UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

**Document de Analiză și Proiectare pentru Aplicația Software**

STUDENT:

Lepădat Ioana-Dana

BUCUREȘTI, 2025

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

**Sistem de gestionare a inventarului pentru mic afaceri**

**StockMaster**

STUDENT:

Lepădat Ioana-Dana

ioana-dana.lepadat@s.unibuc.ro

BUCUREȘTI, 2025

[1. Introducere 2](#_Toc1406138203)

[1.1 Scopul documentului 2](#_Toc2098686201)

[1.2 Domeniul de aplicare 2](#_Toc1134866002)

[1.3 Definiții, acronime și abrevieri 3](#_Toc11823328)

[1.4 Referințe 3](#_Toc2142695885)

[Documente și Standarde 3](#_Toc1508246704)

[Surse Web și Bibliografie 3](#_Toc1000685718)

[1.5 Prezentare generală 4](#_Toc252828748)

[1.5.1. Cerințele funcționale și non-funcționale ale aplicației 4](#_Toc1063497244)

[1.5. 2. Arhitectura în 3 straturi (UI, BLL, DAL) 4](#_Toc962291919)

[1.5. 3. Modele de date și diagrame UML 5](#_Toc591196130)

[1.5. 4 Detalii despre implementare, testare și mentenanță 5](#_Toc974561377)

[1.5.5 Capturi de ecran pentru a evidenția aspecte UI/UX 5](#_Toc1154690763)

[2. Analiza cerințelor 6](#_Toc350515628)

[2.1 Cerințe funcționale 6](#_Toc685087298)

[2.2 Cerințe non-funcționale 7](#_Toc1758616293)

[2.3 Cazuri de utilizare (Use Cases) 8](#_Toc1691058411)

[2.4 Constrângeri 9](#_Toc1807121313)

[2.5 Riscuri 10](#_Toc1947700960)

[3. Arhitectura și proiectarea sistemului 11](#_Toc2004622024)

[3.1 Arhitectura generală 11](#_Toc1238661263)

[3.2 Modelul de date 12](#_Toc451805512)

[Entitățile principale 12](#_Toc2023136329)

[Categorie\_Produse 12](#_Toc1315098590)

[Furnizori 13](#_Toc579432502)

[Hoteluri 13](#_Toc720986188)

[Produse 13](#_Toc718010746)

[3.3 Diagrame UML (Unified Modelling Language) 14](#_Toc126858604)

[1. Diagrama de clase 15](#_Toc1899349470)

[Clase principale 15](#_Toc1478876571)

[Relații 15](#_Toc1356028122)

[2. Diagrama de secvență – Autentificare utilizator 16](#_Toc764560730)

[Flux: 16](#_Toc907765608)

[3. Diagrama de stare (opțional) 17](#_Toc1471928936)

[Stări: 17](#_Toc799249604)

[Tranziții: 17](#_Toc1606647690)

[4. Diagrama de activitate 18](#_Toc1281795127)

[Flux logic: 18](#_Toc1808561831)

[3.4 Tehnologii și stack de dezvoltare 18](#_Toc1553252460)

[Limbaj de programare: 18](#_Toc1971133248)

[Framework și platformă: 18](#_Toc468774189)

[Baze de date: 19](#_Toc1576095686)

[Structură arhitecturală: 19](#_Toc1975060176)

[Managementul configurațiilor: 19](#_Toc655202301)

[Mediu de dezvoltare: 19](#_Toc1366874103)

[Testare: 19](#_Toc963412975)

[3.5 Interfața utilizatorului (UI/UX) 19](#_Toc548876762)

[1. Principii UI/UX aplicate 20](#_Toc238600926)

[2. Structura interfeței 20](#_Toc122018514)

[3. Protocoale și tehnologii UI 27](#_Toc1371405032)

[4. Implementare 27](#_Toc863039590)

[4.1 Structura codului 27](#_Toc1877058625)

[Aplicație desktop (.NET Framework – WinForms) 28](#_Toc471829034)

[StockMaster (Frontend / UI Layer – Interfața utilizatorului) 28](#_Toc295598977)

[Business Logic Layer (BLL) 28](#_Toc2124736759)

[StockMaster.BLL 28](#_Toc1556266789)

[Data Access Layer (DAL) 28](#_Toc1253028028)

[StockMaster.DAL 29](#_Toc764187706)

[Baza de date (SQL Server) 29](#_Toc1721332885)

[Database/Tables 29](#_Toc1635299360)

[Configurare și resurse 29](#_Toc586869818)

[4.2 Module și componente 29](#_Toc833812937)

[1. Modul de autentificare 29](#_Toc184690119)

[2. Modul de gestionare a produselor 30](#_Toc1903526144)

[3. Modul de modificare a stocului 30](#_Toc777293386)

[4. Modul de administrare hoteluri (doar pentru admin) 30](#_Toc1498726343)

[5. Modul de alertare pentru stoc critic 30](#_Toc774352006)

[4.3 Fluxul de date 31](#_Toc1141898620)

[1. Fluxul general al datelor 31](#_Toc995667611)

[2. Procesarea datelor 31](#_Toc950366771)

[3. Flux de date pentru acțiuni specifice 32](#_Toc552381303)

[1. Adăugare produs nou: 32](#_Toc1704710419)

[2. Modificare cantitate produs: 32](#_Toc1176844416)

[3.Căutare produs: 32](#_Toc946939453)

[4.Alerte de stoc critic: 33](#_Toc1789864038)

[4. Notificări și evenimente 33](#_Toc230708981)

[5. Testare 33](#_Toc713989783)

[5.1 Strategia de testare 33](#_Toc960192900)

[1. Testare unitară (manuală) 34](#_Toc996690034)

[2. Testare de integrare 34](#_Toc545958169)

[️ 3. Testare de performanță (manuală, observabilă) 34](#_Toc1126414691)

[4. Testare de securitate (de bază) 35](#_Toc1341782573)

[5. Testare de acceptanță 35](#_Toc1491003039)

[6. Automatizarea testelor 36](#_Toc851284490)

[5.2 Tipuri de teste 36](#_Toc1539581592)

[1. Teste unitare (manuale) 36](#_Toc1551942920)

[2. Teste de integrare 37](#_Toc2103697064)

[3. Teste de validare și UI 37](#_Toc1771166912)

[4. Teste de performanță (vizuale) 37](#_Toc1299236701)

[5. Teste de securitate (de bază) 38](#_Toc511739595)

[6. Teste de acceptanță 38](#_Toc334342387)

[5.3 Plan de testare 38](#_Toc865511037)

[1. Obiectivele testării 38](#_Toc1346451211)

[2. Scenarii de testare 39](#_Toc614169534)

[Testarea autentificării 39](#_Toc1675893526)

[Testarea adăugării unui produs 39](#_Toc1841525231)

[Testarea modificării stocului 39](#_Toc1837002734)

[Testarea alertei de stoc critic 40](#_Toc1288600288)

[Testarea căutării produselor 40](#_Toc1895520833)

[Testarea securității inputului 40](#_Toc1682576557)

[3. Instrumente utilizate pentru testare 41](#_Toc438443338)

[6. Deployment și mentenanță 41](#_Toc1021896097)

[6.1 Strategie de deployment 41](#_Toc30109257)

[1. Medii de implementare 41](#_Toc123498123)

[2. Procesul de implementare 42](#_Toc1592317838)

[3. Tehnologii utilizate în deployment 43](#_Toc1169175892)

[Observație 43](#_Toc1835560981)

[6.2 Administrarea și monitorizarea 43](#_Toc780284678)

[1. Monitorizarea performanței 44](#_Toc533070337)

[2. Monitorizarea securității 44](#_Toc765999726)

[3. Administrarea aplicației 44](#_Toc479945115)

[4. Notificări și alerte 45](#_Toc1119871101)

[6.3 Plan de mentenanță 45](#_Toc178303791)

[1. Tipuri de mentenanță 46](#_Toc1127424798)

[2. Program de mentenanță recomandat 46](#_Toc892988197)

[3. Instrumente și procese de mentenanță 47](#_Toc64560951)

[4. Suport tehnic 47](#_Toc627449016)

[7. Concluzii 47](#_Toc478577868)

[1. Rezumat al principalelor constatări 48](#_Toc624271780)

[2. Recomandări pentru dezvoltări viitoare 48](#_Toc213305287)

## **1. Introducere**

### **1.1 Scopul documentului**

Acest document are scopul de a descrie cerințele, arhitectura, proiectarea și detaliile de implementare ale aplicației software **StockMaster**.  
 Documentul servește ca punct de referință pentru dezvoltatori, testeri și alți membri implicați în proiect, oferind o bază clară pentru testare, mentenanță și posibile extinderi viitoare.

### **1.2 Domeniul de aplicare**

**StockMaster** este o aplicație desktop C# Windows Forms destinată hotelurilor, care permite gestionarea stocurilor de produse pe categorii (ex: Bucătărie, Bar).  
 Utilizatorii principali sunt:

* Administratori (cu drepturi de adăugare și ștergere hoteluri)
* Manageri hotelieri (care pot gestiona stocurile interne)

Funcționalitățile principale includ autentificare, afișare și căutare produse, adăugare și modificare produse, alertă pentru stocuri critice.

### **1.3 Definiții, acronime și abrevieri**

|  |  |
| --- | --- |
| **Termen** | **Definiție** |
| **DAL** | Data Access Layer – Strat de acces la baza de date |
| **BLL** | Business Logic Layer – Strat de logică de afacere |
| **UI** | User Interface – Interfață grafică pentru utilizator |
| **ERD** | Entity Relationship Diagram – Diagramă entitate-relație |
| **SQL** | Structured Query Language – Limbaj pentru baze de date |
| **WinForms** | Windows Forms – Tehnologie pentru aplicații desktop în C# |

### **1.4 Referințe**

Această secțiune conține lista documentelor, standardelor și surselor utilizate în procesul de analiză și proiectare a aplicației. Referințele asigură validitatea și conformitatea soluției propuse cu cele mai bune practici și cerințe industriale.

