
Rezumat

În cadrul acestei lucrări de licență am dezvoltat aplicația "Antrenor personal virtual", pentru a promova un stil de viață sănătos și activ prin intermediul exercițiilor fizice. Scopul principal al aplicației este de a atrage și motiva utilizatorii să facă sport și să își atingă obiectivele de fitness.

Lucrarea se axează pe procesul de dezvoltare al aplicației, prezentând detaliat tehnologiile utilizate, cum ar fi Android, Android Studio, SQLite. Aceste tehnologii au fost selectate pentru a crea o interfață prietenoasă și ușor de utilizat, adaptată la diferite dimensiuni de ecran și rezoluții.

Aplicația oferă utilizatorilor posibilitatea de a completa un set de întrebări prin care sunt colectate informații despre nevoile și obiectivele lor personale. Pe baza acestor răspunsuri, aplicația generează un plan de antrenament personalizat, care cuprinde exerciții optimizate pentru dezvoltarea mușchilor într-un timp cât mai scurt. Utilizatorii au și opțiunea de a modifica și personaliza antrenamentul prin adăugarea sau ștergerea exercițiilor.

Abstract

As part of this thesis we developed the "Antrenor personal virtual" app to promote a healthy and active lifestyle through exercise. The main goal of the app is to attract and motivate users to exercise and achieve their fitness goals.

The paper focuses on the app development process, detailing the technologies used, such as Android, Android Studio, SQLite. These technologies have been selected to create a friendly and easy-to-use interface, adapted to different screen sizes and resolutions.

The app gives users the opportunity to complete a set of questions that collect information about their personal needs and goals. Based on these answers, the app generates a personalised training plan, which includes exercises optimised for muscle development in the shortest possible time. Users also have the option to modify and customize the workout by adding or deleting exercises.

Cuprins

Cuprins	1
Lista Figurilor	3
1 Introducere	4
1.1 Descrierea aplicației	4
1.2 Motivație	5
1.3 Structura lucrării	6
1.4 Obiectivele aplicației	6
1.5 Aplicații asemănătoare	6
1.6 Compararea aplicației	9
2 Tehnologii utilizate	11
2.1 Android	12
2.1.1 Arhitectura sistemului android	12
2.2 Servicii Google Play	14
2.3 Android Studio	15
2.3.1 Gradle	16
2.4 SQLite	17

2.5 Java	18
2.6 XML	20
3 Implementarea aplicației	22
3.1 Logica de funcționare	22
3.2 Baza de date	23
3.3 Structura aplicației	24
3.3.1 Activități	24
3.4 Algoritmi utilizați	26
3.5 Interfață	31
3.6 Analiza SWOT	33
4 Utilizarea aplicației	34
4.1 Scenariu utilizare	35
4.2 De ce să utilizăm aplicații fitness	38
5 Concluzii	40
Referințe bibliografice	41

Lista Figurilor

1.1	Poză aplicație Gym Workout Planner & Tracker	7
1.2	Poză aplicație Home Workout - No Equipments	8
1.3	Poza aplicație asemanătoare, imagine preluata din Gym Workout Bodybuilding Coach	10
2.1	Arhitectura sistemului	13
2.2	Interfață Android Studio	16
2.3	Sintaxa XML	21
3.1	Diagrama UseCase a aplicației	22
3.2	Diagrama ERD	24
3.3	Diagrama UML a activităților importante în afișare	25
3.4	Diagrama UML a activităților	26
3.5	Scheletul interfeței grafice main	32
3.6	Analiza SWOT	33
4.1	Activitatea de pornire a aplicației	35
4.2	Fereastra main a aplicației	36
4.3	Activitatea de ștergere a unui exercițiu	37
4.4	Descriere exercițiu	38

Capitolul 1

Introducere

Evoluția aplicațiilor mobile este o călătorie fascinantă care a transformat modul în care interacționăm cu tehnologia și accesăm informațiile. Primele aplicații mobile au fost aplicații simple, preinstalate, care însătoau cele mai vechi telefoane mobile. Aceste aplicații oferea funcționalități de bază, cum ar fi calculatoare, calendare și liste de contacte. De obicei, acestea erau integrate în firmware-ul telefonului și aveau opțiuni de personalizare limitate.

Disponibilitatea din ce în ce mai mare a internetului de mare viteză și a cloud computing-ului a dat naștere aplicațiilor mobile bazate pe cloud. Aceste aplicații descarcă sarcinile de calcul și stocarea datelor pe serverele de la distanță, reducând dependența de resursele dispozitivului. Aplicațiile bazate pe cloud pot oferi funcționalități mai complexe, cum ar fi colaborarea în timp real, procesarea extinsă a datelor și sincronizarea fără întreruperi între dispozitive. Evoluția aplicațiilor mobile continuă odată cu integrarea tehnologiilor de realitate augmentată (AR) și realitatea virtuală (VR) care transformă jocurile mobile în experiențe imersive. În plus, progresele în domeniul învățării automate și al inteligenței artificiale (AI) permit aplicații inteligente care pot înțelege comportamentul utilizatorului, personaliza conținutul și oferi capacitați predictive.

1.1 Descrierea aplicației

Aplicația are ca scop principal formarea unui plan de antrenament personalizat pentru fiecare utilizator, în funcție de răspunsurile sale la un set de întrebări standard. Prin colectarea informațiilor relevante despre obiectivele, nivelul de fitness și preferințele utilizatorului, aplicația poate crea un antrenament eficient și adaptat nevoilor individuale.

Planul de antrenament generat de aplicație se bazează pe o selecție atentă a exercițiilor fizice, care au fost studiate și validate de către experti și antrenori din domeniul fitness-ului. Aceste exerciții sunt considerate cele mai optime pentru dezvoltarea și tonifierea mușchilor, îmbunătățirea rezistenței și obținerea rezultatelor dorite într-un interval de timp cât mai scurt.

Un aspect distinctiv al aplicației este posibilitatea utilizatorului de a personaliza propriul antrenament. Utilizatorul are libertatea de a adăuga exerciții suplimentare în planul său de antrenament, în funcție de preferințe sau de grupurile musculare pe care dorește să le lăcreeze mai intens într-o zi specifică. Această flexibilitate permite adaptarea antrenamentului în funcție de nevoile și preferințele individuale ale utilizatorului.

Aplicația poate fi utilizată atât acasă, unde utilizatorul poate efectua exercițiile fără a avea nevoie de echipamente sofisticate, cât și la sala de fitness, unde există o gamă mai largă de echipamente și exerciții disponibile. Utilizatorul are posibilitatea de a selecta exercițiile care sunt potrivite pentru mediul în care se antrenează și pentru resursele pe care le are la dispoziție.

Prin intermediul acestei aplicații, utilizatorii beneficiază de un plan de antrenament personalizat, eficient și adaptat nevoilor lor individuale. Aceasta oferă o modalitate convenabilă și accesibilă de a obține rezultatele dorite într-un mod eficient și motivant.

Aplicația pe care doresc să o prezint în continuare, se încadrează în categoria aplicațiilor de sănătate. Scopul principal al aplicației este de a ajuta oamenii pierduți în sală, să-și croiască un drum în lumea fitness-ului.

1.2 Motivație

Motivația în dezvoltarea acestei aplicații este de a-i ajuta pe oameni să își înceapă călătoria în vasta lume a fitness-ului. Într-o societate în care oamenii devin tot mai leneși și mai comodizați, dorim să oferim o soluție accesibilă și convenabilă pentru a-și atinge obiectivele de fitness. Cu ajutorul aplicației, doresc să facilitez oamenilor, să obțină rezultatele dorite, fie că vizează pierderea în greutate sau construirea de mușchi, într-un mod mai rapid și mai eficient decât metodele tradiționale.

Exercițiile fizice, deși extenuante la început, aduc o fericire și o satisfacție imediată. Utilizatorii vor putea experimenta beneficiile transformătoare ale activității fizice într-un timp relativ scurt. Vor simți o îmbunătățire a capacitații lor de a utiliza corpul, vor deveni mai energici și vor resimți o creștere a nivelului de fericire și bunăstare generală.

În plus, dezvoltarea acestei aplicații a fost motivată și de nevoie personală de a urmări progresul meu de antrenament. Uneori, este ușor să uităm exercițiile pe care trebuie să le facem sau exercițiile pe care le-am efectuat în săptămâna precedentă. Prin intermediul acestei aplicații, am creat o soluție practică pentru a urmări progresul personal, ceea ce facilitează monitorizarea și evaluarea evoluției în timp.

Nu în ultimul rând, activitatea fizică regulată are un impact semnificativ asupra calității somnului. Prin intermediul aplicației încurajez oamenii să se bucure de un somn odihnitor și calitativ. Activitatea fizică contribuie la eliberarea stresului acumulat, la îmbunătățirea ritmului somnului și la o odihnă mai profundă, ceea ce se traduce într-o stare generală de bine

și echilibru în viața de zi cu zi. Aceste motive și beneficii subliniază importanța pe care o acord sănătății și fitness-ului, precum și dorința de a împărtăși aceste beneficii cu alții prin intermediul aplicației.

1.3 Structura lucrării

În capitolul 1 este prezentată o introducere a aplicației, motivația și mai multe detalii despre aplicație.

În capitolul 2 sunt prezentate tehnologiile care au fost folosite pentru a crea aplicația.

Capitolul 3 prezintă în detaliu funcționalitățile aplicației dezvoltate, metodele utilizate pentru implementarea acestora.

În capitolul 4 este prezentat un scurt scenariu de cum un utilizator se poate folosi de aplicație.

În capitolul 5 sunt prezentate concluziile finale.

1.4 Obiectivele aplicației

Aplicația își propune să ofere utilizatorilor un antrenament adaptat nevoilor și obiectivelor lor individuale. Prin intermediul unui set de întrebări și informații colectate despre utilizator, aplicația generează un plan de antrenament care să ajute la dezvoltarea mușchilor într-un timp cât mai scurt și eficient.

Oferă utilizatorilor posibilitatea de a-și personaliza propriul antrenament prin adăugarea sau ștergerea exercițiilor. Aceasta permite adaptarea antrenamentului în funcție de preferințe, nivelul de fitness și obiectivele individuale ale utilizatorului.

Un alt obiectiv al aplicației este de a oferi o experiență intuitivă și placută utilizatorilor.

1.5 Aplicații asemănătoare

Aplicații existente ce îndeplinesc același scop sunt:

1. Gym Workout Planner & Tracker
2. Gym Workout BodyBuilding Coach
3. Home Workout - No Equipments

Gym Workout Planner & Tracker este o aplicație de mobil, compatibilă cu Android și iOS, care generează un antrenament fizic, în funcție de câte zile alegi să te antrenezi pe săptămână

și oferă opțiunea de a te focaliza pe o grupă musculară deficentă.

Interfața grafică a aplicației este ușor de folosit, în afară de faptul că dacă vrei să vizualizezi exerciții dintr-o grupă musculară respectivă, trebuie să apeși pe un corp al unui antrenor. Este o metodă foarte interesantă de afișare, dar din punctul meu de vedere este prost implementată, deoarece unele exerciții se suprapun și este dificil să apeși pe ce grupă dorești. Tot odată aplicația cere să plătești o subscriptie lunară dacă dorești să urmărești progresul din săptămâna sau luna respectivă.

