

PROIECT DE DIPLOMĂ

Îndrumător proiect,

Ș.I. Dr. Ing. Cristina Elena ANTON

Absolvent,

Gabriel-Răzvan SORA

Galăț

2021

PROGRAMUL DE STUDII: Calculatoare



APLICAȚIE PENTRU GESTIUNEA UNUI MAGAZIN DE ÎNCĂLȚĂMINTE

Îndrumător proiect,

Ș.I. Dr. Ing. Cristina Elena ANTON

Absolvent,

Gabriel-Răzvan SORA

Galați

2021




DECLARAȚIE

Subsemnatul, Gabriel-Răzvan Sora absolvent al Facultății de Automatică, Calculatoare, Inginerie Electrică și Electronică, din cadrul Universității “Dunărea de Jos” din Galați, promoția 2021, programul de studii Calculatoare, declar pe proprie răspundere că proiectul de diplomă cu titlul „Gestiunea unui magazin de încălțăminte” este elaborat de mine și nu a mai fost prezentat niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusive cele de pe Internet, sunt indicate în proiect, cu respectarea regulilor de evitare a plagiatului.

“Plagiatul: însușirea ideilor, metodelor, procedurilor, tehnologiilor, rezultatelor sau textelor unei persoane, indiferent de calea prin care acestea au fost obținute, prezentându-le drept creație proprie.”

Am luat la cunoștință că prezentarea unui proiect plagiat va conduce la anularea diplomei de licență.

Lucrarea conține un număr de 92 pagini.

Data: 5.07.2021 Semnătura 



Rezumat

Lucrarea prezentată este soluția propusă pentru o gestiune rapidă a unui magazin de încălțăminte cu scopul de a reduce costurile suportate de antreprenor pentru menținerea afacerii.

Soluția propusă este formată dintr-o aplicație simplă, rapidă și accesibilă atât pentru administratori, cât și pentru casieri. Astfel, se va economisi personal, timp prețios și nu în ultimul rând, bani, care vor putea fi folosiți pentru dezvoltarea magazinului și a aplicației. Pentru a satisface nevoile antreprenorului, aplicația propusă dispune de mai multe funcționalități necesare unui magazin de încălțăminte, o parte din acestea fiind adăugarea, vânzarea și vizualizarea produselor, adăugarea și vizualizarea angajaților.

Soluția oferită este dezvoltată folosind mediul de dezvoltare Microsoft Visual Studio 2019 Community, care dispune de o multitudine de librării și instrumente ce fac dezvoltarea aplicației rapidă și eficientă. De asemenea, la dezvoltarea aplicației a fost utilizat .NET Framework, care oferă beneficii în ceea ce privește portabilitatea aplicației pe diferite sisteme hardware.

Securitatea și siguranța datelor este foarte importantă, iar din această cauză aplicația a fost dezvoltată folosind metode de criptare a parolei în ceea ce privește autentificarea utilizatorilor. De asemenea, în timpul dezvoltării s-a ținut cont de posibilele fraude care pot apărea din cauza introducerii eronată a datelor în baza de date, fapt ce a dus la limitarea funcționalităților pentru unii utilizatori.

Design-ul aplicației este unul simplu, dar cu o temă futuristică pentru a oferi utilizatorilor o experiență plăcută și unică, ce asigură aplicației încrederea de care este nevoie pentru o gestiune rapidă a magazinului. Totodată, aplicația este foarte intuitivă prin folosirea unor iconițe reprezentative funcționalităților, făcând astfel și mai simplă utilizarea acestora.



Cuprins

1. Introducere	6
1.1 Breviar	6
1.2 Premisă	6
1.3 Soluție	6
1.4 Implementare	7
2. Tehnologii folosite.....	8
2.1 .NET Framework.....	8
2.2 Limbajul de programare C#.....	9
2.3 MySQL.....	10
2.4 Microsoft Visual Studio Community 2019	11
2.5 XAMPP.....	11
2.6 MySQL Workbench	11
3. Funcționalitățile aplicației.....	13
3.1 Modul casier.....	13
3.2 Modul administrator	15
4. Implementarea funcționalităților	16
4.1 Modelarea bazei de date	16
4.2 Algoritmi pentru implementarea funcționalităților	18
4.2.1 Operațiile CRUD (Create, Read, Update, Delete).....	19
4.2.1.1 Vizualizarea datelor.....	19
4.2.1.2 Adăugarea datelor	20
4.2.1.3 Actualizarea datelor	21
4.2.1.4 Stergerea datelor	22
4.2.2 Alte metode	22
4.2.2.1 Metoda „EmailValidate”	22
4.2.2.2 Metoda „Log”	23
4.2.2.3 Metoda „CriptareParola”	24
5. Scanarea produselor	26
5.1 Codul QR (Quick Response)	26
5.2 Zxing.....	29
5.3 AForge.....	30
5.4 Motivația alegerii codului QR.....	30
6. Design.....	31
6.1 User Experience Design.....	31



6.1.1	Accesibilitate	31
6.1.2	Ușurința completării datelor	32
6.1.3	Ușurința parcurgerii informației	32
6.2	User Interface Design	33
6.2.1	Font-uri	34
6.2.2	Culori	34
7.	Securitate	35
7.1	Confidențialitate	35
7.2	Integritatea datelor	35
7.3	Disponibilitatea datelor	36
7.4	Autentificarea	36
8.	Feedback	38
8.1	Îmbunătățirea serviciilor	38
8.2	Păstrarea clienților	39
9.	Chestionarul	40
10.	Statistici	41
11.	Manual de utilizare	42
11.1	Fereastra de autentificare	42
11.2	Recuperarea parolei	43
11.3	Meniu	44
12.	Concluzii	45
13.	Bibliografie și referințe	46
	Bibliografie	46
	Referințe	46
Anexa 1	47
Anexa 2	75
Anexa 3	84



1. Introducere

1.1 Breviar

Scopul acestei lucrări este prezentarea unei aplicații ce ajută la o bună gestiune a unui magazin de încălțăminte. În acest document sunt incluse motivele pentru implementarea acestei soluții și descrierea conceptului. De asemenea, în lucrare este prezentat în detaliu modul de funcționare al aplicației și funcționalitățile acesteia. Pentru implementarea acestui concept și pentru a oferi o securitate sporită a informațiilor cu caracter sensibil, a fost nevoie de o bază de date, iar pentru crearea acesteia s-a optat pentru folosirea utilităților **MySQL WorkBench** și **XAMPP**.

1.2 Premisă

Multe magazine au o problemă în ceea ce privește gestiunea digitalizată a datelor, acest lucru ducând astfel la o creștere a nevoii de personal și a investiției necesare. Motivul principal al acestei probleme este reprezentat de complexitatea aplicațiilor dedicate pentru digitalizare. Mulți antreprenori evită folosirea acestora, rămânând astfel, la păstrarea datelor în documente fizice ce necesită o atenție sporită în ceea ce privește menținerea acestora în siguranță fiind mai sensibile și mai ușor de distrus .

1.3 Soluție

Cunoscând această problemă și motivele acesteia, soluția propusă va fi reprezentată de o aplicație care poate ține cu ușurință evidența angajaților, a produselor și nu în ultimul rând, a vânzărilor, rezultând o bună digitalizare a informațiilor, o securitate sporită și o reducere de personal, economisind astfel sume importante de bani care vor ajuta afacerea beneficiarului să-și mărească cifra de afaceri. De asemenea, pentru o bună utilizare a



aplicației, aceasta va conține o interfață grafică prietenoasă și intuitivă pentru a fi ușor de utilizat și pentru a se deosebi cu ușurință de aplicațiile deja existente pentru acest domeniu. Design-ul aplicației va trebui să pună accent pe principalele nevoi ale antreprenorului cum ar fi introducerea și vizualizarea datelor.

1.4 Implementare

Această aplicație poate avea o aplicabilitate diversificată, nu doar pentru magazinele de încălțăminte, deoarece îmbină elemente din mai multe domenii, precum procesarea datelor, identificare, interfața cu utilizatorul și multe altele. Pe parcursul acestei lucrări se va face referire la o implementare specifică și nume la gestiunea unui magazin de încălțăminte.

Aplicația este formată din trei componente, mai exact din aplicația ce conține interfața grafică, baza de date și o cameră cu care se vor scana produsele pentru realizarea vânzării. Conceptul aplicației se bazează pe folosirea unor conturi de utilizator cu ajutorul cărora se va realiza autentificarea și afișarea meniului principal, care va conține funcționalitățile specifice și disponibile tipului de utilizator. Vor exista două tipuri de utilizator, administrator și casier, aceștia având drepturi diferite în ceea ce privește accesul la funcționalitățile aplicației. Baza de date folosită în cadrul dezvoltării acestei aplicații va face posibilă digitalizarea și păstrarea datelor în siguranță, acestea putând fi mai apoi prelucrate și afișate. Camera va fi folosită pentru a realiza vânzarea de produse prin scanarea codului QR atașat, urmând ca acestea să fie căutate în baza de date și adăugate în coș.



2. Tehnologii folosite

2.1 .NET Framework

.NET Framework sau DotNet Framework reprezintă o platformă de dezvoltare software care permite programelor să ruleze independent față de sistemul hardware, mai exact, utilizatorul poate să ruleze același program, compilat inițial pe un echipament hardware pentru platforma .NET Framework, pe mai multe echipamente hardware diferite cu condiția ca sistemul de operare care rulează să fie unul ce conține această platformă.

.NET Framework unește mai multe tehnologii (ASP, XML, OOP, SOAP, WDSL, UDDI) și limbaje de programare (VB, C++, C#, J#), asigurând atât portabilitatea codului compilat între diferite calculatoare cu sistem de operare Windows, cât și reutilizarea codului în programe indiferent de limbajul de programare utilizat. Fiind ultima interfață între aplicațiile .NET și sistemul de operare, această platformă stă la baza tehnologiei .NET și este constituită din 3 componente principale.

- Framework Class Library (FCL), care reprezintă bibliotecile necesare pentru realizarea aplicațiilor de tip Desktop sau Web.
- Common Language Runtime (CLR), care reprezintă o platformă de executare a programelor.
- Limbajele de programare, care necesită respectarea specificațiilor de Programare Orientată pe Obiecte, având ca elemente de bază clase, interfețe, polimorfisme, tratarea excepțiilor, tipuri valoare și referință, și mecanisme de moștenire.

Figura următoare reprezintă arhitectura platformei .NET Framework.

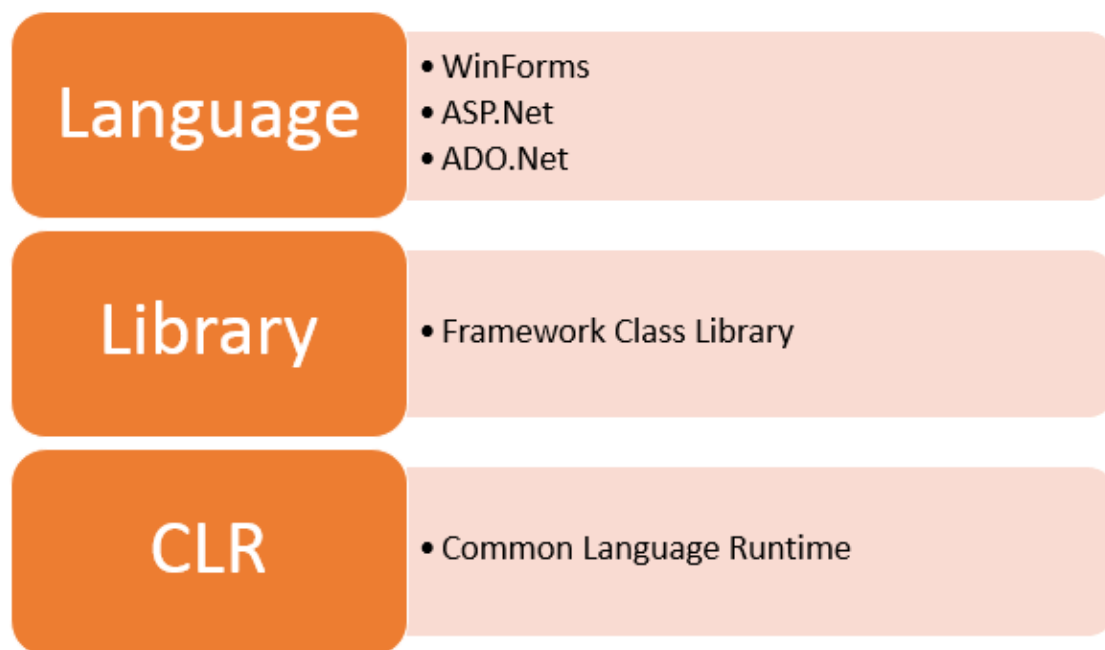


Fig. 1.1 Arhitectura .NET Framework

2.2 Limbajul de programare C#

Limbajul de programare C# este un limbaj compilat, orientat pe obiecte, utilizat în special cu platforma .NET Framework. S-a optat pentru acest limbaj deoarece are o sintaxă mai prietenoasă, preluând o mare parte din sintaxa limbajelor C și C++, oferind o multitudine de librării ce ajută la o dezvoltare mai rapidă și mai ușoară a aplicațiilor.

Librăriile folosite pentru dezvoltarea aplicației prezentate sunt:

- System
- System.Collections.Generic
- System.Data
- System.Drawing
- System.Drawing.Imaging
- System.IO
- System.Linq
- System.Text
- System.Threading.Tasks
- System.Windows.Forms



- System.Security.Cryptography
- System.Net
- System.Net.Mail
- MySql.Data.MySqlClient
- AForge.Video
- AForge.Video.DirectShow
- ZXing
- QRCoder

2.3 MySQL

MySQL este un sistem de gestiune a bazelor de date relaționale, dezvoltat de compania suedeză MySQL AB și distribuit sub Licența Publică Generală GNU. Acest sistem folosește limbajul SQL (Structured Query Language) cu ajutorul căruia se pot manipula, gestiona și primi date prin intermediul interogărilor. Figura următoare reprezintă arhitectura MySQL.

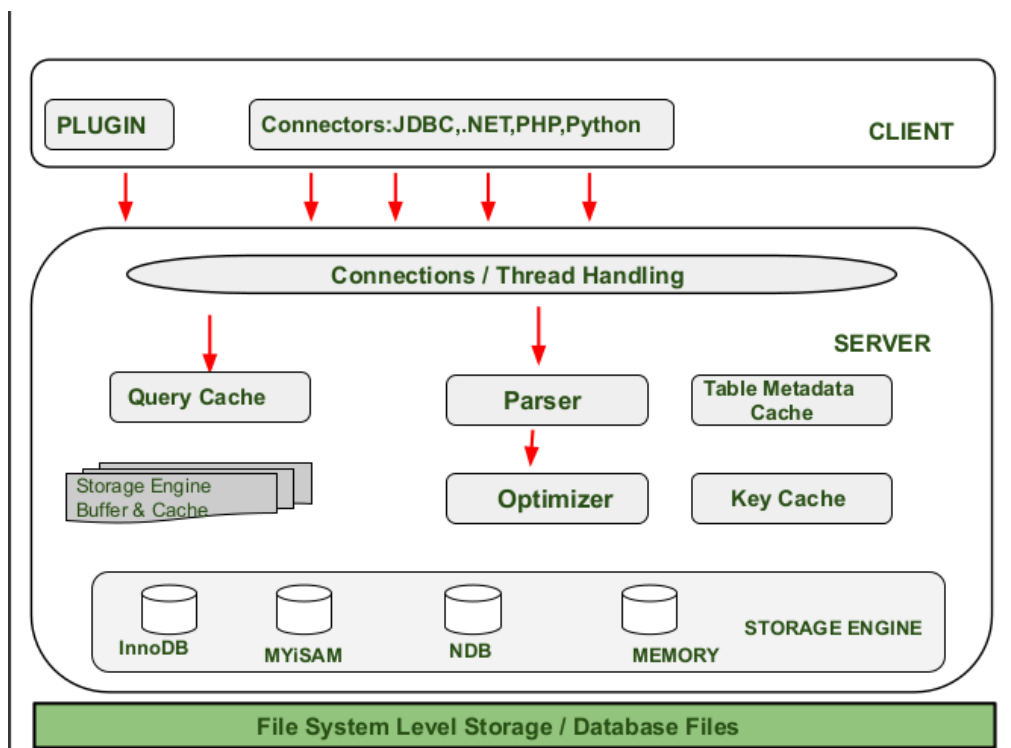


Fig. 1.2 - Arhitectura MySQL



2.4 Microsoft Visual Studio Community 2019

Microsoft Visual Studio Community 2019 este un mediu de dezvoltare integrat (IDE) dezvoltat de Microsoft și include un set complet de instrumente de dezvoltare ce ajută la realizarea de aplicații desktop, aplicații mobile și aplicații web. Acest IDE permite utilizarea limbajului de programare C#, are o interfață foarte predictibilă și oferă gratuitate utilizatorilor din mediul academic, ceea ce face folosirea acestuia preferabilă în vederea evitării practicilor imorale precum pirateria.

