**PROIECT LA DISPOZITIVE ȘI APLICAȚII MOBILE**

**Tema 3 - Aplicația de ridesharing MeGaCar**

**Profesor coordonator:**

**Asist. drd. Alexandru DIȚĂ**

Studenți:

**Stăncescu** Georgian-Constantin,

**Tatu** Marian.

Grupa 1087, seria E

**Obiectiv**

Scopul aplicației mobile MeGaCar constă în crearea unei platforme ai cărei utilizatori, pe de-o parte, pot să posteze curse pe care le vor face cu mașina personală, în vederea găsirii de parteneri de drum, și pe de altă parte, pot să caute persoane care călătoresc spre destinația dorită și care sunt dispuse să ofere un loc în mașina personală, contra cost.

Interesul pentru o astfel de platformă vine atât din partea șoferilor, care pot economisi din banii necesari pentru combustibil prin taxa impusă clienților, cât și din partea călătorilor, care pot ajunge la destinația dorită în condiții mai bune decât cele oferite de mijloacele de transport în comun, plătind un preț similar. Totodată, trebuie menționat impactul benefic asupra mediului, în măsura în care se reduce numărul de autovehicule aflate în circulație, dar și cel social, prin prisma faptului că un drum cu mașina poate deveni mult mai plăcut alături de persoane agreabile.

Astfel, ne-am propus să realizăm o aplicație Android care să stocheze cursele postate de utilizatori într-o bază de date la distanță Firebase și care să permită tuturor utilizatorilor vizualizarea și rezervarea acestor curse de pe propriul telefon mobil.

**Structura aplicației**

Baza aplicației este dată de clasele Utilizator și Cursă, între care există o relație one-to-many (un utilizator poate avea una sau mai multe curse). Cursele sunt împărțite în 2 categorii, prima incluzându-le pe cele în care utilizatorul este șofer și oferă loc în mașina personală, iar cea de-a 2-a include cursele în care utilizatorul este pasager și călătorește în mașina altui utilizator. Pornind de la această idee, funcționalitățile aplicației sunt distribuite în 9 activități:

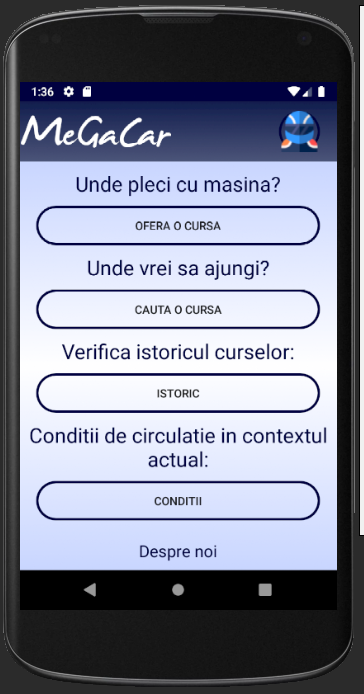
* MainActivity – sau activitatea principală, cu care utilizatorul interacționează prima dată după deschiderea aplicației.

Aceasta cuprinde un mic formular de adăugare a datelor de logare, instanțele clasei Utilizator fiind preluate dintr-o bază de date locală SQLite, prin intermediul bibliotecii Room. Utilizatorului i se dă posibilitatea de memorare a datelor de conectare prin checkbox-ul „Ține-mă minte!”, care transmite într-un fișier de preferințe conținutul edit text-urilor nume de utilizator și parolă, dar și o variabilă booleană care arată dacă acesta a fost bifat sau nu. În cazul în care nu există un cont cu informațiile completate, utilizatorul își poate crea unul prin apăsarea butonului CREARE CONT, care va deschide o nouă activitate.