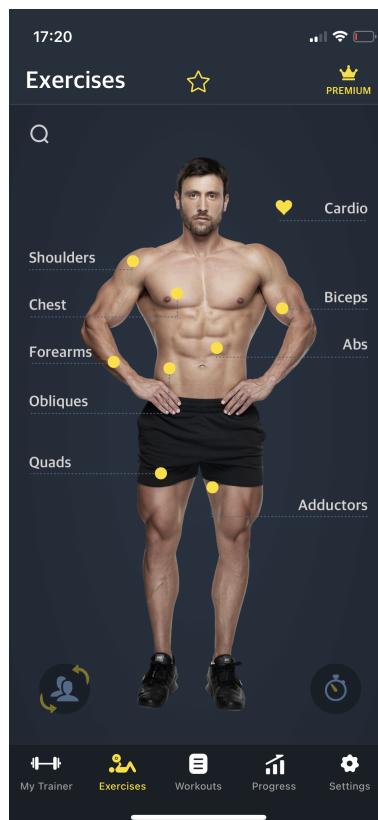


Figura 1.1: Poză aplicație Gym Workout Planner & Tracker

Gym Workout BodyBuilding Coach deși se numește așa și spune că le creează utilizatorilor un antrenament, această aplicație nu face decât să ofere explicații despre exerciții, iar acelea la rândul lor trebuie cumpărate sau ca utilizatorul să vizioneze o reclamă.

Home Workout - No Equipments este o aplicație care după câte spune numele oferă antrenamente acasă. Această aplicație oferă decât antrenamente standard pentru utilizatorii free, deoarece au o subscriptie și doar așa poti primi un antrenament personalizat. Interfața grafică este ușor de utilizat, este foarte descriptiv tot. Ce este foarte interesant la această aplicație este că are antrenamente pe grade de dificultate, ceea ce mi se pare normal pentru exercițiile care necesită doar greutatea corporală, deoarece durează destul de mult timp pentru mușchi să ajungă la forța necesară să execute corect mișcarea.

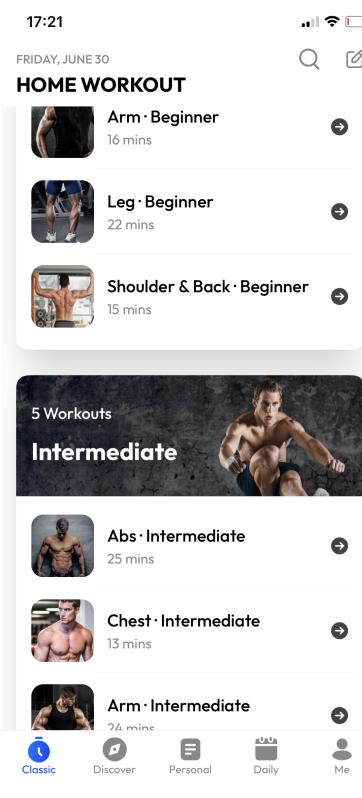


Figura 1.2: Poză aplicație Home Workout - No Equipments

1.6 Compararea aplicației

În ultimii ani, industria fitness-ului a înregistrat o creștere a popularității aplicațiilor pentru antrenament. Aceste aplicații de mobil au revoluționat modul în care indivizii își planifică, urmăresc și își optimizează antrenamentele. Cu toate acestea, este demn de remarcat faptul că majoritatea aplicațiilor necesită un abonament lunar pentru a debloca întreaga gamă de caracteristici și beneficii. Aplicația "Antrenor Personal Virtual" se diferențiază de celelalte aplicații de fitness prin faptul că oferă o gamă completă de caracteristici și beneficii fără a necesita un abonament lunar. Utilizatorii pot beneficia de toate funcționalitățile esențiale pentru planificarea și urmărirea antrenamentelor fără a plăti costuri suplimentare. În contrast cu alte aplicații, consumatorii au opțiunea de a alege între antrenamentele acasă și cele în sala de fitness. Aceasta le permite să se adapteze mai bine la preferințele și nevoile lor individuale în ceea ce privește mediu de antrenament.

Utilizatorii au acces gratuit la toate explicațiile și instrucțiunile necesare pentru a efectua corect exercițiile. Nu impun costuri suplimentare pentru informații esențiale legate de tehniciile corecte de execuție a exercițiilor, astfel încât utilizatorii să se poată antrena în siguranță și eficient. În timp ce alte aplicații pot încărca utilizatorii cu reclame invazive, experiența de utilizare a aplicației este lipsită de publicitate enervantă. Prioritatea este concentrată pe oferirea unui mediu curat și fără intreruperi pentru utilizatori, astfel încât să se poată concentra pe antrenamentul lor.

Aceste idei evidențiază aspectele care fac aplicația unică și atractivă în comparație cu alte aplicații de fitness disponibile pe piață. Ele subliniază beneficiile utilizatorilor și abordarea de a oferi funcționalități și caracteristici valoroase fără a impune costuri suplimentare sau restricții.

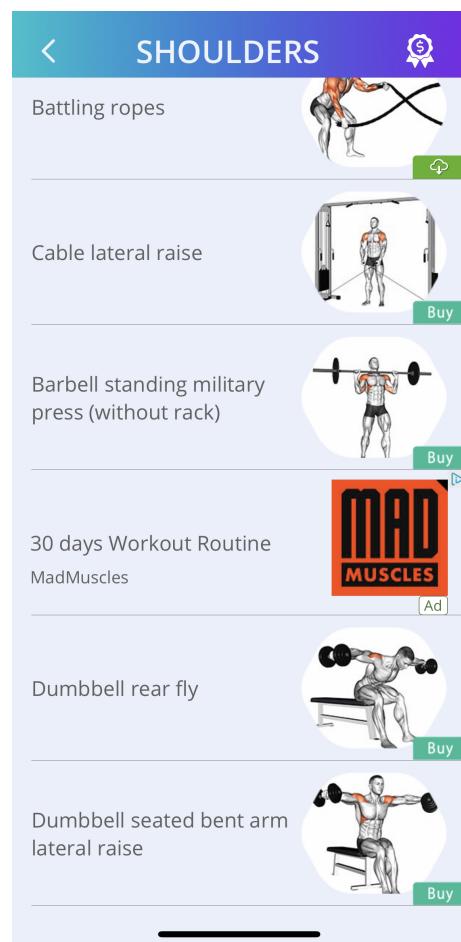


Figura 1.3: Poza aplicație asemanătoare, imagine preluata din Gym Workout Bodybuilding Coach

Capitolul 2

Tehnologii utilizate

Dezvoltarea aplicațiilor mobile este o activitate complexă, care poate varia în durată de la câteva zile până la câțiva ani, implicând adesea atât indivizi izolați, cât și echipe întregi de dezvoltatori. Orice aplicație la început trebuie să facă o planificare și analiză a cerințelor ce determină obiectivele aplicației mobile și se analizează cerințele funcționale și non-funcționale. Se identifică publicul țintă, se stabilesc caracteristicile și funcționalitățile cheie ale aplicației.

După ce se creează structura și fluxul aplicației, se proiectează interfața utilizatorului, se definesc elementele grafice și se stabilesc principiile de design pentru a asigura o experiență utilizator plăcută și intuitivă. Dezvoltarea backend-ului este o etapă ce implică dezvoltarea și implementarea componentelor de backend ale aplicației mobile, inclusiv gestionarea bazelor de date, logica de afaceri, securitatea și integrarea cu servicii externe. Se realizează programarea interfeței utilizatorului, inclusiv designul grafic, animațiile și interacțiunea utilizatorului cu aplicația. Se utilizează tehnologii specifice platformei mobile, cum ar fi Java/Kotlin pentru Android sau Swift/Objective-C pentru iOS.

Se efectuează teste pentru a verifica funcționalitatea, performanța și stabilitatea aplicației. Se rezolvă erorile și se optimizează performanța pentru a asigura o experiență utilizator de calitate. După finalizarea dezvoltării și testării, aplicația este pregătită pentru lansare. Se creează un pachet de instalare (APK pentru Android sau IPA pentru iOS) și se distribuie în magazinele de aplicatii, cum ar fi Google Play Store sau App Store. După lansare, se asigură mențenanța aplicației și se oferă actualizări periodice pentru a adăuga noi caracteristici, a rezolva eventuale probleme și a menține aplicația actualizată cu noile versiuni de sistem de operare și cerințele utilizatorilor.

Dezvoltarea aplicațiilor mobile necesită abilități tehnice avansate, o înțelegere profundă a platformei țintă și o atenție deosebită acordată nevoilor și preferințelor utilizatorilor. Prin urmare, este importantă o abordare bine structurată și o colaborare eficientă între membrii echipei de dezvoltare pentru a obține rezultate de succes.

2.1 Android

¹ Android este un sistem de operare pentru mobil dezvoltat de Google. Este utilizat pe o varietate de dispozitive, inclusiv smartphone-uri, tablete și ceasuri. Android este bazat pe sistemul de operare Linux și este scris în Java și C++. Este conceput să fie flexibil și personalizabil, permitând dezvoltatorilor să creeze o gamă largă de aplicații pentru utilizare pe dispozitive Android. Android este cel mai popular sistem de operare pentru mobil din lume, cu o bază mare și activă de utilizatori. Este în mod constant actualizat și îmbunătățit, cu noi versiuni lansate în mod regulat.

Codul sursă a fost folosit pentru a dezvolta variante ale Android pe o gamă mare de electronice, cum ar fi console de jocuri, camere digitale, playere portabile de media, chiar și unități Desktop, fiecare cu o interfață de utilizator specializată. Unele derivate bine cunoscute include Android TV pentru televizoare și Wear OS pentru dispozitive portabile, ambele dezvoltate de către Google. Pachetele de software de pe Android, care utilizează formatul APK, sunt în general distribuite prin magazine de aplicații proprietare cum ar fi Google Play Store, Amazon Apstore, Samsung Galaxy Store, Huawei AppGallery, Cafe Bazaar și GetJar, sau platforme open source cum ar fi Aptoid sau F-Droid.[1]

2.1.1 Arhitectura sistemului android

² La baza arhitecturii android-ului se află nucleul linux, care asigură stratul de abstractizare hardware, driverele și funcționalitățile esențiale ale sistemului de operare. Nucleul se ocupă de sarcini precum gestionarea memoriei, a proceselor și interacțiunile hardware oferind stabilitate și securitate. Android Runtime este mediul în care rulează aplicațiile, acesta include biblioteci de bază și o mașină virtuală Dalvik, care este responsabil pentru a executa și gestiona codul aplicației. Stratul de framework al aplicațiilor oferă o colecție de API-uri de nivel înalt pe care dezvoltatorii le folosesc pentru a crea aplicații Android. Facilitează dezvoltarea aplicațiilor și oferă resurse pentru manipularea elementelor interfeței cu utilizatorul, gestionarea ciclului de viață al aplicației și accesarea serviciilor de sistem.

În stratul cel mai de sus se află aplicațiile Android, care sunt construite cu ajutorul SDK, și constau într-o combinație de activități, servicii, furnizori de conținut. Ce ne permite să putem vizualiza aplicațiile este componenta Android UI care permite dezvoltatorilor să creeze interfețe grafice interactive. Stratul UI este format din vizualizări, layout-uri și widget-uri. Dezvoltatorii folosesc un limbaj XML pentru a putea crea layout-urile, iar cu ajutorul limbajului Java se poate face interacțiunea cu ecranul. Ultimul strat este stratul de abstractizare hardware care permite sistemului de operare să funcționeze pe diverse dispozitive cu configurații hardware diferite.

