Logo

Description automatically generated

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE,

DEPARTAMENTUL CALCULATOARE

**PROIECT**

LA DISCIPLINA

BAZE DE DATE

TITLUL LUCRĂRII:

**GESTIONAREA SISTEMULUI DE POLICLINICI EYE HYPERCLINIC**

PROF. COORDONATOR: COSMINA IVAN

AUTORI:

**Cristina Man**

**Darius-Iulian Stan**

An academic: 2022-2023

Grupele: 30223, 30221

Contents

[INTRODUCERE 2](#_Toc125060208)

[MODELUL DE DATE 3](#_Toc125060209)

[Diagrama UML 3](#_Toc125060210)

[Soluții de transformare în relațional 3](#_Toc125060211)

[Proceduri și Triggere 5](#_Toc125060212)

[INTERFAȚĂ ȘI MANUAL DE UTILIZARE 11](#_Toc125060213)

[CONCLUZII ȘI EVENTUALE ÎNBUNĂTĂȚIRI 15](#_Toc125060214)

[BIBLIOGRAFIE ȘI RESURSE 15](#_Toc125060215)

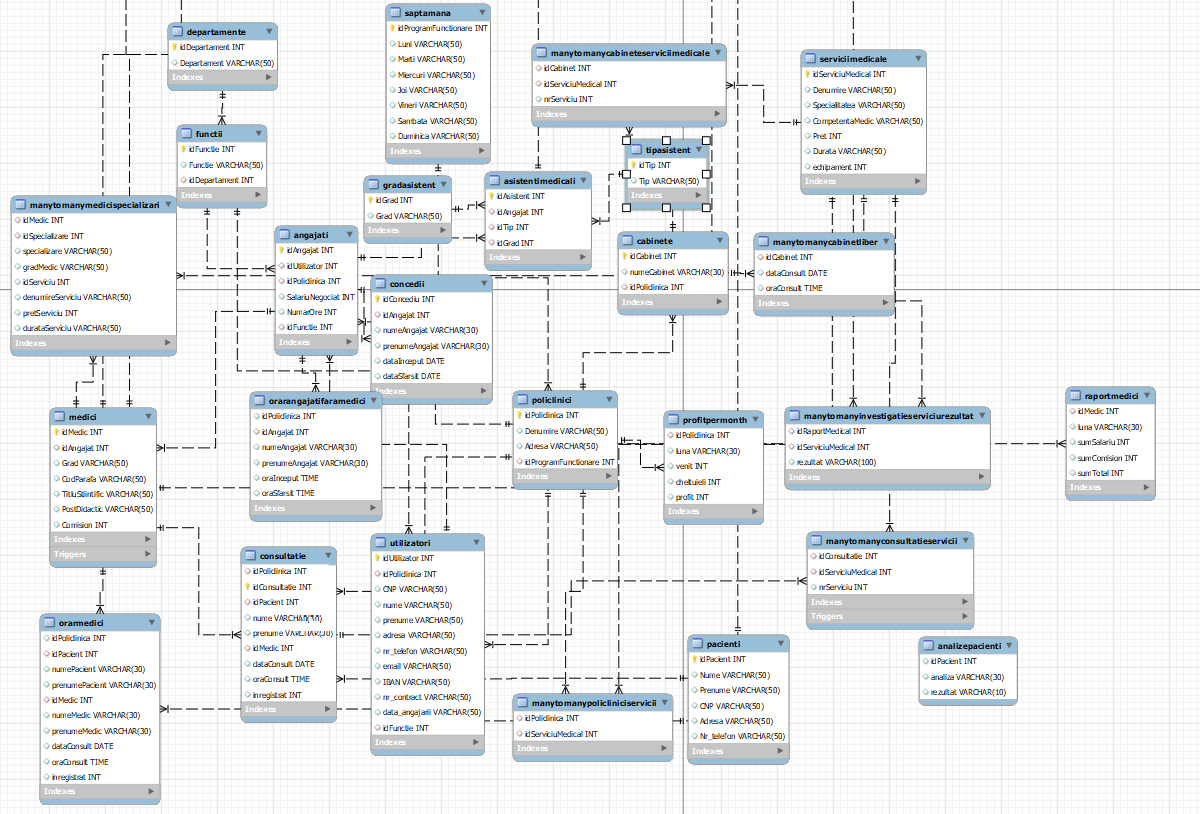
# INTRODUCERE

Acest proiect, “Gestionarea sistemului de policinici Eye Hyperclinicare“ are ca scop crearea unei aplicații menite management-ului unui lanț de policlinici dintr-un oraș, datorită căreia procesul de creare al progrmărilor să fie optimizat, atât pentru personalul implicat cât și pentru pacienți.

