Gandraman Iulian 331AC

# SALON DE COSMETICĂ

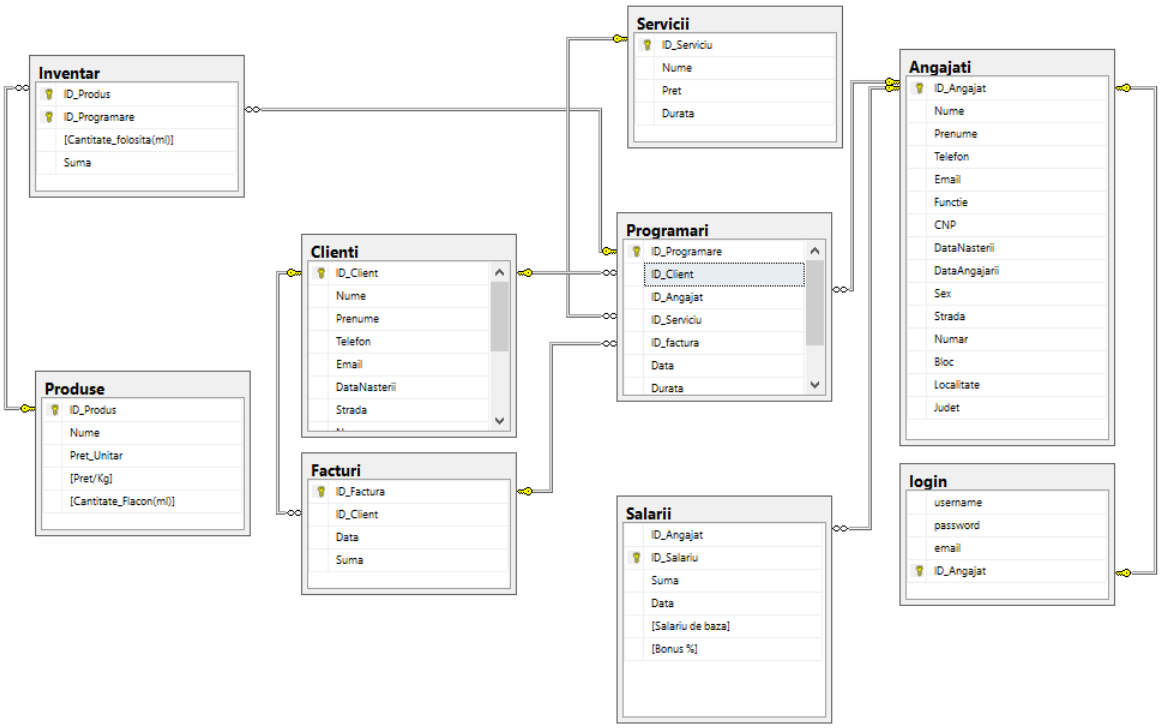
## PROIECT BAZE DE DATE

### Descrierea proiectului

În cadrul proiectului de laborator s-a propus realizarea unei baze de date in Microsoft SQL cu interfața grafică aferentă, într-un limbaj de programare vizual, pentru o interacțiune facilă utilizator – aplicație. Mai exact, se propune crearea unei aplicații complete pentru gestiunea unui salon de cosmetică și tot ce înseamnă acesta: programări, gestiunea clienților și a angajaților, produse folosite sunt doar câteva din instanțele ce trebuie gestionate.

### Arhitectura bazei de date

Pentru a proiecta o bază de date cât mai completă, aferentă scopului propus, se propune următoarea diagramă de tabele.




Tabelul **Programări** conține toate programările efectuate, iar o programare conține un client, un serviciu ce urmează a fi prestat, un angajat ce va presta serviciul, o factură aferentă programării ce urmează a fi plătită, data și durata programării, în cazul în care durata diferă de timpul prestabilit al serviciului.

