



PROIECT

la disciplina
Introducere in Baze de Date

Policlinica Life-Care

Boitor Diana-Elena

Clonța Ștefania-Elena

Jarda Adina-Ionela

An academic :2023–2024

PROIECT de SEMESTRU
Catedra de Calculatoare
Disciplina: Introducere in Baze de Date
Coordonator: s.l. ing. Cosmina IVAN
Data 19.01.2024

Cuprins

1. Introducere

- Introducere, argumente
- Scop
- Obiective specifice

2. Analiza cerintelor utilizatorilor (Specificatiile de proiect)

- Ipoteze specifice domeniului ales pentru proiect (cerinte, constrangeri)
- Organizare structurata(tabelar) a cerintelor utilizator
- Determinarea si caracterizarea de profiluri de utilizatori (admin, user intern, user extern...diversi alti “actori”)

3. Modelul de date si descrierea acestuia

- Entitati si atributele lor (descriere detaliata – **implementarea fizica**)
- Diagrama EER/UML pentru modelul de date complet
- Normalizarea datelor

4. Detalii de implementare

- Descrierea functionala a modulelor (organizarea logica a acestora- de ex . structura claselor Java, cod HTML, JSP, ASP, PhP)
- Manual de utilizare/instalare (diferentiat pe tipuri de actori)
- Elemente de securizare a aplicatiei

5. Concluzii limitari si dezvoltari ulterioare

6. Bibliografie

1. Introducere

Desfășurarea activității unei policlinici reprezintă un proces extrem de complex și delicat, având în vedere impactul direct asupra sănătății pacienților. Orice schimbare intervenită în cadrul acestei activități necesită o atenție deosebită și o strictețe sporită, dat fiind că fiecare operațiune desfășurată într-o policlinică trebuie să ofere rapiditate și flexibilitate pentru a satisface în mod optim nevoile pacienților.

Unul dintre aspectele deosebit de discutate în cadrul operațiunilor desfășurate într-o policlinică se referă la modul de implementare a acestora și la software-ul utilizat pentru gestionarea eficientă a tuturor proceselor. Este esențial ca sistemul informatic să ofere nu doar rapiditate în derularea operațiunilor, ci și flexibilitatea necesară pentru a satisface diversitatea cerințelor și nevoilor pacienților.

În acest context, o componentă de o importanță crucială o reprezintă baza de date care stă la baza tuturor operațiunilor. Această bază de date trebuie să cuprindă de forma exhaustivă toate informațiile relevante referitoare la policlinică și activitățile acesteia. Aici sunt incluse programările pacienților la medici specializați, gestionarea programelor medicilor pentru a evita suprapuneri, precum și păstrarea cu rigurozitate a fișelor pacienților, împreună cu informațiile privind tratamentul prescris și medicamentele necesare.

Policlinica, în calitatea sa de furnizor de servicii medicale, trebuie să dispună de un sistem informatic robust și adaptabil, care să asigure nu doar eficiența operațiunilor curente, dar și securitatea și confidențialitatea datelor pacienților. Îmbunătățirea continuă a acestor sisteme informatice este

esențială pentru a răspunde evoluției cerințelor din domeniul medical și pentru a asigura un standard înalt de calitate în îngrijirea sănătății.

2. Analiza cerintelor utilizatorilor

Dorința de a ușura munca angajaților unei policlinici ne-a determinat să alegem acest proiect și să îl implementăm conform cerințelor. Proiectul este făcut cu ajutorul aplicațiilor IntelliJ și MySQLWorkbench. Rolurile importante ale proiectului sunt admin, superadmin, medic, asistent, recepționar și utilizatorul. Am creat o interfață ușor de utilizat astfel încât medicii și asistenții să poată completa ușor date despre pacienții acestora. Fiecare utilizator își poate accesa datele personale, iar angajații au opțiunea de a-și vedea salariile.

Rolurile și funcționalitățile acestora:

1. Admin (Administrator)

- Are acces complet la toate funcționalitățile sistemului.
- Poate administra policlinica, utilizatorii, și angajații.
- Responsabil pentru gestionarea globală a sistemului.

2. Superadmin (Superadministrator)

- Are aceleași privilegii ca și administratorul.
- Poate gestiona și configura aspecte avansate ale sistemului.

