UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

`

Lucrare de licență

Aplicație Android pentru exercițiile fizice în kinetoterapie

COORDONATOR:
Conf. Dr. Stupariu Mihai-Sorin

ABSOLVENT:
Schipor Ioana-Teodora

BUCUREȘTI, ROMÂNIA
2021

REZUMAT

Lucrarea de față constă într-o aplicație Android destinată persoanelor cu dezechilibre posturale ce doresc corectarea lor.

Se dorește ca prin intermediul acestei aplicații să se aducă la cunoștință utilizatorilor despre predispoziția lor la anumite disfuncții ale poziției corpului și să se susțină un stil de viață sănătos.

În primul rând, aplicația necesită autentificarea utilizatorului pentru a reține datele despre diagnostic și evoluție ce urmează să îi fie atribuite pe parcursul folosirii aplicației.

Se impune folosirea camerei pentru a analiza postura corpului utilizatorului prin intermediul „Google Vision Pose Detection”, API. Acesta determină în timp real coordonatele x, y, z pentru un set de treizeci și trei de puncte ale corpului, precum și probabilitatea ca acestea să se afle în cadru. Astfel, cu ajutorul unor algoritmi implementați în acest proiect, se pot calcula unghiurile dintre punctele de interes major și se stabilește diagnosticul utilizatorului. Acesta din urmă beneficiază și de o pagină în care se înregistrează progresul și de seturi de exerciții ce sunt reprezentate printr-un model 3D animat.

Astfel, această aplicație ușor de folosit reușește să identifice deficiențele fizice și să facă sportul și corectarea lor mai plăcute.

SUMMARY

This paper consists of an Android application for people with postural imbalances who want to correct them.

The aim of this application is to make users aware of their predisposition to certain body position dysfunctions and to support a healthy lifestyle.

Firstly, the application requires the authentication of the user to retain the diagnostic and evolution data that will be assigned to him while using the application.

It is necessary to use the camera to analyze the posture of the user's body through the Google Vision Pose Detection API. It determines in real time the x, y, z coordinates for a set of thirty-three points of the body, as well as the probability that they are in the frame. Thus, with the help of algorithms implemented in this project, the angles between the points of major interest can be calculated and the user's diagnosis is established. The latter also benefits from a page where progress is recorded and sets of exercises that are represented by an animated 3D model.

Thus, this easy-to-use application manages to identify physical deficiencies and make sports and their correction more enjoyable.

CUPRINS

1. Introducere

Știința, tehnologia și informatizarea au pătruns în toate domeniile vieții omului modern, drept urmare i-au ușurat foarte mult munca, crescând productivitatea pe de o parte, iar pe de altă parte au dus tot mai mult la diminuarea efortului fizic în toate aspectele vieții de zi cu zi, sprijinind tot mai mult un mod de viață confortabil, dar cu risc de sedentarism. Acest fenomen generează o serie de afecțiuni din cauza lipsei mișcării și a activităților fizice.

Toate acestea cresc riscul dezvoltării unei posturi incorecte a corpului, producând disconfort fizic, reducând calitatea vieții și generând numeroase afecțiuni, precum: deteriorarea articulațiilor, mărirea nivelului de oboseală și solicitare psihică, îngreunarea respirației.

Confruntându-mă în ultimii doi ani cu disfuncții posturale și parcurgând mai multe etape de corectare a posturii corpului, fiind nevoită să fac o serie întreagă de exerciții pentru aceasta, am dorit ca această aplicație să fie destinată tuturor persoanelor cu afecțiuni ale posturii până la un anumit grad și care doresc o examinare a poziției corpului și executarea de exerciții fizice rectificative.

Am luat în considerare și cazul în care o persoană a urmat un tratament kinetoterapeutic supravegheat într-un centru specializat și care din varii motive nu mai are posibilitatea de a continua, dar dorește reluarea exercițiilor pe cont propriu având un model.

O altă categorie pe care am avut-o în vedere este cea a persoanelor care vor să se inițieze într-un anumit sport sau chiar a sportivilor de performanță unde mai întâi este necesară o analiză a posturii corpului pentru prevenirea dezechilibrelor musculare, ce în timp duc la deteriorări ale sistemelor muscular și osos.

Am încercat implementarea unei aplicații pentru a veni în sprijinul acestor persoane cu deficiențe posturale.

Analizând opțiunile de care dispunem pentru a dezvolta acest proiect, am realizat că aplicațiile mobile au preluat controlul datorită performanțelor și accesibilității lor. La ora actuală, fără îndoială, există mai mulți utilizatori Android decât utilizatori iOS, deoarece telefoanele mobile Android sunt mai convenabile din punct de vedere al costurilor și sunt foarte ușor de utilizat. Datorită accesibilității și capacității sale de a ajunge la un public cât mai larg din întreaga lume, am ales ca acest proiect să fie o aplicație Android.

În această aplicație, analiza posturii se realizează pe principiul simetriei corpului, așadar se evaluează alinierea segmentelor și a centrului de greutate, iar exercițiile fizice sunt reprezentate de animații 3D ale unui model.

Acest proiect este o abordare nouă. Toate aplicațiile lansate până acum în domeniul sport-sănătate se îndreaptă în zona fitness și nu pe cazul concret al utilizatorului.

Această aplicație este una “prietenoasă”, ușor de utilizat, menită să combată o parte a efectelor secundare ale modului de viață a omului modern.

2. Structura aplicației

2.1. Instrucțiuni de utilizare a aplicației

Pentru început, la deschiderea aplicației utilizatorul va întâlni pentru trei secunde un ecran cu sigla aplicației, continuând cu o serie de instrucțiuni ce îl ajută să se familiarizeze cu aplicația. Acestea se pot accesa oricând și din meniu, pentru a ajuta utilizatorul în cazul unui obstacol întâlnit pe parcursul folosirii aplicației.

2.2. Autentificare

Următorul ecran este cel de autentificare ce apare atunci când aplicația este deschisă pentru prima oară. Acesta conține patru câmpuri ce trebuie completate: „Full Name”, „Age”, „Email”, „Password”. Utilizatorul finalizează crearea contului accesând butonul „Register User”, și este redirecționat pe pagina de logare.

2.3. Logare

Logarea conține câmpurile de „Email”, și „Password”. În cazul în care aceasta se efectuează pentru prima dată, are loc verificarea adresei. Utilizatorul mai beneficiază aici și de opțiunea „Forgot Password”, ce trimite un mesaj pe adresa sa de email pentru a-și putea schimba parola.

2.4. Meniu

După ce acești pași au fost parcurși, utilizatorul este redirecționat la meniu. Acesta este alcătuit din cinci butoane: „Open Camera”, „Progress”, „Exercises”, „Tutorial” și „Log Out”.

