Proiect IBD

“ LANȚ DE POLICLINICI”

Stăvar Sebastian
Trif Gabriela Ioana
30221

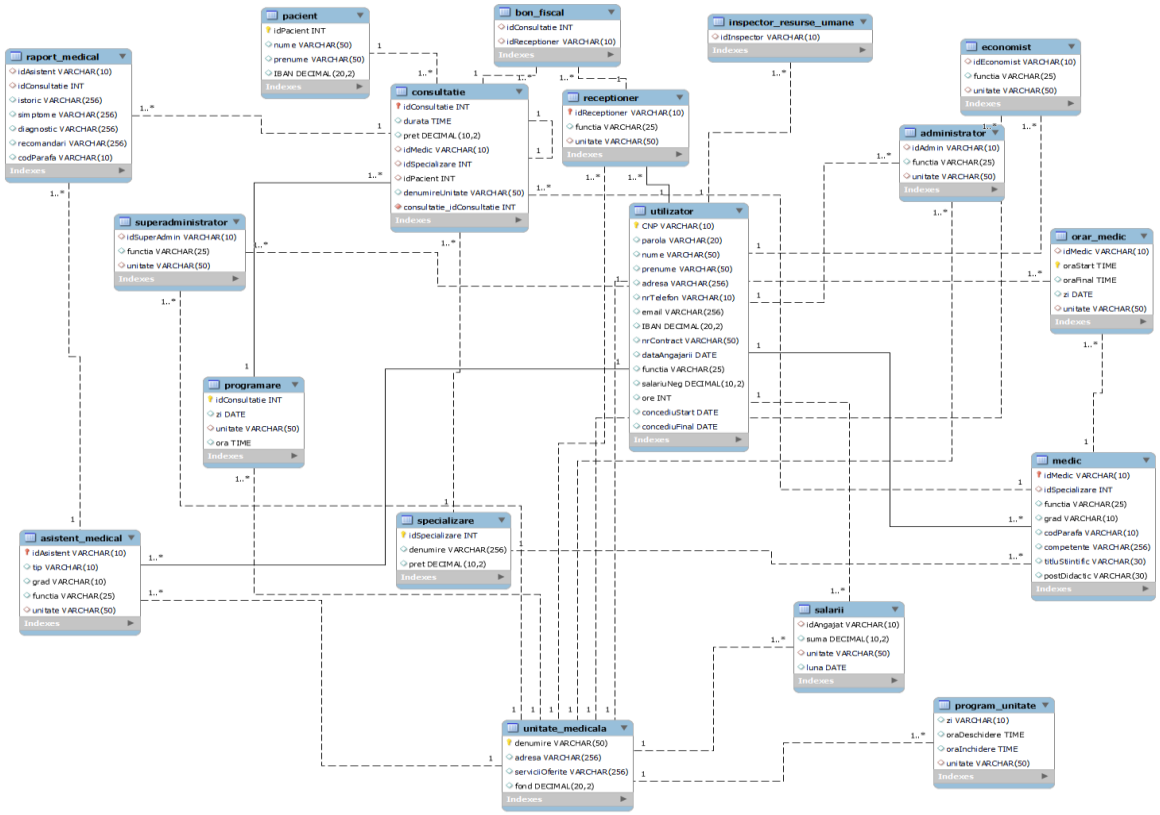
Universitatea Tehnică Cluj-Napoca
Facultatea de Automatică și Calculatoare
2022/2023

Cuprins

Capitolul 1.Tema proiectului

Proiectul constă în realizarea unui sistem de gestiune a unui lanț de policlinici cu ajutorul unei baze de date care conține elementele necesare: utilizatori, unități , pacienți, administratori.

Capitolul 2.Detalii de proiectare conceptuală a bazei de date



Capitolul 3.Soluția de transformare în relațional

3.1 Many to many

Tabela **consultație** are rolul de a elimina relația many-to-many dintre tabelele medic și pacient, având ca și cheie străină idPacient și idMedic, restul câmpurilor din tabelă rețin detalii despre consultație.

3.2 Alegerea cheii primare

Fiecare din tabele conține o cheie primară de forma id[*nume_tabelă*] , cu excepția tabelului **utilizator** care are ca și cheie primară atributul CNP.

Capitolul 4.Descrierea bazei de date relaționale

În acest capitol se vor descrie tabelele împreună cu atributele lor:

4.1 utilizator

Tabela utilizator reține date despre toți angajații lanțului de policlinici.

4.1.1. CNP

Atributul CNP reprezintă codul numeric personal al utilizatorului.

4.1.2. parola

Atributul parola reprezintă parola aleasă cu utilizator pentru accesul în sistem.

4.1.3. nume

Atributul nume reține numele utilizatorului.

4.1.3. prenume

Atributul prenume reține prenumele utilizatorului.

4.1.4. adresa

Acest atribut reține adresa unui utilizator.

4.1.5. nrTelefon

Atributul acesta reține numărul de telefon al utilizatorului.

4.1.5. email

Atributul email memorează adresa de email a utilizatorului.

4.1.6. IBAN

Acest atribut reține numărul de cont IBAN al utilizatorului, în acest cont se va afla valoarea salariului.

4.1.7. nrContract

Acest atribut reține numărul contractului de angajare.

4.1.8. dataAngajarii

Prin acest atribut se va reține data angajării utilizatorului.