3. Pacient

- Poate realiza programări doar sunand la recepția policlinicii.
- Accesează istoricul programărilor și rapoartele medicale personale.
- Poate primi bonuri fiscale pentru serviciile medicale primite.

4. Medic

- Poate gestiona programările și rapoartele medicale ale pacienților săi.
- Are acces la informații despre pacienții săi.

5. Asistent Medical

- Completează rapoartele medicale.

6. Recepționar

- Realizează programările la cererea pacienților.

- Emite bonuri fiscale.

7. HR (Resurse Umane)

- Vederea tuturor datelor angajatilor.

8. Economic

- Gestionarea aspectelor economice ale programului, inclusiv costurile serviciilor medicale.

3. Modelul de date si descrierea acestuia

Baza de date este alcătuită din 12 tabele, fiecare având un rol foarte important în gestionarea tuturor informațiilor.

1. Tabelul policlinici

Atribute:

ID_Policlinica (Cheie Primară): Identificator unic pentru policlinică.

Denumire: Numele policlinicii.

Adresa: Adresa policlinicii.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`policlinici` (
  `ID_Policlinica` INT NOT NULL,
  `Denumire` VARCHAR(100) NOT NULL,
  `Adresa` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`ID_Policlinica`)) ;
```

2. Tabelul utilizatori

Atribute:

ID_Utilizator (Cheie Primară): Identificator unic pentru utilizator.

CNP (Unic): Numărul de identificare personal.

Parola: Parola utilizatorului.

Nume: Numele de familie al utilizatorului.

Prenume: Prenumele utilizatorului.

Adresa: Adresa utilizatorului.

Telefon: Numărul de telefon al utilizatorului.

Email: Adresa de email a utilizatorului.

Cont_IBAN: IBAN-ul utilizatorului.

Numar_Contract: Numărul contractului utilizatorului.

Data_Angajarii: Data angajării.

ID_Policlinica (Cheie Externă): Se leagă de ID_Policlinica din tabelul policlinici.

rol_user: Rolul utilizatorului (admin, superadmin, pacient, medic, asistent, hr, economic, receptioner).

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`utilizatori` (  
  `ID_Utilizator` INT NOT NULL,  
  `CNP` VARCHAR(13) NOT NULL,  
  `Parola` VARCHAR(45) NOT NULL,  
  `Nume` VARCHAR(50) NOT NULL,  
  `Prenume` VARCHAR(50) NOT NULL,  
  `Adresa` VARCHAR(255) NOT NULL,  
  `Telefon` VARCHAR(15) NOT NULL,  
  `Email` VARCHAR(100) NOT NULL,  
  `Cont_IBAN` VARCHAR(30) NOT NULL,  
  `Numar_Contract` INT NOT NULL,  
  `Data_Angajarii` DATE NOT NULL,  
  `ID_Policlinica` INT NULL DEFAULT NULL,  
  `rol_user` ENUM('admin', 'superadmin', 'pacient', 'medic', 'asistent', 'hr', 'economic',  
'receptioner') NOT NULL);
```

3. Tabelul angajati

Atribute:

ID_Angajat (Cheie Primară): Identificator unic pentru angajat.

ID_Utilizator (Cheie Externă): Se leagă de ID_Utilizator din tabelul utilizatori.

Salariu: Salariul angajatului.

Ore_Lucrate: Orele lucrate de angajat.

Tip_Angajati: Tipul de angajat (HR, RECEPTIONER, ECONOMIC, ASISTENT, MEDIC).

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`angajati` (  
  `ID_Angajat` INT NOT NULL,  
  `ID_Utilizator` INT NULL DEFAULT NULL,  
  `Salariu` DECIMAL(10,2) NOT NULL,  
  `Ore_Lucrate` INT NOT NULL,  
  `Tip_Angajati` ENUM('HR', 'RECEPTIONER', 'ECONOMIC', 'ASISTENT', 'MEDIC')  
  NULL DEFAULT NULL);
```

4. Tabelul asistentmedicali

Atribute:

ID_Asistent (Cheie Primară): Identificator unic pentru asistentul medical.

ID_Angajat (Cheie Externă): Se leagă de ID_Angajat din tabelul angajati.

Tip_Asistent: Tipul de asistent medical.

Grad_Asistent: Gradul asistentului medical.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`asistentmedicali` (  
  `ID_Asistent` INT NOT NULL,  
  `ID_Angajat` INT NULL DEFAULT NULL,  
  `Tip_Asistent` VARCHAR(20) NOT NULL,  
  `Grad_Asistent` VARCHAR(20) NOT NULL);
```

5. Tabelul medici

Atribute:

ID_Medic (Cheie Primară): Identificator unic pentru medic.