2.5 XAMPP

XAMPP este un pachet de programe software gratuite, dezvoltat de Apache Friends în 2002 și este constituit din următoarele componente:

- Server Apache HTTP
- Bază de date MariaDB
- Interpretor pentru script-uri PHP

Acest utilitar este lansat sub termenii licenței GNU și acționează ca un server web capabil de a servi pagini dinamice. XAMPP este disponibil pentru sistemele de operare Windows, Linux, Solaris și Mac OS X, și este utilizat în principal pentru dezvoltarea proiectelor web.

2.6 MySQL Workbench

MySQL Workbench este un instrument de proiectare a bazelor de date vizuale, care integrează dezvoltarea SQL, administrarea, proiectarea bazelor de date, crearea și întreținerea într-un singur mediu de dezvoltare integrat pentru sistemul de baze de date MySQL. Figura următoare reprezintă interfața grafică a acestui instrument.

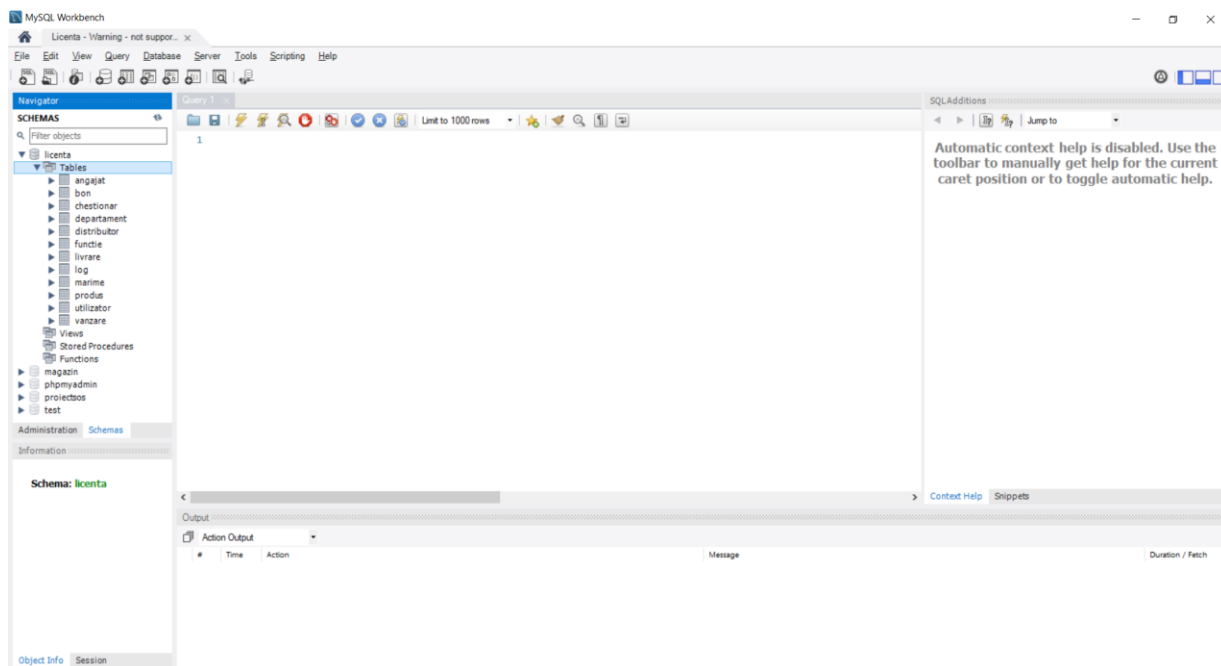


Fig. 1.3 - Interfața MySQL Workbench



3. Funcționalitățile aplicației

Aplicația acoperă o arie mare de funcționalități necesare pentru gestiunea unui magazin de încălțăminte, unele dintre acestea având un acces restrâns datorită impactului pe care îl pot avea. Prin urmare aplicația conține două moduri la care utilizatorii au acces în urma autentificării, acestea fiind modul casier și modul administrator.

3.1 Modul casier

Scopul acestui mod este, în principal, acela de a reduce accesul utilizatorilor de acest tip la funcționalități cu un impact major cum ar fi adăugarea angajaților în baza de date, funcționalități la care ar trebui să aibă acces un personal redus pentru evitarea fraudelor. Totuși, acest mod nu reduce accesul la funcționalitățile principale ale aplicației, permițând astfel acestor utilizatori să contribuie la o gestiune eficientă și rapidă a magazinului în condiții de siguranță. Diagrama următoare reprezintă parcursul ecranelor propus pentru modul casier.

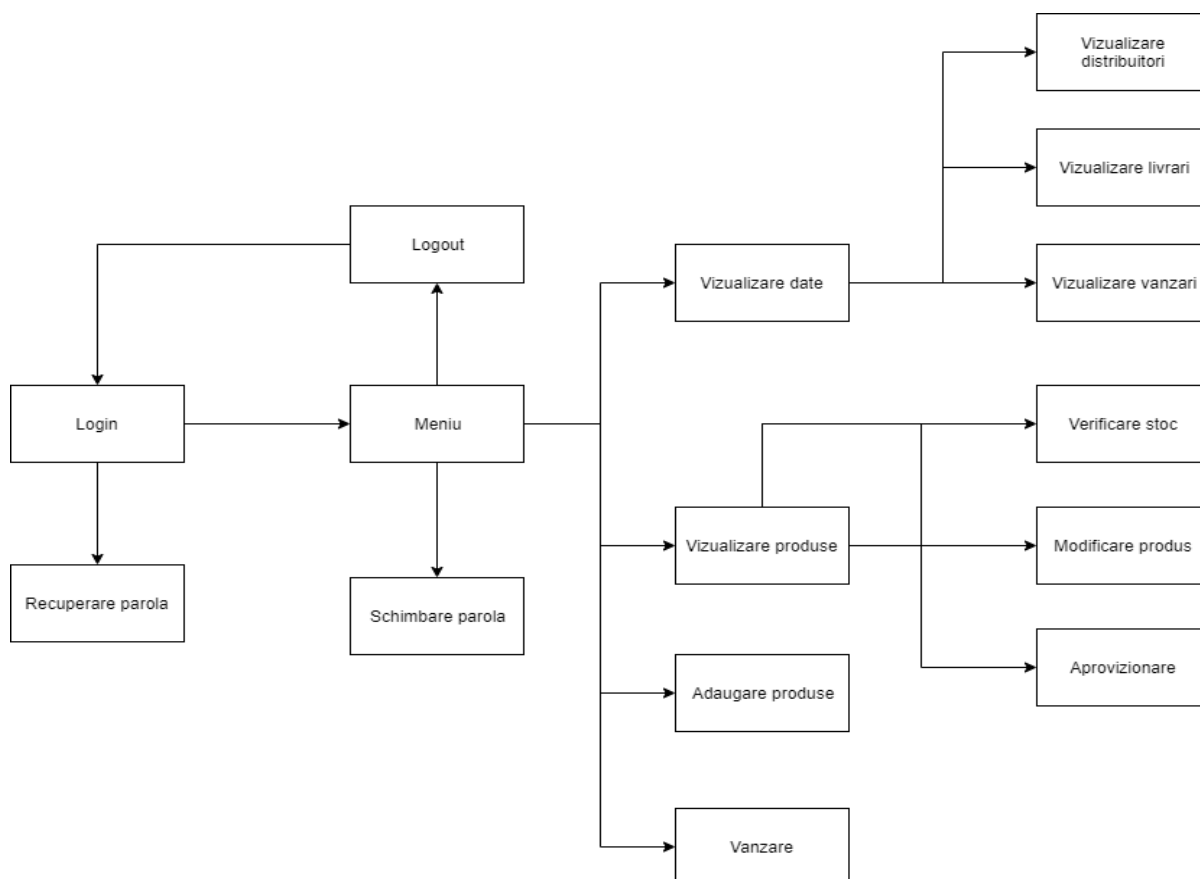


Fig. 2.1 - Parcurs mod Casier

Înainte de autentificare utilizatorul are acces la funcționalitate de recuperare a parolei, în cazul în care acesta a uitat-o.

În urma autentificării realizată în pagina principală și anume pagina de autentificare (Login), utilizatorul are acces la următoarele funcționalități:

- Vizualizarea de date, cu cele 3 subcategorii specifice acestui mod, vizualizare de distribuitori, livrări și vânzări
- Vizualizarea de produse, cu cele 3 funcționalități, verificarea stocului, modificarea produsului și aprovizionare
- Adăugarea de produse
- Vânzarea de produse
- Schimbarea parolei
- Deconectare, care va duce utilizatorul înapoi la pagina principală

3.2 Modul administrator

Scopul acestui mod este, în principal, acela de a oferi acestui tip de utilizatori acces deplin asupra aplicației. Aceștia vor avea acces atât la funcționalitățile disponibile pentru modul casier cât și la restul funcționalităților pentru introducere și vizualizare de date. Diagrama următoare reprezintă parcursul ecranelor propus pentru modul administrator.

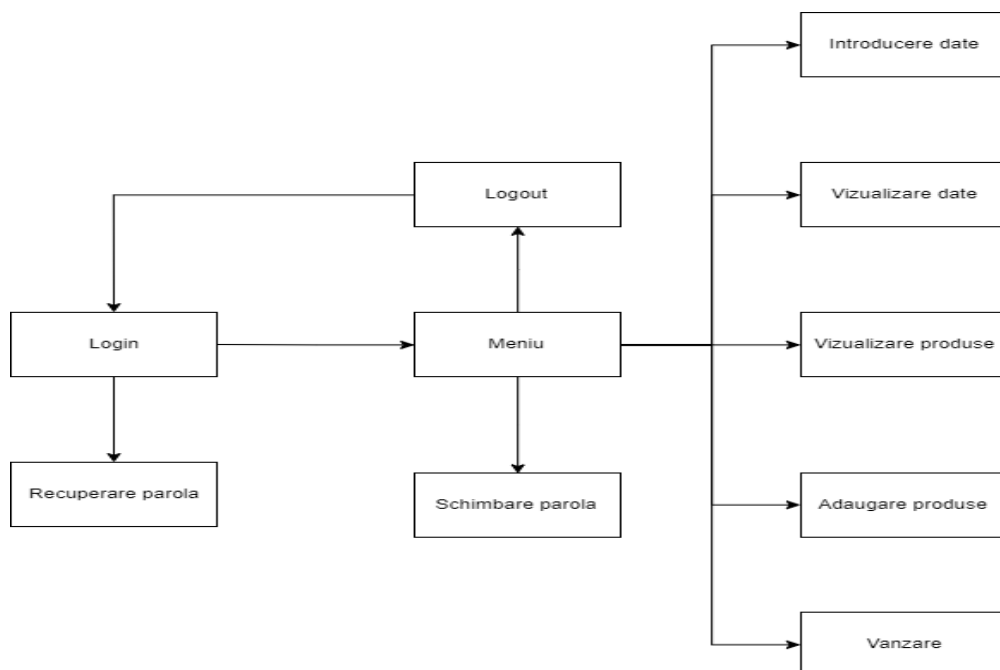


Fig. 2.2 - Parcurs mod Administrator

În urma autentificării realizată în pagina principală, utilizatorul are acces la următoarele funcționalități:

- Introducerea de date, cu cele 5 subcategorii specifice, introducerea departamentelor, funcțiilor, angajaților, distribuitorilor și a utilizatorilor
- Vizualizare de date, cu cele 9 subcategorii, vizualizarea de funcții, departamente, angajați, distribuitori, utilizatori, vânzări, livrări, sugestii și nu în ultimul rând, vizualizarea istoricului
- Adăugarea de produse
- Vânzarea de produse
- Schimbarea parolei
- Deconectare

4. Implementarea funcționalităților

4.1 Modelarea bazei de date

Pentru ca aplicația să funcționeze corect și să beneficieze de o securitate crescută în ceea ce privește protecția datelor, a fost nevoie de realizarea unei baze de date. Așadar s-a încercat proiectarea unei baze de date respectând regulile de integritate teoretice și ținând cont de formele normale. Pentru acest lucru s-a realizat mai întâi modelul conceptual al datelor (MCD), model care definește tabelele ce vor fi utilizate în baza de date, dar și ce legături relaționale există între acestea.

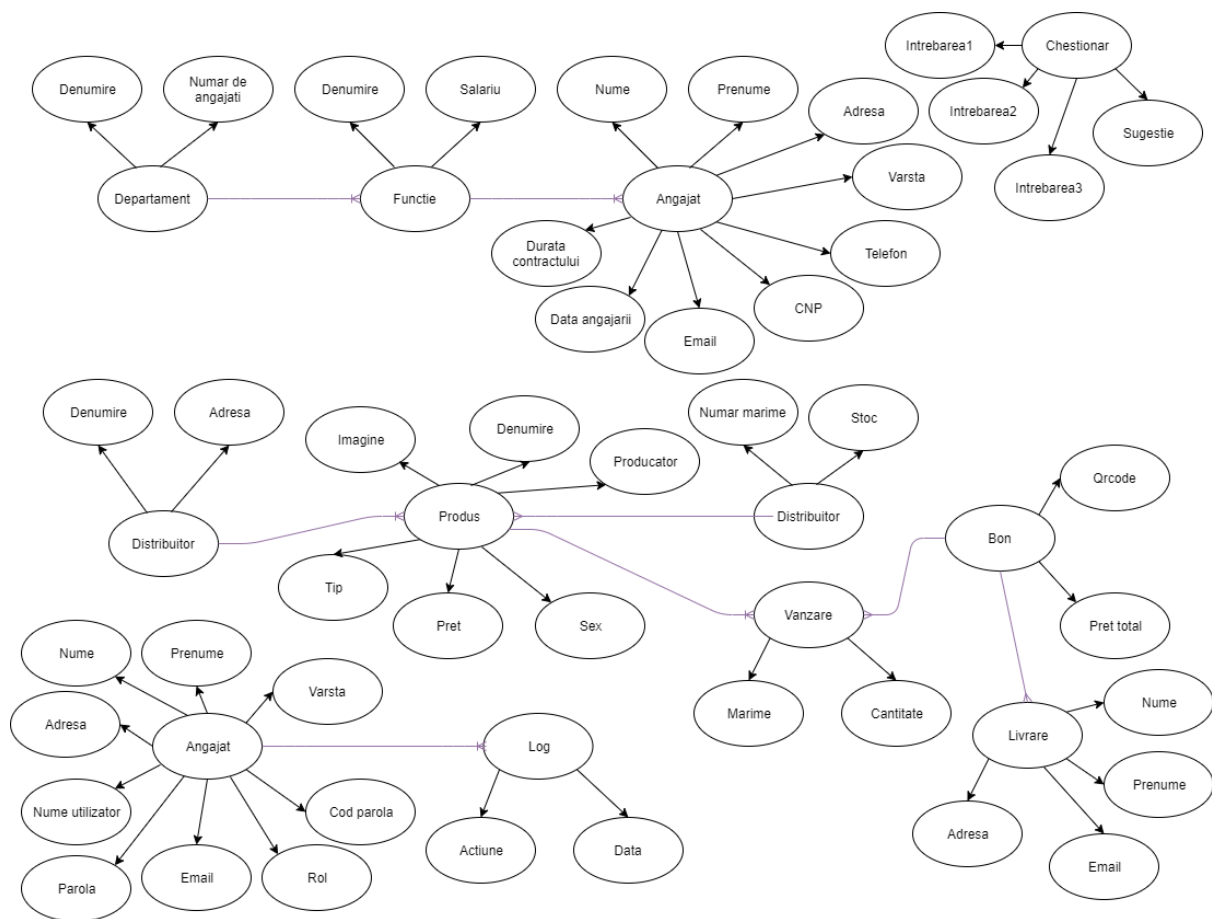


Fig. 3.1 - Modelul Conceptual al Datelor



În urma analizei în detaliu a modelului conceptual al datelor, a rezultat modelul logic al datelor (MLD). Acesta din urmă definește cum trebuie implementată baza de date pentru o bună funcționare având în componență mai multe detalii despre tabelele ce vor fi adăugate. În continuare vor fi prezentate tabelele bazei de date împreună cu cheile primare și secundare sub forma NUME_TABEL[**cheie_primară**, coloana1, coloana2, **cheie_secundară** (dacă există)].

DEPARTAMENT[**Id_departament**, denumire, număr_angajați]

FUNCȚIE[**Id_funcție**, denumire, salariu, **id_departament**]

ANGAJAT[**Id_angajat**, nume, prenume, adresă, vârstă, telefon, cnp, email, data_angajării, durată_contract, **id_funcție**]

DISTRIBUITOR[**Id_distribuitor**, denumire, adresă]

PRODUS[**Id_produs**, imagine, denumire, producător, tip, sex, preț, **id_distribuitor**, qrcode]

MĂRIME[**Id_mărime**, nr_mărime, stoc, **id_produs**]

BON[**Id_bon**, preț_total, qrcode]

VÂNZARE[**Id_vânzare**, **id_produs**, mărime, cantitate, **id_bon**]

LIVRARE[**Id_livrare**, nume, prenume, adresa, email, **id_bon**]

UTILIZATOR[**Id_utilizator**, nume, prenume, adresa, vârstă, email, nume_utilizator, parolă, rol, cod_parolă]

LOG[**Id_log**, acțiune, data, **id_utilizator**]

CHESTIONAR[**Id_chestionar**, întrebarea1, întrebarea2, întrebarea3, sugestie]

Având la dispoziție atât modelul conceptual al datelor cât și modelul logic al datelor, s-a putut realiza baza de date cu ușurință respectând astfel formele normale și regulile de integritate. Figura următoare reprezintă diagrama entitate-relație (ERD) sau modelul fizic al datelor (MFD).

