe puncte de lucru.
Tip relație: N:1
Cheie străină: PuncteLucru.idOras → Orase.idOras

Locomotive – PuncteLucru: fiecare locomotivă aparține unui punct de lucru, iar un punct de lucru poate avea mai multe locomotive.
Tip relație: N:1
Cheie străină: Locomotive.idPunctLucru → PuncteLucru.idPunctLucru

Vagoane – Firme (proprietar): fiecare vagon are un proprietar (firmă client), iar o firmă poate deține mai multe vagoane.
Tip relație: N:1
Cheie străină: Vagoane.idProprietar → Firme.idFirma

Facturi – Firme (furnizor): fiecare factură este emisă de un furnizor, iar un furnizor poate emite mai multe facturi.
Tip relație: N:1
Cheie străină: Facturi.idFurnizor → Firme.idFirma

Facturi – CentreCost: fiecare factură este asociată unui centru de cost, iar un centru de cost poate fi folosit de mai multe facturi.
Tip relație: N:1
Cheie străină: Facturi.idCentruCost → CentreCost.idCentruCost

PieseLocomotive – Piese, Locomotive, Facturi: asociază piesele cu locomotivele și facturile corespunzătoare.
Tip relație: N:N
Chei străine:

- PieseLocomotive.idPiesa → Piese.idPiesa
- PieseLocomotive.idLocomotiva → Locomotive.idLocomotiva
- PieseLocomotive.idFactura → Facturi.idFactura

PieseVagoane – Piese, Vagoane, Facturi: asociază piesele cu vagoanele și facturile corespunzătoare.
Tip relație: N:N
Chei străine:

- PieseVagoane.idPiesa → Piese.idPiesa
- PieseVagoane.idVagon → Vagoane.idVagon
- PieseVagoane.idFactura → Facturi.idFactura

ServiciiPrestate – Servicii, Facturi, Locomotive: asociază serviciile prestate cu facturile și, opțional, cu locomotivele.
Tip relație: N:N
Chei străine:

- ServiciiPrestate.idServiciu → Servicii.idServicu
- ServiciiPrestate.idFactura → Facturi.idFactura
- ServiciiPrestate.idLocomotiva → Locomotive.idLocomotiva (0:N, poate fi NULL)

6 Constrângeri de integritate

În proiectarea bazei de date pentru gestiunea depoului feroviar, am impus mai multe constrângeri de integritate pentru a asigura consistența și corectitudinea datelor. Acestea includ:

- **Integritate referențială:** toate cheile externe (FK) sunt obligatoriu legate de o cheie primară existentă în tabelul sursă. De exemplu, idPunctLucru din tabelul *Locomotive* trebuie să existe în *PuncteLucru*, iar idProprietar din *Vagoane* trebuie să existe în *Firme*.
- **Integritate entitate:** fiecare tabel are o cheie primară (PK) care identifică unic fiecare înregistrare. Nici o valoare de PK nu poate fi nulă.
- **Integritate domeniu:** fiecare atribut este definit cu tip de date specific (numeric, text, dată) și dimensiuni maxime, pentru a preveni introducerea de date nevalide. De exemplu, dataFactura trebuie să fie o dată validă, iar câmpul de email trebuie să respecte formatul standard (de forma *utilizator@domeniu.extensie*).
- **Constrângeri specifice aplicației:**
  - Tipul relației firmă este restricționat la 'C' (client) sau 'F' (furnizor).
  - Pentru facturile asociate centrelor de cost, idLocomotiva poate fi NULL doar dacă centrul de cost este „Central" sau „Manevre", altfel este obligatoriu.
  - În cadrul inserării facturilor, ne-am asigurat prin proiectarea aplicației și prin definirea tipurilor de date că obiectele sunt inserate în cantități finite (valori numerice strict pozitive).
  - Nu se permit duplicări de combinații unice în tabelele de legătură N:N, cum ar fi (idFactura, idLocomotiva, idPiesa) în *PieseLocomotive*.
  - Toate locomotivele trebuie să fie asociate unui punct de lucru, toate vagoanele trebuie să aibă un proprietar.
- **Integritate logică:** relațiile dintre entități respectă logica aplicației. De exemplu, serviciile prestate pentru vagoane nu există; un vagon nu poate avea idLocomotiva asociat.

- **Restricții de nullitate:**
  - Atributele obligatorii nu pot fi NULL. De exemplu, fiecare locomotivă trebuie să fie asociată unui punct de lucru (idPunctLucru) și fiecare vagon trebuie să aibă un proprietar (idProprietar). Aceste câmpuri sunt marcate ca NOT NULL.
  - Atributele opționale pot fi NULL, atunci când logica aplicației permite. De exemplu, idLocomotiva din tabelul *ServiciiPrestate* poate fi NULL dacă serviciul nu este asociat unei locomotive (ex: centrul de cost „Central” sau „Manevre”).

Pentru versiuni ulterioare ale aplicației, setul de constrângeri poate fi extins considerabil pentru a acoperi cazuri mai specifice. De exemplu, numărul de telefon ar putea fi validat strict pentru a conține doar cifre, având o lungime fixă (de 10 cifre sau 12 cifre cu prefix de țară), iar codul fiscal ar putea trece prin algoritmi de validare mai complecși. Deși aceste validări sunt utile în industrie, implementarea lor la acest nivel de detaliu depășește necesitățile didactice ale unui proiect de facultate, focusul fiind pe structura și logica bazei de date.

Aceste constrângeri asigură că baza de date va rămâne coerentă, evitând introducerea de date invalide sau inconsistente și facilitând generarea corectă a rapoartelor financiare și operaționale pentru depou.

