



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE**

PROIECT

la disciplina Baze de date

Titlu

**Aplicație pentru
gestionarea lanțului de policlinici MagicHealth**

Adela Iosif, grupa 30223

Sabina Volcov, grupa 30221

An academic: 2022-2023

Cuprins

1. INTRODUCERE	3
2. MODELUL DE DATE	3
3. TABELE.....	5
4. PROCEDURI	6
5. TRIGGERE	10
6. TIPURI DE UTILIZATORI ȘI PROCEDURI SPECIFICE	11
7. CEA MAI INTERESANTĂ PROCEDURĂ	13
8. NORMALIZAREA BAZEI DE DATE	13

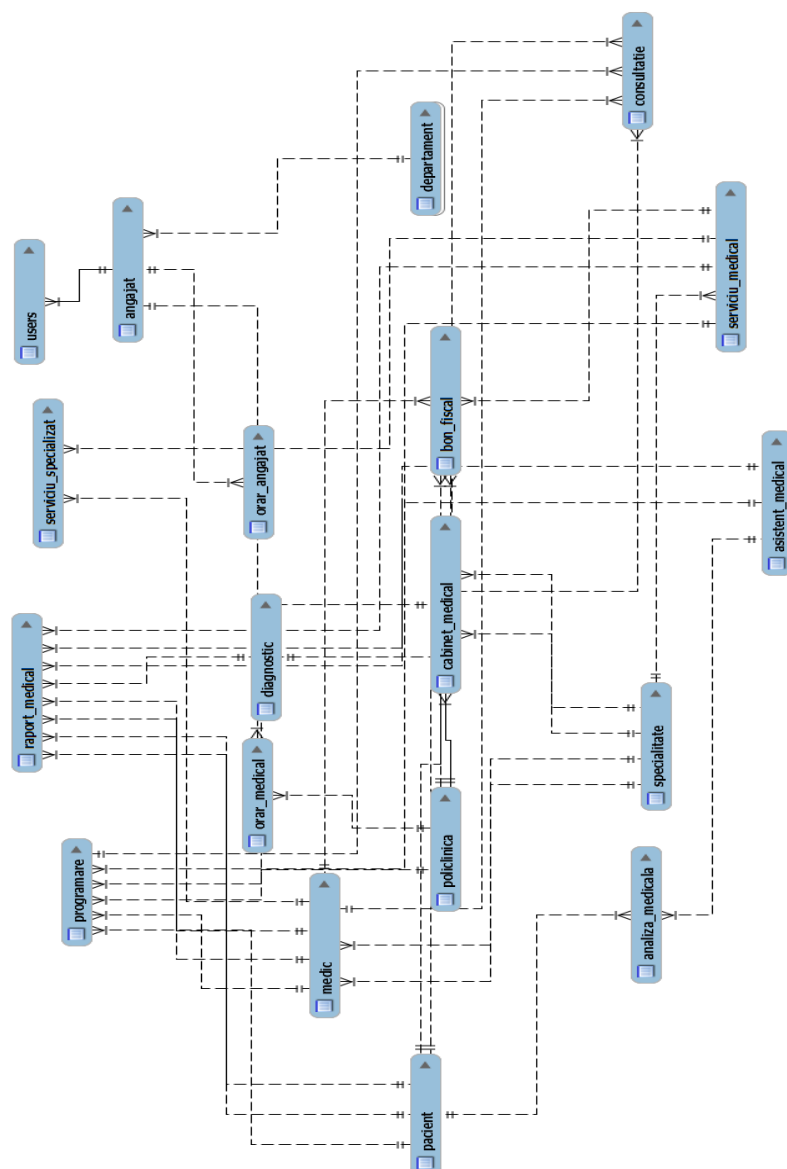
1. INTRODUCERE

Am ales implementarea aplicației pentru gestionarea unui lanț de policlinici, deoarece considerăm că atât persoanele din sistemul sanitar, cât și bolnavii, au nevoie de o aplicație ușor de utilizat, dar sigură și rapidă. De aceea, folosim o interfață grafică prietenoasă cu utilizatorul, care încearcă ghidarea lui la fiecare pas. Mai mult, fiecare tip de utilizator (administrator, super-administrator, asistent medical, contabil, inspector, medic, recepționar) are drepturi de acces specifice postului ocupat.

