**DOCUMENTAȚIA PROIECTULUI**

**Medical Appointments Application**

STUDENȚI

Preda Paula - 10LF213

Stancu Alexia - 10LF213

* **Link git: https://github.com/alexiastancu/Medical-Appointments-Application.git**
* **Prezentarea aplicației**

Aplicația noastră Android reprezintă o soluție pentru programarea pacientilor la un consult medical și oferă o experiență completă în ceea ce privește afișarea programarilor, doctorilor și specializarilor.

Prin intermediul aplicației noastre, pacientii isi pot creea o programare la un anumit medic in functie de afectiunea suferita. Baza de date stochează informații detaliate despre User, Pacient, Doctor si Programare.

Această documentație oferă o prezentare a tabelelor din baza de date definite în fragmentele de cod.

1. Tabela User (Utilizator)

- Numele Tabelei: users

| Nume Coloană | Tip | Descriere |

|-------------------|----------|-----------------------------------------|

| id | Integer | Cheie primară |

| email | String | Adresa de email a utilizatorului |

| password | String | Parola utilizatorului |

2. Tabela Patient (Pacient)

- Numele Tabelei: patients

| Nume Coloană | Tip | Descriere |

|--------------------|----------|--------------------------------------------------------|

| patient\_id | Integer | Cheie primară |

| user\_id | Integer | Cheie străină care face referire la tabela users |

3. Tabela Doctor (Medic)

- Numele Tabelei: doctors

| Nume Coloană | Tip | Descriere |

|--------------------|---------|-------------------------------------------------------|

| doctor\_id | Integer| Cheie primară |

| user\_id | Integer | Cheie străină care face referire la tabela users|

4. Tabela Appointment (Programare)

- Numele Tabelei: users

| Nume Coloană | Tip | Descriere |

|----------------------|----------|-------------------------------------------------------------|

| appointment\_id | Integer | Cheie primară |

| patient\_id | Integer | Cheie straina care face referire la tabela patients |

| doctor\_id | Integer | Cheie straina care face referire la tabela doctors |

* **Tehnologii folosite**

Biblioteca Room în Android oferă un nivel de abstractizare peste SQLite și facilitează lucrul cu bazele de date locale în aplicațiile Android.

În implementarea unei baze de date locale folosind biblioteca Room în Android, există câteva clase cheie care joacă roluri importante în procesul de stocare și interogare a datelor. Aceste clase sunt:

Entități (Entities)

În proiectul dvs., clasele User, Patient, Doctor și Appointment reprezintă entitățile bazei de date.

Fiecare entitate este marcată cu adnotarea @Entity și definește structura datelor care urmează să fie stocate în baza de date.

Entitățile specificate reprezintă tabelele din baza de date și coloanele asociate lor.

Interfețe DAO (Data Access Object)

Pentru a interacționa cu baza de date și a executa operațiuni asupra datelor, sunt definite interfețele DAO: UserDao, PatientDao, DoctorDao și AppointmentDao.

Aceste interfețe conțin metodele necesare pentru a insera, actualiza, șterge și interoga datele din baza de date, specificând operațiile corespunzătoare pentru fiecare entitate.

Metodele din interfețele DAO sunt marcate cu adnotările @Insert, @Update, @Delete și @Query în funcție de tipul de operațiune efectuată.

Clasa bazei de date (AppDatabase)

Clasa AppDatabase reprezintă conexiunea cu baza de date și acționează ca o sursă centrală pentru obținerea instanțelor interfețelor DAO.

Această clasă este marcată cu adnotarea @Database și specifică entitățile asociate (User, Patient, Doctor, Appointment) și versiunea bazei de date (version = 2).

Clasa AppDatabase conține metodele abstracte care returnează instanțele interfețelor DAO corespunzătoare.

Singleton pentru obținerea instanței bazei de date

Pentru a asigura că există o singură instanță a bazei de date în cadrul aplicației, în clasa AppDatabase este implementat un singleton.

Singleton-ul permite accesul ușor la instanța bazei de date în întreaga aplicație și previne crearea multiplă a instanțelor.

Prin utilizarea acestor clase și interfețe în proiectul dvs., puteți stoca, accesa și actualiza datele într-o bază de date locală eficientă și sigură. Interfețele DAO oferă metodele necesare pentru a efectua operațiuni CRUD (create, read, update, delete) asupra datelor, în timp ce clasa bazei de date facilitează gestionarea și accesul la baza de date.

De exemplu, interfața UserDao definește metode pentru a insera un utilizator (insert), a actualiza un utilizator (update), a șterge un utilizator (delete) și a obține un utilizator pe baza adresei de email și parolei (getUser). Similar, interfețele PatientDao, DoctorDao și AppointmentDao oferă metode specifice pentru manipularea datelor referitoare la pacienți, medici și programări.