7 Diagramă relații între tabele

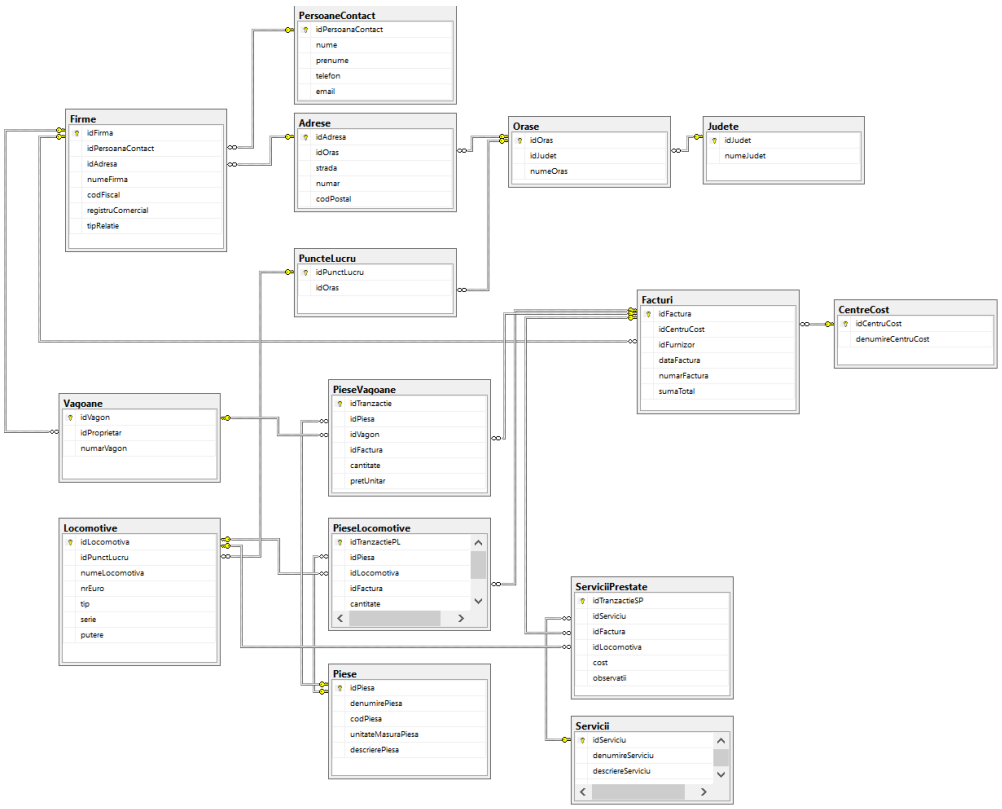

Figura 2: Diagramă Relații între Tabele

8 Implementarea aplicației

8.1 Tehnologii utilizate

Aplicația a fost dezvoltată ca o aplicație desktop pentru sistemul de operare Windows, utilizând tehnologia **Windows Forms** din cadrul platformei **.NET Framework**. Mediul de dezvoltare ales a fost **Microsoft Visual Studio 2022**, datorită suportului extins pentru aplicații .NET și integrării eficiente cu bazele de date relaționale.

Pentru stocarea și gestionarea datelor a fost utilizat **Microsoft SQL Server**, o bază de date relațională robustă și scalabilă. Comunicarea dintre aplicație și baza de date se realizează prin intermediul tehnologiei **ADO.NET**, folosind clase precum `SqlConnection`, `SqlCommand` și `SqlDataReader`.

Această combinație de tehnologii permite realizarea unei aplicații stabile, eficiente și ușor de întreținut, potrivită pentru gestiunea datelor operaționale dintr-un sistem feroviar.

8.2 Arhitectura aplicației

Aplicația este structurată pe baza principiului separării responsabilităților, fiecare componentă având un rol clar definit. Interfața grafică este organizată sub forma mai multor ferestre (*Forms*), fiecare corespunzând unei funcționalități specifice, precum:

- gestionarea facturilor;
- afișarea pieselor și serviciilor asociate;
- filtrarea datelor în funcție de perioadă și centru de cost;
- ștergerea controlată a înregistrărilor.

Interacțiunea utilizatorului cu aplicația se realizează prin intermediul controalelor Windows Forms. Câteva exemple relevante sunt:

- `ComboBox` pentru selecția clienților și a centrelor de cost;
- `ListBox` pentru afișarea listelor de locomotive sau vagoane;
- `ListView` pentru afișarea detaliată a facturilor, pieselor și serviciilor;
- `DateTimePicker` pentru filtrarea informațiilor pe intervale de timp;
- `Label` pentru afișarea valorilor totale calculate.

8.3 Interogări SQL și prelucrarea datelor

Toate calculele financiare sunt realizate direct la nivelul bazei de date, utilizând funcții SQL precum `SUM`, subinterogări și ferestre de tip `OVER()`. Astfel, aplicația nu efectuează calcule în codul C#, ci doar afișează rezultatele returnate de SQL Server. Câteva exemple de astfel de interogări, atât simple, cât și complexe vor fi detaliate într-o secțiune ulterioară.

Filtrarea datelor se realizează în funcție de intervalul de timp selectat de utilizator, folosind controale `DateTimePicker`, iar rezultatele sunt actualizate automat la modificarea acestora.

9 Implementarea logicii aplicației în C#

Logica aplicației este implementată folosind limbajul C#, profitând de paradigma orientată pe obiecte și de modelul bazat pe evenimente (*Event-Driven Programming*) specific Windows Forms. Codul sursă este organizat în clase parțiale (`partial classes`), care separă codul generat automat de designerul vizual de logica de business scrisă de dezvoltator.

9.1 Organizarea ierarhică a formularelor

Arhitectura interfeței grafice este centralizată în jurul formularului principal, `Form1`, care acționează ca un panou de control (*Dashboard*). Acesta gestionează ciclul de viață al aplicației și oferă acces la funcționalitățile principale prin intermediul unui control de tip `TabControl`, care împarte logic aplicația în module (Locomotive, Vagoane, Facturi, Central, etc.).

Pentru operațiunile de manipulare a datelor (inserare, actualizare), aplicația utilizează o serie de formulare auxiliare (de exemplu: `FormFirma`, `FormLocomotiva`, `FormPiesa`). Acestea sunt proiectate să fie modale, fiind deschise prin metoda `.ShowDialog()`. Această abordare blochează interacțiunea cu fereastra principală până la finalizarea operațiunii curente, asigurând astfel integritatea fluxului de date și prevenind erorile de concurență din partea utilizatorului.

9.2 Mecanismul de instanțiere și supraîncărcarea constructorilor

Un aspect esențial al implementării este eficiența codului prin reutilizarea aceluiași formular atât pentru adăugarea de înregistrări noi, cât și pentru modificarea celor existente. Această funcționalitate a fost realizată prin conceptul de **supraîncărcare a constructorilor** (Constructor Overloading).

Fiecare formular auxiliar dispune de doi constructori distincți:

1. **Constructorul fără parametri (Modul Adăugare):** Acesta este apelat atunci când utilizatorul dorește să introducă o entitate nouă. Formularul este instanțiat cu câmpurile goale (sau cu valori implicite), pregătit pentru introducerea datelor. La apăsarea butonului de salvare, se execută o instrucțiune SQL de tip `INSERT`.

2. **Constructorul cu parametri (Modul Editare):** Acesta primește ca argumente detaliile entității selectate din lista principală (ID, nume, coduri etc.). În momentul inițializării (`InitializeComponent`), constructorul populează controalele interfeței (`TextBox`, `ComboBox`) cu valorile primite, permițând utilizatorului să vizualizeze și să modifice datele. La salvare, se execută o instrucțiune SQL de tip `UPDATE` bazată pe ID-ul entității.

Gestiunea operațiunilor asupra datelor (CRUD)

Aplicația implementează setul complet de operațiuni asupra datelor (Create, Read, Update, Delete) pentru toate entitățile gestionate. Funcționalitatea de **adăugare** este disponibilă pentru toate tabelele, asigurând popularea bazei de date.

Deși bunele practici în domeniul bazelor de date recomandă păstrarea istoricului și evitarea ștergerii definitive a informațiilor, a fost implementată și funcționalitatea de **ștergere**. Aceasta este esențială pentru rectificarea erorilor umane de operare (de exemplu, introducerea duplicitară sau eronată a datelor).

De asemenea, pentru majoritatea tabelelor a fost implementată funcționalitatea de **modificare** (Update), permițând actualizarea informațiilor existente fără a necesita ștergerea și reintroducerea acestora, menținând astfel integritatea referențială a bazei de date.