ID_Angajat (Cheie Externă): Se leagă de ID_Angajat din tabelul angajati.

Specialitate: Specializarea medicală.

Grad_Medic: Gradul medicului.

Cod_Parafă: Codul de parafă al medicului.

Competente: Competențele medicului.

Titlu_Stiintific: Titlul științific al medicului.

Post_Didactic: Poziția didactică a medicului.

Procent_Servicii: Procentul de servicii furnizate de medic.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`medici` (  
  `ID_Medic` INT NOT NULL,  
  `ID_Angajat` INT NULL DEFAULT NULL,  
  `Specialitate` VARCHAR(50) NOT NULL,  
  `Grad_Medic` VARCHAR(20) NOT NULL,  
  `Cod_Parafa` VARCHAR(20) NOT NULL,  
  `Competente` VARCHAR(255) NOT NULL,  
  `Titlu_Stiintific` VARCHAR(50) NULL DEFAULT NULL,  
  `Post_Didactic` VARCHAR(50) NULL DEFAULT NULL,  
  `Procent_Servicii` DECIMAL(5,2) NOT NULL);
```

6. Tabelul pacienti

Atribute:

ID_Pacient (Cheie Primară): Identificator unic pentru pacient.

ID_Utilizator (Cheie Externă): Se leagă de ID_Utilizator din tabelul utilizatori.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`pacienti` (  
  `ID_Pacient` INT NOT NULL,  
  `ID_Utilizator` INT NULL DEFAULT NULL);
```

7. Tabelul programari

Atribute:

ID_Programare (Cheie Primară): Identificator unic pentru programare.

ID_Pacient (Cheie Externă): Se leagă de ID_Pacient din tabelul pacienti.

ID_Medic (Cheie Externă): Se leagă de ID_Medic din tabelul medici.

Data_Programare: Data programării.

Ora_Programare: Ora programării.

ID_Serviciu (Cheie Externă): Se leagă de ID_ServiciiDisponibile din serviciidisponibile.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`programari` (  
  `ID_Programare` INT NOT NULL,  
  `ID_Pacient` INT NULL DEFAULT NULL,
```

```
`ID_Medic` INT NULL DEFAULT NULL,  
`Data_Programare` DATE NOT NULL,  
`Ora_Programare` INT NOT NULL,  
`ID_Serviciu` INT NULL DEFAULT NULL);
```

8. Tabelul bonurifiscale

Atribute:

ID_Bon (Cheie Primară): Identificator unic pentru bonul fiscal.

ID_Programare (Cheie Externă): Se leagă de ID_Programare din programari.

Data_Emitere: Data emiterii bonului fiscal.

Valoare: Valoarea bonului fiscal.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`bonurifiscale` (  
  `ID_Bon` INT NOT NULL,  
  `ID_Programare` INT NULL DEFAULT NULL,  
  `Data_Emitere` DATE NOT NULL,  
  `Valoare` DECIMAL(8,2) NOT NULL);
```

9. Tabelul programpoliclinica

Atribute:

ID_Program (Cheie Primară): Identificator unic pentru programul policlinicii.

ID_Policlinica (Cheie Externă): Se leagă de ID_Policlinica din tabelul policlinici.