Prin utilizarea clasei bazei de date AppDatabase, puteți obține instanțele interfețelor DAO și apela metodele corespunzătoare pentru a efectua operațiunile dorite asupra bazei de date.

Această arhitectură solidă bazată pe Room facilitează manipularea datelor locale în aplicația dvs. și vă oferă o bază solidă pentru a dezvolta funcționalități complexe, precum autentificarea, stocarea și afișarea datelor într-un mod eficient și sigur.

* **Arhitectura aplicației**

Aplicatia noastra este organizata in 3 activitati, care la randul lor sunt organizate in fragmente.

1. Doctor Activity

Clasa DoctorActivity reprezintă activitatea principală din aplicația noastra pentru doctori. Această clasă joacă un rol important în arhitectura generală a aplicației și facilitează navigarea între diferitele fragmente și interacțiunea utilizatorului cu aplicația. I

Acestă activitate oferă o interfață utilizator intuitivă și facilitează navigarea între diferite fragmente și afișarea informațiilor specifice doctorului curent.

1. Patient Activity

PatientActivity este clasa principală de activitate care servește ca punct de intrare pentru funcționalitatea legată de pacient.Inițializează controlorul de navigare și gestionează navigarea între diferite fragmente folosind NavController.

De asemenea, inflatează aspectul vizual, inițializează FloatingActionButton și gestionează apăsarea butonului de revenire.

Clasa PatientDetailsFragment este un fragment care afișează detaliile unui pacient în cadrul aplicației dvs. pentru programări medicale. Aceasta extinde clasa Fragment și are responsabilitatea de a afișa informațiile despre pacientul curent.

MainPagePatientFragment reprezintă pagina principală pentru pacient, în care sunt afișate programările pacientului.Conține un RecyclerView pentru a afișa o listă de programări utilizând AppointmentAdapter.

Folosește AsyncTask pentru a obține programările pentru pacient din baza de date și actualizează interfața utilizatorului în consecință.

Metoda refreshUI() este utilizată pentru a actualiza lista de programări atunci când apar modificări.

CreateAppointmentFragment este responsabil de crearea unei noi programări pentru pacient.

Permite utilizatorului să introducă data, ora și să selecteze un medic pentru programare.

Validează intrările și, în cazul în care sunt valide, inserează programarea în baza de date utilizând AsyncTask.

După crearea cu succes a programării, actualizează interfața utilizatorului și navighează înapoi la pagina principală.

1. Authentification Activity

Clasa SignUpFragment este un fragment responsabil pentru înregistrarea utilizatorilor. Acesta conține metode pentru gestionarea interacțiunii cu utilizatorul și tratarea acțiunilor precum completarea câmpurilor de înregistrare, validarea datelor introduse și efectuarea înregistrării în baza de date.

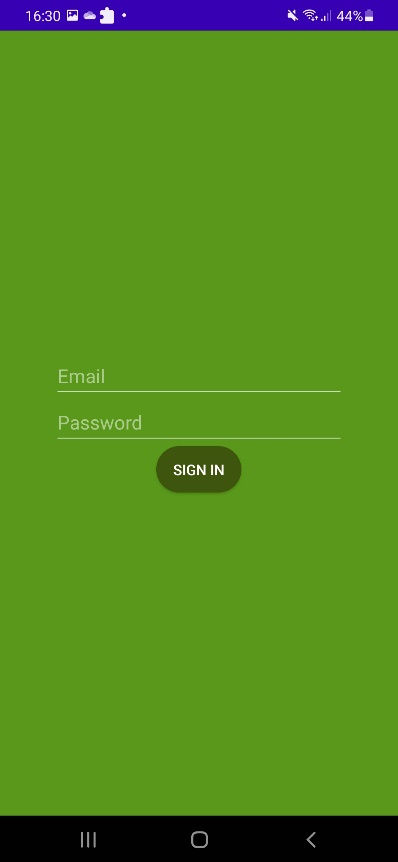
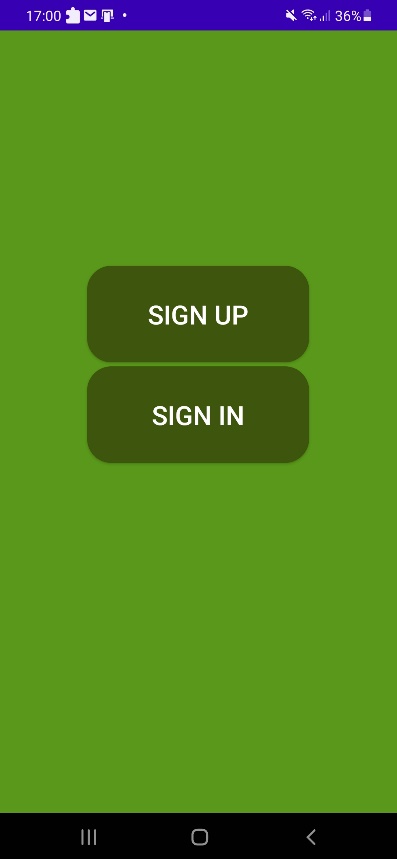
Clasa SignInFragment este un fragment responsabil pentru autentificarea utilizatorilor. Acesta conține metode pentru gestionarea interacțiunii cu utilizatorul și tratarea acțiunilor precum completarea câmpurilor de autentificare, verificarea datelor introduse și navigarea către ecranul corespunzător rolului utilizatorului autentificat.