9.3 Sincronizarea datelor și gestionarea erorilor

Comunicarea între formularele auxiliare și formularul principal se realizează sincron. Deoarece formularele secundare sunt deschise modal, execuția codului în `Form1` se oprește la linia `ShowDialog()` și continuă imediat după închiderea ferestrei secundare.

Acest comportament permite actualizarea automată a interfeței principale imediat după o modificare. De exemplu, după adăugarea unei facturi noi, se apelează o metodă centralizată de reîmprospătare (ex. `RefreshToateTaburile()`), care re-interoghează baza de date și actualizează listele (`ListView`, `ListBox`) pentru a reflecta noile schimbări în timp real.

Pentru a asigura robustețea aplicației, interacțiunile cu baza de date sunt încapsulate în blocuri de tip `try-catch`. Aceasta permite captarea excepțiilor (cum ar fi erori de conexiune sau constrângeri de integritate referențială la ștergere) și afișarea unor mesaje clare pentru utilizator prin intermediul `MessageBox`, evitând astfel blocarea neașteptată a aplicației.

10 Funcționarea aplicației și interfața utilizatorului

Interacțiunea principală cu aplicația se realizează prin intermediul ferestrei principale (*Main Form*), care servește drept panou de comandă centralizat. Aceasta este structurată ergonomic folosind un control de tip `TabControl`, care permite navigarea rapidă între diferitele module funcționale fără a aglomera spațiul vizual. Modulele disponibile sunt organizate în tab-uri distincte: **Locomotive**, **Vagoane**, **Central**, **Manevre**, **Facturi** și **Setări**.

În cele ce urmează, vom detalia funcționalitatea și elementele vizuale specifice fiecărui modul în parte.

Aspecte generale privind interfața grafică

Un aspect tehnic important al implementării ferestrei principale (`Main Form`) este flexibilitatea acesteia în ceea ce privește dimensionarea. Spre deosebire de ferestrele modale de adăugare sau editare, care au dimensiuni fixe pentru a păstra coerența datelor introduse, fereastra principală poate fi redimensionată sau maximizată pe tot ecranul monitorului. Această caracteristică permite utilizatorului să vizualizeze un volum mai mare de date simultan, lucru util în special în cazul rapoartelor lungi pe perioade extinse (de exemplu, un an calendaristic).

De asemenea, designul a fost gândit să fie responsive în limitele funcționalității Windows Forms. În modul de vizualizare implicit (fereastră mică), densitatea informației este mare, astfel că unele controale sau coloane din tabele pot necesita utilizarea barelor de derulare (*scroll bars*) orizontale sau verticale pentru a fi accesate. Maximizarea ferestrei elimină de cele mai multe ori această necesitate, oferind o experiență de utilizare mai fluidă și o vizibilitate completă asupra tuturor câmpurilor descrise anterior.

Un aspect fundamental al ergonomiei și arhitecturii aplicației este utilizarea extensivă a listelor de selecție predefinite pentru introducerea datelor, în detrimentul câmpurilor de text liber. Această abordare a fost adoptată pentru a garanta **integritatea referențială** a bazei de date, eliminând riscul erorilor de dactilografiere sau duplicare a informațiilor esențiale.

Pentru ca aceste liste să fie funcționale, aplicația necesită o etapă inițială de configurare, realizată prin intermediul modulului de **Setări**. Aici, utilizatorul trebuie să definească nomenclatoarele sistemului: de la elemente geografice (județe, orașe de interes) până la elemente operaționale (tipuri de piese, servicii disponibile). Această structură relațională impune o ordine logică strictă de populare a datelor, bazată pe dependențe. De exemplu, fluxul de lucru nu permite introducerea unei facturi dacă nu au fost definite locomotivele beneficiare; la rândul lor, locomotivele nu pot fi înregistrate în sistem fără a avea asociat un punct de lucru valid, definit în prealabil.

10.1 Modulul de gestiune a Locomotivelor

Acest tab este dedicat monitorizării detaliate a costurilor și intervențiilor efectuate asupra parcului de locomotive. Interfața este împărțită în două zone principale pentru a facilita fluxul de lucru.

În partea stângă, utilizatorul are acces la o listă completă (`ListBox`) a locomotivelor înregistrate în baza de date. Selectarea unui element din această listă declanșează automat afișarea detaliilor tehnice (serie, tip, putere, punct de lucru) într-un panou informativ dedicat, precum și actualizarea listelor de cheltuieli din partea dreaptă.

Partea dreaptă a interfeței este destinată analizei financiare și operaționale. Un element cheie aici este sistemul de filtrare temporală. Utilizatorul poate defini un interval de timp specific folosind două controale de tip `DateTimePicker` (Data de început și Data de sfârșit). Această flexibilitate este esențială pentru generarea de rapoarte personalizate; de exemplu, se poate dori vizualizarea cheltuielilor pe ultimul an fiscal, analiza costurilor din ultima lună calendaristică, sau chiar un interval arbitrar, cum ar fi de pe data de 15 a lunii curente până pe data de 27 a lunii următoare.

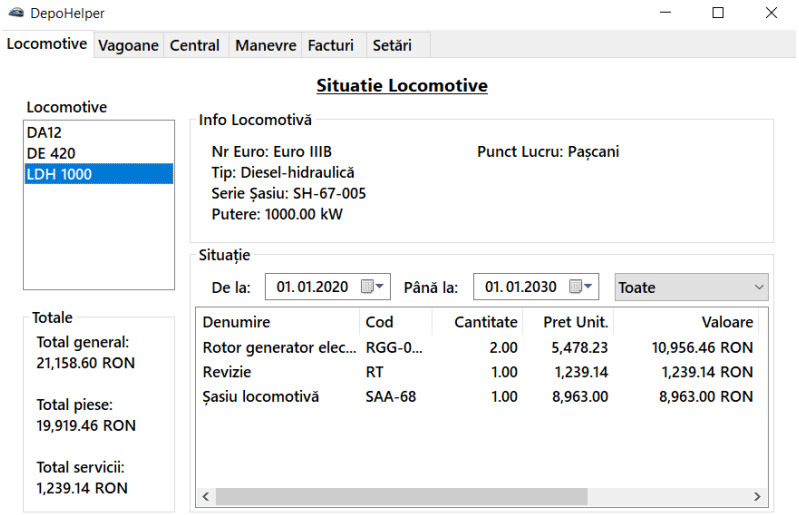

Figura 3: Interfața pentru tabul Locomotive

Odată selectat intervalul și locomotiva, tabelul principal (`ListView`) afișează toate achizițiile și intervențiile. Coloanele tabelului oferă informații granulare precum: denumirea piesei sau serviciului, codul de identificare, cantitatea, prețul unitar, valoarea totală a liniei, furnizorul și data facturii.

Pentru o analiză mai rapidă, în colțul din stânga-jos al acestui panou sunt calculate și afișate automat totalurile financiare:

- **Total Piese:** Suma tuturor pieselor achiziționate în intervalul selectat.
- **Total Servicii:** Costul manoperei sau al serviciilor externe.
- **Total General:** Suma cumulată a celor două categorii anterioare.