2.4.1. Cameră

Prin accesarea „Open Camera”, se cere în primul rând permisiuni pentru accesarea camerei și memoriei dispozitivului. Apoi, după acceptarea acestora se deschide camera. Pentru funcționarea cât mai precisă a aplicației este indicat ca dispozitivul să aibă o poziție fixă, iar utilizatorul să se îndepărteze astfel încât să fie vizibil complet în cadru. Pentru ca timpul să fie îndeajuns, există și un temporizator de cinci secunde ce pornește după deschiderea camerei, iar după terminarea sa, se va face o fotografie. Aplicația o va analiza și va da un diagnostic detaliat ce constă în gradul de severitate al afecțiunii posturii corpului, masurile unghiurilor și imaginea pe care s-au realizat toate

acestea. De aici utilizatorul este trimis explicit printr-un buton la exercițiile destinate afecțiunii sale cu procentajul cel mai mare.

2.4.2. Exerciții fizice

Această pagină este împărțită în afecțiuni, fiecare având destinat câte un buton. După selectarea preferinței, începe antrenamentul corespunzător, îndrumat de un caracter 3D.

2.4.3. Progres

Această pagină este un mod de a ține evidența și de a vedea progresul utilizatorului de-a lungul timpului. Aceasta afișează fotografiile făcute pe parcurs pe care s-au măsurat dezechilibrele, împreună cu data și detaliile diagnosticului.

2.4.4. Tutorial

Prin accesarea acestui buton se afișează setul de instrucțiuni apărut la început pentru o reîmprospătare a memoriei.

2.4.5. Delogare

Prin accesarea acestui buton, utilizatorul se deloghează și este redirecționat la pagina de „Log In”.

3. Tehnologii utilizate

Realizarea acestui proiect a constituit îmbinarea mai multor tehnologii. Utilizarea lor este descrisă în cele ce urmează.

3.1. Blender

Blender este utilizat pentru crearea de filme animate, efecte vizuale, realitate virtuală, aplicații interactive 3D, fiind un set de instrumente de grafică 3D pentru computer. Blender include modelare 3D, texturare, rigging, animare și alte caracteristici [1].

Am utilizat Blender în acest proiect pentru a produce o animație, cu un model 3D ce reprezintă un îndrumător al utilizatorului în antrenamentul de kinetoterapie ce va fi prezent în secția „Exercises” din meniul aplicației.

3.1.1. Schimbarea trăsăturilor caracterului

Pentru a facilita procesul de creare a modelului, am preluat de pe site-ul oficial Blender un caracter deja proiectat și rigged, cu alte cuvinte cu oase asigurate pentru procesul de animare ulterioară. Am utilizat, deci modelul Rain v2.0, o versiune actualizată cu proprietăți noi, utile proiectului.

Pentru această lucrare, i-am adus câteva modificări cu scopul de a face să arate cât mai pregătit pentru o sesiune de sport, și anume, am eliminat anumite piese vestimentare schimbând valoarea parametrului lor în 0. Alte schimbări pe care i le-am aplicat modelului sunt repoziționarea părului astfel încât aspectul să fie cât mai real și fixarea pupilelor pentru a evita mișcarea lor imprevizibilă după punctul de reper vizual atunci când va fi animat.

3.1.2. Animarea modelului 3D

Animarea modelelor 3D este o metodă în care acestea sunt manipulate pentru a apărea ca imagini în mișcare. Ea se poate realiza doar dacă caracterele îndeplinesc condiția de „rigged” deoarece aceasta permite mișcarea și poziționarea scheletului furnizat.

Pentru început, tipul editorului folosit este „3D Viewport” în modul „Pose”, unde se poate vedea din diferite unghiuri modelul împreună cu armătura sa. Se adaugă o nouă fereastră cu editorul pentru animații „Timeline” ce oferă o axă a timpului. Aceasta ajută la identificarea cadrului curent din animație.

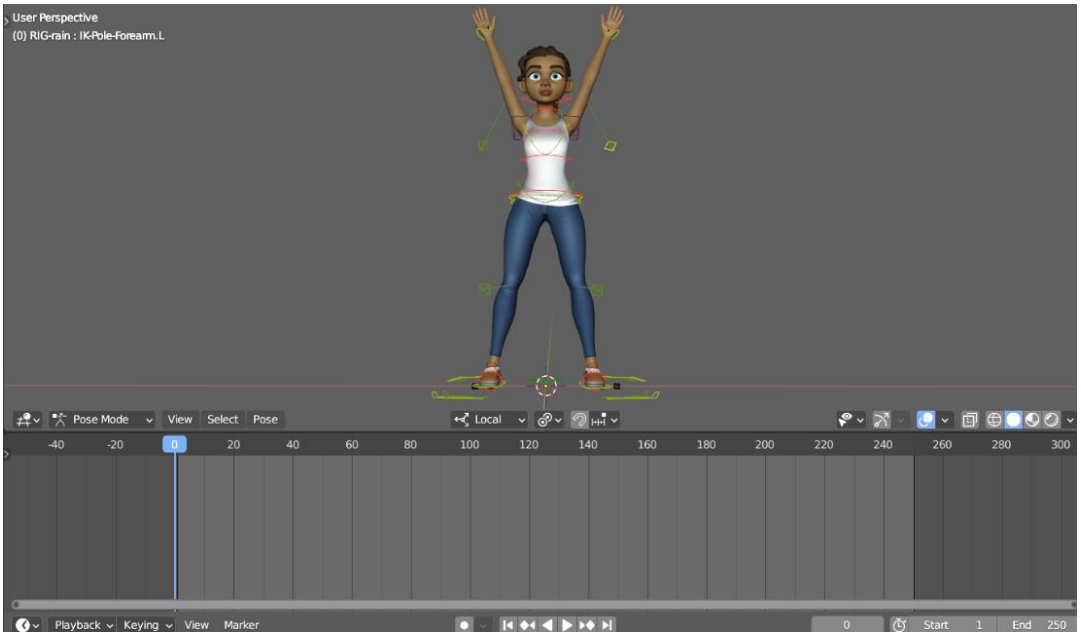

Figura 3.1: Realizarea animațiilor în Blender

Am reușit modificarea poziției caracterului pentru fiecare exercițiu, cu ajutorul tastelor „G”, „R” și „S” ce se folosesc pentru mișcare, rotație, respectiv scalare, aplicate pe articulații. Fiecare transformare a fost salvată pe axa timpului cu opțiunea „Location & Rotation”.

În mod implicit în Blender, pe măsură ce animația progresează între două cadre cheie, se observă că pornește lent, accelerează la mijloc și se oprește lent, așadar, toate acestea se întâmplă pe o curbă liniară. Pentru o editare mai avansată a mișcării și vitezei, am schimbat tipul editorului în „Graph Editor, unde se regăsește graficul animației [1, „Editors - Graph Editor - F-Curves - Introduction”]. Se pot obține o mulțime de detalii pentru a face animația să pară cât mai reală, doar modificând curbele graficului folosind punctele de control, însă în acest proiect am folosit acestea cu alt scop. Pentru a schița întâi părțile importante ale animației, am schimbat modul de interpolare din curbă în constant, pentru a obține o „blocare” între poze. Abia după stabilirea acestor piloni, am revenit la modul implicit.

