**KaizenX - Structura generală a aplicației**

**• KaizenXApp -** servește ca punct de intrare pentru aplicația KaizenX, inițializând și configurând elementele necesare pentru ca aplicația să funcționeze corect.

**• AppDelegate**: Subclasa care implementează protocolul UIApplicationDelegate. Aici se efectuează configurarea inițială a Firebase, asigurându-se că toate serviciile dependente de Firebase sunt pregătite pentru utilizare imediat ce aplicația este lansată.

**• FirebaseApp.configure()**: Această linie de cod este esențială și se execută odată ce aplicația termină lansarea. Ea inițializează și configurează toate serviciile Firebase utilizate în aplicație, cum ar fi autentificarea, stocarea datelor și analitica.

**• KaizenXApp**: Structura principală a aplicației care definește conținutul scenei. Utilizează UIApplicationDelegateAdaptor pentru a înregistra AppDelegate ca delegat al aplicației, permițând configurarea Firebase.

**• WindowGroup**: O scenă care conține ContainerView, care este afișată în fereastra principală a aplicației. ContainerView funcționează ca un comutator între ecranul de încărcare și conținutul principal al aplicației, bazându-se pe starea internă pentru a decide ce să afișeze.

**• Authentication -**  componentă cheie care gestionează procesul de acces al utilizatorilor în aplicație. Acest modul conține o serie de fișiere și clase care lucrează împreună pentru a oferi o experiență de autentificare sigură și eficientă.

**• SignInEmailView**: O interfață SwiftUI care permite utilizatorilor să se autentifice folosind adresa de email. Include câmpuri de text pentru introducerea email-ului și parolei și butoane pentru a iniția procesul de autentificare sau pentru a naviga către ecranul de înregistrare.

**• SignInEmailViewModel**: ViewModel-ul care susține SignInEmailView, menținând logica necesară pentru a procesa autentificarea prin email, gestionând starea vizuală a interfeței de autentificare.

**• SignUpView**: View-ul pentru înregistrarea unui nou utilizator, care oferă câmpuri pentru informațiile necesare înregistrării și declanșează procesul de creare a unui nou cont.

**• SignUpViewModel**: ViewModel-ul pentru SignUpView, care conține logica pentru a înregistra un nou utilizator în sistem, verificând validitatea datelor introduse și comunicând cu Firebase pentru a înregistra utilizatorul.

**• AuthenticationManager**: Clasa centrală a modulului de autentificare, care gestionează toate operațiunile legate de autentificare, inclusiv înregistrarea, conectarea și deconectarea utilizatorilor, precum și gestionarea datelor de autentificare.

**• SignInGoogleHelper**: O clasă ajutătoare care facilitează autentificarea cu Google, gestionând interacțiunea cu SDK-ul GoogleSignIn și extrăgând tokenurile necesare autentificării.

**3. Profile**

**• ProfileView** - interfața utilizatorului pentru profilul curent. Acesta afișează detalii despre utilizator, cum ar fi identificatorul unic, e-mailul și numărul de pași înregistrați pentru ziua curentă. Utilizatorii pot actualiza imaginea de profil prin selectarea unei noi imagini din galerie, datorită integrării cu PhotoPicker.

**• ProfileViewModel** -acționează ca legătura dintre ProfileView și datele de utilizator stocate în Firebase. Acesta gestionează solicitările asincrone pentru încărcarea datelor curente ale utilizatorului și interacțiunea cu serviciile de backend, cum ar fi Firebase Storage pentru încărcarea imaginilor și HealthKit pentru accesul la numărul de pași.

**4. Settings**

**• SettingsView** - componenta vizuală a ecranului de setări, oferind utilizatorilor posibilitatea de a gestiona contul lor. Utilizatorii pot:

• Deconecta: Printr-un simplu tap, utilizatorii pot ieși din contul lor.

• Șterge contul: O funcție de autodistrugere a contului, cu rol de protecție a datelor personale.

• Gestiona parola și email-ul: Opțiuni pentru resetarea parolei și actualizarea adresei de email, pentru a menține securitatea contului.

**• SettingsViewModel -** acționează ca un mediator între SettingsView și serviciile de backend, manipulând logica de autentificare și gestionarea contului. Acesta expune:

**• AuthProviders**: O listă de metode de autentificare disponibile pentru utilizator.

**• Actions**: Metode asincrone care comunică cu Firebase pentru a executa acțiunile de autentificare, cum ar fi sign out, delete account, reset password și update credentials.

• Implementarea @**MainActor** și proprietățile @**Published** asigură că toate actualizările de stare sunt efectuate pe firul principal, menținând interfața utilizatorului sincronizată și responsivă.

Această pereche View-ViewModel demonstrează aplicarea principiilor **MVVM** într-o manieră modernă și eficientă, facilitând o arhitectură curată și scalabilă pentru aplicație.

**3. Firestore**

**• UserManager.swift** - pivotul gestionării datelor utilizatorilor în cadrul bazei de date Firestore. Acesta servește ca un punct centralizat pentru crearea, actualizarea și interogarea informațiilor legate de conturile utilizatorilor.