Zi_Saptamana: Ziua săptămânii pentru program.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`programpoliclinica` (  
  `ID_Program` INT NOT NULL,  
  `ID_Policlinica` INT NULL DEFAULT NULL,  
  `Zi_Saptamana` VARCHAR(10) NOT NULL);
```

10. Tabelul rapoartemedicale

Atribute:

ID_Raport (Cheie Primară): Identificator unic pentru raportul medical.

ID_Programare (Cheie Externă): Se leagă de ID_Programare din tabelul programari.

Continut: Conținutul raportului.

ID_Asistent (Cheie Externă): Se leagă de ID_Asistent din tabelul asistentimedicali.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`rapoartemedicale` (  
  `ID_Raport` INT NOT NULL,  
  `ID_Programare` INT NULL DEFAULT NULL,  
  `Continut` TEXT NOT NULL,  
  `ID_Asistent` INT NOT NULL);
```

11. Tabelul serviciidisponibile

Atribute:

ID_ServiciiDisponibile (Cheie Primară): Identificator unic pentru serviciul disponibil.

Grad: Gradul serviciului.

Cost: Costul serviciului.

Durata: Durata serviciului.

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`serviciidisponibile` (  
  `ID_ServiciiDisponibile` INT NOT NULL,  
  `Grad` INT NOT NULL,  
  `Cost` INT NOT NULL,  
  `Durata` INT NOT NULL);
```

12. Tabelul serviciispecifice

Atribute:

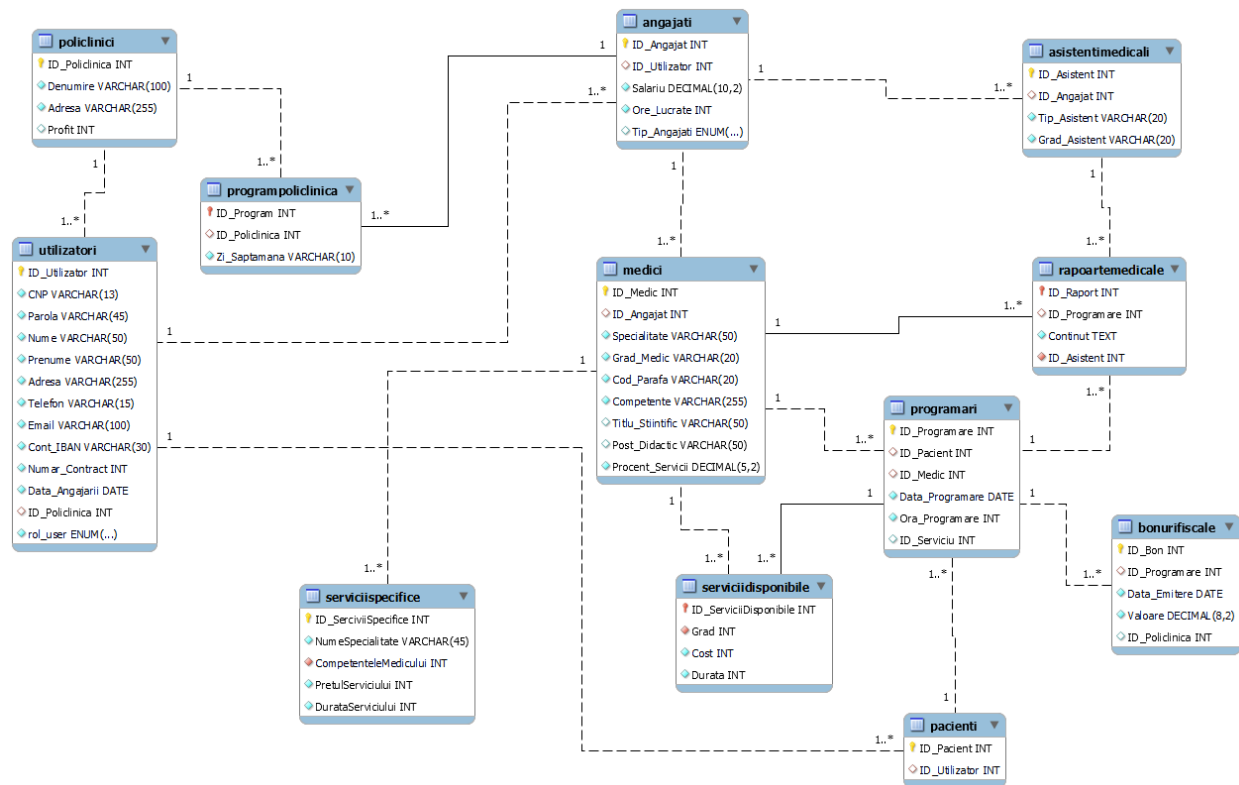
ID_ServiciiSpecifice (Cheie Primară): Identificator unic pentru serviciul specific.

NumeSpecialitate: Numele specializării pentru serviciu

Implementare:

```
CREATE TABLE IF NOT EXISTS `proiect`.`serviciispecifice` (  
  `ID_ServiciiSpecifice` INT NOT NULL,  
  `NumeSpecialitate` VARCHAR(45) NOT NULL,  
  `CompetenteleMedicului` INT NOT NULL,  
  `PretulServiciului` INT NOT NULL,  
  `DurataServiciului` INT NOT NULL);
```

Diagrama ER



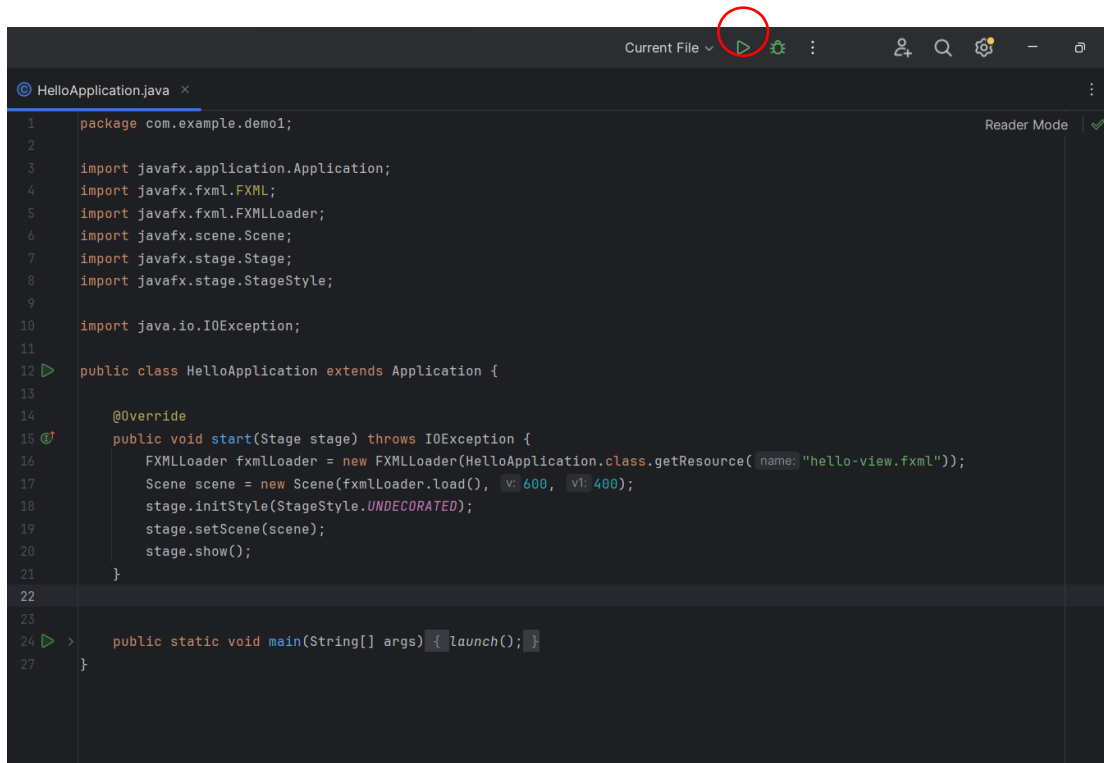
În dezvoltarea unei aplicații pentru manipularea bazei de date, aspectul cel mai crucial este proiectarea corectă a structurii bazei de date. Aceasta este esențială deoarece timpul de acces la date depinde în mare măsură de modul în care sunt organizate și pentru că integritatea referențială a datelor este asigurată de această structură.

Normalizarea reprezintă un proces important în proiectarea bazei de date, având ca scop simplificarea unei structuri complexe de date prin împărțirea ei în entități mai simple și interconectate între ele. Atunci când ne străduim să normalizăm un tabel, vrem să reducem redundanța datelor în acest tabel și să îmbunătățim integritatea informațiilor pe care le furnizează. Este un proces de optimizare a modului în care datele sunt stocate și gestionate în cadrul sistemului, cu beneficii semnificative în ceea ce privește eficiența și consistența informațiilor.

4. Detalii de implementare

Manual de utilizare

Aplicația necesită cunoștințe minime de operare, deoarece are o interfață simplă, pe care o poate folosi oricine. Aplicația se deschide din IntelliJ, din clasa HelloApplication.



