**Documentul de proiectare**

**Aplicație android pentru gestionarea programelor de fitness și alimentație**

Student:Bratu Cosmin-Mihai

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

**1. Introducere**

**1.1 Scopul documentului**

Acest document de proiectare descrie structura arhitecturală, deciziile tehnice și componentele principale ale aplicației mobile de fitness destinată monitorizării antrenamentelor și alimentației. Aplicația este concepută pentru a oferi o experiență personalizată, adaptată obiectivelor fiecărui utilizator, și integrează funcționalități precum generarea automată a planurilor de antrenament, urmărirea progresului și calculul aportului caloric. Documentul are scopul de a ghida procesul de implementare prin oferirea unei viziuni clare asupra organizării interne a sistemului, tehnologiilor utilizate și modului în care sunt gestionate interacțiunile dintre componentele aplicației. Designul este orientat spre modularitate, scalabilitate și ușurință în întreținere, asigurând astfel baza pentru o dezvoltare eficientă și extensibilă.

**2. Prezentare generală și abordări de proiectare**

Proiectul constă în dezvoltarea unei aplicații mobile cross-platform destinată utilizatorilor care doresc să-și monitorizeze antrenamentele și alimentația într-un mod centralizat și adaptat obiectivelor personale. Aplicația se bazează pe o abordare modulară, cu funcționalități împărțite pe secțiuni (Home, Workouts, Calories), oferind astfel o experiență clară și coerentă. Accentul cade pe personalizarea planurilor de antrenament și a obiectivelor nutriționale, adaptate pe baza unor factori inițiali colectați de la utilizator.

**2.1. Prezentare generală**

Aplicația este structurată în jurul unui proces de onboarding, urmat de o interfață intuitivă ce permite accesul rapid la antrenamente și monitorizarea nutriției. Datele introduse de utilizator sunt procesate prin logica aplicată în backend, care generează automat sugestii de planuri și calcule nutriționale. Interacțiunea dintre frontend și backend se face prin apeluri API și sincronizare cu baza de date gestionată prin Appwrite. Scopul general este de a livra un produs ușor de folosit, dar suficient de complex pentru a acoperi nevoile utilizatorilor activi.

**3. Considerații de proiectare**

Dezvoltarea aplicației se bazează pe principii clare de modularitate, reutilizare a componentelor și separare a logicii de business față de interfața utilizatorului. Designul general al aplicației urmărește o experiență prietenoasă, în care utilizatorul să poată interacționa rapid și intuitiv cu funcțiile esențiale. Fiecare tab are un scop bine definit, iar navigarea între secțiuni se realizează ușor, fără a necesita acțiuni complexe. Logica de generare automată a antrenamentelor și a obiectivelor calorice este separată de vizualizarea efectivă a acestora, asigurând astfel scalabilitate și întreținere facilă.

**3.1 Obiective și linii directoare (ghiduri)**

Obiectivul principal al aplicației este de a susține utilizatorul în atingerea obiectivelor sale de fitness și alimentație printr-o soluție digitală eficientă, ușor de utilizat și personalizabilă. Proiectarea urmărește ghiduri moderne de UX/UI pentru aplicații mobile, folosind componente reactive, feedback vizual constant și evitarea suprasolicitării utilizatorului cu informații. Structura codului respectă principiile SOLID și se va utiliza un sistem de versiune (ex: Git) pentru a asigura controlul schimbărilor și colaborarea eficientă în echipă.

**3.2 Metode de dezvoltare**

Proiectul va fi dezvoltat utilizând metoda Agile Scrum, cu sprinturi săptămânale și task-uri organizate în Jira. Planificarea se va face pe baza prioritizării funcționalităților-cheie, cu accent pe livrarea unui produs funcțional minimal (MVP) cât mai devreme. Tehnologiile folosite includ React Native pentru aplicația mobilă, Node.js pentru logica backend și Appwrite pentru autentificare și salvarea datelor. Testarea se va realiza incremental, atât manual, cât și automatizat pentru funcțiile critice..

**4. Arhitectura Sistemului și Proiectarea Arhitecturii**

Arhitectura sistemului propus este centrată pe o aplicație mobilă multiplatformă ce oferă utilizatorilor funcționalități extinse de monitorizare a activității fizice și nutriționale. Structura arhitecturală este de tip client-server, unde aplicația mobilă, realizată în React Native, acționează ca interfață principală pentru utilizator, iar partea de backend este gestionată printr-un server Node.js, în combinație cu platforma Appwrite pentru autentificare, gestionarea utilizatorilor și manipularea datelor. Comunicarea între client și server se face prin API-uri REST, oferind flexibilitate și scalabilitate în dezvoltare.