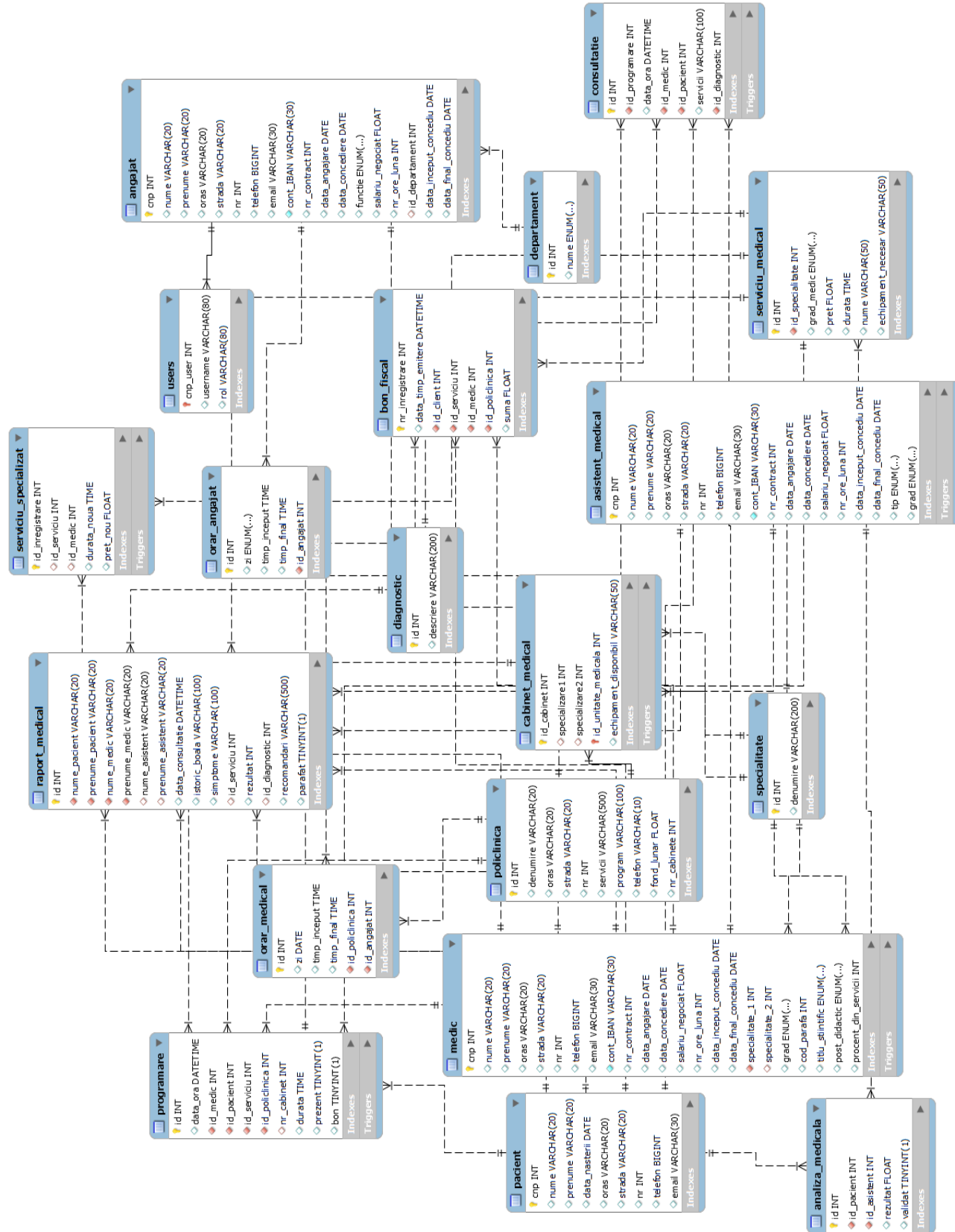
Printre funcțiile importante pe care le oferă aplicația dezvoltată de noi se numără: calcularea venitului lunar al unei policlinici din lanțul de policlinici (în baza bonurilor fiscale înregistrate), a profitul lunar al unui medic la o anumită filială, posibilitatea adăugării unui serviciu specializat (cu durată și preț specifice unui anumit medic) sau a calculării salariului unui medic în baza serviciilor medicale prestate (și nu doar a programului său).

