UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE DEPARTAMENTUL CALCULATOARE

PROIECT

La Disciplina

INTRODUCERE ÎN BAZE DE DATE

TITLUL LUCRĂRII:

Managementul Lanțului de Policlinici Apollo

PROF. COORDONATOR: COSMINA IVAN

AUTORI:

Botiş George

Francioli Daria

An academic 2021-2022

grupa 30221

CUPRINS

1. INTRODUCERE

Proiectul intitulat „Managementul Lanțului de Policlinici Apollo” are scopul de a simula un sistem de management pentru un lanț de policlinici din mai multe orașe(Baia Mare, Baia Sprie, Cluj-Napoca, Bucureşti şi Haţeg), fiecare având:

- program specific,
- tipuri diferite de cabinete,
- tipuri diferite de angajați cu grad şi specializări
- echipamente specifice fiecărui cabinet

Fiecare tip de angajat deține un cont personal de logare, iar în funcție de postul pe care îl ocupă, are acces la diferite operațiuni în funcţie de un rol şi permisiuni: realizarea programărilor, vizualizarea datelor financiare cu privire la unitatea medicală în care lucrează, vizualizarea sau editarea unor date despre angajații policlinicii, chiar şi editarea datelor personale .

Ne-am gândit la o implementare cât mai apropiată de cea a site-urilor, precum: https://www.reginamaria.ro/ şi https://policlinica-baiamare.ro/.

Am creat această interfaţă pentru a facilita comunicarea dintre pacient şi angajaţi, pentru a primi buletinul de analize, observaţii şi factura, economisind timp, având acces constant la tot, chiar dacă dorește doar să se informeze sau să se programeze la un consult.

2. ANALIZA CERINȚELOR ŞI SPECIFICAȚIILE DE PROIECT

Se doreşte implementarea unui sistem informatic destinat gestiunii activităţilor dintr-un lanț de policlinici.

Lanțul de policlinici este format din mai multe unități medicale, fiecare fiind caracterizată prin denumire, adresă, descrierea serviciilor oferite și programul de funcționare, pentru fiecare zi a săptămânii.

Aplicaţia va trebui să utilizeze un sistem de gestiune pentru baze de date MySQL, iar interacţiunea cu acesta va fi realizată doar prin interfaţa grafică .

Funcţionalităţile pe care le va oferi programul vizează operaţii ce ţin de gestiunea angajaţilor, serviciul financiar-contabil şi administrarea operațiilor curente din cadrul policilinicii (gestiunea pacienților programați, completarea unui raport medical, emiterea unui bon fiscal).

Aplicaţia va putea fi accesată, pe baza unui proces de logare, de către mai multe tipuri de utilizatori(pacient, angajat),

operând în departamentele resurse umane, financiar-contabil sau medical. Pentru fiecare tip de utilizator se vor reţine informații precum

- CNP,
- nume,
- prenume,
- adresa,
- număr de telefon,
- email, cont IBAN,
- data angajării,
- funcţia deţinută în cadrul lanțului de policlinici.

Totodată,programul trebuie să ofere și o funcționalitate pentru deautentificare(LOGOUT), prin care se revine la fereastra care solicit datele de acces, astfel încât și un alt utilizator să îl poată folosi ulterior, fără a fi necesară repornirea sa.

Un angajat poate să fie:

i) Medic

- își poate vizualiza datele, salariul și profitul personal
- poate vizualiza pacienții programați în ziua respectivă
- poate completa raportul medical al pacientului

ii) Asistent medical

- își poate vizualiza datele personale și salariul
- completează buletinul de analize(trimite analizele si bonul fiscal)

iii) Recepționer

- își poate vizualiza datele personale și salariul
- realizează programări şi emite bonuri fiscale

iv) Expert contabil (FR)