¹ Android Source, 16.12.2022, <https://source.android.com/>

² Android Source, 28.06.2023, <https://source.android.com/>

Include biblioteci care permit comunicarea cu componente hardware, cum ar fi camera, Wi-Fi și Bluetooth.[2]

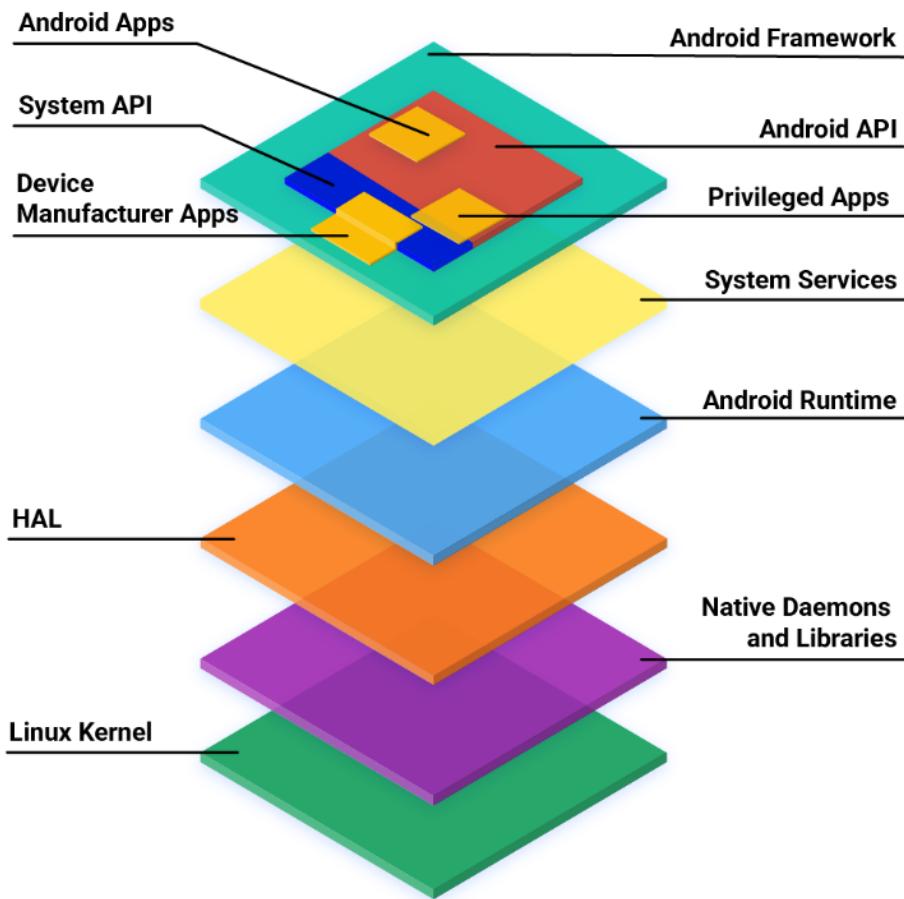


Figura 2.1: Arhitectura sistemului

2.2 Servicii Google Play

³ Google play este un magazin digital care conține aplicații și jocuri pentru dispozitivele Android, cât și muzică, filme, cărți, și show-uri TV. Cateva servicii oferite de Google Play sunt:

1. App Store: Google Play le permite utilizatorilor să navigheze și să descarce o varietate mare de aplicații și jocuri pentru dispozitivele Android.
2. Muzică: Google Play Music este o platformă care le permite utilizatorilor să asculte și să descarce muzică dintr-o librărie cu peste 50 milioane de melodii.
3. Newsstand: Este o aplicație de citit știri sau reviste.

Multe aplicații Android depind de API-urile și funcționalitățile esențiale oferite de Google Play Services. Funcțiile pe care le oferă includ autentificarea, indexarea aplicațiilor, accesul la API-urile Google, autentificarea prin Google Sign-In și mesageria în cloud (cunoscută sub numele de Firebase Cloud Messaging).

Sunt capabile să primească actualizari independent de sistemul de operare Android. Google poate oferi actualizări și caracteristici noi unui număr mai mare de dispozitive Android fără a necesita o actualizare a sistemului de operare completă. Prin urmare, garantează că utilizatorii pot beneficia de cele mai recente caracteristici și îmbunătățiri, indiferent de versiunea Android a dispozitivului.

Dezvoltatorilor li se permite să integreze caracteristici precum realizări, clasamente, suport pentru mai mulți jucători și funcționalitate de salvare în cloud cu ajutorul acestei secțiuni a Google Play Services. Aceasta îmbunătățește experiența de joc a utilizatorilor și facilitează interacțiunile sociale.

Google Play Services oferă API-uri care permit accesarea serviciilor bazate pe localizare, cum ar fi informații de localizare bazate pe GPS, Wi-Fi și retele celulare. Le permite dezvoltatorilor să integreze caracteristici care țin cont de locație, cum ar fi hărțile, geofencing-ul și urmărirea locației, în aplicațiile lor. Include integrarea cu Firebase, o platformă de dezvoltare mobilă completă. O varietate de servicii sunt disponibile pe Firebase, inclusiv stocare în cloud, bază de date în timp real, autentificare, raportare an accidentelor și analiză. Dezvoltatorii pot folosi Firebase prin intermediul Google Play Services pentru an include caracteristici puternice în aplicațiile backend.

Utilizarea API-ului Smart Lock oferit de Google Play Services permite dezvoltatorilor să stocheze și să recupereze parolele și datele de autentificare ale utilizatorilor într-un mod sigur. Această caracteristică face mai ușor pentru utilizatori să se conecteze la diferite dispozitive și aplicații, completând automat acreditările atunci când este necesar.[3]

³ Google Play Services, 17.12.2022, https://play.google.com/console/about/playgamesservices/?gclid=Cj0KCQiA14WdBhD8ARIsANao07ixcm5MKCTWgHw0um2O8Pgh3ICW_iavc9u2j6TTu15JfK-hy6tdcuAaAq8mEALw_wcB

2.3 Android Studio

⁴ Android Studio este mediul de dezvoltare integrat (IDE) oficial pentru dezvoltarea de aplicații Android. Bazat pe un editor de cod puternic și unelte pentru dezvoltatori din IntelliJ IDEA. Android Studio oferă chiar mai multe caracteristici care îmbunătățesc productivitatea atunci când construți aplicații Android, precum:

1. Un sistem de compilare flexibil bazat pe Gradle
2. Un emulator rapid și bogat în funcții
3. Un mediu unit în care puteți să dezvoltați aplicații pentru toate dispozitivele Android
4. Instrumente de testare extinse și framework-uri
5. Suport pentru C++, Java și NDK[4]

Sistemul de compilare vă poate ajuta să creați versiuni diferite ale aceleiași aplicații dintr-un singur proiect. Acest lucru este util atunci când aveți atât o versiune gratuită, cât și una cu plată a aplicației dvs. sau dacă doriți să distribuiți mai multe APK-uri pentru diferite configurații de dispozitive pe Google Play.

Suportul pentru mai multe APK-uri permite să creați rapid mai multe APK-uri în funcție de densitatea ecranului sau ABI. De exemplu, puteți crea APK-uri distincte pentru densitățile de ecran hdpi și mdpi într-o singură aplicație, considerându-le ca o singură variantă și permitându-le să partajeze setările APK de testare, javac, dx și ProGuard.

În scriptul de compilare la nivel de modul, numiți dependențele proiectului dvs. Dependentele sunt găsite de Gradle și făcute disponibile în compilarea dvs. În fișierul build.gradle.kts, puteți declara dependențele modulelor, dependențele binare locale și dependențele binare la distanță.

⁴ Android Studio, 07.01.2023, <https://developer.android.com/studio/intro>

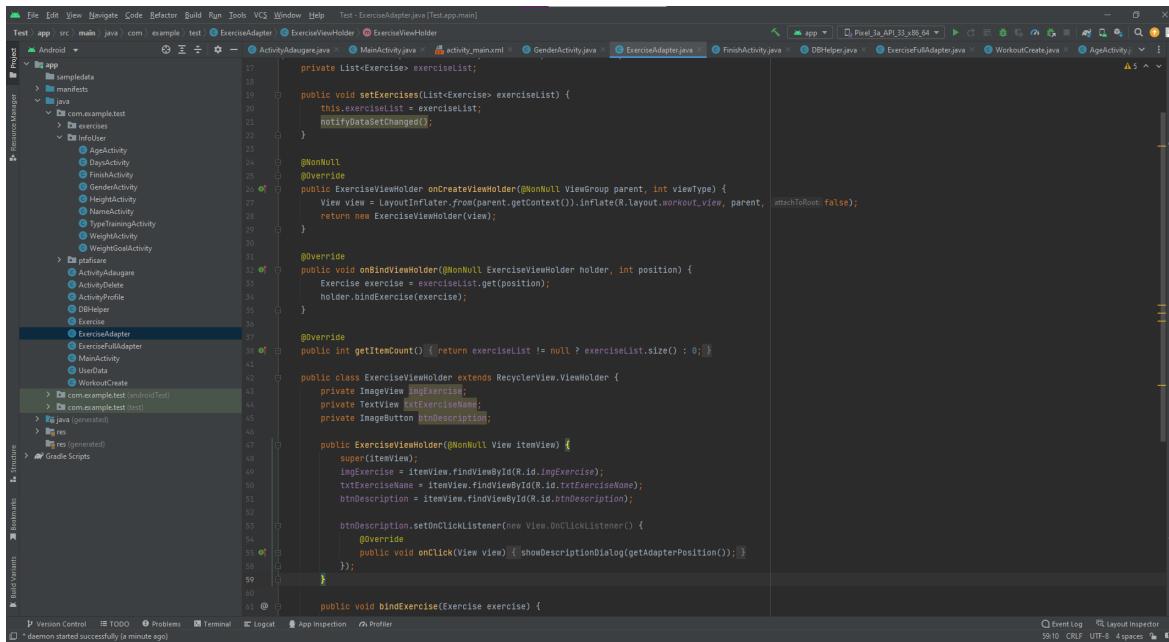


Figura 2.2: Interfață Android Studio

2.3.1 Gradle

⁵ Gradle este un sistem de construire a aplicațiilor, bazat pe Java și care poate fi folosit pentru a automatiza compilarea, testarea, distribuirea și alte sarcini pentru proiectele software. Este de obicei utilizat pentru a construi aplicații Java, dar poate fi utilizat pentru a construi aplicații în orice limbaj de programare care poate fi compilat într-un JVM (Mașina virtuală Java).

Gradle este un sistem de construire modern care oferă o varietate de beneficii față de alte sisteme de construcție, cum ar fi Apache Ant sau Apache Maven. Gradle oferă o configurare mai flexibilă prin intermediul limbajului de scriptare Groovy, poate construi proiecte mai rapid prin intermediul compilării în paralel și poate construi proiecte care implică mai multe limbi de programare.

Gradle este folosit în mod frecvent în proiectele de software open-source și în aplicațiile comerciale pentru a automatiza procesul de construcție. De exemplu, este utilizat pentru a construi aplicații Android în Android Studio. [5]

Gradle este proiectat pentru a avea o performanță ridicată în construirea aplicațiilor. Folosește tehnici avansate, cum ar fi compilarea în paralel și caching-ul rezultatelor anterioare, pentru a accelera procesul de construcție. De asemenea, suportă construirea incrementală, astfel încât să se reconstruască doar părțile modificate ale aplicației, reducând timpul de construcție.

Gradle facilitează gestionarea dependențelor aplicației prin intermediul sistemului de ges-

⁵ Gradle Enterprise, 07.01.2023, <https://gradle.com/who-we-are/?ga=2.71212880.1918857816.1673301538-388143168.1673301538>

tionare a dependențelor. Dezvoltatorii pot specifica dependențele externe ale proiectului în scriptul de construire, iar Gradle se ocupă de descărcarea și gestionarea acestora în mod automat.

În concluzie, Gradle oferă o abordare modernă și puternică pentru construirea aplicațiilor software, facilitând automatizarea, gestionarea dependențelor și performanța ridicată. Este folosit pe scară largă în proiecte open-source și în industria software pentru construirea diverselor tipuri de aplicații, inclusiv aplicații Android.