Aplicația oferă, printre altele, pentru fiecare institut în parte, posibilitatea de logare ca Super-Administrator, Administrator, Recepționer, Expert financiar-contabil, Inspector resurse umane, Asistent Medical și Medic, iar pentru fiecare dintre aceste opțiuni, există o gamă personalizată de acțiuni pe care utilizatorul le poate exercita, precum: vizualizare orar, aflare a profitului obținut de o policlinică anume sau de un doctor, etc.

# MODELUL DE DATE

## Diagrama UML



## Soluții de transformare în relațional

Pentru conectarea datelor, în cazul în care relația este one-to-many sau one-to-one, am ales să inserăm în cel de-al doilea tabel o cheie străină, provenită din primul tabel. Spre exemplu, la o consultație, avem nevoie de datele pacientului (pentru a putea pune un diagnostic ce ține cont de istoricul lui, dar și pentru a putea efecta plata srviciilor), dar și de datele medicului (pentru a ne asigura ca el este disponibil și avizat pentru efectuarea consultației). Astfel, în tabelul Consultație, regasim câmpurile IdMedic și IdPacient, făcând referire la tabelul Pacient și tabelul Medic. Am ales aceste câmpuri datorită faptului că ele se autoincrementează cu fiecare inserare de tuplă nouă, eliminând posibilitatea de a avea mai multe tuple cu aceleași valori pentru atributele respective.

Pe de altă parte, în cazul în care între două tabele există o relație Many-To-Many, am creat un tabel intermediar, în care să avem ca și atribute un câmp provenit din primul tabel și un altul din al doilea, astfel încât asocierea lor să fie corespunzătoare (și totodată să permită un număr nelimitat de asocieri). Spre exemplu, datorită faptului că în timpul unei consultații se pot efectua mai multe servicii medicale, și aceleași servicii medicale pot fi efectuate și în altă consultație, am creat tabelul ManytoManyConsultatieServicii, în care avem idCosultație, reprezentând cheia străină provenită din tabelul Consultații, și cheia idServiciuMedical, provenită din tabelul ServiciiMedicale. Ambele campuri, în tabelele mamă, sunt auto-incrementate, astfel că atunci când căutăm serviciile dintr-o consultație nu există ambiguitatea de a obține mai multe servicii decât cele necesare din cauza repetării unei valori IdConsultație pentru două consultații diferite (acest lucru ar fi fost posibil daca, de exemplu, am fi ales ca și cheie străină din Consultații campul dataConsult).

1. Descrierea bazei de date relaționale și tehnici de normalizare

Câteva dintre tabelele principale sunt: Utilizatori, Consultație, OrarMedici, Medici, Cabinete, Pacienti, ServiciiMediale, Departamente etc.

Tabelul Utilizatori reține date precum: idPoliclinică (instituția la care este repartizat), CNP, nume, prenume, adresa, număr de telefon, IBAN, data angajării, idFuncție. idFuncție este o cheie străină care trimite la tabelul Funcții, unde sunt enumerate tipurile de funcții pe care le poate avea utilizatorul (Admin, Expert financiar-contabil, angajat etc). IdPoliclinică trimite spre tabelul Policlinică, unde ca și date avem numele instituției și un id al acesteia, pentru o mai bună atomicizare a datelor referitoare la hiperclinică în sine.

În tabelul orarMedici, at fiind faptul că programull lor este unul flexibil, dorim să ținem evidența consultațiilor lor. Astfel, avem ca și câmpuri idMedic (o cheie strpină spre tabela medici), dataConsult, oraConsult, idPacient, dar și un câmp înregistrat, care servește la confirmarea că pacientu o să se prezinte la consultație.