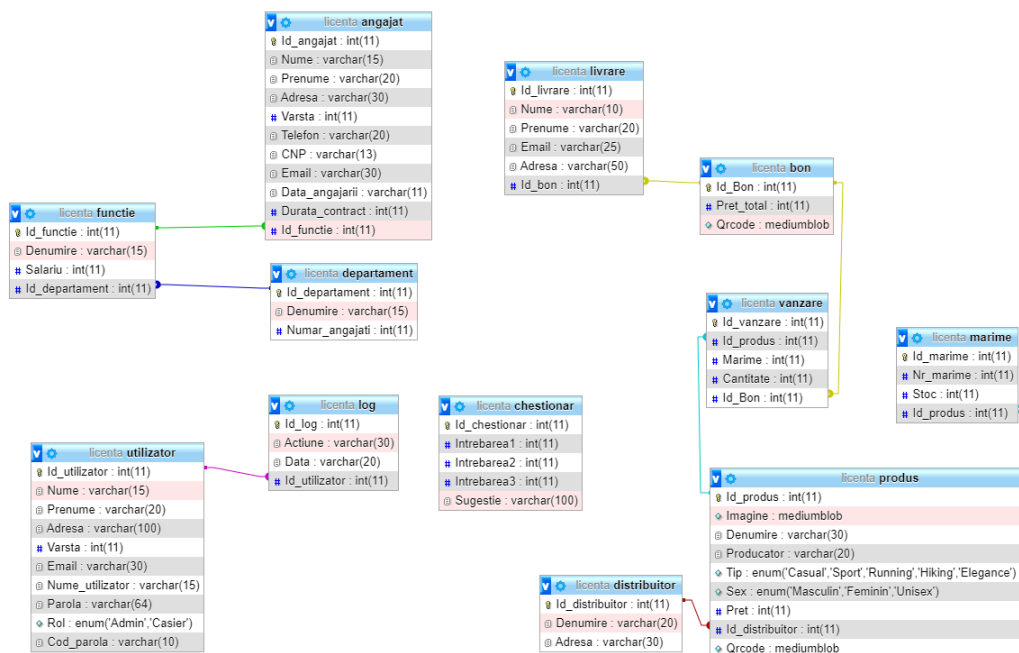


Fig. 3.2 - Modelul Fizic al Datelor

Modelul fizic al datelor conține toate detaliile disponibile legate de baza de date proiectată, mai exact, care sunt cheile primare și secundare, și ce tip de date are fiecare coloană a tabelului.

4.2 Algoritmi pentru implementarea funcționalităților

Pentru a avea o aplicație completă și care să funcționeze corect din punct de vedere al gestionării magazinului a fost nevoie de implementarea unor algoritmi de creare, adăugare, modificare și ștergere a instanțelor în baza de date și alți algoritmi pentru interogarea și preluarea informațiilor din aceasta. Algoritmii au fost implementați cu ajutorul librăriei `MySQL.Data.MySqlConnection` și clasele specifice acesteia.



4.2.1 Operațiunile CRUD (Create, Read, Update, Delete)

Operațiunile CRUD implementează funcționalitățile de adăugare, vizualizare, actualizare și ștergere a angajaților din baza de date a magazinului, unele dintre aceste funcționalități sunt disponibile și pentru alte tipuri de date cum ar fi produsele din magazin.

4.2.1.1 Vizualizarea datelor

Pentru a prezenta această funcționalitate s-a ales ca exemplu algoritmul pentru vizualizarea angajaților, care este descris în codul din figura următoare.

```
private void Populare(string query, string queryCond)
{
    try
    {
        MySqlDataAdapter adapter = new MySqlDataAdapter(query + queryCond, Constante.mySqlConnection);
        Constante.mySqlConnection.Open();
        DataSet ds = new DataSet();
        adapter.Fill(ds, "angajat");
        dataGridView1.DataSource = ds.Tables["angajat"];
        Constante.mySqlConnection.Close();
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Fig. 3.3 - Algoritm pentru vizualizarea angajaților

Folosind funcția „Populare” se va realiza popularea tabelului cu datele din baza de date pentru a putea fi vizualizate. Cei doi parametri ai funcției, „query” și „queryCond”, au rolul de a face interogarea bazei de date mai simplă, permițând utilizatorului să caute și după anumite criterii.

Primul parametru al funcției reprezintă interogarea de bază. Acesta este predefinit într-o variabilă de tipul **string** și are următoarea valoare:

Select id_angajat as `ID`, nume as `Nume`, prenume as `Prenume`, adresa as `Adresa`, varsta as `Varsta`, telefon as `Telefon`, cnp as `CNP`, emai as `Email`, data_angajarii as `Data angajarii`, durata_contract as `Durata contract` f.denumire as `Functie` from angajat a, functie f, where f.Id_functie=a.Id_functie;



Al doilea parametru al funcției reprezintă condițiile ce pot fi adăugate interogării de bază pentru a permite utilizatorului să caute un angajat după anumite criterii, cum ar fi nume, prenume sau CNP. Aceste criterii sunt introduse folosind casete de introducere a datelor denumite și **Textbox**, din care, în urma acționării unui buton specific se va realiza preluarea datelor introduse și crearea condițiilor pentru al doilea parametru. În figura următoare este exemplificat algoritmul pentru funcția de căutare asociată butonului specific acestei funcționalități.

```
private void Cauta_Click(object sender, EventArgs e)
{
    queryConditions = "";
    if (Nume.Text != "")
    {
        queryConditions += "and nume='" + Nume.Text + "' ";
    }

    if (Prenume.Text != "")
    {
        queryConditions += "and prenume='" + Prenume.Text + "' ";
    }

    if (Cnp.Text != "")
    {
        queryConditions += "and cnp=" + Cnp.Text + " ";
    }

    Populare(select, queryConditions);
}
```

Fig. 3.4 - Algoritm pentru căutarea datelor

4.2.1.2 Adăugarea datelor

Pentru a prezenta această funcționalitate, s-a ales ca exemplu algoritmul pentru adăugarea unui angajat, care este descris în figura următoare.

```
try
{
    string insert = "Insert into angajat(nume,prenume,adresa,varsta,telefon,cnp,email,data_angajarii,durata_contract,id_funcție) " +
        "values(@nume,@prenume,@adresa,@varsta,@telefon,@cnp,@email,@data,@durata,@id_funcție)";
    MySqlCommand mySqlCommand = new MySqlCommand(insert, Constante.mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@nume", nume);
    mySqlCommand.Parameters.AddWithValue("@prenume", prenume);
    mySqlCommand.Parameters.AddWithValue("@adresa", adresa);
    mySqlCommand.Parameters.AddWithValue("@varsta", varsta);
    mySqlCommand.Parameters.AddWithValue("@telefon", telefon);
    mySqlCommand.Parameters.AddWithValue("@cnp", cnp);
    mySqlCommand.Parameters.AddWithValue("@email", email);
    mySqlCommand.Parameters.AddWithValue("@data", data);
    mySqlCommand.Parameters.AddWithValue("@durata", durata);
    mySqlCommand.Parameters.AddWithValue("@id_funcție", IdFuncție(funcție.ToString()));
    Constante.mySqlConnection.Open();
    MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
    Constante.mySqlConnection.Close();
}
catch(MySqlException E)
{
    throw E;
}
```

Fig. 3.5 - Algoritm pentru adăugarea datelor



Adăugarea unui angajat necesită introducerea unor date, care, în cazul de față vor fi adăugate în variabila de tip `MySqlCommand` denumită „`mySqlCommand`”, prin intermediul parametrilor și a metodei specifice librăriei `MySQL.Data.MySqlClient` „`Parameters.AddWithValue()`”. În momentul acționării butonului specific pentru realizarea acestei acțiuni, se va verifica nulitatea câmpurilor prin care vor fi preluați parametrii respectivi. De asemenea, acești parametri vor fi validați pentru a evita eventualele erori sau tentativele de introducere eronată a datelor care ar duce la o deteriorare a datelor. Aceste validări avertizează utilizatorul în cazul în care datele introduse nu sunt corecte, arătând și ce date sunt greșite.

4.2.1.3 Actualizarea datelor

Pentru a prezenta această funcționalitate, s-a ales ca exemplu algoritmul pentru actualizarea unui angajat, care este descris în figura următoare.

```
try
{
    MySqlCommand mySqlCommand = new MySqlCommand(update, Constante.mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@id_angajat", VizualizareAngajat.id_angajat_redirect);
    Constante.mySqlConnection.Open();
    MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
    Constante.mySqlConnection.Close();
    Constante.Log("ModificareAngajat", DateTime.Now.ToString("dd-MMM-yyyy-HH-mm-ss"), Meniu.id_utilizator);
}
catch(Exception ex)
{
    throw ex;
}
```

Fig. 3.6 - Algoritm pentru actualizarea datelor

Asemeni introducerii unui angajat, actualizarea acestuia necesită introducerea unor date de către utilizator, cu ajutorul cărora se va realiza variabila „`update`” de tip **string** ce va fi folosită pentru interogarea bazei de date. De asemenea, datele introduse de către utilizator vor fi supuse unor verificări de nulitate, urmând să fie validate pentru evitarea erorilor și a tentativelor de deteriorare a datelor. Totodată, această funcționalitate oferă utilizatorului posibilitatea alegerii coloanei dorite pentru actualizare, de exemplu dacă se dorește doar actualizarea email-ului se va bifa doar căsuța (Checkbox-ul) respectiv acestui câmp.



4.2.1.4 Ștergerea datelor

Pentru a prezenta această funcționalitate, s-a ales ca exemplu algoritmul pentru ștergerea unui angajat, care este descris în figura următoare.

```
int rowIndex = dataGridView1.SelectedCells[0].RowIndex;
DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];
int id_angajat = Int32.Parse(Convert.ToString(selectedRow.Cells[0].Value));
try
{
    string delete = "delete from angajat where id_angajat=@id_angajat;";
    MySqlCommand mySqlCommand = new MySqlCommand(delete, Constante.mySqlConnection);
    mySqlCommand.Parameters.AddWithValue("@id_angajat", id_angajat);
    Constante.mySqlConnection.Open();
    MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
    Constante.mySqlConnection.Close();
}
catch (Exception ex)
{
    throw ex;
}
```

Fig. 3.6 - Algoritm pentru ștergerea datelor

Ștergerea unui angajat se realizează din interfața grafică a aplicației pentru vizualizarea angajaților, acționând butonul specific acestei funcționalități. În urma acționării butonului respectiv se prelua id-ul angajatului selectat din tabel, urmând ca acesta să fie adăugat ca și parametru în variabila de tip MySqlCommand denumita „mySqlCommand” ce va fi folosită pentru realizarea ștergerii din baza de date.

4.2.2 Alte metode

4.2.2.1 Metoda „EmailValidate”

Pentru evitarea introducerii datelor eronate ce pot duce la erori sau neplăceri în ceea ce privește funcționarea aplicației, metoda „EmailValidate” este utilizată pentru validarea adresei de email a utilizatorului, atât la introducerea, cât și la actualizarea datelor. Este foarte important ca această adresă să fie validă și corectă, deoarece va ajuta utilizatorul să-și obțină



datele pentru autentificare și pentru recuperarea parolei în cazul în care acesta o uită. Algoritmul pentru această metodă este descris în figura următoare.

```
public static bool EmailValidate(string email)
{
    try
    {
        var mail = new MailAddress(email);
        return true;
    }
    catch
    {
        return false;
    }
}
```

Fig. 3.7 - Algoritm pentru validarea adresei de email

Metoda primește ca și parametru email-ul introdus de către utilizator prin intermediul unui textbox, urmând ca acesta să fie utilizat în crearea unei adrese de email folosind funcția „MailAddress” specifică librăriei System.Net.Mail. Folosind structura **Try Catch** se va evita apariția erorilor în timpul rulării în cazul în care email-ul introdus este greșit. Această structură ajută aplicația în ceea ce privește tratarea erorilor și va avertiza utilizatorul că a introdus greșit datele.

4.2.2.2 Metoda „Log”

Pentru ca administratorul să poată avea un mod de a urmări activitatea utilizatorilor, iar în cazul în care s-a produs o greșeală să aibă posibilitatea de a ști când a avut loc aceasta și de către cine a fost produsă, metoda „Log” va fi utilizată. Algoritmul pentru această metodă este descris în figura următoare.



```
public static void Log(string actiune, string data, int id_utilizator)
{
    try
    {
        MySqlCommand mySqlCommand = new MySqlCommand("Insert into log(actiune,data,id_utilizator) values(@actiune,@data,@id_utilizator)",mySqlConnection);
        mySqlCommand.Parameters.AddWithValue("@actiune",actiune);
        mySqlCommand.Parameters.AddWithValue("@data", data);
        mySqlCommand.Parameters.AddWithValue("@id_utilizator", id_utilizator);
        mySqlConnection.Open();
        MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();
        mySqlConnection.Close();
        mySqlCommand.Parameters.Clear();
    }
    catch(MySqlException Ex)
    {
        throw Ex;
    }
}
```

Fig. 3.8 - Algoritm pentru istoric

Metoda primește trei parametrii, denumirea acțiunii realizate de utilizator, data la care este efectuată această acțiune și id-ul utilizatorului care a efectuat-o. Această metoda nu necesită ca parametrii să fie introduși de către utilizator, deoarece aceștia sunt adăugați în funcție de acțiunea realizată de utilizator. Metoda „Log” este apelată în urma autentificării, în urma operațiilor CRUD și în urma dezautentificării. În momentul apelării se vor stoca în baza de date informațiile primite ca parametrii pentru a oferi administratorului o bună gestiune a magazinului și un control sporit asupra acestuia.

4.2.2.3 Metoda „CriptareParola”

Pentru a oferi utilizatorilor o securitate sporită, este utilizată metoda „CriptareParola”, care, folosind funcții specifice librăriei System.Security.Cryptography, realizează criptarea parolei introduse de către utilizator folosind tipul hash unidirecțional MD5 (Message Digest Algorithm 5). Algoritmul pentru această metodă este descris în figura următoare.