#### **Documente și Standarde**

* **ISO/IEC 25010:2011** – *Model de calitate pentru produse software*, relevant pentru evaluarea caracteristicilor aplicației StockMaster (fiabilitate, securitate, ușurință în utilizare).
* **UML 2.5 Specification** – *Standard de modelare utilizat pentru diagramele de clase, cazuri de utilizare și secvență în documentația tehnică*.
* **Microsoft Secure Coding Guidelines** – *Ghiduri oficiale Microsoft pentru scrierea unui cod sigur în aplicațiile .NET*.
* **OWASP Secure Coding Practices** – *Recomandări pentru prevenirea vulnerabilităților în aplicațiile desktop și web (ex. SQL Injection, input validation)*. *(opțional)*
* **Microsoft .NET WinForms Development Guidelines** – *Recomandări oficiale pentru dezvoltarea de aplicații desktop robuste și scalabile folosind Windows Forms.*

#### **Surse Web și Bibliografie**

* **Microsoft Learn – Windows Forms Documentation.** <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/>
* **Microsoft Learn – ADO.NET Overview.** <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>
* **Object Management Group (OMG).** *UML 2.5 Specification.* <https://www.omg.org/spec/UML/2.5>

## **1.5 Prezentare generală**

Acest document detaliază procesul de analiză, proiectare și implementare al aplicației **StockMaster**, dezvoltată în limbajul C# utilizând tehnologia Windows Forms. Aplicația este destinată gestionării stocurilor de produse din cadrul hotelurilor, permițând autentificare, afișare și manipulare a produselor în funcție de categorie și locație.

Structura documentului este împărțită în următoarele secțiuni principale:

### **1.5.1.** **Cerințele funcționale și non-funcționale ale aplicației**

Acestea definesc ce trebuie să facă aplicația (autentificare, gestiune stocuri, alerte etc.) și sub ce condiții trebuie să funcționeze (performanță, securitate, fiabilitate etc.).

### **1.5. 2.** **Arhitectura în 3 straturi (UI, BLL, DAL)**

Aplicația respectă o arhitectură modulară și clar separată:

* **UI (User Interface):** implementată în WinForms (Form1, LoginForm etc.)
* **BLL (Business Logic Layer):** gestionează logica aplicației (ProductService)
* **DAL (Data Access Layer):** interacționează cu baza de date SQL Server (ProductRepository, clasa Product)

Această separare facilitează mentenanța, testarea și extinderea aplicației.

### **1.5. 3.** **Modele de date și diagrame UML**

Documentația include următoarele reprezentări vizuale:

* **Model de date (ERD – Entity Relationship Diagram):**
  + Prezintă entitățile principale: *Produse*, *Hoteluri*, *Categorii*, și relațiile dintre ele.
  + De exemplu: un *Hotel* poate avea mai multe *Produse*, iar un *Produs* aparține unei *Categorii*.
* **Diagramă de clase UML:**
  + Ilustrează clasele folosite în aplicație (ex: Product, ProductService, ProductRepository) și relațiile dintre ele (compoziție, agregare, asociere).
  + Se va evidenția organizarea în straturi și responsabilitățile fiecărei clase.
* **Diagramă de cazuri de utilizare (Use Case):**
  + Prezintă interacțiunile utilizatorilor cu sistemul (Login, Adăugare produs, Afișare produse, Modificare stoc etc.)
  + Include actori: *Utilizator*, *Administrator* și *Sistemul*.
* **Opțional: Diagramă de secvență:**
  + Pentru a arăta cum interacționează obiectele între ele în timpul unei operațiuni, cum ar fi actualizarea cantității în stoc.

### **1.5. 4** **Detalii despre implementare, testare și mentenanță**

Fiecare strat din aplicație este explicat în detaliu: clase, metode, fluxuri de date. De asemenea, este descris modul în care aplicația este testată (funcțional, integrare, validări), și planul pentru întreținerea sistemului.

### **1.5.5** **Capturi de ecran pentru a evidenția aspecte UI/UX**

Sunt incluse capturi de ecran reale din aplicație care ilustrează:

* Ecranul de login
* Interfața de adăugare produs
* Afișarea stocurilor
* Alertă de stoc redus
* Formularul de modificare cantitate

Aceste imagini sprijină descrierile funcționale și arhitecturale și oferă claritate asupra modului de utilizare al aplicației.

## **2. Analiza cerințelor**

### **2.1 Cerințe funcționale**

Aplicația **StockMaster** oferă funcționalități pentru gestionarea eficientă a produselor din stoc, organizate pe categorii, în cadrul hotelurilor. Cerințele funcționale principale sunt:

1. **Autentificare utilizator**
   1. Utilizatorii introduc un nume și o parolă pentru a accesa sistemul.
   2. Utilizatorul cu rol „admin” are acces la funcționalități suplimentare precum adăugarea sau ștergerea de hoteluri.
2. **Afișare produse pe categorii**
   1. Utilizatorul poate vizualiza produsele filtrate după categoriile principale: *Bar* și *Bucătărie*.
   2. Se afișează atributele relevante: nume, descriere, cantitate și preț unitar.
3. **Adăugare produs nou**
   1. Se poate adăuga un produs în baza de date prin completarea unui formular cu: nume, descriere, categorie, cantitate și preț.
   2. Se efectuează validări pentru completarea corectă a câmpurilor.
4. **Modificare cantitate produs**
   1. Utilizatorul poate selecta un produs și modifica cantitatea acestuia printr-un formular dedicat.
5. **Căutare produs**
   1. Sistemul permite căutarea produselor după nume, în mod dinamic.
6. **Alerte pentru stoc redus**
   1. Aplicația generează o notificare vizuală atunci când există produse cu stoc mai mic de 4 unități.
7. **Gestionare hoteluri (admin)**
   1. Administratorul poate adăuga un hotel nou completând informațiile necesare: nume hotel, nume administrator, username, parolă și email.
8. **Navigare și control UI**
   1. Interfața afișează sau ascunde elementele de control în funcție de context (categorie selectată, rol utilizator etc.).

### **2.2 Cerințe non-funcționale**

1. **Performanță**
   1. Operațiile de interogare și afișare produse trebuie să se execute în maximum 2 secunde.
   2. Interfața trebuie să fie responsivă și fără întârzieri vizibile.
2. **Securitate**
   1. Parolele sunt validate și nu se stochează în interfață.
   2. Se utilizează interogări SQL parametrizate pentru prevenirea atacurilor de tip SQL Injection.
   3. Datele de conectare și acces sunt gestionate separat pentru utilizatori și administratori.
3. **Ușurință în utilizare**
   1. Interfața este simplă și intuitivă, cu butoane etichetate clar.
   2. Câmpurile vizibile se adaptează în funcție de acțiunile utilizatorului.
4. **Fiabilitate**
   1. Aplicația trebuie să funcționeze fără blocaje sau excepții în condiții normale de utilizare.
   2. Sistemul gestionează corect cazurile de input invalid sau câmpuri necompletate.
5. **Compatibilitate**
   1. Aplicația trebuie să ruleze pe sisteme Windows cu .NET Framework instalat.
6. **Extensibilitate**
   1. Structura modulară (UI-BLL-DAL) permite adăugarea ușoară de noi funcționalități în viitor (ex: raportare, utilizatori multipli).

### **2.3 Cazuri de utilizare (Use Cases)**

UC1: Autentificare utilizator

**Descriere:** Utilizatorul introduce numele și parola pentru a accesa aplicația.  
 **Actori:** Utilizator / Administrator  
 **Pași:**

1. Utilizatorul accesează fereastra de login.
2. Introduce datele de autentificare.
3. Aplicația validează credențialele.
4. Dacă sunt corecte, utilizatorul accesează meniul principal; altfel, apare un mesaj de eroare.

UC2: Afișare produse pe categorie

**Descriere:** Utilizatorul alege o categorie (Bar / Bucătărie) și vizualizează produsele corespunzătoare.  
 **Actori:** Utilizator  
 **Pași:**

1. Se apasă pe butonul categoriei dorite.
2. Aplicația interoghează baza de date.
3. Produsele sunt afișate în tabelul central.

UC3: Adăugare produs nou

**Descriere:** Utilizatorul adaugă un nou produs în stoc.  
 **Actori:** Utilizator  
 **Pași:**

1. Se accesează formularul de adăugare produs.
2. Se completează toate câmpurile obligatorii.
3. Se validează inputul.
4. Produsul este salvat în baza de date și afișat în listă.

UC4: Modificare cantitate produs

**Descriere:** Utilizatorul actualizează cantitatea unui produs.  
 **Actori:** Utilizator  
 **Pași:**

1. Se selectează produsul din listă.
2. Se deschide formularul de modificare.
3. Se introduce cantitatea nouă.
4. Se salvează modificarea în baza de date.

UC5: Alertă stoc critic

**Descriere:** Sistemul verifică automat dacă există produse cu stoc sub limita admisă.  
 **Actori:** Utilizator  
 **Pași:**

1. Utilizatorul apasă butonul de verificare stoc.
2. Sistemul caută produsele cu stoc < 4.
3. Dacă există, se afișează un mesaj de alertă.

### **2.4 Constrângeri**

1. **Tehnice**
   1. Aplicația este compatibilă doar cu sistemul de operare Windows.
   2. Baza de date utilizată este SQL Server.
2. **Funcționale**
   1. Fiecare produs trebuie să fie asociat unui hotel și unei categorii.
   2. Utilizatorii neautentificați nu au acces la funcționalități.
3. **Operaționale**
   1. Aplicația nu trebuie să fie dependentă de conexiune internet.
   2. Interfața trebuie să poată fi utilizată de persoane fără cunoștințe tehnice avansate.

### **2.5 Riscuri**

Această secțiune identifică principalele riscuri asociate dezvoltării și utilizării aplicației **StockMaster**, precum și măsurile de atenuare propuse pentru fiecare categorie.

**1. Riscuri tehnice**  
 Există riscul apariției unor erori în interacțiunea cu baza de date, care pot duce la pierderea datelor sau la rezultate incorecte. Pentru a preveni astfel de situații, aplicația folosește interogări SQL parametrizate și o validare riguroasă a datelor introduse de utilizator.