Clasa AuthMethodFragment este un fragment responsabil pentru afișarea metodei de autentificare și permite navigarea către ecranele de înregistrare și autentificare.

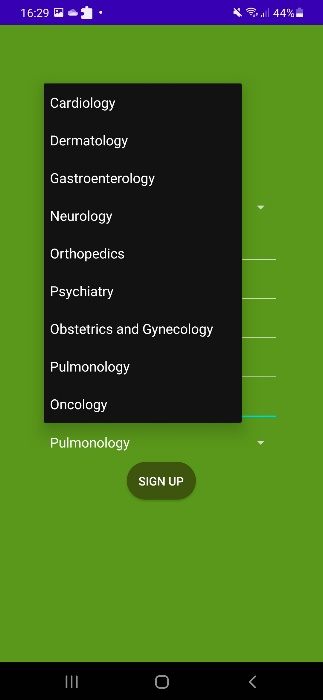
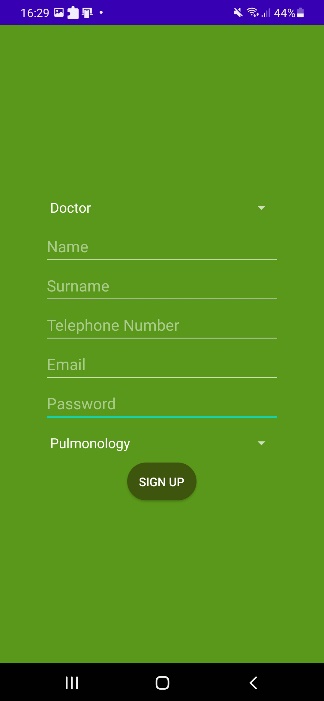
Clasa AuthentificationActivity este activitatea principală care conține fragmentele și este responsabilă pentru gestionarea navigării între fragmente.