De asemenea, interfața dispune de un mecanism de filtrare a tipului de afișare (lângă selectorul de dată), permițând utilizatorului să vizualizeze fie doar piesele, fie doar serviciile, sau lista completă mixtă.

10.2 Modulul de gestiune a Vagoanelor

Deoarece activitatea firmei implică lucrări asupra vagoanelor deținute de clienți terți, structura acestui modul diferă ușor de cea a locomotivelor. Procesul de selecție este ierarhic: utilizatorul alege mai întâi clientul dintr-un meniu derulant (`ComboBox`), moment în care lista de vagoane (`ListBox`) este populată automat doar cu vagoanele aparținând acelui client.

Similar modulului anterior, analiza costurilor se realizează pe baza unui interval de timp selectabil. Totuși, conform cerințelor specifice ale aplicației și logicii de business implementate, pentru vagoane sunt contorizate și afișate exclusiv **piesele** și materialele consumabile. Serviciile nu sunt asociate direct vagoanelor în această vedere, așa cum s-a specificat la începutul documentului.

Tabelul de date afișează denumirea piesei, cantitatea utilizată, prețul și furnizorul, iar eticheta de total din partea dreaptă reflectă suma cheltuielilor cu materialele pentru vagonul și perioada selectată.

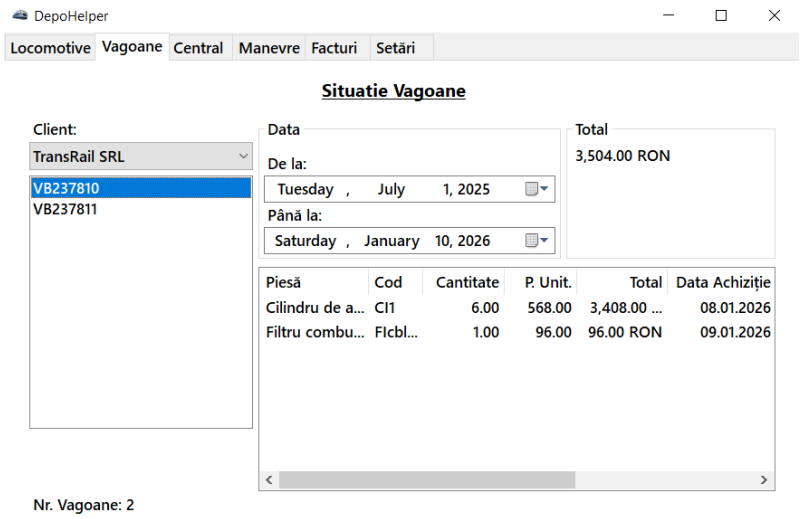

Figura 4: Interfața pentru tabul Vagoane

10.3 Modulul Central (Cheltuieli Administrative)

Acest modul este destinat monitorizării cheltuielilor generale ale întreprinderii, care nu pot fi atribuite direct unui vehicul anume (locomotivă sau vagon). Aici sunt incluse, de regulă, utilitățile, chiriile sau alte servicii administrative.

Interfața este simplificată pentru a oferi o privire de ansamblu rapidă. Utilizatorul selectează perioada de interes (Data de început și Data de sfârșit), iar aplicația interoghează baza de date pentru a extrage toate serviciile înregistrate pe centrul de cost "Central".

Tabelul rezultat prezintă:

- Tipul serviciului prestat;
- Costul acestuia;
- Data efectuării plății sau a facturării;
- Furnizorul și numărul facturii.

Similar celorlalte tab-uri, totalul general al perioadei este afișat proeminent în partea superioară a ferestrei, permițând o verificare rapidă a fluxului de numerar pentru cheltuielile fixe.

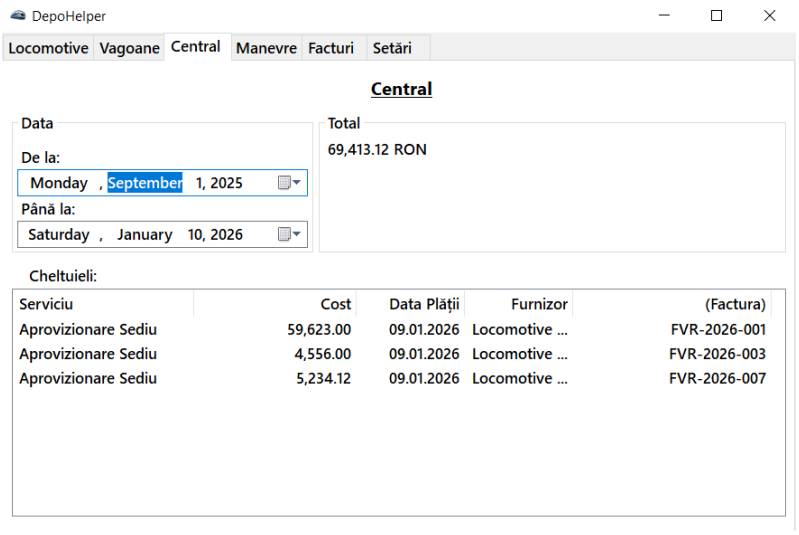

Figura 5: Interfața pentru tabul Central

10.4 Modulul Manevre

Activitatea de manevră feroviară implică costuri specifice, atât materiale cât și de servicii, care necesită o evidență separată.

În partea superioară, utilizatorul regăsește selectorul de interval calendaristic și un panou de sumarizare care afișează trei valori distincte: **Total General**, **Total Piese** și **Total Servicii**.

Zona principală de afișare este divizată în două tabele paralele:

1. **Tabelul din stânga:** Este dedicat pieselor și consumabilelor specifice manevrei. Coloanele includ denumirea piesei, codul de catalog, cantitatea, prețul unitar, valoarea totală, data achiziției și referința facturii.

2. **Tabelul din dreapta:** Listează serviciile prestate (ex. taxe de infrastructură, servicii auxiliare), afișând denumirea serviciului, costul, furnizorul și data.

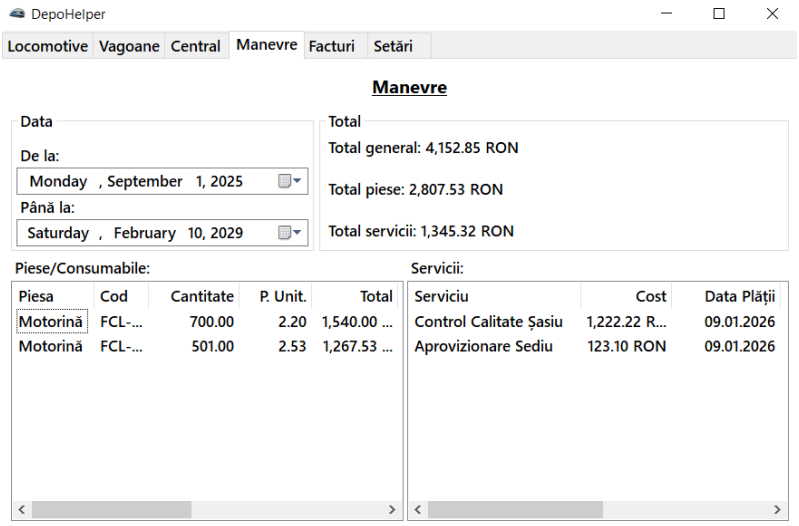

Figura 6: Interfața pentru tabul Manevre

10.5 Modulul de Gestiune a Facturilor

Acest modul reprezintă punctul central de intrare a datelor financiare în sistem. Interfața este proiectată pentru a oferi o transparență totală asupra documentelor fiscale introduse, permițând atât adăugarea de noi facturi, cât și ștergerea celor eronate.