**2. Riscuri funcționale**  
 Utilizatorii ar putea introduce date incorecte, cum ar fi cantități negative sau text în câmpuri numerice. Aceste riscuri sunt gestionate prin mecanisme de validare în interfață, precum și prin mesaje de eroare clare care ghidează utilizatorul.

**3. Riscuri de securitate**  
 Fără măsuri adecvate, aplicația ar putea fi vulnerabilă la atacuri de tip SQL Injection sau acces neautorizat. Aceste riscuri sunt reduse prin utilizarea interogărilor sigure, autentificare obligatorie și separarea rolurilor (utilizator vs. admin).

**4. Riscuri de utilizare**  
 Un design neintuitiv al interfeței ar putea duce la confuzie din partea utilizatorilor. Pentru a reduce acest risc, aplicația folosește un UI simplu, cu controale vizibile doar în contextul necesar și mesaje de feedback relevante.

**5. Riscuri legate de extindere și mentenanță**  
 Pe măsură ce aplicația evoluează, adăugarea de noi funcționalități ar putea deveni dificilă dacă structura nu este bine organizată. Acest risc este diminuat prin adoptarea arhitecturii în trei straturi (UI, BLL, DAL), care permite modificări izolate și testabile.

## **3. Arhitectura și proiectarea sistemului**

### **3.1 Arhitectura generală**

Aplicația **StockMaster** urmează o arhitectură pe trei straturi (3-layer architecture), o practică consacrată în dezvoltarea de aplicații enterprise. Fiecare strat este responsabil pentru o anumită categorie de sarcini, asigurând separarea clară a responsabilităților, ușurință în mentenanță și posibilitate de extindere.

* **Stratul de prezentare (UI – User Interface):**
  1. Implementat în **Windows Forms** (WinForms).
  2. Forme principale: Form1 (interfața principală), LoginForm, ModificareCantitateForm.
  3. Interacționează cu utilizatorul și trimite cereri către stratul logic.
* **Stratul de logică de business (BLL – Business Logic Layer):**
  1. Implementat prin clasa ProductService.
  2. Gestionează logica de procesare a datelor între UI și DAL.
  3. Nu conține interacțiune directă cu UI sau SQL.
* **Stratul de acces la date (DAL – Data Access Layer):**
  1. Include clasele ProductRepository și Product.
  2. Execută operațiile SQL și oferă metode pentru manipularea tabelelor din baza de date.
  3. Accesează baza de date prin **ADO.NET**, folosind string-ul de conexiune din App.config.

Această arhitectură modulară permite izolarea erorilor, testarea separată a componentelor și refactorizarea eficientă.

### **3.2 Modelul de date**

Baza de date a aplicației este implementată în **SQL Server** și conține următoarele tabele principale:

* **Hoteluri** – conține informații despre hoteluri înregistrate în sistem.
* **Produse** – stochează toate produsele existente, cu atribute precum nume, descriere, cantitate și preț.
* **Categorii** – definește tipurile de produse (ex: Bar, Bucătărie), legate prin ID\_Categorie.

Această secțiune descrie structura bazei de date utilizată de aplicația **StockMaster**. Modelul de date este definit într-o arhitectură relațională, implementată în SQL Server și organizată prin intermediul a patru entități principale:

* Hoteluri
* Produse
* Categorie\_Produse
* Furnizori

### **Relații între entități**

* Fiecare **Produs** aparține unei singure **Categorii** și este furnizat de un singur **Furnizor**.
* Fiecare **Produs** este asociat unui singur **Hotel**.
* Un **Hotel** poate avea mai multe produse.
* O **Categorie** poate fi comună mai multor hoteluri (ex: "Bar", "Bucătărie").

### **Entitățile principale**

#### **Categorie\_Produse**

Definiție:CREATE TABLE Categorie\_Produse (  
 ID\_Categorie INT PRIMARY KEY IDENTITY(1,1),  
 Nume VARCHAR(100) NOT NULL,  
 Descriere TEXT  
);

* Exemple: *Bucătărie*, *Bar*
* Relație: 1 - N cu Produse

#### **Furnizori**

Definiție:

CREATE TABLE Furnizori (  
 ID\_Furnizor INT PRIMARY KEY IDENTITY(1,1),  
 Nume VARCHAR(100) NOT NULL,  
 Persoana\_Contact VARCHAR(100),  
 Telefon VARCHAR(20),  
 Email VARCHAR(100),  
 Adresa TEXT  
);

* Descrie furnizorii de produse, pentru trasabilitate și gestionare aprovizionare.

#### **Hoteluri**

* Coloană ID\_Hotel legată de Produse prin FK
* Conține informații despre hoteluri, administratori, email, username etc.

#### **Produse**

Definiție:

CREATE TABLE Produse (  
 ID\_Produs INT PRIMARY KEY IDENTITY(1,1),  
 Nume VARCHAR(100) NOT NULL,  
 Descriere TEXT,  
 ID\_Categorie INT NOT NULL,  
 ID\_Furnizor INT NOT NULL,  
 Cantitate\_In\_Stoc INT DEFAULT 0,  
 Pret\_Unitar DECIMAL(10,2),  
 Data\_Adaugare DATETIME DEFAULT GETDATE(),  
 FOREIGN KEY (ID\_Categorie) REFERENCES Categorie\_Produse(ID\_Categorie),  
 FOREIGN KEY (ID\_Furnizor) REFERENCES Furnizori(ID\_Furnizor)  
);

Atribute cheie:

* ID\_Categorie → referință la Categorie\_Produse
* ID\_Furnizor → referință la Furnizori
* ID\_Hotel (în implementarea ta curentă din cod)

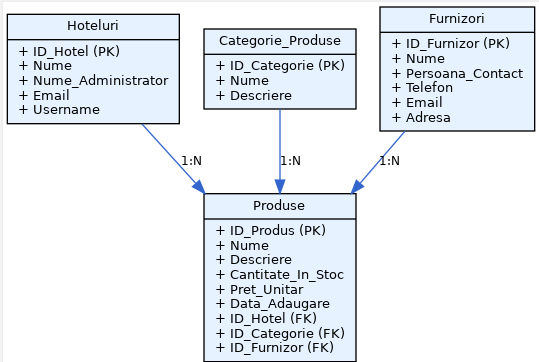


Fig. 1 Diagramă Entitate Relație

## **3.3 Diagrame UML (Unified Modelling Language)**

Această secțiune prezintă principalele diagrame UML utilizate pentru modelarea aplicației **StockMaster**, pentru a evidenția structura, comportamentul și interacțiunile sistemului. Diagramele sunt utile atât în faza de analiză, cât și în etapa de dezvoltare și mentenanță.

### **1. Diagrama de clase**

Diagrama de clase reflectă structura statică a aplicației, evidențiind clasele principale, atributele acestora și relațiile dintre ele. Aplicația este organizată în trei straturi, fiecare având responsabilități bine definite:

#### **Clase principale**

* Product – model de date (entitate persistentă)
* ProductRepository – acces la baza de date (DAL)
* ProductService – logică de business (BLL)
* Form1, LoginForm, ModificareCantitateForm – componente de interfață (UI)

#### **Relații**

* Form1 folosește ProductService
* ProductService utilizează ProductRepository
* ProductRepository gestionează obiecte de tip Product

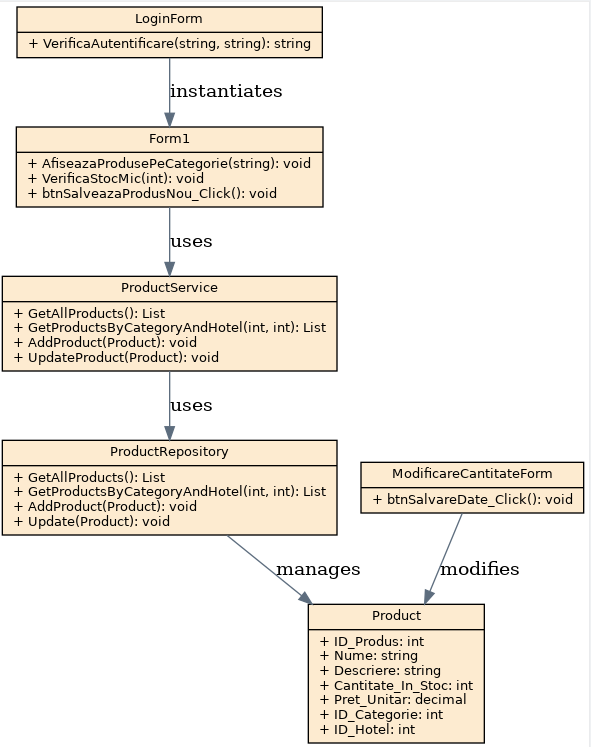


Fig. 2 Diagramă de Clase

### **2. Diagrama de secvență – Autentificare utilizator**

Această diagramă prezintă interacțiunea dintre obiecte în cadrul procesului de autentificare:

#### **Flux:**

1. Utilizatorul introduce username și parolă în LoginForm
2. LoginForm trimite datele către metoda VerificaAutentificare()
3. Se face conexiunea la baza de date
4. Dacă datele sunt valide:
   1. Se obține ID\_Hotel
   2. Se deschide Form1
5. Dacă autentificarea eșuează:
   1. Se afișează un mesaj de eroare

Această diagramă evidențiază succesiunea apelurilor și condițiile de succes/eșec.