* **Detaliile aplicației**

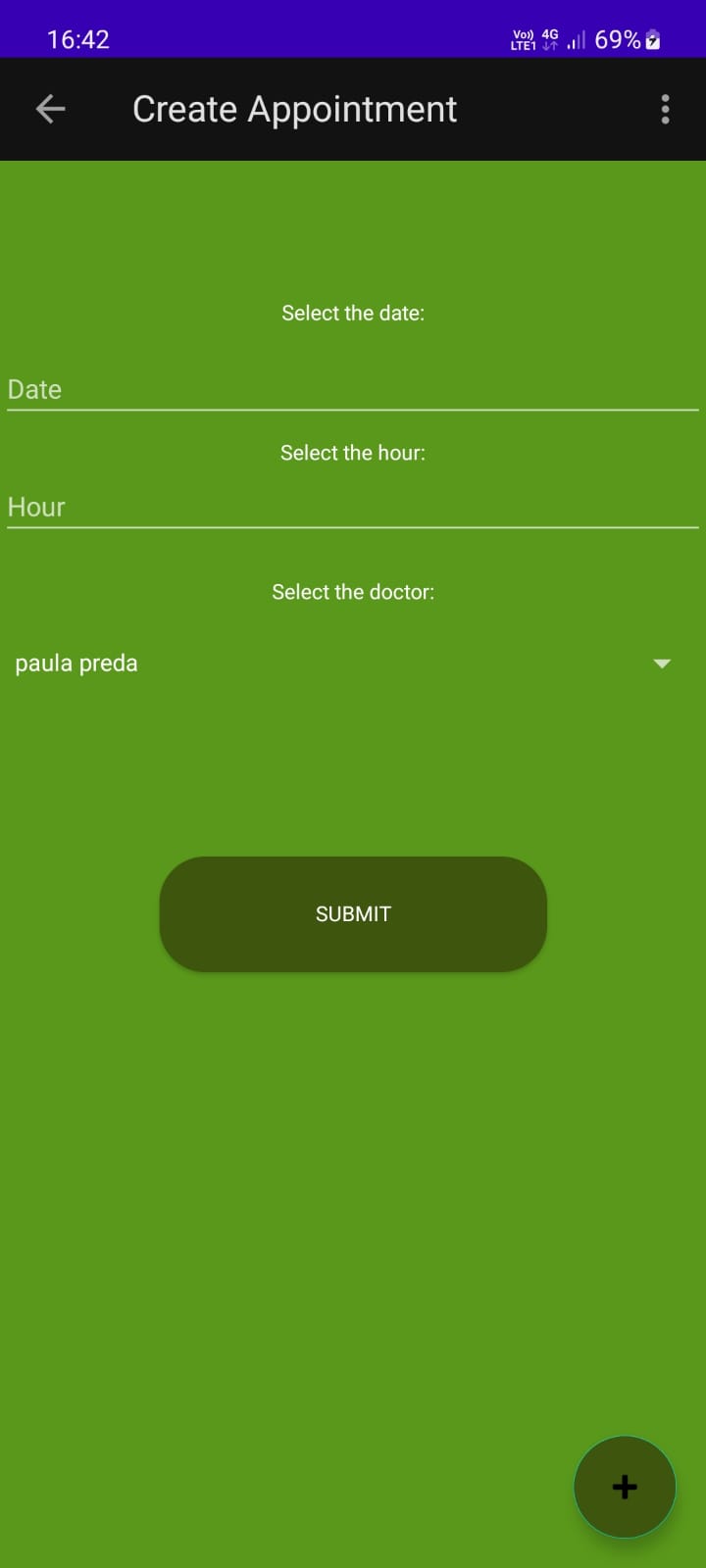
Aplicația implementează un sistem simplu de autentificare și autorizare. Pentru autentificare, utilizatorul trebuie să acceseze pagina de sign in și să introducă adresa de email și parola asociată contului său. Se verifică apoi aceste date și permite accesul utilizatorului autentificat la funcționalitățile aplicației.



Pentru înregistrare, utilizatorul trebuie să acceseze pagina de sign up și să completeze informațiile necesare, cum ar fi: atributia(doctor sau pacient) , numele, prenumele, numarul de telefon, adresa de email, parola și sectia (Cardiologie, Dermatologie, Ortopedie, etc.) . Se verifică dacă adresa de email este deja înregistrată în sistem și, în funcție de rezultat, permite înregistrarea noului utilizator sau afișează un mesaj de eroare.



Autorizarea se realizează prin atribuirea unor roluri utilizatorilor autentificați. În aplicație există două roluri definite: "Doctor" și "Pacient". Controlerele și acțiunile specifice sunt restricționate în funcție de roluri utilizând atribute de autorizare. Utilizatorii cu rolul "Pacient" au acces la funcționalități de creare a unei programari, în timp ce utilizatorii cu rolul "Doctor" pot vedea pacienti care si au facut programare la el.



Serverul verifică rolurile utilizatorului înainte de a permite accesul la acțiuni sau resurse specifice și redirecționează utilizatorii fără permisiuni către o pagină de acces refuzat.

Funcționalitățile aplicației sunt controlate în mod corespunzător în funcție de rolul fiecărui utilizator.

* **Concluzii**

În concluzie, documentația proiectului "Medical Appointments Application" prezintă o aplicație Android care facilitează programarea pacienților la consulturi medicale. Aceasta oferă o experiență completă prin afișarea programărilor, doctorilor și specializărilor disponibile. Aplicația utilizează biblioteca Room în Android pentru a gestiona baza de date locală, oferind un nivel de abstractizare peste SQLite. Arhitectura aplicației este bine structurată, având clasele entităților, interfețelor DAO și clasa bazei de date AppDatabase, care facilitează manipularea datelor locale într-un mod eficient și sigur. Aplicația este organizată în trei activități principale: DoctorActivity, PatientActivity și AuthentificationActivity, fiecare având fragmentele sale specifice. Acestea permit navigarea între diferitele funcționalități și afișarea informațiilor relevante pentru utilizatori. De asemenea, aplicația implementează un sistem de autentificare și autorizare, atribuind roluri utilizatorilor și restricționând accesul la acțiuni și resurse specifice în funcție de rolul utilizatorului. În ansamblu, aplicația oferă o soluție completă pentru programarea și gestionarea consulturilor medicale, facilitând comunicarea între pacienți și medici.