Fereastra este organizată ierarhic pe verticală. În jumătatea superioară se regăsește lista completă (`ListView`) a tuturor facturilor existente în baza de date. Pentru a facilita accesul la informațiile relevante, această listă este sortată descrescător din punct de vedere cronologic, astfel încât cele mai recente documente apar primele.

Selectarea unei facturi din lista principală declanșează popularea automată a celor două liste din partea inferioară a ecranului:

- **Stânga jos:** Lista pieselor și materialelor achiziționate pe factura respectivă;
- **Dreapta jos:** Lista serviciilor prestate și incluse pe aceeași factură.

Această vizualizare detaliată permite operatorului să verifice rapid conținutul unei facturi fără a fi nevoie să deschidă ferestre suplimentare.

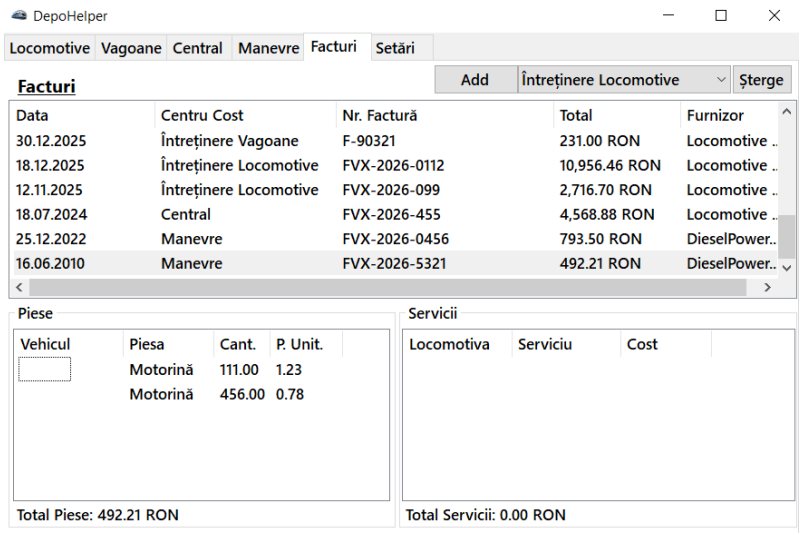

Figura 7: Interfața principală a modulului Facturi

Procesul de adăugare a unei facturi noi

Adăugarea unei facturi noi este un proces contextual, care depinde de centrul de cost căruia îi sunt atribuite cheltuielile. În colțul din dreapta-sus al interfeței, utilizatorul selectează mai întâi tipul centrului facturii dintr-un `ComboBox` (ex: Întreținere Locomotive, Piese Vagoane, Central sau Manevre). Această selecție determină deschiderea unui formular specific de introducere a datelor, adaptat câmpurilor necesare pentru acea categorie.

Luând ca exemplu formularul pentru **Întreținere Locomotive**, interfața de adăugare este structurată pentru a ghida utilizatorul și a preveni erorile de operare. Câmpurile obligatorii, esențiale pentru integritatea bazei de date (precum numărul facturii, data și furnizorul), sunt marcate vizual cu simbolul *.

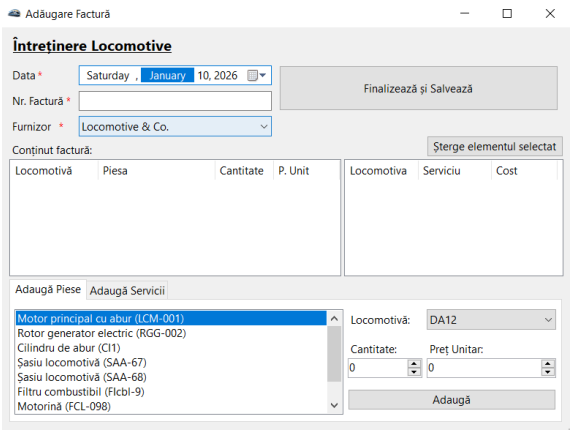

Figura 8: Interfața de adăugare a unei facturi (Întreținere Locomotive)

Gestiunea articolelor de pe factură se realizează prin intermediul unui control `TabControl` situat în partea de jos a ferestrei de adăugare, care separă **Piesele** de **Servicii**.

Fluxul de adăugare a unui articol este următorul:

1. Utilizatorul selectează produsul sau serviciul din listă (gestionată în setări) ;
2. Introduce cantitatea și prețul unitar de achiziție;
3. Selectează destinația specifică a cheltuielii. În cazul formularului de locomotive sau vagoane, este obligatorie selectarea vehiculului (Locomotiva X sau Vagonul Y) pentru care s-a făcut achiziția. Pentru formularele de tip Central sau Manevre, această asociere este implicită.
4. Apasă butonul de adăugare, moment în care linia apare în tabelul de inventar al facturii curente.

Dacă utilizatorul a introdus greșit o linie (de exemplu, o cantitate eronată), formularul dispune de un buton de **ștergere** poziționat deasupra tabelului de inventar, permițând corectarea listei înainte de salvarea finală.

Mecanismul de salvare tranzacțională

Un aspect tehnic important al acestui modul este modul în care sunt gestionate datele în timpul introducerii. Articolele adăugate în listele din formularul de adăugare **nu sunt scrise direct în baza de date** în timp real. Ele sunt salvate temporar în memoria aplicației (în colecții de obiecte sau tabele temporare).

Scrierea efectivă în SQL Server are loc doar în momentul apăsării butonului **Salvează** (sau Finalizează) din colțul dreapta-sus. Acest design asigură atomicitea tranzacției: fie factura este salvată complet (antet + toate liniile de piese și servicii), fie operațiunea este anulată, prevenind astfel apariția în baza de date a unor facturi incomplete.

10.6 Modulul de Setări și Configurare

Acest modul reprezintă componenta administrativă a aplicației, fiind locul unde sunt definite entitățile de bază necesare funcționării sistemului. Este divizat în mai multe sub-categorii pentru a facilita navigarea.

10.6.1 Gestiunea Partenerilor (Firme)

Această secțiune permite administrarea bazei de date de clienți și furnizori. Interfața principală este împărțită pe verticală: în partea stângă sunt afișate două liste distincte (`ListBox`), una pentru Furnizori și una pentru Clienți, iar în partea dreaptă este prezentat un panou de detalii care se actualizează automat la selectarea unei firme.

Acest panou centralizează toate informațiile relevante despre entitate: datele fiscale, adresa sediului și informațiile persoanei de contact.