Tabelul **Clienți** conține toate datele de identificare ale clienților;

Tabelul **Angajați** conține toate datele de identificare ale clienților;

Tabelul **Salarii** conține datele referitoare la plata salariilor;

Tabelul **Login** conține datele referitoare credențialele privind accesul la baza de date;

Tabelul **Produse** conține datele referitoare la produsele disponibile în salon. Acesta este legat de tabelul **Programări** printr-o relație many-to-many, întrucât mai multe produse pot fi folosite în cadrul mai multor programări. Este necesară astfel crearea unui tabel intermediar, **Inventar,** care înregistrează folosirea produselor în cadrul programărilor;

Tabelul **Servicii** conține datele referitoare la serviciile disponibile;

Tabelul **Facturi** conține datele referitoare la facturile emise.

Astfel, se definesc următoarele relații între tabele:

**Clienți – Programări:** one-to-many;

**Facturi – Programări:** one-to-many;

**Angajați – Programări:** one-to-many;

**Servicii – Programări:** one-to-many;

**Produse – Programări:** many-to-many. Apare tabelul **Inventar,** cu relațiile **Produse – Inventar:** one–to–many și relația **Inventar – Programări:** one-to-many.

**Facturi – Clienți:** one-to-many;

**Angajați – Salarii:** one-to-many;

**Angajați – Login:** one-to-one.

## Implementarea aplicației

Pentru implementarea aplicației am ales un limbaj vizual format din HTML și CSS, folosind framework-ul Bootstrap, pentru ușurința și fiabilitatea dezvoltării. Pentru partea de funcționalitate am folosit limbajele JavaScript pentru ușurința interacționării utilizatorului cu aplicația web și PHP pentru conectarea la serverul SQL.

Astfel, pentru conexiunea dintre aplicație și baza de date am folosit secvența de mai jos, ce efectuează cererea de conectare pe baza unui user și a unei parole preexistente.

```
$serverName = "GRANDMAN-TP\\sqlexpress";
$connectionInfo = array("Database" => "Programari", "UID" => "user1", "PWD" =>
"12345");
$conn = sqlsrv_connect($serverName, $connectionInfo);

if ($conn === false) {
    die(print_r(sqlsrv_errors(), true));
}
```

Pentru partea de relaționare baza de date – interfață am folosit diferite tipuri de cereri sau *queries*.

### INSERT

```
INSERT INTO Facturi (Id_Client, Data, Suma) VALUES ($id_client, $data, $suma)
```

pentru introducerea unei facturi în tabelul aferent.

```
INSERT INTO Clienti (Nume, Prenume, Telefon, Email, DataNasterii, Strada,
Numar, Bloc, Localitate, Judet, Promotii, Sex) VALUES ($nume, $prenume,
$telefon, $email, $dataNasterii, $strada, $numar, $bloc, $localitate, $judet,
$promotii, $sex)
```

pentru adăugarea unui nou client în baza de date.

```
INSERT INTO Programari (ID_Client, ID_Serviciu, ID_Angajat, Data, Durata,
Mentiuni) VALUES (%id_client, $id_serviciu, %id_angajat, %data, %durata,
%mentiuni)
```

pentru adăugarea unei noi programări în baza de date.

### DELETE

```
DELETE FROM Clienti WHERE ID_Client = $id_client
```

pentru ștergerea clientului din baza de date

```
DELETE FROM Facturi WHERE ID_Factura = ?
```

pentru ștergerea facturilor din baza de date

```
DELETE FROM Programari WHERE ID_Programare = $id_programare
```

pentru ștergerea programărilor din baza de date

## UPDATE

```
UPDATE Clienti SET Nume = $nume, Prenume = $prenume, Telefon = $telefon, Email = $email, DataNasterii = $datanasterii, Strada = $strada, Numar = $numar, Bloc = $bloc, Localitate = $localitate, Judet = $judet, Promotii = $promotii, Sex = $sex WHERE ID_Client = $id_client
```

Actualizează informații despre un anumit client

```
UPDATE Servicii SET Nume = $nume, Pret = $pret, Durata = $durata WHERE ID_Serviciu = $id_serviciu
```

Actualizează informații despre un anumit serviciu

```
UPDATE Produse SET Nume = $nume, Pret_Unitar = $pret, [Cantitate_Flacon(ml)] = $cantitate, [Pret/Kg] = $pretkg  WHERE ID_Produs = $id_produs
```