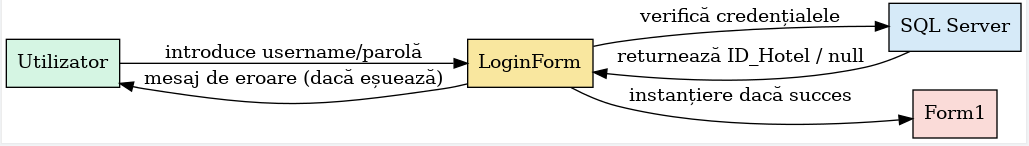


Fig. 3 Diagramă de Autentificare

### **3. Diagrama de stare (opțional)**

Prezintă stările prin care trece un utilizator în timpul utilizării aplicației:

#### **Stări:**

* **Deconectat** → stare inițială
* **Autentificat** → după login reușit
* **Activ** → utilizatorul folosește aplicația
* **Inactiv** → după o perioadă de neutilizare
* **Deconectat** → utilizatorul se întoarce la ecranul de login sau închide aplicația

#### **Tranziții:**

* De la *Deconectat* la *Autentificat* → introducere date corecte
* De la *Activ* la *Inactiv* → lipsă interacțiune
* De la *Inactiv* la *Deconectat* → sesiune încheiată

### **4. Diagrama de activitate**

Ilustrează pașii logici dintr-un proces. Ex: **Adăugare produs nou**

#### **Flux logic:**

1. Utilizatorul apasă „Adaugă”
2. Interfața afișează formularul de completare
3. Se completează câmpurile produsului
4. Se apasă butonul „Salvează”
5. Se validează inputul
6. Se creează obiectul Product
7. Se trimite către ProductService
8. ProductService apelează ProductRepository
9. Produsul este adăugat în baza de date
10. Se afișează mesaj de confirmare

Această diagramă poate ajuta la înțelegerea clară a fluxurilor din aplicație și la identificarea punctelor critice pentru testare.

## **3.4 Tehnologii și stack de dezvoltare**

Această secțiune prezintă tehnologiile și instrumentele utilizate în dezvoltarea aplicației **StockMaster**. Alegerea acestora s-a bazat pe cerințele funcționale, mediul de rulare (desktop Windows) și experiența utilizatorului final.

### **Limbaj de programare:**

* **C#** – limbaj principal de dezvoltare al aplicației, folosit în toate cele trei straturi (UI, BLL, DAL).

### **Framework și platformă:**

* **.NET Framework 4.x** – platforma de execuție utilizată pentru aplicația desktop.
* **Windows Forms (WinForms)** – pentru dezvoltarea interfeței grafice.

### **Baze de date:**

* **Microsoft SQL Server** – SGBD utilizat pentru stocarea datelor despre produse, categorii, hoteluri și furnizori.
* **ADO.NET** – tehnologie de acces la baza de date, utilizată în DAL (Data Access Layer).

### **Structură arhitecturală:**

* **Arhitectură în 3 straturi:**
  + UI (Formulare WinForms)
  + BLL (ProductService)
  + DAL (ProductRepository)

### **Managementul configurațiilor:**

* **App.config** – folosit pentru salvarea stringului de conexiune la baza de date și alte setări.

### **Mediu de dezvoltare:**

* **Visual Studio** – IDE-ul principal utilizat pentru dezvoltare, testare și rulare.

### **Testare:**

* Testare manuală pentru validarea funcționalităților (adăugare, modificare, căutare produse etc.)
* Validări implementate la nivel de UI pentru input-uri incorecte.

## **3.5 Interfața utilizatorului (UI/UX)**

Această secțiune descrie designul interfeței utilizatorului din aplicația **StockMaster**, principiile UI/UX aplicate și modul în care a fost structurată interfața grafică, adaptată specific pentru aplicații desktop Windows dezvoltate cu WinForms.

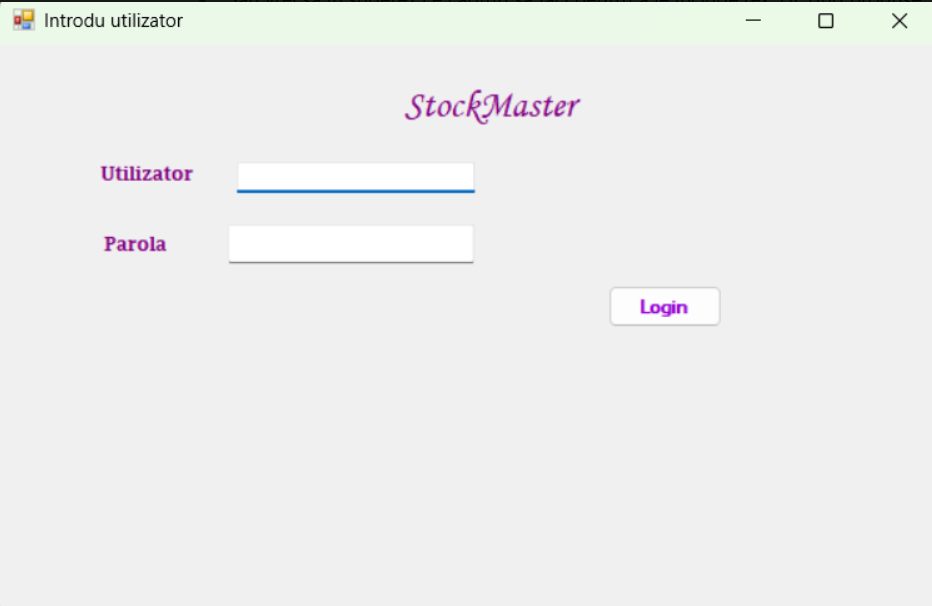
### **1. Principii UI/UX aplicate**

* **Ușurință în utilizare:**  
   Interfața este concepută intuitiv, cu un număr redus de pași pentru fiecare acțiune. Butoanele sunt clar etichetate, iar utilizatorul este ghidat printr-un flux logic de lucru.
* **Claritate și simplitate:**  
   Elementele vizibile în UI se activează doar când sunt necesare (ex: butoane de gestionare produse vizibile doar după alegerea unei categorii).
* **Feedback vizual:**  
   Aplicația oferă mesaje informative și de eroare în mod clar, folosind ferestre MessageBox pentru confirmări, avertismente sau validări.
* **Design funcțional (non-responsive):**  
   Fiind o aplicație desktop, layout-ul este fix, conceput pentru ecrane de laptop/desktop, fără adaptabilitate mobilă.
* **Accesibilitate de bază:**  
   Suport pentru tastatură (tab navigation) și elemente UI standard WinForms, cu contrast vizual acceptabil.

### **2. Structura interfeței**

Aplicația este compusă din mai multe formulare WinForms, fiecare cu un scop clar:

1. **Pagina de autentificare (LoginForm):**
   * Câmpuri pentru username și parolă.
   * Rol special pentru administrator care deblochează funcționalități suplimentare (adăugare hoteluri).
   * Feedback vizual în caz de autentificare eșuată.

Figura 3.5.1 – Formular de autentificare

Această imagine ilustrează interfața LoginForm, unde utilizatorul introduce datele de conectare. În cazul introducerii unui cont de tip **admin**, interfața se extinde și permite gestionarea hotelurilor (adăugare, ștergere, etc.).  
 Dacă autentificarea eșuează, se afișează un mesaj de eroare prin MessageBox.

1. **Dashboard principal (Form1):**
   * Meniu cu butoane pentru categorii de produse: *Bar*, *Bucătărie*.
   * Afișare a produselor în format tabelar (DataGridView).
   * Butoane pentru căutare, adăugare produs, modificare stoc și verificare stocuri critice.

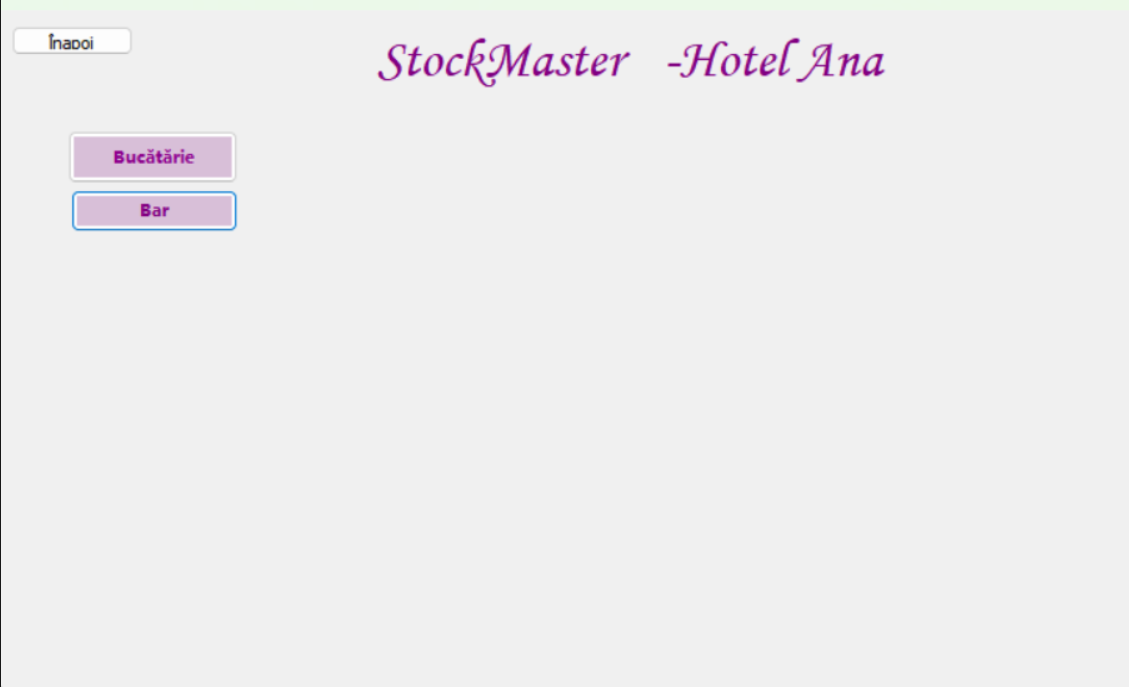


Figura 3.5.2 – Dashboard cu selecție de categorii

După autentificare, utilizatorul ajunge în Form1, unde poate alege categoria de produse (Bar sau Bucătărie). Butoanele sunt simple și clare, activând acțiunile specifice gestionării produselor.