2.4 SQLite

⁶ SQLite este un proiect care a fost înființat pe data de 09-05-2000 ce conține o bibliotecă care implementează un motor de baze de date SQL, autonom, fără server, cu configurație zero și tranzacțional. Codul pentru SQLite este public și prin urmare poate fi utilizat în orice scop, comercial sau privat. SQLite este cea mai răspândită baza de date din lume, cu mai multe aplicații decât putem număra, inclusiv câteva proiecte la profil înalt.

Spre deosebire de majoritatea celorlalte baze de date SQL, SQLite nu are un proces de server separat, citește și scrie direct în fișiere obișnuite de pe disc. O bază de date SQL completă, cu multe tabele, indici, declanșatori și vizualizări, este conținută într-un singur fișier din disc. Formatul fișierului al bazei de date este cross-platform, adică se poate copia liber o bază de date între sisteme de 32 și 64 biți, sau între arhitecturi big-endian și little-endian. Trebuie să ne gândim la SQLite nu ca la un înlocuitor pentru Oracle care este o companie ce a dezvoltat MySql, o altă bază de date foarte populară, ci ca la un înlocuitor pentru fopen(), care este o funcție pentru a deschide un fișier cu numele indicat de calea unde este salvat fișierul și căruia îi asociem un flux.

⁶ SQLite, 08.01.2023, <https://www.sqlite.org/about.html>

Căteva puncte cheie despre SQLite:

1. Bază de date incorporată: SQLite este conceput pentru a fi încorporat direct în aplicații, care nu necesită o configurare separată a serverului.
2. Autonomă: SQLite este un motor de baze de date autonom, care funcționează pe o singură mașină. Aplicațiile interacționează cu baza de date SQLite prin accesarea directă a fișierului bazei de date.
3. Conformitate ACID: Respectă principiile ACID, prescurtare care vine de la atomicitate, coherență, izolare și durabilitate, asigurând integritatea și fiabilitatea datelor.
4. Suport SQL: Suportă un subset larg al standardului SQL, limbaj de interogare structurat. Aceasta permite dezvoltatorilor să creeze, să manipuleze și să interogheze bazele de date relationale utilizând instrucțiunile SQL.
5. Performanță ridicată: SQLite este proiectat pentru eficiență și viteză, folosește tehnici de optimizare cum ar fi memoria cache, și optimizarea interogărilor.

[6]

2.5 Java

⁷ Java este un limbaj de programare, orientat pe obiecte, renumit pentru versatilitatea, independența platformei și aplicațiile sale extinse. Creat de James Gosling, Patrick Naughton și echipa lor de la Sun Microsystems, la mijlocul anilor 1990, Java a fost conceput pentru a fi un limbaj portabil și sigur, capabil să ruleze pe orice dispozitiv sau sistem de operare. De la începuturi, Java a devenit unul dintre cele mai populare limbi de programare la nivel mondial, conducând la dezvoltarea a nenumărate aplicații, de la software de întreprinderi, la aplicații de mobil.

Povestea limbajului începe din anii 1991, când o echipă de ingineri a început să lucreze la un proiect de limbaj numit Oak. Echipa urmărea să creeze un limbaj de programare pentru dispozitive electronice. Cu toate acestea, proiectul a întâmpinat multe provocări din cauza puterii de procesare limitate a dispozitivelor, echipa mutându-și atenția asupra dezvoltării unui limbaj pentru dispozitive conectate în rețea, și astfel s-a născut limbajul de programare cunoscut de noi sub numele de Java.

Java a introdus conceptul de "Write Once, Run Anywhere", tradus din engleză ca "Scrie Odată, Rulează Oriunde", ceea ce înseamnă că acest limbaj de programare poate fi scris pe o platformă și rulat pe orice altă platformă care acceptă Java, fără a fi nevoie de recompliere. Această caracteristică a fost facută posibilă prin intermediul mașinilor virtuale Java care acționează ca un strat de abstractizare între codul Java și hardware-ul.[7]

⁷ Java, 28.06.2023 https://www.java.com/en/download/help/whatis_java.html

Fiind un limbaj complet orientat pe obiecte, el pune accentul pe principiile de:⁸

- 1. Abstractizare**
- 2. Încapsulare**
- 3. Polimorfism**
- 4. Moștenire**

Abstractizarea implică reprezentarea entităților complexe din lumea reală sub forma unor modele simplificate în cadrul codului. Se concentrează pe definirea caracteristicilor și comportamentelor esențiale, dar în același timp, ascunde detalii inutile de implementare. Permite crearea de clase și interfețe, care servesc ca planuri pentru clasele concrete.

Încapsularea se referă la gruparea datelor și a metodelor care operează asupra acestor date într-o singură unitate care se numește obiect. Permite ca datele să fie ascunse și să fie accesate numai prin intermediul metodelor definite, cunoscute sub numele de getters și setters.

Polimorfismul permite ca obiectele din clase diferite să fie tratate ca obiecte ale unei superclase comune. Permite utilizarea unei singure interfețe pentru obiecte de diferite tipuri, oferind flexibilitate în proiectarea codului. Se realizează prin suprascrierea și supraîncărcarea metodelor.

Moștenirea este un mecanism care permite unei clase să moștenească proprietăți și comportamente de la o altă clasă. Clasa care moștenește se numește subclasă, în timp ce clasa de la care se moștenește se numește superclasă. Permite reutilizarea codului și crearea relațiilor ierarhice între clase.^[8]

Java oferă o bibliotecă standard care simplifică dezvoltarea prin oferirea de componente și API-uri gata de utilizare pentru sarcini precum crearea de rețele, operațiuni de intrare/ieșire, conectivitate la baze de date, etc.

⁸ Principii oop, 28.06.2023 <https://www.sciencedirect.com/science/article/abs/pii/004578259290180R>

2.6 XML

⁹ XML sau Extensible Markup Language, este un limbaj de marcă utilizat pe scară largă, conceput pentru a stoca și transporta date structurate. Oferă un format de organizare și descriere a datelor într-o manieră ușoară, care poate fi înțeles de oricine. XML a devenit un standard pentru schimbul și stocarea datelor, în special în aplicații web.

Utilizează o structură bazată pe etichete pentru a defini elementele și relațiile dintre acestea, fiind foarte similar cu limbajul HTML (HyperText Markup Language). Permite reprezentarea a diferite tipuri de date, inclusiv text, numere, date etc. Oferă o abordare flexibilă și extensibilă pentru definirea elementelor și atributelor. Documentele XML pot fi asociate cu DTD (Document Type Definition) sau cu o schemă XML. Sunt utilizate pentru a defini structura, constrângeri și tipuri de date pentru documentul XML. Natura descriptivă și formatul independent de platformă îl face potrivit pentru schimbul de date dintre sisteme și aplicații. Documentele pot fi analizate și procesate cu ușurință de diverse limbi și tehnologii de programare. Aceasta permite interoperabilitatea datelor, oferind un format comun pentru comunicare și integrare.

Este utilizat într-o gamă largă de aplicații, servicii web, sisteme de gestionare a documentelor, stocare de date și fișiere de configurație. Este adesea utilizat în combinație cu alte tehnologii, cum ar fi Simple Object Access Protocol pentru servicii web sau Extensible Hypertext Markup Language pentru continutul web.[9]

XML oferă o modalitate flexibilă și standardizată de reprezentare și schimb de date structurate. Simplitatea limbajului a contribuit la adoptarea sa pe scară largă în diverse industrii și tehnologii, făcând din acesta o parte fundamentală a gestionării și integrării moderne a datelor.

⁹ XML, 28.06.2023 <https://books.google.ro/books?id=NBwnSfoCStAC&dq=%22XML+in+a+Nutshell%22+by+Elliotte+Ru>

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#E1E0E4"
    tools:context=".MainActivity">

    <ScrollView
        android:id="@+id/scrollView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStartToStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <androidx.constraintlayout.widget.ConstraintLayout
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:layout_weight="1">

