JS | AUTH

// js/screens/auth.js

import { app } from "../firebase-config.js";

import {

  getAuth,

  setPersistence,

  browserLocalPersistence,

  onAuthStateChanged,

  createUserWithEmailAndPassword,

  signInWithEmailAndPassword,

  sendPasswordResetEmail,

  GoogleAuthProvider,

  signInWithPopup,

  signOut,

} from "https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";

let auth;

const $ = (sel) => document.querySelector(sel);

function showMsg(msg, type = "info") {

  const box = $("#authMsg");

  if (!box) return;

  box.textContent = msg;

  box.style.color = type === "error" ? "#b00020" : "var(--muted-foreground)";

}

function mapAuthError(err) {

  const code = err?.code || "";

  const msgs = {

    "auth/invalid-email": "Email inválido.",

    "auth/email-already-in-use": "Este email já está registado.",

    "auth/weak-password": "Palavra-passe fraca (mín. 6 caracteres).",

    "auth/invalid-credential": "Credenciais inválidas.",

    "auth/wrong-password": "Palavra-passe incorreta.",

    "auth/user-not-found": "Utilizador não encontrado.",

    "auth/popup-closed-by-user": "Janela fechada antes de concluir.",

  };

  return msgs[code] || "Ocorreu um erro. Tenta novamente.";

}

async function doSignIn(email, password) {

  await signInWithEmailAndPassword(auth, email, password);

}

async function doRegister(email, password) {

  await createUserWithEmailAndPassword(auth, email, password);

}

async function doGoogle() {

  const provider = new GoogleAuthProvider();

  await signInWithPopup(auth, provider);

}

async function doReset(email) {

  await sendPasswordResetEmail(auth, email);

}

export function protectPage() {

  const auth = getAuth(app);

  return new Promise((resolve) => {

    const unsub = onAuthStateChanged(auth, (user) => {

      unsub();

      if (!user) {

        document.body.classList.add("auth-screen");

        window.navigateTo?.("auth");

        resolve(false);

      } else {

        document.body.classList.remove("auth-screen");

        resolve(true);

      }

    });

  });

}

export function initScreen() {

  // esconder footer e afins no ecrã de auth (aproveita o CSS existente)

  document.body.classList.add("auth-screen");

  auth = getAuth(app);

  setPersistence(auth, browserLocalPersistence).catch(() => {});

  // Se já estiver autenticado, segue para a dashboard

  onAuthStateChanged(auth, (user) => {

    if (user) {

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    }

  });

  const emailEl = $("#authEmail");

  const passEl = $("#authPassword");

  $("#btnSignIn")?.addEventListener("click", async () => {

    showMsg("");

    try {

      await doSignIn(emailEl.value.trim(), passEl.value);

      showMsg("Sessão iniciada.");

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

  $("#btnRegister")?.addEventListener("click", async () => {

    showMsg("");

    try {

      await doRegister(emailEl.value.trim(), passEl.value);

      showMsg("Conta criada. A iniciar sessão…");

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

  $("#btnGoogle")?.addEventListener("click", async () => {

    showMsg("");

    try {

      await doGoogle();

      document.body.classList.remove("auth-screen");

      window.navigateTo("dashboard");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

  $("#btnReset")?.addEventListener("click", async () => {

    const email = emailEl.value.trim();

    if (!email) return showMsg("Introduz o teu email para recuperar.", "error");

    try {

      await doReset(email);

      showMsg("Email de recuperação enviado (verifica o spam).");

    } catch (e) {

      showMsg(mapAuthError(e), "error");

      console.error(e);

    }

  });

  // Opcional: expor logout global (podes chamar em Settings)

  window.appSignOut = async () => {

    try {

      await signOut(auth);

    } finally {

      document.body.classList.add("auth-screen");

      window.navigateTo("auth");

    }

  };

}

export async function doLogout() {

  const auth = getAuth(app); // garante que obtemos a instância mesmo fora do init

  try {

    await signOut(auth);

  } finally {

    document.body.classList.add("auth-screen");

    window.navigateTo("auth");

  }

}