Nu a fost nevoie de alte editări mai detaliate ale vitezei deoarece, pentru ca acest model să fie un exemplu corect de urmat în exercițiile kinetoterapeutice, mișcările trebuie executate lent, fără opriri bruște pentru a preveni orice fel de accidentare.

Fiecare animație corespunzătoare fiecărui exercițiu din antrenament a fost exportată drept fișier de tipul „.fbx” pentru a putea fi folosit în continuare în aplicația noastră Android.

3.2. Android Studio

"Android Studio este mediul oficial de dezvoltare integrată pentru sistemul de operare Android" [2], ce ajută la construirea aplicațiilor native Android. Limbajele de programare ce se pot utiliza pentru a realiza astfel de aplicații sunt „Kotlin" și „Java", acesta din urmă fiind varianta aleasă pentru proiectul de față.

La baza unei astfel de aplicații stau mai multe componente importante:

3.2.1. Activități

O primă componentă, fără de care aplicația nu ar putea exista, este activitatea. Aceasta reprezintă interacțiunea utilizatorului cu aplicația. Sistemul inițiază codul într-o instanță de activitate, apelând metode specifice ciclului său de viață [2, „Introduction to Activities"]. Pentru a putea fi utilizate, activitățile se declară în fișierul „AndroidManifest.xml".

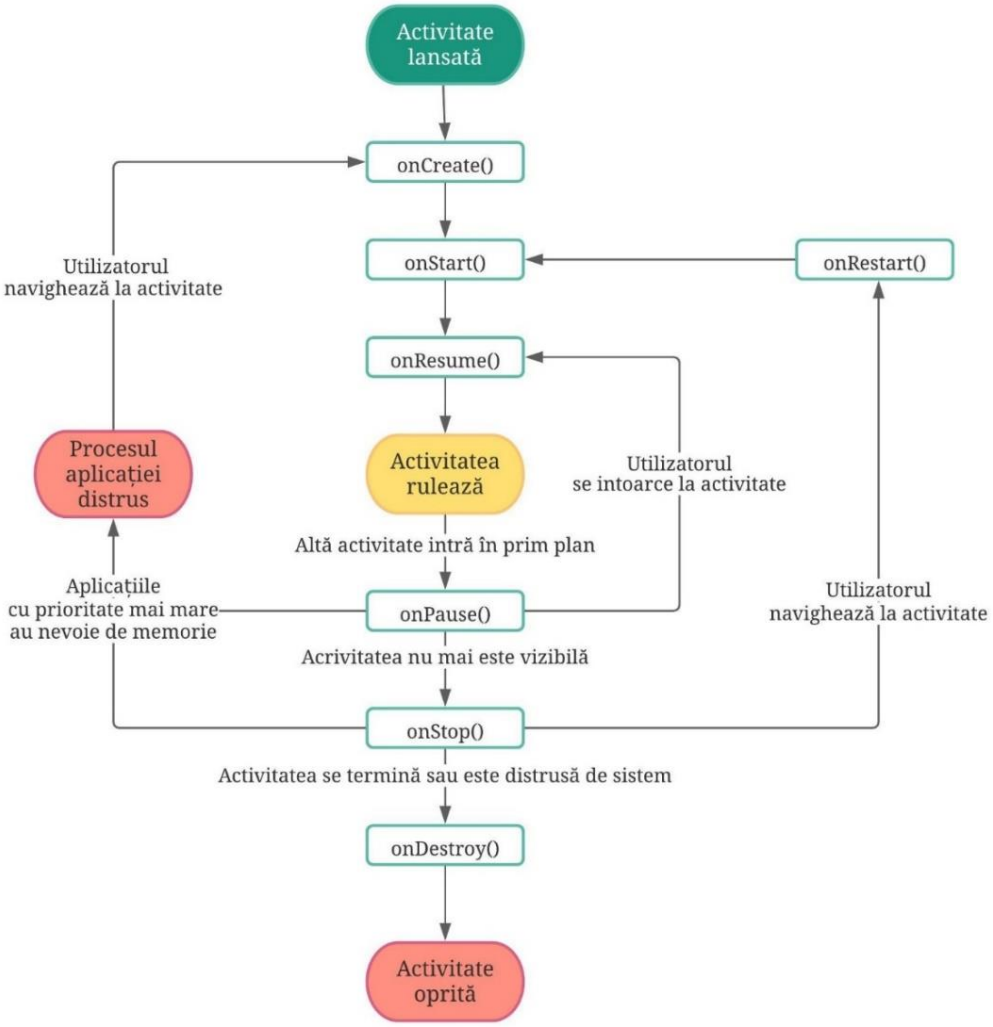

Figura 3.2: Ciclul de viață al unei activități

Înțelegând ciclul de viață al unei activități, putem continua cu navigarea între activități. Pentru a chema o altă activitate utilizăm elementului „Intent”. Utilitatea acestuia este de a chema alte componente principale ale aplicației (cele adăugate și in manifest), cum ar fi activitățile și serviciile. Aici intervine noțiunea de „Back Stack”. Aceasta constă într-o stivă în care se adaugă activitățile în ordinea deschiderii lor, având și o caracteristică importantă, aceea de a permite întoarcerea la o altă activitate.

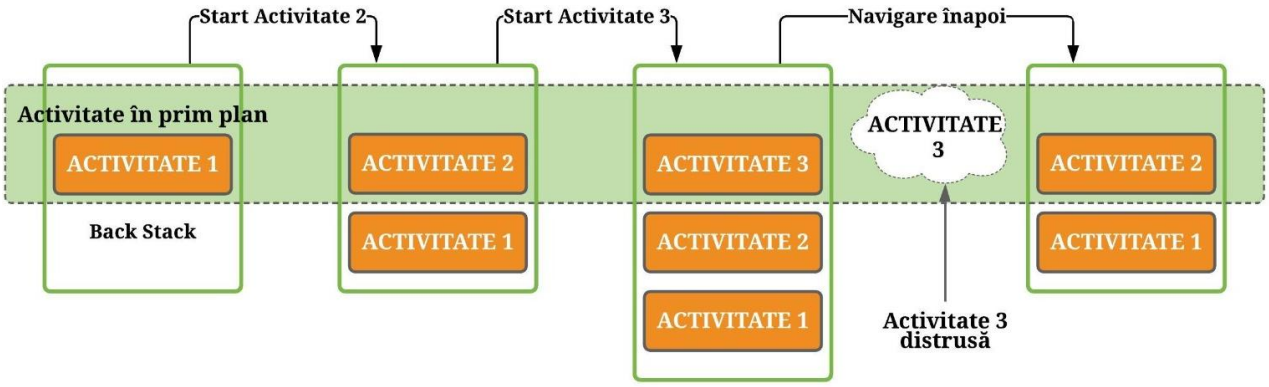

Figura 3.3: Back Stack

Interactivitatea aplicației de față este bazată pe o varietate de activități, așadar am utilizat toate aceste concepte descrise până acum. Exemple de activități implementate:

- „MainActivity” este activitatea de început ce afișează sigla aplicației, ce se termină după trei secunde și inițiază activitatea „Register” sau „LogIn”, după caz. „MainActivity” este declarată în fișierul manifest cu acțiune de tip „MAIN”, cu alte cuvinte, atunci când aplicația este lansată, această activitate este creată.
- „RegisterActivity”, „LogInActivity” și „ForgotPasswordActivity” sunt folosite pentru operațiile legate de autentificarea utilizatorului.
- „CameraActvity” este lansată odată cu accesarea „Open Camera” din meniul aplicației și reprezintă ecranul cu imaginile luate de camera dispozitivului.
- „ExercisesActivity” constituie activitatea lansată de accesarea butonului „Exercises”, ce inițiază animațiile 3D.