• Funcționalități cheie:

• Singleton Pattern: UserManager utilizează un model singleton pentru a asigura o singură instanță de acces la baza de date, evitând astfel redundanța și potențialele conflicte de date.

• Asynchronous Operations: Toate operațiunile de bază de date sunt asincrone, profitând de capabilitățile moderne Swift pentru a oferi un răspuns rapid și eficient al UI-ului.

• Secure Data Handling: Prin utilizarea FirebaseFirestoreSwift, UserManager gestionează datele în mod sigur, cu o structură clară a modelului de date și conversii tipizate între Swift și Firestore.

**4. Utilities** -găzduiește o colecție de instrumente care oferă funcționalități de suport și servicii de bază pentru întreaga aplicație. Acest folder este esențial pentru menținerea unei baze de cod curată și pentru a evita duplicarea logicii în mai multe locuri.

**• PhotoPicker.swift**: Un adaptor între UIKit și SwiftUI care permite utilizatorilor să aleagă și să editeze imagini.

**• Utilities.swift**: O colecție de funcții utilitare, inclusiv una pentru a găsi controlerul de vârf din ierarhia de view controllers, facilitând gestionarea corectă a navigației.

**• HealthKitManager.swift**: Un manager care încapsulează interacțiunile cu HealthKit, permițând aplicației să solicite autorizație și să extragă datele de sănătate, cum ar fi numărul de pași.

**5. ViewManager** - esențial pentru gestionarea și coordonarea diferitelor interfețe ale utilizatorilor (UIs) și a tranzițiilor dintre acestea. Acest folder servește ca o centrală pentru componente cheie care controlează fluxul de navigație și modul în care sunt prezentate diferitele ecrane în aplicație.

**• RootView.swift**

• Scop: RootView funcționează ca un comutator central în aplicație, determinând ce interfață utilizator (UI) să prezinte în funcție de starea de autentificare a utilizatorului.

• Funcționalități cheie:

• Gestionarea autentificării: Verifică dacă utilizatorul este autentificat și, în funcție de acest lucru, decide ce view să afișeze - fie ProfileView pentru utilizatorii autentificați, fie SignInEmailView pentru cei neautentificați.

• Navigație: Include un NavigationStack pentru a permite navigația între diferite ecrane ale aplicației.

• Animație de tranziție: Utilizează o animație de tip fullScreenCover pentru a afișa ecranul de autentificare într-un mod vizual atractiv.

**• ContainerView.swift**

• Scop: ContainerView acționează ca un strat intermediar care gestionează tranziția dintre ecranul de splash și conținutul principal al aplicației.

• Funcționalități cheie:

• Comutare între view-uri: Alternează între afișarea SplashScreenView și RootView în funcție de un flag de stare.

• Logica de inițiere: Inițializează aplicația cu un ecran de splash, oferind timp pentru încărcarea resurselor sau a datelor inițiale.

• Simplu și eficient: Implementează o logică simplă, dar esențială pentru o bună experiență a utilizatorului la deschiderea aplicației.

**• SplashScreenView.swift**

• Scop: SplashScreenView este destinat afișării unui ecran de încărcare inițial (splash screen) la deschiderea aplicației.

• Funcționalități cheie:

• Animații vizuale: Include animații pentru elementele vizuale, cum ar fi logo-ul, îmbunătățind astfel prima impresie a utilizatorilor.

• Controlul opacității și scalării: Utilizează variabile de stare pentru a controla opacitatea și dimensiunea elementelor UI, creând un efect dinamic.

• Temporizare și tranziție: Gestionează temporizarea pentru cât timp ecranul de splash rămâne vizibil și cum face tranziția către conținutul principal.

***Demo - Splash Screen***

**6. Configurations** - dedicat pentru stocarea și gestionarea setărilor esențiale necesare funcționării aplicației. Acest folder conține trei fișiere:

**• KaizenX.entitlements**: Acest fișier de privilegii definește caracteristicile speciale pe care le solicită aplicația, inclusiv accesul la HealthKit. Prin acest fișier se configurează capabilitățile necesare pentru a accesa și utiliza datele de sănătate ale utilizatorului, pentru obținerea consimțământului acestuia. Astfel, aplicația poate interacționa cu HealthKit pentru a colecta și utiliza informații precum numărul de pași făcuți de utilizator pe zi.

**• GoogleService-Info.plist**: Este un fișier configurat automat de Firebase care conține chei de autentificare și alte date necesare pentru utilizarea serviciilor Google, cum ar fi autentificarea utilizatorilor, stocarea datelor și analiza comportamentului de utilizare. Acest fișier este esențial pentru implementarea și sincronizarea cu serviciile Firebase, permițând o integrare fluentă în aplicație.

**• Info.plist**: Este un fișier de configurare standard în proiectele iOS, care include metadate despre aplicație, precum identificatori unici, setări de securitate, permisiuni de acces la date și alte configurări necesare sistemului de operare.

Demo - Prezentare aplicație