                <HorizontalScrollView
                    android:id="@+id/horizontalScrollView"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="60dp"
                    app:layout_constraintBottom_toTopOf="@+id/txtWorkout"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintHorizontal_bias="0.0"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/userName"
                    app:layout_constraintVertical_bias="0.0">

```

Figura 2.3: Sintaxa XML

Capitolul 3

Implementarea aplicației

3.1 Logica de funcționare

La deschiderea aplicației, utilizatorul este întâmpinat de o serie de întrebări pentru a afla cât mai multe despre utilizator. După ce utilizatorul răspunde la toate întrebările, un antrenament va fi creat pe baza răspunsurilor oferite dintr-o listă de exerciții fizice, care sunt cele mai optime pentru dezvoltarea mușchilor într-un timp cât mai scurt.

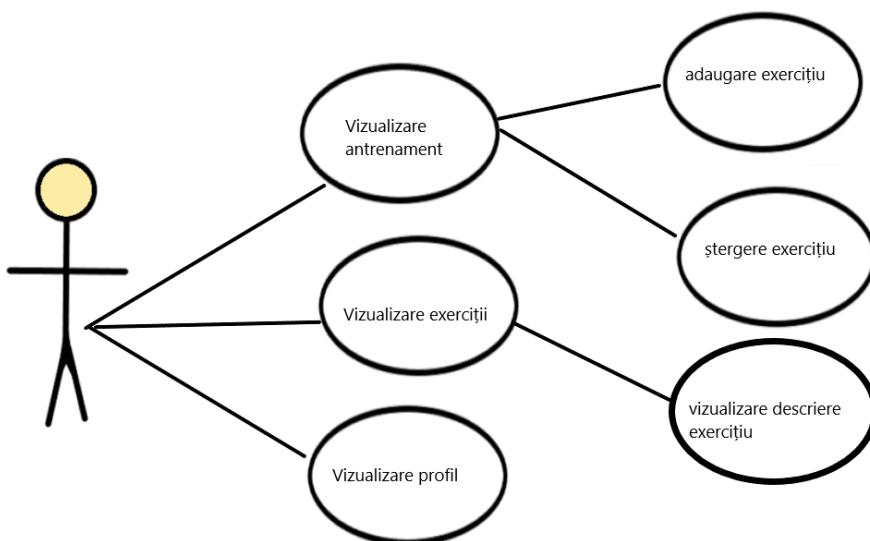


Figura 3.1: Diagrama UseCase a aplicației

3.2 Baza de date

Baza de date reprezintă componenta centrală a aplicației, asigurând stocarea și gestionarea informațiilor relevante, cum ar fi exercițiile, antrenamentul și datele despre utilizator. Această bază de date este esențială pentru funcționarea corectă a aplicației și asigură persistența datelor pe termen lung.

Odată ce aplicația este instalată pentru prima dată pe dispozitivul utilizatorului, datele despre utilizator sunt obținute prin intermediul interogărilor și sunt stocate în baza de date. Aceste date pot include informații precum vârsta, înălțimea, greutatea, nivelul de fitness și obiectivele individuale ale utilizatorului. Aceste informații sunt esențiale pentru generarea unui antrenament personalizat și adaptat nevoilor fiecărui utilizator.

Pe lângă datele despre utilizator, baza de date conține și informații despre exerciții. Exercițiile sunt introduse în baza de date după finalizarea activităților de colectare a datelor despre utilizator. Acestea pot include informații precum denumirea exercițiului, grupul muscular ţintă, instrucțiuni de execuție și nivelul de dificultate. Există o relație de tip one-to-one între utilizator și antrenament generat, ceea ce înseamnă că fiecare utilizator are un antrenament specific în baza de date.

De asemenea, există o relație de tip one-to-many între antrenament și exerciții. Aceasta înseamnă că fiecare antrenament poate conține mai multe exerciții. Prin asocierea exercițiilor cu antrenamentul specific, utilizatorul poate vizualiza și accesa exercițiile incluse în antrenamentul său personalizat.

Utilizarea unei baze de date permite aplicației să stocheze și să acceseze eficient informațiile necesare pentru generarea și personalizarea antrenamentelor. Baza de date oferă o structură organizată pentru stocarea datelor și permite interacțiunea flexibilă cu acestea în cadrul aplicației. Prin gestionarea datelor într-o bază de date, aplicația devine mai robustă și poate oferi o experiență coerentă și personalizată utilizatorilor săi.

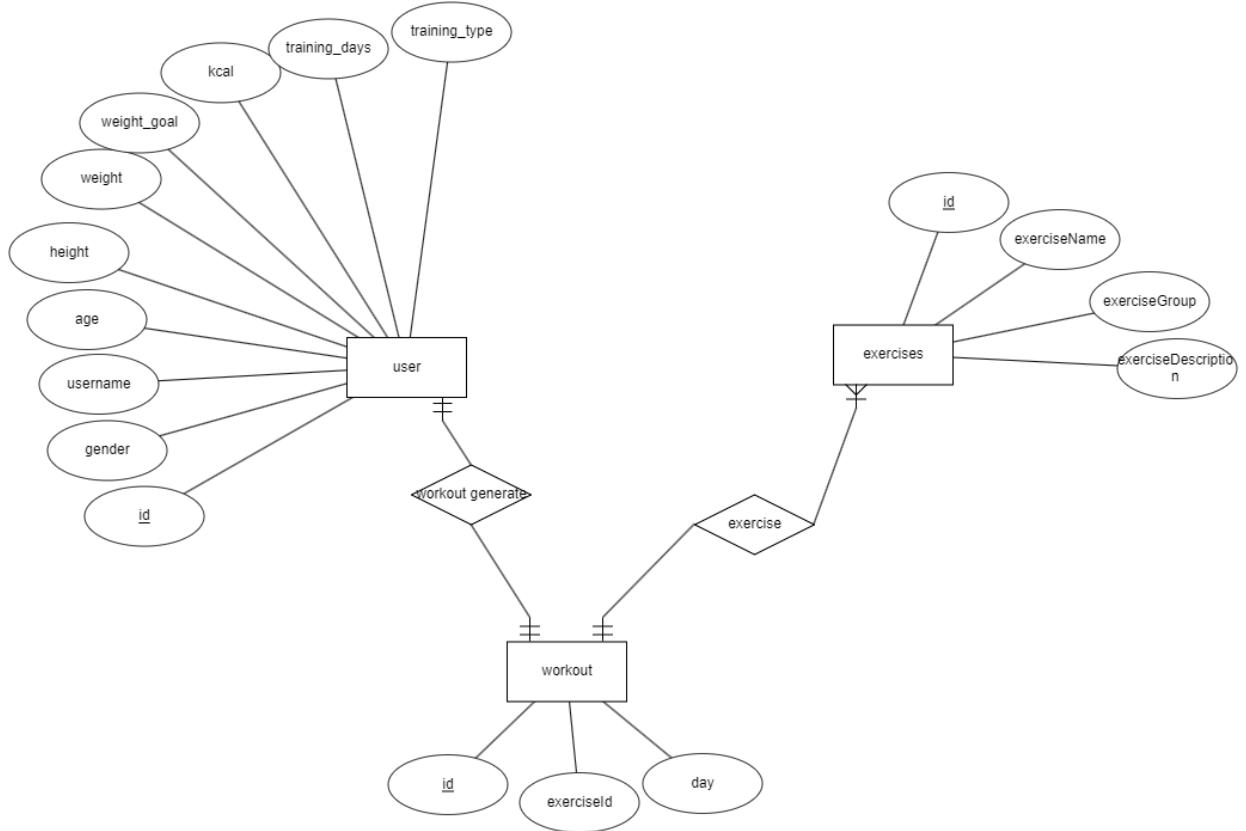


Figura 3.2: Diagrama ERD

3.3 Structura aplicației

Aplicația ”Antrenor personal virtual” este împărțită în trei mari activități prin care utilizatorul își va petrece majoritatea timpului. În MainActivity sunt afișate antrenamentele utilizatorului pentru o zi respectivă, utilizatorul putând să navigheze printre zile, și să modifice o zi respectivă prin apăsarea butoanelor de adăugare sau de ștergere. Prin apăsarea butonului de profil, se va face trecerea la ProfileActivity, unde sunt afișate datele despre cine folosește aplicația, iar cu ajutorul butonului de exercises, se va face tranziția la ExercisesActivity, unde va apărea pe ecran o listă cu toate exercițiile.

3.3.1 Activități

¹ Clasa activităților este o componentă importantă a aplicațiilor Android, modul în care sunt asamblate este o parte fundamentală a modelului de aplicație al platformei. Spre deosebire

¹ Android Developer, 27.06.2023, <https://developer.android.com/guide/components/activities/intro-activities>

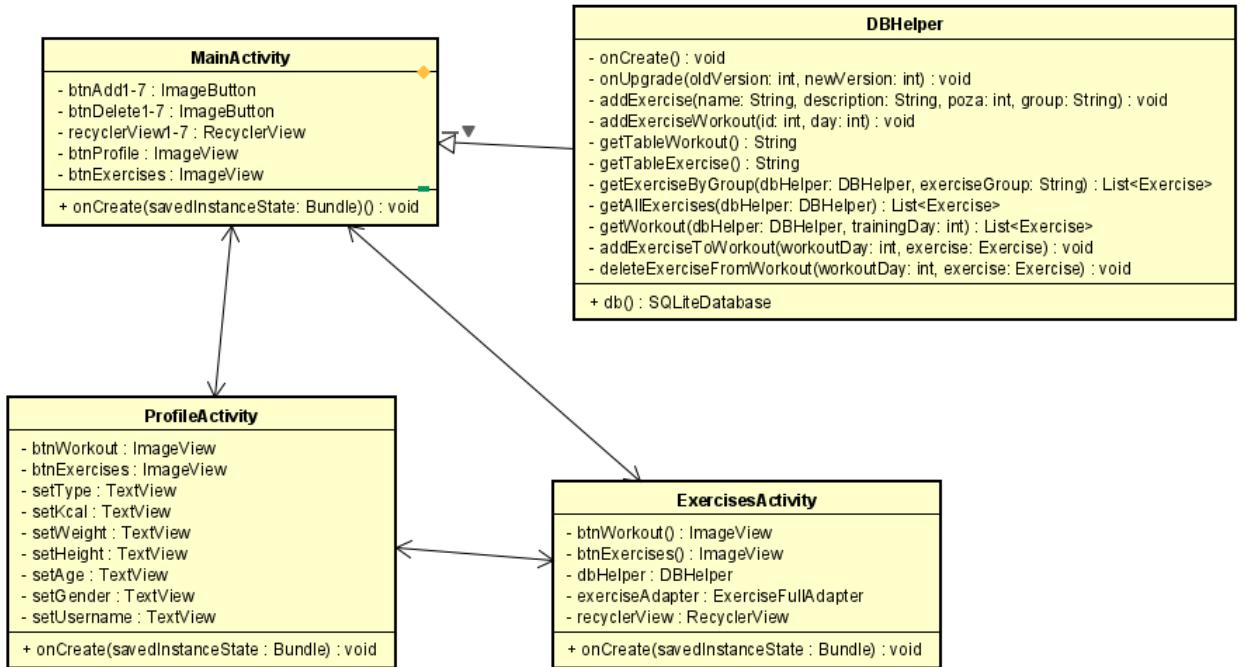


Figura 3.3: Diagrama UML a activităților importante în afișare

de alte modele de programare în care aplicațiile sunt lansate cu ajutorul funcției `main()`, sistemul Android inițiază codul într-o instantă de activitate prin invocarea unor metode specifice de callback care corespund unor etape specifice ciclului său de viață. Experiența de pe aplicațiile pentru mobil este diferită de echivalentul său de pe calculator, prin faptul că interacțiunea unui utilizator cu aplicația nu începe întotdeauna în același loc. Clasa activităților este concepută pentru a facilita aceasta paradigmă. Atunci când o aplicație invocă altă aplicație, ea de fapt invocă o activitate din cealaltă aplicație, decât să invoke aplicația ca un întreg atomic, iar în acest fel, activitatea servește ca un punct de intrare pentru interacțiunea unei aplicații cu consumatorul.[10]

Fiecare activitate în parte oferă o fereastră în care dezvoltatorul construiește interfața grafică. De exemplu în activitatea main utilizatorul vede informațiile despre antrenamentul său pe când în activitatea de profil, va vedea informațiile despre contul său. Pe lângă activitățile prezentate mai sus, acum trebuie să vorbim despre cum aplicația reține și transmite informațiile în timp real despre client.

Cu ajutorul clasei `UserData`, putem stoca datele despre utilizator preluate din activitățile ruleate la începutul aplicației, și anume, genul, numele, vârstă, înălțimea, greutatea, greutatea la care vrea să ajungă, câte zile dorește să se antreneze și tipul de antrenament. Pentru că aceste date nu sunt reținute de memorie, la finalul activității `Finish`, toate datele sunt introduse în baza de date prin intermediul clasei `DBHelper`, unde sunt toate funcțiile de manipulare a bazei de date. De asemenea, detaliile despre fiecare exercițiu sunt introduse tot la finalizarea activităților de introducere.

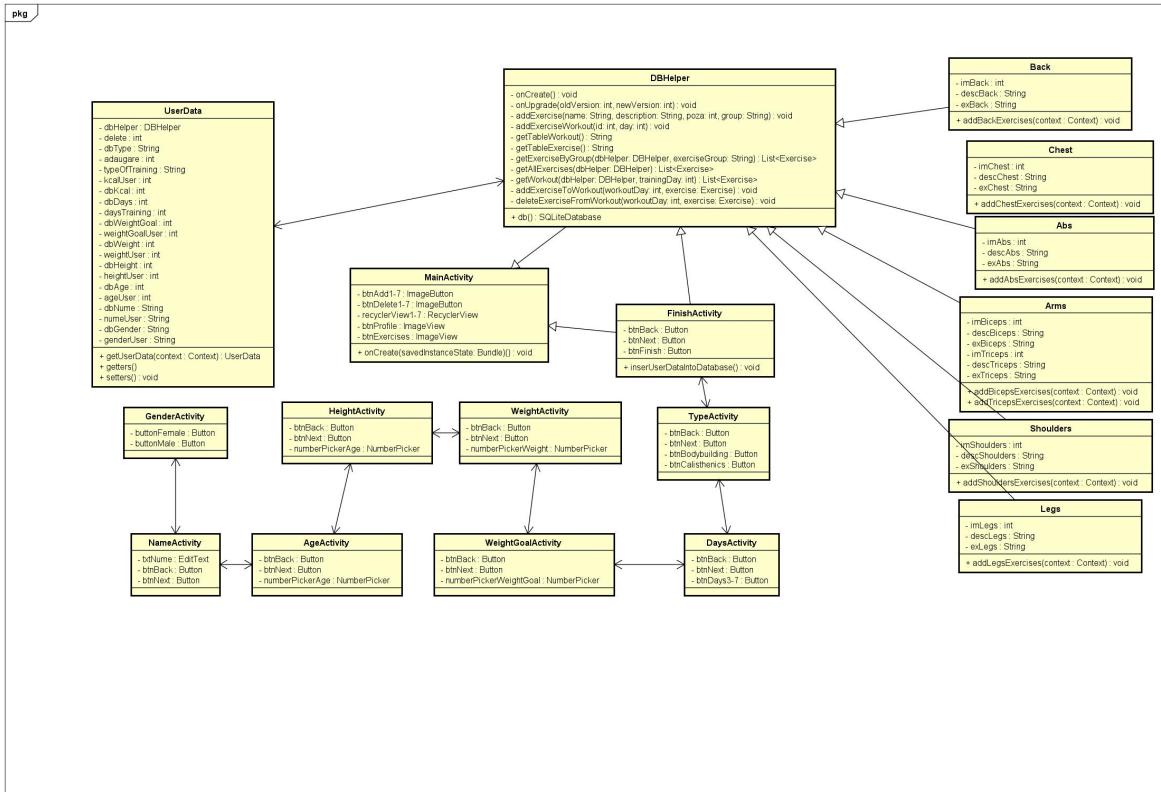


Figura 3.4: Diagrama UML a activităților

3.4 Algoritmi utilizati

În cadrul aplicației sunt utilizati o serie de algoritmi specializați pentru a asigura funcționalitatea și eficiența proceselor de desfășurare. Acești algoritmi reprezintă elementele cheie care stau la baza generării antrenamentelor și a adaptării acestora în funcție de preferințele și obiectivele utilizatorilor. Vom explora în detaliu algoritmii utilizati în aplicație și modul în care aceștia contribuie la experiența consumatorilor.

Primul algoritm pe care aș vrea să îl prezint este cel care creează baza de date, deoarece fără acest algoritm, aplicația nu ar afișa nimic. Deși sintaxa SQLite este asemănătoare cu cea din MySQL sau Access, pentru a putea genera un tabel, este nevoie mai întâi să creăm numele coloanelor. De exemplu, pentru a genera tabelul user, unde sunt stocate datele despre utilizator, trebuie să creăm string-uri pentru fiecare coloană și să le denumim:

```
public static final String TABLE_USER = "user";
public static final String COLUMN_ID = "_id";
public static final String COLUMN_GENDER = "gender";
```

După ce am generat toate coloanele unui tabel, putem să ne folosim de sintaxa SQL pentru a crea tabelul user cu ajutorul funcției

```
@Override
public void onCreate(SQLiteDatabase db) {
```

```

db . execSQL (CREATE_USER_TABLE);
db . execSQL (CREATE_EXERCISE_TABLE);
db . execSQL (CREATE_WORKOUT_TABLE);
}

```

Pentru a adăuga antrenamentul, aplicație se folosește de o funcție care are ca parametrii id-ul exercițiului și ziua respectivă în care trebuie introdus. Deoarece orice combinație de exerciții generează același rezultat în ceea ce privește atingerea telurilor utilizatorilor, funcția de generare al antrenamentului se folosește de metoda Random() a limbajului Java și se verifică dacă exercițiul a mai fost deja introdus în ziua respectivă. Pentru scheletul antrenamentului, am ales un număr fix de exerciții.

```

if (UserData . daysTraining == 3)
{
    for (int i = 0; i < 2; i++) {
        exerciseId = random.nextInt(6) + 1;
        if (!dbHelper.isExerciseAlreadyAdded(exerciseId , 1))
            dbHelper.addExerciseWorkout(exerciseId , 1);
    }

    dbHelper.addExerciseWorkout(random.nextInt(3) + 7, 1);
    dbHelper.addExerciseWorkout(random.nextInt(2) + 34, 1);

    for (int i = 0; i < 2; i++) {
        exerciseId = random.nextInt(3) + 31;
        if (!dbHelper.isExerciseAlreadyAdded(exerciseId , 1))
            dbHelper.addExerciseWorkout(exerciseId , 1);
    }

    for (int i = 0; i < 2; i++) {
        exerciseId = random.nextInt(3) + 71;
        if (!dbHelper.isExerciseAlreadyAdded(exerciseId , 1))
            dbHelper.addExerciseWorkout(exerciseId , 1);
    }

    for (int i = 0; i < 2; i++) {
        exerciseId = random.nextInt(6) + 80;
        if (!dbHelper.isExerciseAlreadyAdded(exerciseId , 1))
            dbHelper.addExerciseWorkout(exerciseId , 1);
    }

    for (int i = 0; i < 2; i++) {
        exerciseId = random.nextInt(4) + 15;
        if (!dbHelper.isExerciseAlreadyAdded(exerciseId , 2))
            dbHelper.addExerciseWorkout(exerciseId , 2);
    }
}

```

```

        for (int i = 0; i < 2; i++) {
            exerciseId = random.nextInt(6) + 80;
            if (!dbHelper.isExerciseAlreadyAdded(exerciseId, 2))
                dbHelper.addExerciseWorkout(exerciseId, 2);
        }
        dbHelper.addExerciseWorkout(random.nextInt(2) + 36, 2);
        dbHelper.addExerciseWorkout(24, 2);

        for (int i = 0; i < 2; i++) {
            exerciseId = random.nextInt(6) + 42;
            if (!dbHelper.isExerciseAlreadyAdded(exerciseId, 3))
                dbHelper.addExerciseWorkout(exerciseId, 3);
        }
    }
}

```

Un mic exemplu despre cum sunt introduse două exerciții, iar funcțiile apelate sunt definite astfel:

1. isAExerciseAlreadyAdded(exerciseId, 1)

```

public boolean isExerciseAlreadyAdded(int id, int day) {
    SQLiteDatabase db = getReadableDatabase();
    String selection = COLUMN_EXERCISE_ID_FK + " = ? AND "
        + COLUMN_DAY + " = ?";
    String[] selectionArgs = {String.valueOf(id),
        String.valueOf(day)};
    Cursor cursor = db.query(TABLE_WORKOUT, null, selection,
        selectionArgs, null, null, null);
    boolean isAdded = cursor.getCount() > 0;
    cursor.close();
    db.close();
    return isAdded;
}

```

2. addExerciseWorkout(exerciseId, 1)

```

public void addExerciseWorkout(int id, int day) {
    SQLiteDatabase db = getWritableDatabase();

    String selection = COLUMN_EXERCISE_ID_FK + " = ? AND "
        + COLUMN_DAY + " = ?";
    String[] selectionArgs = {String.valueOf(id),
        String.valueOf(day)};
    Cursor cursor = db.query(TABLE_WORKOUT, null, selection,
        selectionArgs, null, null, null);

```

```

        if (cursor.getCount() == 0) {
            ContentValues values = new ContentValues();
            values.put(COLUMN_EXERCISE_ID_FK, id);
            values.put(COLUMN_DAY, day);
            db.insert(TABLE_WORKOUT, null, values);

            cursor.close();
            db.close();
        }
    }
}

```

Prima funcție verifică dacă un exercițiu este deja adăugat în antrenament, după parametrii de id, deoarece id-ul de exerciții din tabelul workout are cheie străină către id-ul exercițiului din tabelul exercises. Pentru a putea face această operație trebuie să ne folosim de interogările bazelor de date pentru a verifica dacă mai există o înregistrare. Deschidem baza de date în modul de citire și definim structura interogării, în cazul nostru vrem să găsim înregistrarea unde id-ul exercițiul se potrivește cu id-ul primit, iar ziua din tabel cu ziua din funcție. Efectuăm interogarea pe tabelul specificat cu criteriile definite și recuperăm setul de rezultate. Se verifică dacă setul de rezultate are înregistrări. Dacă da, înseamnă că exercițiul este deja adăugat la antrenament și închidem metoda cursor, după care eliberăm resursele acestuia. Se închide conexiunea la baza de date, iar în cele din urmă returnăm o valoare de tip boolean care indică dacă exercițiul este deja adăugat sau nu.

A doua funcție adaugă exerciții în antrenament, după id și ziua respectivă. Prima oară trebuie să facem conexiunea cu baza de date, dar de data asta în modul de scriere, după care definim criteriul de selecție ca o combinație între id și zi. Efectuăm interogarea tabelului de exerciții folosind criteriile de selecție și recuperăm setul de rezultate sub forma unui cursor. Verificăm dacă numărul de înregistrări este 0, ceea ce indică faptul că nu s-au găsit înregistrări corespunzătoare. Acest lucru înseamnă că exercițiul nu a fost adăugat la antrenament. În cazul în care numărul este 0, adăugăm exercițiul în tabelul antrenament. Creăm un obiect ContentValues și îl completăm cu id-ul și ziua. Introducem valorile în tabelul de exerciții, închidem cursorul și eliberăm resursele acestuia, după care închidem conexiunea la baza de date. Pentru că metoda de interogare a bazelor de date SQLite se folosește de string-uri, este nevoie să convertim parametrii de tip int în cei de tip string cu ajutorul metodei String.valueOf().

Un alt algoritm foarte important este modul în care este afișat antrenamentul, deoarece ar fi durat zile întregi să creăm interfață grafică al unui antrenament în funcție de tipul de antrenament ales și în câte zile să fie împărtit. Pentru a optimiza totul cât mai mult, am folosit funcția de RecyclerView, pentru a afișa antrenamentele din zilele respective. Când se pornește activitatea main, este apelat un algoritm care verifică câte zile sunt în baza de date și generează atâtea zile câte sunt necesare. RecyclerView-urile sunt poziționate pe o altă funcție și anume CardView.

```

for (int i = 1; i <= 7; i++) {
    int cardViewId = getResources().getIdentifier("Day" + i,
        "id", getPackageName());
}

```

```

CardView cardView = findViewById(cardViewId);
if (i >= startDay && i <= endDay) {
    List<Exercise> exerciseList = dbHelper.getWorkout(dbHelper, i);

    if (exerciseList.isEmpty()) {
        cardView.setVisibility(View.GONE);
    }
    else {
        int recyclerViewId = getResources().getIdentifier(
            "recyclerDay"+ i, "id", getPackageName());
        RecyclerView recyclerView =
            cardView.findViewById(recyclerViewId);
        LinearLayoutManager layoutManager = new LinearLayoutManager
        recyclerView.setLayoutManager(layoutManager);

        ExerciseFullAdapter adapter = new ExerciseFullAdapter();
        recyclerView.setAdapter(adapter);
        adapter.setExercises(exerciseList);