```
public string CriptareParola(string parola)
{
    UTF8Encoding utf8 = new UTF8Encoding();
    MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
    byte[] parolaCriptata = md5.ComputeHash(utf8.GetBytes(parola));
    return Convert.ToBase64String(parolaCriptata);
}
```

Fig. 3.9 - Algoritm pentru criptarea parolei



Această metodă este apelată atât la introducerea utilizatorului în baza de date, cât și la autentificarea acestuia, parola acestuia fiind stocată sub forma criptată conform standardului MD5. Figura următoare reprezintă procesul de transformare a parolei introduse de utilizator într-o parolă criptată.

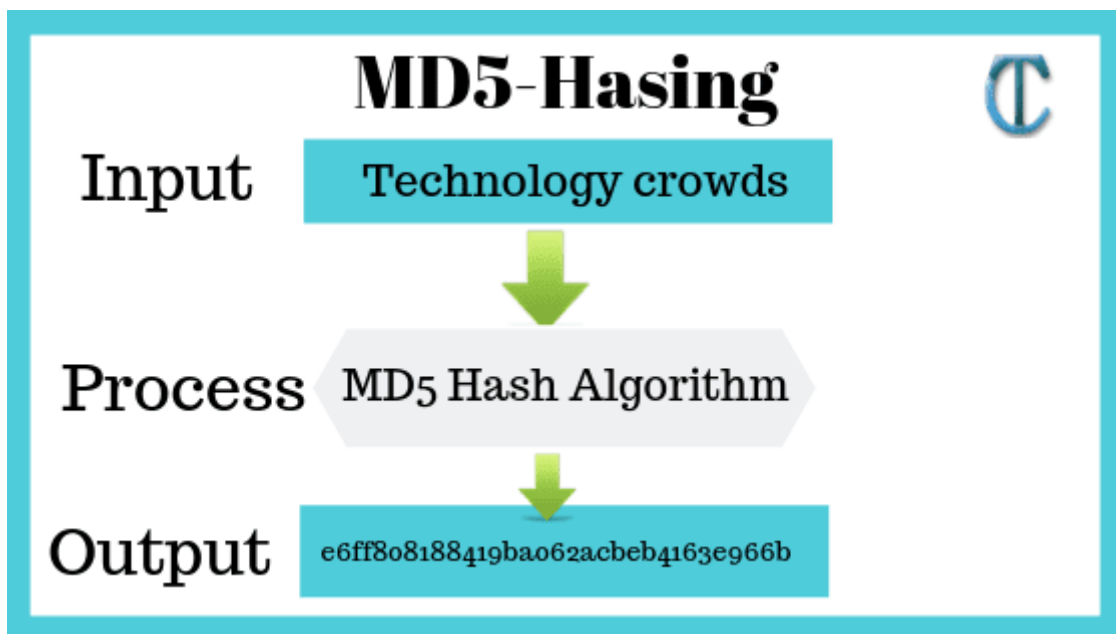


Fig. 3.10 – Procesul de criptare a parolei



5. Scanarea produselor

Una din principalele funcționalități ale aplicației este vânzarea de produse, care poate fi realizată de ambele tipuri de utilizatori, administrator și casier. Pentru ca această acțiune să poată fi realizată cât mai rapid și sigur s-a optat pentru folosirea unui cod QR, ce conține informațiile necesare produselor, cum ar fi denumirea, producătorul și prețul acestora.

5.1 Codul QR (Quick Response)

Codul QR este o gamă de standarde de codare cu formă de bare bidimensionale (cod matrice). Primul astfel de standard a fost creat în 1994 de către compania japoneza Denso Wave. Creatorii codului au vrut să producă un decodor rapid. Codul QR este marcă înregistrată a companiei Denso Wave, Inc. Deși aceste coduri nu sunt încă utilizate în multe părți ale lumii, ele sunt foarte frecvente în Japonia, unde acestea sunt cea mai populară formă de bare bidimensionale. Între timp s-au răspândit și în SUA, Canada, Germania și alte țări. În figura următoare este reprezentată adresa site-ului Facultății de Automatică, Calculatoare, Inginerie Electrică și Electronică din Galați sub forma codului QR.



Fig. 4.1 – Exemplu cod QR



Structura codului QR este destul de simplă, mai exact sub forma unui pătrat. Acest lucru se datorează faptului că scanerul de coduri QR pot recunoaște mai ușor această formă. În plus, pătratul poate maximiza eficiența păstrării și trimiterii informațiilor. În interior, imaginea asemănătoare pixelilor este foarte asemănătoare cu un puzzle și are șapte elemente cruciale:

1. **Marcaje de poziționare** : indică în ce mod este tipărit codul

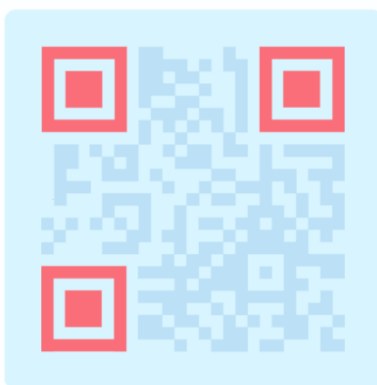


Fig. 4.2 – Marcaje de poziționare

2. **Marcaje de aliniere**: atunci când se utilizează coduri mai mari, aceste marcaje pot ajuta la orientare

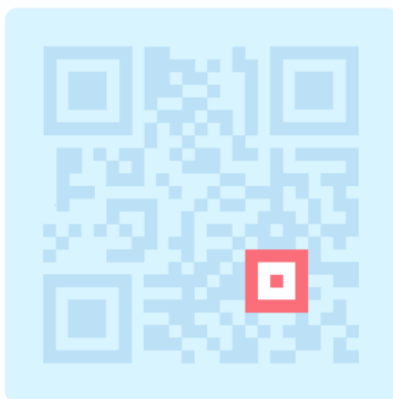


Fig. 4.2 - Marcaje de aliniere



3. **Modele de sincronizare:** acestea sunt linii care indică scannerului dimensiunea matricei de date

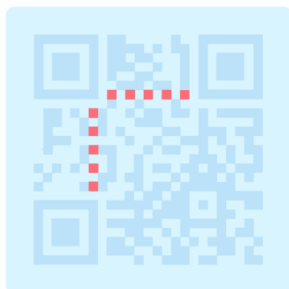


Fig. 4.3 - Marcaje de sincronizare

4. **Informații despre versiune:** cu această secțiune se poate specifica ce versiune a tuturor codurilor QR este utilizată. Există mai mult de 40 de versiuni ale acestora, dar în mod normal sunt folosite de la 1 la 7



Fig. 4.4 – Informații despre versiune

5. **Informații despre format:** Aceste modele includ informații despre toleranța la erori și modelul de mască de date, pentru a simplifica scanarea codului

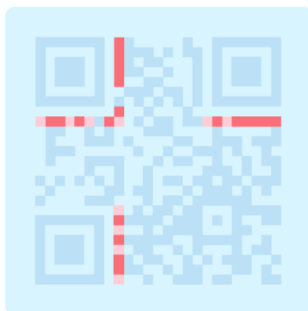


Fig. 4.5 – Informații despre format

6. **Taste de corectare a datelor și erorilor:** aici sunt afișate datele

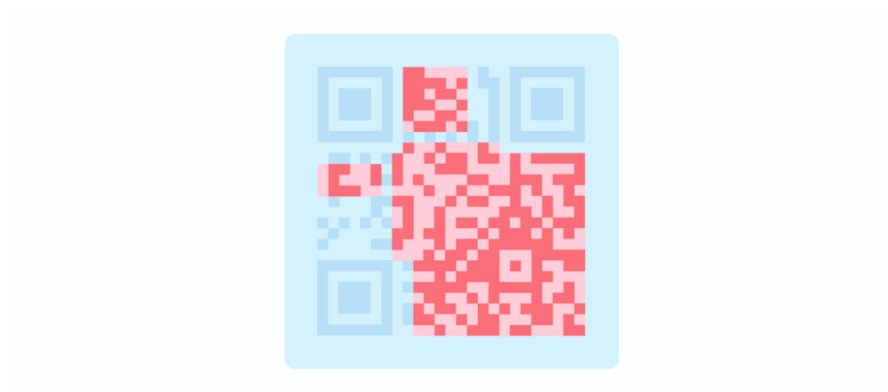


Fig. 4.6 – Taste de corectare a datelor și erorilor

7. **Zona liniștită:** acesta este un spațiu gol care ajută la scanarea codului prin diferențierea codului de împrejurimi

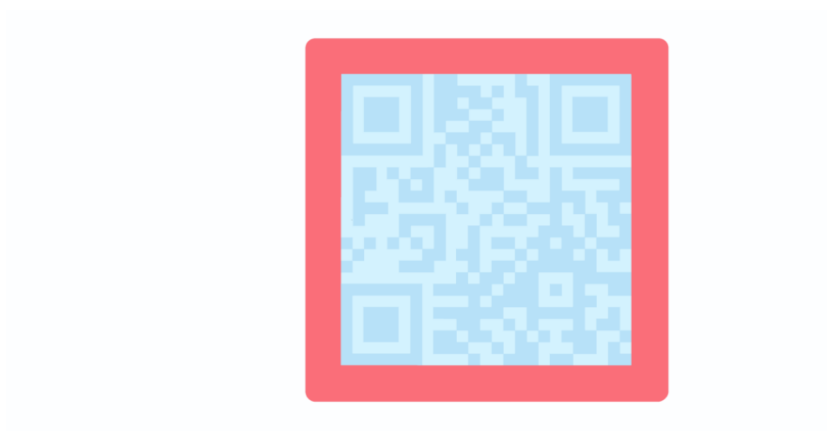


Fig. 4.7 – Zona liniștită

5.2 Zxing

Zxing (Zebra Crossing) este unul din cele mai populare API-uri (Application Programming Interface) open-source pentru procesarea codurilor QR. Este o librărie de procesare a imaginilor cu coduri de bare implementată în java, cu porturi către alte limbaje, în cazul aplicației prezentate, limbajul C#. Pentru importarea acestei librării a fost nevoie de folosirea **NuGet Package Manager**-ului din componența utilitarului Microsoft Visual Studio 2019 Community. A fost nevoie de această librărie pentru a fi posibilă captarea codului QR și transformarea acestuia în text.



5.3 AForge

Pentru a putea fi folosit un scanner sau o cameră video în timp real pentru captarea codului QR, a fost nevoie de librăria AForge, respectiv AForge.Video și AForge.Video.DirectShow. Asemeni librăriei precedente, Zxing, și această librărie a fost adăugată în aplicația prezentată tot cu **NuGet Package Manager**. AForge este disponibilă sub GNU, oferind dezvoltatorilor o soluție bună și rapidă în ceea ce privește procesarea de imagini în timp real, și totodată se înlătură posibilitatea pirateriei unor API-uri.

5.4 Motivația alegerii codului QR

În ciuda faptului că majoritatea magazinelor se folosesc de codurile de bare ale produselor pentru scanarea acestora, aplicația prezentată folosește coduri QR pentru acest lucru, deoarece sunt mai ușor și rapid de scanat, marea majoritate a smartphone-urilor dețin funcția de scanare a codurilor QR, fapt ce poate ajuta clienții să obțină mai rapid informații despre produsul de care aceștia sunt interesați.

De asemenea, crescând în popularitate, codul QR oferă aplicației un plus în ceea ce privește design-ul aplicației, deoarece este o metodă mult mai actuală de scanat, populară și cel mai important, tinde spre viitor, iar acest lucru se îmbină perfect cu tema aplicației.



6. Design

6.1 User Experience Design

Experiența utilizatorului (UX) este procesul de creare a produselor care oferă utilizatorilor experiențe semnificative și relevante. Aceasta implică proiectarea întregului proces de creare și integrare a produsului, inclusiv aspecte legate de branding, design, utilitate și funcționalitate.^[1] În acest caz, produsul este obiectul acestei lucrări, mai exact aplicația dezvoltată pentru gestiunea unui magazin de încălțăminte.

În continuare vor fi prezentate deciziile luate în legătură cu design-ul aplicației și motivul acestora. Acest aspect este important, deoarece cu cât interfața este mai favorabilă și intuitivă, cu atât utilizatorii o vor putea folosi mai ușor și mai rapid, făcând astfel mediul de lucru mai plăcut, obiectiv vital pentru ca afacerea beneficiarului să decurgă eficient și să ofere rezultate plăcute.

6.1.1 Accesibilitate

Această secțiune are ca scop găsirea cât mai rapidă a informațiilor căutate de către utilizator în momentul accesării aplicației. Pentru a îndeplini acest scop a fost implementată funcționalitatea de căutare pentru angajați, produse și nu numai. Locul unde se vor introduce criteriile de căutare și butonul pentru realizarea acestei acțiuni este poziționat deasupra tabelelor pentru vizualizarea datelor, fiind astfel vizibil și ușor de utilizat în orice moment.

Pentru a oferi o utilizare rapidă și eficientă a aplicației, am adăugat în interfața principală două meniuri prin care utilizatorul poate accesa funcționalitățile disponibile. Primul meniu este situat în partea superioară a ferestrei și este format din categorii ce conțin subcategorii prin intermediul cărora sunt accesate funcționalități pentru introducere, vizualizare, modificare și dezautentificare. Al doilea meniu este situat în partea centrală a ferestrei și este format doar din categorii prin intermediul cărora sunt accesate



funcționalitățile principale ale aplicației, introducerea, vânzarea și vizualizarea produselor. Pentru o și mai bună orientare și recunoaștere a acestor funcționalități, fiecare categorie, subcategorie și buton are o iconiță reprezentativă.

6.1.2 Ușurința completării datelor

Formularele pentru introducerea datelor sunt organizate astfel încât utilizatorul să știe fără a avea probleme ce date trebuie să introducă. Figura următoare reprezintă formularul de introducere a unui produs.

Imagine	<input type="button" value="Adaugă imagine..."/>
Denumire	<input type="text"/>
Producător	<input type="text"/>
Tip	<input type="text" value="Selectează tipul"/>
Sex	<input type="text" value="Selectează sexul"/>
Preț	<input type="text"/>
Distribuitor	<input type="text" value="Selectează distribuitorul"/>
Mărime	<input type="button" value="Selectează mărimile..."/>
<input type="button" value="Adaugă"/>	

Fig. 5.1 – Formular pentru introducerea unui produs

6.1.3 Ușurința parcurgerii informației

Este important ca datele afișate să fie bine structurate și concise, deoarece conform unui studiu recent, oamenii nu citesc în totalitate informațiile oferite.^[2] Ținând cont de acest aspect, datele ce vor putea fi vizualizate vor fi adăugate într-un tabel simplu, care are specificat în capul de tabel, ce reprezintă datele de pe coloana respectivă. De asemenea,



tabelele respective oferă utilizatorului posibilitatea de a ordona coloanele acestuia crescător sau descrescător, sporind astfel, ușurința cu care pot fi identificate datele. În figura următoare este prezentat tabelul pentru vizualizarea istoricului acțiunilor realizate de utilizatori.

ID	Actione	Data	Id utilizator
156	Logout	14-May-2021-14:38:53	1
157	Login	14-May-2021-14:39:28	1
158	Logout	14-May-2021-14:39:37	1
159	Login	14-May-2021-14:40:15	1
160	Logout	14-May-2021-14:40:20	1
161	Login	14-May-2021-15:13:26	1
162	Logout	14-May-2021-15:13:37	1
163	Login	14-May-2021-20:30:11	1
164	Logout	14-May-2021-20:30:24	1
165	Login	25-May-2021-18:11:01	1
166	Introducere/Vanzare	25-May-2021-18:13:53	1
167	Logout	25-May-2021-18:23:24	1
168	Logout	25-May-2021-18:23:26	1
169	Login	26-May-2021-12:25:44	1
170	Logout	26-May-2021-12:25:45	1
171	Login	02-Jun-2021-14:44:13	27
172	Logout	02-Jun-2021-14:53:51	27
173	Logout	02-Jun-2021-14:55:05	27
174	Login	02-Jun-2021-15:16:53	1
175	Logout	02-Jun-2021-15:16:55	1
176	Logout	02-Jun-2021-15:16:59	1
177	Login	02-Jun-2021-15:17:50	1
178	Logout	02-Jun-2021-15:28:52	1
179	Login	02-Jun-2021-15:43:46	1
180	Logout	02-Jun-2021-15:59:43	1
181	Login	07-Jun-2021-17:53:12	1
182	Login	07-Jun-2021-18:08:54	1
183	Logout	07-Jun-2021-18:09:18	1
184	Login	07-Jun-2021-18:11:00	1

Fig. 5.2 – Tabel istoric

6.2 User Interface Design

User Interface Design (UI) este procesul de realizare a interfețelor software sau a dispozitivelor computerizate punându-se accent pe aspect sau stil.^[3] Scopul principal al UI-ului este ca produsele sale să fie plăcute din punct de vedere estetic și ușor de utilizat. Studiile arată că 94% din prima impresie a unui utilizator este dată de estetica aplicației. Credibilitatea aplicației depinde și ea foarte mult de estetica acesteia, astfel s-a optat pentru păstrarea unui logo pe toate ferestrele aplicației.^[4]

În continuare se va face referire la elementele necesare UI-ului, mai exact la font-urile și culorile folosite la design-ul site-ului, dar și motivele pentru care au fost alese.



6.2.1 Font-uri

În cadrul dezvoltării aplicației s-au utilizat trei font-uri. Unul de tip *Segoe UI* cu stilul **Bold**, pentru evidențierea meniului central, unul de tip *Segoe UI Semibold*, pentru evidențierea meniului situat în partea superioară a ferestrei, ce conține categorii și subcategorii, și unul de tip *Sans Serif*, pentru restul conținutului. Aceste font-uri oferă aplicației un plus de profesionalism și seriozitate, caracteristici ce ajută la dobândirea încrederii utilizatorului în ceea ce privește aplicația.

6.2.2 Culori

Pentru tot ce înseamnă text, am ales să folosesc culorile alb și negru pentru a induce o seriozitate sporită în ceea ce privește aplicația. Pentru fundalul ferestrelor și butoane am ales culorile albastru, cyan, roșu și magenta folosindu-le pe o temă futuristică ce utilizează figuri geometrice, oferind senzația de viitor și evoluție. Nu s-a dorit încărcarea aplicației cu foarte multe culori, deoarece se puteau crea neplăceri și dificultăți în atenția utilizatorilor. În figura următoare se pot observa culorile alese și contrastul plăcut pe care acestea îl oferă.

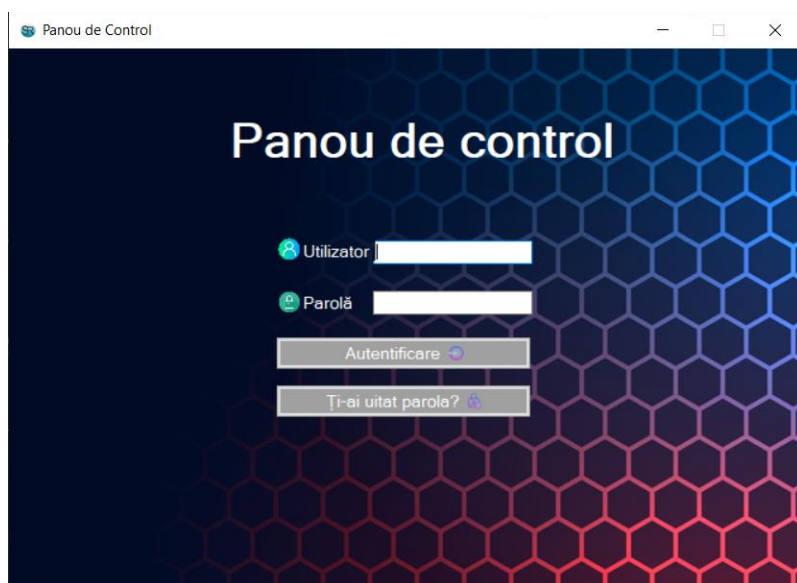


Fig. 5.3 – Fereastra pentru autentificare



7. Securitate

.NET Framework are implementate măsuri de securitate pe care dezvoltatorii le pot exploata și pot profita de acestea. Aceste măsuri țin cont de obiectivele criptografiei moderne și anume de confidențialitatea informației, integritatea datelor, disponibilitatea datelor și autentificarea.^[5]

7.1 Confidențialitate

