DOCUMENTAȚIA PROIECTULUI

Bloodhub

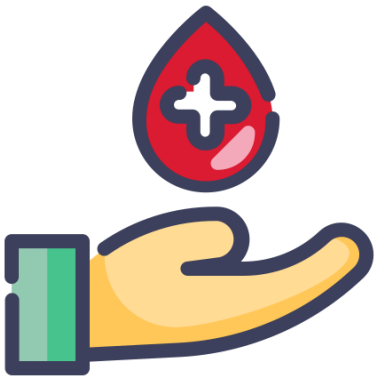
STUDENȚI
Constăndoiu Cezar
Ivana Florin-Andrei

Cuprins

Prezentarea proiectului

Introducere

Bloodhub este o aplicație mobilă destinată să faciliteze programările pentru donarea de sânge la clinicile alese de utilizatori. Aceasta se adresează atât donatorilor obișnuiți cât și celor care doresc să răspundă rapid la urgențele de sânge.



Probleme rezolvate

Lipsa de sânge în clinicile medicale este o problemă critică în multe regiuni. Adesea, donatorii nu sunt conștienți de necesitățile urgente ale clinicilor sau întâmpină dificultăți în a-și face programări eficiente pentru donare. *Bloodhub* adresează aceste probleme prin:

- Oferirea unei platforme ușor de utilizat pentru programări.
- Informarea utilizatorilor despre urgențele de sânge.
- Stocarea și gestionarea eficientă a datelor donatorilor și programărilor.

Soluții oferite

Bloodhub permite utilizatorilor:

- Să creeze și să gestioneze un profil personal cu detalii esențiale precum grupa sanguină și data nașterii.
- Să vizualizeze și să aleagă dintr-o listă de clinici disponibile pentru donare.
- Să facă programări direct prin aplicație.
- Să vizualizeze urgențele de sânge și să răspundă rapid la aceste solicitări.

Tehnologiile folosite

Limbaje și Framework-uri

- **Kotlin**: Limbajul principal de programare utilizat pentru dezvoltarea aplicației *Bloodhub*, datorită suportului său robust pentru programarea Android și interoperabilității cu Java.
- **XML**: Folosit pentru definirea și personalizarea interfeței utilizatorului.

Servicii și Biblioteci

Autentificare și Stocare

- **Firebase Auth**: Utilizat pentru autentificarea utilizatorilor. Facilitează înregistrarea și autentificarea folosind adrese de email și parole.
- **Firebase Firestore**: Oferă o bază de date NoSQL pentru stocarea datelor de profil ale utilizatorilor.

Baze de Date

- **Room Database:** Implementată pentru stocarea persistentă a datelor obținute de la API-urile externe și pentru a asigura accesul offline la acestea.

Stocare Locală

- **SharedPreferences:** Utilizat pentru stocarea unor mici seturi de date în mod persistent, cum ar fi preferințele utilizatorului și alte date de configurare.

Navigare și Arhitectura

- **AndroidX Navigation:** Utilizat pentru gestionarea tranzacțiilor între fragmente și activități, asigurând o navigare fluidă în cadrul aplicației.

Interfețe și UI

- **Material Design Components:** Bibliotecă utilizată pentru a asigura un design modern și consecvent al interfeței utilizatorului.
- **Glide:** Utilizat pentru încărcarea și afișarea eficientă a imaginilor din resurse locale sau remote.

Rețea și Operațiuni Asincrone

- **Volley:** Folosit pentru efectuarea de cereri HTTP pentru a obține date de la API-urile externe.
- **Kotlin Coroutines:** Utilizat pentru gestionarea operațiunilor asincrone, asigurând o performanță mai bună și un cod mai curat și mai ușor de întreținut.

Backend-ul aplicației

Autentificare și Înregistrare

Pentru **autentificare**, utilizatorii pot accesa aplicația prin introducerea adresei lor de email și a parolei asociate. Această funcționalitate este implementată folosind Firebase Authentication, care asigură un proces sigur și eficient de autentificare. După autentificare cu succes, utilizatorii sunt redirecționați către pagina principală a aplicației unde pot naviga la diverse secțiuni cum ar fi profilul lor, centrele de donare, programările existente și urgențele de sânge.

În ceea ce privește **înregistrarea**, utilizatorii pot crea conturi noi introducând informații cum ar fi adresa de email și parola, numele complet, data nașterii, grupa de sânge și genul. Aceste date sunt stocate în baza de date Firestore a Firebase, unde fiecare utilizator are un document dedicat în colecția „users”. Înregistrarea include și opțiunea de a adăuga o poză de profil, care este încărcată în Firebase Storage. După înregistrarea reușită, utilizatorii sunt direcționați către o pagină de succes înregistrare.