```
1 package com.example.demo1;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXML;
5 import javafx.fxml.FXMLLoader;
6 import javafx.scene.Scene;
7 import javafx.stage.Stage;
8 import javafx.stage.StageStyle;
9
10 import java.io.IOException;
11
12 public class HelloApplication extends Application {
13
14     @Override
15     public void start(Stage stage) throws IOException {
16         FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
17         Scene scene = new Scene(fxmlLoader.load(), 600, 400);
18         stage.initStyle(StageStyle.UNDECORATED);
19         stage.setScene(scene);
20         stage.show();
21     }
22
23
24     public static void main(String[] args) { launch(); }
25
26 }
```

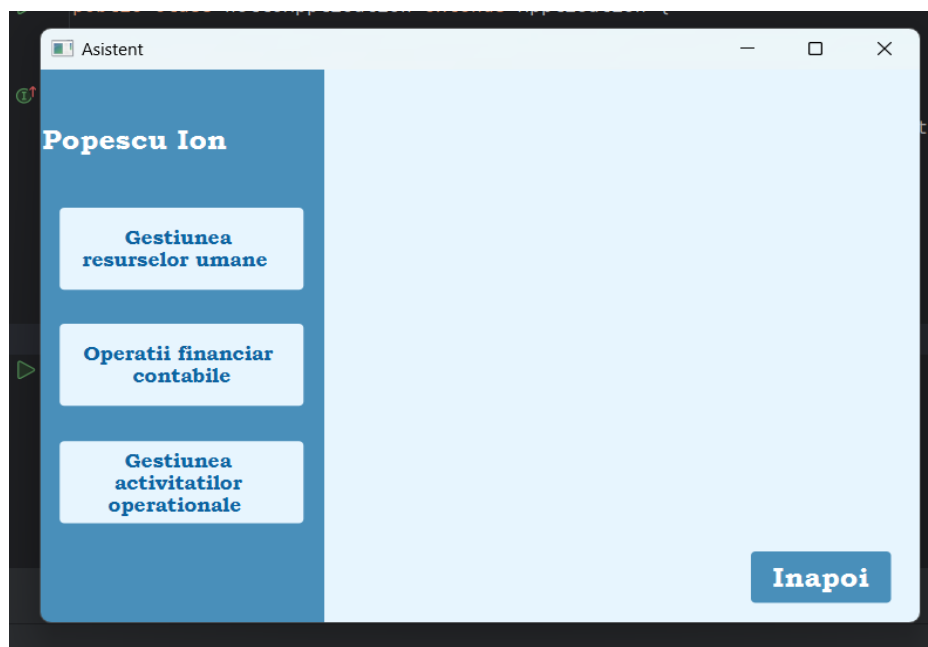
Odată cu rularea, se va deschide interfața în care trebuie să se introducă datele utilizatorului. Fiecare angajat are ca date de conectare un user, adică CNP-ul acestuia, și o parolă.



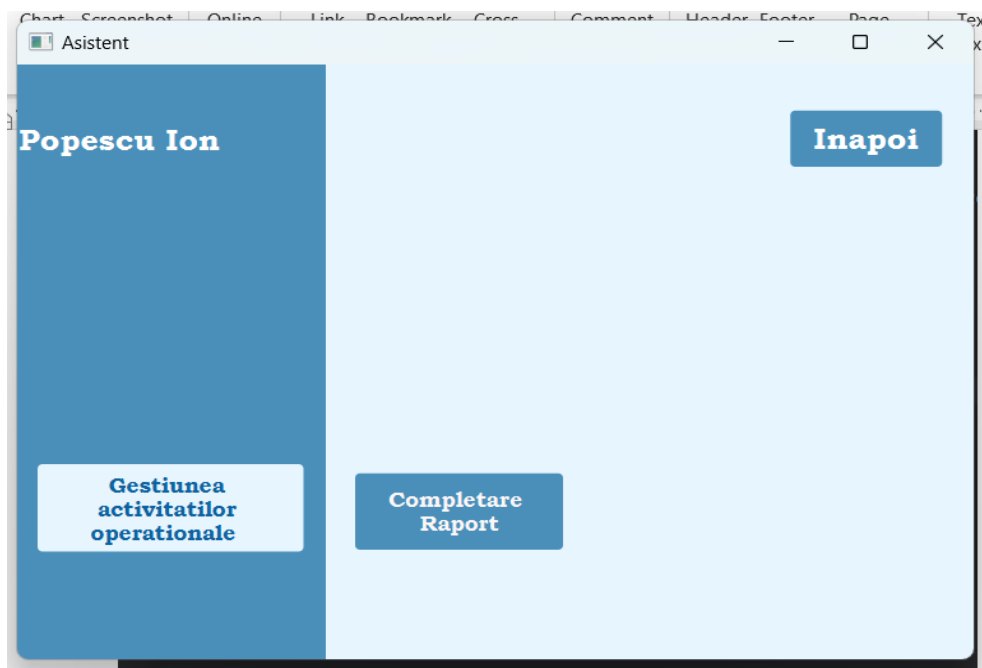
În cazul în care utilizatorul a introdus greșit user-ul sau parola, se va afișa un mesaj de eroare.



Luăm spre exemplu, conectarea unui asistent. După introducerea datelor, dacă acestea sunt corecte, se va deschide o nouă fereastră care conține mai multe butoane corespunzătoare unei acțiuni.



Asistentul poate vedea datele personale, salariul, și poate completa raportul unui pacient. Acesta poate naviga în interfață cu ajutorul celor trei butoane și butonul 'Inapoi'.



5. Concluzii limitari si dezvoltari ulterioare

In concluzie rolul aplicației noastre este acela de a simula o baza de date din cadrul unei policlinici.