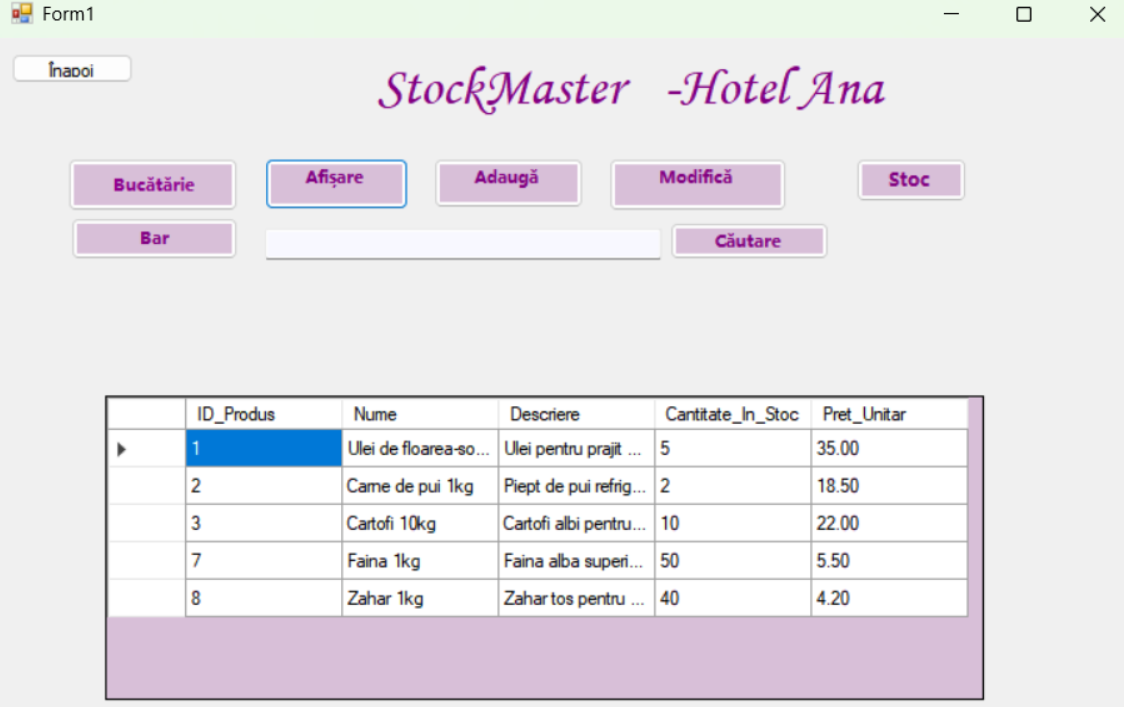


Figura 3.5.3 – Afișare produse într-un DataGridView

Produsele sunt afișate într-un tabel (dataGridProduse) care permite vizualizarea și selectarea elementelor. Coloanele afișează numele, descrierea, cantitatea în stoc și prețul unitar. Selectarea unei linii permite accesul la funcția de modificare a cantității.

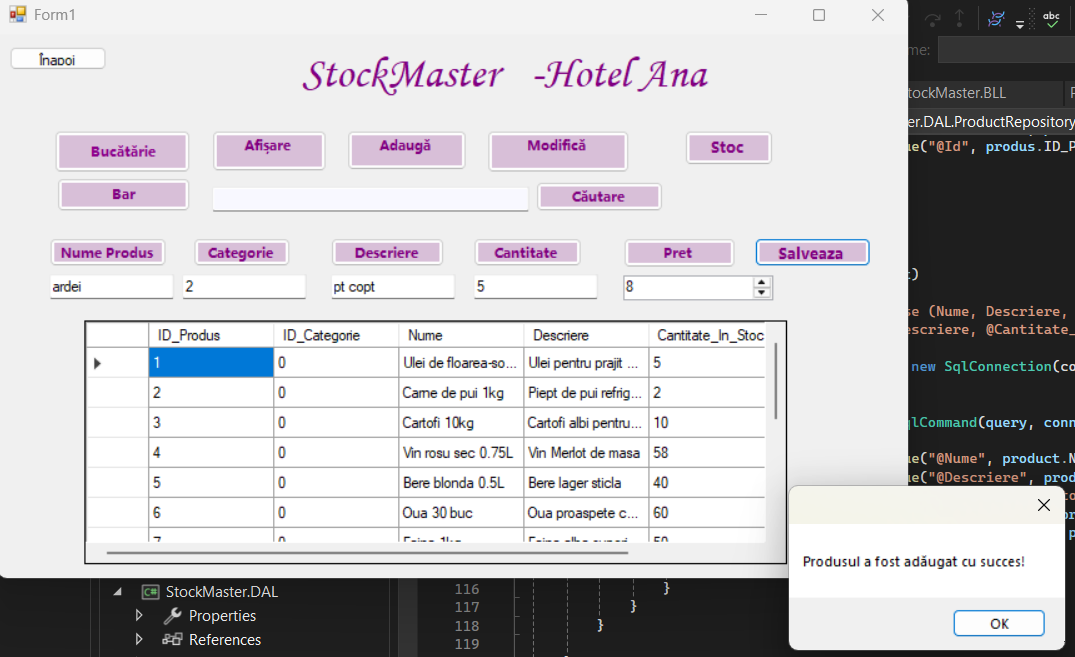


Figura 3.5.4 – Formular adăugare produs nou

Când utilizatorul apasă „Adaugă produs”, sunt afișate câmpurile pentru completare: nume, descriere, cantitate, preț și categoria produsului. Butonul „Salvează produs” validează câmpurile și adaugă produsul în baza de date.

1. **Formular de modificare cantitate (ModificareCantitateForm):**
   * Interfață simplă care permite introducerea unei noi cantități pentru produsul selectat.

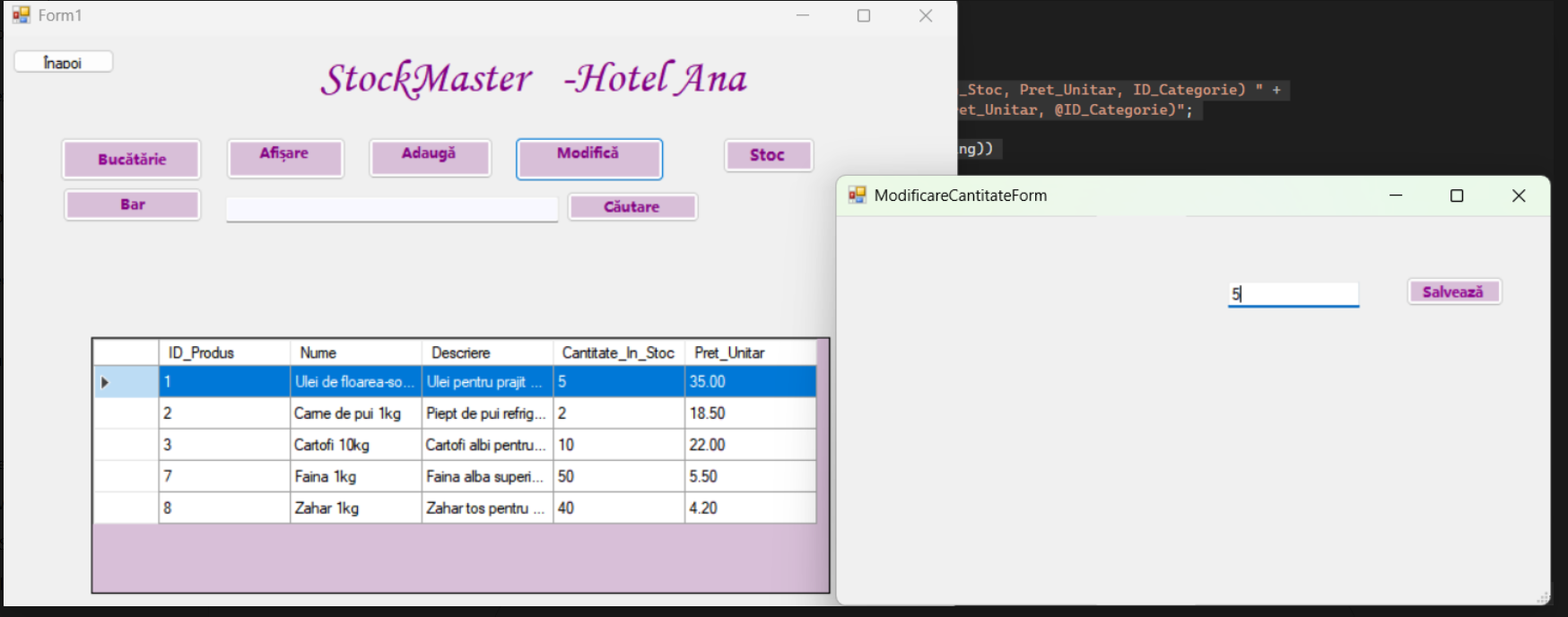


Figura 3.5.5 – Formular modificare cantitate

Formularul ModificareCantitateForm permite actualizarea rapidă a stocului unui produs. Se validează ca valoarea introdusă să fie numerică și pozitivă.

1. **Formular de adăugare hotel (pentru admin):**
   * Afișat doar dacă autentificarea se face ca administrator.
   * Permite completarea datelor unui hotel nou: nume, admin, email, username și parolă.