        cardView.setVisibility(View.VISIBLE);
    }
}
else {
    cardView.setVisibility(View.GONE);
}
}

```

Clasa ExerciseFullAdapter este utilizată pentru a gestiona și afișa o listă de exerciții într-un RecyclerView. Contine o listă de obiecte de exerciții care vor fi afișate în RecyclerView. Oferă o metodă de actualizare a listei de exerciții, permitând modificări dinamice, suportă două tipuri de ascultători de evenimente: unul pentru adăugarea de exerciții și altul pentru stergerea exercițiilor. Permite înregistrarea de listener pentru a gestiona aceste evenimente.

Reprezintă o vizualizare a unui element în cadrul RecyclerView. Aceasta conține referințe la vizualizările (ImageView, TextViews, ImageButton) care afișează informații despre exerciții. Leagă datele exercițiilor de vizualizările corespunzătoare din cadrul ViewHolder, stabilind imaginea, numele și grupul de exerciții.

Afișează un dialog de descriere atunci când utilizatorul face clic pe butonul de descriere. Dialogul afișează numele și descrierea exercițiului. În funcție de valoarea lui UserData.adaugare și UserData.delete, setează ascultătorii de clicuri pe TextView cu numele exercițiului. Dacă UserData.adaugare este mai mare decât 0, dacă se face clic pe numele exercițiului se declanșează evenimentul de adăugare a exercițiului. Dacă UserData.delete este mai mare decât 0, dacă se face clic pe numele exercițiului se declanșează evenimentul de stergere a exercițiului. Se definesc interfețele pentru ascultătorii de evenimente de adăugare și de stergere a exercițiilor. Aceste interfețe specifică metodele care trebuie implementate de către ascultători pentru a

gestiona evenimentele respective.

Pe scurt, clasa ExerciseFullAdapter se ocupă de gestionarea datelor, de crearea vizualizării și de gestionarea evenimentelor pentru afișarea exercițiilor într-un RecyclerView. Aceasta oferă metode și interfețe convenabile pentru a interacționa cu adaptorul și pentru a răspunde la acțiunile utilizatorului, cum ar fi adăugarea și ștergerea exercițiilor.

3.5 Interfață

Interfața reprezintă elementul central al experienței oferite și din această cauză am investit destul de mult timp pentru a crea o interfață prietenoasă.

Pentru a crea o experiență vizuala atrăgătoare, am selectat o paletă de culori potrivite pentru aplicație. Culorile acestea au fost alese pentru a oferi o experiență de neuitat și totodată lizibilitate. Totodată, am acordat și o atenție deosebită designului user-friendly, asigurându-mă că fluxul de navigare și interacțiune în aplicație este ușor de înțeles. Interfața a fost proiectată să se adapteze la toate dimensiunile de ecran și rezoluții.

Prin intermediul butoanelor de Back și Next, utilizatorii pot verifica dacă și-au introdus datele corect în activitățile de pornire, iar în activitatea principală sunt întâmpinați de alte 4 butoane, fiecare dintre ele fiind reprezentate de o activitate nouă

- ▷ Vizualizarea exercițiilor
- ▷ Vizualizarea profilului
- ▷ Adăugarea de exerciții
- ▷ Ștergerea unui exercițiu

La apăsarea uneia dintre butoane, se va face tranzitia la activitatea respectivă, la apăsarea butonului de exerciții va apărea lista cu toate exercițiile aplicației, la apăsarea profilului vor apărea date despre utilizator, iar la celelalte două butoane, după ce vor fi trimiși la activitatea respectivă, la apăsarea pe numele exercițiului cu ajutorul interogărilor tabelului workout, exercițiul va fi adăugat sau șters, revenindu-se apoi la activitatea principală.

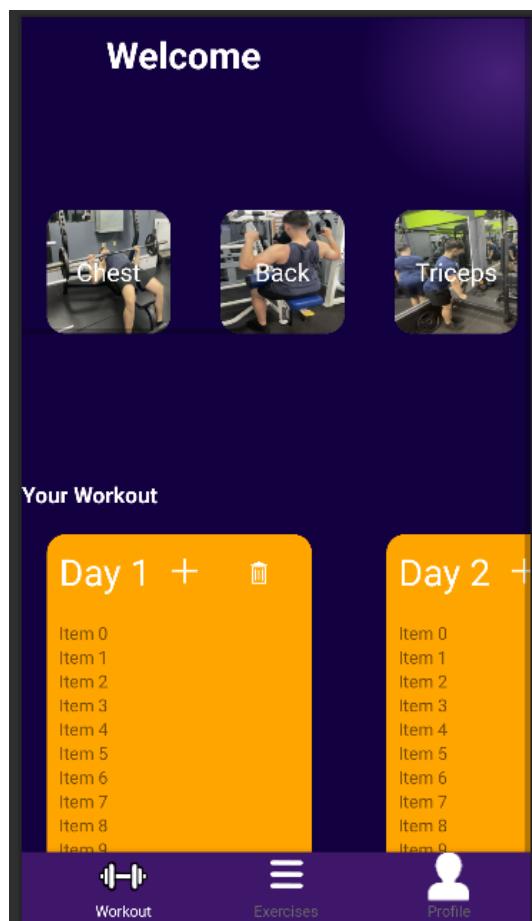


Figura 3.5: Scheletul interfeței grafice main

3.6 Analiza SWOT

O analiză SWOT este un instrument de planificare strategică utilizat pentru a evalua punctele forte, punctele slabe, oportunitățile și amenințările unei afaceri, ale unui proiect sau ale unei idei. Aceasta oferă o evaluare cuprinzătoare a factorilor interni și externi care pot avea un impact asupra succesului sau eșecului entității analizate.

Prin efectuarea unei analize SWOT, întreprinderile și persoanele fizice înteleg mai bine situația lor actuală și pot lua decizii în cunoștință de cauză cu privire la strategiile, obiectivele și acțiunile lor. Aceasta ajută la identificarea domeniilor de îmbunătățire, a potențialelor oportunități de creștere și a riscurilor care trebuie atenuate. Analiza poate fi utilizată în diverse scopuri, cum ar fi elaborarea de planuri de afaceri, evaluarea de noi proiecte, evaluarea viabilității pieței sau îmbunătățirea competitivității.

Strengths:	Weaknesses:
<ul style="list-style-type: none"> -funcționalități unice -antrenament customizabil -ușor de utilizat -conține cele mai bune exerciții 	<ul style="list-style-type: none"> -aplicația nu are un mod de a reține numărul de set-uri sau repetări -nu oferă explicații video -nu oferă un mod de a adăuga exerciții proprii
Opportunities:	Threats:
<ul style="list-style-type: none"> -este o piață care crește în continuu -colaborări cu brand-uri fitness -integrarea cu platforme sau tehnologii populare. 	<ul style="list-style-type: none"> -competiție mare -tehnologiile se schimbă rapid și va trebui să fac modificări în continuu -schimbarea preferințelor utilizatorilor sau schimbarea tendințelor pieței.

Figura 3.6: Analiza SWOT

Capitolul 4

Utilizarea aplicației

La deschiderea aplicației, utilizatorul este întâmpinat de o experiență interactivă, fiind invitat să completeze o serie de întrebări pentru a furniza informații relevante despre el. Scopul acestor întrebări este de a personaliza antrenamentul în funcție de nevoile și preferințele fiecărui utilizator. Pe baza răspunsurilor oferite, aplicația generează un antrenament personalizat, alcătuit dintr-o listă selectată de exerciții fizice. Aceste exerciții sunt riguros alese pentru a maximiza dezvoltarea musculară într-un timp cât mai scurt, în concordanță cu obiectivele declarate de utilizator.

Un aspect important al aplicației este flexibilitatea oferită utilizatorului. Dacă un exercițiu nu este pe placul utilizatorului sau acesta nu poate efectua anumite exerciții din motive de sănătate, aplicația permite înlocuirea exercițiului respectiv cu o alternativă adecvată, astfel încât utilizatorul să-și poată adapta antrenamentul la nevoile și preferințele sale din ziua respectivă.

În funcție de frecvența antrenamentelor selectată de utilizator (de exemplu, 3 zile pe săptămână), aplicația generează un plan de antrenament cu o structură standardizată. De exemplu, în prima zi, utilizatorul va efectua un antrenament focalizat pe exerciții de împingere, în a doua zi se va concentra pe exerciții de tragere, iar în a treia zi va fi dedicată exercițiilor pentru dezvoltarea musculaturii picioarelor. Această abordare structurată permite optimizarea timpului dedicat antrenamentului și ajută utilizatorul să-și atingă obiectivele mai rapid. Prin intermediul acestei aplicații, am urmărit să ofer utilizatorilor posibilitatea de a beneficia de un program de antrenament personalizat și eficient, adaptat nevoilor lor individuale. Am dorit să facilitez optimizarea timpului petrecut în sala de antrenament și să le ofer utilizatorilor o modalitate rapidă și eficientă de a-și atinge obiectivele de fitness.

4.1 Scenariu utilizare

Pentru a putea exemplifica cum se utilizează aplicația, vom presupune existența unui utilizator pe nume Cristi. În momentul pornirii aplicației, este declanșat activitatea de pornire.

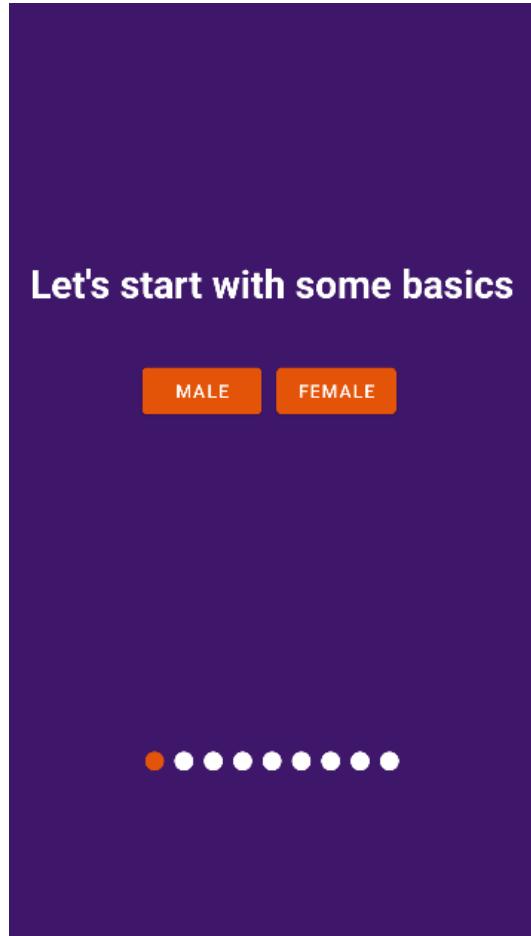


Figura 4.1: Activitatea de pornire a aplicației