La inițializarea fiecărei activități este necesară apelarea funcției „setContentView(view)” pentru a-i defini interfața.

3.2.2. Interfață vizuală

Interfața vizuală este cea pe care o vede utilizatorul și cu care poate interacționa într-o aplicație. Aceasta se alcătuiește din diverse componente, cum ar fi obiecte de tip „layout”, meniuri, notificări și altele [2, „User Interface & Navigation”].

Pentru structurarea interfeței proiectului, am utilizat obiecte „layout” pentru toate activitățile și fragmentele. Acestea sunt construite ca o ierarhie de „ViewGroups” și „Views”, „ViewGroups” fiind containere care țin sub control modul în care vizualizările sunt dispuse pe ecran, iar acestea din urmă fiind elemente precum butoane, câmpuri de completare de tip text, imagini, texte, liste dinamice („Button”, „EditText”, „ImageView”, „TextView”, „RecyclerView”). Acestea sunt preluate apelând funcția „findViewById(int)” pentru a putea interacționa apoi programatic cu ele.

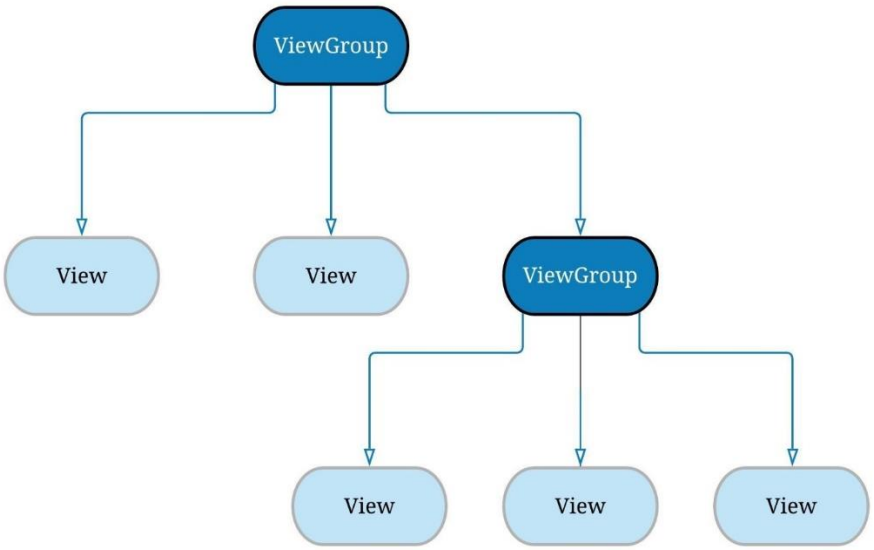

Figura 3.4: Ierarhia „ViewGroups” și „Views”

Pentru a obține o interfață prietenoasă am utilizat o varietate de culori, mărimi și așezări în pagină ale vizualizărilor, precum și proprietatea „elevation” pentru unele elemente pentru a crea umbră și pentru a ieși în evidență și View-ul „ProgressBar” care apare atunci când se încarcă o pagină cerută de utilizator.

3.2.3. Fragmente

Fragmentele sunt porțiuni de interfață ce se pot refolosi, ce nu pot trăi singure, astfel încât trebuie găzduite de o activitate sau de un alt fragment. În acest proiect am folosit un fragment

pentru afișarea paginii „Progress”. Acest fragment este apelat de „ProgressActivity” și se folosește la rândul lui de un „Adapter”. Acesta din urmă este definit ca fiind un obiect ce primește un set de date, oferă acces la elementele respective, totodată realizând „Views” pentru fiecare dintre ele. Astfel pot prelua datele din entitatea „Diagnostic”, iar prin intermediul elementelor descrise în această secțiune le afișez adecvat.

3.2.4. Handler

„Handler” permite transmiterea și procesarea obiectelor „Message” și „Runnable”, acestea fiind asociate cu coada de mesaje „MessageQueue” a unui thread (fir de execuție). Fiecare instanță „Handler” este asociată cu un singur fir de execuție și coada de mesaje a acelui fir de execuție [2, „Handler”].

Am utilizat un „handler” cu un „runnable” în „CameraActivity” pentru a implementa temporizatorul ce durează cinci secunde. Cu ajutorul metodei „postDelayed()”, ce face ca „runnable” să fie adăugat la coada de mesaje pentru a fi rulat după scurgerea timpului specificat ca parametru, am reușit ca textul corespondent temporizatorului să fie rescris la fiecare secundă. Am programat un al doilea „runnable” ce se execută după cinci secunde pentru ca textul să dispară și să se facă fotografia.

3.2.5. SceneForm

„Sceneform” este o librărie ce face posibilă redarea scenelor 3D. Am utiliza-o în această aplicație pentru a reda animațiile 3D realizate în Blender, sub forma unei scene „AR”.

În primul rând, am adăugat toate dependențele și am instalat un „plugin” necesar pentru a folosi această librărie, apoi am importat animațiile pe care le folosim în acest proiect.

O primă funcționalitate pe care am implementat-o este detectarea suprafeței din scenă, pe care poate fi poziționat modelul. De fiecare dată când scena este actualizată, utilizez un „Frame” ce reprezintă cadrul curent, iar în conținutul acestuia căutăm un obiect de tip „Plane” (plan), ce poate fi utilizat. Pentru planul detectat, creăm un obiect „Anchor” pentru a fixa locația unde se amplasează modelul.

Am poziționat modelul creând un obiect „ModelRenderable” căruia i-am trimis animația noastră, apoi am creat un nod „SkeletonNode” ce conține obiectul anterior.

Animația începe după ce accesăm butonul de pe ecran cu textul „Exercise”. Am realizat aceasta cu ajutorul metodelor unui obiect „ModelAnimator” ce primește „ModelRenderable” deja creat, pentru a controla progresul și starea animației și pentru a actualiza animația.

Deoarece o animație este proiectată executând doar o repetare a exercițiului, am setat numărul repetărilor necesare cu ajutorul funcției „setRepeatCount(repeatCount) ”. După finalizarea unui exercițiu, se poate începe următorul exercițiu prin accesarea butonului de pe ecran, căruia după executarea primei animații i-am schimbat textul, pentru a fi mai clar, în „Next exercise” prin „setText()”.

