**Proiect Android:**

**MyTinder**

Proiect realizat de:

**Gorea Tudor-Andrei**

**Cuprins:**

1. **Introducere și obiective**................................................................
2. **Parte teoretică**…………………………………………………………………………..
   1. *Client*………………………………………………………………………………..
   2. *Server*……………………………………………………………………………….
3. **Proiectare și implementare**……………………………………………………….
   1. *Client*………………………………………………………………………………..
   2. *Server*……………………………………………………………………...……….
   3. *API*........................................................................................
4. **Bibliografie**………………………………………………………………………………...

**Introducere**

Da

**Parte Teoretică**

**Client:**

**Android Studio** este mediul de dezvoltare integrat (IDE) oficial pentru dezvoltarea de aplicații Android, bazat pe IntelliJ IDEA. Pe lângă editorul de cod puternic și instrumentele de dezvoltare ale IntelliJ, Android Studio oferă și mai multe funcții care vă îmbunătățesc productivitatea atunci când construiți aplicații Android. În aplicația client am folosit limbajul Java, limbajul clasic folosit de Android Studio.

**Java** este un limbaj de programare orientat pe obiecte care produce software pentru mai multe platforme. Când un programator scrie o aplicație Java, codul compilat (cunoscut sub numele de bytecode) rulează pe majoritatea sistemelor de operare (OS), inclusiv Windows, Linux și Mac OS. Java derivă o mare parte din sintaxa sa din limbajele de programare C și C++.

**Librăria Diolor/Swipecards** este un efect de cărți asemănător Tinder. Puteți glisa la stânga sau la dreapta pentru a aprecia sau a nu aprecia conținutul. Biblioteca creează un efect similar cu cardurile care se pot glisa de la Tinder cu animație Fling. Se ocupă de încărcarea asincronă a datelor adaptorului și utilizează aceiași parametri de aspect ca FrameLayout. Instalare: adăugați în fișierul *gradle:app*, în secțiunea *dependencies* linia:

**implementation ‘com.lorentzos.swipecards:library:1.0.9’**

apoi reîncărcați proiectul. Pentru a crea efectul, trebuie creat un layout-ul unui card. Va trebui să creăm un set de date pe care dorim să îl legăm de setul de carduri (fiecare element din set va reprezenta un card). Pentru a realiza acest lucru va trebui să creăm o clasă adaptor (care moștenește clasa ArrayAdapter<>), în care să precizăm cum dorim să legăm setul de date de layout-ul cardului. În activitatea dorită vom crea un obiect de tip *SwipeFlingAdapterView*, să îi dăm id-ul la containerul din activitate pe care îl va înlocui, să creem setul de date și să instanțăm adaptorul creat de noi, dându-i setul de date, apoi să îl setăm ca adaptor pentru obictul de tip *SwipeFlingAdapterView*. În cele din urmă vom lega un FlingListener, dar va trebui să suprascriem metodele:

**removeFirstObjectInAdapter()**  - care descrie ce se va întâmpla când cardul de deasupra este dat la o parte (aici se scoate primul element din setul de date și se notifică adaptorul că setul de date a fost schimbat)

**onLeftCardExit()** - care descrie ce se va întampla când cardul va fi scos la stânga (în cazul aplicației noastre, va fi ignorată persoana)

**onRightCardExit()** - care descrie ce se va întampla când cardul va fi scos la dreapta (în cazul aplicației noastre, vom da like la persoana și se va trimite request către server să salveze faptul că am dat like la acea persoană)

**onAdapterAboutToEmpty()** - care descrie ce se va întampla când se termină setul de elemente. Am observat un bug în care se apelează această metodă fără ca setul de date să se fi terminat sau să fi fost aproape de a se termina

**onScroll()** - metoda care se apelează la orice fel de interacțiune cu cardul

**Librăria Bottom Navigation View** – Este o librărie de la Google care oferă posibilitatea de a crea o bară de navigare in partea de jos a unei activități. Instalare: adăugați în fișierul *gradle:app*, în secțiunea *dependencies* linia:

**implementation ‘com.google.android.material:material:1.2.0-alpha04’**

apoi se reîncarcă proiectul. Creăm în folderul *res* un folder nou (*Android Resource Directory*), selectăm Resource type “menu”, numim folderul ”*menu*”, apăsăm ok. În folderul creat creem un nou *Resource file*, numit *bottom\_nav\_menu*, în care adăugăm iteme, sub forma:

**<item android:id=”@+id/nume\_item”**

**android:title=”titlu”**

**android:icon=”@drawable/nume\_iconita”/>**

În activitatea dorită vom adăuga un FrameLayout, care va fi containerul ce se va modifica la apăsarea unui buton din bara de navigare. Mai adăugam și bara de navigare și îi vom adăuga proprietatea **app:menu=”@menu/*bottom\_nav\_menu*”**. Creem fragmentele corespunzătoare fiecărui buton din bară, iar legătura dintre buton și schimbarea fragmentului se face instanțând clasa *BottomVavigationView* și apelând metoda **setOnItemSelectedListener()**, în care descriem ce fragment se va afișa în locul containerului.