* CreareContActivity – activitatea cuprinde un formular de introducere a datelor necesare pentru crearea unei instanțe de utilizator, cu validările corespunzătoare (printre acestea, nu este permisă crearea mai multor conturi cu același nume de utilizator). Apăsarea butonului CREARE CONT NOU după completarea corectă a câmpurilor duce la inserarea în baza de date a noului utilizator (care primește automat un id unic la care utilizatorul nu are acces) și la revenirea în MainActivity. Utilizatorul poate completa acum câmpurile Nume de utilizator și Parola cu noile date de logare pentru a putea trece în următoarea activitate prin apăsarea butonului INTRARE.
* ContinutPrincipalActivity – această activitate face legătura cu funcționalitățile aplicației și primește de la activitatea Main o instanță de utilizator reprezentând contul curent, identificat în baza de date prin numele de utilizator unic și parolă. Variabila utilizatorCurent va fi necesară celorlalte activități pentru diferite prelucrări, în măsura în care cursele nu pot exista independent de un utilizator.

Activitatea ia forma unui meniu cu 6 butoane, unul dintre acestea fiind un ImageButton care deschide un formular populat cu datele contului curent (icon-ul de sus, din partea dreaptă a ecranului).

* ProfilActivity – activitatea este organizată sub forma unui formular, ce permite vizualizarea informațiilor despre contul la care suntem conectați. Popularea controalelor vizuale se face prin intermediul utilizatorului curent, primit prin Intent de la activitatea ContinutPrincipal.

Dacă nu dorim să actualizăm/ștergem contul sau să ne deconectăm, putem apăsa iar pe ImageButton-ul din dreapta sus pentru a reveni la activitatea ContinutPrincipal. Altfel, pentru update putem modifica direct conținutul edit text-urilor, după care să apăsăm pe butonul ACTUALIZEAZA CONT. Update-ul în baza de date se va realiza doar dacă sunt validate toate câmpurile, inclusiv numele de utilizator, care trebuie să aibă minim 3 caractere și să fie unic. Dacă actualizarea în SQLite este efectuată fără probleme, instanța de utilizator curent (care a primit între timp noile valori ale atributelor din formular) este trimisă prin intent înapoi către ContinutPrincipal, alături de codul RESULT\_UPDATE (101 în cazul de față) pentru a înlocui vechea instanță. De menționat că fișierul de preferință primește și el noile date de logare: nume de utilizator și parolă.

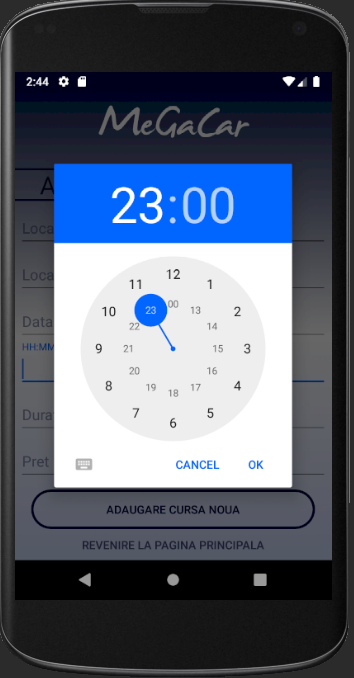
În cazul butoanelor de ștergere și deconectare, comportamentul este unul similar, în măsura în care ambele închid activitatea Profil și se întorc în ContinutPrincipal cu câte un resultCode specific (102 – delete, 103 – disconnect), pe baza cărora activitatea ContinutPrincipal se încheie și este redeschisă activitatea Main. Singura diferență evidentă apare în cadrul butonului de ștergere, care apelează un delete către baza de date pentru utilizatorul curent. Odată cu ștergerea acestuia, sunt șterse și cursele asociate lui prin atributul id\_fkutilizator (coloana id\_utilizator în tabela Curse), care este cheie externă și face legătura între cele 2 tabele, permițând totodată ștergerea în cascadă.

Revenind la activitatea ContinutPrincipal, primul buton este OFERA O CURSA, care deschide o nouă activitate de tip formular, căreia îi trimite instanța de utilizator curent.