3.2.6. CameraX

„CameraX” este o librărie ce facilitează dezvoltarea aplicațiilor ce folosesc camera [2, „CameraX overview”]. Am utilizat-o în două modalități în acest proiect și anume: pentru primirea unui flux de imagini continuu, în timp real, și pentru captarea și salvarea în memorie a unei imagini. Pentru început am integrat cererea permisiunilor folosirii camerei și salvarea imaginii, ce apare doar atunci când utilizatorul folosește aplicația pentru prima oară. Aceste permisiuni au fost introduse in fișierul „AndroidManifest.xml” și implementate în activitatea „CameraActivity” cu ajutorul unui „ListenableFuture”. Apoi am adăugat un „PreviewView” care va afișa imaginile pe ecran atunci când camera devine activă. Pentru stabilirea camerei folosite (frontală „LENS_FACING_FRONT” sau din spate „LENS_FACING_BACK”), am folosit un obiect de tip „CameraSelector”.

„MyImageAnalyzer” este implementată în această aplicație drept o clasă ce implementează „ImageAnalysis.Analyzer” care este o interfață a librăriei „CameraX”, utilizată cu scopul de a procesa o imagine. Aceasta implementează o metodă de analiză care se execută pe fiecare cadru [2, „Analyze images”]. „MyImageAnalyzer” este instanțiată în „CameraActivity”, iar prin „imageAnalysis.setAnalyzer(image, analyzer)” se analizează imaginea apelând funcția implementată ce se ocupă cu „Pose Detection”.

Pentru a gestiona fluxul de cadre ce îl primim, am utilizat un „ExecutorService” care oferă metode ce urmăresc progresul uneia sau mai multor sarcini asincrone. La început am folosit metoda „newSingleThreadExecutor(”) ce creează un singur fir de execuție care operează pe o coadă nelimitată, apoi împreună cu „MyImageAnalyzer” am putut analiza fiecare cadru.

Pentru captarea imaginii pe care se stabilește diagnosticul, am folosit „Image Capture”, furnizată de „CameraX”. Aceasta este proiectată pentru a face imagini de calitate înaltă, ceea ce ajută „Pose Detection” să întoarcă rezultate cât mai precise [2, „Image capture”]. Aceasta este folosită în „CameraActivity”, la executarea celui de-al doilea „handler” ce oprește temporizatorul și totodată captează și salvează imaginea captată. Pentru implementarea sa, am creat întâi un

director în memoria dispozitivului, apoi am salvat imaginea în folder-ul nou sub o denumire unică formată din ora curentă în milisecunde (apelând funcția „System.currentTimeMillis()”).

3.3. Firebase

„Firebase este un serviciu ce oferă programatorilor o mulțime de instrumente și servicii, cu scopul dezvoltării unor aplicații de calitate.” Asigură autentificarea utilizatorilor integrând una sau mai multe metode de conectare, o bază de date cu sincronizare în timp real, stocare în cloud, acestea fiind câteva servicii importante furnizate de Firebase. O caracteristică importantă a sa este că bazele de date sunt NoSQL, ele stocând datele în documente asemănătoare JSON [3].

Pentru a ușura dezvoltarea aplicației pe partea de autentificare și bază de date, am folosit deci Firebase Authentication și Firebase Realtime Database.

3.3.1. Firebase Authentication

Firebase ne ușurează integrarea serviciului de autentificare în acest proiect prin Firebase Authentication. După crearea proiectului pe site-ul oficial al acestui serviciu, apoi conectarea acestuia la aplicația din Android Studio și adăugarea dependențelor necesare în fișierul „build.gradle(:app)”, am putut înainta cu serviciile de autentificare.

- Crearea unui cont are loc în „RegisterActivity”. Am implementat, în primul rând, validările intrărilor, în funcție de fiecare câmp: adresa de email trebuie să aibă structura standard, parola să fie de lungime minimă șase, iar completarea tuturor câmpurilor este obligatorie. Creând instanța „FirebaseAuth”, am reușit accesarea metodelor sale pentru a crea un utilizator pe baza metodei de conectare stabilite („createUserWithEmailAndPassword(email, password)”), adăugarea sa în baza de date și asigurarea că acest proces a fost finalizat cu succes.
- Logarea în cont are loc în „LogInActivity”. De asemenea, am implementat și aici validări pentru câmpuri, iar cu ajutorul instanței „FirebaseAuth” înființate aici și metoda „signInWithEmailAndPassword(email, password) ”, am logat utilizatorul. În plus, legat de un cont proaspăt creat, pentru care logarea are loc pentru prima oară, se verifică email-ul prin funcția „sendEmailVerification()”, ce trimite un mesaj pe adresa de email.
- În cazul în care utilizatorul uită parola contului său, o poate schimba accesând „Forgot Password” de pe pagina de logare. Astfel se lansează activitatea „ForgotPasswordActivity”, ce trimite un mesaj pe adresa de email prin care se poate

reseta parola, utilizând metoda „sendPasswordResetEmail(email)”. Acest fapt este posibil tot prin metodele din acest serviciu de autentificare.

- Delogarea utilizatorului se realizează foarte simplu în „MenuActivity”, doar apelând funcția „signOut()” pe instanța „FirebaseAuth”.

În toate aceste procese au fost implementate de asemenea și cazurile în care acestea nu sunt finalizate cu succes, afișându-se pe ecran mesaje corespunzătoare.

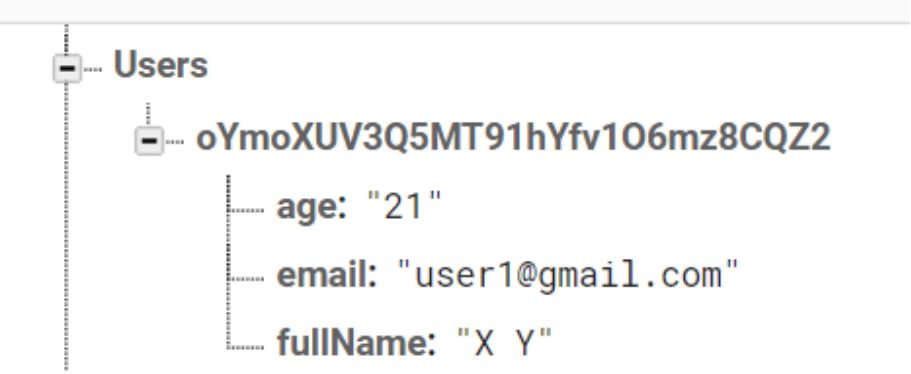

Figura 3.5: „Users” în baza de date după autentificarea unui utilizator cu numele „X Y”

3.3.2. Firebase Realtime Database

Firebase Realtime Database se constituie dintr-o bază de date în timp real care funcționează și fără conectivitate la internet. *„În cazul în care utilizatorul este offline, datele sunt stocate în memoria dispozitivului și sincronizarea va avea loc după ce se revine la conexiunea la internet. ”* În plus, un alt avantaj al acestei baze de date este că face posibilă configurarea permisiunilor de date, așadar se ocupă și de problemele de securitate a datelor [4].

Știind că în această bază de date nu există tabele sau înregistrări și că toate datele sunt salvate ca obiecte de tip JSON, ne putem imagina că ele sunt dispuse într-un arbore JSON [5].