Confidențialitatea se referă la păstrarea secretului informației, accesul la informația sensibilă, fiind disponibilă doar persoanelor autorizate. În cazul acestei aplicații există mai multe tipuri de informații sensibile cum ar fi datele despre angajați și utilizatori, datele despre departamente și funcțiile acestora. Aceste informații sunt disponibile doar pentru administratori și pot fi accesate folosind meniul situat în partea superioară a aplicației din fereastra ce se va deschide în urma autentificării realizate cu succes.

7.2 Integritatea datelor

Integritatea datelor se referă la eliminarea posibilității de modificare (schimbare, ștergere și inserare) neautorizată a informației. Pentru evitarea atacurilor la integritatea datelor numite și SQL Injection, .NET Framework utilizează limbajul LINQ și funcțiile prestabilite de preluare a datelor din baza de date. De asemenea, utilizatorii care nu au rol de administrator au acces limitat în ceea ce privește introducerea, modificarea și ștergerea datelor prin diminuarea opțiunilor disponibile din meniuri.



7.3 Disponibilitatea datelor

Disponibilitatea datelor se referă la permiterea utilizatorilor autentificați să acceseze ușor și în timp util informația. Totodată, disponibilitatea aplicației este dependentă de conexiunea la baza de date, toate datele fiind stocate în aceasta.

7.4 Autentificarea

Pentru a fi cât mai sigură, autentificarea transformă parola utilizatorului în **hash code** înainte de inserarea în baza de date, dar și în momentul verificării acesteia când se realizează autentificarea. Așadar, parola nu este niciodată reținută în **plain text**, în plus parola utilizatorilor este generată la întâmplare folosind litere mari, litere mici și cifre, crescând foarte mult complexitatea acesteia, făcând-o mai greu de compromis. De asemenea, administratorul nu are acces la parolele din baza de date, acestea fiind trimise prin email către utilizatori în momentul adăugării lor. Figura următoare reprezintă modul în care sunt afișate parolele în baza de date.

Parola	Rol	Cod_parola
JyuJg75aPHWNnG9l30+Adg==	Admin	NULL
JyuJg75aPHWNnG9l30+Adg==	Casier	NULL
d2zzf4V/G9KU295itJVJmQ==	Casier	NULL
NULL	NULL	NULL

Fig. 6.1 – Parola salvată sub forma de hash code

În cazul în care un utilizator își uită parola o poate reseta simplu folosindu-se de adresa de email. În figura următoare este prezentată fereastra pentru recuperarea parolei.

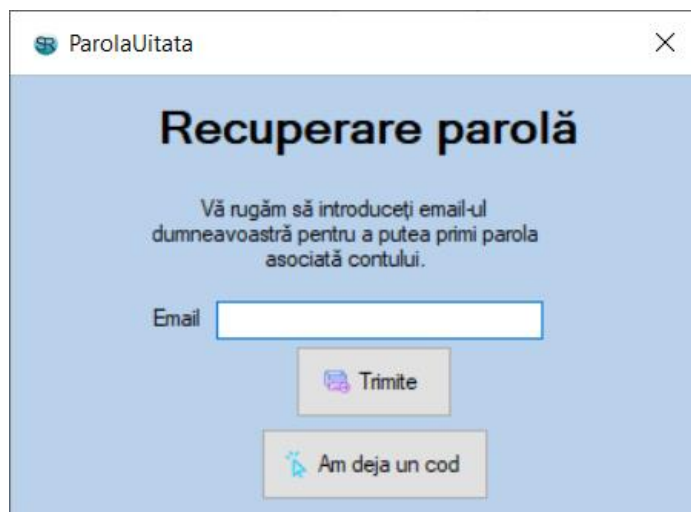


Fig. 6.2 – Fereastra pentru recuperarea parolei

În urma introducerii email-ului, acesta este mai întâi validat, iar apoi, dacă este valid și există în baza de date, aplicația va genera un cod în tabela *utilizator* în dreptul utilizatorului care deține acel email, iar pentru a evita posibilele probleme, în baza de date se poate adăuga doar o singură dată un email. Acest cod generat va fi transmis utilizatorului, acesta putând să-l folosească pentru a avea acces la fereastra pentru resetarea parolei. Dacă utilizatorul și-a resetat parola folosind codul generat de aplicație, codul va fi imediat șters din baza de date pentru a evita compromiterea contului.



8. Feedback

Opinia clientului (feedback) este o informație ce reprezintă nivelul lor de satisfacție referitor la serviciile și produsele puse la dispoziție de către magazin și angajații acestuia. Opinia lor are o importanță foarte mare pentru dezvoltarea serviciilor oferite și pentru a oferi clienților o experiență plăcută. Deși nu se pot mulțumii toți clienții, este important pentru afacere ca procentul celor nemulțumiți să fie cât mai mic, deoarece dacă sunt prea multe opinii negative la adresa magazinului, acesta va începe să piardă clienți, ajungând astfel, la un posibil faliment.

8.1 Îmbunătățirea serviciilor

Magazinul trebuie să ofere servicii de calitate clienților, pentru ca aceștia să rămână fideli afacerii respective și să aibă o experiență plăcută. Oferindu-le clienților oportunitatea de a-și exprima opinia în legătură cu serviciile primite, atât magazinul, cât și clientul vor avea beneficii. Pentru ca antreprenorul să-și poată îmbunătăți serviciile oferite, acesta trebuie să cunoască toate detaliile, atât negative, cât și pozitive. Cel mai ușor mod de a putea afla aceste detalii este prin aflarea opiniei clienților, deoarece aceștia știu cel mai bine care este toată interacțiunea cu angajații și magazinul.

Un magazin care cunoaște foarte bine ce probleme există în legătură cu serviciile oferite, va putea cu ușurință să gestioneze situația, să îmbunătățească aplicația și să nu piardă clienții.



8.2 Păstrarea clienților

Feedback-ul clienților va ajuta afacerea să înțeleagă exact ce experiență au clienții cu serviciile oferite și pot fixa rapid eventualele probleme, lucru ce va duce la o experiență mult mai plăcută. O experiență bună și o satisfacție a clienților, va duce rapid la păstrarea acestora. Este important ca magazinul să nu piardă clienți, deoarece este vital ca cifra de afaceri să nu scadă. De asemenea, profitul magazinului va crește semnificativ, dacă se vor păstra clienții, aceștia recomandând serviciile primite și altor persoane. Totodată, clienții mulțumiți vor fi dispuși să cheltuiască mai mulți bani în produsele oferite de magazin având încredere în serviciile primite, acest lucru va duce la o promovare a afacerii.

Este important ca cei care folosesc serviciile oferite de magazin și aplicația prezentată să aibă un drept la o opinie sinceră, deoarece astfel se va căpăta o încredere mai mare în afacerea respectivă. Este cunoscut faptul că găsirea unui client nou și câștigarea încrederii acestuia este mai costisitoare decât păstrarea unui client vechi.

9. Chestionarul

Cunoscând importanța opiniei oferite de clienți aplicația dispune de un chestionar simplu format din trei întrebări simple, care conține un sistem de măsurare a satisfacției clientului ce utilizează numere de la 1 la 5, 1 reprezentând foarte nesatisfăcător, iar 5 reprezentând foarte satisfăcător. De asemenea, acest chestionar dispune și de posibilitatea de a oferi o sugestie de către client, astfel antreprenorul va ști ce nevoi există în rândul clienților, iar aceștia se vor simți apreciați și de valoare știind că opinia lor contează și că poate ajuta la dezvoltarea afacerii. În figura următoare este prezentată fereastra pentru completarea chestionarului.

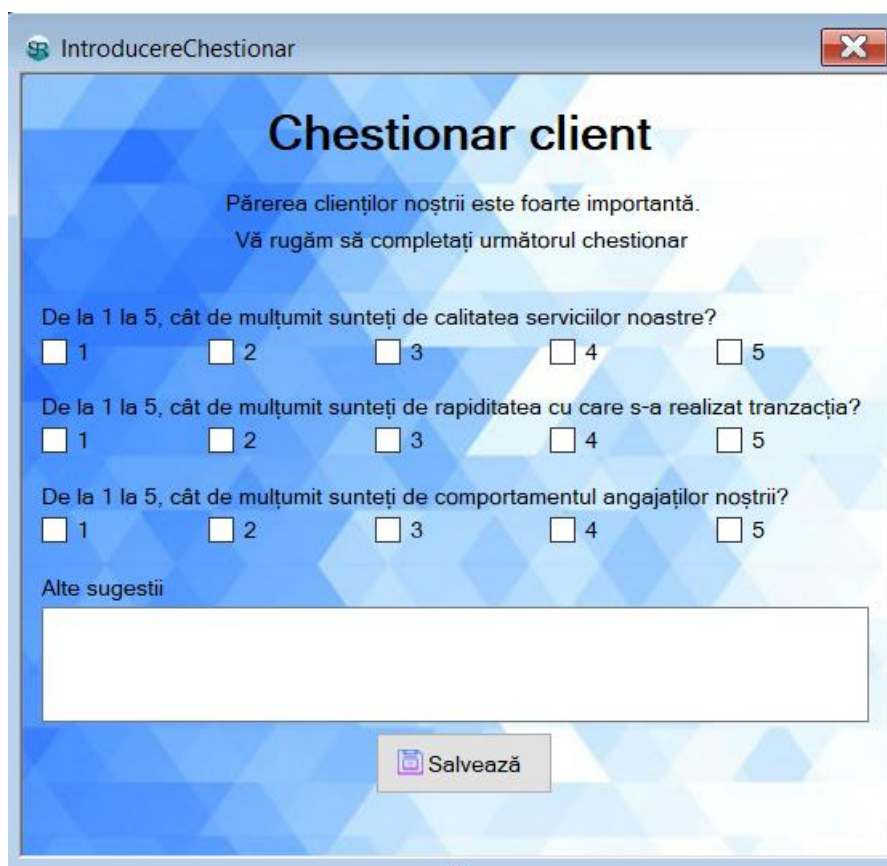


Fig. 7.1 – Chestionarul pentru client

Pentru evitarea introducerii eronate a acestui chestionar, aplicația va oferi posibilitatea completării acestui chestionar, doar în urma achiziției unui produs, astfel, nu se va putea completa de către angajații magazinului în scopul creării unei opinii greșite.

10. Statistici

Un rol important în dezvoltarea unei afaceri o constituie statisticile despre diferite obiective ale antreprenorului. Acesta are nevoie de aceste date pentru a ști ce schimbări trebuie făcute, ce produse trebuie adăugate și așa mai departe. Pentru acest lucru, aplicația oferă utilizatorilor cu rol de administrator accesul la diagrame, care, la momentul prezentării aplicației conțin informații despre vândute și numărul acestora, despre cele mai vândute mărimi, și despre media notelor din chestionar. Bineînțeles, aceste diagrame pot fi dezvoltate pe viitor în funcție de necesitatea și preferința antreprenorului. În figura următoare este prezentată diagrama pentru afișarea produselor vândute și numărul acestora.

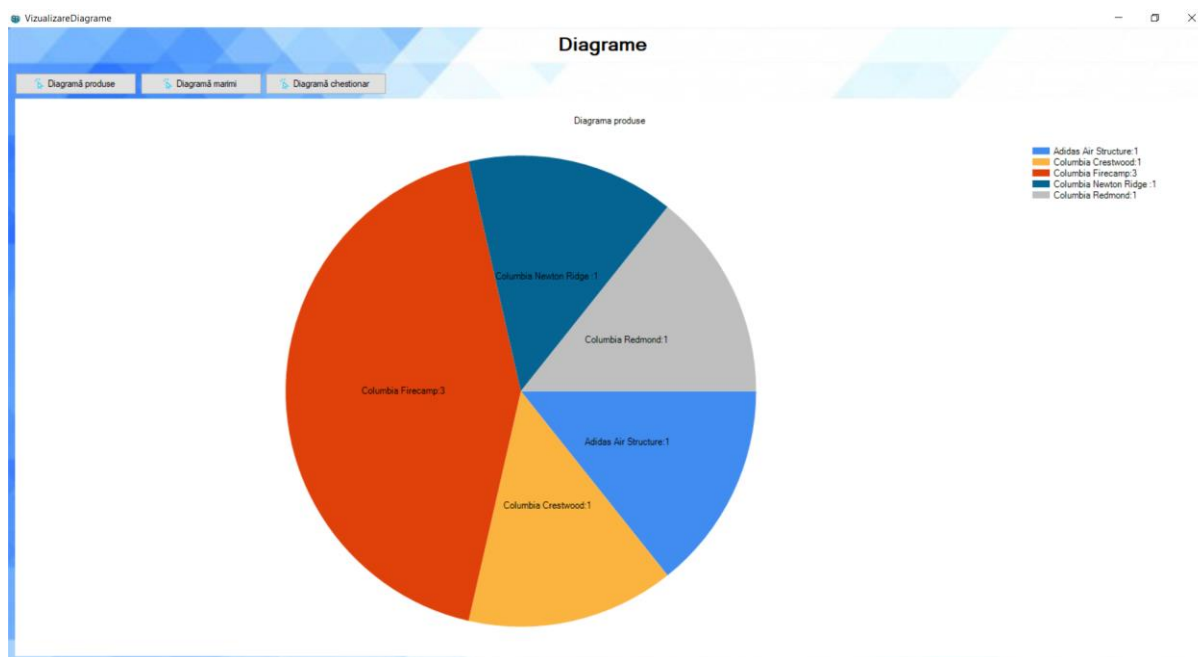


Fig. 8.1 – Diagrama pentru produsele vândute

Folosind această diagramă de tip **PieChart**, se poate depista foarte rapid și simplu care este cea mai vândută pereche de încălțăminte, astfel, magazinul poate crește cu ușurință aprovizionarea cu produsul respectiv, deoarece acesta va aduce profit afacerii.



11. Manual de utilizare

11.1 Fereastra de autentificare

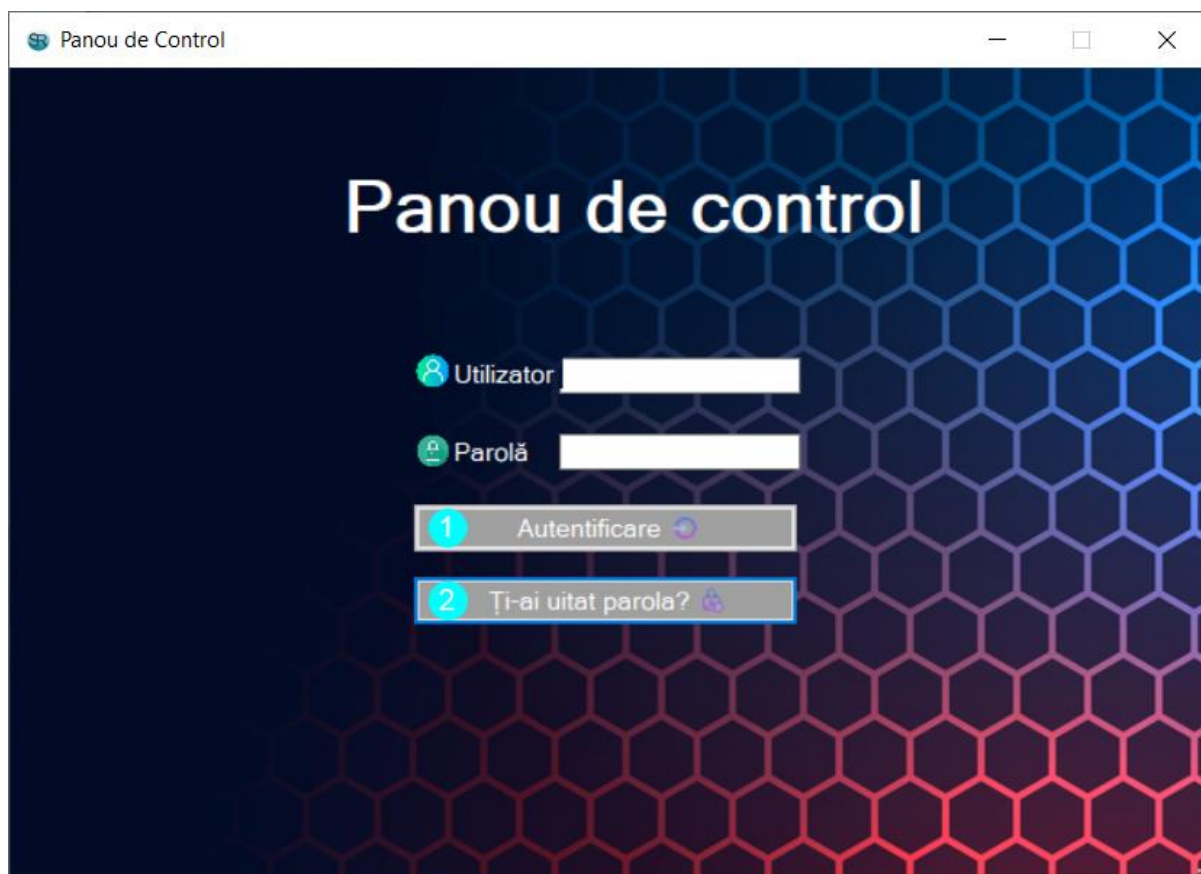


Fig. 9.1 – Utilizare fereastră de autentificare

1. Butonul **Autentificare** realizează autentificarea și face trimiterea către fereastra principală a aplicației. Pentru realizarea cu succes a acestei acțiuni este nevoie de completarea casetelor text **Utilizator** și **Parolă** cu datele corespunzătoare valide.
2. Butonul **Ți-ai uitat parola?** face trimiterea către fereastra pentru recuperarea parolei în cazul în care utilizatorul a uitat-o.

11.2 Recuperarea parolei

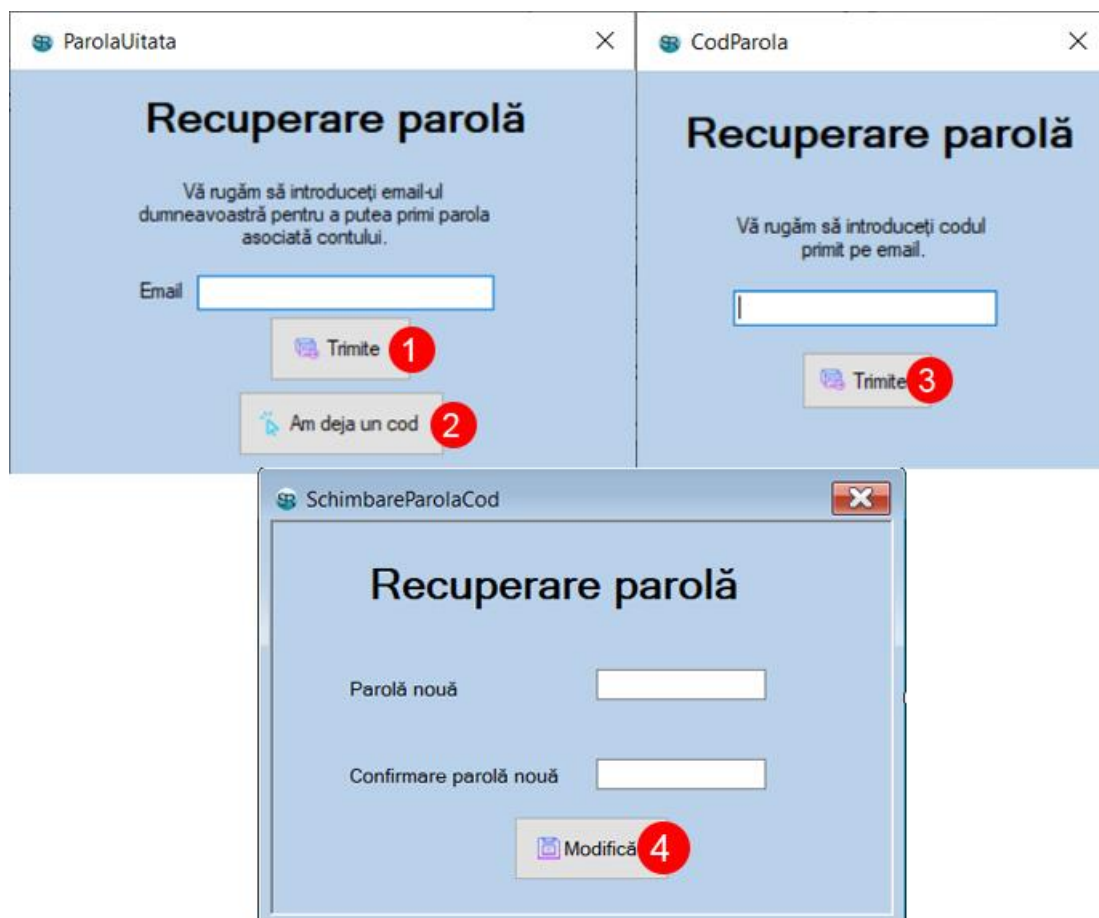


Fig. 9.2 – Utilizare ferestre pentru schimbarea parolei

1. Butonul **Trimite** din fereastra **ParolaUitata** creează un cod unit în baza de date pentru utilizatorul ce deține email-ul introdus în caseta text dacă acesta este valid, iar apoi trimite către acesta codul creat ce va fi folosit pentru schimbarea parolei.
2. Butonul **Am deja un cod** face trimiterea către fereastra **CodParola** în cazul în care utilizatorul are deja un cod pentru schimbarea parolei.
3. Butonul **Trimite** din fereastra **CodParola** face trimiterea către fereastra pentru schimbarea parolei în cazul în care codul introdus este valid.
4. Butonul **Modifică** schimbă parola existentă în baza de date pentru contul ce aparține email-ului introdus, cu parola nouă, introdusă în caseta text.

11.3 Meniu

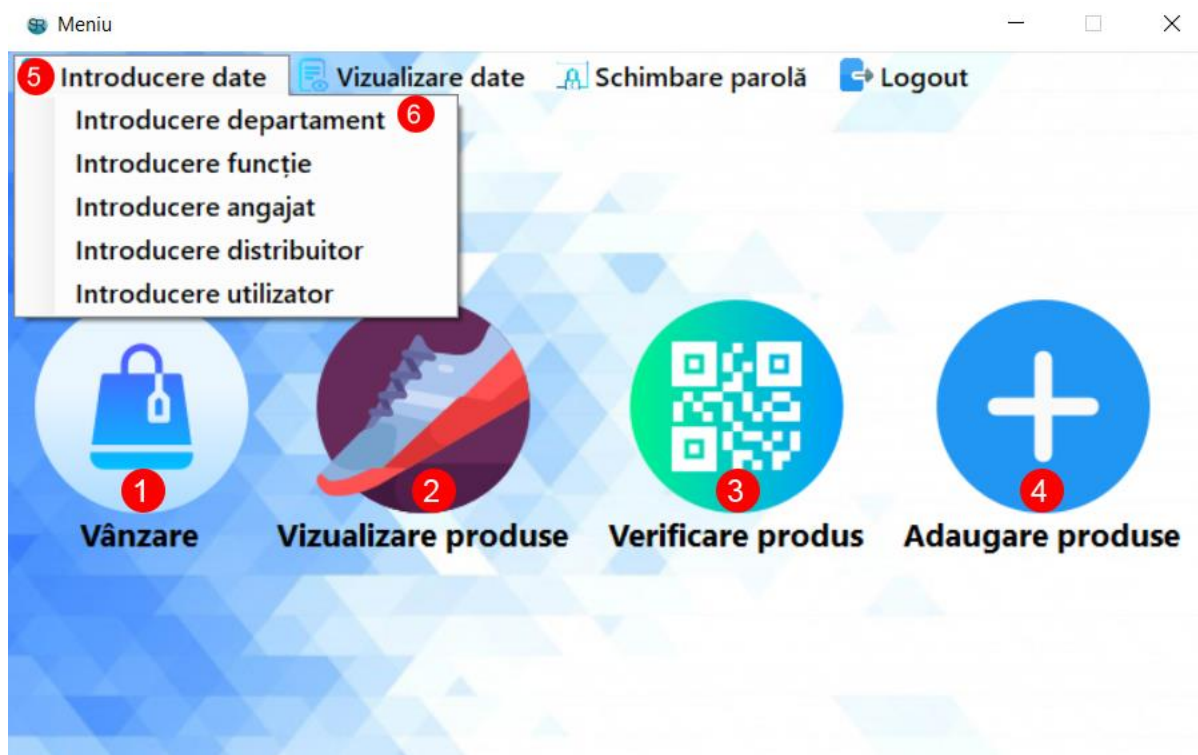


Fig. 9.3 – Utilizarea meniului

1. Butonul **Vânzare** face trimiterea către fereastra unde se realizează vânzarea de produse.
2. Butonul **Vizualizare produse** face trimiterea către fereastra unde se pot vizualiza produsele existente în baza de date.
3. Butonul **Verificare produs** face trimiterea către fereastra unde se va putea scana un produs pentru a verifica măsurile împreună cu stocul disponibil în magazin.
4. Butonul **Adăugare produse** face trimiterea către fereastra unde se vor putea adăuga produse noi în baza de date a magazinului.
5. Butoanele din meniul secundar poziționat în partea superioară a ferestrei, afișează subcategoriile disponibile pentru tipul de utilizator autentificat.
6. Butoanele subcategoriilor vor face trimiterea către fereastra specifică acestora.



12. Concluzii

Utilizând cunoștințele acumulate în ultimii 3 și framework-ul .NET Framework împreună cu o bază de date MySQL, a avut loc cu succes, realizarea unui produs finit, mai exact o aplicație Desktop pentru a gestiona mai ușor un magazin de încălțăminte și nu numai.

Pe parcursul dezvoltării aplicației, au apărut probleme de proiectare a bazei de date și probleme în ceea ce privește scanarea obiectelor folosind un cod QR. Pentru a rezolva aceste soluții a fost necesară folosirea soluțiilor găsite în cursurile predate la facultate sau în mediul online, sursele pot fi găsite în capitolul Bibliografie și referințe.

Cel mai mare beneficiu al acestei lucrări este că aplicația poate ajuta antreprenorii să economisească foarte mult timp, personal și bani. Versiunea prezentată în această lucrare poate fi îmbunătățită, în funcție de cerințele adiționale ale beneficiarului, de exemplu mai multe diagrame pentru diferite statistici sau mai multe tabele. Totodată, fiind o aplicație cu o interfață simplistă și intuitivă va permite utilizatorilor să o folosească cu ușurință, făcând mediul de lucru mai plăcut.



13. Bibliografie și referințe

Bibliografie

1. <https://stackoverflow.com>, 13-06-2021
2. <https://www.c-sharpcorner.com/UploadFile/deepak.sharma00/send-email-from-C-Sharp-windows-application-using-gmail-smtp>, 13-06-2021
3. <https://en.wikipedia.org>, 13-06-2021
4. https://www.youtube.com/watch?v=0_u-9nykBrg&ab_channel=FoxLearn, 13-06-2021
5. <https://www.geeksforgeeks.org/architecture-of-mysql>, 13-06-2021
6. <https://www.technologycrowds.com/2019/09/how-to-compute-md5-hash-message-digest.html>, 13-06-2021
7. <https://www.c-sharpcorner.com/UploadFile/mirfan00/uploaddisplay-image-in-picture-box-using-C-Sharp>, 13-06-2021
8. https://www.freepik.com/free-vector/hexagonal-technology-pattern-mesh-background-with-text-space_9106770.htm#page=1&query=background&position=43 13-06-2021
9. <https://icons8.com> 13-06-2021
10. Suport de curs - Ș.I. Dr. Ing. Cristina Elena Anton

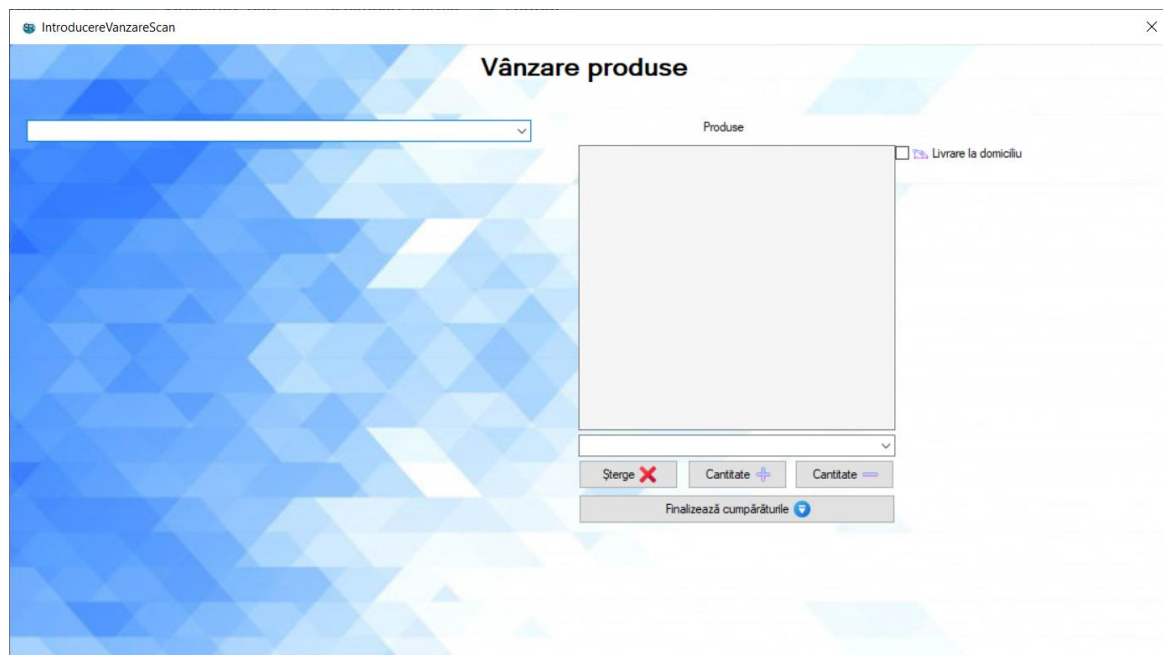
Referințe

- [1] <https://www.interaction-design.org/literature/topics/ux-design> 13-06-2021
- [2] https://rubygarage.org/blog/ecommerce-ux-tips#article_title_8 13-06-2021
- [3] <https://www.interaction-design.org/literature/topics/ui-design> 13-06-2021
- [4] <https://www.invisionapp.com/inside-design/statistics-on-user-experience> 13-06-2021
- [5] Ruxandra Olimid, Curs Criptografie și Securitate, Facultatea de Matematică și Informatică, Universitatea din București



Anexa 1

Figurile următoare ilustrează capturi de ecran ale aplicației urmate de codul corespunzător.



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AForge.Video;