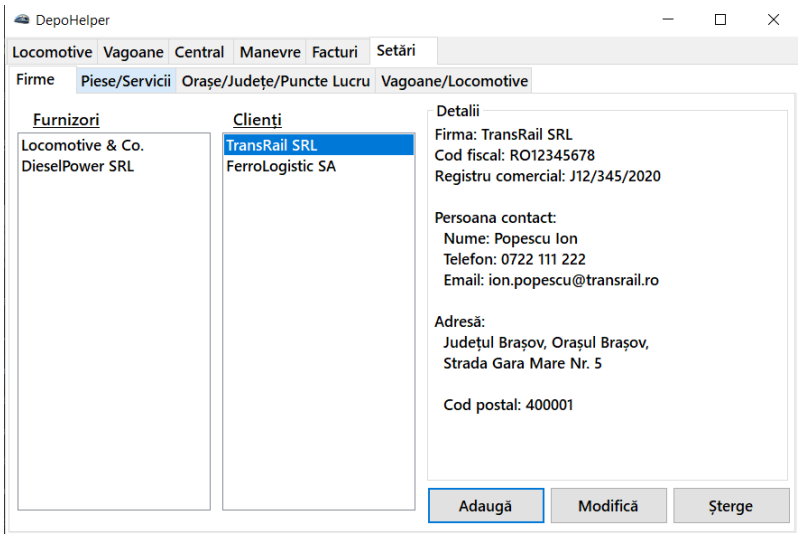

Figura 9: Interfața de gestionare a Firmelor (Clienți și Furnizori)

Formularul de adăugare și editare a firmelor este construit intuitiv. Câmpurile obligatorii sunt marcate vizual cu simbolul *, validarea acestora fiind necesară pentru salvarea înregistrării. Utilizatorul selectează tipul relației (Client sau Furnizor), introduce datele de identificare ale companiei, detaliile persoanei de contact și elementele de adresă (inclusiv codul poștal).

Din punct de vedere al implementării backend, deși utilizatorul completează un singur formular unificat, aplicația gestionează automat distribuirea datelor în tabelele normalizate ale bazei de date (`Firme`, `Adrese`, `PersoaneContact`), asigurând coerența structurii relaționale.

10.6.2 Nomenclatoarele de Piese și Servicii

În această secțiune se construiește catalogul de materiale și operațiuni disponibile pentru facturare. Interfața permite adăugarea, modificarea și ștergerea entităților.

Un aspect critic implementat aici este **integritatea referențială**: aplicația restricționează ștergerea unei piese sau a unui serviciu dacă acesta a fost deja utilizat într-o factură existentă în sistem. Această validare previne coruperea istoricului de achiziții și invalidarea rapoartelor financiare anterioare.

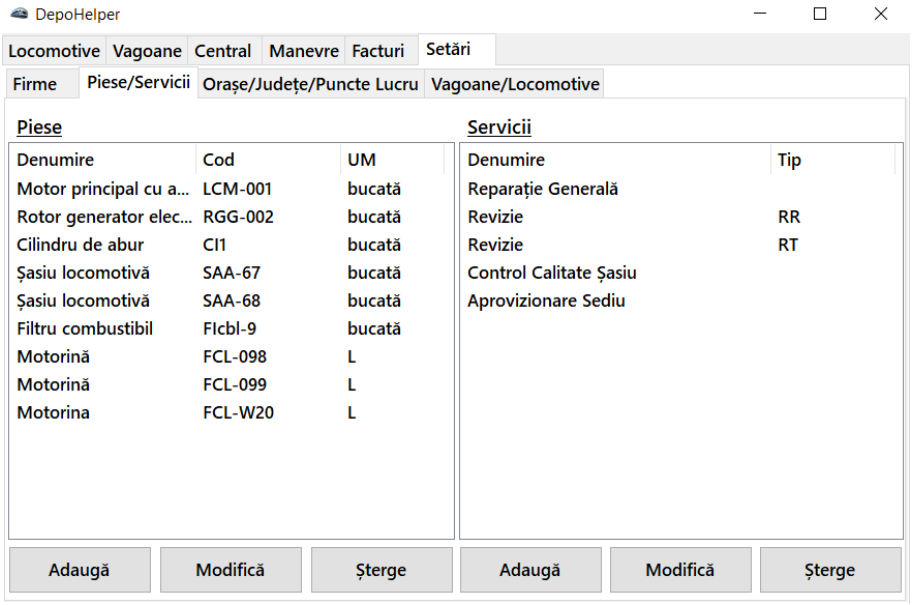

Figura 10: Interfața pentru Piese și Servicii

10.6.3 Configurarea Geografică și a Punctelor de Lucru

Această interfață gestionează structura teritorială a operațiunilor. Layout-ul este organizat pentru a reflecta ierarhia datelor:

- **Stânga-Sus:** Lista Județelor;
- **Stânga-Jos:** Lista Orașelor de interes din județul selectat anterior.

Manipularea datelor se face prin butoanele dedicate `+` (Adăugare) și `-` (Ștergere). Pentru modificarea unei înregistrări (de exemplu, corectarea numelui unui oraș), a fost implementat un mecanism de editare rapidă: **dublu-click** pe elementul din listă încarcă numele acestuia în caseta de text și schimbă culoarea fontului în albastru, semnalizând vizual modul de editare. După modificare, apăsarea butonului `+` salvează schimbarea (Update), în loc să creeze o intrare nouă.

Partea dreaptă a interfeței este dedicată definirii **Punctelor de Lucru**. Utilizatorul poate selecta un oraș deja definit din lista derulantă (`ComboBox`) situată în partea inferioară și, prin apăsarea butonului `+`, îl promovează la statutul de Punct de Lucru activ, făcându-l disponibil pentru asocierea cu locomotivele.

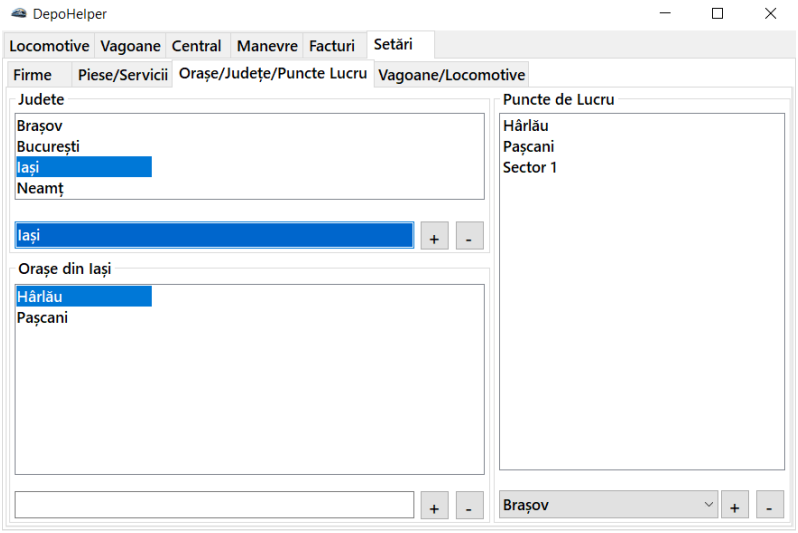

Figura 11: Interfața pentru Județe, Orașe și Puncte de Lucru

10.6.4 Gestiunea Locomotivelor și Vagoanelor

Ultimul tab al modulului de setări este dedicat configurării parcului de material rulant.