Baza de date în timp real „Firebase” permite „nesting data” (cuibărirea datelor) pe multe niveluri, însă aceasta nu este cea mai bună practică de a reține datele. Spre exemplu, în cazurile în care se dorește preluarea unor date, se iau și toate nodurile copil ale acestora. La fel se întâmplă și atunci când permitem citirea sau scrierea unui nod, permitem accesul și la toți copiii acestuia [5]. Așadar am ales construirea unei structuri de date mai plane, mai ușor de citit și de implementat.

În această aplicație, avem nevoie de o bază de date în care să salvăm datele despre conturile utilizatorilor, diagnosticele lor pentru a putea înregistra evoluția, cât și datele despre animațiile modelului 3D. Prin urmare, am utilizat „Firebase Realtime Database” pentru a facilita procesul de implementare și pentru a garanta utilizatorilor securitatea datelor sale.

O primă entitate pe care am creat-o este cea a afecțiunilor împreună cu exercițiile pentru acestea, numită „AnimationDetails”. Am creat-o astfel încât să conțină id-uri generate automat, ce conțin la rândul lor diagnosticul corespunzător și calea către animație, ca de exemplu: „<<GVHBJK67afJNVAOSI>> : diagnostic: <<kyphosis>>, animationPath : <<sampledata/models/kyphosisFirstEx>>”, așa cum este și entitatea „Diagnostic has Angles” din Figura 3.6. Acestea au fost salvate în baza de date o singură dată, fără să sufere alte modificări pe parcurs.

Entitățile ce se actualizează pe parcursul aplicației sunt „Angles”, „Diagnostic”, „Diagnostic has Angles” și „User is diagnosed”.

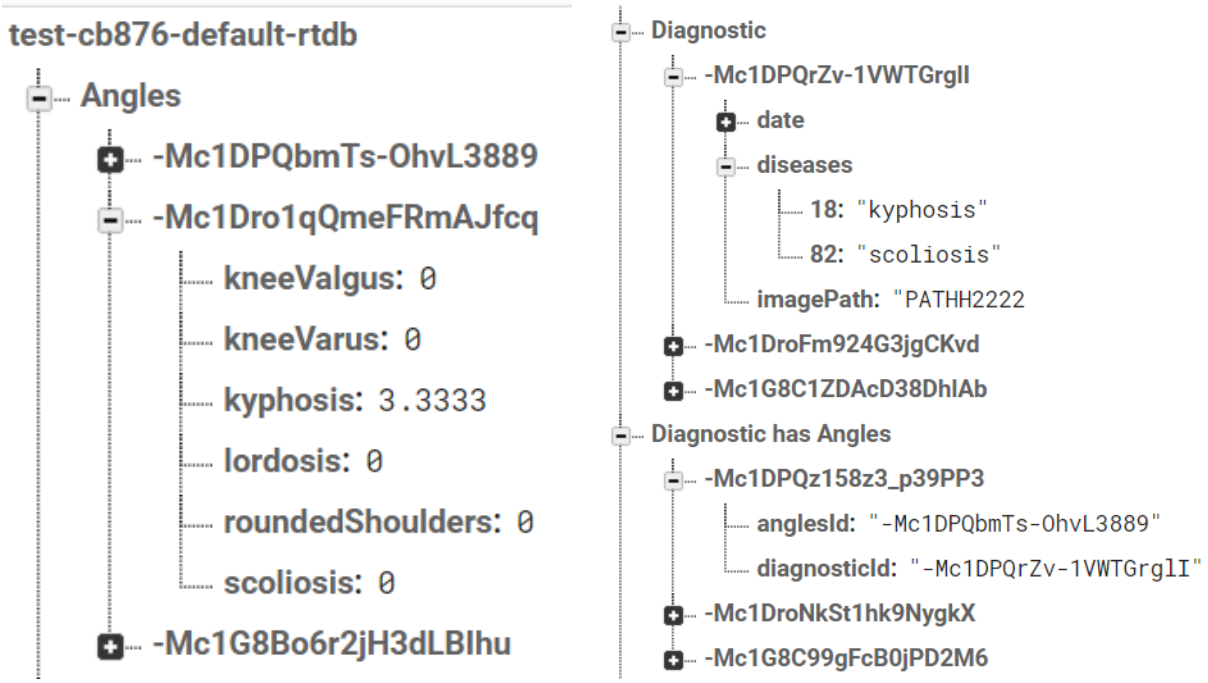

Figura 3.6: „Angles”, „Diagnostic”, „Diagnostic has angles”din baza de date

Toate acestea sunt actualizate în „CameraActivity” după ce a avut loc captarea imaginii și procesarea ei în „MyImageAnalyzer”. Aceasta se întâmplă de fiecare dată când utilizatorul deschide camera în aplicație pentru a obține un nou diagnostic, toate acestea fiind adăugate în baza de date și folosite apoi în cadrul paginii Progress. Cu ajutorul rezultatelor returnate de analizator, ce constau în lista de 33 de puncte detectate, am adăugat date entităților, pe rând: „Angles” ce calculează, „Diagnostic”, „Diagnostic has angles” și „User is diagnosed”, apoi le-am salvat corespunzător în baza de date utilizând funcțiile de actualizare.

Pentru a adăuga în baza de date, am accesat în primul rând locația dorită prin metoda „getReference()”, am creat un nou nod, generând o cheie unică, iar cu ajutorul acestora am adăugat la lista de date ce trebuie salvat. Folosind cheia generată am indentificat locul exact în care trebuie introduse datele și am actualizat valorile de acolo („.child(generatedKey).setValue(newList) ”).

În ceea ce privește operațiile de regăsire a datelor în baza de date, pentru a afișa detaliile diagnosticelor utilizatorului în pagina „Progress”, ne-am folosit de mai multe entități. Pentru început am implementat mai multe „listeners” în „ProgressActivity” care se declanșează odată cu lansarea activității. În aceste „listeners” am căutat întâi în „User is diagnosed” diagnosticele utilizatorului curent, cu ajutorul acestora am căutat și am afișat detaliile din „Diagnostic”, cele cu id-ul diagnosticelor extrase anterior. Aceeași abordare am avut-o și pentru pagina „Exercises”, cu excepția faptului că am facut aceste operații de găsire a datelor într-o singură entitate, și anume „AnimationDetails”.

3.4. ML Kit

Google pune la dispoziția programatorilor pachetul ML Kit, puternic și ușor de utilizat. Acesta folosește machine learning pe dispozitivele Android și IOS [6]. Pachetul conține două tipuri de API-uri și anume unul care analizează imagini și video-uri, iar unul care procesează limbajul natural. „Pose Detection” API, folosit în această aplicație, face parte din prima categorie, iar pentru implementarea sa am urmat recomandările din documentația oficială [6, „Detect Poses with ML Kit on Android”].

