20minCoach

Course: Diseño de Software

Professor: Rodrigo Núñez

Date: September 25, 2025

Students:

- Carlos Ávalos Mendieta — 2024207640
- Daniel Pulido Castagno — 2019196502
- Ian Porras — 2024182540

0) Executive Summary

- **Auth (PoC)** using **Supabase** (OTP email) + **TOTP MFA** flows (setup & challenge) integrated with Expo.
  Rationale: Supabase is Expo-friendly; replaces Auth0 MFA without custom native config, satisfying the case requirements for login/MFA.
- **Roles** from raw_app_meta_data.roles: BasicUser, PremiumUser.
  - **Action A** ("Start 20-min request"): Basic & Premium.
  - **Action B** ("View earnings"): Premium only.
- **UI Security**: RequireAuth, RequireRole, and RoleGate (component/action-level gates). Dashboards use these guards.
- **HTTP Interceptor**: injects bearer token; on **401** → sign-out, clear Redux, redirect **/auth**.
  "**Force 401**" button added to both dashboards for verification.
- **State & Data plumbing**:
  - **Redux Toolkit** slice auth ({ email, roles }).
  - **TanStack Query** provider (retries, cache, refetch on focus).
  - httpJson helper (timeout + safe JSON parse) layered on the interceptor.
- **Verification page**: tools/data-test (inputs in blue for dark mode), uses TanStack Query + httpJson.

1) Technology Choices

- **Framework/Navigation:** React Native + **Expo** + **expo-router** (file-based routes, minimal native config).
- **Auth: Supabase** (@supabase/supabase-js) with **Email OTP** + **TOTP MFA**, secure storage via **expo-secure-store**.
- **State: Redux Toolkit** for app state (auth), **TanStack Query** for server data (cache, retries, refetch on foreground).
- **Networking:** fetch wrapped by **withAuth** (interceptor) + httpJson (timeout + parse).

- **Styling:** RN primitives; blue inputs/text to ensure readability in dark mode.
- **Testing (planned tomorrow):** Jest + React Testing Library.

Why this stack? It minimizes native setup in Expo Go, satisfies the case's auth/MFA/roles/guard/interceptor requirements, and separates **global state** (Redux) from **remote data** (Query).

2) Authentication & Authorization

- **OTP Login (email-only)**: request/verify 6-digit code; session persisted by Supabase; Redux stores { email, roles }.
- **Roles**: set in Supabase **app metadata** (["BasicUser"] or ["PremiumUser"]).
- **MFA (TOTP)**:
    - **Setup** (/auth/mfa-setup): enroll factor → show secret → verify → redirect to proper dashboard by role.
    - **Challenge** (/auth/mfa-challenge): list factors → enter code → verify → redirect.

3) UI Security: Guards & Permissions

- RequireAuth: no session → redirect /auth.
- RequireRole: lacks required capability → redirect fallback (basic).
- RoleGate: show/hide action by capability:
    - **A:** BasicUser or PremiumUser
    - **B:** PremiumUser only

Dashboards use these guards so UI capabilities match the business rules.

4) HTTP Interceptor & httpJson

- **withAuth** adds Authorization: Bearer <token> on every request.
- On **401**: Supabase signOut(), Redux clearAuth, router.replace('/auth').
- **httpJson(url, init)**: 15s timeout, robust JSON parsing, leverages the interceptor.

- **Verification**: "**Force 401**" button in both dashboards triggers a 401 endpoint to confirm the sign-out + redirect flow.

5) State & Data

- **Redux Toolkit**: auth slice (email/roles) consumed by guards and dashboards.
- **TanStack Query**: QueryClient with retry <= 2 (except 401), staleTime = 30s, refetch on foreground via focusManager.

Verification page tools/data-test confirms:

- Provider wiring works.
- Cache/refetch behaviors are correct.
- httpJson + interceptor path is exercised.