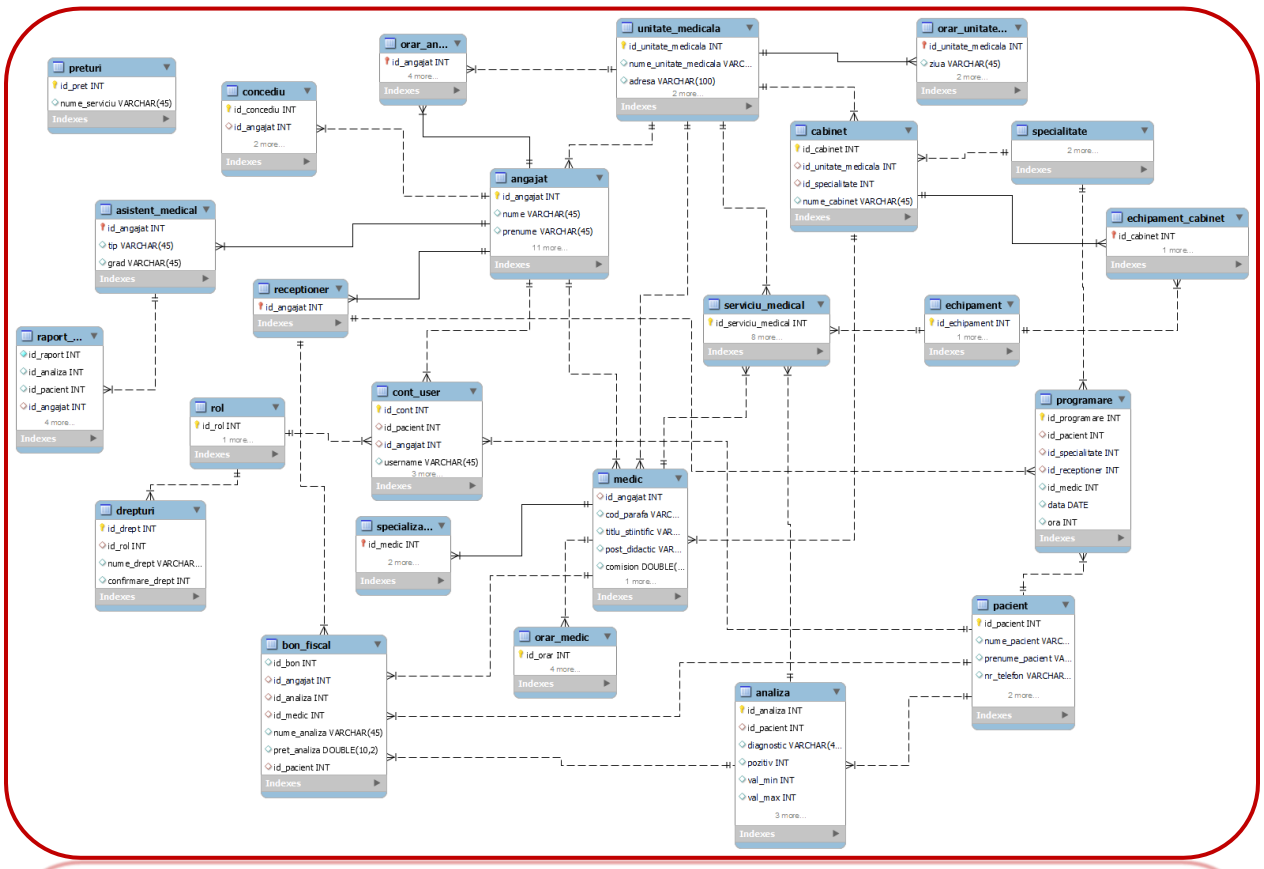
- își poate vizualiza datele personale și salariul propriu
- are acces la datele referitoare la profitul unității medicale, salariile angajaților, profitul generat de medici
- poate vizualiza tranzacțiile bancare ale policlinicii

v) Inspector resurse umane (HR)

- își poate vizualiza datele personale și salariul propriu
- adaugă/șterge/caută un angajat
- verifică concediile acestuia
- poate edita orarul și concediile
- poate edita detaliile unui angajat

La nivel structural proiectul este alcătuit din 24 de tabele create în limbajul SQL, folosind mediul de dezvoltare MySQL Workbench. Interfața grafică este realizată în limbajul Java, folosindu-se IDE-ul IntellIj de la JetBrains.