3.4.1. Implementarea Pose Detection API

API-ul „Pose Detection” detectează în cadru o persoană, acest lucru fiind posibil doar dacă se recunoaște prezența feței. Pe baza rezultatului se identifică un set de treizeci și trei de puncte ca de exemplu umeri, genunchi, șolduri, pentru care se calculează coordonatele x, y, z, precum și probabilitatea ca acestea să se afle în imagine. Acest API poate detecta și poziții parțiale ale corpului, în cazul în care acesta nu apare în întregime în imagine, iar reperelor nerecunoscute li se atribuie coordonate din afara cadrului [6, „ Pose Detection”]. Acest proiect are ca funcție principală detectarea poziției corpului, așadar, se observă că aceasta este o tehnologia potrivită pentru ce își propune aplicația.

Pentru a pune în aplicare acest API, am studiat în amănunt tot ce cuprinde, iar pentru început, pentru a detecta o poziție în imagine, trebuie crearea unei instanțe a „PoseDetector” și specificarea setărilor detectorului. Am observat că API-ul include și varianta

„AccuratePoseDetector” ce are la bază modele optimizate pentru a obține precizie, așa că am optat pentru aceasta pentru a asigura utilizatorilor un diagnostic cât mai reușit. Pentru modul se detectare am optat pentru „STREAM_MODE” ce acționează în general pe un flux de date, în aplicația noastră fiind în primul rând folosit pentru fiecare cadru începând de la deschiderea camerei până la captarea imaginii pentru diagnostic. Referitor la modul de detectare pentru analiza posturii pe imaginea statică obținută, am rămas la alegerea anterioară deoarece aceasta rulează chiar și pe un singur cadru (aici, poza statică), dar are în plus faptul că ia în considerare persoana din imagine cea mai proeminentă, în cazul în care poate pe fundal se întâmplă să apară o altă persoană.

Pentru a capta imaginile în timp real, am folosit librăria „CameraX” descrisă în secțiunea 3.2.6 a subcapitolului „3.2. Android Studio”.

Procesarea imaginii sau a cadrului s-a realizat utilizând „Task<Pose>” ce detectează poziția corpului. Dacă cel puțin fața utilizatorului este prezentă, identificarea se realizează cu succes și prin intermediul obiectului „Pose” se returnează treizeci și trei de puncte cu valori nenule, iar în caz contrar cu valori nule. Acestea sunt accesate în funcția „readerPoseData(pose)” implementată în această aplicație, cu ajutorul metodelor disponibile din „PoseLandmark”, o clasă Obiect ce reprezintă un punct de reper pentru poziția „pose” primită ca parametru.

După parcurgerea acestor pași, am obținut punctele poziției corpului dintr-un cadru. Aceste puncte sunt folosite pentru diagnosticarea utilizatorului pe imaginea captată.

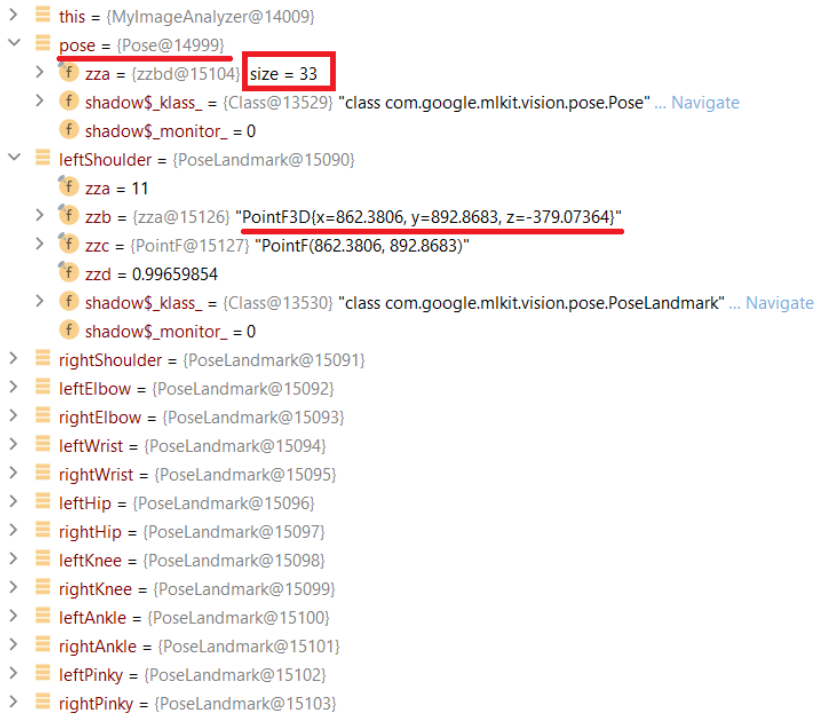

Figura 3.7: Datele returnate de „Pose Detection” API pe o imagine

Din Figura 3.7 se pot observa date importante ce urmează a fi folosite, precum: variabila „pose” de tip „Pose”, lungimea sa, conținutul ei și al punctelor returnate (de exemplu: coordonatele umărului stâng „PointF3D(x=862.3806, y=892.8683, z=-379.07364)”).

3.4.2. Analiza posturii

Examinarea poziției corpului utilizatorului se realizează pe imaginea captată cu scopul de a identifica potențialele afecțiuni posturale. Aceasta a fost realizată cu ajutorul punctelor returnate de obiectul „Pose”, ce au fost analizate în funcțiile implementate în proiectul de față. Acestea au la bază același concept, și anume calculează măsura unghiului dintre trei puncte de tipul „PoseLandmark”.

Pentru realizarea acestor metode, am studiat în primul rând obiectul „PoseLandmark”. Acesta cuprinde diverse atribute și metode, cele ce le-am utilizat aici sunt „getPosition().α” (returnează coordonata α) și „getInFrameLikelihood()” (returnează un număr aflat în intervalul [0.0f, 1.0f] ce reprezintă probabilitatea ca acel punct să se afle în cadru).

Algoritmul de calculare a unghiului format de trei puncte cunoscute, dezvoltat în documentația oficială a API-ului [16, „Recognizing a yoga pose with angle heuristics”], stă la baza metodelor implementate în proiect, ce determină unghiurile dintre punctele de interes major pentru fiecare afecțiune. De exemplu, pentru afecțiunea „scolioză”, verificăm dacă umerii sunt la același nivel , astfel: am proiectat umărul coborât (A) având aceeași coordonată x cu umărul ridicat (B), am obținut punctul C și am calculat măsura unghiului BAC. Analog am făcut și pentru șolduri.