2. MODELUL DE DATE

a. Schema UML restrânsă



b. Schema UML desfășurată



3. TABELE

- Cele 19 tabele folosite în proiect sunt:
 - policlinica
 - departament
 - specialitate
 - cabinet_medical
 - diagnostic
 - pacient
 - angajat
 - users
 - asistent_medical
 - medic
 - serviciu_medical
 - programare
 - consultatie
 - raport_medical
 - bon_fiscal
 - analiza_medicala
 - orar_medical
 - orar_angajat
 - serviciu_specializat

- Pentru a face legătura între tabele, folosim următoarele chei străine:
 - angajat - departament (id_departament)
 - serviciu medical - specialitate (id)
 - cabinet_medical - policlinica (id_unitate_medicala)
 - cabinet_medical - specialitate (specializare1)
 - cabinet_medical - specialitate (specializare2)
 - medic - specialitate (specialitate_1)
 - medic - specialitate (specialitate_2)
 - serviciu_medical - specialitate (id_specialitate)
 - programare - medic (id_medic)
 - programare - pacient (id_pacient)
 - programare - serviciu_medical(id_serviciu)
 - programare - policlinica (id_policlinica)
 - programare - cabinet_medical(nr_cabinet)
 - consultatie - programare (id_programare)
 - consultatie - medic (id_medic)

- consultatie - pacient (id_pacient)
- consultatie - diagnostic (id_diagnostic)
- raport_medical - pacient (nume_pacient)
- raport_medical - pacient (prenume_pacient)
- raport_medical - medic (nume_medic)
- raport_medical - medic (prenume_medic)
- raport_medical - asistent_medical (nume_asistent)
- raport_medical - asistent_medical (prenume_asistent)
- raport_medical - serviciu_medical (id_serviciu)
- raport_medical - diagnostic (id_diagnostic)
- bon_fiscal - pacient (id_client)
- bon_fiscal - serviciu_medical (id_serviciu)
- bon_fiscal - medic (id_medic)
- bon_fiscal - policlinica (id_policlinica)
- analiza_medicala - pacient (id_pacient)
- analiza_medicala - asistent_medical (id_asistent)
- orar_medical - policlinica (id_policlinica)
- orar_medical - angajat(id_angajat)
- orar_angajat - angajat (id_angajat)
- serviciu_specializat - serviciu_medical (id_serviciu)
- serviciu_specializat - medic (id_medic)
- users - angajat (cnp_user)

4. PROCEDURI

Procedurile se pot grupa în funcție de cele trei module ale aplicației (modulul pentru gestiunea resurselor umane, modulul de operații financiar contabile și modulul pentru gestiunea activităților operaționale).

- Proceduri caracteristice modulului pentru gestionarea operațiilor financiar contabile:

i. **CREATE PROCEDURE** venituri_lunare(an **int**, luna **int**, nr_poli **int**, OUT venit **float**)

Procedura determină venitul lunar al policlinicii care are câmpul id egal cu parametrul nr_poli pentru anul și luna indicată, returnând valoarea prin intermediul parametrului de ieșire venit. Calculul venitului se face pe baza înregistrărilor din tabela bon_fiscal, unde sunt păstrate plățile pentru serviciile medicale prestate în cadrul unității medicale. De asemenea, fiecare policlinică are un fond lunar, care se adună la sumele de pe bonuri.