Gestionarea Profilului – Firestore

Gestionarea profilului utilizatorului este realizată prin intermediul Firebase Firestore. Acest fragment permite afișarea și actualizarea informațiilor personale ale utilizatorului, inclusiv numele complet, data nașterii, genul și grupa de sânge. Datele sunt recuperate din baza de date Firestore și afișate în interfața utilizatorului în timp real, asigurând o experiență fluidă și consistentă pentru utilizatori.

HTTP Requests

În aplicația Bloodhub, gestionarea cererilor HTTP este esențială pentru interacțiunea cu serviciile externe și actualizarea datelor din aplicație. În fragmentele DonationCentersListFragment și DonationRequestsFragment, sunt folosite cereri HTTP pentru a obține date despre centrele de donare și cereri de urgență. Aceste cereri sunt realizate folosind biblioteca Volley, care oferă o abordare simplă și eficientă pentru gestionarea cererilor HTTP în Android.

În DonationCentersListFragment, când lista de centre de donare este goală, se face o cerere GET către un URL mock pentru a obține datele despre clinici. Răspunsul JSON este parsat folosind Gson, iar obiectele rezultate sunt inserate în baza de date locală folosind ViewModel-ul asociat. Astfel, se actualizează lista de centre de donare în timp real în interfața utilizatorului.

În paralel, în DonationRequestsFragment, se face o cerere similară pentru a obține date despre urgențele de donare de sânge. Răspunsul este, de asemenea, parsat și salvat în baza de date locală pentru a afișa informațiile corespunzătoare în fragmentul de urgențe.

Arhitectura aplicației

BloodHub este o aplicație mobilă dedicată gestionării informațiilor despre donarea de sânge și creării unei comunități de donatori. Arhitectura sa modernă este concepută pentru a oferi o experiență utilizator optimizată, bazată pe tehnologii avansate și o gestionare eficientă a datelor utilizatorilor.

Structura aplicației

Aplicația *BloodHub* utilizează o arhitectură MVVM (Model-View-ViewModel), care separă logic datele (Model), interfața utilizator (View) și logica de afișare și interacțiune (ViewModel). Principalele componente ale aplicației includ:

1. **Activități** și **Fragmentare**:
    - **MainActivity**: Activitatea principală care gestionează navigarea între ecranele de autentificare și înregistrare, în funcție de starea utilizatorului.
    - **RegisterActivity**: Activitatea unde utilizatorii pot crea un cont nou. Include fragmente pentru selectarea imaginii de profil și completarea detaliilor personale.
    - **AppActivity**: Activitatea principală după autentificare, care conține meniul de navigare în partea de jos și găzduiește fragmentele pentru diferite funcționalități ale aplicației.
2. **Fragmente** și **Utilizarea lor**:
    - **UserProfileFragment**: Afișează detalii despre utilizatorul curent, inclusiv imaginea de profil, numele, data nașterii și grupa de sânge. Datele sunt preluate din Firestore și afișate utilizatorului într-un mod responsiv.
    - **RegisterFragment**: Permite utilizatorilor să completeze un formular detaliat pentru înregistrare. Utilizează coroutines pentru a gestiona operațiile asincrone de salvare a datelor în baza de date locală și cloud.
    - **RegisterSuccessFragment**: Confirmă înregistrarea cu succes și permite utilizatorilor să navigheze către ecranul de autentificare pentru a începe sesiunea lor BloodHub.
3. **Componente de UI** și **Adaptoare**:
    - **Adaptoare RecyclerView**: Cum ar fi DonationCentersAdapter, sunt folosite pentru a afișa lista de centre de donare de sânge disponibile. Acestea sunt optimizate pentru performanță și extensibilitate, permițând utilizatorilor să acceseze rapid informațiile și să interacționeze cu elementele listate.

Navigarea în aplicație

În aplicația *BloodHub*, navigarea este implementată folosind XML pentru definirea interfeței utilizatorului (UI) și gestionarea tranzacțiilor între ecranele aplicației. Structura de navigare se bazează pe activități și fragmente, optimizate pentru a oferi o experiență coerentă și eficientă utilizatorilor.

Activitatea principală, AppActivity, servește ca punct de intrare în aplicație și organizează fragmentele într-un container principal. Fiecare ecran principal, cum ar fi HomeFragment, DonationCentersFragment și UserProfileFragment, este definit în XML și încărcat dinamic în funcție de acțiunile utilizatorilor sau de starea aplicației.

Navegarea între fragmente și gestionarea stării aplicației sunt implementate folosind manageri de fragmente și metode standard Android pentru a menține consistența și performanța aplicației. De exemplu, tranzacțiile între fragmente sunt gestionate prin interfețe și schimbări de stare, asigurând o experiență fluidă în timpul navigării.