Figura 3.5.6– Formular adăugare hotel

1. **Notificări și alerte:**
   * Sistem de mesaje prin MessageBox.Show() pentru acțiuni precum:
     + “Produsul a fost adăugat cu succes”
     + “Stoc critic: X produse sub 4 bucăți”
     + “Te rog să completezi toate câmpurile”

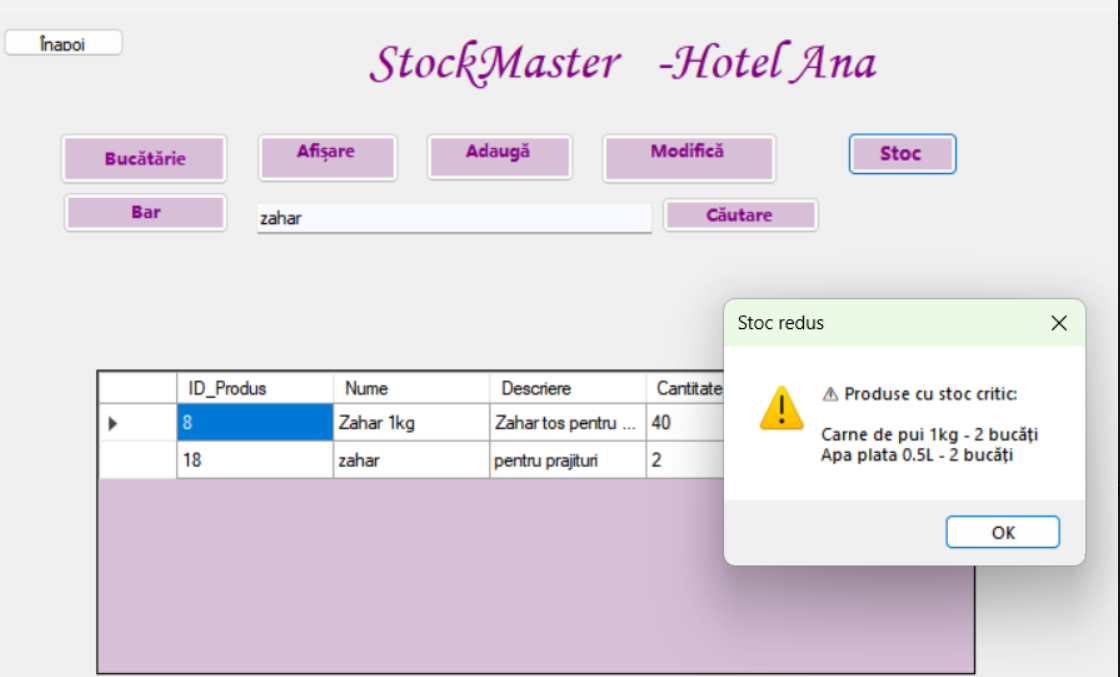


Figura 3.5.7 – Alertă pentru stoc critic

### **3. Protocoale și tehnologii UI**

* **Tehnologie utilizată:**
  + **Windows Forms (WinForms)** – framework-ul principal pentru interfața grafică.
  + **.NET Framework 4.x** – pentru compatibilitate și rulare pe sisteme Windows.
* **Librării suplimentare:**
  + Nu se folosesc biblioteci externe – toate controalele sunt native WinForms.
* **Configurabilitate:**
  + Elemente UI dinamice (buton, textbox etc.) devin vizibile/ascunse în funcție de selecțiile utilizatorului.

## **4. Implementare**

### **4.1 Structura codului**

Codul aplicației este organizat pe baza unei arhitecturi modulare de tip **3 straturi (3-layer architecture)**, urmând principiile separării responsabilităților și dezvoltării scalabile.

Structura principală este următoarea:

### **Aplicație desktop (.NET Framework – WinForms)**

#### **StockMaster (Frontend / UI Layer – Interfața utilizatorului)**

* Form1.cs – formular principal cu funcționalități de afișare, adăugare, căutare, modificare produse și alerte stoc.
* LoginForm.cs – autentificare utilizatori și administratori; controlează accesul și vizibilitatea funcțiilor.
* ModificareCantitateForm.cs – formular secundar pentru modificarea cantității unui produs selectat.
* Program.cs – punctul de intrare în aplicație.
* App.config – configurare pentru stringul de conexiune SQL (OptiStocConnectionString).

### **Business Logic Layer (BLL)**

#### **StockMaster.BLL**

* ProductService.cs – clasele de business care:
  + centralizează logica aplicației
  + intermediază între UI și DAL
  + validează datele înainte de salvare

### **Data Access Layer (DAL)**

#### **StockMaster.DAL**

* Product.cs – clasa model pentru entitatea produs (cu atributele din baza de date)
* ProductRepository.cs – clasele de acces la baza de date:
  + execută SELECT, INSERT, UPDATE
  + lucrează cu SqlConnection, SqlCommand, SqlDataReader

### **Baza de date (SQL Server)**

#### **Database/Tables**

* Produse, Categorie\_Produse, Hoteluri, Furnizori
* Structura normalizată, cu chei primare și străine
* Date legate între tabele prin ID\_Hotel, ID\_Categorie, ID\_Furnizor

### **Configurare și resurse**

* App.config: conține detalii de conexiune la SQL Server
* Resources: imagini, icon-uri pentru butoane (dacă sunt utilizate)

### **4.2 Module și componente**

Această secțiune descrie modulele software ale aplicației și funcționalitatea fiecăruia, organizate conform arhitecturii pe 3 straturi.

#### **1. Modul de autentificare**

* Gestiunea autentificării se face în LoginForm
* Verifică utilizatorul în baza de date (prin interogare SQL)
* Rolul „admin” deblochează funcții suplimentare: adăugare hotel, configurare

#### **2. Modul de gestionare a produselor**

* Afișează produsele din baza de date în funcție de categoria selectată
* Permite adăugarea unui produs nou (cu validări)
* Include căutare după nume, modificare cantitate și alertă stoc redus
* Interacțiunea se face din Form1 și ProductService

#### **3. Modul de modificare a stocului**

* Deschis prin selectarea unui produs din listă
* Formular separat (ModificareCantitateForm)
* Permite modificarea cantității și actualizarea bazei de date
* Validări pentru input numeric, pozitiv

#### **4. Modul de administrare hoteluri (doar pentru admin)**

* Adminul poate introduce un hotel nou în sistem
* Formularele se activează doar dacă autentificarea este făcută cu admin/admin123
* Datele se salvează în tabela Hoteluri (nume, email, username, parolă)

#### **5. Modul de alertare pentru stoc critic**

* Analizează toate produsele pentru un hotel
* Verifică dacă Cantitate\_In\_Stoc < 4
* Generează o alertă vizuală cu lista produselor critice (prin MessageBox)
* Este o funcție de tip suport, dar esențială pentru gestionare eficientă

## **4.3 Fluxul de date**

Această secțiune descrie modul în care datele circulă prin diferitele componente ale aplicației **StockMaster**, de la introducerea lor de către utilizator prin interfața grafică, până la procesarea și salvarea acestora în baza de date SQL Server.

### **1. Fluxul general al datelor**

1. **Utilizatorul interacționează cu interfața aplicației**, care este construită în **Windows Forms**. Aici introduce informații despre produse, cantitate, categorie etc.
2. **Datele sunt validate local** în interfață (ex: verificări pentru câmpuri goale, valori numerice pozitive).
3. **Datele valide sunt transmise** către stratul de business logic (**BLL**), unde pot fi prelucrate suplimentar (filtrare, rutare).
4. BLL apelează metode din stratul de acces la date (**DAL**), care construiește și execută comenzi SQL.
5. **DAL interacționează cu baza de date** SQL Server pentru a salva, actualiza sau extrage date.
6. **Rezultatele sunt returnate** în sens invers (DAL → BLL → UI) și sunt afișate în interfață (ex: în DataGridView).

### **2. Procesarea datelor**

* **Validare:**
  + Câmpurile introduse de utilizator (nume, cantitate, preț etc.) sunt verificate în UI înainte de a fi procesate.
  + Se validează că numerele sunt întregi și pozitive, iar câmpurile text nu sunt goale.
  + În cazul unor date invalide, aplicația afișează mesaje de eroare (ex: “Cantitatea trebuie să fie un număr pozitiv.”).
* **Autentificare și autorizare:**
  + La pornirea aplicației, utilizatorul se autentifică în LoginForm.
  + Se verifică dacă utilizatorul este admin sau utilizator simplu.
  + Doar administratorul are acces la funcționalitățile de creare hotel.
* **Stocare:**
  + Datele despre produse sunt salvate în tabela Produse, folosind comenzi SQL executate prin ADO.NET (INSERT, UPDATE, SELECT).
  + Se folosesc SqlCommand și SqlParameter pentru siguranță împotriva SQL Injection.
* **Recuperare date:**
  + Produsele sunt afișate în interfața principală (Form1) pe baza interogărilor SQL (ex: produse dintr-o anumită categorie și hotel).
  + Datele sunt încărcate într-un DataGridView pentru a fi ușor de vizualizat și selectat.

### **3. Flux de date pentru acțiuni specifice**

#### **1.** **Adăugare produs nou:**

* Utilizatorul completează formularul.
* Se creează un obiect Product cu datele introduse.
* Se trimite către ProductService.AddProduct(), apoi către ProductRepository.AddProduct().
* Produsul este salvat în baza de date și afișat în tabelul de produse.

#### **2.** **Modificare cantitate produs:**

* Se selectează un produs din listă.
* Se deschide ModificareCantitateForm cu cantitatea actuală.
* Utilizatorul introduce noua cantitate.
* Se actualizează baza de date prin Update() în DAL.

#### **3.****Căutare produs:**

* Utilizatorul introduce text într-un câmp de căutare.
* Produsele sunt filtrate local (din listă deja încărcată) sau printr-o nouă interogare SQL.

#### **4.****Alerte de stoc critic:**

* Se rulează o interogare SQL care caută produse cu Cantitate\_In\_Stoc < 4.
* Lista este afișată utilizatorului într-un mesaj MessageBox.

### **4. Notificări și evenimente**

* Aplicația afișează **feedback vizual instant**:
  + Mesaje informative: "Produsul a fost adăugat cu succes"
  + Mesaje de eroare: "Te rog completează toate câmpurile"
  + Alerte: " Produse cu stoc critic"
* Nu sunt implementate notificări prin e-mail, SMS sau loguri externe, deoarece aplicația este destinată utilizării interne, offline.
* Tot feedbackul este oferit prin controale clasice WinForms (Label, MessageBox, TextBox, Buttons).

## **5. Testare**

### **5.1 Strategia de testare**

Strategia de testare a aplicației **StockMaster** se bazează în principal pe **testare manuală funcțională**, adaptată specificului unei aplicații desktop dezvoltate în C# cu WinForms. Având în vedere lipsa unor API-uri expuse sau a unui sistem de automatizare CI/CD, testarea se concentrează pe validarea interfeței, a fluxurilor logice și a corectitudinii operațiilor de acces la baza de date.