- ii. **CREATE PROCEDURE** cheltuieli_lunare(an **int**, luna **int**, id_poli **int**, OUT cheltuieli **float**)



Această procedură determină cheltuielile lunare suportate de policlinica cu câmpul id egal cu parametrul id_poli, la fel pentru luna și anul indicat, returnând valoarea prin parametrul de ieșire cheltuieli. Calculul se face acum pe baza tabelii orar_medical, unde sunt păstrate informații despre orarul de lucru al angajaților implicați în activitățile operaționale, și respectiv al celor care vor primi salariu în luna respectivă. Dacă angajații sunt globali, cum ar fi inspectorii resurse umane sau contabilii, aceștia primesc salariul de la sediul central, policlinica cu numărul 1 (în cazul dat, cea situată în Cluj-Napoca). Pe lângă toate acestea, cheltuieli reprezintă și procentul primit de medici din fiecare serviciu medical prestat de aceștia, care se scade, folosind un cursor pe tabela bon_fiscal.

- iii. **CREATE PROCEDURE** profit_lunar(an **int**, luna **int**)

Procedura creează o tabelă temporară pentru afișarea profitului pe fiecare policlinică, pentru fiecare lună precedentă celei indicate, din anul transmis ca parametru. Calculul profitului se face folosind formula: profit = venit - cheltuieli, venitul calculându-se prin apelul procedurii venituri_lunare, iar cheltuielile prin apelul procedurii cheltuieli_lunare.

- iv. **CREATE PROCEDURE** profit_curent_lunar(an **int**, luna **int**)

La fel ca procedura profit_lunar, profit_curent_lunar creează o tabelă temporară, dar în care stochează ca date doar profitul pentru luna și anul dați ca parametri, pentru fiecare policlinică. Calculul se face în același mod, prin apelul celor 2 proceduri pentru venituri și cheltuieli.

- v. **CREATE PROCEDURE** profit_lunar_medic_locatie(an **int**, luna **int**, location **int**)

Procedura determină profitul lunar al fiecărui medic care a activat în anul și luna indicate, pe unitatea medicală dată prin parametrul location. Calculul se face adunând procentul din prețul fiecărui serviciu medical prestat, accesându-se tabela bon_fiscal. Exact ca și procedurile precedente, datele sunt stocate într-o tabelă temporară, apoi afișate printr-o instrucțiune SELECT.

- vi. **CREATE PROCEDURE** profit_lunar_medic_specialitate(an **int**, luna **int**, spec **varchar**(200))

Procedura calculează profitul lunar al fiecărui medic care a activat în anul și luna indicate, indiferent de locație, dar cu constrângerea că dețin specialitatea transmisă prin parametrul spec. Profitul constă în procentul obținut din prețul fiecărui serviciu medical prestat, și se salvează într-o tabelă temporară.

- Proceduri caracteristice modului de gestionare al resurselor umane:

- i. **CREATE PROCEDURE** afisare_date_personale (username **varchar**(80))

Procedura respectivă efectuează legătura dintre utilizatorul conectat la baza de date și datele sale personale, păstrate în tabela angajat. Legătura se face accesând tabela users,

unde sunt păstrate username-urile și cnp-urile angajaților, apoi se caută prin cnp datele personale.

- ii. **CREATE PROCEDURE** gasire_angajat(ume_a **varchar**(20), prenume_a **varchar**(20), functie_a **varchar**(50))

Această procedură este specifică pentru inspectorii resurse umane și constă în găsirea angajatului cu numele, prenumele și funcția menționate prin parametri. Rezultatul se obține printr-o proiecție pe tabela angajat, dacă este un angajat generic, sau medic, sau asistent_medical, dacă procedura a fost apelată pentru funcțiile respective.