```
using AForge.Video.DirectShow;

using ZXing;

using QRCode;

using MySql.Data.MySqlClient;

using System.IO;

using System.Drawing.Imaging;

using System.Net.Mail;

using System.Net;

namespace Licenta.Introductere
{
    public partial class IntroducereVanzare : Form
    {
        FilterInfoCollection filterInfoCollection;

        VideoCaptureDevice captureDevice;

        public static int id_produs_redirect;

        public List<int> id_produs = new List<int>();

        public List<string> produse=new List<string>();

        public List<int> nr_marime = new List<int>();

        public List<int> cantitate = new List<int>();

        public List<int> pret = new List<int>();

        public List<int> cantitate_maxima = new List<int>();

        public static string result_copy;
```



```
public int id_bon;
```

```
public IntroducereVanzare()
```

```
{
```

```
    InitializeComponent();
```

```
}
```

```
private void IntroducereVanzareScan_Load(object sender, EventArgs e)
```

```
{
```

```
    resetDomiciliuPanel();
```

```
    filterInfoCollection = new FilterInfoCollection(FilterCategory.VideoInputDevice);
```

```
    foreach (FilterInfo filterInfo in filterInfoCollection)
```

```
        Camera.Items.Add(filterInfo.Name);
```

```
}
```

```
public int Id_Produs(byte[] qr)
```

```
{
```

```
    try
```

```
    {
```

```
        MySqlCommand mySqlCommand = new MySqlCommand("Select id_produs from  
produs where qrcode=@qrcode", Constante.mySqlConnection);
```

```
        mySqlCommand.Parameters.AddWithValue("@qrcode", qr);
```

```
        Constante.mySqlConnection.Open();
```



```
MySQLDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

if (mySqlDataReader.Read())

{

    int x = mySqlDataReader.GetInt32(0);

    Constante.mySqlConnection.Close();

    return x;

}

else

{

    Constante.mySqlConnection.Close();

    return 0;

}

} catch (Exception ex){

    throw ex;

}

}

public int Id_Bon(byte[] qr)

{

    try

    {

        MySqlCommand mySqlCommand = new MySqlCommand("Select id_bon from bon

where qrcode=@qrcode", Constante.mySqlConnection);
```



```
mysqlCommand.Parameters.AddWithValue("@qr", qr);

Constante.mySqlConnection.Open();

MySqlDataReader mySqlDataReader = sqlCommand.ExecuteReader();

if (mySqlDataReader.Read())

{

    int x = mySqlDataReader.GetInt32(0);

    Constante.mySqlConnection.Close();

    return x;

}

else

{

    Constante.mySqlConnection.Close();

    return 0;

}

}

catch (Exception ex)

{

    throw ex;

}

}

private void Camera_SelectedIndexChanged(object sender, EventArgs e)