* OferaCursaActivity – activitate de tip formular, ce permite adăugarea unei noi curse (utilizatorul se află în postura de șofer în acest caz, motiv pentru care atributul șofer al cursei adăugate va avea valoarea 1). Cursa va fi vizibilă atât în Firebase, unde poate fi găsită de utilizatorii interesați, cât și în Sqlite, unde va fi asociată cu utilizatorul prin id-ul utilizatorului curent.

Data și ora sunt completate cu ajutorul controalelor specifice DatePicker, respectiv TimePicker, deschise prin intermediul unei ferestre separate (alertDialog) la click pe edit text-ul corespunzător. Odată cu respectarea condițiilor de validare, marcate prin Toast-uri la nevoie, o nouă cursă poate fi adăugată în baza de date la click pe ADAUGARE CURSA NOUA. Vor exista 2 instanțe ale aceleiași curse, una pentru Firebase cu o structură specifică ce permite celorlalți utilizatori să intre în contact cu șoferul (cursele din Firebase includ numele și numărul de telefon ale celui care o postează) și cealaltă pentru SQLite, pe care utilizatorul o poate vizualiza în istoricul curselor proprii.

După efectuarea insert-ului, se revine în ContinutPrincipal și este afișat un Toast care anunță utilizatorul că noua cursă a fost adăugată cu succes.

* CautaCursaActivity – activitate tip formular pentru filtrarea curselor din Firebase în funcție de preferințele utilizatorului.

În acest caz, utilizatorul joacă rolul de pasager și își dorește să găsească o cursă pe o anumită rută la o anumită dată. După validare, preferințele utilizatorului sunt transmise prin intent sub forma unui obiect de tipul Cursa la apăsarea butonului de căutare. Este transmis și utilizatorul curent pentru a crea legătura între potențialele curse rezervate și contul de pe care sunt făcute rezervările.

* ListaCurseCautateActivity – activitate pentru rezervarea unei curse care respectă condițiile impuse în activitatea precedentă (CautaCursaActivity).

Cursele care corespund cerințelor utilizatorului sunt preluate din Firebase și afișate într-un listview cu adapter personalizat (imaginile corespund câmpurilor: nume prenume utilizator – șoferul care a postat cursa, punct plecare, destinație, ora plecare, durata cursei în minute, prețul în RON și numărul de telefon al șoferului), fiind ordonate crescător după oră. La click pe un element al listview-ului, acestuia i se schimbă culoarea fundalului și conținutul cursei este memorat într-o variabilă cursaSelectata. Pe baza acesteia este construită cursa ce va fi inserată în baza de date SQLite la apăsarea butonului REZERVA (de aici și necesitatea instanței de utilizatorCurent). Tot pe baza variabilei cursaSelectata este ștearsă și cursa din Firebase pentru a nu putea fi rezervată de mai multe ori. Există și o condiție care nu permite utilizatorului curent să își rezerve propriile curse.

După ștergerea din Firebase și insert-ul din SQLite, se revine la activitatea de căutare curse, din care putem căuta noi curse sau ne putem întoarce în ContinutPrincipal.

* IstoricCurseActivity – activitate ce împarte cursele asociate utilizatorului curent în curse oferite și curse rezervate, pe baza atributului sofer din clasa Cursa (acesta ia valoarea 0 dacă utilizatorul a fost pasager în cursa respectivă, deci a rezervat-o, și valoarea 1 dacă utilizatorul a fost șofer, deci a postat cursa în aplicație).

Codul din spatele acestei activități preia din baza de date SQLite o listă cu toate cursele utilizatorului curent, pe care o împarte apoi în alte 2 liste, curse oferite și curse rezervate. La click pe butoanele de jos, variabila listaCurenta preia referința listei corespunzătoare butonului ales și adapter-ul se schimbă pentru a popula listview-ul în funcție de selecție.

Există eveniment de longClick pe elementele listview-ului, în urma căruia este deschisă o nouă activitate, similară cu ProfilActivity.