- iii. **CREATE PROCEDURE** afisare_orar_angajat (ume_a **varchar**(20),
prenume_a **varchar**(20), functie_a **varchar**(50))

Procedura aplică o instrucțiune SELECT fie pe tabela orar_medical, dacă angajatul este parte din acest departament, fie pe tabela orar_angajat, unde se stochează un orar generic, pentru fiecare zi a săptămânii.

- iv. **CREATE PROCEDURE** adaugare_serviciu_specializat(serviciu **int**, cnp_medic **int**,
durata_specifica **time**, pret_specific **float**)



Procedura este utilizată de către utilizatorii de tip inspector resurse umane, care pot adăuga o specializare a oricărui serviciu medical, prin modificare prețului și duratei pentru acesta, în funcție de medicul care îl prestează. Operația care are loc este o inserare în tabela serviciu_specializat.

- Proceduri caracteristice modului pentru gestionarea activităților operaționale:

- i. **CREATE PROCEDURE** emitere_bon(data_prog **date**, data_emitere **date**,
ora **time**, pacient **int**, serviciu **int**, medic **int**)

Procedura aceasta inserează o înregistrare nouă în tabela bon_fiscal folosind datele transmise ca parametri, apoi actualizează atributul "bon" din cadrul tabelii programare cu valoarea "true", pentru a indica faptul că bonul pentru respectiva programare a fost emis.

- ii. **CREATE PROCEDURE** vizualizare_pacienti_programati(zi **date**, medic **int**)

Această procedură afișează toate înregistrările din tabela programare, pentru ziua indicată și care reprezintă servicii medicale prestate de medicul cu cnp-ul dat ca parametru.

- iii. **CREATE PROCEDURE** vizualizare_istoric(id_pacient **int**)

Procedura afișează istoricul medical al pacientului cu cnp-ul transmis prin parametrul id_pacient, folosind datele din tabela raport_medical (unde se salvează toate informațiile despre istoricul tuturor bolilor al respectivului pacient).

- iv. **CREATE PROCEDURE** completare_raport (nume_p **varchar**(20), prenume_p **varchar**(20),
nume_m **varchar**(20), prenume_m **varchar**(20), ume_a **varchar**(20),
prenume_a **varchar**(20), data_c **date**, istoric **varchar**(100), simpt **varchar**(100),


```
id_serv int, diagnostic varchar(200), recomandari varchar(50))
```

Procedura va insera o înregistrare nouă în tabela raport_medical, folosind datele transmise prin parametri. De asemenea, se va actualiza atributul „prezent” pentru programarea pentru care se completează raportul și se va adăuga o înregistrare în tabela consultație.

v. **CREATE PROCEDURE** parafare_raport (nr_raport int)

Această procedură este specifică pentru utilizatorii de tip medic și constă în parafarea raportului cu numărul dat ca parametru. Parafarea se face prin setarea valorii „true” pe atributul „parafat” al tabelii raport_medical pentru înregistrarea respectivă.

vi. **CREATE PROCEDURE** afisare_rapoarte_neparafate(cnp_medic int)

Fiind o procedură specifică medicului cu cnp-ul transmis prin cnp_medic, aceasta afișează toate rapoartele care conțin numele respectivului medic, dar încă nu au fost parafate.

vii. **CREATE PROCEDURE** vizualizare_programari_pe_cabinet(unitatea int, cabinetul int)

Procedura, la fel ca cea de vizualizare_pacienti_programati, afișează programările pentru cabinetul medical indicat, dar numai pentru datele viitoare (se verifică condiția data_ora din programare >= now()).

```
CREATE PROCEDURE creare_programare(data_pr date, timp_pr time, p_id_pacient int,  
p_id_medic int, p_id_serviciu int, p_id_cabinet int, p_oras varchar(20))
```

viii.