{
```



```
captureDevice = new
VideoCaptureDevice(filterInfoCollection[Camera.SelectedIndex].MonikerString);

captureDevice.NewFrame += CaptureDevice_NewFrame;

captureDevice.Start();

timer1.Start();
}

private void CaptureDevice_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
    ScanImage.Image = (Bitmap)eventArgs.Frame.Clone();
}

private void PopulateBon()
{
    int pret_total = 0;

    Bon.Text = "";

    ProduseSelect.Items.Clear();

    for (int i = 0; i < produse.Count; i++)
    {
        Bon.Text += produse[i] + Environment.NewLine + "Marime: " + nr_marime[i] +
        Environment.NewLine + "Cantitate: " + cantitate[i] + Environment.NewLine + "Pret: " + pret[i]
        + Environment.NewLine ;

        pret_total = pret_total + (cantitate[i] * pret[i]);

        ProduseSelect.Items.Add(produse[i]);
    }
}
```



```
}

if (pret_total > 0)

{

    Bon.Text += Environment.NewLine + Environment.NewLine + "Pret total: " +
pret_total;

}

}

private bool ValidateProdus(List<string> list,string produs)

{

    bool x=false;

    if (list.Count != 0)

    {

        for (int i = 0; i < list.Count; i++)

        {

            if (list[i] == produs)

            {

                x = true;

                break;

            }

        }

    }

}

return x;
```



```
}
```

```
private bool ValidateMarime(List<int> list, int marime)
```

```
{
```

```
    bool x = false;
```

```
    if (list.Count != 0)
```

```
    {
```

```
        for (int i = 0; i < list.Count; i++)
```

```
        {
```

```
            if (list[i] == marime)
```

```
            {
```

```
                x = true;
```

```
                break;
```

```
            }
```

```
        }
```

```
    }
```

```
    return x;
```

```
}
```

```
private void timer1_Tick(object sender, EventArgs e)
```

```
{
```

```
    if (ScanImage.Image != null)
```

```
    {
```



```
BarcodeReader barcodeReader = new BarcodeReader();

Result result = barcodeReader.Decode((Bitmap)ScanImage.Image);

if (result != null)
{
    timer1.Stop();

    if (captureDevice.IsRunning)
    {
        captureDevice.Stop();
    }

    produse.Add(result.ToString());

    result_copy = result.ToString();

    QRCodeGenerator qr = new QRCodeGenerator();

    var codeData = qr.CreateQrCode(result.ToString(),
QRCodeGenerator.ECCLLevel.H);

    var code = new QRCode(codeData);

    QrVerify.Image = code.GetGraphic(50);

    Image imageQR = QrVerify.Image;

    MemoryStream memoryStreamQR = new MemoryStream();

    imageQR.Save(memoryStreamQR, ImageFormat.Png);

    byte[] imageByteQR = memoryStreamQR.ToArray();

    if (Id_Produs(imageByteQR) == 0)
    {
        MessageBox.Show("Fail");
    }
}
```




```
        ScanImage.Image = null;

        captureDevice.NewFrame += CaptureDevice_NewFrame;

        captureDevice.Start();

        timer1.Start();

    }

    else

    {

        id_produc_redirect = Id_Produc(imageByteQR);

        IntroducereMarimeCantitate    introducereMarimeCantitate    =    new
IntroducereMarimeCantitate();

        introducereMarimeCantitate.ShowDialog();

    }

}

}

if (IntroducereMarimeCantitate.start_stop)

{

    if (IntroducereMarimeCantitate.marime_redirect == 0)

    {

        produse.RemoveAt(produse.Count-1);

    }

    else

    {

        if(ValidateProduc(produse,result_copy)    &&    ValidateMarime(nr_marime,
IntroducereMarimeCantitate.marime_redirect)    &&
```



```
produse.IndexOf(result_copy)==nr_marime.IndexOf(IntroducereMarimeCantitate.marime_r  
edirect))  
  
    {  
  
        if(cantitate[produse.IndexOf(result_copy)]+IntroducereMarimeCantitate.cantitate_redirect>  
cantitate_maxima[produse.IndexOf(result_copy)])  
  
            {  
  
                produse.RemoveAt(produse.Count - 1);  
  
                MessageBox.Show("Nu se poate adauga numarul de produse deoarece stocul  
disponibil este de: " + cantitate_maxima[produse.IndexOf(result_copy)]);  
  
            }  
  
            else  
  
            {  
  
                produse.RemoveAt(produse.Count - 1);  
  
                cantitate[produse.IndexOf(result_copy)] =  
cantitate[produse.IndexOf(result_copy)] + IntroducereMarimeCantitate.cantitate_redirect;  
  
            }  
  
        }  
  
        else  
  
        {  
  
            id_produs.Add(id_produs_redirect);  
  
            nr_marime.Add(IntroducereMarimeCantitate.marime_redirect);  
  
            cantitate.Add(IntroducereMarimeCantitate.cantitate_redirect);  
  
            cantitate_maxima.Add(IntroducereMarimeCantitate.stoc_redirect);
```



```
        pret.Add(IntroducereMarimeCantitate.pret_redirect);

    }

}

PopulateBon();

ScanImage.Image = null;

captureDevice.NewFrame += CaptureDevice_NewFrame;

captureDevice.Start();

timer1.Start();

IntroducereMarimeCantitate.start_stop=false;

}

}

private void Sterge_Click(object sender, EventArgs e)

{

    if (ProduseSelect.SelectedItem == null)

    {

        MessageBox.Show("Selecteaza un produs");

    }

    else

    {

        int index = produse.IndexOf(ProduseSelect.SelectedItem.ToString());

        id_produs.RemoveAt(index);

    }

}
```



```
        produse.RemoveAt(index);

        nr_marime.RemoveAt(index);

        cantitate.RemoveAt(index);

        pret.RemoveAt(index);

        cantitate_maxima.RemoveAt(index);

        ProduseSelect.ResetText();

        ProduseSelect.SelectedIndex = -1;

        PopulateBon();

    }

}

private void CantitateAdd_Click(object sender, EventArgs e)

{

    if (ProduseSelect.SelectedItem == null)

    {

        MessageBox.Show("Selecteaza un produs");

    }

    else

    {

        int index = produse.IndexOf(ProduseSelect.SelectedItem.ToString());

        if (cantitate[index] < cantitate_maxima[index])

        {

            cantitate[index]++;

        }

    }

}
```



```
    }  
  
    else  
  
    {  
  
        MessageBox.Show("Nu se poate adauga numarul de produse deoarece stocul  
disponibil este de: " + cantitate_maxima[index]);  
  
    }  
  
    ProduseSelect.ResetText();  
  
    ProduseSelect.SelectedIndex = -1;  
  
    PopulateBon();  
  
    }  
}
```

```
private void CantitateRemove_Click(object sender, EventArgs e)  
{  
  
    if (ProduseSelect.SelectedItem == null)  
  
    {  
  
        MessageBox.Show("Selecteaza un produs");  
  
    }  
  
    else  
  
    {  
  
        int index = produse.IndexOf(ProduseSelect.SelectedItem.ToString());  
  
        if (cantitate[index] > 1)  
  
        {
```



```
cantitate[index]--;  
  
}  
  
else  
  
{  
  
    id_produs.RemoveAt(index);  
  
    produse.RemoveAt(index);  
  
    nr_marime.RemoveAt(index);  
  
    cantitate.RemoveAt(index);  
  
    pret.RemoveAt(index);  
  
    cantitate_maxima.RemoveAt(index);  
  
}  
  
ProduseSelect.ResetText();  
  
ProduseSelect.SelectedIndex = -1;  
  
PopulateBon();  
  
}  
  
}  
  
  
private void Vanzare_Click(object sender, EventArgs e)  
  
{  
  
    if (Bon.Text == "")  
  
    {  
  
        return;  
  
    }  
  
}
```



```
        if (checkBox1.Checked)

        {

            VanzareCuLivrare();

        }

        else

        {

            VanzareNormala();

        }

    }

private void checkBox1_CheckedChanged(object sender, EventArgs e)

{

    if (checkBox1.Checked)

    {

        DomiciliuPanel.Visible = true;

        resetDomiciliuPanel();

    }

    else

    {

        DomiciliuPanel.Visible = false;

    }

}
```



```
private void resetDomiciliuPanel()
```

```
{
```

```
    Nume.Text = "";
```

```
    Prenume.Text = "";
```

```
    Email.Text = "";
```

```
    Adresa.Text = "";
```

```
}
```

```
private void VanzareNormala()
```

```
{
```

```
    try
```

```
    {
```

```
        Bon.Text += Environment.NewLine + "Data: " + DateTime.Now.ToString();
```

```
        QRCodeGenerator qr = new QRCodeGenerator();
```

```
        var codeData = qr.CreateQrCode(Bon.Text, QRCodeGenerator.ECCLLevel.H);
```

```
        var code = new QRCode(codeData);
```

```
        QrBon.Image = code.GetGraphic(50);
```

```
        Image imageQR = QrBon.Image;
```

```
        MemoryStream memoryStreamQR = new MemoryStream();
```

```
        imageQR.Save(memoryStreamQR, ImageFormat.Png);
```

```
        byte[] imageByteQR = memoryStreamQR.ToArray();
```




```
MySQLCommand insertBon = new MySQLCommand("insert into bon(qrcode)
values(@qrcode)", Constante.mySqlConnection);

insertBon.Parameters.AddWithValue("@qrcode", imageByteQR);

Constante.mySqlConnection.Open();

MySQLDataReader mySqlDataReader = insertBon.ExecuteReader();

Constante.mySqlConnection.Close();

id_bon = Id_Bon(imageByteQR);

if (id_bon == 0)
{
    MessageBox.Show("Bon inexistent");
}
else
{
    for (int i = 0; i < id_produs.Count; i++)
    {
        try
        {
            MySQLCommand insertVanzare = new MySQLCommand("insert into
vanzare(id_produs,marime,cantitate,id_bon)
values(@id_produs,@marime,@cantitate,@id_bon)", Constante.mySqlConnection);

            insertVanzare.Parameters.AddWithValue("@id_produs", id_produs[i]);

            insertVanzare.Parameters.AddWithValue("@marime", nr_marime[i]);

            insertVanzare.Parameters.AddWithValue("@cantitate", cantitate[i]);

            insertVanzare.Parameters.AddWithValue("@id_bon", id_bon);
```



```
        Constante.mySqlConnection.Open();

        MySqlDataReader mySqlDataReader1 = insertVanzare.ExecuteReader();

        Constante.mySqlConnection.Close();

        insertVanzare.Parameters.Clear();

    }

    catch (Exception E)

    {

        throw E;

    }

    try

    {

        MySqlCommand update = new MySqlCommand("Update marime set
        stoc=stoc-@cantitate where id_produs=@id_produs and nr_marime=@nr_marime",
        Constante.mySqlConnection);

        update.Parameters.AddWithValue("@cantitate", cantitate[i]);

        update.Parameters.AddWithValue("@id_produs", id_produs[i]);

        update.Parameters.AddWithValue("@nr_marime", nr_marime[i]);

        Constante.mySqlConnection.Open();

        MySqlDataReader mySqlDataReader2 = update.ExecuteReader();

        Constante.mySqlConnection.Close();

        update.Parameters.Clear();

    }catch(Exception Ex)

    {
```



```
        throw Ex;

    }

}

}

}

catch (Exception ex)

{

    throw ex;

}

    Constante.Log("IntroducereVanzare",    DateTime.Now.ToString("dd-MMM-yyyy-HH-
mm-ss"), Meniu.id_utilizator);

    Save.Visible = true;

    Chestionar.Visible = true;

}

private void VanzareCuLivrare()

{

    if (Nume.Text == "")

    {

        Nume.Focus();

        MessageBox.Show("Introduceti un nume");

        return;

    }

}
```



```
if (Prenume.Text == "")  
  
    {  
  
        Prenume.Focus();  
  
        MessageBox.Show("Introduceti un prenume");  
  
        return;  
    }  
  
if (Email.Text == "")  
  
    {  
  
        Email.Focus();  
  
        MessageBox.Show("Introduceti un email");  
  
        return;  
    }  
  
if (Adresa.Text == "")  
  
    {  
  
        Adresa.Focus();  
  
        MessageBox.Show("Introduceti o adresa");  
  
        return;  
    }  
  
if (!Constante.EmailValidate(Email.Text))  
  
    {  
  
        MessageBox.Show("Introduceti un email valid");  
  
        Email.Focus();  
  
        return;  
    }
```



```
}  
  
try  
  
{  
  
    Bon.Text += Environment.NewLine + "Data: " + DateTime.Now.ToString();  
  
    QRCodeGenerator qr = new QRCodeGenerator();  
  
    var codeData = qr.CreateQrCode(Bon.Text, QRCodeGenerator.ECCLevel.H);  
  
    var code = new QRCode(codeData);  
  
    QrBon.Image = code.GetGraphic(50);  
  
    Image imageQR = QrBon.Image;  
  
    MemoryStream memoryStreamQR = new MemoryStream();  
  
    imageQR.Save(memoryStreamQR, ImageFormat.Png);  
  
    byte[] imageByteQR = memoryStreamQR.ToArray();  
  
    MySqlCommand insertBon = new MySqlCommand("insert into bon(qrcode)  
values(@qrcode)", Constante.mySqlConnection);  
  
    insertBon.Parameters.AddWithValue("@qrcode", imageByteQR);  
  
    Constante.mySqlConnection.Open();  
  
    MySqlDataReader mySqlDataReader = insertBon.ExecuteReader();  
  
    Constante.mySqlConnection.Close();  
  
    id_bon = Id_Bon(imageByteQR);  
  
    if (id_bon == 0)  
  
    {  
  
        MessageBox.Show("Bon inexistent");  
  
    }  
  
}
```



```
else

{

    for (int i = 0; i < id_produs.Count; i++)

    {

        try

        {

            MySqlCommand insertVanzare = new MySqlCommand("insert into
vanzare(id_produs,marime,cantitate,id_bon)
values(@id_produs,@marime,@cantitate,@id_bon)", Constante.mySqlConnection);

            insertVanzare.Parameters.AddWithValue("@id_produs", id_produs[i]);

            insertVanzare.Parameters.AddWithValue("@marime", nr_marime[i]);

            insertVanzare.Parameters.AddWithValue("@cantitate", cantitate[i]);

            insertVanzare.Parameters.AddWithValue("@id_bon", id_bon);

            Constante.mySqlConnection.Open();

            MySqlDataReader mySqlDataReader1 = insertVanzare.ExecuteReader();

            Constante.mySqlConnection.Close();

            insertVanzare.Parameters.Clear();

        }

        catch (Exception E)

        {

            throw E;

        }

    }

    try
```



```
{  
  
    MySqlCommand update = new MySqlCommand("Update marime set  
stoc=stoc-@cantitate where id_produs=@id_produs and nr_marime=@nr_marime",  
Constante.mySqlConnection);  
  
    update.Parameters.AddWithValue("@cantitate", cantitate[i]);  
  
    update.Parameters.AddWithValue("@id_produs", id_produs[i]);  
  
    update.Parameters.AddWithValue("@nr_marime", nr_marime[i]);  
  
    Constante.mySqlConnection.Open();  
  
    MySqlDataReader mySqlDataReader2 = update.ExecuteReader();  
  
    Constante.mySqlConnection.Close();  
  
    update.Parameters.Clear();  
  
}  
  
catch (Exception Ex)  
  
{  
  
    throw Ex;  
  
}  
  
}  
  
}  
  
  
try  
  
{  
  
    MySqlCommand insertLivrare = new MySqlCommand("insert into  
livrare(numa,prenume,email,adresa,id_bon)  
values(@numa,@prenume,@email,@adresa,@id_bon)", Constante.mySqlConnection);
```



```
insertLivrare.Parameters.AddWithValue("@nume", Nume.Text);

insertLivrare.Parameters.AddWithValue("@prenume", Prenume.Text);

insertLivrare.Parameters.AddWithValue("@email", Email.Text);

insertLivrare.Parameters.AddWithValue("@adresa", Adresa.Text);

insertLivrare.Parameters.AddWithValue("@id_bon", id_bon);

Constante.mySqlConnection.Open();

MySqlDataReader mySqlDataReader1 = insertLivrare.ExecuteReader();

Constante.mySqlConnection.Close();


string produse_cumparate = "";

for(int i = 0; i < produse.Count; i++)

{

    produse_cumparate +=produse[i] + "\nCantitate: "+cantitate[i]+"Pret:
"+pret[i]+"
";

}


MailMessage mesaj = new MailMessage();

mesaj.From = new MailAddress("licenta98@gmail.com");

mesaj.To.Add(Email.Text);

mesaj.Subject = "Comanda Nr."+id_bon;

mesaj.Body = "Comanda dumneavoastra cu numarul "+ id_bon+" a fost procesata
si urmeaza sa fie livrata.\nDetalii comanda:\nNume: "+Nume.Text+"\nPrenume:
"+Prenume.Text+"\nAdresa: "+Adresa.Text+"\n\nProdusele
dumneavoastra:\n"+produse_cumparate+"\n\nVa dorim o zi buna!";
```