Testarea se desfășoară în mai multe etape:

### **1. Testare unitară (manuală)**

* Fiecare componentă este testată individual, în izolare, prin scenarii simple:
  + Testarea funcției AddProduct() cu date valide/invalide
  + Testarea funcției UpdateProduct() pentru modificarea cantității
  + Verificarea autentificării cu utilizatori corecți și incorecți

Exemple:

* Se adaugă un produs nou și se verifică prezența acestuia în DataGridView
* Se introduce o cantitate invalidă (ex: literă în loc de număr) și se validează mesajul de eroare

### **2. Testare de integrare**

* Se verifică dacă modulele interacționează corect:
  + UI ↔ BLL (ProductService)
  + BLL ↔ DAL (ProductRepository)
  + DAL ↔ SQL Server (prin ADO.NET)

Exemple:

* Se adaugă un produs din UI, care este procesat în BLL și salvat în SQL
* Se caută un produs și se verifică dacă filtrarea este corectă

### **️ 3. Testare de performanță (manuală, observabilă)**

* Se monitorizează timpii de răspuns în următoarele cazuri:
  + Afișarea unei liste mari de produse (ex: 1000 înregistrări)
  + Navigarea între ferestre
  + Răspunsul aplicației la comenzi rapide (adăugare/modificare)

Nu s-au folosit instrumente automate precum JMeter, dar aplicația răspunde fluent la comenzi locale și baze de date medii.

### **4. Testare de securitate (de bază)**

* Aplicația folosește interogări SQL parametrizate (SqlParameter), reducând riscul de **SQL Injection**.
* Nu se păstrează parole în clar în interfață.
* Accesul la funcționalități avansate este restricționat pentru contul „admin”.

Exemple:

* S-a încercat introducerea unui string SQL malițios în câmpurile de login – aplicația nu execută comenzi neautorizate.
* Parola este mascată în UI (PasswordChar = '\*')

### **5. Testare de acceptanță**

* Aplicația a fost testată din perspectiva utilizatorului final (manager hotel).
* Scenariile testate reflectă cerințele funcționale: login, adăugare produs, căutare, alertă stoc etc.
* Feedback-ul a fost integrat pentru îmbunătățirea vizibilității controalelor și mesajelor de eroare.

Exemplu:

* Când un câmp rămâne necompletat la adăugare produs, apare mesajul „Te rog completează toate câmpurile!” – validare funcțională reușită.

### **6. Automatizarea testelor**

* Nu a fost implementată o suită automată de teste (ex: MSTest, NUnit), deoarece aplicația este monolit desktop și nu rulează în pipeline CI/CD.
* Pentru extindere viitoare, se poate adăuga un proiect de testare unit test cu MSTest sau xUnit pentru metodele din ProductService.

## **5.2 Tipuri de teste**

Pentru asigurarea calității aplicației **StockMaster**, au fost aplicate mai multe tipuri de teste, adaptate la structura și modul de funcționare al aplicației desktop. Testele au fost efectuate manual, pe baza cerințelor funcționale și a scenariilor reale de utilizare.

### **1. Teste unitare (manuale)**

Scopul testelor unitare este verificarea funcționării corecte a metodelor individuale din BLL și DAL:

* ProductService.AddProduct() – verifică dacă un obiect Product este transmis corect spre salvare
* ProductRepository.Update() – actualizează cantitatea unui produs în baza de date
* VerificaAutentificare() din LoginForm – returnează admin sau ID\_Hotel valid/invalid în funcție de credențiale

Exemple de verificări:

* Se creează un produs valid → se salvează fără erori
* Se trimite un produs fără cantitate → se generează mesaj de eroare

### **2. Teste de integrare**

Teste care validează interacțiunea dintre componentele aplicației:

* UI ↔ BLL: Form1 apelează metode din ProductService
* BLL ↔ DAL: ProductService interacționează corect cu ProductRepository
* DAL ↔ Bază de date: comenzi SQL executate cu succes, fără excepții

Scenarii testate:

* Adăugare produs din interfață → verificare în baza de date
* Afișare produse după categorie → confirmare filtrare corectă
* Autentificare și redirecționare către Form1

### **3. Teste de validare și UI**

Verificări directe în interfață asupra comportamentului vizual:

* Validare câmpuri goale la adăugare produs
* Restricționare acces funcții admin pentru utilizatori obișnuiți
* Verificare afișare mesaj de succes/eroare în funcție de rezultat

Exemple:

* Introducere „abc” în câmp de cantitate → eroare: „Cantitatea trebuie să fie un număr întreg valid.”
* Nu se poate salva produs fără denumire

### **4. Teste de performanță (vizuale)**

Testare empirică asupra comportamentului aplicației în condiții normale și cu volum mai mare de date (până la 1000 produse):

* Afișarea listelor este instantanee
* Timp de deschidere aplicație sub 2 secunde
* Timp de execuție pentru salvare produs: <1 sec.

### **5. Teste de securitate (de bază)**

Testare de intruziune și validare prin constrângeri logice:

* Se încearcă introducerea de comenzi SQL în câmpurile de login
* Se verifică dacă parolele sunt afișate în clar – acestea sunt mascate
* Interogările către baza de date folosesc SqlParameter → prevenție SQL Injection

### **6. Teste de acceptanță**

Evaluarea aplicației conform cerințelor și scenariilor reale:

* Adăugare produs → vizualizare în listă → validare completă
* Alertă stoc critic → se declanșează când există produse < 4 bucăți
* Modificare cantitate → valoarea nouă apare în listă după salvare

## **5.3 Plan de testare**

Această secțiune definește strategia practică de testare a aplicației **StockMaster**, incluzând scenarii specifice pentru verificarea funcționalităților cheie, a validărilor și a interacțiunilor între module. Toate testele sunt realizate manual, în mediul de dezvoltare și execuție local.

### **1. Obiectivele testării**

* Asigurarea corectitudinii funcționalităților esențiale (login, adăugare, modificare, alertă stoc).
* Verificarea comunicării corecte între UI, BLL și DAL.
* Validarea comportamentului aplicației în fața inputului greșit sau malițios.
* Confirmarea faptului că doar utilizatorii autorizați accesează funcții sensibile (ex: adăugare hotel).
* Evaluarea vizuală a performanței aplicației în condiții normale de utilizare.

### **2. Scenarii de testare**

#### **Testarea autentificării**

**Scenariu:** Un utilizator încearcă să se autentifice cu credențiale valide.  
 **Pași:**

1. Lansarea aplicației → se deschide LoginForm
2. Introducere username: admin, parolă: admin123
3. Apăsare pe butonul „Login”
4. Aplicația identifică utilizatorul ca administrator și redirecționează spre Form1
5. Se activează funcțiile de administrare hotel

**Rezultat așteptat:** Acces complet pentru admin, fără erori.

#### **Testarea adăugării unui produs**

**Scenariu:** Un utilizator adaugă un produs nou în stocul hotelului.  
 **Pași:**

1. Accesarea categoriei (ex: Bucătărie) din Form1
2. Apăsarea butonului „Adaugă produs”
3. Completarea tuturor câmpurilor: nume, descriere, cantitate, preț, categorie
4. Apăsarea butonului „Salvează produs nou”

**Rezultat așteptat:** Produsul este adăugat în baza de date și apare în listă.

#### **Testarea modificării stocului**

**Scenariu:** Un utilizator modifică cantitatea unui produs.  
 **Pași:**

1. Selectarea unui produs din DataGridView
2. Apăsarea pe „Modifică cantitate”
3. Se deschide ModificareCantitateForm
4. Se introduce noua cantitate și se apasă „Salvează”

**Rezultat așteptat:** Cantitatea este actualizată în baza de date.

#### **Testarea alertei de stoc critic**

**Scenariu:** Utilizatorul verifică dacă există produse cu stoc scăzut.  
 **Pași:**

1. Apăsarea butonului „Stoc critic”
2. Aplicația verifică în baza de date dacă Cantitate\_In\_Stoc < 4
3. Dacă există produse în această stare → se afișează lista într-un mesaj

**Rezultat așteptat:** Apare o alertă cu produsele care au stoc critic.

#### **Testarea căutării produselor**

**Scenariu:** Căutarea unui produs după nume.  
 **Pași:**

1. Se introduce textul „Apă” în câmpul de căutare
2. Se apasă butonul „Caută”
3. DataGridView afișează doar produsele care conțin cuvântul „Apă”

**Rezultat așteptat:** Filtrarea funcționează corect și rapid.

#### **Testarea securității inputului**

**Scenariu:** Introducerea unui string periculos în câmpul de autentificare.  
 **Pași:**

1. În câmpul username se scrie: ' OR 1=1 --
2. Se apasă pe „Login”

**Rezultat așteptat:** Sistemul respinge datele. Interogările SQL fiind parametrizate, aplicația nu este vulnerabilă la SQL Injection.

### **3. Instrumente utilizate pentru testare**

Deoarece aplicația este desktop, testarea s-a realizat **manual**, fără framework-uri automate.

* **Visual Studio Debugger** – pentru identificarea erorilor la runtime
* **SQL Server Management Studio** – pentru verificarea datelor din tabele
* **MessageBox** și alte controale WinForms – pentru feedback vizual
* **Testare empatică (UX)** – comportamentul aplicației observat direct de utilizator

## **6. Deployment și mentenanță**

### **6.1 Strategie de deployment**

Procesul de implementare al aplicației **StockMaster** utilizează o abordare simplificată, specifică aplicațiilor desktop Windows, fără infrastructură cloud, CI/CD sau containere. Deployment-ul se realizează **manual** pe stațiile de lucru ale utilizatorilor finali, în cadrul fiecărui hotel.

### **1. Medii de implementare**

* **Mediu de dezvoltare (Development)**  
   Proiectul este dezvoltat și testat local în Visual Studio, folosind baza de date OptiStoc pe instanță locală de SQL Server.
* **Mediu de testare (Local testing)**  
   Aplicația este rulată pe stații diferite pentru a valida compatibilitatea și stabilitatea cu diverse versiuni de Windows.
* **Mediu de producție (Production)**  
   Aplicația este copiată și rulată pe stațiile de lucru din cadrul hotelurilor. Baza de date poate fi locală sau pe rețea, în funcție de infrastructura fiecărui punct de lucru.