**Server:**

**.NET** este o platformă pentru dezvoltatori open source, multiplatformă, gratuită, pentru construirea multor tipuri diferite de aplicații. Cu .NET, se pot utiliza mai multe limbaje de programare, editori și biblioteci pentru a crea aplicatii web, mobile, desktop, jocuri și IoT.

**C#** este un limbaj de programare *strongly-typed*, *orientat pe obiecte*. C# este *open-source*, simplu, modern, flexibil și versatil.

**Microsoft.Data.Sqlite** este un furnizor lightweight ADO.NET pentru SQLite. Furnizorul Entity Framework Core pentru SQLite este construit pe deasupra acestei biblioteci. Cu toate acestea, poate fi folosit și independent sau cu alte biblioteci de acces la date. În cazul aplicației noastre, va fi folosită varianta ADO.NET în locul EntityFramework, deoarece oferă accesul direct la cum vom modela baza de date. Problema în EF este că nu putem salva într-un model o listă de elemente (exemple testate: listă de numere, sau de presoane prietene). Instalarea se face cu comanda **Install-Package Microsoft.Data.Sqlite** înăuntrul consolei din Visual Studio.

**Proiectare și Implementare**

**Client:**

În momentul în care intră in aplicație utilizatorul se află în **LogInActivity**, și trebuie să se autentifice. Dacă nu are un cont va putea apăsa pe textul “*No Account? Press here to register*” pentru a ajunge în activitatea de înregistrare a contului (*RegisterActivity*). În momentul în care utilizatorul apasă pe butonul **Sign In**, atunci se va crea un Thread nou în care se compune un mesaj către server alături de mail-ul și parola scrise în câmpurile activității, iar serverul va răspunde *OK* alături de datele utilizatorului dacă găsește mail-ul și parola în baza de date, sau *ERR* dacă datele trimise sunt incorecte. În cazul răspunsului *OK*, se va salva utilizatorul curent, și se va face trecerea la activitatea *SelectActivity*.

În **RegisterActivity**, utilizatorul trebuie să completeze corect câmpurile pentru crearea contului. Câmpurile nu pot fi goale.

* Câmpul pentru mail trebuie să fie de forma: **“[a-zA-Z][a-zA-Z0-9]\*@gmail\\.com”**.
* Câmpurile pentru parole trebuie să aibă o parolă între 4 și 20 de caractere, iar câmpul pentru confirmarea parolei trebuie să conțină exact aceeași parolă.
* Câmpul pentru nume trebuie să conțină numele complet și să fie între 5 și 30 de caractere.
* Câmpul pentru vârstă trebuie să conțină un număr intre 18 și 100.
* La partea de alegere a genului, utilizatorul va putea alege unul dintre butoanele radio, dar in mod implicit este ales butonul pentru ascunderea genului.

În momentul în care câmpurile sunt corect completate și utilizatorul apasă pe butonul “*Register*”, se va crea un Thread nou și se va transmite un mesaj către server, alături de datele din câmpuri, iar serverul va răspunde cu *OK* dacă a fost creat un cont nou, sau *ERR* dacă a apărut vreo eroare sau contul deja există. Pentru a ieși din această activitate, utilizatorul va trebui să apese pe butonul “*înapoi*”, reîntorcându-se în *LogInActivity*.

În **SelectActivity** utilizatorul poate să aleagă vârsta minimă și maximă și genurile persoanelor pe care vrea să le aleagă. În momentul în care utilizatorul apasă pe butonul “*Search*”, se va crea un thread nou care va trimite un mesaj către server, alături de filtrele date, iar serverul va trimite *OK* împreună cu primele *n* persoane (ex: n=20, dar se hardcodează valoarea) care respectă filtrul, *INV* dacă vârsta minimă este mai mare strict decât vârsta maximă sau nu a fost selectat niciun gen pentru căutare, *ERR* pentru altfel de erori. În cazul răspunsului *OK*, se vor salva utilizatorii primiți într-o listă și se vor transmite la *MainActivity*. La apăsarea butonului “*înapoi*” utilizatorul va fi dus înapoi pe activitatea *LogInActivity*.

În **MainActivity** se inițializează cele 3 fragmente (*HomeFragment, MatchesFragment, ProfileFragment*), și se setează primul fragment ca fiind *HomeFragment* în locul containerului activității. În partea de jos se află bara de navigare în aplicație. Bara de navigație conține 3 butoane: *Home, Matches* și *Profile*, corespunzătoare fragmentelor de mai sus. La apăsarea butonului “*înapoi*”, utilizatorul se va întoarce la *SelectActivity*.

În **HomeFragment** se încarcă și apar cardurile cu persoane. Când dăm swipe la stânga vom ignora persoana de pe card, iar când dăm swipe la dreapta se creeaza un nou Thread în care se transmite un mesaj către server, alături de mail-ul utilizatorului curent și mail-ul persoanei de pe cardul tocmai scos; nu se așteaptă răspuns de la server.