Secțiunea superioară gestionează **Locomotivele**. Aici pot fi introduse noi unități, specificând caracteristicile tehnice și punctul de lucru alocat. Similar pieselor, o locomotivă nu poate fi ștearsă din sistem dacă există facturi sau intervenții asociate acesteia.

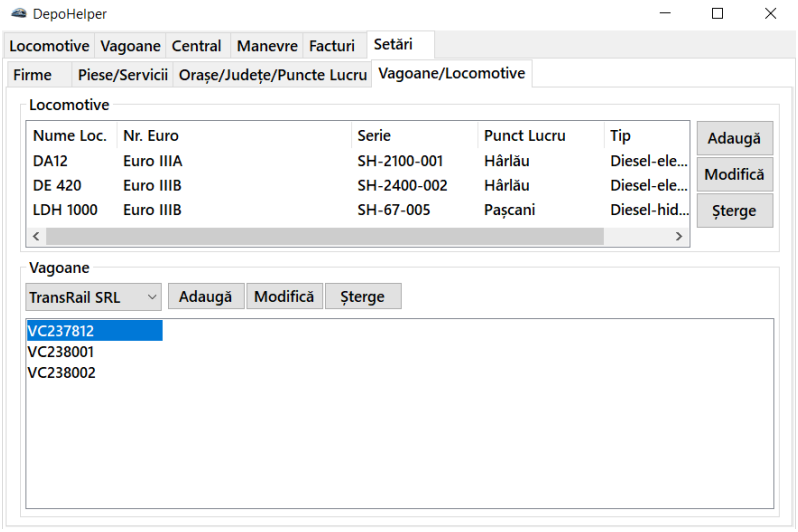

Figura 12: Interfața pentru Locomotive și Vagoane

Secțiunea inferioară este dedicată **Vagoanelor**. Deoarece acestea aparțin clienților, lista este filtrată dinamic pe baza proprietarului selectat în `ComboBox`-ul aferent. La adăugarea unui vagon nou, aplicația preia implicit clientul selectat în lista de filtrare, simplificând procesul de introducere a datelor, însă permite modificarea acestuia dacă se dorește.

11 Explicarea unor secvențe de cod SQL

Interacțiunea dintre aplicația C# și baza de date SQL Server este realizată prin interogări optimizate, menite să asigure atât viteza de răspuns, cât și integritatea datelor. În această secțiune sunt analizate o serie de interogări reprezentative, grupate în funcție de complexitatea și scopul lor.

11.1 Interogări cu complexitate redusă (Simple Joins)

Aceste interogări sunt utilizate în principal pentru popularea listelor și afișarea detaliilor entităților. Ele se bazează pe clauza `JOIN` pentru a reconstrui informația provenită din tabele normalizate.

11.1.1 Identificarea locației unei locomotive

Această interogare ilustrează navigarea prin structura relațională a bazei de date. Tabelul `Locomotive` stochează doar identificatorul punctului de lucru, iar punctul de lucru conține la rândul său identificatorul orașului. Prin urmare, este necesară utilizarea a două îmbinări succesive (`JOIN`) pentru a obține numele orașului asociat unei locomotive.

```
SELECT _oras.numeOras
FROM Locomotive _locomotiva
JOIN PuncteLucru _punct
    ON _locomotiva.idPunctLucru = _punct.idPunctLucru
JOIN Orase _oras
    ON _punct.idOras = _oras.idOras
WHERE _locomotiva.idLocomotiva = @idLocomotiva;
```

11.1.2 Raport detaliat de piese pentru vagoane

Această interogare generează lista completă a pieselor utilizate pentru un anumit vagon într-un interval de timp. Sunt îmbinate cinci tabele pentru a obține un raport complet ce include piesa, vagonul, factura și firma furnizoare. De asemenea, se realizează un calcul aritmetic direct în lista de selecție (`cantitate * pretUnitar`) și se filtrează rezultatele prin eliminarea componentei de timp a datei folosind `CAST`.

```
SELECT
    p.denumirePiesa AS Piesa,
    p.codPiesa AS Cod,
    pv.cantitate AS Cantitate,
    pv.pretUnitar AS PretUnitar,
    (pv.cantitate * pv.pretUnitar) AS Total,
    f.numeFirma AS Furnizor,
    fct.numarFactura AS NumarFactura,
    fct.dataFactura AS DataAchizitie
FROM PieseVagoane pv
JOIN Piese p ON pv.idPiesa = p.idPiesa
JOIN Vagoane v ON pv.idVagon = v.idVagon
JOIN Facturi fct ON pv.idFactura = fct.idFactura
JOIN Firme f ON fct.idFurnizor = f.idFirma
WHERE v.idProprietar = @idClient
  AND v.numarVagon = @numarVagon
  AND CAST(fct.dataFactura AS DATE)
      BETWEEN @dataStart AND @dataEnd
ORDER BY fct.dataFactura;
```

11.1.3 Calculul totalului investit într-un vagon

Spre deosebire de interogarea anterioară, care returnează o listă de înregistrări, această interogare furnizează o singură valoare scalară reprezentând totalul general. Se utilizează funcția de agregare `SUM`, iar funcția `ISNULL` previne returnarea valorii `NULL` în lipsa înregistrărilor.

```
SELECT ISNULL(SUM(pv.cantitate * pv.pretUnitar), 0) AS TotalGeneral
FROM PieseVagoane pv
JOIN Vagoane v ON pv.idVagon = v.idVagon
JOIN Facturi fct ON pv.idFactura = fct.idFactura
WHERE v.idProprietar = @idClient
  AND v.numarVagon = @numarVagon
  AND CAST(fct.dataFactura AS DATE)
      BETWEEN @dataStart AND @dataEnd;
```

11.1.4 Fișa completă a unei firme

Această interogare este utilizată în modulul de setări pentru afișarea completă a detaliilor unei firme. Complexitatea constă în numărul ridicat de îmbinări necesare pentru reconstruirea adresei complete (stradă, oraș, județ) și a datelor persoanei de contact.

```
SELECT
    _firma.numeFirma,
    _firma.codFiscal,
    _firma.registruComercial,
    _psc.nume,
    _psc.prenume,
    _psc.telefon,
    _psc.email,
    _adresa.strada,
    _adresa.numar,
    _adresa.codPostal,
    _oras.numeOras,
    _judet.numeJudet
FROM Firme _firma
JOIN PersoaneContact _psc
    ON _firma.idPersoanaContact = _psc.idPersoanaContact
JOIN Adrese _adresa
    ON _firma.idAdresa = _adresa.idAdresa
JOIN Orase _oras
    ON _adresa.idOras = _oras.idOras
JOIN Judete _judet
    ON _oras.idJudet = _judet.idJudet
WHERE _firma.idFirma = @id;
```