Această procedură este caracteristică pentru utilizatorii de tip recepționar și în cadrul ei se verifică mai multe condiții: existența și corectitudinea locației medicale și a cabinetului medical indicat ca parametri, existența specializării necesare pentru prestarea serviciului medical de către medic și în cabinetul medical ales, prezența echipamentului necesar, dar și nesuprapunerea intervalului orar cu alte programări, nici ale medicului, nici ale pacientului. La final, dacă toate verificările au fost trecute cu succes, se înserează o înregistrare nouă în tabela programare.

- Proceduri pentru administrator:

i. **CREATE PROCEDURE** user_nou(cnp int,
functie varchar(50), username_nou varchar(80), parola_user varchar(80))

Procedura va insera în tabela users noul utilizator, care reprezintă un angajat care trebuie deja să existe în cadrul bazei de date. Angajatul este căutat folosind parametrii cnp și/sau funcție, apoi îi este setat noul username și parola, apoi i se grantează privilegiile necesare pentru a putea accesa datele corecte în timpul utilizării aplicației.

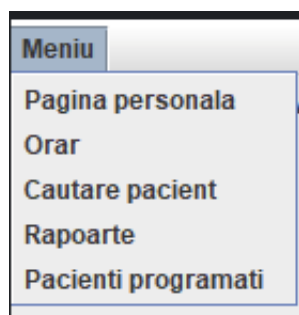
5. TRIGGERE

- **CREATE TRIGGER** adaugare_asistent **BEFORE INSERT ON** asistent_medical
Acest trigger va adăuga datele despre asistentul medical adăugat și în tabela angajat.
- **CREATE TRIGGER** adaugare_medic **BEFORE INSERT ON** medic
Trigger-ul va adăuga datele despre medicul nou-adăugat și în tabela angajat.
- **CREATE TRIGGER** stergere_asistent **BEFORE DELETE ON** asistent_medical
În caz că un asistent medical este șters din baza de date, datele despre acesta sunt șterse și din tabela generică angajat.
- **CREATE TRIGGER** stergere_medic **BEFORE DELETE ON** medic
Dacă un medic este șters din baza de date, datele sale sunt șterse și din tabela generică angajat.
- **CREATE TRIGGER** update_asistent **BEFORE UPDATE ON** asistent_medical
Pentru a sincroniza modificările, datele modificate ale unui asistent medical sunt modificate și în tabela angajat.
- **CREATE TRIGGER** update_medic **BEFORE UPDATE ON** medic
La fel ca în trigger-ul precedent, datele modificate despre un medic sunt modificate și în tabela angajat.
- **CREATE TRIGGER** consult **BEFORE INSERT ON** consultatie
Acest trigger actualizează câmpul "prezent" din cadrul tabelii programare, în momentul în care a avut loc o consultație (deoarece pacientul a fost prezent).
- **CREATE TRIGGER** orar_medic **BEFORE INSERT ON** programare
Acest trigger actualizează orar unui medic în momentul în care a fost creată o programare pentru un serviciu medical pe care îl prestează.
- **CREATE TRIGGER** cabinet_nou **BEFORE INSERT ON** cabinet_medical
Trigger-ul dat actualizează numărul de cabinete medicale pe care îl are policlinica indicată prin atributul id_unitate_medicala, atunci când este creat.
- **CREATE TRIGGER** duplicate_servicii_specializate **BEFORE INSERT ON** serviciu_specializat
Acest trigger elimină înregistrările precedente din tabelul serviciu_specializat, în momentul în care pentru același serviciu și același medic s-a păstrat deja o valoare specializată pentru preț și/sau durată.

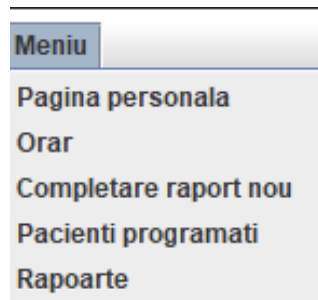
6. TIPURI DE UTILIZATORI ȘI PROCEDURI SPECIFICE

În baza de date sunt înregistrate 7 tipuri de utilizatori: medic, asistent medical, recepționar, contabil, inspector resurse umane, administrator și super-administrator, fiecare cu acces la proceduri diferite. Toate procedurile caracteristice sunt disponibile din meniul din colțul stânga-sus al aplicației. O pagină comună pentru toate tipurile de utilizatori este cea de date personale, care se deschide după ce s-a efectuat cu succes autentificarea.