3. MODELUL DE DATE ȘI DESCRIEREA ACESTUIA



APOLLO

Dintre cele 24 de tabele, 5 dintre ele sunt:

- **Angajat** - stochează informații precum: nume, prenume, adresa, nr_telefon, email, salariu, funcţie, nr_ore prevăzute în contractul de muncă, data angajării, iban, cnp, id_unitate_medicală, id_rol;
  Acest tabel este în relație cu medic, cont_user, asistent medical, recepţioner, orar?angajat şi concediu ;

- **Pacient** - stochează informații precum: nume, prenume, nr_telefon, email, adresa;

- **Cont User** - stochează informații precum: username, parola, id_cont, id_pacient, id_angajat, id_rol;

Fiecare user va avea un id diferit , Pacientul va avea 0, iar angajaţii vor avea numere de la 1 la 6.

De exemplu, Medicul va avea 1, iar Contabilul va avea 4. Acest lucru a fost folosit pentru accesul restrâns şi politica de confidenţialitate. Medicul nu poate sa vadă toate salariile celorlalţi angajaţi, dar contabilul da.

- **Medic** - stochează informații detaliate particulare despre medicii unității medicale: CNP, parafă, comision, titlu științific, titlu didactic.
  Acest tabel este în relație cu serviciu_medical, angajat, specializare, orar_medic, programare;

- **Programare** - stochează informații precum: id_pacient, data, ora, specialitate, nume medic;
  Pacientul poate să îşi facă o programare la o anumită specialitate, apoi va fi direcţionat la lista de medici, data şi ora libere.

Procedurile stocate în baza de date verifică corectitudinea datelor trimise prin parametrii, precum existența unui angajat la ștergerea acestuia, verificarea disponibilității unui medic pentru o programare sau programul policlinicii atunci când se modifică orarul unui angajat.

Din cele 16 proceduri ale bazei de date, cele mai complexe sunt:

- procedure stergere_orar_angajat(in `_id_angajat` int, in `_id_unitate_medicala` int, in `_ziua` varchar(45), out `confirmare` int)

Această procedură este folosită în cazul în care un medic părăsește unitatea medicală din diferite motive: fie își dă demisia, sau este concediat.

- create procedure inserare_concediu(in `_id_angajat` int, in `_data_incepere` date, in `_data_final` date, out `confirmare` int)

Această procedură este folosită în cazul în care un anumit angajat vrea să își ia concediu, complexitatea a fost dată de faptul că trebuia verificat in programare, dacă există una in acel interval, dacă da, să o șteargă.

- create procedure confirmare_raport_medical(in `_id_raport` int, in `_cod_parafa` varchar(45), out `confirmare` int)

Această procedură este folosită când raportul medical este complet, astfel medicul îl confirmă, punând parafa pe hârtie.

- create procedure profit_per_unitate(in `_id_unitate_medicala` int, out `profit` int (10))

Această procedură este folosită astfel: însumează toate prețurile serviciilor medicale, iar la final scade suma salariilor angajaților

- create procedure adaugare_programare(in `_id_pacient` int, in `_id_specialitate` int, in `_id_receptioner` int, in `_id_medic` int, in `_data` date, in `_ora` int, out `confirmare` int)

Această procedură este folosită astfel: dacă am găsit specialitatea, pacientul și medicul în baza de date, iar data și ora sunt libere.

Trigger-ele create sunt secvențe de cod care execută operații de acces la date asupra tabelelor. Trigger-ele sunt utile deoarece permit implementarea regulilor de business specifice aplicației, în mod direct, atașat bazei de date.

Sunt pre-filtre sau post-filtre care pot să împiedice sau să execute anumite acțiuni asupra datelor.