11.1.5 Identificarea orașelor disponibile pentru puncte de lucru

Interogarea utilizează un `LEFT JOIN` combinat cu condiția `IS NULL` pentru a selecta doar orașele care nu sunt deja asociate unui punct de lucru, evitând astfel duplicatele logice în interfața aplicației.

```
SELECT _oras.idOras, _oras.numeOras
FROM Orase _oras
LEFT JOIN PuncteLucru _pl
    ON _oras.idOras = _pl.idOras
WHERE _pl.idOras IS NULL
ORDER BY _oras.numeOras;
```

11.1.6 Listarea facturilor cu detalii despre furnizor și centru de cost

Această interogare este utilizată pentru raportare și afișează facturile împreună cu denumirile furnizorilor și ale centrelor de cost, ordonate descrescător după dată.

```
SELECT
    f.idFactura,
    f.dataFactura,
    cc.denumireCentruCost AS CentruCost,
    f.numarFactura,
    f.sumaTotal,
    frm.numeFirma
FROM Facturi f
JOIN CentreCost cc ON f.idCentruCost = cc.idCentruCost
JOIN Firme frm ON f.idFurnizor = frm.idFirma
ORDER BY f.dataFactura DESC;
```

11.2 Interogări complexe

Aceste interogări implică operații avansate precum uniuni de seturi de date, subinterogări și funcții de fereastră (*Window Functions*).

11.2.1 Unificarea fluxului de piese și servicii (`UNION ALL`)

Pentru obținerea unei imagini cronologice complete a cheltuielilor unei locomotive, datele din tabelele `PieseLocomotive` și `ServiciiPrestate` sunt combinate folosind operatorul `UNION ALL`. Se construiește un set de rezultate unificat, standardizat printr-o coloană artificială `Tip`.

```
SELECT *
FROM (
    SELECT
        'Piesa' AS Tip,
        p.denumirePiesa AS Denumire,
        (pl.cantitate * pl.pretUnitar) AS Valoare,
        fa.dataFactura
    FROM PieseLocomotive pl
    -- JOIN-uri specifice pieselor
    WHERE pl.idLocomotiva = @idLocomotiva

    UNION ALL

    SELECT
        'Serviciu' AS Tip,
        s.denumireServiciu AS Denumire,
        sp.cost AS Valoare,
        fa.dataFactura
    FROM ServiciiPrestate sp
```

```
    -- JOIN-uri specifice serviciilor
    WHERE sp.idLocomotiva = @idLocomotiva
) X
WHERE (@tip = 0)
   OR (@tip = 1 AND X.Tip = 'Piesa')
   OR (@tip = 2 AND X.Tip = 'Serviciu')
ORDER BY X.dataFactura;
```

11.2.2 Calcularea simultană a totalurilor agregate

Interogarea utilizează subinterogări scalare pentru a obține într-un singur apel totalul pieselor și totalul serviciilor aferente unei locomotive.

```
SELECT
(
    SELECT ISNULL(SUM(pl.cantitate * pl.pretUnitar), 0)
    FROM PieseLocomotive pl
    JOIN Facturi fa ON pl.idFactura = fa.idFactura
    WHERE pl.idLocomotiva = @idLocomotiva
      AND CAST(fa.dataFactura AS DATE)
          BETWEEN @dataStart AND @dataEnd
) AS TotalPiese,
(
    SELECT ISNULL(SUM(sp.cost), 0)
    FROM ServiciiPrestate sp
    JOIN Facturi fa ON sp.idFactura = fa.idFactura
    WHERE sp.idLocomotiva = @idLocomotiva
      AND CAST(fa.dataFactura AS DATE)
          BETWEEN @dataStart AND @dataEnd
) AS TotalServicii;
```

11.2.3 Filtrare dinamică bazată pe subinterogare

Această interogare evită utilizarea identificatorilor fixați, determinând dinamic ID-ul centrului de cost pe baza denumirii sale.

```
SELECT
    s.denumireServiciu AS Serviciu,
    sp.cost AS Cost,
    fct.dataFactura AS DataPlatii
FROM ServiciiPrestate sp
-- JOIN-uri relevante
WHERE CAST(fct.dataFactura AS DATE)
      BETWEEN @dataStart AND @dataEnd
  AND fct.idCentruCost = (
        SELECT idCentruCost
        FROM CentreCost
        WHERE denumireCentruCost = 'Central'
  )
ORDER BY fct.dataFactura;
```

11.2.4 Calculul totalului general folosind funcții de fereastră

Utilizarea funcției `SUM(...) OVER()` permite calcularea totalului general fără a pierde detaliile individuale ale fiecărei linii, oferind aplicației atât lista detaliată, cât și totalul într-un singur set

de date.

```
SELECT
    T.Destinatar,
    T.denumirePiesa,
    T.cantitate,
    T.pretUnitar,
    T.Valoare,
    SUM(T.Valoare) OVER() AS TotalGeneral
FROM (
    SELECT
        L.numeLocomotiva AS Destinatar,
        (PL.cantitate * PL.pretUnitar) AS Valoare
    FROM PieseLocomotive PL
    -- JOIN-uri
    WHERE PL.idFactura = @idFactura

    UNION ALL

    SELECT
        'Vagon ' + V.numarVagon AS Destinatar,
        (PV.cantitate * PV.pretUnitar) AS Valoare
    FROM PieseVagoane PV
    -- JOIN-uri
    WHERE PV.idFactura = @idFactura
) AS T;
```

12 Concluzii

Proiectul de față a avut ca obiectiv principal proiectarea și implementarea unei soluții software robuste pentru digitalizarea și eficientizarea activității de gestiune a unui depou feroviar.

Din punct de vedere tehnic, utilizarea platformei .NET Framework alături de sistemul de gestiune a bazelor de date Microsoft SQL Server s-a dovedit a fi o alegere optimă, oferind stabilitatea și performanța necesare manipulării unui volum mare de date. Arhitectura aplicației, bazată pe separarea logicii de interfață de logica datelor și implementarea unor mecanisme stricte de validare (precum listele predefinite și constrângerile de integritate referențială la ștergere), asigură o fiabilitate ridicată și minimizează riscul erorilor umane.

Funcționalitatea sistemului acoperă întregul ciclu de viață al datelor operaționale, de la definirea nomenclatoarelor și configurarea punctelor de lucru, până la introducerea facturilor complexe și generarea rapoartelor de costuri pe centre de profit (Locomotive, Manevre, Central). Utilizarea interogărilor SQL avansate a permis delegarea calculelor complexe către serverul de baze de date, menținând aplicația rapidă și responsivă.

În concluzie, aplicația realizată răspunde cerințelor actuale de modernizare a infrastructurii informatice feroviare, oferind un instrument practic pentru monitorizarea cheltuielilor și luarea deciziilor manageriale. Ca direcții viitoare de dezvoltare, sistemul poate fi extins prin implementarea unui modul de autentificare cu drepturi granulare de acces sau prin migrarea către o arhitectură distribuită, accesibilă via web, pentru a permite operarea de la distanță.