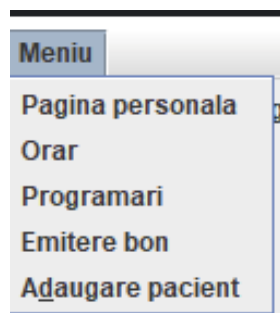
- Procedurile specifice pentru un utilizator de tip medic:
 - vizualizarea orarului personal
 - căutarea unui pacient după nume și afișarea istoricului medical al acestuia
 - vizualizarea rapoartelor neparafate, apoi prin dublu click, deschiderea detaliilor pentru oricare dintre acestea, cu posibilitatea de parafare a raportului
 - vizualizarea pacienților programați pentru ziua în curs



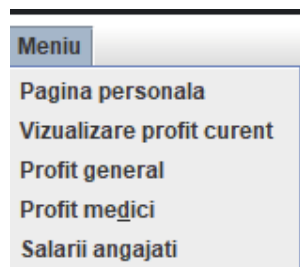
- Procedurile specifice pentru un utilizator de tip asistent medical:
 - vizualizarea orarului personal
 - completarea unui raport nou
 - vizualizarea pacienților programați pentru ziua în curs
 - vizualizarea rapoartelor deja completate de respectivul asistent



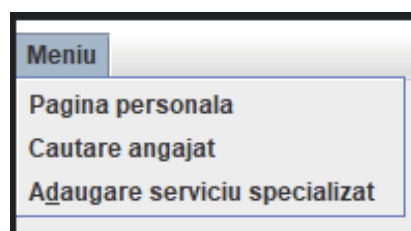
- Procedurile specifice unui utilizator de tip recepționar:
 - vizualizarea orarului personal
 - vizualizarea programărilor pentru ziua în curs (cu opțiune de vizualizare a programărilor pentru un cabinet medical specific și crearea unei noi programări)
 - emiterea unui bon nou (alegând din listă programarea pentru care se face emiterea)
 - adăugarea unui nou pacient în sistem



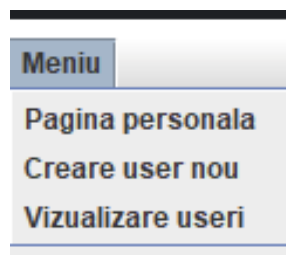
- Procedurile specifice pentru un utilizator de tip contabil:
 - vizualizarea profitului curent (pentru luna în curs) pe fiecare policlinică în parte
 - vizualizarea profitului general al tuturor policlinicilor (pentru toate lunile din an până la cea indicată)
 - vizualizarea profitului medicilor (fie pe locație medicală, fie pe specialitate)
 - vizualizarea salariilor tuturor angajaților



- Procedurile specifice unui utilizator de tip inspector resurse umane:
 - căutarea unui angajat după nume, prenume și funcția ocupată în cadrul lanțului de policlinici (cu opțiune pentru vizualizarea orarului și concediului acestuia, dacă este găsit)
 - adăugarea unui serviciu specializat (dublu click pe unul dintre serviciile disponibile din listă)