La primirea răspunsurilor, activitățile vor decurge în ordine, iar dacă Cristi consideră ca a greșit o întrebare, se poate întoarce din nou la activitățile din urmă pentru a modifica răspunsurile. După terminarea activităților de introducere, el va fi trimis către main, unde va primi antrenamentul.

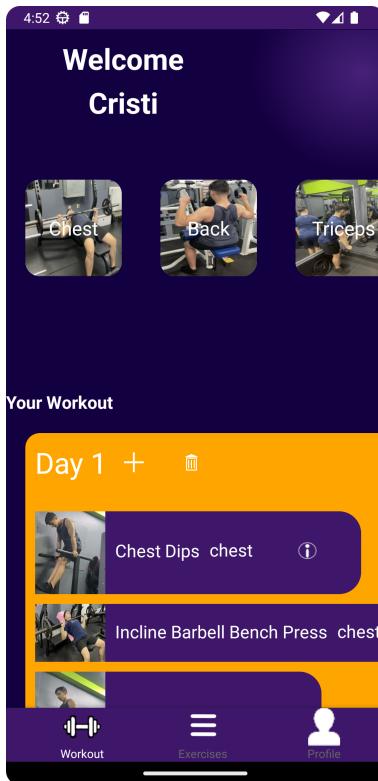


Figura 4.2: Fereastra main a aplicației

După verificarea antrenamentului, Cristi poate să-și adauge exerciții apăsând pe butonul de adăugare "+" din dreapta zilei respective, de unde se deschide fereastra de exerciții, unde va căuta exercițiul pe care dorește să îl introducă în zi. Tot odată, poate să steargă un exercițiu, de exemplu Cristi are o problemă la umărul stâng, iar doctorul i-a spus să nu facă exerciții care pun stres pe umeri, iar dips-urile sunt un bun exemplu pentru asta deoarece trebuie să ai forță în umeri pentru a putea executa corect mișcarea, el trebuie doar să apese pe butonul de gunoi din dreapta butonului de adăugare, de unde se va deschide o fereastră cu toate exercițiile din ziua respectivă

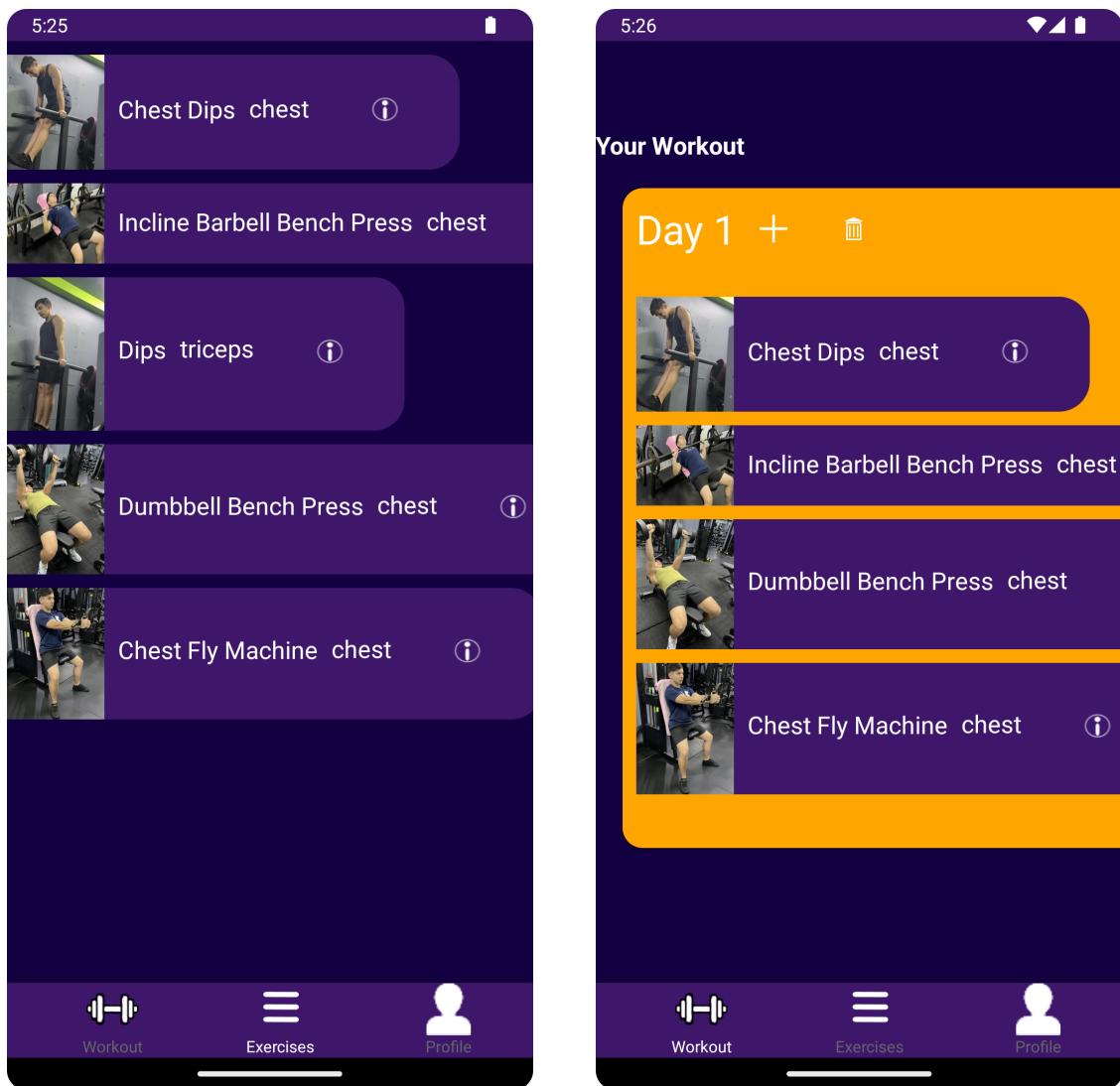


Figura 4.3: Activitatea de stergere a unui exercițiu

Să spunem că Cristi vede pentru prima oară exercițiul: Fluturări de scripeți la piept, cu ajutorul butonului de descriere, pe ecran va apărea un text despre cum să execute mișcarea.

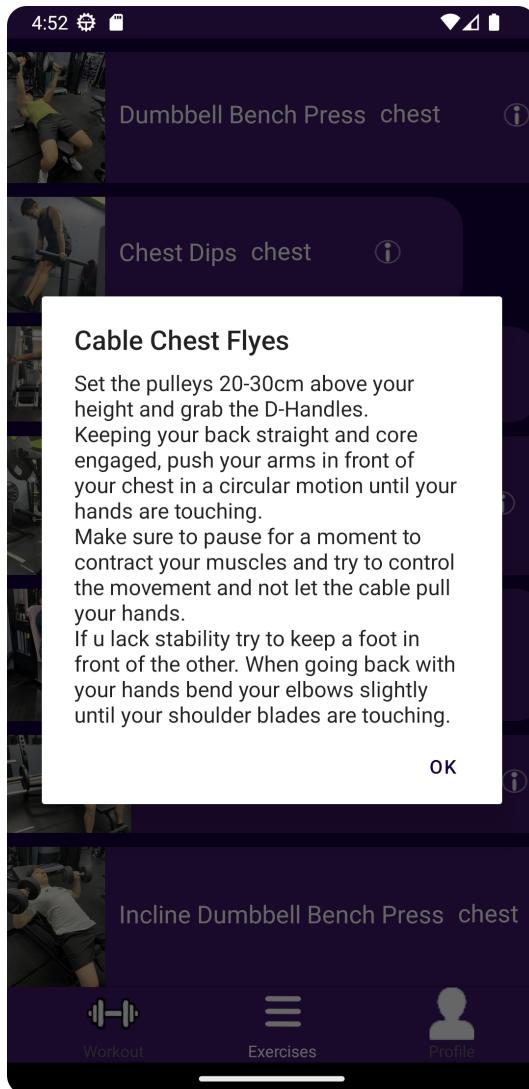


Figura 4.4: Descriere exercițiu

4.2 De ce să utilizăm aplicații fitness

Importanța mișcării și a exercițiilor fizice în viața noastră nu poate fi subestimată. Indiferent de vârstă sau de nivelul de condiție fizică, activitatea fizică regulată aduce numeroase beneficii pentru sănătatea noastră fizică și mentală.

În primul rând, exercițiile fizice contribuie la menținerea unei stări generale bune de sănătate. Prin intermediul activității fizice regulate, corpul nostru devine mai puternic și mai rezistent. Sistemul cardiovascular este stimulat, iar fluxul sanguin este îmbunătățit, ceea ce reduce riscul de boli cardiovasculare, precum hipertensiunea arterială, bolile de inimă și ac-

cidentele vasculare cerebrale. De asemenea, exercițiile fizice ajută la menținerea unui nivel sănătos de colesterol și la controlul greutății corporale, prevenind astfel apariția obezității și a afecțiunilor asociate, cum ar fi diabetul de tip 2.

Pe lângă beneficiile pentru sănătatea fizică, mișcarea are un impact semnificativ și asupra sănătății mintale. Exercițiile fizice eliberează endorfine, cunoscute și sub denumirea de hormoni ai fericirii, care ne produc o stare de bine și ne ajută să ne relaxăm. Practicarea regulată a exercițiilor fizice poate reduce stresul, anxietatea și depresia, îmbunătățind starea noastră de spirit și creând un sentiment general de echilibru și optimism.

În plus, mișcarea și exercițiile fizice contribuie la îmbunătățirea capacitaților cognitive. Studiile arată că activitatea fizică stimulează creierul, îmbunătățind memoria, atenția și capacitatea de concentrare. De asemenea, exercițiile fizice regulate sunt asociate cu un risc mai scăzut de apariție a bolilor neurodegenerative, precum Alzheimer și Parkinson. Un alt beneficiu al exercițiilor fizice este creșterea nivelului de energie și îmbunătățirea calității somnului. Chiar dacă poate părea paradoxal, mișcarea ne face să ne simțim mai energici și mai în formă, deoarece stimulează circulația sanguină și oxigenarea organismului. Totodată, exercițiile fizice regulate ne ajută să obținem un somn mai odihnitor, favorizând relaxarea și reducând insomniile.

Nu în ultimul rând, exercițiile fizice pot fi o ocazie excelentă pentru socializare și pentru crearea de noi legături sociale. Participarea la activități de grup, cum ar fi sporturile de echipă sau clasele de fitness, ne oferă posibilitatea de a interacționa cu alți oameni, de a împărtăși aceleași interese și de a ne bucura de momente pline de distractie și motivație împreună.

În concluzie, beneficiile exercițiilor fizice sunt incontestabile. Prin mișcare și activitate fizică regulată, ne putem menține sănătatea, ne putem îmbunătăți starea de spirit, putem avea o minte mai ageră și un somn mai odihnitor. Indiferent de vîrstă sau de nivelul de condiție fizică, oricine poate găsi modalități adecvate de a se bucura de beneficiile aduse de exercițiile fizice. Așadar, să facem mișcare și să aducem un plus de sănătate și bucurie în viața noastră!

Capitolul 5

Concluzii

Dezvoltarea aplicației ”Antrenor Personal Virtual” a necesitat utilizarea cunoștințelor practice și teoretice învățate în facultate, cum ar fi structuri de date, baze de date, programare orientată pe obiecte, dar și învățatului pe parcurs, limbaj xml, cum să utilizez IDE-ul Android Studio, sintaxa SQLite.

Prin intermediul generării automate a antrenamentului, aplicația asigură o abordare eficientă și bine structurată a exercițiilor fizice, luând în considerare informațiile furnizate de utilizator în cadrul întrebărilor initiale. Această funcționalitate ajută utilizatorii să economisească timp și efort în planificarea și organizarea antrenamentelor, oferindu-le în schimb un program de exerciții optim pentru dezvoltarea musculară într-un timp cât mai scurt.

Posibilitatea de a vizualiza, adăuga sau elimina exerciții din antrenament conferă utilizatorilor un grad ridicat de control și personalizare. Această funcționalitate permite ajustarea antrenamentului în funcție de preferințele individuale și nivelul de confort al utilizatorului. Astfel, utilizatorii pot adapta antrenamentul în timp real, fiind capabili să-și atingă obiectivele specifice și să rămână motivați în cadrul programului lor de fitness.

Utilizatorii au posibilitatea de a avea un partener virtual de antrenament, care să le ofere ghidare și suport în parcursul lor de fitness. Aceasta nu doar îi ajută să-și atingă obiectivele de dezvoltare musculară sau de slăbire, ci și să-și îmbunătățească starea de sănătate.

Referințe bibliografice

- [1] Android. Android (operating system). 16.12.2022. [https://source.android.com/.](https://source.android.com/)
- [2] Android, 28.06.2023. [https://source.android.com/docs/core/architecture.](https://source.android.com/docs/core/architecture)
- [3] Google Play, 17.12.2022. [https://play.google.com/console/about/playgammeservices/?gclid=Cj0KCQiA14WdBh%20D8ARIsANao07ixcm5MKCTWgHw0um2O8Pgh3ICW%20iavc9u2j6TTu15JfK-%20hy6tdcuAaAq8mEALw%20wCB.](https://play.google.com/console/about/playgammeservices/?gclid=Cj0KCQiA14WdBh%20D8ARIsANao07ixcm5MKCTWgHw0um2O8Pgh3ICW%20iavc9u2j6TTu15JfK-%20hy6tdcuAaAq8mEALw%20wCB)
- [4] Android, 07.01.2023.
- [5] Gradle Enterprise, 07.01.2023. [https://gradle.com/who-we-are/?ga=2.71212880.1918857816.1673301538-388143168.1673301538.](https://gradle.com/who-we-are/?ga=2.71212880.1918857816.1673301538-388143168.1673301538)
- [6] SQLite, 08.01.2023. [https://www.sqlite.org/about.html.](https://www.sqlite.org/about.html)
- [7] Java, 28.06.2023. [https://www.java.com/en/download/help/whatis_java.html.](https://www.java.com/en/download/help/whatis_java.html)
- [8] Patricia Bomme Thomas Zimmermann, Yves Dubois-Pèlerin. *Object-oriented finite element programming: I. Governing principles, Computer Methods in Applied Mechanics and Engineering*,. 1992, 28.06.2023.
- [9] W. Scott Means Elliotte Rusty Harold. *XML in a Nutshell: A Desktop Quick Reference*. 2004, 28.06.2023.
- [10] Android, 27.06.2023. [https://developer.android.com/guide/components/activities/intro-activities.](https://developer.android.com/guide/components/activities/intro-activities)