4.1.9. functia

Atributul funția indică funcția pe care o ocupă utilizatorul în lanțul de policlinici, aceasta poate fi: medic, asistent medical, inspector resurse umane, superadministrator, administrator, economist sau recepționer.

4.1.10. salariuNeg

Atributul salariuNeg reprezintă salariul negociat de angajat pentru fiecare oră de lucru.

4.1.11. ore

Acest atribut reprezintă numărul de ore de muncă pe care angajatul trebuie să le execute pe parcursul unei luni.

4.1.12. concediuStart

Atributul concediuStart reprezintă data primei zile a concediului anual .

4.1.13. concediuFinal

Acest atribut reprezintă data ultimei zile din concediul anual.

4.2. medic

În acest tabel se reține datele medicilor angajați în lanțul de policlinici.

4.2.1. idMedic

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.2.2. idSpecializare

Atributul idSpecializare face legătura dintre acest tabel și tabelul specializare.

4.2.3. functia

Acest atribut este echivalent cu atributul cu acelasi nume din tabelul utilizator.

4.2.4. grad

Acest atribut reține gradul medicului, acesta poate fi: specialist sau primar.

4.2.5. codParafa

Atributul codParafa reține codul de parafă al medicului.

4.2.6. competente

Acest atribut descrie competețele medicului.

4.2.7. titluStiintific

Acest atribut poate avea următoarele valori: doctorand, doctor în științe medicale sau NULL.

4.2.8. postDidactic

Acest atribut reține postul didactic ocupat de medic, acesta poate fi: preparator, asistent, șef de lucrări, conferențiar sau profesor. Nu este obligatoriu ca un medic să dețină un post didactiv, deci acest atribut poate avea chiar și valoarea NULL .

4.3. asistent_medical

În această tabelă se găsesc toți utilizatori care au funcția de asistent medical.

4.3.1. idAsistent

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.3.2. tip

Acest atribut reține tipul asistentului medical , care poate fi: generalist, laborator sau radiologie.

4.3.3. grad

Atributul grad specific asistentului medical poate fi secundar sau principal.

4.3.4. functia

Acest atribut este echivalent cu atributul cu acelasi nume din tabelul utilizator

4.3.5. unitate

Atributul unitate este utilizat pentru a reține numele unității la care este angajat asistentul medical.

4.4. recepționer

4.4.1.idRecepționer

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.4.2. funcția

Acest atribut are valoarea receptioner pentru fiecare tuplă din tabel.

4.4.3. unitate

Atributul unitate este utilizat pentru a reține numele unității la care este angajat asistentul medical.

4.5. economist

Acest tabel reține utilizatorii responsabili de modulul pentru gestiunea operațiilor financiar-contabile.

4.5.1. idEconomist

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.5.2. functia

Acest atribut are aceeași valoare pentru fiecare tuplă din tabel.

4.5.3. unitate

Atributul unitate este utilizat pentru a reține numele unității la care este angajat asistentul medical.

4.6. administrator

Tabelul administrator este utilizat pentru a reține datele utilizatorilor de tip administrator care pot adăuga, șterge sau actualiza informații legate de utilizatori.

4.6.1. idAdmin

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.6.2. functia

Acest atribut are aceeași valoare pentru fiecare tuplă din tabel.

4.6.3. unitate

Atributul unitate este utilizat pentru a reține numele unității la care este angajat asistentul medical.

4.7. superadministrator

Acest tabel reține informații despre utilizatorul de tip superadministrator, acest utilizator poate opera inclusiv asupra utilizatorilor de tip administrator.

4.7.1.idSuperAdmin

Acest atribut este cheia primară a tabelului , este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.7.2. functia

Acest atribut are aceeași valoare pentru fiecare tuplă din tabel.

4.7.3. unitate

Atributul unitate este utilizat pentru a reține numele unității la care este angajat asistentul medical.

4.8. inspector_resurse_umane

Acest tabel reține id-urile angajaților care gestionează resursele umane.

4.8.1. idInspector

Acest atribut este cheia primară a tabelului. Câmpul idInspector este echivalent cu atributul CNP din tabelul utilizator, care este cheie străină în acest tabel.

4.9. orar_medic

Fiecare tuplă din acest tabel cuprinde programul unui medic pentru o zi anume.

4.9.1. idMedic

Atributul idMedic referentiaza id-ul unui utilizator cu functia de medic.

4.9.2. oraStart

Atributul oraStart reprezintă ora de începere a programului pentru ziua din atributul zi.

4.9.3. oraFinal

Atributul oraFinal reprezintă ora de sfârșit a programului.

4.9.4. zi

Atributul zi este de tipul DATE si reține ziua pentru care este specificat programul în aceeași tuplă.

4.9.5.unitate

Atributul unitate reține denumirea unității în care își va desfășura medicul programul în ziua precizată în câmpul zi.

4.10. unitate_medicala

Această tabele reține informații despre unitățile medicale aflate în lanțul de policlinici.

4.10.1. denumire

Atributul denumire este cheia primară a tabelului unitat_medicală și reține denumirea unității medicale.

4.10.2. adresa

Acest câmp reține adresa unității medicale.

4.10.3. serviciiOferite

Acest câmp memorează specializările disponibile în unitatea medicală.

4.10.4. fond

Acest câmp reprezintă fondul unității medicale, se actualizează în funcție de plata salariilor sau achitarea consultațiilor.

4.11. program_unitate

Acest tabel reține programul unităților medicale, pentru fiecare zi a săptămânii în parte.