Integrarea Bazei de Date cu Coroutines

BloodHub utilizează Room Database, un ORM (Object-Relational Mapping) care facilitează interacțiunea cu SQLite, și coroutines pentru a gestiona accesul și actualizarea datelor în baza de date.

Room: Gestionarea Bazei de Date

Room este o bibliotecă oficială Android care facilitează lucrul cu baze de date SQLite prin intermediul unui nivel superior de abstractizare. În aplicația BloodHub, Room este utilizat pentru a stoca și gestiona datele local într-o manieră eficientă și structurată.

Entități și Relații

Entitățile în Room reprezintă modelele de date utilizate în aplicație și sunt mapate direct pe tabele în baza de date SQLite.

Room permite definirea a trei tipuri principale de relații între entități:

1. One-to-One
2. One-to-Many
3. Many-to-Many

Aceste relații sunt gestionate folosind anotările specifice din Room și sunt esențiale pentru a modela corect structura și interacțiunile datelor în aplicație.

DAO (Data Access Object)

DAO reprezintă o interfață prin care se definesc operațiile de acces la date pentru fiecare entitate. În aplicația *BloodHub*, DAOs sunt utilizate pentru a declara metodele care permit accesul la datele din baza de date SQLite, inclusiv operațiile CRUD (Create, Read, Update, Delete).

De asemenea, pentru a accesa datele utilizând relațiile, DAO-urile pot defini metode care să întregească informațiile din mai multe entități într-o singură interogare.

Baza de Date

Clasa care extinde RoomDatabase servește ca bază de date principală a aplicației și conține metode pentru a obține instanțe DAO și pentru a configura baza de date.

Coroutines: Gestionarea Operațiilor Asincrone

Coroutines sunt utilizate intensiv în aplicația *BloodHub* pentru a gestiona operațiile asincrone, cum ar fi interogările la baza de date, solicitările de rețea și alte sarcini intensive.

