UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

PROIECT

La Disciplina
INTRODUCERE ÎN BAZE DE DATE

TITLUL LUCRĂRII:
Managementul Lanțului de Policlinici *Sanitas*

PROF. COORDONATOR: COSMINA IVAN

AUTORI: Nicoară Marusea Ioana
Pătac Mara Iulia
Popescu Ovidiu Ștefan
Stoica Iulia Cătălina

An academic 2020-2021
grupa 30224

CUPRINS

INTRODUCERE

Proiectul intitulat „Managementul Lanțului de Policlinici *Sanitas”* are ca scop simularea unui sistem de management pentru un lanț de policlinici din mai multe orașe, având fiecare particularitățile sale: program specific, număr diferit de cabinete, diferite tipuri de angajați și echipamente specifice.

Fiecare tip de angajat deține un cont personal, iar în funcție de postul pe care îl ocupă el are acces la diferite operațiuni: realizarea programărilor, vizualizarea datelor financiare cu privire la unitatea în care lucrează, vizualizarea sau editarea unor date despre angajații policlinicii.

ANALIZA CERINȚELOR UTILIZATORULUI (SPECIFICAȚIILE DE PROIECT)

Aplicația gestionează atât activitățile cadrelor medicale, cât și cele ale departamentelor financiar-contabil și resurse umane. Fiecare angajat este caracterizat printr-un set unic de date căruia îi este asociat un cont singular, care vine la pachet cu anumite permisiuni legate de responsabilitățile pe care acesta le are. Totodată, aplicația permite și înregistrarea pacienților și a datelor acestora, cu posibilitatea de vizualizare a programărilor și analizelor.

Un angajat poate să fie:

i) Doctor
   - poate vizualiza pacienții programați în ziua respectivă
   - își poate vizualiza datele, salariul și profitul personal
   - poate vizualiza tot istoricul pacientului (rapoartele medicale anterioare)
   - poate completa raportul medical al pacientului
   - adaugă/șterge servicii medicale

ii) Asistent medical
   - își poate vizualiza datele personale și salariul
   - completează buletinul de analize

iii) Recepționer
   - își poate vizualiza datele personale și salariul
   - realizează programări
   - emite bonuri fiscale

iv) Expert contabil (FR)

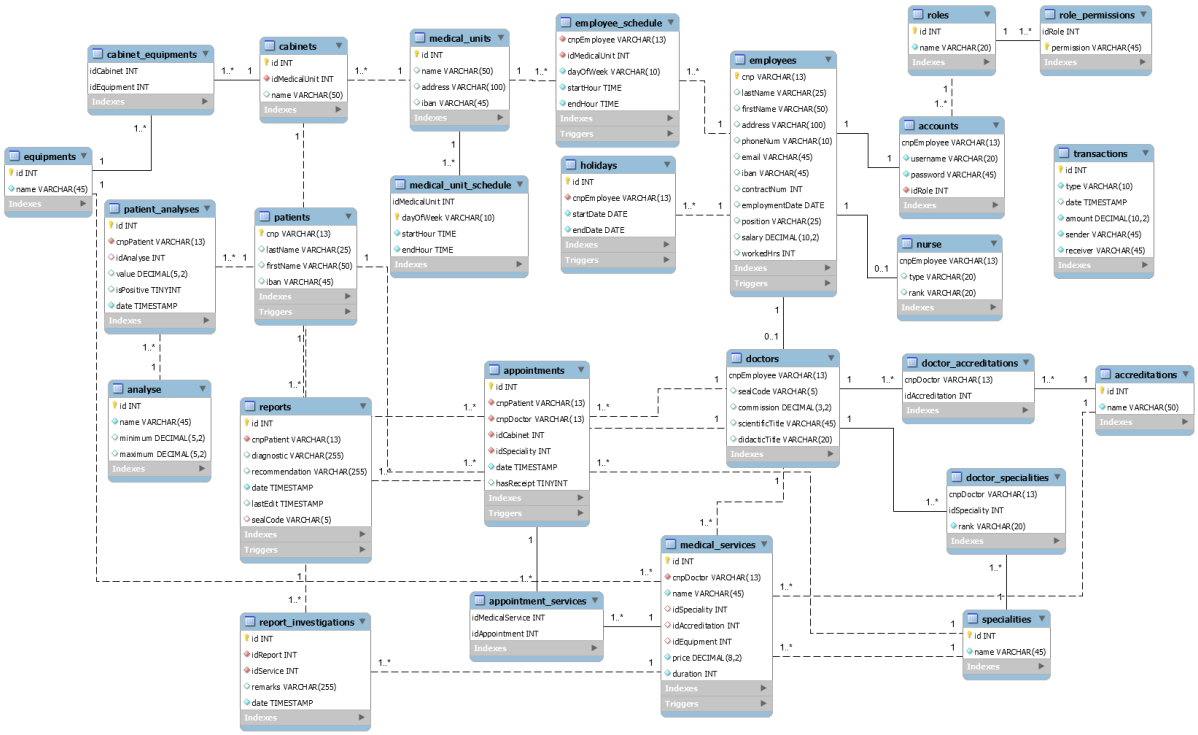
- își poate vizualiza datele personale și salariul propriu
- are acces la datele referitoare la profitul unității medicale, salariile medicilor și ale angajaților, profitul generat de medici
- poate vizualiza tranzacțiile bancare ale policlinicii, defalcate pe venituri și cheltuieli

v) Inspector resurse umane (HR)

- își poate vizualiza datele personale și salariul propriu
- adaugă/șterge/caută un angajat
- verifică orarul de muncă al unui angajat și concediile acestuia
- poate edita orarul și concediile
- poate edita detaliile unui angajat

La nivel structural proiectul este alcătuit din 26 de tabele create în limbajul SQL, folosind mediul de dezvoltare MySQL Workbench. Interfața grafică este realizată în limbajul Java, folosindu-se IDE-ul IntellIj de la JetBrains.

MODELUL DE DATE ȘI DESCRIEREA ACESTUIA



Dintre cele 26 de tabele, 6 dintre ele sunt:

I. Accounts – stochează informații despre conturile utilizatorilor: nume de utilizator, parolă, CNP angajat și rolul angajatului. Aceste tabel este în relație cu tabelele *employees* și *roles*.

II. Medical Units - stochează informații despre numele unității medicale, adresa și IBAN-ul acesteia. Acest tabel este în relație cu *cabinets*, *employee_schedule* și *medical_unit_schedule.*

III. Employees – stochează informații despre angajații lanțului de policlinici: CNP, nume și prenume, adresă, număr de telefon, email, IBAN, număr de contract, data angajării, post, salariu, orele de muncă prevăzute în contractul de muncă. Acest tabel este în relație cu *doctors*, *accounts, nurse* și *holidays.*

IV. Doctors - stochează informații detaliate despre medicii unității medicale: CNP, parafă, comision, titlu științific, titlu didactic. Acest tabel este în relație cu *reports*, *employees, specialities, accreditation, medical_services, appointments.*

V. Patients - stochează informații despre pacienții lanțului de policlinici, precum: CNP, nume, prenume, IBAN. Acest tabel este în relație cu *doctors*, *accounts, nurse* și *holidays.*

VI. Reports - stochează informații fișele medicale ale pacienților: CNP pacient, diagnostic, recomandări, data, ultima editare și parafă. Acest tabel este în relație cu *doctors.*

Procedurile stocate în baza de date verifică corectitudinea datelor trimise prin parametrii, precum existența unui angajat la ștergerea acestuia, verificarea disponibilității unui medic pentru o programare sau programul policlinicii atunci când se modifică orarul unui angajat. Din cele 36 de proceduri ale bazei de date, cele mai complexe sunt:

- **insert_employee**(cnp varchar(13), lastName varchar(25), firstName varchar(50), address varchar(100), phoneNum varchar(10), email varchar(45), iban varchar(45), contractNum int, employmentDate date, position varchar(25), salary decimal(10,2), workedHrs int, validation int)

Adaugă un angajat nou, completând toate câmpurile necesare(CNP, nume, prenume, adresă, email, etc.)

- **get_employee_salary**(cnp varchar(13), startDate date, endDate date, *result decimal(10,2)*)

Procedura returnează salariul unui angajat, al cărui CNP este transmis prin parametrul "cnp", pentru intervalul menționat.

- **get_profit_by_doctor**(cnp varchar(13), startDate date, endDate date, *result decimal(10,2)*)

Procedura returnează profitul generat de un medic, al cărui CNP este transmis prin parametrul "cnp", pentru intervalul menționat, fără a include salariul, făcând apel și la procedurile **get_doctor_profit_total** și **get_doctor_salary**.

- **get_doctor_salary**(cnp varchar(13), startDate date, endDate date, *result decimal(10,2)*)

Procedura returnează salariul unui medic, al cărui CNP este transmis prin parametrul "cnp", pentru intervalul menționat.

- **get_doctor_profit_total**(cnp varchar(13), startDate date, endDate date, *result decimal(10,2)*)

Procedura returnează profitul generat de un medic, al cărui CNP este transmis prin parametrul "cnp", pentru intervalul menționat, pe baza consultațiilor efectuate.

- **update_employee_schedule**(cnpEmployee varchar(13), idMedicalUnit int, dayOfWeek varchar(10), startHour time, endHour time, new_startHour time, new_endHour time, force int, *validation int*)

Procedura care modifică programul unui angajat verifică programul unității medicale în care lucrează angajatul, iar dacă angajatul este medic procedura verifică dacă modificarea de orar face imposibilă realizarea unor consultații.

- **save_medical_service**(id int, cnp varchar(13), name varchar(45), idSpeciality int, in idAccreditation int, idEquipment int, price decimal(8, 2), duration int, *validation int*)

Procedura care adaugă un serviciu medical nou verifică dacă medicul al cărui CNP este furnizat ca parametru are calificarea necesară pentru a putea realiza procedura medicală, și dacă echipamentul necesar procedurii există.

- **confirm_report**(id varchar(13), sealCode varchar(5), *validation int*)

Procedura care parafează fișele medicale, astfel încât ele nu mai pot fi modificate.

- **get_receipt**(id int, *name varchar(50)*, *patientName varchar(50), address varchar(100)*, *date timestamp*, *services varchar(255)*, *price decimal(10,2)*)

Procedura care generează bonul fiscal și returnează numele unității medicale care emite bonul fiscal, numele pacientului, adresa, data, serviciile și prețul final de achitat.

Trigger-ele create sunt secvențe de cod care se execută în mod implicit atunci când asupra tabelelor se execută operații de acces la date. Trigger-ele sunt utile deoarece permit implementarea regulilor de business specifice aplicației, în mod direct, atașat bazei de date. Sunt pre-filtre sau post-filtre care pot să împiedice sau să execute anumite acțiuni asupra datelor.

1. **on_employee_schedule_delete** – trigger pentru ștergerea orarului unui angajat
2. **on_employee_schedule_update** – trigger pentru editarea orarului unui angajat
3. **on_holiday_insert** – trigger pentru inserarea concediului
4. **on_employee_insert** – trigger pentru crearea contului unui angajat
5. **on_employee_delete** – trigger pentru ștergerea unui angajat
6. **on_patient_delete** – trigger pentru ștergerea unui pacient
7. **on_report_delete** – trigger pentru ștergerea unei investigații din fișa medicală
8. **on_appointment_delete** – trigger pentru ștergerea unei programări
9. **on_medical_service_delete** – trigger pentru ștergerea unui serviciu medical
10. **on_add_position** – trigger pentru crearea tabelului corespunzător pentru medic/asistent

Vederile create ne ajută la salvarea interogărilor pe care vrem să le utilizăm în cod ori de câte ori este necesar. Prin utilizarea vederilor simplificăm și specializăm viziunea utilizatorilor asupra datelor din baza de date.

1. **view_employees** - view cu informații despre angajați
2. **view_accounts** - view cu informații despre conturi
3. **view_employee_schedule** - view cu informații despre orarul angajaților
4. **view_holidays** - view cu informații despre concediul angajaților
5. **view_nurses** - view cu informații despre asistenți
6. **view_services** - view cu informații despre serviciile oferite
7. **view_services_by_cabinet** - view cu informații despre serviciile oferite din fiecare cabinet
8. **view_permissions** - view cu informații despre permisiunile conturilor din aplicație
9. **view_specialities_by_doctor** - view cu informații despre specialitățile unui doctor
10. **view_specialities** - view cu informații despre toate specialitățile
11. **view_accreditations_by_doctor** - view cu informații despre acreditările unui doctor
12. **view_equipments** - view cu informații despre echipamentele din cabinete
13. **view_patients** - view cu informații despre pacienți
14. **view_reports** - view cu informații despre fișa pacienților
15. **view_investigations** - view cu informații despre investigațiile unui pacient
16. **view_analyses** - view cu informații despre analize
17. **view_patient_analyses** - view cu informații despre analizele unui pacient
18. **view_doctors** - view cu informații despre doctor
19. **view_doctors_by_medical_unit** - view cu informații despre doctori din unitățile medicale
20. **view_doctors_by_speciality** - view cu informații despre doctori în funcție de specialitate
21. **view_appointments** - view cu informații despre programări
22. **view_transactions** - view cu informații despre tranzacții

23. **view_appointment_services** - view cu informații despre o programare pentru un anumit serviciu
24. **view_medical_units** - view cu informații despre unitățile medicale
25. **view_cabinets** - view cu informații despre cabinete
26. **view_role** - view cu informații despre roluri
27. **view_employee_cnp_name** - view cu informații despre cnp, numele și prenumele angajaților
28. **view_doctor_cnp_name** - view cu informații despre cnp, numele și prenumele doctorilor

Normalizarea datelor

Definiția formei normale Boyce-Codd (FNBC) este: Fie R o schemă de relație si F mulțimea de dependențe funcționale asociată. Se spune ca R este în forma normală Boyce-Codd dacă și numai dacă oricare ar fi o dependență netrivială X →Y din F , atunci X este supercheie pentru R.

Baza de date respectă. Atributele fiecărui tabel nu depind de alte atribute. Fiecare tabel are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date.

În fiecare tabel avem doar o cheie și toate dependențele au în partea stângă o supercheie (cheia primară a tabelei). De exemplu, pentru tabela *Employees* avem cheia primară *cnp* și toate dependențele au în stânga această supercheie, în tabelul Doctors această supercheie este *cnpEmployee* etc.

Interogări din MySQL în algebră booleană:

```
SELECT P.`lastName`, ' ', P.`firstName`
        FROM `appointments` A, `patients` P
        WHERE P.`cnp` = A.`cnpPatient` AND A.`id` = `_id`;
```

$\Pi_{PLASTNAME,PFIRSTNAME}(\sigma_{ID=\_ID}(APPOINTMENTS))_{CNPPACIENT=CNP}(PATIENTS)$

```
SELECT COUNT(*)
        FROM `reports`
        WHERE `id` = `_idReport`
```

$\Pi_{COUNT(*)}(\sigma_{ID=\_IDREPORT}(REPORTS))$

```
SELECT COUNT(*)
        FROM `reports`
        WHERE `id` = `_id` AND `sealCode` IS NULL
```
$\Pi_{COUNT(*)}(\sigma_{ID=\_ID \wedge SEALCODE\ IS\ NULL}(REPORTS))$

```
SELECT IFNULL(SUM(`amount`), 0)
        FROM `transactions`
        WHERE DATE(`date`) >= `startDate` AND DATE(`date`) <= `endDate` AND `type`
= 'outcome';
```
$\Pi\ IFNULL(SUM(`AMOUNT`), 0)\ (\sigma\ DATE(`DATE`) >= `STARTDATE` \wedge DATE(`DATE`) <= `ENDDATE` \wedge TYPE` = 'OUTCOME'\ (TRANSACTIONS))$

```
SELECT CONCAT(P.`lastName`, ' ', P.`firstName`) INTO `_patientName`
                FROM `appointments` A, `patients` P
      WHERE P.`cnp` = A.`cnpPatient` AND A.`id` = `_id`;
```
$\Pi\ CONCAT(P`LASTNAME`, ' ', P`FIRSTNAME`)\ (\sigma\ A.`ID` = `\_ID\ (APPOINTMENTS)\ )_{CNPPATIENT=CNP}$
(PATIENTS)

```
SELECT `idMedicalService`, COUNT(*)
        FROM `appointment_services` A, `appointments` AP
        WHERE AP.`cnpDoctor` = `_cnp` AND AP.`id` = A.`idAppointment` AND
DATE(AP.`date`) >= `startDate` AND DATE(AP.`date`) <= `endDate` GROUP BY
A.`idMedicalService`;
```

$idMedicalService\Pi_{`IDMEDICALSERVICE`,\ COUNT(*)}\ (\sigma\ DATE\ (`DATE`) >= `STARTDATE \wedge DATE(`DATE`) <= `ENDDATE` \wedge CNPDOCTOR` = `\_CNP`\ (APPOINTMENTS))\ .`ID` = A.`IDAPPOINTMENT`$
(APPOINTMENT_SERVICES)

DETALII DE IMPLEMENTARE

Nicio aplicație nu poate există fără o interfață grafică care să faciliteze accesul utilizatorului la date și specificații. Acest proiect implementează o interfață grafică în Java, astfel la realizarea ei s-au folosit clase de obiecte, dintre care amintim cele mai importante pentru fiecare sector de activitate în parte:

ER:

PanelDoctorProfitTotal – panel pentru afișarea în aplicație a profitului total al unui doctor, calculat într-un anumit interval de timp

PanelDoctorSalary – panel pentru afișarea în aplicație a salariului unui doctor, calculat într-un anumit interval de timp

PanelEmployeeSalary – panel pentru afișarea în aplicație a salariului unui angajat, calculat într-un anumit interval de timp

PanelMedicalUnitProfit – panel pentru afișarea în aplicație a profitului unei unități medicale, calculat într-un anumit interval de timp

PanelProfitByDoctor – panel pentru afișarea în aplicație a profitului unui doctor, calculat într-un anumit interval de timp

PanelProfitBySpeciality – panel pentru afișarea în aplicație a profitului pe specialități, calculat într-un anumit interval de timp

PanelShowTransactions – panel pentru afișarea în aplicație a tranzacțiilor, dintr-un anumit interval de timp

PanelTotalProfit – panel pentru afișarea în aplicație a profitului total, calculat într-un anumit interval de timp

HR:

PanelAddEmployee – panel pentru afișarea în aplicație a posibilității de adăugare a unui angajat din aplicație

PanelEditEmployee – panel pentru afișarea în aplicație a posibilității de editare a datelor despre angajați, ștergere a unui angajat, adăugarea unui angajat

PanelEditSchedule – panel pentru afișarea în aplicație a posibilității de editare a orarului angajaților

PanelInsertHoliday – panel pentru afișarea în aplicație a posibilității de inserare a concediilor

PanelInsertSchedule – panel pentru afișarea în aplicație a posibilității de inserare a orarelor

PanelShowEmployee – panel pentru căutarea în aplicație a angajaților

PanelViewEmployee – panel pentru afișarea în aplicație a angajaților

PanelViewHoliday – panel pentru afișarea în aplicație a concediilor

PanelViewSchedule – panel pentru afișarea în aplicație a orarelor

MR:

PanelAddAnalyse – panel pentru afișarea în aplicație a posibilității de adăugare a unui buletin de analize

PanelAddAppointment – panel pentru afișarea în aplicație a posibilității de adăugare a unei programări

PanelAddInvestigation – panel pentru afișarea în aplicație a posibilității de adăugare a unei investigații

PanelAddPatient – panel pentru afișarea în aplicație a posibilității de adăugare a unui pacient

PanelEditPatient – panel pentru afișarea în aplicație a posibilității de adăugare a unui buletin de analize

PanelSearchPatient – panel pentru afișarea în aplicație a posibilității de căutare a unui pacient

PanelShowAnalyses – panel pentru afișarea în aplicație a analizelor

PanelShowAppointments – panel pentru afișarea în aplicație a programărilor

PanelShowMedicalServices – panel pentru afișarea în aplicație a serviciilor medicale

PanelShowPatients – panel pentru afișarea în aplicație a pacienților

PanelShowRecipt – panel pentru afișarea în aplicație a rețetelor

PanelShowReports – panel pentru afișarea în aplicație a fișelor medicale

PanelViewAppointment – panel pentru afișarea în aplicație a unei programari

PanelViewInvestigation – panel pentru afișarea în aplicație a unei investigații

PanelViewMedicalService – panel pentru afișarea în aplicație a unui serviciu medical

PanelViewPatient – panel pentru afișarea în aplicație a unui pacient

PanelViewReport – panel pentru afișarea în aplicație a unei fișe medicale

PROFILE

PanelEmployeeProfile – panel pentru afișarea în aplicație a datelor unui angajat

Securitatea bazei de date

Securitatea este o problemă cu care se confruntă orice aplicație sau sistem de gestiune, însă există soluții pentru rezolvarea acestei probleme. Baza de date a fost proiectată astfel încât, prin crearea unui utilizator al bazei de date cu drepturi restrânse, datele din tabele nu pot fi compromise.

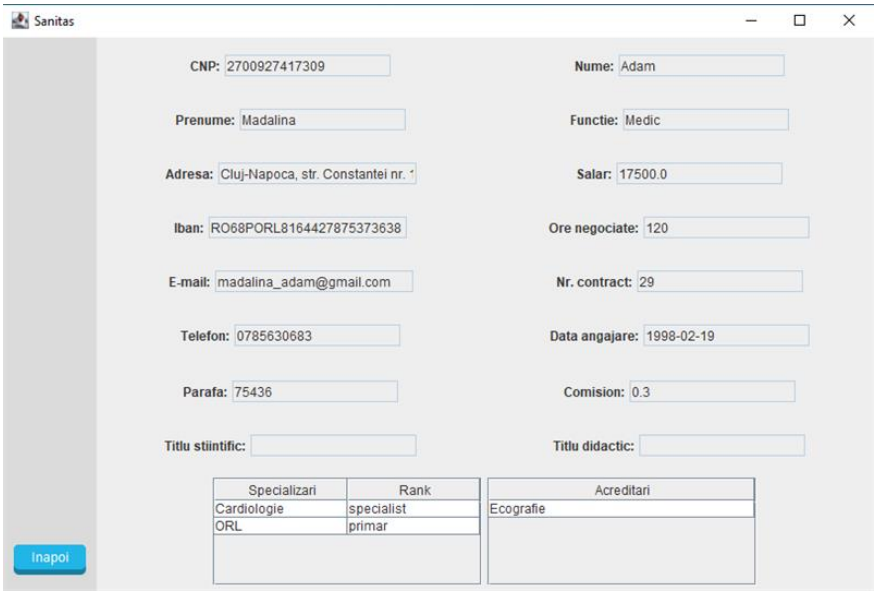
O măsură de siguranță folosită pentru conectarea utilizatorului este folosirea unei proceduri care compară datele trimise ca parametru cu cele din baza de date, fără a trimite către aplicație parola sau user-ul corect.

Instrucțiuni de utilizare

Pentru utilizarea aplicației utilizatorul își va introduce datele contului în fereastra de mai jos:

Iar apoi în funcție de sarcina pe care angajatul o are de făcut, el alege butonul corespunzător:

Exemplu pentru butonul “Your profile”:



CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE

La momentul actual aplicația simulează gestiunea sistemului unui lanț de policlinici, însă ca dezvoltare ulterioară se poate adaugă un sistem de sugestii pentru data programărilor, a orarelor angajaților (algoritmul va sugera date valide din programul fiecărei entități menționate, sugerând astfel cea mai bună variantă pentru realizarea unei programări).

Pentru a spori securitatea aplicației se dorește introducerea autentificării prin 2 factori (2-factor authentication).

Totodată se pot adăuga în viitor grafice și statistici cu privire la evoluția veniturilor și a cheltuielilor, profitul generat de policlinici sau număr de angajați existenți.