Cele 7 triggere ale bazei de date sunt:

- create trigger stergere_orar_medic– trigger pentru ștergerea orarului unui medic

```
create trigger  stergere_orar_medic before delete on `orar_medic` for each row
begin
    if ((select `functie` from `angajat` where `id_angajat` = old.`id_medic`) like 'Medic') then
        delete from `programare` P where P.`id_medic`=old.`id_medic` and dayname(P.`data`)=old.`ziua` and old.`ora_incepere` >= P.`ora` and old.`ora_final` <= P.`ora`;
    end if;
end;
```

- create trigger editare_orar_medic – trigger pentru editarea orarului unui medic

```
create trigger  editare_orar_medic before update on `orar_medic` for each row
begin
    if ((select `functie` from `angajat` where `id_angajat` = old.`id_medic`) like 'Medic') then
        delete from `programare` P where P.`id_medic`=old.`id_medic` and dayname(P.`data`)=old.`ziua` and `ora_incepere` >= P.`ora` and `ora_final` <= P.`ora`;
    end if;
end;
//delimiter ;
```

- create trigger introducere_concediu – trigger pentru inserarea concediului

```
create trigger introducere_concediu after insert on `concediu` for each row
begin
    if((select `functie` from `angajat` where `id_angajat` = new.`id_angajat`) like 'medic') then
    delete from `programare` P where P.`id_medic` = new.`id_angajat` and P.`data` >= new.`data_incepere` and P.`data` <= new.`data_final`;
end if;
end;
// delimiter ;
```

- create trigger inserare_angajat – trigger pentru crearea contului unui angajat

```
create trigger inserare_angajat after insert on `angajat` for each row
begin
    set @id_rol = 0;
    if(new.`functie` like 'Medic') then #100 ore
    set @id_rol = 1;
    end if;
    if(new.`functie` like 'Asistent Medical') then #150 ore
    set @id_rol = 2;
    end if;
    if(new.`functie` like 'Receptioner') then #120
    set @id_rol = 3;
    end if;
    if(new.`functie` like 'Inspector Resurse Umane') then #120
    set @id_rol = 4;
    end if;
    if(new.`functie` like 'Expert Financiar Contabil') then #50
    set @id_rol = 5;
    end if;

end;
```

- create trigger stergere_angajat – trigger pentru ștergerea unui angajat

- create trigger stergere_pacient – trigger pentru ștergerea unui pacient

Vederile create salvează interogările pe care vrem să le utilizăm în cod ori de câte ori este necesar. Prin utilizarea vederilor simplificăm și specializăm viziunea utilizatorilor asupra datelor din baza de date. Vederile pot fi folosite in cereri SQL simple cand de fapt ele intorc rezultatul unor cereri SQL complexe.

CREATE VIEW `policlinica`.`vedere_raport_medical` = informații despre fișa pacienților

CREATE VIEW `policlinica`.`vedere_cabinet` = informații despre fiecare cabinet

CREATE VIEW `policlinica`.`vedere_preturi` = informații despre prețuri

CREATE VIEW `policlinica`.`vedere_medic` = view cu informații despre medic

CREATE VIEW `policlinica`.`vedere_programare` = view cu informații despre programări

CREATE VIEW `policlinica`.`vedere_concediu` = informații despre concediul angajaților

CREATE VIEW `policlinica`.`vedere_angajat` = view cu informații despre angajați

CREATE VIEW `policlinica`.`vedere_serviciu_medical` AS SELECT `pret`, `id_unitate_medicala`, `id_medic` FROM `serviciu_medical`; = view cu informații despre serviciile oferite

CREATE VIEW `policlinica`.`vedere_analiza` = view cu informații despre analize

CREATE VIEW `policlinica`.`vedere_unitate_medicala ` = informații despre unitățile medicale

Baza de date respectă atributele fiecărui tabel .Fiecare table are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date. Cheile străine sunt atribute sau un set de atribute dintr-o relaţie care are corespondent o cheie candidat a altei relaţii (sau chiar a relaţiei curente, caz în care o numim cheie străină recursivă).

Obţinerea de informaţii din baza de date se realizează cu ajutorul interogărilor SQL. Rezultatul unei interogări este un nou tabel.

INTEROGĂRILE:

```
SELECT nume, prenume
FROM pacient ORDER BY nume, prenume ASC;
```

Ordonează numele pacienților în ordine crescătoare

```
SELECT nume, prenume
FROM angajat
ORDER BY nume, prenume DESC;
```

Ordonează numele angajaților în ordine descrescătoare

```
SELECT nume, prenume, salariu
FROM angajat
WHERE salariu > 3000;
```

Angajații care au salariu mai mare de 3000 lei

```
SELECT nume, prenume, data_angajare AS Vechime
FROM angajat
ORDER BY Vechime ASC;
```

Angajații, ordonați după vechime

```
SELECT nume, prenume
FROM angajat
WHERE id_rol = 1;
```

Angajații cu ID-ul 1(medici)

4. DETALII DE IMPLEMENTARE

Nicio aplicație nu poate exista fără o interfață grafică care să faciliteze accesul utilizatorului(pacient sau angajat) la date și specificații. Proiectul nostru implementează o interfață grafică în Java.