În tabelul ProfitPerMonth se dorește reținerea dinamică a venitului (preturile consultațiilor efectuate), a cheltuielilor (salariile staff-ului și comisioanele medicilor), ca mai apoi să se poată calcula venitul unei policlinici, grupat fiind pe lunile anului.

În tabelul Saptamână dorim să oferim hiperclinicii posibilitatea de a avea mai multe programe de activitate. Spre exemplu, în timpul sărbătorilor de iarnă, policlinica ar fi închisă în anumite zile, iar în altele ar fi deschisă mai puține ore în comparație cu o perioadă oaarecare din an. Astfel, având un câmp pentru fiecare zi a săptămînii, se pot crea mai multe programe, fiecare reprezentând o tuplă în tabel. În plus, avem și un câmp IdProgram, pentru a ține evidența acestora.

Astfel, în tabelul Policlinici, avem ca și câmpuri: idPoliclinică (pentru a diferenția instituțiile între ele), Denumire, Adresa și idProgramFuncționare, care este o referință către tabelul anterior, câmp care poate fi modificat atunci când este nevoie.

Mai mult, avem și tabelul Cabinete, care are ca și câmpuri idCabinet, numeCabinet și IdPoliclinică (acest câmp a fost adăugat în ideea că, dat fiind faptul că sunt mai multe instituții în lanț, ar putea exissta doua cabinete cu același nume), pe baza căruia va avea loc repartizarea medicilor în cabinete.

Către tabelul Cabinet, există o cheie străină, idCabinet, în tabelul manytomanycabinetserviciimedicale, care reprezintă de fapt un tabel auxiliar, pentru soluționarea legaturii many to many între Cabinete și serviciiMedicale. Mai exact, un cabinet poate dispune de aparatură necesară unei anumite proceduri, astfel că dorim să ținem o evidență a posibilităților de efectuare a procedurilor pentru fiecare cabinet.

## Proceduri și Triggere

Pentru majoritatea tabelelor, am creat proceduri de tipul Insert, pentru a adăuga tuple noi. Spre exemplu, pentru tabelul medici, avem următoarea procedură:

DROP PROCEDURE IF EXISTS P\_Insert\_Medici;

DELIMITER //

CREATE procedure P\_Insert\_Medici (

idPoliclinica int,

NumarOre int,

idFunctie int,

SalariuNegociat int,

Grad varchar(50),

CodParafa varchar(50),

TitluStiintific varchar(50),

PostDidactic varchar(50),

Comision int

)

BEGIN

DECLARE idAngajatNOU int;

DECLARE idUtilizatorNOU int;

SELECT MAX(Utilizatori.idUtilizator) INTO idUtilizatorNOU FROM Utilizatori;

INSERT INTO Angajati(idUtilizator,

idPoliclinica,

NumarOre,

idFunctie,

SalariuNegociat

)

VALUES (idUtilizatorNOU,

idPoliclinica,

NumarOre,

idFunctie,

SalariuNegociat);

SELECT MAX(Angajati.idAngajat) INTO idAngajatNOU FROM Angajati;

INSERT INTO Medici ( idAngajat, Grad, CodParafa, TitluStiintific, PostDidactic, Comision )

VALUES ( idAngajatNOU, Grad, CodParafa, TitluStiintific, PostDidactic, Comision );

END // DELIMITER ;

Mai mult, pentru a actualiza tabelul profitpermonth, după fiecare inserare în Medic, dorim să modificăm cheltuielile policlinicii, motiv pentru care am create următorul trigger:

CREATE DEFINER=`root`@`localhost` TRIGGER `medici\_AFTER\_INSERT` AFTER INSERT ON `medici` FOR EACH ROW BEGIN

DECLARE salariu int;

SELECT Angajati.SalariuNegociat INTO salariu FROM Angajati

WHERE Angajati.idAngajat = NEW.idAngajat;

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '1', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '2', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '3', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '4', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '5', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '6', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '7', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '8', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '9', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '10', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '11', '0', '0', '0');

