MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

PROIECT

la disciplina

Introducere in Baze de Date

Policlinica

An academic: 2020 – 2021

PROIECT de SEMESTRU
Catedra de Calculatoare
Disciplina : Introducere in Baze de Date
Realizatori:
Blaj Sergiu-Emanuel
Borbei Raul-Aurelian
Mihalache Rareș
Preda Cristian
Coordonator: s.l. ing. Cosmina IVAN
Data 15.01.2021

CUPRINS

1.Introducere

- Introducere, argumente, scop si obiective specifice

Info Med este o aplicație cu interfața grafica care interacționează cu un sistem de gestiune a informațiilor si activităților medicale desfășurate in cadrul unei *Policlinici*, oferind funcționalități utilizatorilor in funcție de drepturile pe care aceștia le dețin in sistemul informatic.

Principalele funcționalități ale sistemului Info Med sunt:

- Stabilirea programului pentru fiecare unitate medicala, respectiv intervalul de lucru al angajaților;
- Reținerea competentelor si specialităților pentru fiecare medic, respectiv gradul si tipul pentru asistenți;
- Activitatea medicilor – durata consultației, pacienții investigați si tranzacțiile efectuate;
- Evidenta pacienților, consultațiilor si tratamentelor aplicate acestora (fisa medicala);
- Urmărirea evoluției tratamentelor si analizelor medicale (istoricul pacienților);
- Calculul costurilor serviciilor medicale, salariile angajaților, cheltuielile, veniturile si profitul policlinicii.

Info Med se armonizează perfect cu activitatea unităților medicale de tratament si contribuie simțitor la **înlăturarea erorilor ce pot apărea in urma intervențiilor medicale**. Aplicația realizează **urmărirea fluxului de pacienți** in clinica si **stocarea informațiilor medicale** rezultate din consultații, aspecte care îmbunătățesc considerabil **calitatea actului medical**.

Sistemul asigura o **gestiune excelenta** a relațiilor cu clienții prin **evidenta listei pacienților si programărilor**. De asemenea, sistemul calculează pe baza preturilor introduse de utilizator **sumele de plata ale pacienților** si **emite un bon fiscal**. Astfel, circulația corecta si fluida a informațiilor dinspre unitatea medicala

spre pacient si viceversa **ajuta la eficientizarea activității medicale**, contribuind totodată la **fidelizară continua a clienților**. Pe baza datelor introduse se pot obține anumite rapoarte cum ar fi: programări, orarul medicilor.

Info Med ii oferă fiecărui medic posibilitatea de a-si vizualiza programările si de a introduce informațiile medicale rezultate din consultai in fisa pacientului. Astfel, se acorda o mai mare importanta actului medical, lucru ce contribuie considerabil la **satisfacerea pacienților**.

Aplicația permite colectarea datelor din întreg sistemul pentru a le evidenția in rapoarte bine definite ce pot fi accesate in orice moment pentru **decizii manageriale rapide, corecte si eficiente**.

2.Analiza cerințelor utilizatorilor și specificațiile de proiect

- Ipoteze specifice domeniului ales pentru proiect (cerințe, constrângeri)

Aplicația gestionează acțiunile mai multor tipuri de actori dintr-un lanț de policlinici. Aceasta utilizează o baza de date, care a fost gândita conform următoarelor cerințe:

- Exista doua tipuri de utilizator, administrator si super-administrator.
- Un angajat este unic identificat prin ‘Id’-ul sau. Acesta mai are un ‘CNP’, ‘nume’, ‘prenume’, ‘salariu’, ‘nr_minim_ore’, ‘data_angajarii’ si o ‘functie’.
- Fiecare angajat se poate loga in aplicație cu ajutorul unui e-mail si a unei parole. Câmpul ‘email’ din cadrul tabelei ‘utilizator’ este cheie primara.
- Exista mai multe unități, identificate prin ‘id_unitate’ si fiecare dintre acestea oferă anumite servicii. Tabela ‘unități’ comunica cu ‘servicii’ si fiecare serviciu are asociat un ‘Id’, un ‘preț’, o ‘durata’ si anumite câmpuri pentru a stabili daca un anumit medic angajat la acea policlinica poate presta serviciul ales de client.
- Fiecare medic are o anumita specialitate si poate avea anumite competente necesare pentru prestarea anumitor servicii.
- Un medic poate avea mai multe programări si fiecare pacient programat are un istoric asociat, unde sunt specificate anumite câmpuri relevante pentru a ușura munca medicului, cum ar fi: ‘diagnostic’, ‘recomandări’ , ‘simptome’ si un scurt text care detaliază problema pacientului.

- Organizare structurata(tabelar) a cerințelor utilizator

Baza de date trebuie sa stocheze următoarele informații:

- Unitățile lanțului de policlinici
- Serviciile oferite
- Utilizatorii existenți in sistem
- Angajații care pot face anumite operații
- Programul de lucru al unităților
- Programul de lucru al angajaților
- tabela ‘medici’ si o tabela ‘asistenți’.
- Competentele unui medic pentru prestarea anumitor servicii medicale.
- tabela ‘programări’, asociata fiecărui medic.
- Istoric pacienți pentru o prezentare detaliata a situației pacienților. Fiecare programare are unul sau mai multe istorice asociate.

Mai mult, trebuie sa permită si o gama larga de operații in funcție de angajatul care este logat. Astfel, avem mai multe tipuri de actori: inspector resurse umane, expert financiar-contabil, recepționer, asistent medical si medic.

Modulul pentru gestiunea resurselor umane are următoarele operații:

- Un inspector resurse umane poate caută un angajat de orice tip. Acesta ii poate crea un anumit orar de lucru, caracterizat la ziua la care se refera, interval orar si locație.
- Un inspector resurse umane poate programa o perioada de concediu pentru angajați.

Modulul pentru operații financiar-contabile are următoarele operații:

- Un expert poate vizualiza informații cu privire la profitul realizat de lanțul de policlinici, pentru lunile precedente in care s-au înregistrat activități.

- De asemenea, un expert poate efectua rapoarte cu privire la profitul realizat de fiecare medic in parte, pe fiecare locație (unitate medicala) sau pe fiecare specialitate.

Modulul pentru gestiunea activităților operaționale are următoarele operații:

- Un recepționer poate realiza o programare pentru un pacient.
- Un recepționer poate emite un bon fiscal ulterior consultației, cuprinzând fiecare serviciu medical ce a fost efectuat.
- Un asistent medical poate completa informații in rapoartele pentru analizele medicale corespunzătoare pacienților care au fost înregistrați pentru aceasta.
- Un medic poate vizualiza pacienții programați la el într-o anumita zi care urmează.
- Pentru fiecare pacient consultat medicul completează un raport medical.
- Medicul poate gestiona lista serviciilor medicale (adăugare, ștergere) in funcție de necesitatea / inoportunitatea realizării anumitor proceduri.
- In momentul in care medicul a terminat raportul, acesta este parafat, nemaiputând fi modificat si se poate vizualiza in cadrul istoricului pacientului.

Aplicația Java trebuie sa permită prelucrarea informațiilor din baza de date, căutarea si afișarea informațiilor si conectarea utilizatorilor înregistrați.

- Determinarea si caracterizarea de profiluri de utilizatori (admin, user intern, user extern…diverși alți “actori”)

Avem doua tipuri de utilizator:

1) Administratori, care au următoarele posibilități:
   - Pot adaugă, modifica si șterge informații in baza de date legate de utilizatori.
2) Super-administratori, care au următoarele posibilități:
   - Fac același lucru ca si administratorii.
   - Pot opera si pe informații legate de administratori, si pot decide daca scot rolul de ‘administrator’ al unui utilizator sau nu.

3.Modelul de date si descrierea acestuia

- Entități si atributele lor (descriere detaliata – implementarea fizica)

O bază de date reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

SQL (Structured Query Language - Limbaj Structurat de Interogare), dezvoltat de catre D.D.Chamberlin si R.F.Boyce, este un limbaj de programare specific pentru manipularea datelor în sistemele de manipulare a bazelor de date relaționale (RDBMS), iar la origine este un limbaj bazat pe algebra relațională. Acesta are ca scop inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date.

SQL este împărțit in mai multe elemente, fiecare având un rol unic in scrierea codului unei baze de date. Aceste elemente sunt : clauze, care sunt componente ale instrucțiunilor și interogărilor; expresii, al căror efect este producerea de valori scalare sau tabele; predicatele, pot specifica condiții care sunt evaluate de SQL conform logicii ternare sau logicii booleene, în scopul limitării efectelor instrucțiunilor, sau pentru a influența cursul programului; interogările, au ca scop regăsirea datelor după criterii specifice; instrucțiunile, pot avea un efect persistent asupra datelor sau structurii datelor, sau pot controla tranzacțiile, conexiunile sau cursul programului.

Operațiile de bază pe o bază de date sunt:

- crearea unui tabel

```
CREATE TABEL tbl_name ([data_type] [col_name], ...)
```

- afișarea datelor unui table

```
SELECT select_expr [, select_expr ...]
[FROM table_references [WHERE where_condition]
```

- inserarea în baza de date

```
INSERT [INTO] tbl_name [(col_name,...)] VALUE ({expr},...),(...),...)
```

- modificarea datelor

```
UPDATE table_reference
SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}]
...
[WHERE where_condition]
```

- ștergerea datelor

```
DELETE FROM tbl_name
[WHERE where_condition]
```

În tabela Unități sunt stocate unitățile medicale din policlinica, fiecare fiind identificata unic prin id.

```
create table unitati (
   id_unitate int(20) auto_increment primary key,
   denumire varchar(50),
   oras varchar(50)
);
```

Tabela servicii stochează serviciile oferite de către lanțul de policlinici, de asemenea o descriere a serviciului și prețul acestuia, împreuna cu durata de timp necesara pentru efectuarea sa. Tabela are ca și cheie primara id_serviciu, astfel asigurând-ne ca un serviciu este unic.

```
create table servicii (
    id_serviciu varchar(20) primary key,
    nume varchar(50),
    descriere varchar(500),
    specialitate varchar(50),
    competenta_necesara varchar(50),
    pret float,
    durata int
);
```

Intre tabela servicii și unități avem tabela servicii oferite care ne spune ce servicii sunt efectuate la ce unitate din policlinica.

```
create table servicii_oferite (
    id_unitate int,
    id_serviciu varchar(20),
    foreign key (id_unitate) references unitati (id_unitate),
    foreign key (id_serviciu) references servicii (id_serviciu)
);
```

În tabela program unități reținem programul de lucru al fiecărei unități în parte.

```
create table program_unitati (
    id_unitate int(20),
    zi_calendaristica enum('Luni', 'Marti', 'Miercuri', 'Joi', 'Vineri', 'Sambata', 'Duminica'),
    ora_deschidere int,
    ora_inchidere int,
    foreign key (id_unitate) references unitati (id_unitate)
);
```

Tabela angajați retine datele despre un angajat precum sunt specificate în cerința. În plus în câmpul funcție reținem funcția angajatului respectiv.

```
create table angajati (
        id      int auto_increment primary key,
    cnp varchar(20) unique,
    nume varchar(50),
    prenume varchar(50),
    salariu int,
    nr_minim_ore int,
    nr_contract varchar(20),
    data_angajarii date,
    functie enum('inspector resurse umane', 'expert financiar-contabil', 'receptioner', 'asistent medical', 'medic')
);
```

În cazul în care în coloana funcție din angajat avem asistent sau medic, pentru aceștia avem nevoie sa reținem câteva date suplimentare, de aceea mai avem doua tabele asistenți și medici.

```
create table asistenți (
    cnp varchar(20) primary key,
    grad enum('principal', 'secundar'),
    tip enum('generalist', 'laborator', 'radiologie'),
    foreign key (cnp) references angajati (cnp)
);


create table medici (
    cnp varchar(20),
    parafa varchar(50),
```

```
    titlu_stiintific enum('doctorand', 'doctor în stiinte medicale'),
    post_didactic enum('preparator', 'asistent', 'sef de lucrari', 'conferentiar', 'profesor'),
    procent_venit float,
    primary key (cnp , parafa),
    foreign key (cnp) references angajati (cnp)
);
```

În tabela program angajați reținem programul de munca al unui angajat. În plus reținem unitatea la care lucrează în ziua respectiva și daca este o zi de concediu sau nu.

```
create table program_angajati (
    data_program date,
    id_unitate int,
    cnp varchar(20),
    ora_inceput int,
    ora_sfarsit int,
    zi_lucratoare boolean,
    zi_calendaristica enum('Luni', 'Marti', 'Miercuri', 'Joi', 'Vineri', 'Sambata', 'Duminica'),
    primary key (cnp, data_program),
    foreign key (id_unitate) references unitati (id_unitate),
    foreign key (cnp) references angajati (cnp)
);
```

Pentru un medic este nevoie sa ii reținem specialitatea sau specialitățile și competentele dobândite. Cu asta ne ajuta următoarele doua tabele:

```
create table specialitate_medici (
    id_specialitate int primary key,
    parafa varchar(50),
    specialitate varchar(20),
```

```
    grad enum('specialist', 'primar'),
    cnp varchar(20),
    foreign key (cnp , parafa) references medici (cnp , parafa)
);

create table competente (
    parafa varchar(50) primary key,
    cnp varchar(20),
    ecografie boolean,
    endoscopie_digestiva boolean,
    ecocardiografie boolean,
    cardiologie_interventionala boolean,
    bronhoscopie boolean,
    eeg boolean,
    emg boolean,
    dializa boolean,
    chirurgie_laparoscopica boolean,
    chirurgie_toracica boolean,
    chirurgie_spinala boolean,
    litotritie_extracorporeala boolean,
    explorare_computer_tomograf boolean,
    imagistica_prin_rezonanta_magnetica boolean,
    foreign key (cnp , parafa) references medici (cnp , parafa)
);
```

În tabela programări se vor retine programările pacienților cu data și ora, cu datele personale ale pacientului, cu medicul care l-a trimis la investigații și cu medicul care se va ocupa de pacient în cadrul policlinicii.

```
create table programări (
    id_programare varchar(20),
    data_programare date,
    ora_programare int,
    cnp_medic varchar(20),
    parafa varchar(50),
    cnp_pacient varchar(20),
    nume_pacient varchar(50),
    prenume_pacient varchar(50),
    nume_medic_recomandat varchar(50),
    prenume_medic_recomandat varchar(50),
    durata_consultatie int,
    primary key (id_programare),
    foreign key (cnp_medic , parafa) references medici (cnp , parafa)
);
```

Tabela investigații stochează investigațiile efectuate de către un pacient în cadrul unei programări.

```
create table investigatii (
    id_programare varchar(20),
    id_serviciu varchar(20),
    concluzie varchar(500),
    nume_asistent varchar(50),
    prenume_asistent varchar(50),
    foreign key (id_serviciu) references servicii (id_serviciu),
    foreign key (id_programare) references programări (id_programare)
);
```

Tabela istoric pacienți retine toate investigațiile efectuate de către un pacient în cadrul policlinicii noastre. De asemenea la finalizarea unui raport medical se va genera o noua înregistrare în aceasta tabela pentru pacientul respectiv.

```
create table istoric_pacienti (
    id_programare varchar(20),
    nume_asistent varchar(50),
    prenume_asistent varchar(50),
    simptome varchar(50),
    diagnostic varchar(50),
    recomandari varchar(50),
    istoric varchar(500),
    cnp_medic varchar(20),
    parafa varchar(50),
    primary key (id_programare),
    foreign key (id_programare) references programări (id_programare),
    foreign key (cnp_medic, parafa) references medici (cnp, parafa)
);
```

Tabela utilizatori retine datele necesare pentru crearea unui cont în aplicație de către un angajat al policlinicii.

```
create table utilizatori (
    email varchar(50) primary key,
    parola varchar(50),
    cnp varchar(20),
    oras varchar(50),
    telefon varchar(20),
    iban varchar(25),
```

```
        tip enum('administrator', 'super-administrator', 'angajat'),
    foreign key (cnp) references angajati(cnp)
);
```

- Normalizare

Baza de date respecta FN1 pentru ca este proiectata fără sa avem atribute compuse. Nu respecta FN2 (deci, implicit, nici cele superioare) de exemplu din cauza tabelei program_angajati (de ex: ora_inceput nu depinde de cnp).

- Interogări

#1. Căutam CNP-ul pacientului cu programare actuala în tabela programări.

```
SELECT
   @CNP_PACIENT:=CNP_PACIENT
FROM
   PROGRAMARI
WHERE
   PROGRAMARI.ID_PROGRAMARE = ID_PROGRAMARE;
```

π(@CNP_PACIENT)σ ID_PROGRAMARE = ID_PROGRAMARE (PROGRAMARI)

#2.Selectam ora de final al programului medicului cu CNP-ul trimis de noi, în data respectiva.

```
SELECT
   @ora_medic_sfarsit:=ora_sfarsit
FROM
   program_angajati
WHERE
   program_angajati.CNP = CNP_medic
      AND program_angajati.data_program = data_programare;
```

Π(@ora_medic_sfarsit)σ(CNP_medic, data_programare) = CNP, data_program (program_angajati)

#3. Selectam ora de început, ora de sfârșit si zi lucrătoare din program angajati

```
select ora_inceput, ora_sfarsit, zi_lucratoare into @a_ora_inceput, @a_ora_sfarsit, @a_zi_lucratoare
 from program_angajati
 where program_angajati.CNP = CNP_asistent
```

π (ora_inceput, ora_sfarsit, zi_lucratoare)σ CNP= CNP_asistent (program_angajati)

#4.Selectam numărul de investigații ale unei programări

```
select count(*) into @nr_inv from investigatii where investigatii.id_programare = id_programare;
```

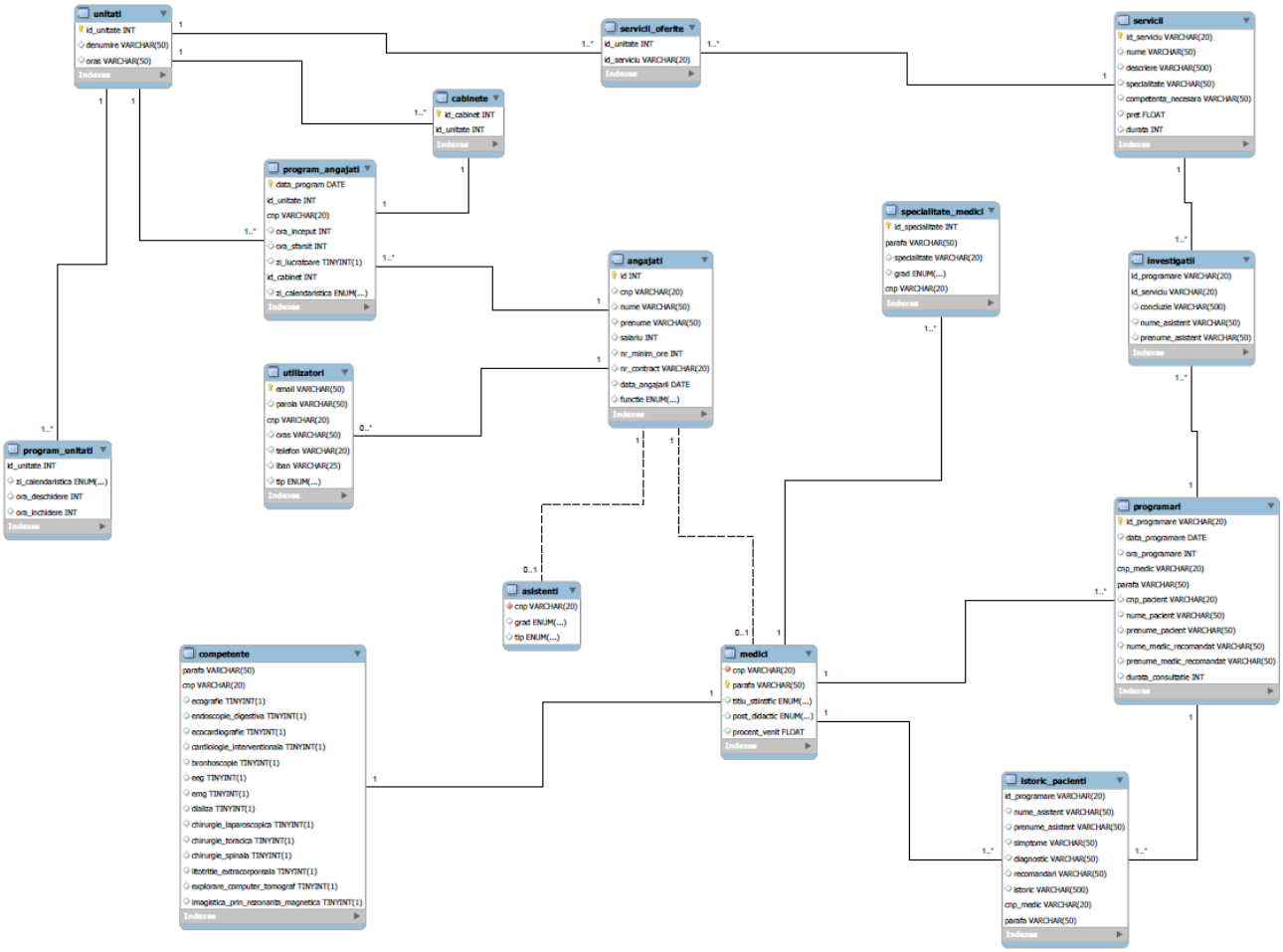
π (COUNT(*))σ id_programare = id_programare (investigatii)

#5. Selectam venitul realizat de un medic din serviciile oferite, conform programărilor si in luna data de noi.

```
SELECT
   @VENIT:=SUM(SERVICII.PRET)
FROM
   Servicii,investigatii,programari
WHERE
   investigatii.id_serviciu = servicii.id_serviciu
      AND investigatii.id_programare = (SELECT programari.id_programare
      FROM
         programari,angajati,program_angajati
      WHERE
```

```
        programari.cnp_medic = CNP_angajat
    AND (SELECT @LUNA:=EXTRACT(MONTH FROM
PROGRAM_ANGAJATI.DATA_PROGRAM)) = LUNA);
```

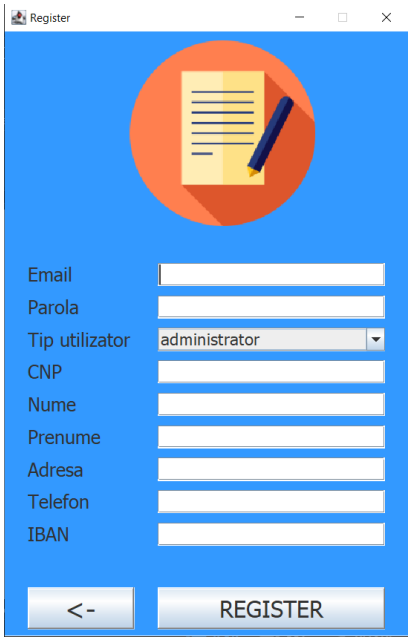
- Diagrama EER/UML pentru modelul de date complet



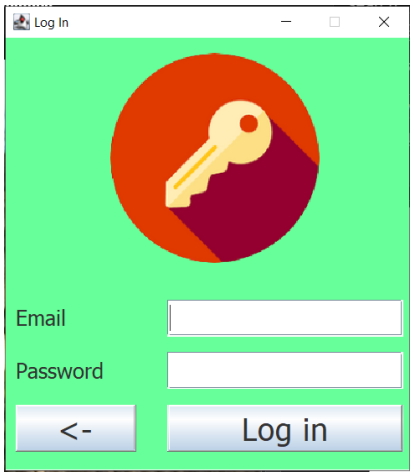
4.Detalii de implementare

- Descrierea funcțională a modulelor (organizarea logica a acestora- de ex . structura claselor Java, cod HTML, JSP, ASP, PhP)

Interfața grafică a aplicației este realizată în limbajul Java. Fiecare fereastră are propriul ei Jframe și Jpanel. Meniul principal conține o imagine reprezentativă (a unui JLabel), titlul aplicației și două butoane (JButton): unul de LogIn și unul de Register. În cazul în care un angajat nu are un cont, acesta își poate crea unul nou accesând pagina de creare a unui cont nou, prin butonul de Register.

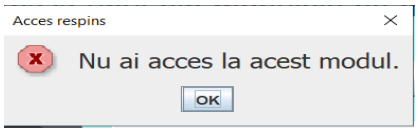


În fereastra pentru creare a unui cont nou, angajatului i se cere să specifice mai multe informații în căsuțe JTextField, precum: adresa de email, folosită la logarea în aplicație, parola pe care aceasta o va folosi când se va loga în aplicație, tipul de utilizator (acesta are de ales dintre angajat, administrator și super-administrator, dintr-un JComboBox), CNP, numele, prenumele, adresa, telefonul și IBAN-ul. În momentul în care utilizatorul apasă pe butonul Register, dacă verificările făcute (de exemplu, verificarea ca acel utilizator să nu aibă deja cont, sau numele să fie introdus corect, etc) se execută cu succes, contul este creat și utilizatorul poate accesa aplicația.

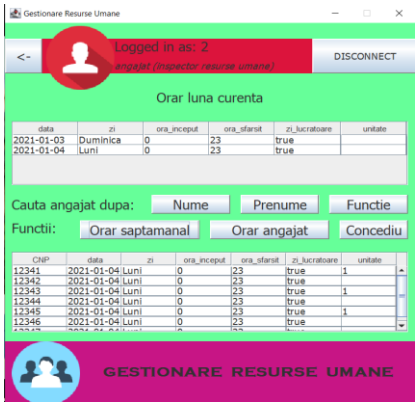


În cazul în care un angajat are cont de aplicație, acesta își poate introduce datele sale în câmpurile (JTextField-uri) din fereastra de LogIn. Dacă ambele câmpuri sunt completate, următoarea verificare care este făcută este necesitatea existenței unui cont care are ca email ceea ce a introdus utilizatorul. Dacă acesta există dar parola introdusă nu corespunde cu cea prezentă în baza de date, se afișează un mesaj de eroare (JoptionPane). În caz contrar, utilizatorul poate accesa modulele aplicației.

După o logare cu succes, utilizatorul poate alege în care modul să-și desfășoare activitatea: resurse umane, contabil sau operațional. În plus, un administrator/ super-administrator mai poate opta să-și desfășoare activitatea în panelul specific administratorilor. Antetul paginii conține informații (scurte date despre utilizatorul conectat în aplicație în momentul respectiv) și butoane, precum un buton "Înapoi" (nu este prezent în meniul principal deoarece nu se poate naviga înapoi) sau "Deconectare", care deloghează utilizatorul din aplicație, fereastra schimbându-se la cea de LogIn. Suplimentar, dacă se face click pe poza ce reprezintă userul se deschide o fereastră nouă ce conține informații mai detaliate despre utilizatorul care este logat în acel moment în aplicație, precum nume, prenume, email, parolă, telefon, adresă etc.



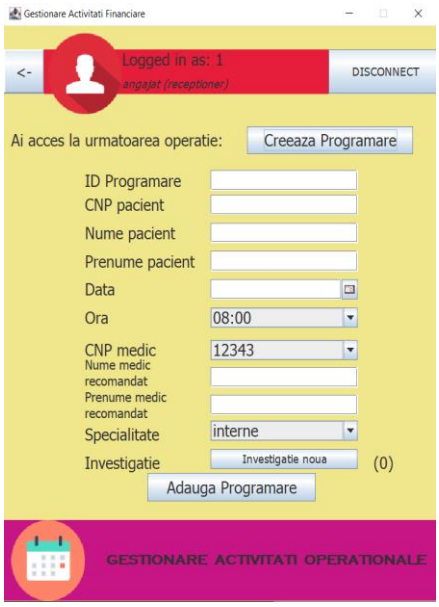
Dacă un angajat nu are permisiune pentru un anumit modul, se va afișa un mesaj de eroare printr-un JOptionPane când acesta dorește să acceseze acel modul.



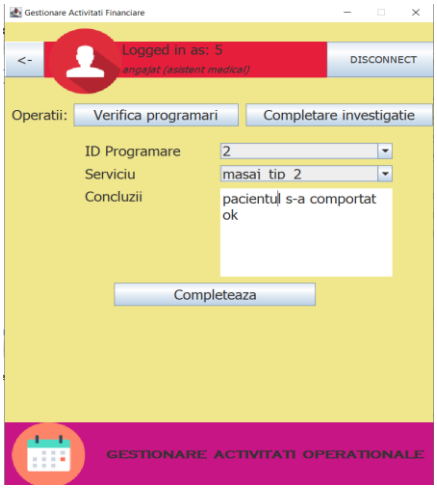
Modulul pentru Gestionarea Resurselor Umane: aici, fiecare angajat își poate vedea orarul săptămânal, ce conține data, ora de început, ora de sfârșit, etc. Dacă un utilizator are funcția de inspector resurse umane, acesta poate căuta un angajat după nume, prenume sau funcție (într-o căsuță de dialog inspectorul va introduce numele/ prenumele/ funcția angajatului pe care dorește să-l caute și vor apărea informații despre acesta), poate verifica orarul săptămânal, orarul unui angajat sau poate oferi concediu unui angajat.

Modulul pentru Operații Financiar Contabile: în această fereastră, fiecare angajat poate alege luna pentru care dorește să își vizualizeze salariul. Suplimentar, un medic poate vizualiza profitul generat de el pentru o luna aleasă, iar dacă un angajat are funcția de expert financiar contabil, acesta poate vizualiza profitul pe policlinică alegând luna dorită, profitul pe unitate, alegând unitatea și luna, profitul pe specialitate, alegând specialitatea și luna sau profitul generat de un medic, dar poate

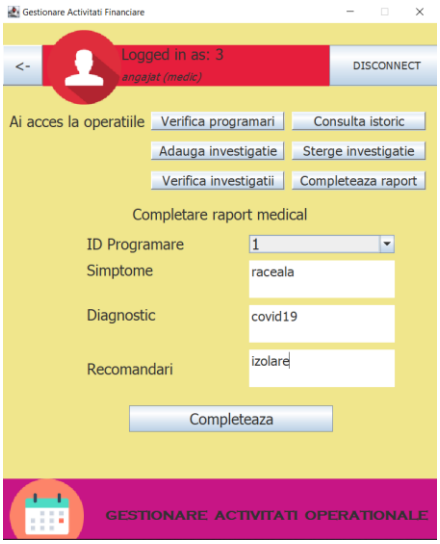
verifica și salariul fiecărui angajat, alegând CNP-ul angajatului dintr-o listă. Toate funcțiile mai sus menționate au propriul lor JPanel, pentru a facilita accesul în pagină.



Modulul pentru Gestiunea Activităților Operaționale: modul la care au acces numai recepționerii, asistenții medicali și medicii. Un recepționer poate programa în cadrul acestui modul o consultație. Acesta trebuie să specifice ID-ul ei, date despre pacient (daca un pacient are mai mult de o consultație în istoricul său, numele și prenumele vor fi completate automat în momentul în care CNP-ul său a fost completat), data programării (recepționerul alege o dată ulterioară dintr-un calendar – JDateChooser), ora programării (JComboBox), medicul care va face consultația (va alege CNP-ul unui medic dintr-un JComboBox), medicul care recomandă consultația și o listă de servicii. În momentul în care recepționerul apasă pe buton, se fac verificările necesare în funcția de adăugare a unei programări în baza de date din MySQL (medicul să lucreze la acea oră, pacientul să nu fie ocupat cu altă programare, etc); dacă acestea au loc cu succes, pacientul este programat cu succes (se face introducerea unei înregistrări în tabela programări).



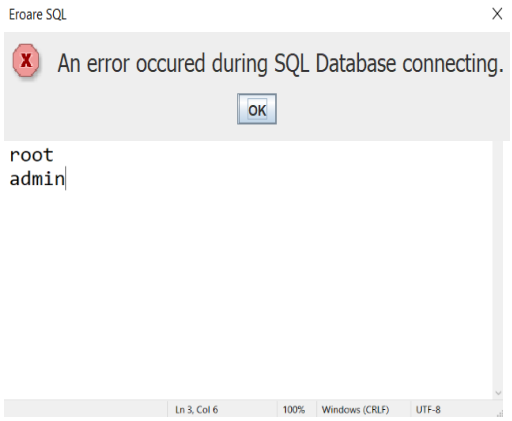
Un asistent medical, poate verifica, în cadrul acestui modul, programările din săptămâna curentă, afișate într-un tabel (JTabel) sau poate completa rezultatul unei analize medicale de la o anumită consultație (selectate automat în JComboBox-uri), dacă raportul ce conține acea analiză nu a fost parafat de medic, sau dacă nu a fost deja stabilită concluzia pentru acea investigație. Când asistentul completează o investigație, automat sunt introduse numele și prenumele lui în baza de date, la investigația pe care acesta a completat-o



Un medic poate vedea pacienții programați pentru ziua curentă, poate consulta istoricul unui pacient după un CNP dat, poate adăuga sau șterge un serviciu medical, după cum consideră el. Pentru un raport medical, el poate verifica analizele care s-au făcut, iar dacă consideră că totul este realizat, poate completa diagnosticele, simptomele și recomandările pentru o consultație după care o parafează. Din acel moment, nu se mai pot adăuga/ șterge servicii medicale legate de acea consultație. Însă diagnosticul pus la acea consultație va apărea în istoricul pacientului. La fel ca mai sus, toate aceste butoane fac vizibile anumite JPanel-uri, ce conțin JLabel-uri, JTextField-uri și JButton-e.

- Elemente de utilizare/ instalare

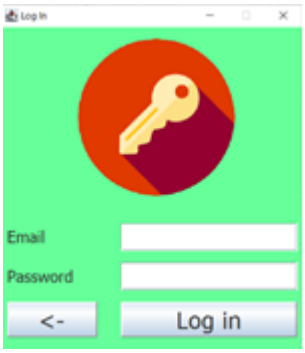
Pentru utilizarea aplicației, este nevoie de pregătirea mediului de lucru. Presupunând că pe computerul utilizatorului sunt instalate Java și MySql, primul pas este crearea bazei de date. Cu ajutorul scriptului aflat în arhivă, se vor crea mai întâi tabelele (“CreareBD.sql”), apoi se vor popular (“PopulareBD.sql”), urmând ca mai apoi să fie create, tot cu ajutorul scriptului, procedurile, funcțiile și vederile (“Functii.SQL”, “Proceduri.SQL” și “Vederi.sql”).



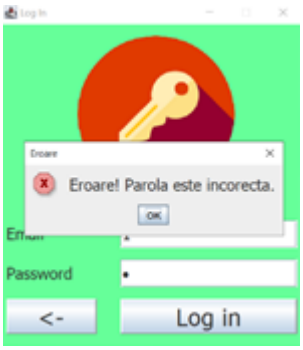
În arhivă există un fișier numit “settings.txt” ce conține 3 linii de text: prima corespunde numelui bazei de date (în cazul în care la pasul anterior s-a modificat numele bazei de date din scriptul “CreareBD.sql”, acesta trebuie să fie înlocuit in acest fișier text), a doua corespunde numele utilizatorului din MySQL Workbench (în cazul de față a fost setat la “root”, însă trebuie avut grijă ca acesta să corespundă cu cel din Workbench-ul folosit pe computerul propriu) iar pe ultima linie este parola utilizatorului Workbench (la fel ca mai sus, aceasta trebuie să fie cea de pe computerul propriu”). Și cu aceasta, mediul de lucru a fost pregătit.

Pentru a folosi propriu-zis aplicația, se va deschide arhiva Java (.JAR) “Policlinica.jar”, care se găsește și ea tot în arhivă. Dacă pașii de mai sus nu au fost respectați, va apărea eroare la conectarea aplicației la baza de date. În caz contrar, va apărea meniul principal iar aplicația va fi gata de utilizare.

- Elemente de securizare a aplicației



Aplicația este utilizată de către mai multe tipuri de utilizatori: angajat, administrator și super-administrator, fiecare având în plus acces la anumite funcții sau responsabilități. De exemplu, un raport medical poate fi completat numai de un medic; nu am vrea ca cineva care nu are cunoștințe în domeniu (precum un inspector de resurse umane sau un expert financiar) să completeze aiurea un astfel de raport. Totodată, un utilizator de tipul angajat își poate vedea informațiile personale fără a le modifica, permisiune acordată unui administrator/ super-administrator.



Astfel, fiecare utilizator ce dorește să folosească aplicația trebuie să aibă un cont valid (în caz contrar, se poate înregistra) cu care acesta se poate loga. Dacă un utilizator care nu are cont dorește să intre în aplicație acesta va primi un mesaj de eroare: "Utilizator inexistent". De asemenea, dacă un utilizator nu introduce parola sa corectă, acestuia nu i se permite accesul în aplicație și este afișat mesajul "Parola incorectă". Dar, dacă acreditările utilizatorului corespund, acesta poate intra în aplicație și p oate accesa modulele la
care acesta are permisiuni, în funcție de tipul său (angajat,
administrator sau super-administrator).

5.Concluzii, limitări si dezvoltări ulterioare

Info Med este într-o stare perfecta de utilizare atât din punct de vedere al interfeței, cat si al bazei de date. Clinicile medicale si serviciile prestate de medicii din cadrul acestora sunt de un real ajutor in viată fiecărui om, având un succes deosebit in actualitate. Aplicația este ușor navigabila si indispensabila pentru orice companie medicala. In baza de date, clienții si angajații vor găsi toate informațiile dorite într-o maniera simpla si organizata.

Ca orice aplicație, Info Med are si câteva limitări. Medicii lucrează la o singura unitate in ziua curenta, neavând posibilitatea de a-si schimba după câteva ore spațiul de lucru. De asemenea, serviciile prestate de medici sunt limitate si oricând s-ar putea extinde aplicația cu noi unități si noi angajati care sa mărească gradul de profesionalism.

- Dezvoltări ulterioare:

- Urmărirea fluxului de pacienți in cabinete in timp real;
- Posibilitatea de a trimite automat SMS-uri către pacienți, atât pentru confirmarea programărilor, cat si in scop de informare referitor la alte date considerate importante;
- Crearea unei metode de triaj al pacienților in funcție de gravitate;
- Crearea unui mecanism care sa permită decontarea de către casele de asigurări;
- Dispecerat pentru ambulanta care sa asigure evidenta permanenta a solicitărilor, păstrarea istoricului solicitărilor la nivel de pacient si care sa permită configurarea tipurilor de urgente, a ambulantelor si a șoferilor;
- Logare prin amprenta sau identificare faciala in aplicație.

BIBLIOGRAFIE

https://www.w3schools.com/sql/

https://www.reginamaria.ro/

http://redoxwap.freehostia.com/cnpgenerator.php

http://randomiban.com/?country=Netherlands