Realizarea ei a constat in folosirea unor clase de obiecte, pentru fiecare modul cerut:

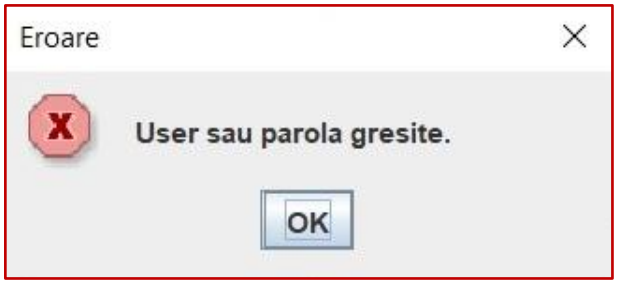
Sistemul va fi format din mai multe module care vor putea fi accesate de angajaţi în funcţie de drepturile pe care le deţin. Astfel, vor fi implementate un modul pentru **gestiunea resurselor umane(HR)** ce vizează gestiunea programului de lucru şi al concediilor angajaţilor, un modul pentru **operaţii financiar-contabile(FR)** care determină profitul operaţional ca diferenţă între venituri (sume încasate pentru serviciile medicale) şi cheltuieli (plăţi efectuate către angajaţi aferente salariilor) și un **modul pentru gestiunea activităților operaționale(MR)** (programarea pacienților pentru servicii medicale și înregistrarea acestora în momentul în care se prezintă în clinica medicală, emiterea bonului fiscal de către recepționeri, completarea rapoartelor medicale de către asistenții medicali și medici).

Toate aceste module vor fi integrate în cadrul aceluiași sistem informatic, sub forma unor meniuri care vor conține funcționalitățile pe care acestea le oferă, disponibilitatea lor fiind însă limitată și de permisiunile pe care le posedă utilizatorul autentificat la momentul respectiv de timp.

Instrucțiuni de utilizare

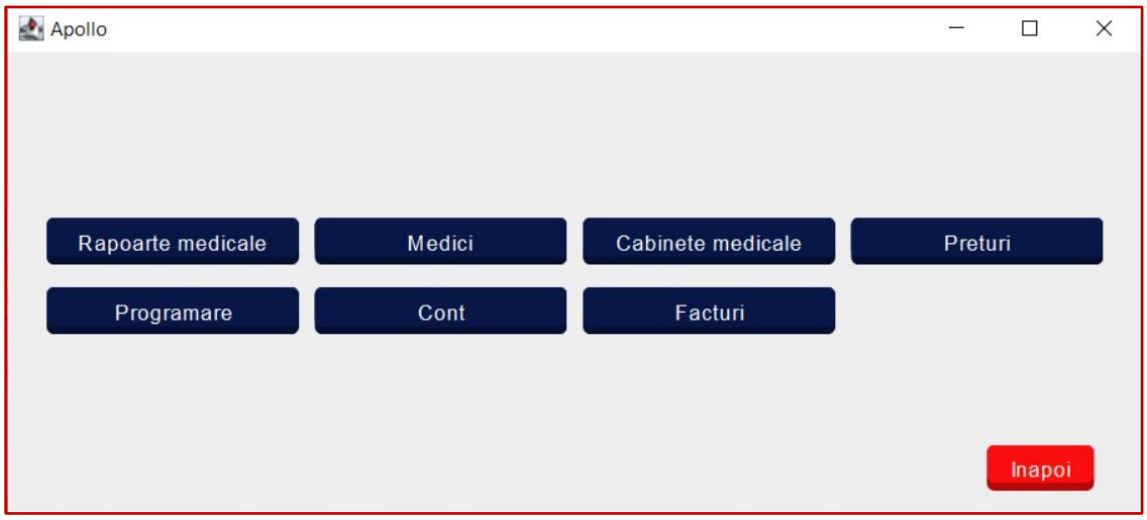
Pentru utilizarea aplicației utilizatorul își va introduce datele contului în fereastra de mai jos:

Dacă parola sau username-ul sunt sunt greșite, atunci va aparea mesajul de eroare:

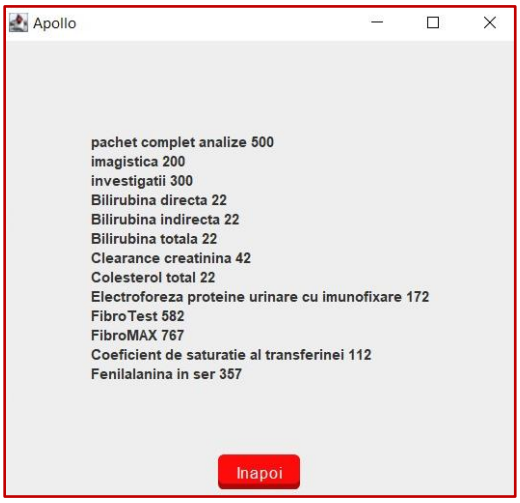


Dacă acestea sunt corecte, în funcție de id-ul său, aplicația va depista dacă utilizatorul este un pacient sau angajat.

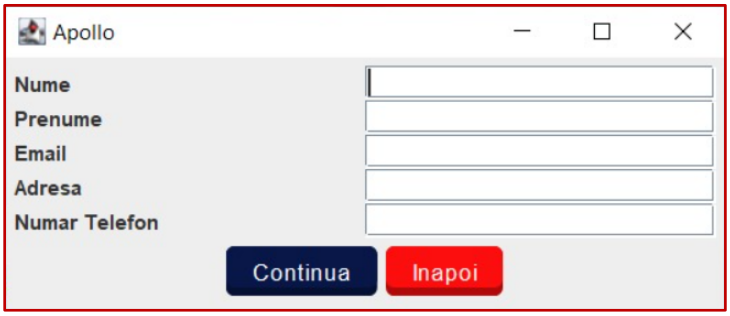
În primul caz, când este **PACIENT**, se va deschide următoarea fereastră:



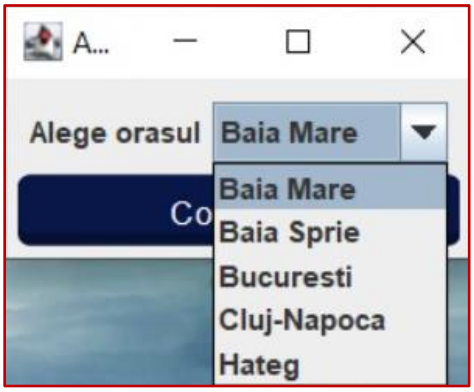
Aici, pacientul are posibilitatea să vadă listele cu: medici, prețuri, cabinete medicale, rapoarte medicale și facturi.



La Cont, își poate vedea datele personale și le poate modifica după preferințe.