**4.1 Vedere logică**

Vederea logică a aplicației reflectă o separare clară a responsabilităților între componentele client și server. Aplicația mobilă conține logica de interfață cu utilizatorul și gestionează interacțiunile directe, în timp ce serverul procesează cererile și se ocupă de logica de business, stocarea și validarea datelor, precum și calculul obiectivelor calorice sau gestionarea antrenamentelor. Această separare permite o organizare clară și modulară a codului și facilitează întreținerea și extinderea ulterioară a aplicației.

**4.2 Arhitectură hardware**

Aplicația este destinată dispozitivelor mobile Android, astfel că cerințele hardware sunt minimale, fiind compatibilă cu majoritatea telefoanelor moderne. Serverul backend poate fi găzduit pe o infrastructură cloud sau locală, în funcție de nevoi. Pentru testare și dezvoltare, aplicația este rulată pe emulator Android și dispozitive fizice, iar serverul Node.js rulează pe o stație de lucru cu procesor multicore, memorie RAM de minim 8GB și conexiune stabilă la internet.

**4.3 Arhitectură software**

Pe partea de software, aplicația mobilă este dezvoltată în React Native, utilizând biblioteci moderne pentru navigație, formulare, stocare locală și interacțiuni UI. Backend-ul este dezvoltat în Node.js, iar Appwrite este folosit ca platformă pentru gestionarea utilizatorilor, autentificare, stocare de fișiere și baze de date. Comunicarea între componente se face prin API-uri REST. Arhitectura software este gândită în mod modular, astfel încât fiecare funcționalitate majoră (ex: tracking calorii, gestionare antrenamente, scanare etichete) este gestionată independent.

**4.4 Arhitectura informațiilor**

Arhitectura informațiilor presupune organizarea clară a datelor colectate și prezentate utilizatorului. Datele sunt împărțite în mai multe colecții: utilizatori, antrenamente (predefinite și custom), exerciții, rapoarte zilnice de calorii, alimente consumate, obiective. Datele introduse de utilizator sunt stocate și pot fi accesate în funcție de dată, iar sistemul asigură consistența și sincronizarea acestora între client și server. Se acordă o atenție deosebită păstrării istoricului activităților și nutriției pentru o urmărire ușoară în timp.

**4.5 Arhitectura de comunicații interne**

Comunicarea internă dintre componente se realizează prin apeluri HTTP către API-urile oferite de serverul Node.js și Appwrite. React Native utilizează biblioteci precum Axios pentru a trimite cereri și a primi răspunsuri, în timp ce Appwrite gestionează autentificarea și accesul la baze de date. Datele sunt transmise în format JSON, iar fiecare răspuns al serverului este tratat prin intermediul promisiunilor sau funcțiilor asincrone. Această arhitectură permite o comunicare rapidă, clară și ușor de gestionat.

**5. Proiectarea sistemului**

Proiectarea sistemului este realizată având în vedere ușurința în utilizare, modularitatea și posibilitatea extinderii. Toate funcționalitățile aplicației sunt gândite în jurul experienței utilizatorului, care trebuie să poată accesa rapid și intuitiv toate secțiunile importante: antrenamentele, alimentația și obiectivele.

**5.1 Proiectarea bazei de date**

Baza de date este organizată în colecții specifice, fiecare cu un rol bine definit. Utilizatorii au un document unic ce conține informațiile personale și obiectivele, iar fiecare zi are asociate înregistrări de tip jurnal alimentar și antrenamente efectuate sau planificate. Structura este gândită astfel încât să permită căutarea rapidă și filtrarea pe criterii diverse (ex: după dată, după tip de activitate).

**5.1.1 Obiecte de date și structuri de date rezultante**

Principalele obiecte de date includ: User, Workout, Exercise, CustomWorkout, FoodEntry, DailyReport. Fiecare dintre acestea conține atribute esențiale (ex: Exercise include nume, serii, repetări, greutate, timp estimat etc.). Structurile de date sunt modelate sub formă de documente JSON în Appwrite și sunt ușor extensibile.