În **MatchesFragment** se pot vizualiza persoanele cu care utilizatorul curent se află în relație de “*match*”. O relație de match apare atunci când utilizatorul curent a dat like (sau swipe-right) la o persoană, iar acea persoană a dat like la utilizatorul curent.

**Server:**

aaaa

**API:**

Comunicarea dintre client și server este foarte importantă în această aplicație.

În comenzile de mai jos se pot observa **c!** și **s!**, care reprezintă cine a transmis mesajul (c=client, s=server).

Caracterul **#** este folosit pentru a despărții comenzile de parametrii lor și de spațiul liber rămas din socket.

Am marcat cu culori diferite tipurile de dată în care trebuie să fie datele trimise împreună cu comanda astfel:

**comandă**

**șir de caractere (string)**

**valoare numerică (1, 2, 3, .....)**

**tip de dată enum (nope / male / female / other)**

**valoare booleană (true / false)**

API-urile folosite in proiect sunt:

**c!test\_add#** - verifică dacă se pot efectua operații de introducere în baza de date pe un tabel de teste (*test\_table*).

**c!test\_get#** - verifică dacă se pot efectua query-uri pe baza de date pe un tabel de test (*test\_table*).

**c!login\_req#*mail*#*pwd*#** - clientul trimite o cerere de login, alături de mailul și parola (parola sub formă de hash) către server, așteptând unul din răspunsurile:

**s!login\_ok#*mail,pwd,fullname,age,gender*#** - serverul trimite răspuns că totul a decurs în regulă (contul există și parola este bună), alături de toate datele utilizatorului,

**s!login\_err#** - serverul trimite mesaj de eroare dacă contul nu există sau parola nu este bună, sau alte cauze, fară a specifica eroarea.

**c!reg\_req#*mail,pwd,fullname,age,gender*#** - clientul trimite o cerere de înregistrare cont, împreună cu datele de înregistrare (clientul trebuie să se asigure că parola este hash-uită înainte de a fi trimisă), așteptând unul din răspunsurile:

**s!reg\_ok#** - dacă contul a fost adăugat cu succes în baza de date,

**s!reg\_err#** - dacă există deja un cont cu același mail sau apar erori la comunicarea cu baza de date.

**c!getP\_req#*ageMin*#*ageMax*#*sMale*#*sFemale*#*sOther*#** - clientul trimite o cerere că dorește să primească o listă de persoane, care au vârsta între *ageMin* și *ageMax*, iar valorile booleene reprezintă ce genuri să apară în rezultatul căutării.

(ex: sMale=true, sFemale=true, sOther=false - serverul va returna doar persoane cu genul *female* și *nope*(ascuns),

sMale=true, sFemale=false, sOther=true - serverul va returna doar persoanele cu genul *male*, *other*(non-binar, sau orice altceva) și *nope*)

Serverul va trimite unul din răspunsurile:

**s!getP\_inv#** - dacă *ageMax* < *ageMin* sau toate cele 3 valori booleene sunt false,

**s!getP\_err#** - dacă există erori in la baza de date

**s!getP\_ok#*mail1,fullname1,age1,gender1*$*mail2,fullname2,age2,gender2*$...#** – serverul va trimite un mesaj că totul a mers bine, alături de o listă cu un număr maxim de persoane (ex: primele 20 de persoane) care respectă filtrele date. Persoanele din listă vor fi despărțite de caracterul **$**.

**c!like\_set#*mail1*#*mail2*#** - clientul trimite un request la server să salveze în baza de date că user-ul cu mail=*mail1* i-a dat like la user-ul cu mail=*mail2*. Clientul nu așteaptă niciun răspuns de la server în legătură cu această operație.

**Bibliografie:**

* [Android Fundamentals: https://www.youtube.com/playlist?list=PLQkwcJG4YTCTq1raTb5iMuxnEB06J1VHX](https://www.youtube.com/playlist?list=PLQkwcJG4YTCTq1raTb5iMuxnEB06J1VHX)
* [Swipe cards - tutorial: https://www.youtube.com/watch?v=SJW\_4UMXbu8&ab\_channel=SimCoder](https://www.youtube.com/watch?v=SJW_4UMXbu8&ab_channel=SimCoder)
* [Swipe cards - library: https://github.com/Diolor/Swipecards](https://github.com/Diolor/Swipecards)
* [Bottom navigation: https://www.youtube.com/watch?v=AL\_1UDa9l3U&list=PLQkwcJG4YTCTq1raTb5iMuxnEB06J1VHX&index=24&ab\_channel=PhilippLackner](https://www.youtube.com/watch?v=AL_1UDa9l3U&list=PLQkwcJG4YTCTq1raTb5iMuxnEB06J1VHX&index=24&ab_channel=PhilippLackner)
* [Android documentation: https://developer.android.com/docs](https://developer.android.com/docs)