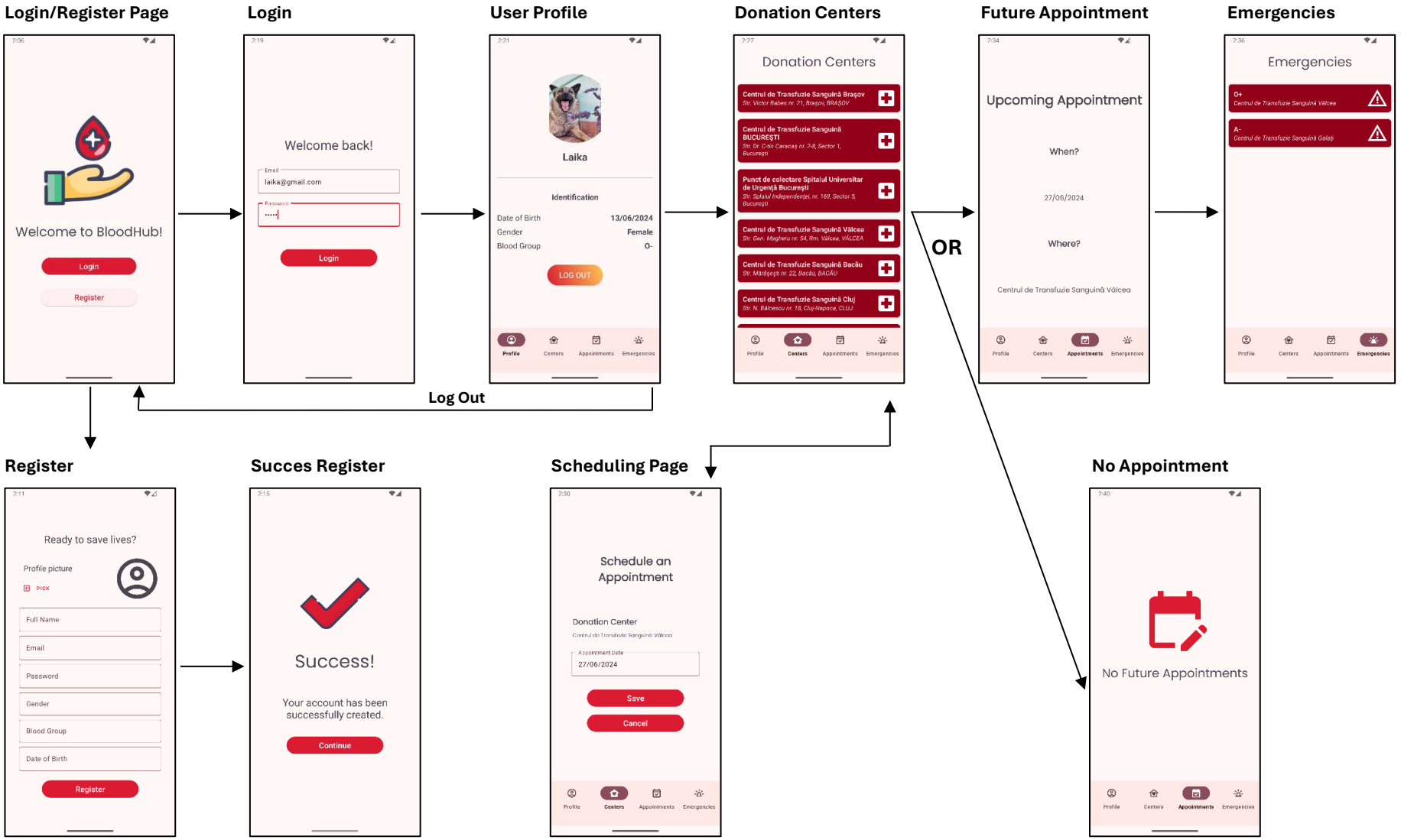
Utilizarea Coroutines în DAO

În DAOs, metodele care interacționează cu Room sunt adnotate cu suspend, permițându-le să fie apelate în contextul coroutines, fără a bloca thread-ul principal al aplicației.

Utilizarea Coroutines în ViewModel

În straturile superioare ale aplicației, cum ar fi ViewModels, coroutines sunt utilizate pentru a executa operațiile asincrone și pentru a livra rezultatele către interfața utilizator într-un mod non-blocant.

Flow-ul aplicației



- Pe primul ecran, utilizatorul poate fie **să creeze un cont nou (Register)** sau **să se autentifice (Login)**
    - La **Register**, utilizatorul specifică toate datele esențiale: nume complet, adresă de e-mail, parolă, sex, grupă de sânge, data nașterii și opțional o fotografie de profil; După ce se apasă butonul de **Register** și toate datele introduse sunt corecte și toate câmpurile au fost completate, utilizatorul este redirecționat către ecranul de **Succes Register** de unde poate mai apoi apăsa pe **Continue**, fiind redirecționat la ecranul de **Login**
    - La **Login**, utilizatorul își introduce e-emailul și parola pentru a se autentifica. Dacă acestea sunt corecte și corespund unui cont din baza de date, utilizatorul este autentificat cu succes și are acces la aplicație.
- După autentificare, primul ecran disponibil utilizatorului este cel de **Profile**, unde sunt prezente datele utilizatorului: nume, data nașterii, sex, grupă de sânge poză de profil. De asemenea, pe această pagină se regăsește și butonul de **Log Out**, care deloghează utilizatorul și aplicația revine la primul ecran.
- **Centers** conține un RecyclerView cu toate centrele de donare de sânge din baza de date. Dacă utilizatorul alege se apese pe unul dintre card-urile din RecyclerView, va fi redirecționat către pagina de creare a unei programări la acel centru. După ce se completează data la care se dorește a se face donația, utilizatorul apasă **Save**, pentru a salva programarea în baza de date și a reveni la **Centers**, sau apasă **Cancel** pentru a închide programarea și a reveni la **Centers**.
- **Appointments** conține fie următoarea programare a utilizatorului, menționând data și locația unde se va face donarea de sânge, dacă aceasta exista; Dacă nu există o donație viitoare a utilizatorului în baza de date, acesta va fi întâmpinat de o pagină care să îl anunțe de acest lucru.
- **Emergencies** conține un RecyclerView al urgențelor de crize de sânge de un anumit tip de la un anumit centru de donare. Utilizatorul poate consulta această listă înainte de a face o decizie în alegerea centrului din **Centers**.

Concluzii

Închei documentația aplicației BloodHub cu o privire asupra unei arhitecturi bine structurate și eficiente. Integrarea Room pentru gestionarea bazei de date locale și utilizarea coroutines pentru gestionarea operațiilor asincrone asigură o performanță optimă și o experiență fluidă pentru utilizatori. Firebase aduce un suport esențial pentru autentificare, stocare și gestionarea datelor în cloud, contribuind la scalabilitate și securitatea aplicației.

UI-ul construit în întregime în XML permite o personalizare detaliată a interfeței utilizatorului, asigurând o navigare intuitivă și o experiență plăcută pentru utilizatori.

Prin implementarea solidă a funcționalităților cheie precum gestionarea clinicilor, programări și situații de urgență, aplicația BloodHub demonstrează angajamentul pentru oferirea unei soluții eficiente și valoroase în domeniul sănătății.