INSERT INTO RaportMedici (idMedic, luna, sumSalariu, sumComision, sumTotal)

VALUES(NEW.idMedic, '12', '0', '0', '0');

END

Mai mult, pentru actualizarea tabeluli când vine vorba de profitul generat de medici prin urma efectuării unei consultații, ținem cont că medicul respectiv ar putea avea un preț diferit pentru consultație, datorită gradului său, astfel, următorul trigger ia problema în considerare, însă ține cont și de comisionul medicului, modificând cheltuielile policlinicii:

CREATE DEFINER=`root`@`localhost` TRIGGER `manytomanyconsultatieservicii\_AFTER\_INSERT` AFTER INSERT ON `manytomanyconsultatieservicii` FOR EACH ROW BEGIN

DECLARE luna varchar(30);

DECLARE pret int;

DECLARE idPoliclinica int;

DECLARE dataConsult date;

DECLARE procentComision int;

DECLARE idMedic int;

DECLARE idSpecializare int;

DECLARE denumireSpecializare varchar(30);

SELECT ServiciiMedicale.Specialitatea INTO denumireSpecializare

FROM ServiciiMedicale WHERE ServiciiMedicale.idServiciuMedical = NEW.idServiciuMedical;

SELECT SpecializareMedici.idSpecializare INTO idSpecializare

FROM SpecializareMedici WHERE SpecializareMedici.Specializare = denumireSpecializare;

SELECT Consultatie.idMedic INTO idMedic

FROM Consultatie WHERE Consultatie.idConsultatie = NEW.idConsultatie;

SELECT Medici.Comision INTO procentComision

FROM Medici WHERE Medici.idMedic = idMedic;

SELECT Consultatie.idPoliclinica INTO idPoliclinica

FROM Consultatie WHERE NEW.idConsultatie = Consultatie.idConsultatie;

SELECT Consultatie.dataConsult INTO dataConsult

FROM Consultatie WHERE NEW.idConsultatie = Consultatie.idConsultatie;

SELECT MONTH(dataConsult) INTO luna;

SELECT ManyToManyMediciSpecializari.pretServiciu INTO pret FROM ManyToManyMediciSpecializari

WHERE ManyToManyMediciSpecializari.idServiciu = NEW.idServiciuMedical

AND ManyToManyMediciSpecializari.idMedic = idMedic;

UPDATE ProfitPerMonth

SET ProfitPerMonth.venit = ProfitPerMonth.venit + pret,

ProfitPerMonth.cheltuieli = ProfitPerMonth.cheltuieli + procentComision / 100 \* pret,

ProfitPerMonth.profit = ProfitPerMonth.venit - ProfitPerMonth.cheltuieli

WHERE ProfitPerMonth.idPoliclinica = idPoliclinica

AND ProfitPerMonth.luna = luna;

UPDATE RaportMedici

SET RaportMedici.sumSalariu = RaportMedici.sumSalariu + pret,

RaportMedici.sumComision = RaportMedici.sumComision + procentComision / 100 \* pret,

RaportMedici.sumTotal = RaportMedici.sumTotal + procentComision / 100 \* pret + pret

WHERE RaportMedici.idMedic = idMedic

AND RaportMedici.luna = luna;

UPDATE RaportSpecializari

SET RaportSpecializari.sumTotal = RaportSpecializari.sumTotal + pret - procentComision / 100 \* pret

WHERE RaportSpecializari.idSpecializare = idSpecializare

AND RaportSpecializari.luna = luna;

END

De altfel, avem o procedură de inserare și pentru tabela consultații, care primește toate datele necesare pentru completarea câmpurilor din tabel și, pe lângă asta, primește și maxim trei servicii medicale ce vor avea loc în aceasta.