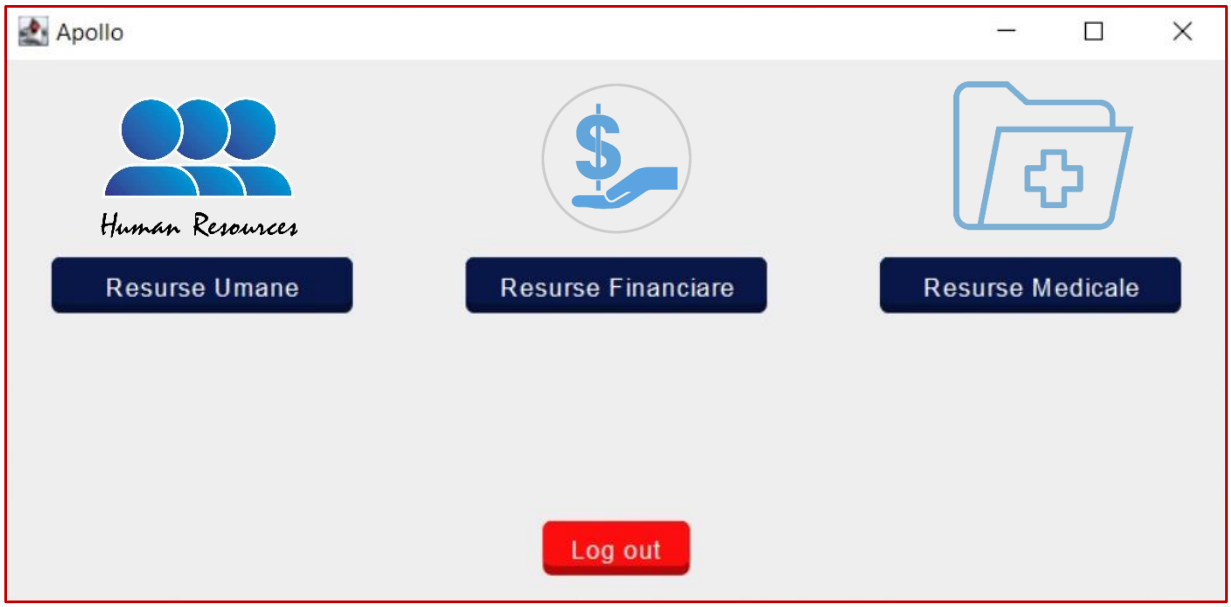
La Programare va putea să își creeze o nouă programare, unde se va deschide fereastra:

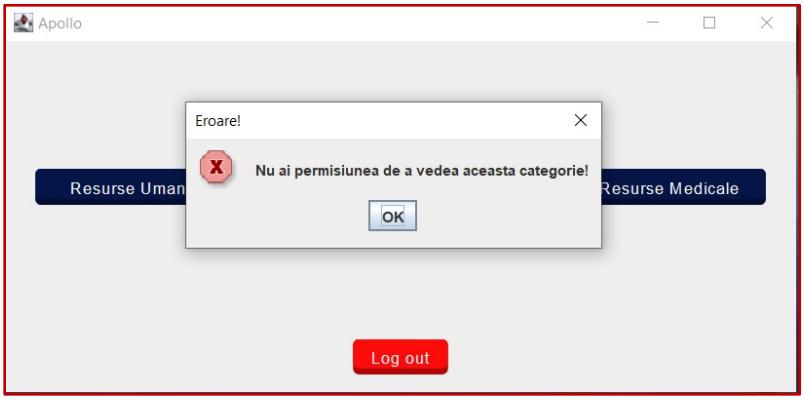


Pacientul își poate alege cea mai apropiata clinică pentru programare, după aceea va aparea meniul de specialitate:

Apoi, va putea alege medical dorit din listă, alături de data și ora dorită(dacă acestea sunt disponibile bineînțeles):

În al doilea caz, când este **ANGAJAT**, se va deschide următoarea fereastră:





Inspectorul de resurse umane nu are acces decât la butonul său, unde vor părea următoarele butoane în următoarea fereastră.

Acesta poate vedea și edita programările și concediile angajaților.

Asemenea, la resurse financiare.

Contabilul poate vedea salariile și istoricul tranzacțiilor efectuate în unitățile medicale.

La resurse medicale, va apărea un formular unde se poate căuta pacientul după nume, prenume și CNP.

Apoi, medicul sau asistentul medical poate trimite analiza pacientului, iar recepționerul emite bonul în funție de acestea.

Pentru delogare , utilizatorul va folosi butoanele Inapoi, pana la meniul principal, unde este butonul de delogare.

5. CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE

În concluzie, aplicația simulează gestiunea sistemului unui lanț de policlinici, iar pe viitor se pot adăuga mai multe orașe de funcționare, un site pe internet, un cloud care să rețină toți angajații, pacienții ca o arhivă pentru a face o statistică anuală a profitului fiecărei policlinici și evoluția veniturilor și a cheltuielilor.

Se dorește o actualizare a codului în Java și a bazei de date pentru a fi mai eficiente, adăugând algoritmi optimi pentru programări, calculul salariilor etc.

Tot odată, se poate adăuga un calendar cu toate programările deja făcute de exemplu: