



UNIVERSITATEA TEHNICĂ "GH ASACHI" IAȘI
FACULTATEA AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA BAZE DE DATE

Gestionare cabinet medical

Coordonator,

Prof. Mironeanu Cătălin

Student,

Dumitru Ioan-Andrei

Holban Mihnea-Bogdan

Grupa 1306B

Iași, 2022

Titlul proiectului

Aplicatie de gestionare a unui cabinet medical

Analiza, proiectarea și implementarea unei baze de date care să modeleze gestiunea unui cabinet medical.

Descrierea proiectului

Aplicația va fi construită utilizând framework-ul Spring Boot pe partea de backend și HTML, CSS și Bootstrap pe partea de frontend. Spring Boot ne va oferi o serie de instrumente puternice pentru a construi rapid și eficient o aplicație web care să răspundă rapid cererilor utilizatorilor.

În ceea ce privește funcționalitățile aplicației, aceasta va oferi posibilitatea de a gestiona pacienții cabinetului medical, inclusiv adăugarea de noi pacienți și actualizarea informațiilor lor. De asemenea, aplicația va permite programarea de consultații și va oferi posibilitatea de a vedea toate programările existente într-un tabel pentru medicii care au acces la programari.

Dupa pornirea aplicatiei din I.D.E-ul preferat se introduce in bara de search localhost:PORT. Portul apare in fisierul applications.properties din folderul resources. Pagina principala are 3 butoane. Sign up, sign in care pot fi utilizate doar de medici si butonul de appointments in care un pacient isi poate face o programare completand Numele, emailul, data la care vrea sa fie programat si doctorul la care vrea sa aiba programarea. Daca totul este in regula acesta va fi redirectionat pe o pagina care il va informa ca programarea s-a realizat cu succes. Daca nu, acesta va vedea un mesaj de eroare care poate aparea din cauza ca a selectat o ora/zi la care este alta programare, sau un doctor care are deja programarea la acea ora/zi.

Pentru conectarea doctorilor, dupa apasarea butonului de sign in va fi redirectionat catre o pagina care il roaga sa introduca emailul si parola, ambele fiind stocate in tabela USER. Parola este hashuita in baza de date folosind biblioteca BCrypt din Spring Security. Daca logarea se face cu succes acesta va intra pe o pagina unde va vedea un tabel cu toti pacientii care au o programare la el, data programarii etc. De asemenea apar diferite butoane.

În cazul în care doctorul apasă pe butonul de update acesta va fi redirectionat la o pagină în care poate selecta data rezultatului, operația pe care pacientul va trebui să o efectueze (sau nu) și problema medicală a acestuia. Dacă un client are mai multe probleme medicale depistate, la un nou update problemele vor dispărea din listă. Doctorul este obligat să selecteze o dată rezultat mai mare decât data la care pacientul a realizat programarea pentru a nu apărea o eroare de tip check.

În cazul în care doctorul apasă pe butonul de delete, programarea va fi ștearsă din baza de date și pagina se va actualiza corespunzător.

De asemenea, doctorul mai are opțiunea de a selecta un pacient apăsând pe butonul cu numele lui fiind redirectionat pe o pagină unde sunt afișate toate probleme medicale ale acestuia și costul total al operațiilor. Apare și un buton de return care îl va întoarce pe pagina cu programările.

Un ultim buton este cel de sign out care îl deloghează pe doctor din aplicație. Butonul de signup poate fi utilizat doar de medicii care au în prealabil un email și hire_date în baza de date. Altfel va apărea o eroare care îl roagă să contacteze administratorul de sistem. Dacă totul merge în regulă, parola și emailul vor fi salvate în aceeași tabelă USER. În concluzie, aplicația web pentru gestionarea cabinetului medical va fi un instrument indispensabil pentru orice cabinet medical care dorește să-și automatizeze procesele și să-și ușureze munca. Utilizând tehnologii puternice precum Spring Boot și Bootstrap, aceasta va fi rapidă, eficientă și ușor de utilizat pentru toți utilizatorii.

Tehnologii utilizate

Front-End

HTML (HyperText Markup Language) este un limbaj de marcare folosit pentru a crea documente web. HTML folosește etichete pentru a marca diferite părți ale documentului, cum ar fi titlul, paragrafele, imagini și legături, pentru a le face mai ușor de interpretat de către un browser web. HTML este un limbaj de bază pentru dezvoltarea de site-uri web și este utilizat împreună cu alte tehnologii precum **CSS** (Cascading Style Sheets) și JavaScript pentru a crea site-uri web interactive și frumoase.

Bootstrap este o colecție de instrumente de design pentru front-end care ofera un set de clase CSS, componente HTML si plugin-uri JavaScript predefinite pentru a crea un design responsiv pentru aplicatiile web si site-urile. Bootstrap a fost creat pentru a facilita procesul de dezvoltare a site-urilor prin oferirea unui set standard de elemente de design care pot fi folosite in orice proiect. Acesta poate fi utilizat atat de programatori cat si de designeri pentru a crea site-uri atractive si usor de utilizat.

Back-End

Spring Boot este o platforma open-source Java care ofera un mod simplu de a crea aplicatii standalone care pot functiona fara a necesita configurare suplimentara. **Spring MVC** este un framework web open-source care se bazeaza pe modelul-vizual de controler pentru a construi aplicatii web bazate pe Java. **Spring Security** este o suita de instrumente open-source care ofera protectie pentru aplicatiile web Java prin autentificare, autorizare si protectie impotriva atacurilor.

BCrypt este un algoritm de criptare de parola care poate fi folosit pentru a stoca parolele de utilizator intr-un mod mai sigur decat prin utilizarea altor metode de criptare. Acesta poate fi utilizat impreuna cu Spring Security pentru a oferi o protectie suplimentara pentru parolele utilizatorilor. BCrypt este cunoscut pentru faptul ca este dificil de spart, deoarece utilizeaza o cheie de criptare generata aleatoriu pentru a cripta parolele de utilizator.

Thymeleaf este un motor de template Java care poate fi utilizat pentru a procesa documente HTML, XML, sau alte tipuri de documente care contin elemente de marcare. Acesta poate fi integrat cu aplicatiile Java pentru a le oferi o metoda de afisare a continutului dinamic in documentele de marcare, in loc sa se foloseasca cod pur Java pentru a genera continutul HTML. Thymeleaf ofera suport pentru variate tipuri de aplicatii, inclusiv aplicatii web, aplicatii standalone si aplicatii mobile. Acesta poate fi utilizat impreuna cu alte framework-uri Java, cum ar fi Spring MVC, pentru a oferi o metoda de afisare a continutului dinamic intr-un site web sau aplicatie web.

Lombok este o bibliotecă Java care oferă suport pentru codul concis și redus prin adăugarea de anotatii la clasele Java. Acest lucru permite utilizatorilor să scrie cod mai puțin verbos și mai ușor de citit, deoarece Lombok poate genera automat metode comune, cum ar fi getter-i, setter-i, constructorii și metodele equals și hashCode. Lombok poate fi folosit cu orice editor de cod Java și poate funcționa în orice proiect Java care utilizează compilarea prin anotatii. Utilizarea Lombok poate face codul mai ușor de scris și de întreținut, deoarece utilizatorii nu mai trebuie să scrie manual acele metode comune care sunt necesare în multe clase Java.

SQL (Structured Query Language) este un limbaj standard pentru accesarea bazelor de date. **Oracle** este un sistem de gestiune a bazelor de date (DBMS) care folosește SQL ca limbaj de interogare. Oracle oferă o varietate de funcții puternice pentru gestionarea bazelor de date, inclusiv suport pentru transacții, indexuri, și constrângeri de cheie. Oracle poate fi folosit pentru a stoca și gestiona o mare varietate de date, cum ar fi datele financiare, datele de producție, și datele de client.

Structura și relațiile dintre tabele

Tabelele din această aplicație sunt:

- o Appointments
- o Doctors
- o Medical_Problems
- o Patient
- o Problems_Patient_FK
- o Surgery
- o Users

În proiectarea acestei baze de date s-au identificat următoarele tipuri de relații:
1:1 (one-to-one), 1:n (one-to-many), n:m(many-to-many).

Între tabela **Doctors** și tabela **Users** se stabilește o relație 1:1, deoarece fiecare doctor este unic, legatura este făcută de coloana **doctor_id**.

Între tabela **Doctors** și tabela **Appointments** se stabilește o relație 1:n, deoarece o consultatie este facuta de un medic iar un medic poate face mai multe consulatii, iar legatura dintre tabele este **doctor_id**.

Între tabelele **Patient** și **Appointments** este o relație 1:n, deoarece un pacient isi poate face mai multe programari la o consultatie, dar la o consultatie poate participa un singur pacient . Legătură este factură prin **patient_id**.

Între tabelele **Surgery** și **Appointments** este o relație 1:n deoarece la rezultatul consultatiei unui pacient rezulta o singura operatie (sau niciuna) si aceeasi operatie poate aparea la mai multe consulatii. Legatura dintre tabele este făcută prin **surgery_id**.

Între tabelele **Patient** și **Medical_Problems** este o relație n:m deoarece mai mulți pacienți pot avea aceeași problema medicala și mai multe probleme medicale pot aparea la acelasi pacient. Legatura este o cheie compusa **prob_id si patient_id**.

Normalizari

Urmatoarele tabelele sunt normalizate la Forma Normală 1 (1NF):

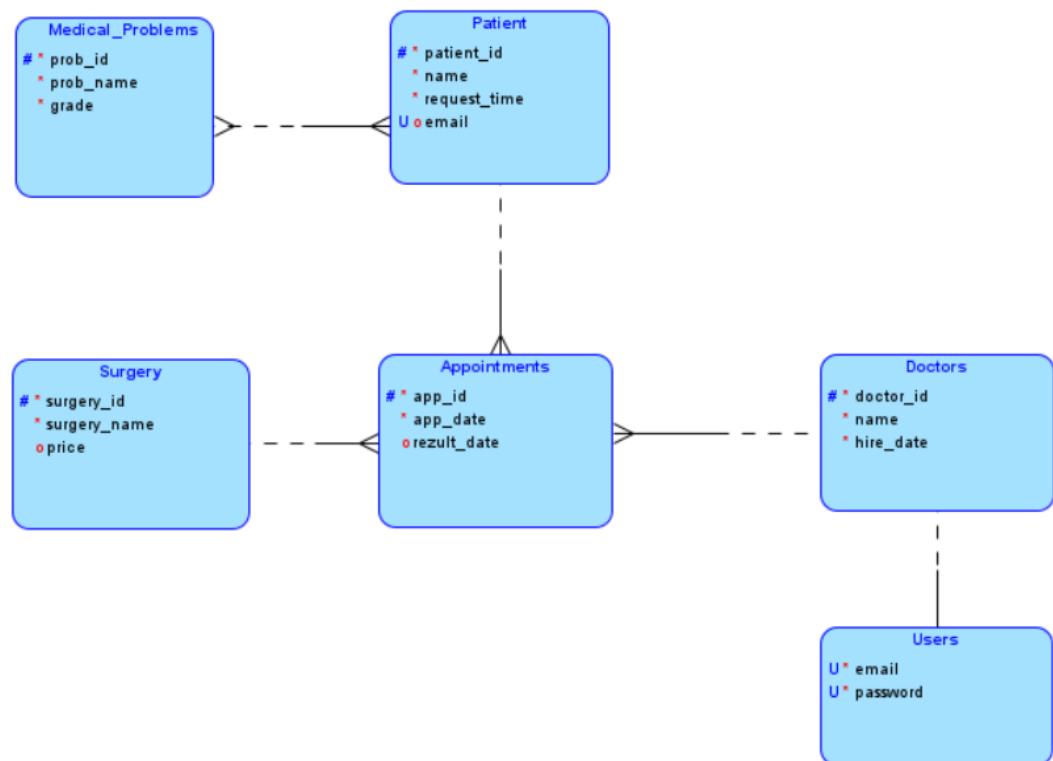
- "doctors"
- "patient"
- "surgery"
- "medical_problems"
- "users"
- "appointments"

Toate aceste tabele au o cheie primară și nu au dependențe funcționale nedorite între coloanele care nu fac parte dintr-o cheie primară. Acest lucru îndeplinește criteriile pentru Forma Normală 1 (1NF).

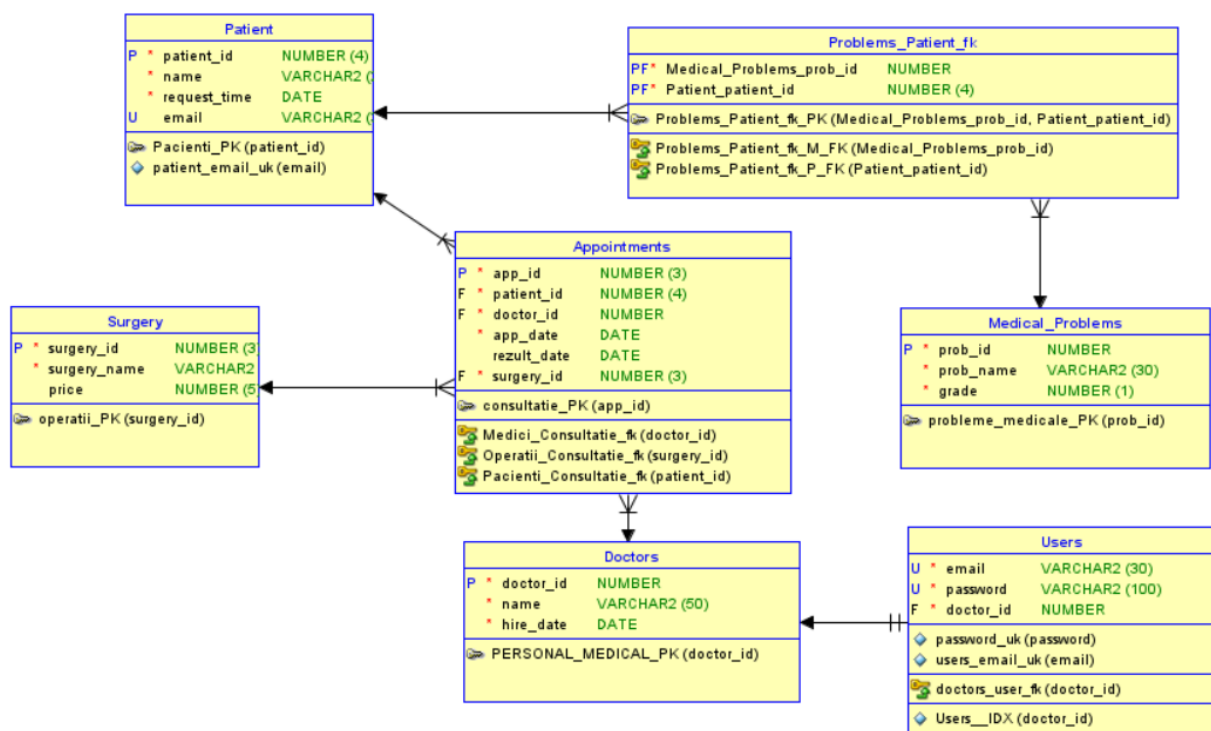
Tabela "problems_patient_fk" nu este normalizată la Forma Normală 1 (1NF) deoarece nu are o cheie primară și este utilizată doar pentru a stabili o relație între tabelele "medical_problems" și "patient". Aceasta este o tabelă de legătură, care nu conține date proprii-zise, ci doar chei străine care fac legătura între rândurile din celelalte două tabele.

Tabela "patient" este normalizată la Forma Normală 3 (3NF). Coloana "patient_id" este cheia primară și nu există dependențe funcționale nedorite între coloane, astfel încât tabela este normalizată la forma 3NF

Modelul logic



Modelul relațional



Descrierea constrângerilor

- **Appointments**

Constrangerea "app_date_ck" este de tip CHECK si verifica daca data programarii (stocata in coloana "app_date") este dupa data de 1 ianuarie 2024.

Constrangerea "consultatie_pk" este de tip PRIMARY KEY si defineste coloana "app_id" drept cheie primara a tabelului.

Constrangerea "medici_consultatie_fk" este de tip FOREIGN KEY care face referinta la tabela doctors.

Constrangerea "operatii_consultatie_fk" este de tip FOREIGN KEY care face referinta la tabela surgery.

Constrangerea "operatii_consultatie_fk" este de tip FOREIGN KEY care face referinta la tabela patient.

Apare o constrangere de tip NOT NULL la coloanele "app_date".

- **Doctors**

Constrangerea "doctors_name_ck" este de tip CHECK si verifica daca numele doctorului este format din cel putin trei caractere si daca numele este alcatuit doar din litere si spatii.

Constrangerea "personal_medical_pk" este de tip PRIMARY KEY si defineste coloana "doctor_id" drept cheie primara a tabelului.

Apar 2 constrangeri de tip NOT NULL la coloanele "name" si "hire_date".

- **Medical_Problems**

Constrangerea "grade_ck" este de tip CHECK si verifica daca valoarea din coloana "grade" este una dintre cele cinci valori posibile: 1, 2, 3, 4, 5.

Constrangerea "probleme_medicale_pk" este de tip PRIMARY KEY si defineste coloana "prob_id" drept cheie primara a tabelului.

Apar 2 constrangeri de tip NOT NULL la coloanele "surgery_name" si "grade".

- **Patient**

Constrangerea "patient_name_ck" este de tip CHECK si verifica daca numele pacientului este format din cel putin trei caractere si daca numele este alcatuit doar din litere si spatii.

Constrangerea "patient_email_ck" este de tip CHECK si verifica daca adresa de email a pacientului respecta formatul standard pentru adresele de email (adica contine un nume de utilizator, un caracter "at" si un domeniu).

Constrangerea "pacienti_pk" este de tip PRIMARY KEY si defineste coloana "patient_id" drept cheie primara a tabelului.

Constrangerea "patient_email_uk" este de tip UNIQUE si garanteaza faptul ca nu exista doua inregistrari in tabel cu aceeasi valoare pentru coloana "email".

Apar 2 constrangeri de tip NOT NULL la coloanele "name" si "request_time".

- **Surgery**

Constrangerea "operatii_pk" este de tip PRIMARY KEY si defineste coloana "surgery_id" drept cheie primara a tabelului.

Apare o constrangere de tip NOT NULL la coloanele "surgery_name".

- **Users**

Constrangerea "email_ck" este de tip CHECK si verifica daca adresa de email a utilizatorului respecta formatul standard pentru adresele de email (adica contine un nume de utilizator, un caracter "@" si un domeniu).

Constrangerea "password_ck" este de tip CHECK si verifica daca parola utilizatorului are cel putin opt caractere.

Constrangerea "doctors_users_fk" este de tip FOREIGN KEY care face referinta la tabela doctors.

Apar 2 constrangeri de tip NOT NULL la coloanele "email" si "password".

Apar 2 constrangeri de tip UNIQUE la coloanele "email" si "password".

- **Problems_Patient_FK**

Constrangerea "problems_patient_fk_pk" este de tip PRIMARY KEY si defineste coloanele "medical_problems_prob_id" si "patient_patient_id" drept cheie primara a tabelului.

Descrierea modalitatii de conectare la baza de date din aplicatie

Acest cod conectează la o bază de date Oracle folosind JDBC (Java Database Connectivity).

Mai întâi, se încarcă driverul Oracle JDBC prin apelarea metodei `Class.forName` și apoi se creează o conexiune la baza de date specificând adresa de conectare ("jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl") și credențialele de autentificare (utilizator: "bd024" și parolă: "bd024").

Apoi, se creează un obiect `PreparedStatement` folosind metoda `con.prepareStatement` și se specifică o interogare SQL care selectează înregistrările din tabela "users" unde coloana "email" este egală cu email-ul specificat în obiectul `user`.

După aceea, se apelează metoda `setString` a obiectului `PreparedStatement` pentru a seta parametrul interogării (în acest caz, "?") cu valoarea email-ului utilizatorului.

În final, se apelează metoda `executeQuery` pentru a trimite interogarea la baza de date și se obține un obiect `ResultSet` care conține rezultatele interogării.

Dacă obiectul `ResultSet` conține cel puțin o înregistrare (se verifică apelând metoda `next`), atunci se extrag valorile pentru coloanele "doctor_id" și "password" și se verifică dacă parola introdusă de utilizator se potrivește cu parola din baza de date folosind metoda `passService.checked`.

În final, se închide conexiunea la baza de date prin apelarea metodei `con.close`.

```
try {  
  
    Class.forName( className: "oracle.jdbc.driver.OracleDriver");  
  
    Connection con = DriverManager.getConnection( url: "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl", user: "bd024", password: "bd024");  
  
    //Statement st = con.createStatement();  
    PreparedStatement statement = con.prepareStatement( sql: "select * from users where email = ?");  
    statement.setString( parameterIndex: 1, user.getEmail());  
  
    ResultSet resultSet = statement.executeQuery();  
  
    if(resultSet.next()) {  
        doctor_id= resultSet.getInt( columnLabel: "doctor_id");  
        password = resultSet.getString( columnLabel: "password");  
        flag = passService.checked(user.getPassword(), password);  
    }  
  
    con.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

Acesta este conectarea din fiecare metoda diferind doar string-ul din statement cat si anumite catch-uri unde a fost nevoie.

Capturi de ecran

Programarile pentru fiecare medic din cod cat si de pe aplicatia web.

```
try {

    Class.forName( className: "oracle.jdbc.driver.OracleDriver");

    Connection con = DriverManager.getConnection( url: "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl",
        user: "bd024", password: "bd024");

    //Statement st = con.createStatement();
    PreparedStatement statement = con.prepareStatement( sql: "\n" +
        "SELECT APP_ID, PATIENT_ID, DOCTOR_ID, APP_DATE, REZULT_DATE, SURGERY_ID, TO_CHAR(APP_DATE , 'HH24:MI') AS HOUR\n" +
        "FROM APPOINTMENTS\n" +
        "WHERE DOCTOR_ID=?\n" +
        "ORDER BY app_date asc");

    PreparedStatement statement2 = con.prepareStatement( sql: "\n" +
        "select a.app_id , p.name\n" +
        "from appointments a, patient p\n" +
        "where a.patient_id=p.patient_id");


    PreparedStatement statement3 = con.prepareStatement( sql: "select a.app_id, s.surgery_name \n" +
        "from appointments a, surgery s\n" +
        "where a.surgery_id=s.surgery_id(+)" );

    statement.setInt( parameterIndex: 1, doctor_id);
    ResultSet resultSet = statement.executeQuery();
    ResultSet resultSet2 = statement2.executeQuery();
    ResultSet resultSet3 = statement3.executeQuery();

    while (resultSet.next() && resultSet2.next() && resultSet3.next()) {
        int appID = resultSet.getInt( columnLabel: "app_id");
        int patientID = resultSet.getInt( columnLabel: "patient_id");
        java.util.Date appDate = resultSet.getDate( columnLabel: "app_date");
        Date rezultDate = resultSet.getDate( columnLabel: "rezult_date");
        int surgeryID = resultSet.getInt( columnLabel: "surgery_id");
        String patient_name = resultSet2.getString( columnLabel: "name");
        String surgery_name = resultSet3.getString( columnLabel: "surgery_name");
        String hour=resultSet.getString( columnLabel: "hour");
        Appointment appointment = new Appointment(appID, patientID, doctor_id, appDate, rezultDate, surgeryID,
            patient_name, surgery_name, hour);
        appointmentList.add(appointment);
        redirectAttributes.addFlashAttribute( attributeName: "appointment", appointment);
    }

    con.close();

} catch (Exception e) {
    e.printStackTrace();
}
```

 Home Sign Out								
Appointment ID	Patient Name	Doctor ID	Appointment Date	Appointment Time	Surgery Name	Result Date		
39	Ion Clopotel	2	2023-01-25	10:00	Arthroscopy	2023-01-28	Update	Delete
42	Ion Clopotel	2	2023-01-25	14:00	Arthroscopy	2023-01-31	Update	Delete
40	Razvan Clopotel	2	2023-01-26	10:00	Laparascopy	2023-02-01	Update	Delete
41	Ion Clopotel	2	2023-01-27	10:00	Appendectomy	2023-01-29	Update	Delete

Creare unui cont cu serviciile aferente de criptare in baza de date.

```

@RequestMapping(value = "/dosignup", method = RequestMethod.POST)
public ModelAndView doSignUp(@ModelAttribute("user") User user, @ModelAttribute("doctor") Doctor doctor,
                             BindingResult bindingResult) {

    SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd-MMM-yyyy");
    ModelAndView modelAndView = null;
    String hireDate = dateFormat.format(doctor.getHireDate());
    hireDate = hireDate.replace( target: "Sept", replacement: "Sep");
    System.out.println(hireDate);

    flagEmail = true;
    flagIncorrect=true;

    try {

        Class.forName( className: "oracle.jdbc.driver.OracleDriver");

        Connection con = DriverManager.getConnection( url: "jdbc:oracle:thin:@bd-dc.cs.tuiasi.ro:1539:orcl",
                                                    user: "bd024", password: "bd024");

        //Statement st = con.createStatement();
        PreparedStatement statement = con.prepareStatement( sql: "select * from doctors where name = ? and hire_date=?");
        statement.setString( parameterIndex: 1, doctor.getName());
        statement.setString( parameterIndex: 2, hireDate);
    }
}

```

```
ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

    int doctor_id = resultSet.getInt( columnLabel: "doctor_id");
    statement = con.prepareStatement( sql: "insert into users values (?, ?, ?)");
    statement.setString( parameterIndex: 1, user.getEmail());
    statement.setString( parameterIndex: 2, hashPassService.hashed(user.getPassword()));
    statement.setInt( parameterIndex: 3, doctor_id);
    statement.executeQuery();
    modelAndView = new ModelAndView( viewName: "index");

} else {
    modelAndView = new ModelAndView( viewName: "redirect:/signup");
    flagIncorrect = false;
}
con.close();

} catch (java.sql.SQLIntegrityConstraintViolationException emailCatch) {
    flagEmail = false;
    modelAndView = new ModelAndView( viewName: "redirect:/signup");


} catch (Exception e) {
    e.printStackTrace();
}

return modelAndView;
}
```



Please sign up

Enter your hire date:

☐ Remember me

Sign up

We did not find a doctor with the name
or hire date specified. Please contact the
system administrator!

Medical Problems

Severe abdominal pain and fever

swollen synovium

Swollen apendix


Total Cost of surgery: 11500\$

[Return](#)




Update appointment


Enter your result date:

Select surgery:

Select Problem:

Update

Welcome to AM Medical! We are a state-of-the-art healthcare center dedicated to providing high-quality care to our patients. Our team of experienced doctors and nurses are here to help you with all of your medical needs.

We offer a wide range of services, including primary care, specialized treatment, diagnostic testing, and more. No matter what your healthcare needs are, we have the expertise and resources to help.



Our facility is equipped with the latest technology, ensuring that you receive the best care possible. We are committed to providing a comfortable and welcoming

Task-uri

Comune:

- Interfata aplicatiei web
- Documentatie
- Proiectarea bazelor de date

Dumitru Ioan - Andrei: Clasele Controller “AppointmentsController” si “SignInController”

Holban Mihnea-Bogdan : Clasele Controller “CrudController” si “SignUpController”