DROP PROCEDURE IF EXISTS P\_Insert\_Consultatie;

DELIMITER //

CREATE procedure P\_Insert\_Consultatie (idPoliclinica int, nume varchar(30), prenume varchar(30), idMedic int, dataConsult date, oraConsult time,

serviciu\_1 varchar(30), serviciu\_2 varchar(30), serviciu\_3 varchar(30))

BEGIN

DECLARE numeMedic varchar(30);

DECLARE prenumeMedic varchar(30);

DECLARE idAngajat int;

DECLARE idUtilizator int;

DECLARE idServiciuMedical int;

DECLARE idConsultatie int;

DECLARE idPacient int;

SELECT Pacienti.idPacient INTO idPacient FROM Pacienti WHERE Pacienti.Nume = nume AND Pacienti.Prenume = prenume;

SELECT Medici.idAngajat INTO idAngajat FROM Medici WHERE idMedic = Medici.idMedic;

SELECT Angajati.idUtilizator INTO idUtilizator FROM Angajati WHERE Angajati.idAngajat = idAngajat;

SELECT Utilizatori.nume INTO numeMedic FROM Utilizatori WHERE idUtilizator = Utilizatori.idUtilizator;

SELECT Utilizatori.prenume INTO prenumeMedic FROM Utilizatori WHERE idUtilizator = Utilizatori.idUtilizator;

INSERT INTO Consultatie( idPoliclinica, idPacient, nume, prenume, idMedic, dataConsult, oraConsult )

VALUES ( idPoliclinica, idPacient, nume, prenume, idMedic, dataConsult, oraConsult);

INSERT INTO OrarMedici( idPoliclinica, idPacient, numePacient, prenumePacient, idMedic, numeMedic, prenumeMedic, dataConsult, oraConsult )

VALUES ( idPoliclinica, idPacient, nume, prenume, idMedic, numeMedic, prenumeMedic, dataConsult, oraConsult);

SELECT Consultatie.idConsultatie INTO idConsultatie FROM Consultatie WHERE Consultatie.idPacient = idPacient

AND Consultatie.idMedic = idMedic AND Consultatie.dataConsult = dataConsult

AND Consultatie.oraConsult = oraConsult;

IF (serviciu\_1 != "") THEN

SELECT ServiciiMedicale.idServiciuMedical INTO idServiciuMedical FROM ServiciiMedicale WHERE serviciu\_1 = ServiciiMedicale.Denumire;

INSERT INTO ManyToManyConsultatieServicii( idConsultatie, idServiciuMedical, nrServiciu )

VALUES ( idConsultatie, idServiciuMedical, '1' );

END IF;

IF (serviciu\_2 != "") THEN

SELECT ServiciiMedicale.idServiciuMedical INTO idServiciuMedical FROM ServiciiMedicale WHERE serviciu\_2 = ServiciiMedicale.Denumire;

INSERT INTO ManyToManyConsultatieServicii( idConsultatie, idServiciuMedical, nrServiciu )

VALUES ( idConsultatie, idServiciuMedical, '2' );

END IF;

IF (serviciu\_3 != "") THEN

SELECT ServiciiMedicale.idServiciuMedical INTO idServiciuMedical FROM ServiciiMedicale WHERE serviciu\_3 = ServiciiMedicale.Denumire;

INSERT INTO ManyToManyConsultatieServicii( idConsultatie, idServiciuMedical, nrServiciu )

VALUES ( idConsultatie, idServiciuMedical, '3' );

END IF;

END

// DELIMITER ;

# INTERFAȚĂ ȘI MANUAL DE UTILIZARE

Pentru crearea interfeței, am ales să folosim limbajul de programare C#, având ca template Windows Forms App (.NET FrameWork), iar ca IDEE Visual Studio. Clasele folosite sunt cele pentru crearea interfeței propriu-zise, deja create: Form, Button, TextBox, Label, ListBox, GreedView etc.

Utilizare: in primul tab al aplicației, utilizatorul trebuie să se conecteze.