**5.1.2 Fișiere și baze de date**

Datele sunt gestionate prin baza de date NoSQL oferită de Appwrite, care stochează documentele structurate pe colecții. Pentru fișiere (etichete scanate, eventual imagini), este folosit storage-ul integrat Appwrite, care permite upload și gestionare securizată.

**5.2 Conversii de date**

Conversiile de date sunt necesare atunci când, de exemplu, utilizatorul introduce gramajul unui aliment, iar sistemul trebuie să convertească valorile nutriționale la porția respectivă. De asemenea, timpul introdus manual este convertit în format standardizat pentru a putea fi comparat sau sumarizat zilnic/săptămânal. Totodată, unele date de intrare trebuie validate sau normalizate pentru a fi compatibile cu cerințele aplicației (ex: valori calorice, număr serii).

**5.3 Interfețe utilizator**

Interfața utilizatorului este gândită să fie intuitivă, rapidă și eficientă, adaptată pentru utilizatorii activi și orientați spre rezultate. Cele trei secțiuni principale (Home, Workouts și Calories) sunt accesibile dintr-un meniu de tip tab-bar persistent.

**5.3.1 Intrări**

Utilizatorii introduc date prin câmpuri de text, selectoare numerice, formulare de răspuns la întrebări inițiale, și prin scanarea codurilor de bare sau a etichetelor alimentare. În zona de workouts, aceștia pot adăuga sau modifica exerciții, iar în zona de alimentație pot introduce mese și cantități.

**5.3.2 Ieșiri**

Aplicația afișează informații sintetizate precum lista de antrenamente zilnice, obiective calorice rămase, progres săptămânal și sumar de macronutrienți. Totodată, oferă feedback vizual în timp real (ex: timer activ, progres pe zi, indicatori de atingere a obiectivelor).

**5.4 Proiectarea interfețelor cu utilizatorul**

Designul este orientat spre claritate și utilizare rapidă, cu componente mobile friendly, culori echilibrate și feedback vizual consistent. Aplicația folosește un sistem de design bazat pe carduri, butoane clare și un calendar interactiv pentru planificarea antrenamentelor.

**6. Scenarii de utilizare**

Un scenariu tipic presupune ca utilizatorul să instaleze aplicația, să răspundă la întrebările inițiale, iar apoi să primească un plan personalizat de antrenamente și un obiectiv caloric zilnic. În fiecare zi, utilizatorul poate vizualiza antrenamentele, începe unul în timp real sau adăuga altele personalizate, iar în tab-ul de calorii poate introduce alimentele consumate sau scana etichete. La final de zi, poate vedea progresul realizat și dacă și-a atins obiectivele.

**7. Proiectare de detaliu**

**7.1 Proiectare hardware de detaliu**

Sistemul este optimizat pentru a rula pe dispozitive Android cu minim Android 8.0 și resurse hardware modeste. Serverul Node.js poate rula pe o mașină virtuală sau un serviciu cloud, iar Appwrite necesită o instanță Docker cu resurse minime (2 vCPU, 4GB RAM).

**7.2 Proiectare software de detaliu**

Frontend-ul React Native este structurat pe componente funcționale, cu utilizarea bibliotecii react-navigation pentru tab-uri și stack-uri, iar datele sunt gestionate prin context API sau librării precum Redux dacă este necesară o scalare ulterioară. Backend-ul Node.js gestionează cererile REST și rulează logica de business pentru generarea antrenamentelor, calculul caloriilor și logarea activităților.

**7.3 Proiectare detaliată de securitate**

Accesul la date este controlat prin sistemul de permisiuni oferit de Appwrite, iar autentificarea utilizatorilor se face prin email și parolă. Token-urile de sesiune sunt folosite pentru a securiza accesul la resurse, iar datele sensibile sunt criptate atât în tranzit cât și la stocare.

**7.4 Proiectare de detaliu pentru performanța sistemului**

Pentru a menține performanța ridicată, datele sunt încărcate asincron, interfețele sunt optimizate prin lazy loading, iar backend-ul folosește cache unde este posibil. Se evită apelurile redundante și se folosește paginarea pentru afișarea listelor mari.

**7.5 Proiectare detaliată a comunicațiilor interne (între componente)**

Componentele aplicației comunică printr-o arhitectură RESTful: frontend-ul trimite cereri HTTP către backend și Appwrite, care returnează răspunsuri în format JSON. Fluxul de date este bidirecțional, controlat prin middleware pentru autentificare, validare și procesare de răspunsuri.