Actualizează informații despre un anumit produs

## SELECT

```
SELECT TOP 50 F.ID_Factura, F.Data, F.Suma, C.Nume, C.Prenume FROM Facturi F INNER JOIN Clienti C ON F.ID_Client = C.ID_Client WHERE C.Nume LIKE %?% AND C.Prenume LIKE %?% AND F.Suma >= ? AND F.Suma <= ? AND F.Data >= ? AND F.Data <= ? AND F.ID_Factura LIKE %?% ORDER BY ? DESC
```

Această interogare se folosește pentru a afișa diferite detalii despre facturi intr-un tabel centralizator, filtrate după anumiți parametri dați

```
SELECT Programari.*, Servicii.Nume AS Serviciu, Servicii.Pret AS ServiciuPret, Servicii.Durata as Durata, Angajati.Nume AS AngajatNume, Angajati.Prenume AS AngajatPrenume FROM Programari LEFT JOIN Servicii ON Programari.ID_Serviciu = Servicii.ID_Serviciu LEFT JOIN Angajati ON Programari.ID_Angajat = Angajati.ID_Angajat WHERE Programari.ID_Factura = $id_factura
```

Această interogare se folosește pentru a afișa pe factură, informațiile legate de serviciile prestate, ce sunt legate doar printr-un ID.

```
SELECT Facturi.*, Clienti.Nume, Clienti.Prenume, Clienti.Strada, Clienti.Numar, Clienti.Localitate, Clienti.Judet, Clienti.Telefon
          FROM Facturi
          JOIN Clienti ON Facturi.ID_Client = Clienti.ID_Client
          WHERE Facturi.ID_Factura = $id_factura
```

Această interogare se folosește pentru a afișa toate informațiile despre clientul de pe o anumită factură, informații ce sunt legate doar printr-un ID.

```
SELECT Programari.*, Inventar.[Cantitate_folosita(ml)] AS Cantitate, Produse.Nume AS Produs, Inventar.Suma AS ProdusPret
               FROM Programari
               LEFT JOIN Inventar ON Programari.ID_Programare = Inventar.ID_Programare
               LEFT JOIN Produse ON Inventar.ID_Produs = Produse.ID_Produs
```

```
               WHERE Programari.ID_Factura = %id_factura
```

Această interogare se folosește pentru a afișa pe factură, informațiile legate de produsele consumate, ce sunt legate doar printr-un ID.

```
SELECT TOP 50 S.ID_Salariu, A.Nume + ' ' + A.Prenume AS NumeAngajat,
               A.Functie, S.[Salariu de baza],
               S.[Bonus %], S.Suma,
               S.Data FROM Angajati A
JOIN Salarii S ON A.ID_Angajat = S.ID_Angajat
WHERE A.Nume LIKE ?
      AND A.Prenume LIKE ?
      AND A.Functie LIKE ?
      AND S.ID_Salariu = ?
      AND S.[Salariu de baza] >= ?
      AND S.[Salariu de baza] <= ?
      AND S.[Bonus %] >= ?
      AND S.[Bonus %] <= ?
      AND S.Suma >= ?
      AND S.Suma <= ?
ORDER BY ? ?;
```

Această interogare se folosește pentru a afișa pe un tabel centralizator toate salariile dintr-o perioada selectabilă, după anumite filtre date

```
SELECT P.ID_Programare, P.Data, P.ID_Serviciu, P.Durata, S.Durata AS
DurataServiciu, A.Nume + ' ' + A.Prenume AS NumeAngajat
    FROM Programari P
    JOIN Servicii S ON P.ID_Serviciu = S.ID_Serviciu
    JOIN Angajati A ON P.ID_Angajat = A.ID_Angajat
    WHERE P.ID_Angajat = ?
```