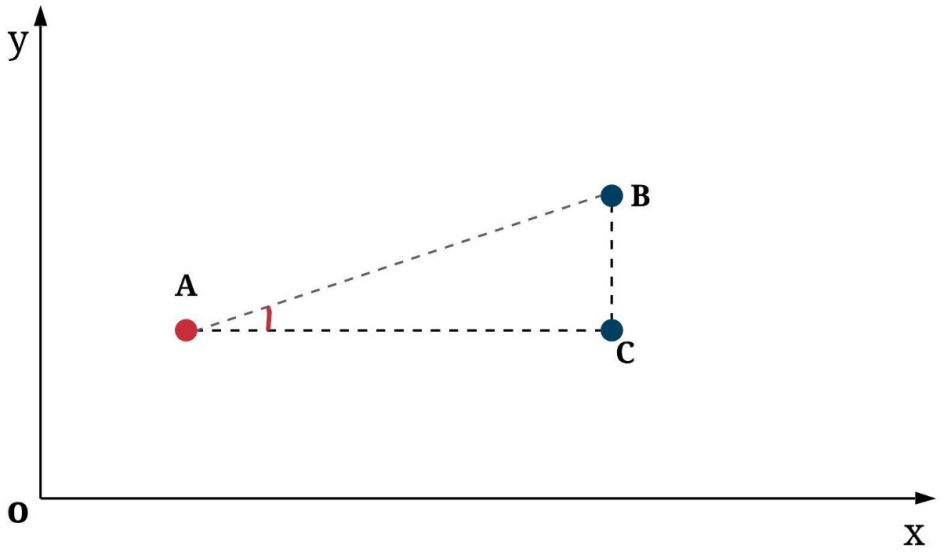

Figura 3.8: Stabilirea punctelor pentru calculul unghiului

Pentru analizele posturii ce se efectuează în realitate din lateral, am luat în calcul în acest proiect coordonata z a punctelor. Astfel, calculul unghiurilor pentru cifoză au fost bazate pe coordonatele y și z ale urechilor, umerilor și șoldurilor.

Severitatea unei afecțiuni se stabilește pe baza măsurii acestor unghiuri (de exemplu, pentru scolioză, măsura unghiului dintre umeri sau șolduri ce se află în intervalul [1.5, 3.5] reprezintă un caz mediu, cele de peste 3.5 fiind cazuri severe ce necesită consultul urgent de un medic specialist). După calcularea lor și introducerea lor în baza de date în entitatea „Angles", se fac comparațiile cu ajutorul metodelor din entitatea „Diagnostic", se setează datele acesteia (calea către imaginea pe care s-au calculat valorile, data curentă și diagnosticele împreună cu procentajul lor) și se introduce în baza de date. Pe baza acestor date aflate și salvate, se afișează pe ecran o ierarhie a afecțiunilor din punct de vedere al gravității, împreună cu nivelul de severitate.

4. Testare

Aplicația a fost rulată doar pe telefonul propriu pentru a preveni, în primul rând, întâmpinarea diferitelor conflicte și pentru a testa programul, în special funcționalitatea „Pose Detection” API și metodele implementate pentru analiza posturii.

La fiecare actualizare a aplicației am utilizat debugger-ul din Android Studio pentru a urmări comportamentul programului, metodelor și variabilelor. Am testat API-ul folosind drept modele mai multe persoane, simulând diverse dezechilibre posturale, făcând diverse tipuri de fotografii pentru a verifica veridicitatea rezultatelor și pe poze parțiale. O parte din rezultatele obținute din programul implementat se pot observa în tabelul de mai jos.

AFECȚIUNE SIMULATĂ	TIP FOTOGRAFIE	UMĂR STÂNG			UMĂR DREPT		
		X	Y	Z	X	Y	Z
Scolioză cu umărul drept ridicat (subiectul 1)	Integral	829	741	-116	430	810	-121
Scolioză cu umărul drept ridicat (subiectul 2)	Bust	881	743	-1294	317	759	-1256
Scolioză cu umărul stâng ridicat (subiectul 1)	Bust	791	526	-920	347	504	-918
Scolioză cu umărul stâng ridicat (subiectul 2)	Integral	700	776	-202	311	762	-233

Tabel 4.1 Rezultate testări „Pose Detection API”

În urma simulărilor, am confirmat corectitudinea funcționării algoritmului și mi-am format o idee mai clară despre datele furnizate.

Așadar, în acest mod am reușit să înțeleg și să implementez restul funcționalităților.

5. Concluzii și propuneri de dezvoltare ulterioară

5.1. Concluzii

Această aplicație ar putea fi considerată o necesitate în societatea actuală, un instrument la îndemâna majorității, ușor de utilizat, cu scopul de a atenționa și corecta până la un anumit nivel disfuncțiile posturii.

5.2. Dezvoltare ulterioară

Această lucrare ar putea cu siguranță să dispună de o varietate de îmbunătățiri începând de la interfața vizuală până la adăugarea de noi funcționalități, ce le voi enunța în acest capitol.

Interfața unei aplicații este mereu un catalizator pentru o primă impresie a utilizatorului. Pentru a asigura o mai bună experiență pentru cei ce folosesc această aplicație, se pot adăuga mai multe componente ce susțin interactivitatea precum: animații și componente de navigare. De asemenea, pe pagina unde se înregistrează progresul, se poate adăuga și un grafic ce se actualizează la fiecare diagnosticare nouă, pentru ca utilizatorul să dispună și de o reprezentare vizuală a evoluției sale în timp.

O altă idee ce s-ar putea adăuga acestui proiect este notificarea zilnică a utilizatorului pentru a urma antrenamentul. Astfel, se susține un antrenament regulat pentru obținerea unor rezultate mai bune.

Utilizatorul ar putea dispune și de un set de configurații ce permite schimbarea caracterului cu care se antrenează, la fel și schimbarea peisajului în care se află acesta. Astfel, se poate induce senzația unui joc, iar aplicația ar deveni mai plăcută. Tot așa, se poate adăuga și implementarea unor antrenamente cursive.

Pentru a îmbunătăți acuratețea diagnosticului, se poate capta și o a doua imagine cu utilizatorul din profil pentru a nu folosi și coordonata z, ce încă este o valoare experimentală a API-ului. Pe baza secvenței de preluare a fluxului de date deja implementate, împreună cu un model antrenat, se poate introduce detecția anumitor poze în timpul antrenamentului, pentru a afișa avertizări și a preveni mișcările incorecte ce cauzează accidentări. O altă utilitate a acestei implementări ce ia și analizează fiecare cadru din „stream”, se poate realiza și un desen în timp real al unui „schelet” format din punctele returnate de „Pose Detection” API.

Pentru ca această aplicație să fie utilizată cu încredere, ar aduce un plus semnătura unui cadru medical avizat ce verifică și stabilește veridicitatea aplicației și își dă acordul în privința siguranței utilizării acesteia.

6. Figuri și tabele

7. Bibliografie și referințe

Android developers, documentație oficială https://developer.android.com/studio/intro data accesării: 09.06.2021 [2]

Blender, documentație oficială https://www.blender.org/about/ data accesării: 09.06.2021 [1]

Firebase, documentație oficială https://firebase.google.com/docs/database/web/structure-data data accesării: 10.06.2021 [5]

Google developers, documentație oficială https://developers.google.com/ml-kit data accesării: 11.06.2021 [6]

Jessica Clark. *Top 10 Advantages of Firebase* https://blog.back4app.com/advantages-of-firebase/ data accesării: 10.06.2021 [4]

What is Firebase? https://www.educative.io/edpresso/what-is-firebase data accesării: 10.06.2021 [3]

Difference Between View and ViewGroup in Android https://www.geeksforgeeks.org/difference-between-view-and-viewgroup-in-android/ data accesării: 12.06.2021 [7]