### **2. Procesul de implementare**

* **Pas 1: Build aplicație**  
   Aplicația este compilată în Visual Studio și se generează fișierul StockMaster.exe.
* **Pas 2: Configurare conexiune**  
   Fișierul App.config este modificat pentru a include conexiunea la baza de date locală sau de rețea:

<connectionStrings>  
 <add name="OptiStocConnectionString" connectionString="Data Source=.;Initial Catalog=OptiStoc;Integrated Security=True" />  
</connectionStrings>

* **Pas 3: Distribuție fișiere**  
   Folderul bin\Release (sau Publish, dacă se folosește opțiunea „Publish”) este copiat pe stația destinată.
* **Pas 4: Instalare SQL Server + scripturi**  
   Se instalează SQL Server Express sau versiunea standard și se rulează scripturile de creare a bazei de date (Categorie\_Produse, Produse, Hoteluri, Furnizori).
* **Pas 5: Testare inițială**  
   Se verifică dacă aplicația se lansează fără erori, dacă se poate face autentificare și dacă interogările funcționează.

### **3. Tehnologii utilizate în deployment**

|  |  |
| --- | --- |
| **Componentă** | **Tehnologie folosită** |
| Limbaj programare | C# (.NET Framework 4.x) |
| Interfață grafică | Windows Forms (WinForms) |
| Bază de date | SQL Server 2019 / Express |
| Conectivitate BD | ADO.NET |
| IDE | Visual Studio 2022+ |
| Configurare conexiune | App.config |
| Distribuție aplicație | Manual (copiere fișiere .exe) |
| Sisteme țintă | Windows 10 / 11 |

### 

### **Observație**

Deoarece aplicația este destinată **rulării locale și offline**, nu se folosesc:

* Sisteme de versionare distribuite (ex: GitHub)
* Pipeline-uri CI/CD (ex: Jenkins, GitHub Actions)
* Instrumente DevOps moderne (Docker, Kubernetes, Prometheus)

### **6.2 Administrarea și monitorizarea**

Această secțiune descrie strategiile și instrumentele utilizate pentru administrarea și monitorizarea aplicației **StockMaster**, în scopul asigurării performanței, securității și funcționării corecte în timp. Deoarece aplicația este o soluție **desktop locală**, monitorizarea se realizează **manual** și **prin instrumente standard Windows și SQL Server**.

### **1. Monitorizarea performanței**

Aplicația fiind instalată local pe stații individuale, nu folosește un sistem de monitorizare centralizat. Totuși, următoarele metode pot fi folosite pentru supravegherea funcționării:

* **Task Manager (Windows):**
  + Verificarea consumului de memorie și CPU al aplicației StockMaster.exe
  + Observarea eventualelor blocaje sau încărcări anormale
* **SQL Server Management Studio (SSMS):**
  + Monitorizarea rulării interogărilor și a performanței bazei de date
  + Verificarea conexiunilor active și blocajelor SQL
* **Observație directă în UI:**
  + Întârzieri la afișarea listelor sau salvarea datelor pot indica probleme de performanță
  + Timpul de reacție este evaluat vizual, deoarece aplicația nu folosește tool-uri de profiling

### **2. Monitorizarea securității**

Securitatea aplicației este tratată la un nivel de bază, prin bune practici implementate în cod:

* **Validări locale în UI:**
  + Se interzice introducerea de date invalide (ex: stringuri în câmpuri numerice)
* **Interogări SQL parametrizate (SqlParameter):**
  + Se previn atacuri de tip SQL Injection
* **Autentificare separată pe roluri:**
  + Doar utilizatorul „admin” are acces la gestionarea hotelurilor
  + Parolele nu sunt vizibile și sunt protejate prin PasswordChar
* **Acces la aplicație restricționat fizic:**
  + Fiind aplicație desktop, nu este expusă pe internet

### **3. Administrarea aplicației**

Administrarea se face direct de pe calculatorul unde este instalată aplicația:

* **Instalare și configurare manuală:**
  + Aplicația se copiază în format .exe împreună cu fișierele necesare
  + App.config este modificat pentru conexiunea la baza de date
* **Bază de date administrată prin SSMS:**
  + Modificarea datelor, backup și restaurare se fac manual de către un administrator
* **Debugging și întreținere în Visual Studio:**
  + În cazul unor erori, aplicația poate fi deschisă în modul debug pentru depanare rapidă

### **4. Notificări și alerte**

Aplicația **nu utilizează notificări automate** (email, Slack etc.). Toate mesajele sunt gestionate local, prin UI:

* **Mesaje vizuale (MessageBox.Show()):**
  + Confirmări (ex: „Produsul a fost adăugat cu succes”)
  + Erori (ex: „Te rog completează toate câmpurile”)
  + Alerte (ex: „Produse cu stoc critic”)
* **Comunicare manuală cu dezvoltatorul:**
  + Problemele sunt raportate verbal sau prin email către persoana care întreține aplicația

Deși nu folosește un ecosistem de monitorizare modern, **StockMaster** rămâne ușor de supravegheat și întreținut, fiind proiectat pentru uz local, cu interfață clară și fluxuri stabile.

### **6.3 Plan de mentenanță**

Planul de mentenanță asigură funcționarea optimă și actualizarea continuă a aplicației **StockMaster**, incluzând intervenții periodice pentru remedierea problemelor, optimizarea performanței și extinderea funcționalităților în funcție de nevoi. Mentenanța este realizată manual, local, de către dezvoltator sau o persoană tehnică din echipa beneficiarului.

### **1. Tipuri de mentenanță**

* **Mentenanță corectivă**  
   Remedierea erorilor raportate de utilizatori (ex: blocaje, validări nefuncționale, probleme de salvare în baza de date). Se face direct în Visual Studio, prin depanare și recompilare.
* **Mentenanță preventivă**  
   Verificarea periodică a bazei de date (consistență, relații, indexuri), testarea interfeței și observarea eventualelor scăderi de performanță. Poate include optimizarea interogărilor SQL.
* **Mentenanță evolutivă**  
   Adăugarea de noi funcționalități conform cerințelor utilizatorilor (ex: export în Excel, rapoarte, gestionare utilizatori multipli, filtrare avansată).
* **Mentenanță adaptivă**  
   Adaptarea aplicației la modificări ale mediului (ex: mutarea pe un nou server SQL, actualizare Windows, trecerea la o versiune nouă de .NET Framework).

### **2. Program de mentenanță recomandat**

|  |  |
| --- | --- |
| **Frecvență** | **Activitate** |
| **Zilnic** | Testare funcțională: pornirea aplicației, verificare login și salvare date |
| **Săptămânal** | Backup manual al bazei de date SQL Server (.bak) |
| **Lunar** | Verificarea și testarea interfeței, eventuale mici actualizări |
| **Trimestrial** | Curățare tabele inutile, optimizare performanță, feedback de la utilizatori |
| **Anual** | Evaluarea arhitecturii și planificarea de upgrade sau re-tehnologizare |

### **3. Instrumente și procese de mentenanță**

* **Visual Studio** – pentru corecții în cod, compilare și re-deploy local
* **SSMS (SQL Server Management Studio)** – pentru gestionarea bazei de date, backup/restore
* **Fișiere partajate / Google Drive** – pentru partajarea versiunilor actualizate ale .exe și scripturilor SQL
* **Email sau Excel** – pentru urmărirea problemelor raportate de utilizatori
* **Backup manual periodic** – export .bak pentru protecție împotriva pierderii de date

### **4. Suport tehnic**

* **Asistență locală (on demand)** – oferită de dezvoltator sau persoană tehnică desemnată
* **Documentație locală (fișier PDF / Word)** – cu instrucțiuni pentru instalare, resetare date, utilizare funcții
* **Comunicare directă** – raportare prin email sau telefon, fără sistem de ticketing

## **7. Concluzii**

Această secțiune oferă o sinteză a principalelor concluzii rezultate în urma analizei, proiectării și implementării aplicației **StockMaster**, precum și recomandări pentru extinderi și optimizări ulterioare.

### **1. Rezumat al principalelor constatări**

* Aplicația **StockMaster** a fost proiectată și implementată utilizând o **arhitectură pe 3 straturi (UI, BLL, DAL)**, care oferă modularitate, claritate și separare a responsabilităților.
* Cerințele **funcționale** (autentificare, gestionare produse, alerte stoc, validări) și **non-funcționale** (performanță locală, simplitate, securitate de bază) au fost definite și implementate corect, asigurând o experiență stabilă utilizatorului.
* S-a utilizat un stack tehnologic simplu, dar eficient:
  + **C# și Windows Forms** pentru interfața grafică
  + **ADO.NET** pentru acces la date
  + **SQL Server** pentru stocarea datelor
* Aplicația a fost testată extensiv manual, acoperind scenarii de:
  + funcționare normală
  + validare a inputului
  + securitate de bază (SQL Injection)
  + alertare și feedback în UI
* Administrarea și distribuția aplicației se pot realiza fără dependențe complexe, ceea ce o face potrivită pentru implementare în medii mici și mijlocii (ex: recepții de hotel, unități de catering, spații de gestiune).

### **2. Recomandări pentru dezvoltări viitoare**

* **Optimizarea performanței:**  
   Adăugarea unui sistem simplu de cache local pentru produsele frecvent accesate și optimizarea interogărilor SQL pentru seturi mari de date.
* **Extinderea funcționalităților:**
  + Exportul listelor de produse în format PDF sau Excel
  + Modul de **raportare lunară** a consumului
  + Integrarea cu imprimante de etichete sau coduri de bare
* **Îmbunătățirea experienței utilizatorilor:**
  + Modernizarea UI-ului cu controale custom sau migrarea către WPF pentru o interfață mai atractivă
  + Oferirea de sugestii automate la căutare (autocomplete)
* **Consolidarea securității:**
  + Criptarea parolelor în baza de date
  + Adăugarea unui sistem simplu de jurnalizare a acțiunilor importante (loguri locale)
* **Automatizarea mentenanței:**
  + Un mic modul intern pentru backup/restaurare bazei de date
  + Adăugarea unei secțiuni de configurare direct în interfață (fără editarea App.config)

Proiectul **StockMaster** reprezintă un exemplu solid de aplicație desktop de gestiune, ușor de folosit, clar organizată și pregătită pentru extinderi viitoare în funcție de nevoile reale ale beneficiarului.