4.11.1. zi

Câmpul zi este de tipul VARCHAR(10), poate conține una din următoarele valori: luni, marți, miercuri, joi, vineri.

4.11.2. oraDeschidere

Acest câmp reține ora de deschidere a unității medicale.

4.11.3. oraInchidere

Acest câmp reține ora de închidere a unității medicale.

4.11.4.unitate

Câmpul unitate reține denumirea unității medicale pentru care este specificat programul în tupla respectivă.

4.12. specializare

Acest tabel este utilizat pentru a reține serviciile medicale disponibile.

4.12.1. idSpecializare

Câmpul idSpecializare este folosit pentru a avea un număr de ordine al specializărilor.

4.12.2. denumire

În atributul denumire se găsește numele specializării.

4.12.3. pret

Atributul pret , de tipul flotant , reprezintă valoarea prețului unei consultații pentru specializarea respectivă.

4.13. pacient

Prin intermediul acestui tabel se vor reține toți pacienți care au avut parte de una sau mai multe consultații la orice unitate medicală.

4.13.1. idPacient

Câmpul idPacient reține numărul de ordine al pacientului, reprezintă cheia primară a tabelului.

4.13.2. nume
Acest câmp reține numele pacientului.

4.13.3. prenume
Acest câmp reține prenumele pacientului.

4.13.4 IBAN
Prin acest câmp se va reține soldul curent al pacientului, din care se va efectua plata unei consultații.

4.14. programare
In acest tabel se va reține orice programare efectuată de un recepționer.

4.14.1. idConsultatie
Câmpul idConsultație este cheia primară a tabelului, duce la tupla cu valoarea idConsultatie din tabelul consultatie.

4.14.2. zi
Acest atribut reprezintă data în care se va efectua programarea.

4.14.3. unitate
Acest atribut reține numele unității la care va trebui să se prezinte pacientul pentru consultație.

4.14.4. ora
Atributul ora reprezintă ora programării.

4.15. consultatie
In acest tabel se vor face înregistrările pacienților în momentul în care se prezintă la unitatea medicală pentru efectuarea consultației.

4.15.1. idConsultatie
Acest câmp este corespondentul câmpului idProgramare din tabelul programare.

4.15.2 durata
Atributul durata caracterizează durata unei consultații.

4.15.3. pret
Acest câmp reprezintă costul unei consultații pe o anumită specializare și este preluat din tabelul specializare.

4.15.4. idMedic
Acest atribut cuprinde id-ul medicului care va efectua consultația.

4.15.5. idSpecializare
Atributul idSpecializare reprezintă numărul de ordine al specializării consultației.

4.15.6. idPacient
Acest atribut reține id-ul pacientului înregistrat în tabelul pacient.

4.15.7. denumireUnitate
Acest atribut reține numele unității în care se va desfășura consultația.

4.16. salarii

Tabela salarii cuprinde salariile pentru fiecare angajat.

4.16.1. idAngajat

Acest câmp reprezintă id-ul angajatului căruia îi corespunde salariul din câmpul suma.

4.16.2. suma

Câmpul suma de tip flotant reprezintă valoarea salariului.

4.16.3. unitate

Acest câmp reprezintă unitatea de la care angajatul îți primește salariul.

4.16.4. luna

Acest câmp reține luna în care s-a efectuat plata salariului.

4.17. bon_fiscal

Tabelul bon_fiscal va reține orice bon fiscal eliberat după efectuarea plății unei consultații.

4.17.1. idConsultație

Acest câmp reprezintă id-ul consultației pentru care s-a eliberat bonul fiscal.

4.17.2. idReceptioner

Câmpul idRecepționer reține id-ul recepționerului care va încasa plata consultației și va elibera bonul fiscal.

4.18. raport_medical

Acest tabel reține informațiile prezente în raportul medical întocmit pe baza consultației.

4.18.1. idConsultatie

Acest câmp reține numărul de ordine al consultației pentru care s-a întocmit raportul.

4.18.2. idAsistent

Acest câmp reține id-ul asistentului prezent la consultație, poate avea valoarea NULL în cazul în care consultația este efectuată fără prezența unui asistent medical.

4.18.3. istoric

Acest câmp descrie istoricul medical al pacientului.

4.18.4. simptome

Câmpul simptome caracterizează simptomele pacientului.

4.18.5. diagnostic

Acest câmp descrie diagnostic acordat de medic în urma consultației.

4.18.6. recomandari

În acest câmp se vor găsi recomandările pentru pacient.

4.18.7. codParafa

Acest câmp poate fi completat doar de către medic, după ce este completat, raportul medical nu mai poate fi modificat

Capitolul 5. Nivelul de normalizare

Baza de date a aplicației de gestiune a unui lanț de policlini se află în forma normală Boyce-Codd.

Scopul BNCF este de spori integritatea datelor prin organizarea coloanelor și tabelelor dintr-o bază de date relațională pentru a realiza normalizarea bazei de date.

Nivelul de normalizare Boyce-Codd a fost creat ca o extindere la a treia formă normală, în 1974.

Baza de date policlinică îndeplinește următoarele afirmații:

- Orice atribut este atomic.
- Orice atribut care nu face parte din cheia primară este total dependent functional de cheia primară a relației.
- Orice atribut reprezintă o singură informație.
- Tabelele au chei primare, reprezentate de id-ul unic al tabelelor.
- Atributul cheii primare este suficient pentru a determina celelalte atribute ale tabelelor.
- Nu există un al atribut, excepând cheia primară care să poate determina în mod unic o tuplă.

Capitolul 6. Interogări

În acest capitol se se vor prezenta mai multe interogări care oferă continutul bazei de date:

- Afișarea unităților medicale din Cluj:

```
SELECT unitate_medicala.denumire, unitate_medicala.adresa
FROM unitate_medicala
WHERE unitate_medicala.adresa LIKE '%Cluj%';
```

- Afișarea angajaților/utilizatorilor:

```
SELECT utilizator.nume, utilizator.prenume, utilizator.functia
FROM utilizator
ORDER BY utilizator.functia ASC;
```

- Afișarea tuturor medicilor care pot efectua ecografii:

```
SELECT utilizator.nume, utilizator.prenume, specializare.denumire
FROM utilizator, medic, specializare
WHERE utilizator.CNP=medic.idMedic AND
medic.idSpecializare=specializare.idSpecializare AND
specializare.denumire="ecografie";
```

- Afișarea programărilor din ziua curentă:

```
SELECT *
FROM  programare
WHERE programare.zi=curdate();
```

- Afișarea specializărilor cu prețul mai mic decât 500 de lei:

```
SELECT specializare.denumire, specializare.pret
FROM specializare
WHERE specializare.pret<500
ORDER BY specializare.pret ASC, specializare.denumire ASC;
```

- Afișarea recepționerilor și unitatea medicală la care lucrează:

```
SELECT utilizator.nume, utilizator.prenume, receptioner.unitate
FROM utilizator, receptioner
WHERE utilizator.CNP=receptioner.idReceptioner
ORDER BY receptioner.unitate ASC, utilizator.nume ASC, utilizator.prenume ASC;
```

- Afișarea programului unei unități medicale:

```
SELECT * FROM program_unitate
WHERE program_unitate.unitate="Regina Maria";
```

Capitolul 7. Obiecte de tip vedere

Pentru utilizatori de tip medic

```
CREATE VIEW medici AS
  SELECT
    utilizator.nume,
    utilizator.prenume,
    medic.idSpecializare,
    medic.grad,
    medic.competente,
    medic.postDidactic,
    medic.titluStiintific,
    utilizator.salariuNeg,
    utilizator.ore
  FROM
    utilizator, medic, specializare
  WHERE
    utilizator.CNP = medic.idMedic
      AND medic.idSpecializare = specializare.idSpecializare
  ORDER BY nume ASC;
```

Pentru utilizatori de tip asistent

```
CREATE VIEW asistenti_medicali AS
  SELECT
    utilizator.nume,
    utilizator.prenume,
    asistent_medical.tip,
    asistent_medical.grad,
    asistent_medical.unitate
  FROM
```

```
    utilizator, asistent_medical, unitate_medicala
        WHERE
    asistent_medical.idAsistent = utilizator.CNP
        AND asistent_medical.unitate = unitate_medicala.denumire
ORDER BY nume ASC;
```

Pentru pacienți unităților medicale din lanțul de policlini

```
CREATE VIEW pacienti AS
    SELECT
        pacient.idPacient,
        pacient.nume,
        pacient.prenume,
        consultatie.idConsultatie,
        consultatie.pret
    FROM
        pacient, consultatie
            WHERE
        consultatie.idPacient = pacient.idPacient
    ORDER BY nume ASC;
```

Pentru istoricul unui pacient

```
CREATE VIEW istoric AS
    SELECT DISTINCT
        pacient.idPacient,
        pacient.nume,
        pacient.prenume,
        consultatie.idConsultatie,
        consultatie.pret,
        consultatie.idMedic
    FROM
        pacient, consultatie, utilizator
    WHERE
        consultatie.idPacient = pacient.idPacient
            AND consultatie.idMedic = utilizator.CNP
    ORDER BY CNP ASC;
```

Pentru vizualizarea orarului unei unități

```
CREATE VIEW orar AS
    SELECT
        program_unitate.zi,
        program_unitate.oraDeschidere,
        program_unitate.oraInchidere,
        unitate_medicala.denumire
    FROM
        program_unitate, unitate_medicala
    WHERE
        unitate_medicala.denumire = program_unitate.unitate
    ORDER BY denumire ASC;
```

Pentru vizualizarea utilizatorilor de tip economist:

```
CREATE VIEW economisti AS
  SELECT
    economist.idEconomist,
    economist.unitate,
    utilizator.nume,
    utilizator.prenume
  FROM
    economist, utilizator
  WHERE
    economist.idEconomist = utilizator.CNP
  ORDER BY unitate ASC;
```

Pentru specializările disponibile

```
CREATE VIEW specializari AS
  SELECT
    specializare.idSpecializare,
    specializare.denumire,
    specializare.pret
  FROM
    specializare
  ORDER BY idSpecializare ASC , pret DESC;
```

Pentru utilizatorii de tip recepționer

```
CREATE VIEW receptioneri AS
  SELECT
    receptioner.idReceptioner,
    receptioner.unitate,
    utilizator.nume,
    utilizator.prenume
  FROM
    receptioner, utilizator
  WHERE
    receptioner.idReceptioner = utilizator.CNP
  ORDER BY unitate ASC;
```

Pentru vizualizarea programărilor

```
CREATE VIEW programari AS
  SELECT
    programare.idConsultatie,
    programare.ora,
    programare.unitate
  FROM
    programare, orar_medic, medic
  WHERE
    medic.idMedic = orar_medic.idMedic
  ORDER BY ora ASC;
```

Pentru vizualizarea consultațiilor

```
CREATE VIEW consultatii AS
  SELECT
```

```
    consultatie.idConsultatie,
    consultatie.idPacient,
    pacient.nume,
    pacient.prenume,
    consultatie.idMedic,
    consultatie.durata,
    consultatie.pret
  FROM
    consultatie,
    utilizator, pacient
  WHERE
    consultatie.idMedic = utilizator.CNP
      AND consultatie.idPacient = pacient.idPacient
  ORDER BY idConsultatie ASC;
```

Pentru vizualizarea rapoartelor

```
CREATE VIEW rapoarte_medicale AS
  SELECT DISTINCT
    raport_medical.idRaport,
    raport_medical.idConsultatie,
    consultatie.idPacient,
    pacient.nume,
    pacient.prenume,
    utilizator.nume,
    utilizator.prenume,
    consultatie.idMedic,
    raport_medical.idAsistent,
    raport_medical.istoric,
    raport_medical.simptome,
    raport_medical.diagnostic,
    raport_medical.recomandari
  FROM
    raport_medical, asistent_medical, consultatie, pacient, medic, utilizator
  WHERE
    asistent_medical.idAsistent = raport_medical.idAsistent
      AND consultatie.idPacient = pacient.idPacient
      AND raport_medical.idConsultatie = consultatie.idConsultatie
  ORDER BY idRaport;
```

Pentru utilizatorii de tip inspector resurse umane

```
CREATE VIEW inspectori_resurse_umane AS
  SELECT
    inspector_resurse_umane.idInspector,
    utilizator.nume,
    utilizator.prenume
  FROM
    inspector_resurse_umane, utilizator
  WHERE
    inspector_resurse_umane.idInspector = utilizator.CNP
  ORDER BY nume;
```

Capitolul 8. Proceduri stocate

8.1.Procedura de adăugare a unui utilizator

DELIMITER //

CREATE PROCEDURE adaugareUtilizator(in CNP VARCHAR(10), in parola VARCHAR(20), in nume VARCHAR(50), in prenume VARCHAR(50), in adresa VARCHAR(256), in nrTelefon VARCHAR(10), in email VARCHAR(256), in IBAN DECIMAL(20,2), in functia VARCHAR(25))

BEGIN

INSERT INTO utilizator (CNP, parola, nume, prenume, adresa, nrTelefon, email, IBAN, functia)

VALUES (CNP, parola, nume, prenume, adresa , nrTelefon, email, IBAN, functia);

END //

DELIMITER ;

8.2.Procedura de adăugare a unui medic

DELIMITER //

CREATE PROCEDURE adaugareMedic(in idMedic VARCHAR(10), in idSpecizalizare VARCHAR(10), in functia VARCHAR(25), in grad VARCHAR(10), in codParafa VARCHAR(10), in competente VARCHAR(256), in titluStiintific VARCHAR(30), in postDidactic VARCHAR(30))

BEGIN

INSERT INTO medic (idMedic, idSpecializare, functia, grad, codParafa, competente, titluStiintific, postDidactic)

VALUES (idMedic, idSpecializare, functia, grad, codParafa, competente, titluStiintific, postDidactic);

END //

DELIMITER ;

8.3.Procedura de adaugare a unui asistent

DELIMITER //

CREATE PROCEDURE adaugareAsistentMedical(in idAsistent VARCHAR(10), in tip VARCHAR(10), in grad VARCHAR(10), in functia VARCHAR(25), in unitate VARCHAR(50))

BEGIN

INSERT INTO asistent_medical(idAsistent, tip, grad, functia, unitate)

VALUES (idAsistent, tip, grad, functia, unitate);

END //

DELIMITER ;

8.4.Procedura de adaugare a unui receptioner

DELIMITER //

CREATE PROCEDURE adaugareReceptioner(in idReceptioner VARCHAR(10), in functia VARCHAR(25), in unitate VARCHAR(50))

BEGIN

INSERT INTO receptioner(idReceptioner, functia, unitate)

VALUES (idReceptioner, functia, unitate);

END //

DELIMITER ;

8.5.Procedura de adaugare a unui economist

```
DELIMITER //
CREATE PROCEDURE adaugareEconomist(in idEconomist VARCHAR(10), in functia VARCHAR(50), in unitate VARCHAR(50))
BEGIN
    INSERT INTO economist (idEconomist, functia, unitate)
     VALUES (idEconomist, functia, unitate);
END //
DELIMITER ;
```

8.6. Procedura de adăugare a unui inspector de resurse umane

```
DELIMITER //
CREATE PROCEDURE adaugareInspector(in idInspector VARCHAR(10))
BEGIN
    INSERT INTO inspector_resurse_umane (idInspector)
      VALUES (idInspector);
END //
DELIMITER ;
```

8.7.Procedura de adaugare a unui admin

```
DELIMITER //
CREATE PROCEDURE adaugareAdmin(in idAdmin VARCHAR(10), in functia VARCHAR(25), in unitate VARCHAR(50))
BEGIN
    INSERT INTO administrator (idAdmin, functia, unitate)
    VALUES (idAdmin, functia, unitate);
END //
DELIMITER ;
```

8.8.Procedura de adaugare a unui pacient

```
DELIMITER //
CREATE PROCEDURE adaugarePacient(  in nume VARCHAR(50), in prenume VARCHAR(50), in IBAN DECIMAL(20,3))
BEGIN
    INSERT INTO pacient (nume, prenume, IBAN)
   VALUES (nume, prenume, IBAN);
END //
DELIMITER ;
```

8.9. Procedura de adaugare unitate medicală

```
DELIMITER //
CREATE PROCEDURE adaugareUnitate(in denumire VARCHAR(50), in adresa VARCHAR(256), in serviciiOferite VARCHAR(256))
BEGIN
    INSERT INTO unitate_medicala(denumire, adresa , serviciiOferite)
  VALUES (denumire, adresa , serviciiOferite);
END //
DELIMITER ;
```

8.10.Procedura de adaugare a unei programari

```
DELIMITER //
CREATE PROCEDURE adaugareProgramare( in id INT ,in nume VARCHAR(50), in ora TIME)
BEGIN
   INSERT INTO programare (idConsultatie, Unitate, ora)
   VALUES (id,nume, ora);
 END //
DELIMITER ;
```

8.11. Procedura de adaugare a unei consultatii

```
DELIMITER //
CREATE PROCEDURE adaugareConsultatie( in ora TIME, in idM VARCHAR(10), in idS VARCHAR(10), in idP INT)
BEGIN
   INSERT INTO consultatie( durata, idMedic, idSpecializare, idPacient)
   VALUES ( ora ,idM, idS, idP);
 END //
DELIMITER ;
```

8.12.Procedura de adaugare a unui raport medical

```
DELIMITER //
CREATE PROCEDURE adaugareConsultatie( in ora TIME, in idM VARCHAR(10), in idS VARCHAR(10), in idP INT)
BEGIN
   INSERT INTO consultatie( durata, idMedic, idSpecializare, idPacient)
  VALUES ( ora ,idM, idS, idP);
 END //
DELIMITER ;
```

8.13.Procedura de adaugare bon fiscal

```
DELIMITER //
CREATE PROCEDURE emitereBonFiscal(in id INT , in idC VARCHAR(10))
BEGIN
   INSERT INTO bon_fiscal(idConsultatie, idReceptioner)
   VALUES (id, idC);
 END //
DELIMITER ;
```

8.14. Procedura de adaugare program unitate

```
DELIMITER //
CREATE PROCEDURE adaugareProgramUnitate(in unitate VARCHAR(50), in   zi VARCHAR(10), in oraDeschidere TIME,in oraInchidere TIME)
BEGIN
        DELETE FROM program_unitate WHERE program_unitate.unitate=unitate AND program_unitate.zi=zi;
   INSERT INTO program_unitate (unitate,zi ,oraDeschidere, oraInchidere)
   VALUES (unitate,zi ,oraDeschidere, oraInchidere);
END //
DELIMITER ;
```

8.14.Procedura de stergere a unui angajat

```
DELIMITER //
CREATE PROCEDURE stergereAngajat(in CNP VARCHAR(10) , in nume VARCHAR (50), in prenume VARCHAR(50))
BEGIN
        DELETE FROM orar_medic WHERE CNP=orar_medic.idMedic;
        DELETE FROM medic WHERE CNP=medic.idMedic;
        DELETE FROM asistent_medical WHERE asistent_medical.idAsistent=CNP;
        DELETE FROM receptioner WHERE receptioner.idReceptioner=CNP;
        DELETE FROM economist WHERE economist.idEconomist=CNP;
       DELETE FROM inspector_resurse_umane
       WHERE inspector_resurse_umane.idInspector=CNP;
       DELETE FROM administrator WHERE administrator.idAdmin=CNP;
       DELETE FROM utilizator WHERE CNP = utilizator.CNP;
  END //
DELIMITER ;
```

8.15. Procedura de stergere unitate medicala

```
DELIMITER //
CREATE PROCEDURE stergereUnitate(in denumire VARCHAR(50))
BEGIN
        DELETE FROM unitate_medicala WHERE unitate_medicala.denumire=denumire;
END //
DELIMITER ;
```

8.16. Procedura de stergere a unei specializari

```
DELIMITER //
CREATE PROCEDURE stergereServiciuMedical( in idSpecializare INT, in denumire VARCHAR(256))
BEGIN
        DELETE FROM specializare WHERE specializare.denumire=denumire AND specializare.idSpecializare=idSpecializare;
END //
DELIMITER ;
```

8.17.Procedura de actualizare salariu

```
DELIMITER //
CREATE PROCEDURE actualizareSalariu(IN CNP VARCHAR(10), IN x DECIMAL(10,2))
BEGIN
  UPDATE utilizator SET salariuNeg = x WHERE utilizator.CNP=CNP ;
END //
DELIMITER ;
```

8.18.Procedura de actualizare a numarului de ore a unui angajat

```
DELIMITER //
CREATE PROCEDURE actualizareNrOre(IN CNP VARCHAR(10), IN x INT)
BEGIN
  UPDATE utilizator SET ore = x WHERE utilizator.CNP=CNP ;
```

```
END //
DELIMITER ;
```

8.19.Procedura de parafare raport

```
DELIMITER //
CREATE PROCEDURE parafareRaport(IN idRaport VARCHAR(10),IN x VARCHAR(10))
BEGIN
   UPDATE raport_medical SET codParafa = x WHERE raport_medical.idRaport=idRaport
AND codParafa = NULL;
END //
DELIMITER ;
```

8.20.Procedura de adaugare parola

```
DELIMITER //
CREATE PROCEDURE creeareParola(in CNP VARCHAR(10), in x VARCHAR(20))
BEGIN
        UPDATE utilizator SET  parola = x WHERE utilizator.CNP = CNP;
END //
DELIMITER ;
```

8.21.Procedura de determinare a pretului unei consultatii

```
DELIMITER //
CREATE PROCEDURE determinarePret(IN idC INT , IN idS VARCHAR(10))
BEGIN
        UPDATE consultatie SET consultatie.pret=(SELECT specializare.pret FROM
specializare WHERE specializare.idSpecializare=idS) WHERE idC=consultatie.idConsultatie;
END//
DELIMITER;
```

8.22.Procedura de determinare a functiei unui utilizator

```
DELIMITER //
CREATE PROCEDURE gasireFunctieUtilizator(IN CNP VARCHAR(10))
BEGIN
        SELECT functia FROM utilizator WHERE utilizator.CNP = CNP;
END //
DELIMITER ;
```

8.23.Procedura de login

```
DELIMITER //
CREATE PROCEDURE login(IN CNP VARCHAR(10), IN parola VARCHAR(20))
BEGIN
        SELECT CNP FROM utilizator WHERE utilizator.CNP = CNP
        AND utilizator.parola = parola;
END //
DELIMITER ;
```

8.24.Procedura de determinare a programului de lucru al unui medic

```
DELIMITER //
CREATE PROCEDURE programDeLucru (in CNP VARCHAR(10))
```

```
BEGIN
        SELECT nume, prenume, zi, oraStart, oraFinal, unitate FROM utilizator, orar_medic
WHERE CNP=utilizator.CNP AND CNP=orar_medic.idMedic;
END//
DELIMITER ;
```

8.25.Procedura de determinare a programului de lucru al unui asistent

```
DELIMITER //
CREATE PROCEDURE programDeLucruAsistent (in CNP VARCHAR(10))
BEGIN
        SELECT nume, prenume, unitate_medicala.denumire, program_unitate.zi,
program_unitate.oraDeschidere, program_unitate.oraInchidere FROM utilizator, asistent_medical,
unitate_medicala, program_unitate
    WHERE CNP=utilizator.CNP AND idAsistent=CNP AND
asistent_medical.unitate=unitate_medicala.denumire AND
unitate_medicala.denumire=program_unitate.unitate;
END//
DELIMITER ;
```

8.26.Procedura de determinare a profitului

```
DELIMITER //
CREATE PROCEDURE determinareFond( in denumire VARCHAR(50))
BEGIN
        SELECT unitate_medicala.denumire, unitate_medicala.fond FROM unitate_medicala
        WHERE denumire=unitate_medicala.denumire;
END //
DELIMITER ;
```

8.27.Procedura de determinare a programului unui medic

```
DELIMITER //
CREATE PROCEDURE programMedic(in idMedic VARCHAR(10))
BEGIN
        SELECT utilizator.nume, utilizator.prenume,orar_medic.zi, orar_medic.unitate,
orar_medic.oraStart, orar_medic.oraFinal
     FROM utilizator, orar_medic
    WHERE orar_medic.idMedic=idMedic AND utilizator.CNP=idMedic;
END //
DELIMITER ;
```

8.28.Procedura de afisare a concediului

```
DELIMITER //
CREATE PROCEDURE afisareConcediu(in CNP VARCHAR(10))
BEGIN
        SELECT utilizator.nume, utilizator.prenume, utilizator.concediuStart,
        utilizator.concediuFinal FROM utilizator
        WHERE utilizator.CNP=CNP;
END //
DELIMITER ;
```

8.29.Procedura de afisare profit medic

```
DELIMITER //
CREATE PROCEDURE profitMedic(in idMedic VARCHAR(10))
BEGIN
        SELECT utilizator.nume, utilizator.prenume , salarii.suma, salarii.unitate FROM
utilizator, salarii
   WHERE utilizator.CNP=idMedic AND salarii.idAngajat=idMedic;
END //
DELIMITER ;
```

8.30.Procedura de afisare salarii

```
DELIMITER //
CREATE PROCEDURE afisareSalarii(in idAngajat VARCHAR(10))
BEGIN
        SELECT utilizator.nume, utilizator.prenume , salarii.suma, salarii.unitate FROM
utilizator, salarii
   WHERE utilizator.CNP=idAngajat AND salarii.idAngajat=idAngajat;
END //
DELIMITER ;
```

Capitolul 9. Descrierea aplicației

9.1.Operații posibile

În scopul utilizării mai ușoare a bazei de date , am creat o aplicație demonstrativă prin care se pot efectua următoarele operații:

- Conectarea la baza de date, în funție de rolul pe care îl are atribuit fiecare utilizator
- Vizualizarea specializărilor oferite de lanțul de unități medicale.
- Vizualizarea salariilor primite.
- Vizualizarea oricărui tip de angajat.
- Efectuarea unei programări la un centru medical.
- Efectuarea unei consultații, întocmirea unui raport medical, parafarea unui raport medical(funție disponibilă doar medicilor), eliberarea unui bon fiscal, achitarea unei consultații.
- Afișarea tuturor unităților medicale, programului unei unități medicale.
- Adăugarea unui utilizator nou, crearea unei parole pentru utilizatorul nou.
- Ștergerea unui utilizator.
- Afișarea fondului unei unități medicale.

- Determinarea programului de lucru al unui angajat, determinarea programului unei unități medicale, determinarea concediului unui angajat.
- Stergerea unui serviciu medical, a unei unități medicale.

9.2. Tehnologii folosite

Pentru realizarea aplicației am folosit programul IDE-ul Intellij IDEA , Hibernate, MySQL WorKbench.

9.2.1. Intellij IDEA este un mediu special de programare sau mediu de dezvoltare integrat destinat în mare parte Java, este utilizat în special pentru utilizarea programelor. Folosit pentru realizarea interfeței grafice pentru utilizator.

9.2.2. Hibernate este o librărie obiectual-relațională pentru limbajul Java, oferind un framework pentru maparea dintre un model obiectual și o bază de date relațională. Hibernate rezolvă problemele tehnice legate de accesul la baza de date în vederea obținerii persistenței obiectelor prin punerea la dispoziția utilizatorului a unor funcții care lucrează direct cu obiectele Java, luând asupra lor realizarea cumunicării directe cu baza de date.

9.2.3. MySQL Workbench este un instrument vizual de proiectare a bazelor de date care integrează dezvoltarea, administrarea, proiectarea, crearea și întreținerea bazei de date SQL într-un singur mediu de dezvoltare integrat pentru sistemul de baze de date MySQL.

9.3. Conectare

Conectarea în aplicație se face în urma specificării codului numeric personal și a parolei de utilizator. Pe baza valorii din câmpul functia , se va deschide o pagină diferită pentru fiecare functie.

Această aplicație are și opțiunea de deconectare.

9.4. Vizualizare meniu

Interfața de meniu este diferită pentru utilizatori, întrucât unii utilzatori au mai multe opțiuni pentru meniu.

Toți utilizatori pot să își vizualizeze datele despre propria persoană, doar inspectorul de resurse umane are acces la informații despre orice utilizator. Utilizatorul de tip economist are acces la toate informațiile financiar-contabile despre orice persoană, iar ceilalți utilizatori au acces doar la informațiile personale din această categorie. Gestiunea activităților operaționale poate fi efectuată doar de medici, asistenți medicali sau recepționeri, de exemplu un recepționer are capacitatea de a efectua o programare, de a înregistra un pacient și de a elibera un bon fiscal, un medic are acces la istoricul medical al unui pacient, efectuează consultații, completează rapoarte medicale și poate parafa un raport medical, iar un asistent medical poate să completeze un raport medical. Înregistrarea unui utilizator nou poate fi efectuată doar de către un utilizator de tip admin, acesta poate modifica, adăuga și șterge informații legate de utilizatori, iar un super admin poate opera inclusiv asupra utilizatorilor de tip admin.

9.5. Clase

Proiectul are următoarea structură:

App.configuration

Reprezintă directorul care asigură conectarea între Java si MySQL prin folosirea Hibernate-ului.

App.controller.gui

Reprezintă directorul în care se află clasele folosite la controlul interfeței grafice.

App.exception

Reprezintă directorul care asigură afișarea erorilor.

App.model

Reprezintă directorul în care se află clasele în care au fost interpretate tabelele.

App.single_point_access

Reprezintă directorul care permite o utilizare mai ușoară a interfeței în scopul de a o modela.

App.view

Reprezintă directorul în care se află interfețele grafice folosite în cadrul proiectului

App.main

Reprezintă directorul în care se află clasa Main din care se pornește programul.

App.view.admin

Reprezintă directorul în care se află interfețele grafice care au legătură cu activitatea administratorului.

App.view.firstModule

Reprezintă directorul în care se află interfețele grafice pentru medic, asistent, recepționer și economist care sunt folosite pentru îndeplinirea funcționalităților primului modul.

App.view.firstModuleInspector

Reprezintă directorul în care se află interfețele grafice pentru inspector resurse umane care sunt folosite pentru îndeplinirea funcționalităților primului modul.

App.view.menu

Reprezintă directorul în care se află interfețele grafice pentru implementarea meniului angajaților.

App.view.role

Reprezintă directorul în care se află interfețele grafice pentru asignarea rolurilor unui angajat.

App.view.secondModule

Reprezintă directorul în care se află interfețele grafice pentru toți utilizatorii care sunt folosite pentru îndeplinirea funcționalităților pentru al doilea modul.

App.view.thirdModule

Reprezintă directorul în care se află interfețele grafice pentru toți utilizatorii care sunt folosite pentru îndeplinirea funcționalităților pentru al treilea modul.

App.view.LoginView

Reprezintă interfața grafică utilizată pentru logarea în baza de date.

Resources.hibernate.cfg.xml

Reprezintă fișierul de hibernate în care s-a realizat conectarea la baza de date.

Capitolul 10. Detalii de programare a aplicației

Pentru realizarea bazei de date am utilizat un model relațional. Toate procedurile stocate pe care le-am folosit sunt implementate folosind SQL, iar mai apoi apelate în clasele pe care le-am creat în IDE-ul Intellij .

Capitolul 11. Manual de instalare și configurări necesare

Pentru realizarea acestui proiect am folosit :

- MySQL Workbench
- Intellij IDEA
- Hibernate

Capitolul 12. Concluzii și dezvoltări ulterioare

În concluzie, dezvoltarea aplicațiilor folosind Java și MySQL este posibilă.

Aplicația creată se poate dezvolta ulterior prin adăugarea mai multor servicii medicale , prin creearea unor funcții care oferă posibilitatea eliminării repetiției unor instrucțiuni de bază, care vor reduce numărul de pagini.