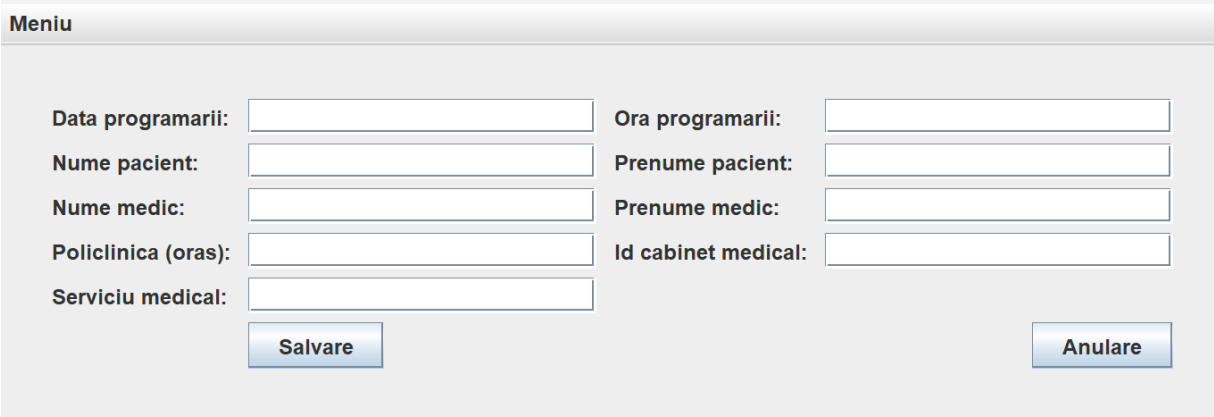
- Procedurile specifice unui utilizator de tip administrator/super-administrator:
 - vizualizarea tuturor utilizatorilor din sistem (+ ștergerea oricărui utilizator din listă prin dublu click)
 - crearea unui nou utilizator



7. CEA MAI INTERESANTĂ PROCEDURĂ

```
CREATE PROCEDURE creare_programare(data_pr date, timp_pr time, p_id_pacient int,  
p_id_medec int, p_id_serviciu int, p_id_cabinet int, p_oras varchar(20))
```

Parametri procedurii sunt introduși de la tastatură de recepționar:



The screenshot shows a window titled "Meniu" with a light gray background. It contains several input fields arranged in two columns. The left column has labels: "Data programarii:", "Nume pacient:", "Nume medic:", "Policlinica (oras):", and "Serviciu medical:". The right column has labels: "Ora programarii:", "Prenume pacient:", "Prenume medic:", and "Id cabinet medical:". Each label is followed by a white rectangular input field. At the bottom of the window, there are two buttons: "Salvare" on the left and "Anulare" on the right, both with a blue gradient and white text.

În procedură au loc verificări ale consistenței informațiilor furnizate. Astfel, ne asigurăm că :

- data și ora sunt cândva în viitor
- policlinica introdusă există, iar cabinetul selectat aparține policlinicii indicate
- medicul are specialitatea necesară pentru a presta serviciul dat, iar cabinetul are specializarea și echipamentul adecvat
- programările introduse nu se suprapun nici pentru medic, dar nici pentru pacient sau cabinet

Pe parcursul scrierii acestei proceduri am întâmpinat unele dificultăți în verificarea suprapunerilor programelor (pentru medic, pacient și cabinet), reușind să rezolvăm aceste constrângeri cu ajutorul funcțiilor DATE_ADD, TIME_TO_SEC și a celor de cast și concatenare (cast, concat). Un alt lucru interesant căruia i-am găsit soluția ar fi transmiterea semnalelor de excepție din Sql către Java, pentru a arăta că datele introduse nu sunt valide, sau alte erori (SIGNAL SQLSTATE '45000').

8. NORMALIZAREA BAZEI DE DATE

Baza de date dezvoltată de noi se află în nivelul de normalizare Boyce-Codd (BCNF), prin care am încercat minimizarea redundanțelor și sporirea integrității datelor. De aceea, în baza noastră de date:

- Fiecare atribut este atomic și nu depinde de alte atribute (cu excepția câtorva câmpuri datetime)
- Fiecare atribut reprezintă o singură informație
- Avem atribute chei, reprezentate de id-ul unic al tabelor
- Cheia primară este suficientă ca să determine celelalte atribute ale tabelor
- Cheia primară este singurul atribut care determină celelalte atribute, nu există un alt atribut care poate determina în mod unic o tuplă