Modul simplificat creat cu ajutorul aplicațiilor IntelliJ și SceneBuilder fac să poată fi posibilă legarea bazei de date la interfață cu ajutorul conectorului JDBC. Prin intermediul acestuia, este posibilă preluarea datelor din baza de date.

In MYSQL se pot realiza afisarea a diferitelor tabele, crearea a noi legături, proceduri sau triggere.

Dezvoltari ulterioare posibile ar fi îmbunătățirea interfeței, folosirea mai multor imagini și lucrarea la aspectul acesteia. Baza de date ar putea să fie îmbunătățită, prin gestionarea unui lanț de policlinici și permiterea medicilor de a lucra în mai multe locuri.

6. Bibliografie

- <http://www.mysql.com>
- <http://www.w3schools.com/sql>
- https://www.youtube.com/watch?v=Z1W4E2d4Yxo&ab_channel=GenuineCoder
- https://docs.oracle.com/javase/8/scene-builder-2/get-started-tutorial/jfxsb-get_started.htm

Anexa

Proceduri

- Calcularea Salariului pe oră

DELIMITER //

CREATE PROCEDURE CalculSalariuOra(IN medicID INT)

BEGIN

DECLARE salariuLunar DECIMAL(10, 2);

DECLARE oreLucrate INT;

DECLARE salariuOra DECIMAL(10, 2);

-- Obține salariul lunar al medicului

SELECT Salariu INTO salariuLunar

FROM proiect.medici m

JOIN proiect.angajati a ON m.ID_Angajat = a.ID_Angajat

WHERE m.ID_Medic = medicID;

-- Obține numărul de ore lucrate de medic

SELECT Ore_Lucrate INTO oreLucrate

FROM proiect.angajati

WHERE ID_Angajat = medicID;

-- Calculează salariul pe oră

SET salariuOra = salariuLunar / oreLucrate;

-- Afișează salariul pe oră

SELECT salariuOra AS SalariuOra;

END //

DELIMITER ;

- Selectarea datelor personale

DELIMITER //