Graphical user interface, application

Description automatically generated

Dacă utilizatorul este administrator, el va avea următoarele opțiuni:

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

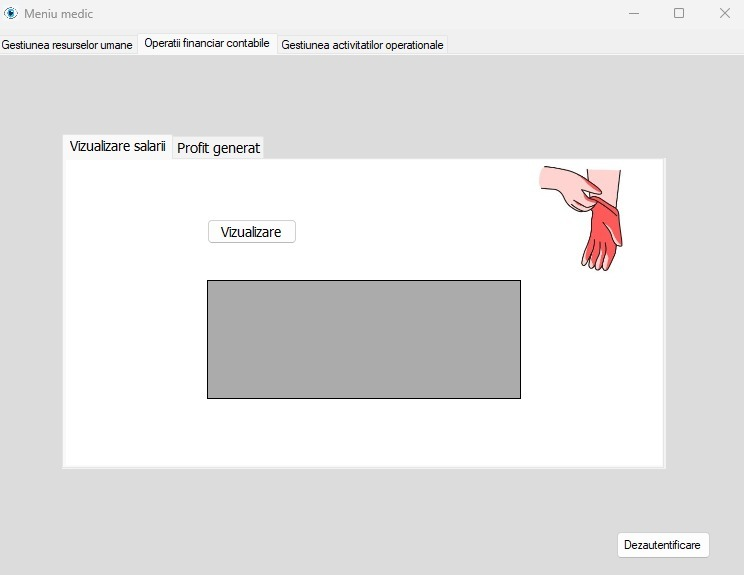
Adăugare utilizator: Administratorul are dreptul de a adăuga noi utilizatori, după completarea câmpurile și apăsarea butonului de Adăugare, în Sql se va apela procedura de P\_insert\_Utilizator (similar, pentru Stergere Utilizator și Modificare Utilizator).

Totodată, administratorul, ca orice alt utilizator, își poate vedea datele personale.

Dacă utilizatorul este Super-Administrator, el are aceleași opțiuni, cu excepția faptului că poate adăuga și administratori noi.

Dacă Utilizatorul este medic, el va avea, pe modulul Gestiunea Activităților operaționale posibilitatea de a își vedea propriu orar, perioada de concediu sau satele sale personale. În modului Operații financiar contabile, poate vizualiza atât venitul său pe fiecare lun, cât și profitul pe care l-a adus policlinicii în urma efectuării consultațiilor. Cât despre modului Gestinea activităților operaționale, el are dreptul de a vizualiza ordinea consultațiilor din ziua curentă și istoricul pacienților programați.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Un asistent, ca orice angajat, are modulul Gestiunea resurselor umani și Operații financiar contabile identice cu cele prezentate mai sus(ale medicului).

Graphical user interface, application

Description automatically generated

Însă, în ultimul modul, el are dreptul de a completa rezultatele analizelor pentru un anumit pacient.

Când se loghează un expert financiar-contabil, în modului specific, el are posibilitatea de a verifica, profitul total al lanțului de clinici, veniturile generate de medicii unei anumite specializări sau clinici și de a-și vizualiza propriul salariu.

Table

Description automatically generated

Un inspector pe resurse umane, în modului specific, are dreptul de a căuta orice angajat după nume, prenume și funcție și de a vizualiza toate informațiile despre orarul său.

Table

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Totodată, un inspector poate adăuga servicii medicale speciale în funcție de medic.

Un recepționer poate crea o programare noua doar pentru o dată calendaristică ulterioară zilei curente, poate confirma sosirea pacienților pentru consult și totodată, poate genera un bon fiscal după terminarea consultației.

Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

# CONCLUZII ȘI EVENTUALE ÎNBUNĂTĂȚIRI

In privința dezvoltărilor ulterioare, se dorește optimizarea algoritmului de repartizare a medicilor in cadrul cabinetelor policlinicii prin folosirea programării dinamice. De asemenea, pentru a ușura munca recepționerilor si a asigura o desfășurare a activităților mai eficienta din punct de vedere a timpului, poate fi implementata o funcționalitate de check-in online, prin intermediul căreia pacienții sosiți la policlinica pentru consult se pot înregistra singuri.

Așadar, folosind Sql și C# ca limbaje pentru implementarea aplicație, lanțul de policlinici poate beneficia de optimizarea proceselor interne din cadrul instituțiilor, ușurând activitatea fiecărui angajat.

# BIBLIOGRAFIE ȘI RESURSE

<https://www.w3schools.com/MySQL/default.asp>

<https://www.canva.com/>

<https://www.mysql.com/>