

Universitatea Tehnică Cluj-Napoca
Facultatea de Automatică și Calculatoare



Proiect de Semestru

la disciplina
Introducere în Baze de Date

Lanț de Policlinici

Trei Frați Pătați

Studenti : Hoțupan Rareș

Lînaru Petra

Mudura Ana

Grupa : 30221

An academic 2021 – 2022

Cuprins

- 1.Introducere
- 2.Analiza cerințelor
 - 2.1 Cerințe și constrângeri
 - 2.2 Organizarea structurată tabelar a cerințelor utilizatorilor
 - 2.3 Determinarea și caracterizarea profilurilor de utilizator
- 3.Modelul de date și descrierea acestuia
 - 3.1 Entități și atributele lor
 - 3.2 Diagrama EER/UML pentru modelul de date complet
 - 3.3 Proceduri, triggere și view-uri
 - 3.4 Normalizarea datelor
 - 3.5 Interogări MySql
 - 3.6 Cod MySql
 - 3.7 Cod pentru crearea bazei de date și a tabelelor
 - 3.8 Cod Proceduri
 - 3.9 Cod Useri
 - 3.10 Cod View-uri
 - 3.11 Cod Trigger
- 4 Detalii de implementare
 - 4.1 Structura claselor în Java
 - 4.2 Interfața grafică
 - 4.3 Manual de utilizare/Instalare

1. Introducere

Acest proiect propune implementarea unei aplicații care interacționează cu un sistem informatic de gestiune pentru baze de date, respectiv cu un sistem destinat gestiunii activităților dintr-un lanț de policlinici. Această aplicație posedă o interfață grafică prin care utilizatorul poate interacționa cu sistemul de gestiune al datelor, oferind funcționalități utilizatorilor conform cu rolurile care sunt atribuite acestora.

Totodată, această aplicație a fost proiectată astfel încât să îndeplinească nevoile întregului personal al unei policlinici. Structura acesteia a fost proiectată în așa fel încât să cuprindă informații cu privire la gestiunea angajaților, a pacienților și a activităților medicale, precum informații referitoare la serviciul financiar-contabil. Funcționalitățile sale au fost gândite astfel încât toate activitățile din cadrul unei unități să poată fi înregistrate, arhivate și refolosite cu ușurință, mulate pe nevoile utilizatorilor săi.

Aplicația poate fi accesată, pe baza unui proces de autentificare, de către mai multe tipuri de utilizatori. Aceștia operează în departamentele medical, resurse umane, financiar-contabil sau medical. De asemenea, există două roluri importante : administrator și super-administrator, care supraveghează funcționarea în condiții optime a întregului sistem de gestiune.

Scopul nostru a fost acela de a proiecta un sistem informatic care să se potrivească cu nevoile tuturor, să fie ușor de utilizat și să ofere utilizatorului informațiile necesare în timp util. De asemenea, această implementare vine în ajutorul unor probleme întâlnite în viața reală:

- Gestiunea operațiilor financiar-contabile este monitorizată, calculele din spatele oricăror activități sunt făcute automat, nu permitem introducerea sumelor în sistem
- Programele de funcționare sunt automatizate, la fel și programările pacienților, astfel că evităm pe cât de mult se poate așteptarea în fața cabinetului
- Angajații își pot vizualiza și monitoriza activitatea în contul personal de utilizator
- Datele tuturor utilizatorilor, precum și ale pacienților sunt securizate

- Datele duplicate se încearcă a fi eliminate, pe cât posibil

Pentru implementarea proiectului au fost folosite:

- MySQL Workbench - pentru crearea bazei de date, popularea acesteia, scrierea de view-uri, proceduri și trigger-uri și pentru crearea diagramei UML a tabelor
- Eclipse/Intelij – mediu de dezvoltare Java

2. Analiza cerințelor

2.1 Cerințe și constrângeri

În primul rând, acest mediu în care se lucrează cu datele personale ale oamenilor, pacienți sau angajați, trebuie să fie unul securizat, structurat pe nivele de securitate și categorii de roluri. Utilizatorul de tip Administrator gestionează datele legate de utilizatori, în timp ce un utilizator de tip Super-Administrator gestionează inclusiv utilizatorii de tip Administrator. De asemenea, există utilizatori de tip angajat, care pot deține în cadrul lanțului de policlinici diferite roluri : inspector resurse umane, expert financiar contabil, receptioner, asistent medical și medic.

Lanțul de policlinici oferă un set de servicii medicale , astfel că baza de date trebuie să cuprindă informații referitoare la medicul care oferă respectivul serviciu medical, specialitatea, prețul asociat și durata. Pentru un medic se vor reține suplimentar următoarele informații : gradul , codul de parafă, titlul științific, postul didactic (dacă există). Pentru un asistent se va reține suplimentar tipul și gradul.

În funcție de drepturile pe care le dețin și rolurile pe care le îndeplinesc în policlinică, angajații pot vizualiza anumite informații din sistemul informatic. Sistemul este format din mai multe module : gestiunea resurselor-umane , gestiunea operațiilor financiar-contabile și gestiunea activității medicale. Astfel, aceste module, prin nivelele de acces și drepturile care sosesc o dată cu ele, păstrează confidențialitatea datelor celorlalți utilizatori.
















Modulul pentru gestiunea resurselor umane permite inspectorului căutarea oricărui angajat și afișarea numelui și a prenumelui său, funcția , precum și orarul de muncă și perioada de concediu. Fiecare angajat are un orar prestabilit între anumite ore, în anumite zile ale săptămânii. Cabinetele în care lucrează angajații prezintă și ele orare de funcționare, precum și întreaga unitate medicală.




Angajații de tip inspector resurse umane și expert financiar pot consulta orarul săptămânal și concediile angajaților. Restul angajaților, în cadrul acestui modul își pot vizualiza doar informațiile proprii.

Modulul pentru operații financiar contabile permite unui expert vizualizarea profitului policlinicii. Acest modul permite angajaților săi vizualizarea istoricului salariilor lor, al profitului (în cazul în care exista) și al comisionului.

Modulul pentru gestiunea activităților operaționale permite unui recepționar să programeze un pacient la o anumită dată și la un anumit medic. Pentru fiecare pacient consultat, medicul va completa ulterior un raport medical referitor la starea de sănătate a pacientului, diagnostic și recomandări. După completarea raportului medical, acesta este parafat de către medic. Un medic poate lucra în mai multe policlinici în funcție de programul său, acesta fiind repartizat cabinetului care este conform cu specializarea sa. De asemenea, un medic poate fi identificat ca angajat prin CNP-ul său, apoi acesta poate fi identificat ca medic dintr-un anumit cabinet prin codul său de specializare.

Drepturile utilizatorilor aparținând unui departament sunt prezentate mai jos:

departament / tip angajat modul	resurse umane	economic	medical		
			recepționar	asistent medical	medic
gestiunea resurselor umane		 doar date referitoare la propria persoană	 doar date referitoare la propria persoană	 doar date referitoare la propria persoană	 doar date referitoare la propria persoană
operații financiar contabile	 doar date referitoare la propria persoană		 doar date referitoare la propria persoană	 doar date referitoare la propria persoană	 doar date referitoare la propria persoană + profitul propriu
gestiunea activităților operaționale			 doar submodulele programare, înregistrare pacient, emitere bon fiscal	 doar submodulul raport medical analize	 doar submodulele istoric și raport medical

Legendă	
	utilizatorul are drepturi de citire și de scriere
	utilizatorul are doar drepturi de citire / limitate la anumite funcționalități
	utilizatorul nu are nici un fel de drepturi

a. Organizarea structurată tabelar a cerințelor utilizatorilor

Sistemul de gestiune al informațiilor trebuie să cuprindă următoarele date:

- Unitățile medicale
- Programul de funcționare al fiecărei unități medicale
- Cabinetele și specializările acestora
- Programul de funcționare al fiecărui cabinet
- Utilizatorii sistemului de gestiune (respectiv angajații, indiferent de rolul pe care îl au în cadrul modulelor)
- Orarul specific al fiecărui utilizator (angajat) și concediile
- Funcțiile fiecărui angajat, specializarea , informații referitoare la competențele și specializările acestuia (în cazul în care există)
- Repartizarea angajaților în cabinete, respectiv module (resurse umane, financiar contabil , activități medicale)
- Serviciile oferite atât de cabinet, cât și de angajați
- Informații referitoare la pacienți, rapoarte medicale, istoric medical, istoricul analizelor
- Informații referitoare la salariile angajaților, încasările clinicilor, profit, cheltuieli, pierderi (dacă există)

b. Determinarea și caracterizarea profilurilor de utilizator

1. Administrator și Super-Administrator

- adaugă, modifică și șterge informații în baza de date legate de utilizatori
- asigură funcționarea în condiții optime a bazei de date
- Super-Administratorul are drepturi și asupra utilizatorilor de tip Administrator

2. Angajat

Angajații pot îndeplini anumite roluri :

2.1 Inspector resurse umane (Resurse Umane)

- are acces la datele angajaților
- are drepturi de citire și scriere asupra datelor angajaților
- poate consulta orarul săptămânal și concediile angajaților

2.2 Expert financiar-contabil (Financiar-Contabil)

- poate consulta orarul săptămânal și concediile angajaților
- poate vizualiza informații cu privire la profitul unității medicale, cheltuieli, plățile realizate de pacienți

2.3 Recepționar (Financiar-Contabil)

- poate realiza o programare pentru un pacient, la un anumit cabinet din cadrul unității medicale
- emite bonul fiscal ulterior consultatiei
- poate vizualiza doar informațiile proprii în ceea ce privește gestiunea resurselor umane și operațiunile financiar contabile

2.4 Asistent Medical

- poate completa informații în rapoartele pentru analizele medicale
- poate vizualiza doar informațiile proprii în ceea ce privește gestiunea resurselor umane și operațiunile financiar contabile

2.5 Medic

- poate vizualiza lista de pacienți programați la el pentru ziua calendaristică în curs
- poate consulta istoricul medical al pacientului
- completează raportul medical al pacientului
- gestionează lista de servicii medicale oferite
- parafează rapoartele medicale
- poate vizualiza doar informațiile proprii în ceea ce privește gestiunea resurselor umane și operațiunile financiar contabile

3. Modelul de date și descrierea acestuia

3.1 Entități și atributele lor

Useri_Clinică (departament, CNP , nume , prenume, adresa, nr_telefon, email, cont IBAN, nr_contract, data_angajării, funcția). Aceasta tabela ofera informații cu privire la utilizator, departamentul din care face parte și funcția pe care o îndeplinește. Un user se identifică prin intermediul CNP-ului.

Funcție (CNP, inspector_resurse_umane, expert_financiar_contabil, receptioner, asistent_medical, medic, cod_specializare) . Prin aceasta tabelă, fiecărui utilizator i se validează funcția pe care o are în unitatea medicală respectivă și este redirecționat către tabela aferentă funcției sale și tabela de salarii. Atributele acestei tabele sunt de tip tiny int, corespunzător valorilor boolene adevărat -1 și fals -0, iar legăturile către alte tabele sunt realizate printr-un cod de specializare. Dacă un utilizator este inspector de resurse umane sau un expert financiar contabil , tabela de funcție redirecționează spre tabelele program_funcționar și concediu. Dacă un utilizator este asistent medical , tabela de funcție redirecționează spre tabela corespunzătoare asistenților medicali.

Asistent (grad, tip, cod_specializare) Această tabelă reține informații despre pregătirea și competențele unui asistent.

Medic (nume , prenume, specialitatea, gradul, codul de parafă, titlul științific , postul didactic, cod_specializare, ID_cabinet). Această tabelă reține informații cu privire la medicii unei unități medicale și face legătura cu tabela programărilor prin intermediul tabelelor de diagnostic și de analize. Am plecat de la presupunerea că o unitate medicală are un număr predefinit de cabinete și cabinetele sunt repartizate câte unul pe câte o specializare, iar medicii, în funcție de specializările lor, pot lucra în mai multe cabinete medicale din unități diferite . Practic, cabinetele sunt „statice” , medicii se pot modifica. Astfel, în momentul în care un pacient intră în unitatea medicală, acesta își face o programare la un anume cabinet, nu la un anume doctor.

Program_Funcționar (ID, luni, marți, miercuri, joi, vineri , sâmbătă, duminica, locație, ID_concediu). Această tabelă conține informații referitoare la orarul fiecărui angajat în parte și face legătura cu tabela de concediu aferent fiecărui angajat prin ID_concediu.

Concediu (ID, dată_început, dată_sfârșit). Această tabelă reține data de început și de final pentru concediile fiecărui angajat.

Salariu (salariu, comision , CNP). Această tabelă reține în formații referitoare la salariul fiecărui angajat, comision și CNP-ul prin care se identifică un angajat.

Diagnostic (ID_Parafă, descriere_simptome, diagnostic, medicamentație, observații). Această tabelă oferă informații referitoare la diagnosticul pe care medicul îl pune unui pacient. Medicul este identificat prin ID-ul de parafă. Tabela de diagnostic redirecționează apoi spre tabela de analize, prin ID-ul de parafă al medicului.

Analize (ID_Parafă, ID_programare, denumire_analize, pret, durata controlului, valoare_normală, rezultate). Această tabelă cuprinde informațiile aferente analizelor făcute ulterior controlului medical (realizat de către medicul identificat prin ID_Parafă), în cadrul unei programări din unitatea medicală (programare identificată prin ID_programare).

Cabinet_Medical (ID, descriere, medic) . Această tabelă oferă informații cu privire la cabinetele medicale dintr-o anumită unitate medicală.

Program_Cabinet (ID, luni, marți, miercuri, joi , vineri, sâmbătă, duminică) . Această tabelă oferă informații referitoare la programul de funcționare al unui cabinet medical. Programul unui cabinet medical poate coincide sau nu cu programul medicului.

Unitate_Medicală (ID, adresă, denumire, descriere_servicii) . Această tabelă cuprinde informații referitoare la toate unitățile medicale dintr-un lanț de policlinici, unități care pot fi identificate printr-un ID).

Program (ID, luni, marți, miercuri, joi, vineri, sâmbătă, duminică). Această tabelă oferă informații cu privire la programul de funcționare al unei unități medicale.

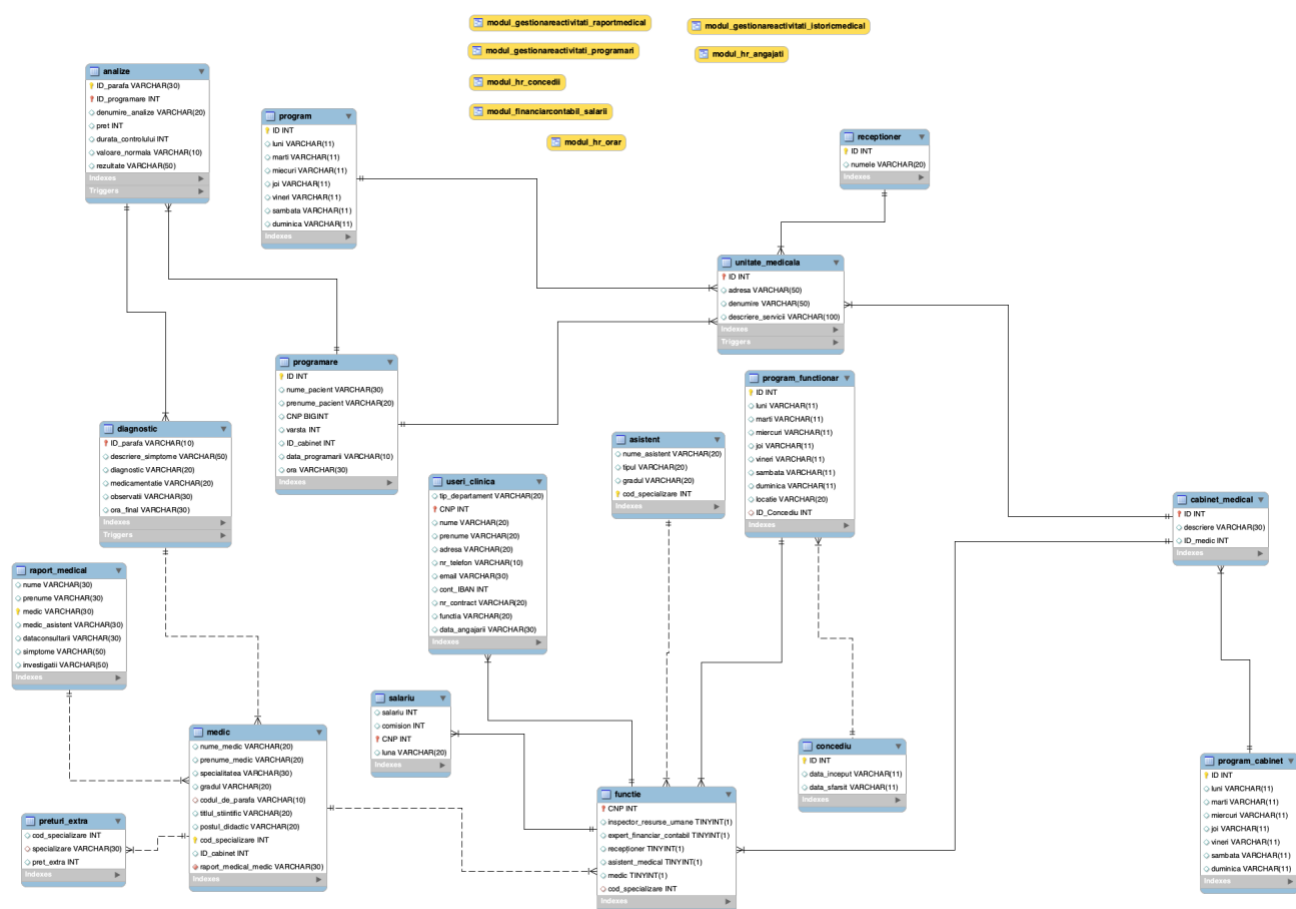
Recepționar (ID, numele). Această tabelă oferă informații cu privire la un recepționar dintr-o unitate medicală.

Programare (ID, nume_pacient, prenume_pacient, varsta, ID_cabinet, data_programării, ora). Această tabelă este una destul de importantă deoarece oferă informații legate de programările pacienților în cadrul unei anumite unități medicale, însă, mai interesant , acesta este un inel important din spatele logicii prin care medicul este asociat unui cabinet medical din unitate și prin care acesta este asociat unui pacient consultat. Această logica cuprinde două tabele intermediare, analize și diagnostic. Am optat pentru această logică pentru a elimina informațiile duplicate care ar fi putut apărea și pentru a diminua consumul de spațiu pentru informațiile duplicate.

Raport_Medical (nume, prenume, medic, dataconsultării, simptome, investigații). Această tabelă oferă informații referitoare la rapoartele medicale completate de către medic, în momentul în care un pacient este consultat.

Preturi_Extra (cod_specializare, specializare, preț_extra) Această tabelă este o tabelă care conține doar informații referitoare la prețurile extra pe care un medic le poate stabili în funcție de specializarea pe care o are. Astfel, în momentul în care o consultație durează mai mult de 2h (timp stabilit arbitrar), medicul poate încasa sume care se vor adăuga la salariul acestuia (pentru fiecare oră în plus, acesta poate încasa, spre exemplu, +40 lei).

3.2 Diagrama EER/UML pentru modelul de date complet



3.3 Proceduri, triggere și view-uri

Proceduri :

Init_Orar (String orar, int zi) - setează orarul pe o zi pentru un anumit angajat

Set_Concediu (String Data_Inceput, String Data_Sfârșit, String CNP) - setează concediul pentru persoana identificată prin CNP

Profit_Clinica () - calculează profitul clinicii ca fiind diferența dintre încasările clinicii și cheltuieli

Triggere :

Update_UnitateMedicală ()

Update_Diagnostic ()

Update_Analize ()

View-uri :

View-urile sunt structurate astfel încât să acopere toate cele 3 Module și să implementeze toate funcționalitățile necesare. Fiecare vedere stă la baza logicii din spatele interacțiunii Utilizator – Interfață – BD.

View_Angajati (nume, prenume, functie)

View_Orar (nume, prenume, functie , orar)

View_Concediu (nume , prenume, data_inceput, data_final)

View_Salarii (luna, nume, prenume, salariu, commission)

View_Programari (nume , data_programarii, ora)

View_IstoricMedical (nume , prenume, simptome, analize, rezultate, medicamentatie, observatii)

View_RaportMedical (nume_pacient, prenume_pacient,nume_medic, prenume_medic, descriere_simptome, diagnostic, observatii, denumire_analize, rezultate)

ViewBonFiscal(CNP, denumire_analize

3.4 Normalizarea datelor

Normalizare bazelor de date este un proces de optimizare a bazei de date prin care se încearcă minimizarea redundanței datelor, și a anomaliilor de introducere, actualizare și ștergere. În cadrul acestui proiect normalizarea nu este o problemă din pricina proiectului se afla în Boyce-Codd. Nici un tabel din baza de date nu are coloane care să depindă una de alta și ca urmare nu există legături tranzitive sau parțiale.

3.5 CodMysql

3.6 Cod pentru crearea bazei de date și a tabelor

```
create database if not exists LANT_POLICLINICI;
use LANT_POLICLINICI;
```

```
/* Tabele */
```

```
create table program
```

```
(ID int primary key,luni varchar(11),marti varchar(11),miecuri varchar(11),joi
varchar(11),vineri varchar(11),sambata varchar(11),duminica varchar(11));
```

```
create table unitate_medicala
```

```
(ID int primary key,adresa varchar(50),denumire varchar(50),descriere_servicii varchar(100),
CONSTRAINT FK_ID_unitate_medicala FOREIGN KEY (ID) references program(ID));
```

```
create table cabinet_medical
```

```
(ID int primary key,descriere varchar(30),ID_medic int);
```

```
create table program_functionar
```

```
(ID int primary key,luni varchar(11),marti varchar(11),miecuri varchar(11),joi
varchar(11),vineri varchar(11),sambata varchar(11),duminica varchar(11), locatie varchar(20),
ID_Concediu int);
```

```
create table program_cabinet
```

```
(ID int primary key,luni varchar(11),marti varchar(11),miecuri varchar(11),joi
varchar(11),vineri varchar(11),sambata varchar(11),duminica varchar(11));
```

```
create table diagnostic
```

```
(ID_parafa varchar(10) primary key,descriere_simptome varchar(50),diagnostic
varchar(20),medicamentatie varchar(20),observatii varchar(30));
```

```
create table useri_clinica
```

```
(tip_departament varchar(20),CNP int primary key,nume varchar(20),prenume
varchar(20),adresa varchar(20),nr_telefon varchar(10),email varchar(30),cont_IBAN
int,nr_contract varchar(20),data_angajarii varchar(30),
functia varchar(20));
```

create table functie

(CNP int primary key, inspector_resurse_umane boolean, expert_financiar_contabil boolean, receptioner boolean, asistent_medical boolean, medic boolean, cod_specializare integer);

create table asistent

(nume_asistent varchar(20), tipul varchar(20), gradul varchar(20));

create table medic

(nume_medic varchar(20), prenume_medic varchar(20), specialitatea varchar(30), gradul varchar(20), codul_de_parafa varchar(10), titlul stiintific varchar(20), postul_didactic varchar(20));

create table receptioner

(ID int primary key, numele varchar(20));

create table programare

(ID int primary key, nume_pacient varchar(30), prenume_pacient varchar (20), CNP BIGINT, varsta int, ID_cabinet int, data_programarii varchar(10), ora varchar(30));

create table analize

(ID_parafa varchar(30), ID_programare int, denumire_analize varchar(20), pret int, durata_controlului int, valoare_normala varchar(10), rezultate varchar(50), primary key(ID_parafa, ID_programare));

drop table concediu;

create table concediu

(ID int primary key, data_inceput varchar(11), data_sfarsit varchar (11));

create table salariu

(salariu int, comision int, CNP int primary key);

create table raport_medical

```
(nume varchar(30),prenume varchar(30),medic varchar(30) primary key,medic_asistent
varchar(30),dataconsultarii varchar(30),simptome varchar(50),investigatii varchar(50));
```

```
create table preturi_extra
```

```
(cod_specializare int, specializare varchar(30), pret_extra int);
```

```
/* Modificari ale tabelelor - constraints pentru foreign keys */
```

```
alter table useri_clinica add constraint foreign key(CNP) references functie(CNP);
```

```
alter table asistent add column cod_specializare integer primary key;
```

```
alter table functie add constraint foreign key(cod_specializare) references
asistent(cod_specializare);
```

```
alter table medic add column cod_specializare integer primary key;
```

```
alter table functie add constraint foreign key(cod_specializare) references
medic(cod_specializare);
```

```
alter table medic add column ID_cabinet integer;
```

```
alter table unitate_medicala add constraint foreign key (ID) references cabinet_medical(ID);
```

```
alter table functie add constraint foreign key (CNP) references cabinet_medical(ID);
```

```
alter table functie add constraint foreign key(CNP) references program_functionar(ID);
```

```
alter table medic add constraint foreign key (codul_de_parafa) references
diagnostic(ID_parafa);
```

```
alter table cabinet_medical add constraint foreign key(ID) references program_cabinet(ID);
```

```
alter table unitate_medicala add constraint foreign key(ID) references receptioner(ID);
```

```
alter table unitate_medicala add constraint foreign key(ID) references programare(ID);
```

```
alter table analize add constraint foreign key (ID_programare) references programare(ID);
```

```
alter table diagnostic add constraint foreign key (ID_parafa) references analize(ID_parafa);
```

```
alter table program_functionar add constraint foreign key (ID_concediu) references
concediu(ID);
```

```
alter table salariu add constraint foreign key (CNP) references functie(CNP);
```

```
alter table raport_medical add constraint foreign key (medic) references
medic(codul_de_parafa);
```

```
alter table diagnostic add column ora_final varchar(30);
```

```
alter table preturi_extra add constraint foreign key (specializare) references
medic(codul_de_parafa);
```

```

alter table raport_medical drop foreign key raport_medical_ibfk_1;
alter table salariu add column luna varchar(20);
alter table program_functionar rename column miecuri to miercuri;
alter table program_cabinet rename column miecuri to miercuri;

```

3.8 Cod Proceduri

```
/*Proceduri*/
```

```

use lant_policlinici;
DROP PROCEDURE IF EXISTS CREARE_JOC;
DELIMITER //

```

```
CREATE PROCEDURE
```

```
Init_Orar(IN Orar VARCHAR(40), IN CNP INT, IN zi INT)
```

```
BEGIN
```

```
IF(Funcie.cnp=cnp)then
```

```
Case zi%7
```

```
When 1 Then Update program_functionar SET luni=orar;
```

```
When 2 Then Update program_functionar SET marti=orar;
```

```
When 3 Then Update program_functionar SET miercuri=orar;
```

```
when 4 then Update program_functionar SET joi=orar;
```

```
when 5 then Update program_functionar SET vineri=orar;
```

```
when 6 then Update program_functionar SET sambata=orar;
```

```
else Update program_functionar SET duminica=orar;
```

```
END case;
```

```
ENd if;
```

```
end;//
```

```
DELIMITER;
```

```
use LANT_POLICLINICI;
```

```
DROP PROCEDURE IF EXISTS SET_CONCEDIU;
```

```
DELIMITER //
```

```

Create procedure Set_concediu(In DataConcediuI varchar(40),IN DataConcediuO
varchar(40),In CNP INT)
  BEGIN
  IF(Funcție.CNP=cnp) then
    update concediu set data_inceput=DacaConcediuI;
    update concediu set data_sfarsit=DataConcediuU;
  End IF;
  END;
use LANT_POLICLINICI;
DROP PROCEDURE IF EXISTS SET_CONCEDIU;
DELIMITER //
Create procedure Profit_clinica()
  BEGIN
  Select Difference(SUM(pret),SUM(funcție.salariu)) from analize join salariu;
  END;

```

3.9 Cod Useri

```

/*Users BD*/
/* Partea de Securitate (Utilizatori, Acces) incepe aici. */

use lant_policlinici;
create user 'Administrator'@localhost;
grant alter on lant_policlinici.useri_clinica to 'Administrator'@localhost;
grant update on lant_policlinici.useri_clinica to 'Administrator'@localhost;
grant select on lant_policlinici.useri_clinica to 'Administrator'@localhost;
grant delete on lant_policlinici.useri_clinica to 'Administrator'@localhost;
grant insert on lant_policlinici.useri_clinica to 'Administrator'@localhost;

create user 'Super-Administrator'@localhost;
update mysql.user set Super_Priv='Y' where user='Super-Administrator' and host='localhost';
grant all privileges on *.* to 'Super-Administrator'@localhost;
show grants for 'Super-Administrator'@localhost;

```



```
grant super on *.* to 'Super-Administrator'@localhost;
```

```
create user 'Angajat'@localhost;
```

```
create user 'Asistent-Medical'@localhost
```

```
create user 'Medic'@localhost;
```

3.10 Cod View-uri

```
/* Module pentru gestiunea policlinicii */
```

```
/*Resurse umane*/
```

```
drop view Modul_HR_Angajati;
```

```
create view Modul_HR_Angajati as
```

```
select nume, prenume, functia from useri_clinica us
```

```
left join functie f on us.CNP=f.CNP;
```

```
select * from Modul_HR_Angajati;
```

```
/*Orar*/
```

```
drop view Modul_HR_Orar;
```

```
create view Modul_HR_Orar as
```

```
select nume, prenume, functia, luni, marti, miercuri, joi, vineri, sambata, duminica from  
useri_clinica us
```

```
left join functie f on us.cnp=f.cnp
```

```
left join program_functionar pr on pr.id=f.cnp;
```

```
select * from Modul_HR_Orar;
```

```
/*Concedii*/
```

```
drop view Modul_HR_Concedii;
```

```
create view Modul_HR_Concedii
```

```

as select nume, prenume, functia, data_inceput, data_sfarsit from useri_clinica us
left join functie f on us.CNP=f.CNP
left join program_functionar pr on pr.id=f.cnp
left join Concediu c on c.ID=pr.ID_Concediu;

```

```

select * from Modul_HR_Concedii;

```

```

/*Operatii financiar contabile*/

```

```

/*Salarii*/

```

```

drop view Modul_FinanciarContabil_Salarii;
create view Modul_FinanciarContabil_Salarii
as select luna, nume, prenume, functia, salariu, comision
from useri_clinica us
left join functie f on us.cnp=f.cnp
left join salariu s on s.cnp=f.cnp;

```

```

/*Gestiunea activitatilor medicale*/

```

```

/*Programari*/

```

```

create view Modul_GestionareActivitati_Programari
as select id, nume_pacient, data_programarii, ora
from programare prog
join analize an on prog.id=an.id_programare;

```

```

select * from Modul_GestionareActivitati_Programari;

```

```

/*Istoric Medical*/

```

```

create view Modul_GestionareActivitati_IstoricMedical
as select ID, nume_pacient, descriere_simptome, diagnostic, denumire_analize, rezultate,
medicamentatie, observatii
from programare prog
join analize an on prog.id=an.id_programare
join diagnostic di on an.id_parafa=di.id_parafa;

```

```
select * from Modul_GestionareActivitati_IstoricMedical;
```

```
/*Raport Medical*/
```

```
create view Modul_GestionareActivitati_RaportMedical
as select nume_pacient, prenume_pacient,nume_medic, prenume_medic, descriere_simptome,
diagnostic, observatii, denumire_analize, rezultate
from programare prog
join analize an on prog.id=an.id_programare
join diagnostic di on an.id_parafa=di.id_parafa
join medic m on di.id_parafa=m.codul_de_parafa;
```

```
/* Bon fiscal */
```

```
create view BonFiscal
as select CNP , denumire_analize , pret , data_programarii
from programare p
join analize an on p.id=an.id_programare;
```

```
select * from BonFiscal;
```

3.11 Cod Trigger

```
/* Triggere */
```

```
delimiter $$
```

```
create trigger update_unitate_medicala after UPDATE on unitate_medicala
for each row begin
insert into unitate_medicala
set ID=programare.ID,
adresa = OLD.adresa,
denumire=OLD.denumire,
descriere_servicii=OLD.descriere_servicii;
END $$
delimiter;
```

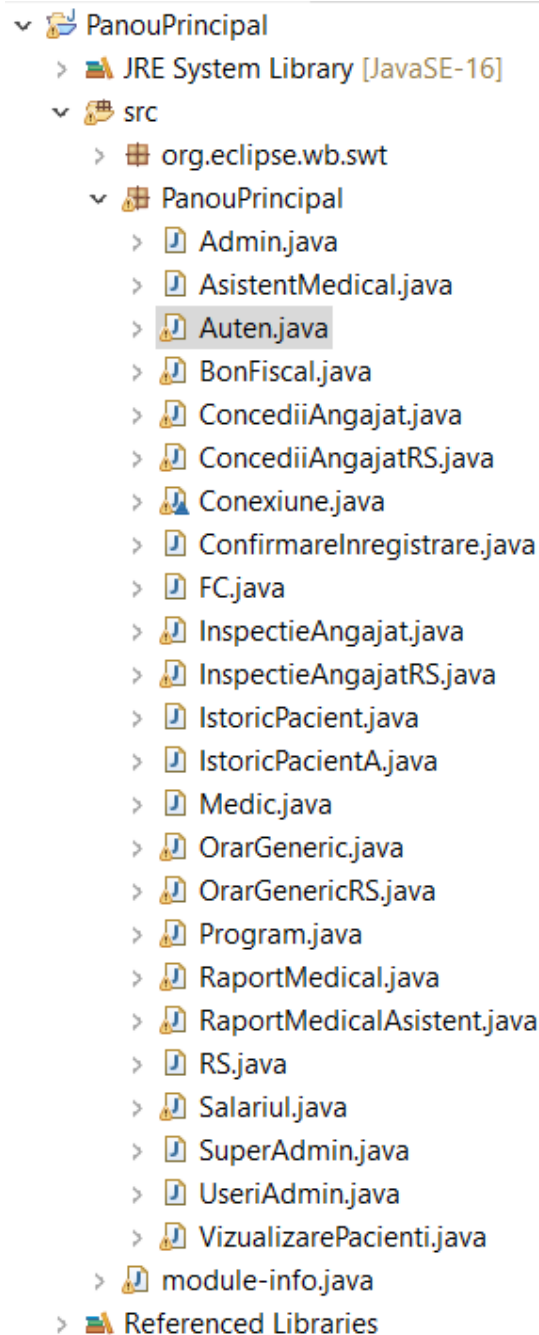
```
delimiter //  
create trigger update_diagnostic after update on diagnostic  
for each row begin  
insert into diagnostic  
set ID_parafa = medic.codul_de_parafa,  
descriere_simptome = OLD.descriere_simptome,  
diagnostic=OLD.descriere_simptome,  
medicamentatie=OLD.medicamentatie,  
observatii=OLD.observatii;  
END //
```

```
delimiter;
```

```
delimiter ??  
create trigger update_analize after update on analize  
for each row begin  
insert into analize  
set ID_parafa=diagnostic.ID_parafa,ID_programare = programare.ID,  
denumire_analize=OLD.denumire_analize,  
pret=OLD.pret,  
durata_controlului=OLD.durata_controlului,  
valoare_normala=OLD.valoare_normala,  
rezultate=OLD.rezultate;  
END??  
delimiter ??
```

4 Detalii de implementare

4.11 Structura claselor în Java



Fiecărei clase îi corespunde o fereastră din Interfața Grafică.

4.12 Interfața grafică

Secțiunea de autentificare :



PANOU COMANDA

BINE ATI VENIT LA POLICLINICA

TREI FRATI PATATI

AUTENTIFICARE

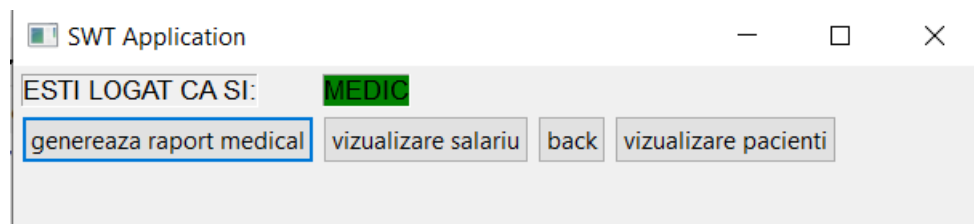
SELECTATI TIPUL DE UTILIZATOR

MEDIC ASISTENT MEDICAL

RESURSE UMANE

FINANCIAR-CONTABIL ADMINISTRATOR SUPERADMIN

Secțiunea Medicilor :

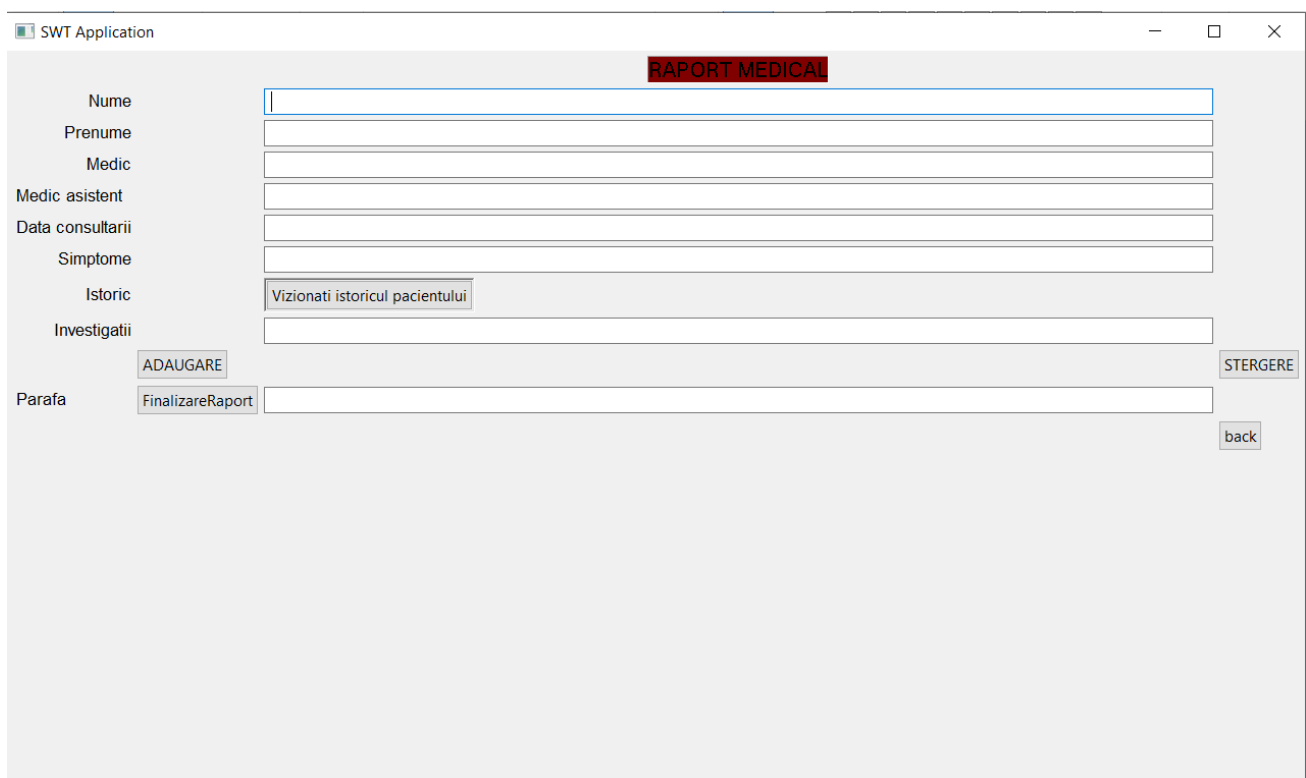


SWT Application

ESTI LOGAT CA SI: MEDIC

genereaza raport medical vizualizare salariu back vizualizare pacienti

Rapoarte :



SWT Application

RAPORT MEDICAL

Nume

Prenume

Medic

Medic asistent

Data consultarii

Simptome

Istoric

Investigatii

ADAUGARE

STERGERE

Parafa

FinalizareRaport

back

Istoric :

SWT Application

—

□

×

Nume

Prenume

Istoric

afiseaza istoric

back

Salarii :

SWT Application

—

□

×

SECTIUNEA SALARIU

Nume

Prenume

Luna

Profit

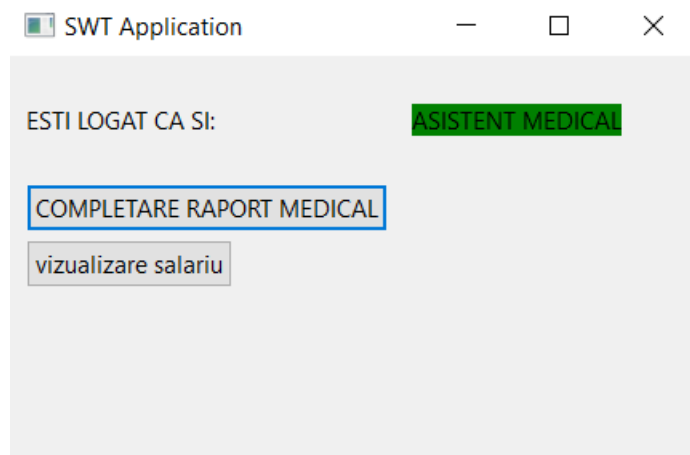
Comision

Salariu

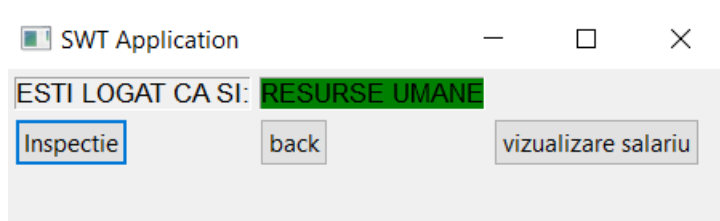
back

afiseaza salariu

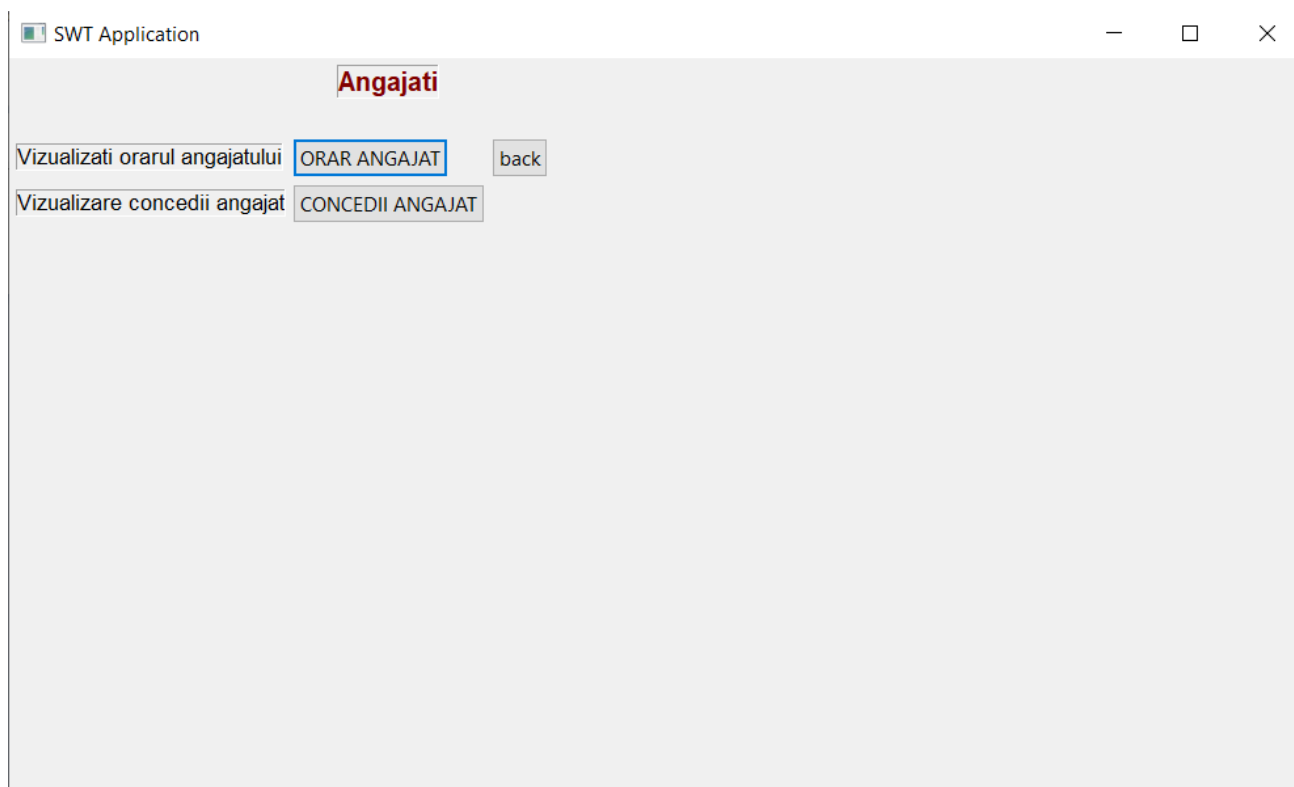
Asistent Medical :



Modul Resurse Umane :



Inspecție :



Orar :

SWT Application

ORARUL ANGAJATULUI

Nume

Prenume

Functie

Locatia

Afiseaza

Luni

Marti

Miercuri

Joi

Vineri

Sambata

Duminica

back

Concediu

SWT Application

CONCEDII ANGAJATULUI

Nume

Prenume

Functia

Data inceput

Data final

back

Afiseaza

Financiar – Contabil

SWT Application

ESTI LOGAT CA SI: FINANCIAR CONTABIL

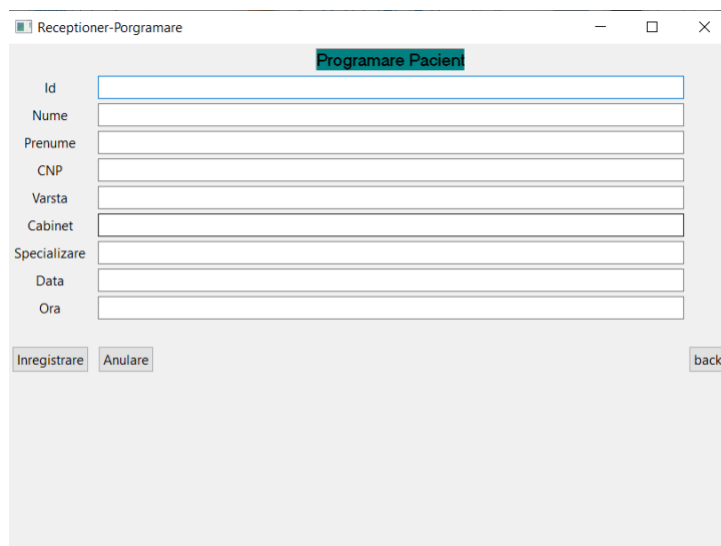
Inspectie

Efectueaza programare

back

vizualizare salariu

Programare

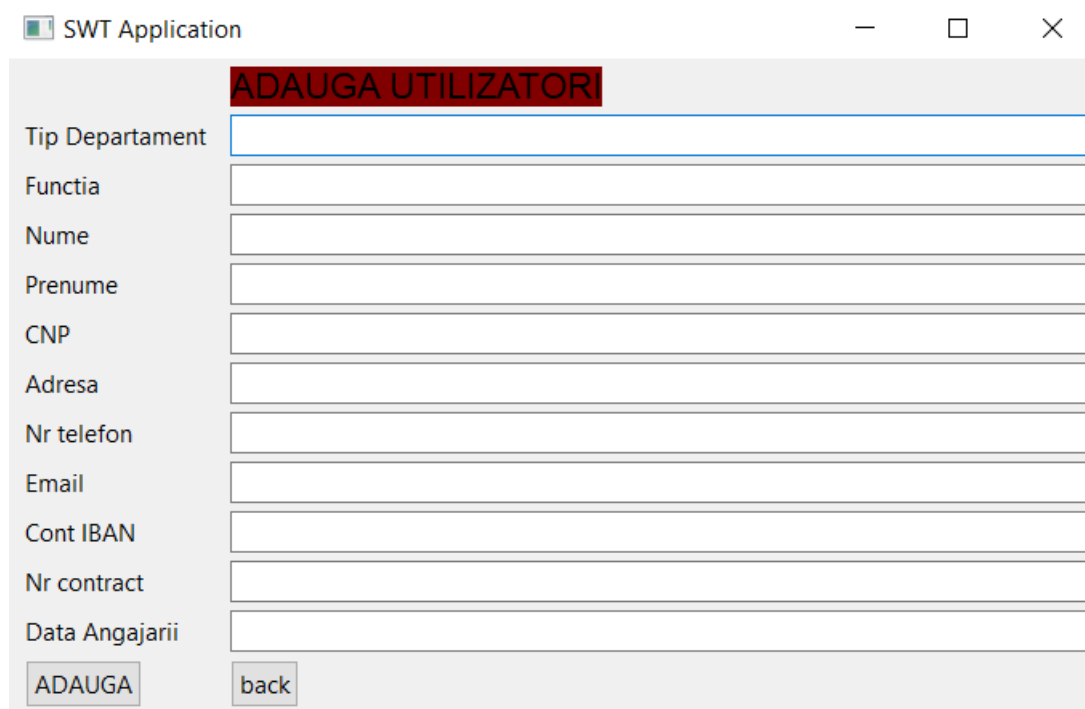


The screenshot shows a window titled "Receptioner-Programare" with a sub-header "Programare Pacient". The form contains the following fields:

Field	Input Type
Id	Text
Nume	Text
Prenume	Text
CNP	Text
Varsta	Text
Cabinet	Text
Specializare	Text
Data	Text
Ora	Text

At the bottom, there are three buttons: "Inregistrare", "Anulare", and "back".

Administrator și Super – Administrator



The screenshot shows a window titled "SWT Application" with a sub-header "ADAUGA UTILIZATORI". The form contains the following fields:

Field	Input Type
Tip Departament	Text
Functia	Text
Nume	Text
Prenume	Text
CNP	Text
Adresa	Text
Nr telefon	Text
Email	Text
Cont IBAN	Text
Nr contract	Text
Data Angajarii	Text

At the bottom, there are two buttons: "ADAUGA" and "back".

4.13 Manual de utilizare/Instalare

// INFO pentru intelegerea modului de functionare
 // Acesta este un file README care contine informatii referitoare la instalarea, folosirea si functionalitatile pe care le regasiti in aceasta arhiva. Arhiva contine codul MYSQL pentru crearea unui sistem de gestiune de date pentru un lant de policlinici. Contextul este urmatorul > O unitate medicala are angajati (, repartizati pe mai multe module in functie de rolurile pe care acestia le au), cabinete, pacienti si servicii . Introducerea datelor pentru fiecare data in parte se face din interfata GUI implementata. Fiecare fisier are un nume reprezentativ pentru datele continute

////////////////////////////////////

Pentru MYSQL > se ruleaza in urmatoarea ordine

> ProiectBDMain (baza de date efectiva)

>ProceduriBD.sql

>Triggere.sql

>Vederi.sql

>UserBD.sql

>PopulareBD.sql

////////////////////////////////////

Pentru Java > Se deschide fisierul PanouPrincipal

> In acest fisier veti gasi toate clasele aferente funcrtionalitatilor

>Veti gasi clasa java pentru ConexiuneaDB pe care o rulate

> Rulate apoi restul proiectului incepand cu Auten.java si explorati fiecare buton si fiecare fereasta care se deschide

////////////////////////////////////

!!Atentie > Pentru a verifica corectitudinea , in ProiectBDMain.sql exista interogari de tipul "Select * from (vedere/ abela) care pot fi rulate

!! Atentie , pentru a folosi aplicatia Java aveti nevoie de Eclipse IDE si extensia WindowBuilder (Consultati StackOverflow , pagina Eclipse >Sectiunea WindowBuilder pentru detalii)

////////////////////////////////////

Pentru probleme intalnite, acestea se pot raporta la petanarar@yahoo.com

Pentru probleme de securitate intalnite, va rugam sa nu faceti publice datele afisate!