```
CREATE PROCEDURE SelectUtilizator(IN utilizatorID INT)
BEGIN
    SELECT Nume, Prenume, CNP, rol_user, Email, Adresa, Telefon, Cont_IBAN, Numar_Contract,
    Data_Angajarii
    FROM proiect.utilizatori
    WHERE ID_Utilizator = utilizatorID;
END //
```

DELIMITER ;

- Rapoartele medicului

DELIMITER //

```
CREATE PROCEDURE SelectRaportMedic(IN utilizatorID INT)
BEGIN
    SELECT u.nume, u.prenume, r.Continut, r.ID_Raport
    FROM proiect.utilizatori u
    JOIN proiect.programari p ON u.ID_Utilizator = p.ID_Pacient
    JOIN proiect.rapoartemedicale r ON p.ID_Programare = r.ID_Programare
    WHERE u.ID_Utilizator = utilizatorID;
END //
```

DELIMITER ;

- Procentul medicului din programari

DELIMITER //

```
CREATE PROCEDURE CalculProfitMedic(IN medicID INT)
BEGIN
    DECLARE medicProfit DECIMAL(10,2);

    SELECT SUM(p.Valoare * m.Procent_Servicii / 100) INTO medicProfit
    FROM proiect.medici m
    JOIN proiect.programari pr ON m.ID_Medic = pr.ID_Medic
```

```
JOIN proiect.bonurifiscale p ON pr.ID_Programare = p.ID_Programare
WHERE m.ID_Medic = medicID;
```

```
SELECT medicProfit AS ProfitMedic;
END //
```

```
DELIMITER ;
```

- Salarii angajati

```
DELIMITER //
```

```
CREATE PROCEDURE SelectSalariuAngajat(IN utilizatorID INT)
BEGIN
    SELECT u.nume, u.prenume, a.salariu
    FROM proiect.utilizatori u
    JOIN proiect.angajati a ON u.ID_Utilizator = a.ID_Utilizator
    WHERE u.ID_Utilizator = utilizatorID;
END //
```

```
DELIMITER ;
```

- Calculul profitului policlinicii procentaj

```
DELIMITER //
```

```
CREATE PROCEDURE CalculProfitPoliclinica(IN medicID INT)
BEGIN
    DECLARE medicProfit DECIMAL(10, 2);
    DECLARE policlinicaProfit DECIMAL(10, 2);

    -- Calculează profitul medicului
    SELECT SUM(p.Valoare * m.Procent_Servicii / 100) INTO medicProfit
    FROM proiect.medici m
    JOIN proiect.programari pr ON m.ID_Medic = pr.ID_Medic
    JOIN proiect.bonurifiscale p ON pr.ID_Programare = p.ID_Programare
    WHERE m.ID_Medic = medicID;

    -- Calculează profitul policlinicii pentru medic
    SELECT SUM(p.Valoare) INTO policlinicaProfit
    FROM proiect.programari pr
```

```

JOIN proiect.bonurifiscale p ON pr.ID_Programare = p.ID_Programare
WHERE pr.ID_Medic = medicID;

-- Afîșează profitul medicului
SELECT medicProfit AS ProfitMedic;

-- Afîșează profitul policlinicii pentru medic
SELECT policlinicaProfit AS Profit;
END //

DELIMITER ;

```

Trigger

- Actualizează profitul policlinicii după fiecare emiteră de bon
- DELIMITER //

```

CREATE TRIGGER actualizeaza_venit
AFTER INSERT ON proiect.bonurifiscale
FOR EACH ROW
BEGIN
    DECLARE user_policlinica INT;

    SELECT policlinici.ID_Policlinica INTO user_policlinica
    FROM policlinici
    JOIN bonurifiscale ON policlinici.ID_Policlinica = bonurifiscale.ID_Policlinica
    JOIN pacienti ON pacienti.ID_Pacient = NEW.ID_Programare
    WHERE bonurifiscale.ID_Programare = NEW.ID_Programare
    LIMIT 1;

    IF user_policlinica IS NOT NULL THEN
        -- Verificăm și actualizăm corect coloana corespunzătoare (Profit sau Venit, depinde de denumirea
coloanei)
        UPDATE proiect.policlinici
        SET Profit = Profit + NEW.Valoare
        WHERE ID_Policlinica = user_policlinica;
    END IF;
END;
//
DELIMITER ;

```