```
Smtplib.Smtplib.SMTPClient smtp = new Smtplib.Smtplib.SMTPClient();

smtp.Port = 587;

smtp.EnableSsl = true;

smtp.Host = "smtp.gmail.com";

smtp.UseDefaultCredentials = false;

smtp.Credentials = new NetworkCredential("licenta98@gmail.com",
"Sora1998!");

smtp.DeliveryMethod = Smtplib.Smtplib.DeliveryMethod.Network;

smtp.Send(mesaj);
}

catch (Exception E)

{

    throw E;

}

}

catch (Exception ex)

{

    throw ex;

}

Save.Visible = true;

Chestionar.Visible = true;

}
```



```
private void Save_Click(object sender, EventArgs e)

{

    saveFileDialog1.FileName = "BonNr_" + id_bon;

    saveFileDialog1.Filter = "Images | *.png";

    saveFileDialog1.AddExtension = true;

    saveFileDialog1.DefaultExt = "*.png";

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)

    {

        QrBon.Image.Save(saveFileDialog1.FileName, ImageFormat.Png);

    }

}

private void Chestionar_Click(object sender, EventArgs e)

{

    IntroducereChestionar introducereChestionar = new IntroducereChestionar();

    introducereChestionar.ShowDialog();

}

private void IntroducereVanzare_FormClosed(object sender, FormClosedEventArgs e)

{

    if (captureDevice != null)

        captureDevice.Stop();

    timer1.Stop();

}
```



```
}
```

```
private void Nume_KeyPress(object sender, KeyPressEventArgs e)

{

    if      (char.IsDigit(e.KeyChar)      ||      char.IsWhiteSpace(e.KeyChar)      ||
char.IsPunctuation(e.KeyChar) || char.IsSymbol(e.KeyChar))

    {

        e.Handled = true;

    }

}

private void Prenume_KeyPress(object sender, KeyPressEventArgs e)

{

    if (char.IsDigit(e.KeyChar) || char.IsSymbol(e.KeyChar))

    {

        e.Handled = true;

    }

}

}
```



Anexa 2

ID	Imagine	Denumire	Producator	Tip	Sex	Pret	Distributor	QR
1		Air Structure	Adidas	Casual	Masculin	405	Footshop	
2		NMD_R1 FTW	Adidas	Casual	Masculin	789	Comodo	
3		Superstar	Adidas	Casual	Masculin	299	Comodo	

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Licenta.Modificare;

using MySql.Data.MySqlClient;

namespace Licenta



```
{

public partial class VizualizareProduse : Form

{

    public VizualizareProduse()

    {

        InitializeComponent();

    }


    public static int id_produs_redirect, pret_redirect;

    public static string denumire_redirect, producator_redirect, tip_redirect, sex_redirect, distribuitor_redirect;

    public string queryConditions = "";

    public string select= "Select id_produs as 'ID', imagine as 'Imagine', p.denumire as 'Denumire', producator as 'Producator', tip as 'Tip', sex as 'Sex', pret as 'Pret', d.denumire as 'Distribuitor', qrcode as 'QR' from produs p, distribuitor d where d.Id_distribuitor=p.Id_distribuitor ";

    private void VizualizareProduse_Load(object sender, EventArgs e)

    {

        Populare(select, queryConditions);

    }


    private void Populare(string query, string queryCond)

    {
```



```
try

{

    MySqlDataAdapter adapter = new MySqlDataAdapter(query+queryCond,
Constante.mySqlConnection);

    Constante.mySqlConnection.Open();

    DataSet ds = new DataSet();

    adapter.Fill(ds, "produs");

    dataGridView1.DataSource = ds.Tables["produs"];

    DataGridViewImageColumn imagine = new DataGridViewImageColumn();

    imagine = (DataGridViewImageColumn)dataGridView1.Columns[1];

    imagine.ImageLayout = DataGridViewImageCellLayout.Stretch;

    imagine.Width = 250;

    DataGridViewImageColumn qr = new DataGridViewImageColumn();

    qr = (DataGridViewImageColumn)dataGridView1.Columns[8];

    qr.ImageLayout = DataGridViewImageCellLayout.Stretch;

    qr.Width = 250;

    Constante.mySqlConnection.Close();

}catch (Exception ex)

{

    throw ex;

}

}
```



```
private string VerificareProdus(int id_produs)

{

    int marime, stoc;

    string marimi_disponibile = "";

    try

    {

        Constante.mySqlConnection.Open();

        MySqlCommand mySqlCommand = new MySqlCommand();

        mySqlCommand.CommandText = "Select nr_marime,stoc from marime where
id_produs=@id_produs";

        mySqlCommand.Parameters.AddWithValue("@id_produs", id_produs);

        mySqlCommand.Connection = Constante.mySqlConnection;

        MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        while (mySqlDataReader.Read())

        {

            marime = mySqlDataReader.GetInt32(0);

            stoc = mySqlDataReader.GetInt32(1);

            marimi_disponibile += "Marime: " + marime + " - Stoc: " + stoc + "\n";

        }

        Constante.mySqlConnection.Close();

        return marimi_disponibile;

    }

}
```



```
    }

    catch (Exception Ex)

    {

        throw Ex;

    }

}

private void ModificImagine_Click(object sender, EventArgs e)

{

    int rowIndex = dataGridView1.SelectedCells[0].RowIndex;

    DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];

    id_produs_redirect = Int32.Parse(Convert.ToString(selectedRow.Cells[0].Value));

    ModificareImagine modificareImagine = new ModificareImagine();

    modificareImagine.ShowDialog();

}

private void Producator_KeyPress(object sender, KeyPressEventArgs e)

{

    if      (char.IsDigit(e.KeyChar)      ||      char.IsWhiteSpace(e.KeyChar)      ||
char.IsPunctuation(e.KeyChar))

    {

        e.Handled = true;

    }

}
```




```
}  
  
}  
  
private void Aprovizionare_Click(object sender, EventArgs e)  
{  
  
    int rowIndex = dataGridView1.SelectedCells[0].RowIndex;  
  
    DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];  
  
    id_produs_redirect = Int32.Parse(Convert.ToString(selectedRow.Cells[0].Value));  
  
    Aprovizionare aprovizionare = new Aprovizionare();  
  
    aprovizionare.ShowDialog();  
  
}  
  
private void VerificareStoc_Click(object sender, EventArgs e)  
{  
  
    int rowIndex = dataGridView1.SelectedCells[0].RowIndex;  
  
    DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];  
  
    int id_produs = Int32.Parse(Convert.ToString(selectedRow.Cells[0].Value));  
  
    MessageBox.Show(VerificareProdus(id_produs));  
  
}  
  
private void cauta_Click(object sender, EventArgs e)
```



```
{  
  
    queryConditions = "";  
  
    string denumire = Denumire.Text;  
  
    string producator = Producator.Text;  
  
  
    if (denumire != "")  
    {  
        queryConditions += "and p.denumire=\"\" + denumire + \"' \"";  
    }  
  
  
    if (producator != "")  
    {  
        queryConditions += "and producator=\"\" + producator + \"' \"";  
    }  
  
  
    if (Tip.SelectedItem!=null)  
    {  
        string tip = Tip.SelectedItem.ToString();  
        queryConditions += "and tip=\"\" + tip + \"' \"";  
    }  
  
    if (Sex.SelectedItem != null)  
    {  
        string sex = Sex.SelectedItem.ToString();
```



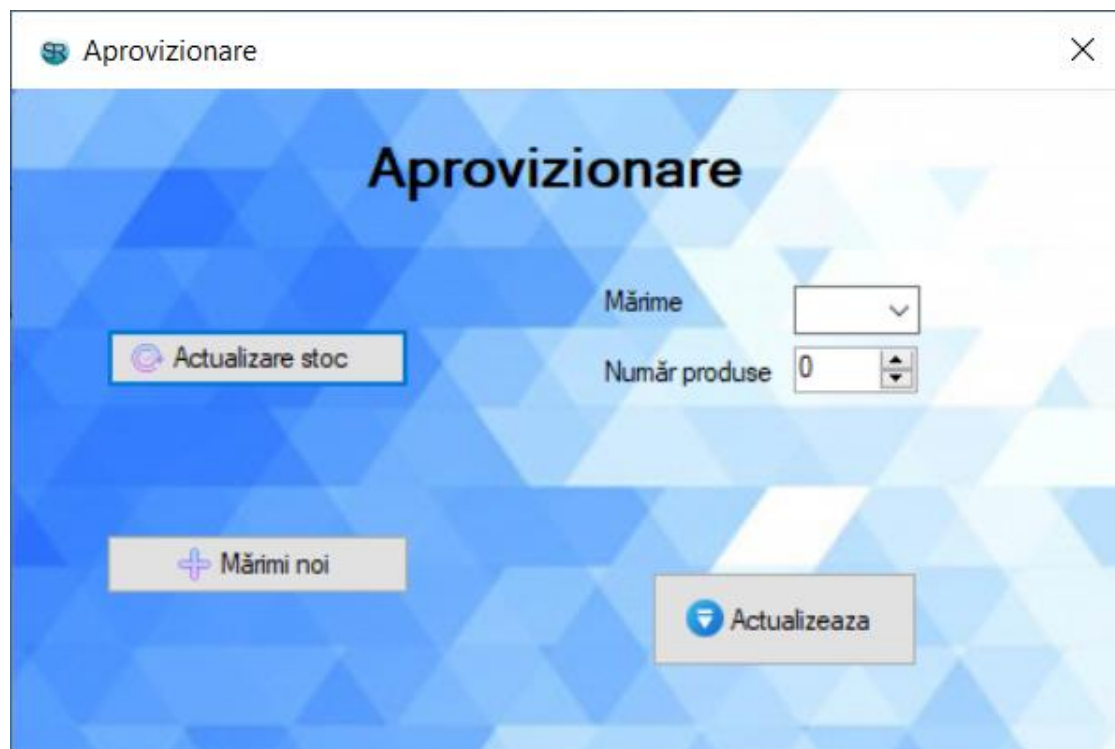
```
        queryConditions += "and sex=\" + sex + "\" ";  
    }  
  
    Populare(select, queryConditions);  
}  
  
private void ArataTot_Click(object sender, EventArgs e)  
{  
    queryConditions = "";  
    Populare(select, queryConditions);  
}  
  
private void Modifica_Click(object sender, EventArgs e)  
{  
    int rowIndex = dataGridView1.SelectedCells[0].RowIndex;  
    DataGridViewRow selectedRow = dataGridView1.Rows[rowIndex];  
    id_produs_redirect = Int32.Parse(Convert.ToString(selectedRow.Cells[0].Value));  
    denumire_redirect = Convert.ToString(selectedRow.Cells[2].Value);  
    producator_redirect = Convert.ToString(selectedRow.Cells[3].Value);  
    tip_redirect = Convert.ToString(selectedRow.Cells[4].Value);  
    sex_redirect = Convert.ToString(selectedRow.Cells[5].Value);  
    pret_redirect = Int32.Parse(Convert.ToString(selectedRow.Cells[6].Value));  
    distribuitor_redirect = Convert.ToString(selectedRow.Cells[7].Value);
```



```
ModificareProdus modificareProdus = new ModificareProdus();  
  
modificareProdus.ShowDialog();  
  
}  
  
}  
  
}
```



Anexa 3



```
using MySql.Data.MySqlClient;  
  
using System;  
  
using System.Collections.Generic;  
  
using System.ComponentModel;  
  
using System.Data;  
  
using System.Drawing;  
  
using System.Linq;  
  
using System.Text;  
  
using System.Threading.Tasks;  
  
using System.Windows.Forms;
```



namespace Licenta.Modificare

{

public partial class Aprovizionare : Form

{

public Aprovizionare()

{

InitializeComponent();

}

private void ActualizareStoc_Click(object sender, EventArgs e)

{

ActualizarePanel.Visible = true;

AdaugaPanel.Visible = false;

Marime.Items.Clear();

PopulareMarimi(VizualizareProduse.id_produc_redirect);

}

private void MarimiNoi_Click(object sender, EventArgs e)

{

AdaugaPanel.Visible = true;

ActualizarePanel.Visible = false;

MarimeExistenta.Text = "";

MarimeExistente(VizualizareProduse.id_produc_redirect);



```
}
```

```
private void PopulareMarimi(int id_produș)

{

    try

    {

        MySqlCommand mySqlCommand = new MySqlCommand();

        mySqlCommand.CommandText = "Select nr_marime from marime where
id_produș=@id_produș";

        mySqlCommand.Parameters.AddWithValue("@id_produș", id_produș);

        mySqlCommand.Connection = Constante.mySqlConnection;

        Constante.mySqlConnection.Open();

        MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        while (mySqlDataReader.Read())

        {

            int marime = mySqlDataReader.GetInt32(0);

            Marime.Items.Add(marime);

        }

        Constante.mySqlConnection.Close();

    }

    catch (MySqlException E)

    {

        throw E;

    }

}
```



```
}  
  
}  
  
private void MarimiExistente(int id_produș)  
{  
    try  
    {  
        MySqlCommand mySqlCommand = new MySqlCommand();  
        mySqlCommand.CommandText = "Select nr_marime from marime where  
id_produș=@id_produș";  
        mySqlCommand.Parameters.AddWithValue("@id_produș", id_produș);  
        mySqlCommand.Connection = Constante.mySqlConnection;  
        Constante.mySqlConnection.Open();  
        MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();  
        while (mySqlDataReader.Read())  
        {  
            MarimeExistentă.Text += mySqlDataReader.GetInt32(0) + " ";  
        }  
        Constante.mySqlConnection.Close();  
    }  
    catch (MySqlException E)  
    {  
        throw E;  
    }  
}
```




```
}
```

```
}
```

```
private bool ValitateMarime(int marime,int id_produș)

{

    try

    {

        MySqlCommand mySqlCommand = new MySqlCommand();

        mySqlCommand.CommandText = "Select  nr_marime  from  marime  where
nr_marime=@nr_marime and id_produș=@id_produș";

        mySqlCommand.Parameters.AddWithValue("@nr_marime", marime);

        mySqlCommand.Parameters.AddWithValue("@id_produș", id_produș);

        mySqlCommand.Connection = Constante.mySqlConnection;

        Constante.mySqlConnection.Open();

        MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

        if (mySqlDataReader.Read())

        {

            Constante.mySqlConnection.Close();

            return true ;

        }

        else

        {

            Constante.mySqlConnection.Close();
```



```
        return false;
    }

}

catch (MySqlException E)
{
    throw E;
}
}

private void Actualizeaza_Click(object sender, EventArgs e)
{
    if (Marime.SelectedItem != null && Stoc.Value!=0)
    {
        try
        {
            MySqlCommand mySqlCommand = new MySqlCommand();

            mySqlCommand.CommandText = "update marime set stoc=stoc+@stoc where
nr_marime=@nr_marime and id_produș=@id_produș";

            mySqlCommand.Parameters.AddWithValue("@stoc",Stoc.Value);

            mySqlCommand.Parameters.AddWithValue("@nr_marime",
Int32.Parse(Marime.SelectedItem.ToString()));

            mySqlCommand.Parameters.AddWithValue("@id_produș",
VizualizareProduce.id_produș_redirect);
```



```
mySqlCommand.Connection = Constante.mySqlConnection;

Constante.mySqlConnection.Open();

MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

Constante.mySqlConnection.Close();

Constante.Log("AprovizionareStoc", DateTime.Now.ToString("dd-MMM-yyyy-HH-
mm-ss"), Meniu.id_utilizator);

    MessageBox.Show("Actualizare realizată cu succes");

}

catch(Exception ex)

{

    throw ex;

}

}

else

{

    MessageBox.Show("Selectati o valoare");

}

}

private void Adauga_Click(object sender, EventArgs e)

{

    if (ValitateMarime(Convert.ToInt32(MarimeNoua.Value),
VizualizareProduse.id_produc_redirect))

    {
```



```
        MessageBox.Show("Marimea exista deja");

    }

    else
        if(!ValitateMarime(Convert.ToInt32(MarimeNoua.Value),
VizualizareProduse.id_produc_redirect) && StocNou.Value==0)

    {

        MessageBox.Show("Stocul nu poate fi 0");

    }

    else

    {

        try

        {

            MySqlCommand mySqlCommand = new MySqlCommand();

            mySqlCommand.CommandText = "insert into marime(nr_marime,stoc,id_produc)
values(@nr_marime,@stoc,@id_produc);";

            mySqlCommand.Parameters.AddWithValue("@nr_marime",
MarimeNoua.Value);

            mySqlCommand.Parameters.AddWithValue("@stoc", StocNou.Value);

            mySqlCommand.Parameters.AddWithValue("@id_produc",
VizualizareProduse.id_produc_redirect);

            mySqlCommand.Connection = Constante.mySqlConnection;

            Constante.mySqlConnection.Open();

            MySqlDataReader mySqlDataReader = mySqlCommand.ExecuteReader();

            Constante.mySqlConnection.Close();
```



```
        Constante.Log("AprovizionareMarimi", DateTime.Now.ToString("dd-MMM-yyyy-  
HH-mm-ss"), Meniu.id_utilizator);  
  
        MessageBox.Show("Adaugare realizată cu succes");  
  
    }  
  
    catch (Exception ex)  
  
    {  
  
        throw ex;  
  
    }  
  
    }  
  
    }  
  
    }  
  
}
```