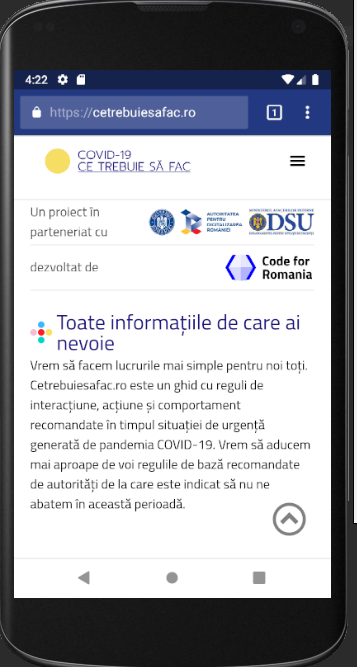
 

* EditareStergereCursaActivity – activitate pentru editarea/ștergerea cursei selectate, similară cu ProfileActivity (pentru editare/ștergere utilizator curent).

Această activitate primește prin intent cursaSelectata, controalele vizuale sunt prepopulate cu atributele ei, iar utilizatorul o poate modifica sau șterge. Decizia lui rezultă în update/delete în baza de date și în trimiterea prin intent a unui resultCode pentru update (201), însoțit de cursa modificată, respectiv de un simplu resultCode pentru delete (202). În activitatea cu istoricul curselor, este înlocuită/ștearsă cursaSelectata după caz.

În afara acestor funcționalități, în cadrul activității ContinutPrincipal există încă 2 butoane care nu trimit însă către alte activități:

* butonul CONDITII face trimitere la URL-ul: <https://cetrebuiesafac.ro>, unde utilizatorii pot afla ultimele informații oficiale cu privire la pandemia de COVID-19 și la regulile impuse pentru combaterea acesteia. (tocmai datorită contextului actual și implicit necesității de distanțare socială am proiectat aplicația astfel încât o singură persoană să poată rezerva o cursă).
* butonul DESPRE NOI deschide o fereastră (dialog) populată cu un șir de caractere parametrizat. Primul parametru este chiar numele utilizatorului curent, iar ceilalți sunt preluați dintr-un fișier JSON încărcat la o adresă URL (<https://jsonkeeper.com/b/Y8U4>) .Textul este unul informativ și oferă câteva detalii despre aplicație, creatori și contextul apariției sale, dar și date de contact ale noastre, în cazul în care utilizatorii au recomandări/observații cu privire la MeGaCar. La apăsarea butonului OK, este afișat un nou dialog, unde utilizatorul poate interacționa cu un ratingBar.



**Modul de utilizare a aplicației**

Rezumând informațiile de mai sus, MeGaCar este o aplicație de ridesharing în care utilizatorii pot adăuga/rezerva călătorii cu mașina în locații din România. Pentru a putea accesa conținutul distribuit de ceilalți utilizatori, este necesară crearea unui cont și logarea ulterioară cu numele de utilizator și parola (acestea pot fi memorate prin bifarea butonului „Ține-mă minte!”).

Odată conectat, utilizatorul se poate deconecta, poate modifica informațiile contului sau să îl șteargă apăsând pe iconița din partea dreaptă sus. Totodată, acesta poate adăuga anunțuri sub forma unor curse pentru cei interesați, oferind detalii despre călătorie precum punct de plecare, destinație, ora plecare sau preț. Pe lângă postat, el poate rezerva curse adăugate de alți utilizatori, curse regăsite în urma filtrării din formularul de căutare curse.

În secțiunea de Istoric curse, utilizatorul poate vedea toate cursele pe care le-a adăugat/rezervat și poate modifica detaliile acestora sau să le șteargă cu ajutorul unui formular deschis printr-o apăsare de lungă durată (longClick) pe cursa dorită.

Butonul de condiții deschide pagina web cetrebuiesafac.ro, unde utilizatorul se poate documenta cu privire la restricțiile impuse de pandemia de coronavirus, iar „Despre noi” deschide o fereastră cu informații generale despre aplicație și creatorii acesteia.