Această interogare se folosește în cadrul adăugării unei programări, pentru a selecta toate programările deja existente ale unui angajat și a verifica dacă se suprapun

```
SELECT TOP 50 C.ID_Client, C.Strada, C.Numar, C.Bloc, C.Localitate, C.Judet,
C.Promotii, C.Sex, C.Nume, C.Prenume, C.Telefon, C.DataNasterii,
       (SELECT COUNT(*) FROM Programari P WHERE P.ID_Client = C.ID_Client AND
P.ID_Factura IS NULL AND P.Data < GETDATE()) AS ProgramariNeplatite
FROM Clienti C
WHERE C.Nume LIKE ? AND C.Prenume LIKE ? AND C.Telefon LIKE ? AND
C.DataNasterii >= ? AND C.DataNasterii <= ? AND F.ID_Factura LIKE ?
ORDER BY ? ?;
```

Această interogare se folosește pentru a afișa centralizat clienții salonului, după anumite criterii, precum și numărul programărilor onorate dar neplătite.

```
SELECT P.ID_Programare, C.Nume + ' ' + C.Prenume AS NumeClient, P.Data, S.Nume
AS Serviciu, A.Nume +' '+ A.Prenume AS Angajat, P.ID_Client, S.Pret +
ISNULL((SELECT SUM(Suma) FROM Inventar WHERE ID_Programare = P.ID_Programare),
0) AS PretTotal
```

```
      FROM Programari P
      LEFT JOIN Clienti C ON P.ID_Client = C.ID_Client
      LEFT JOIN Servicii S ON P.ID_Serviciu = S.ID_Serviciu
      LEFT JOIN Angajati A ON P.ID_Angajat = A.ID_Angajat
      LEFT JOIN Inventar I ON P.ID_Programare = I.ID_Programare
      WHERE P.ID_Factura IS NULL"
```

Această interogare se folosește pentru a afișa pe pagina de adăugare a facturilor toate programările neplătite, pentru ușurința adăugării unei facturi

```
SELECT TOP 50 A.ID_Angajat, A.Nume, A.Prenume, A.Telefon, A.Functie,
       (SELECT COUNT(*) FROM Programari P WHERE P.ID_Angajat = A.ID_Angajat
AND P.Data > GETDATE()) AS ProgramariViitoare
FROM Angajati A
WHERE A.Nume LIKE ? AND A.Prenume LIKE ? AND A.Telefon LIKE ? AND
A.DataNasterii >= ? AND A.DataNasterii <= ? AND A.Functie LIKE ? AND
A.ID_Angajat = ?
ORDER BY ? ?;
```

Această interogare se folosește pentru a afișa atât detaliile angajaților, după anumite criterii, cât și numărul de programări ce nu au fost încă efectuate.

```
SELECT P.Nume, I.[Cantitate_folosita(ml)], I.Suma, ISNULL((SELECT SUM(Suma)
FROM Inventar WHERE ID_Programare = I.ID_Programare), 0) AS Total
                                    FROM Inventar I JOIN Produse P ON
I.ID_Produs = P.ID_Produs
                                    WHERE I.ID_Programare = ?
```

Această interogare se folosește pentru a afla toate produsele folosite în cadrul unei programări, precum și valoarea totală a acestora.

```
SELECT TOP $top A.ID_Angajat, A.Nume, A.Prenume, ISNULL(((SELECT SUM(F.Suma)
              FROM Programari P
              INNER JOIN Facturi F ON P.ID_Factura = F.ID_Factura
              WHERE P.ID_Angajat = A.ID_Angajat
              AND F.Data BETWEEN $data_min AND $data_max)
              - ISNULL((SELECT SUM(I.Suma)
                 FROM Inventar I
                 INNER JOIN Programari P ON I.ID_Programare = P.ID_Programare
                 inner join facturi F on P.ID_Factura = F.ID_Factura
                 WHERE P.ID_Angajat = A.ID_Angajat
                 AND F.Data BETWEEN $data_min AND $data_max
                 ),0)), 0) AS VenituriLunare
FROM Angajati A ORDER BY VenituriLunare DESC
```

Această interogare se folosește pentru a afișa topul angajaților și a profitului adus de aceștia într-o perioadă dată