Api\_cliente.dart “import 'package:dio/dio.dart';

import 'package:flutter\_secure\_storage/flutter\_secure\_storage.dart';

class ApiClient {

  ApiClient.\_();

  static final ApiClient I = ApiClient.\_();

  /// Cambia si tu API usa otro prefijo. Si ya pasas rutas con '/api/...'

  /// no pasa nada; el cliente detecta y NO lo duplica.

  static const String \_apiPrefix = '/api';

  final Dio dio = Dio(

    BaseOptions(

      baseUrl: const String.fromEnvironment(

        'API\_BASE\_URL',

        // Emulador Android -> 10.0.2.2; dispositivo físico usa tu IP o adb reverse

        defaultValue: 'http://10.0.2.2:8000',

      ),

      headers: {

        'Accept': 'application/json',

        'Content-Type': 'application/json',

      },

      connectTimeout: const Duration(seconds: 15),

      receiveTimeout: const Duration(seconds: 20),

    ),

  );

  final \_storage = const FlutterSecureStorage();

  /// Llama esto una sola vez al iniciar la app (ya lo haces en `app.dart`)

  Future<void> init() async {

    dio.interceptors.add(

      InterceptorsWrapper(

        onRequest: (opt, handler) async {

          // Bearer

          final access = await \_storage.read(key: 'access\_token');

          if (access != null && access.isNotEmpty) {

            opt.headers['Authorization'] = 'Bearer $access';

          }

          // Normaliza la ruta para asegurar el prefijo /api

          opt.path = \_withApiPrefix(opt.path);

          handler.next(opt);

        },

        onError: (e, handler) async {

          // Si expira el access token, intenta refrescar y reintenta UNA vez

          if (e.response?.statusCode == 401) {

            final ok = await \_refresh();

            if (ok) {

              final req = e.requestOptions;

              final access = await \_storage.read(key: 'access\_token');

              if (access != null) {

                req.headers['Authorization'] = 'Bearer $access';

              }

              // OJO: hay que clonar la request y volver a normalizar path

              final RequestOptions retry = RequestOptions(

                path: \_withApiPrefix(req.path),

                method: req.method,

                data: req.data,

                queryParameters: req.queryParameters,

                headers: req.headers,

                baseUrl: req.baseUrl,

                sendTimeout: req.sendTimeout,

                receiveTimeout: req.receiveTimeout,

                extra: req.extra,

                contentType: req.contentType,

                responseType: req.responseType,

                followRedirects: req.followRedirects,

                listFormat: req.listFormat,

                maxRedirects: req.maxRedirects,

                receiveDataWhenStatusError: req.receiveDataWhenStatusError,

                requestEncoder: req.requestEncoder,

                responseDecoder: req.responseDecoder,

                validateStatus: req.validateStatus,

              );

              final resp = await dio.fetch(retry);

              return handler.resolve(resp);

            }

          }

          handler.next(e);

        },

      ),

    );

  }

  /// Guarda tokens (útil en login)

  Future<void> saveTokens(

      {required String access, required String refresh}) async {

    await \_storage.write(key: 'access\_token', value: access);

    await \_storage.write(key: 'refresh\_token', value: refresh);

  }

  /// Borra tokens (útil en logout)

  Future<void> clearTokens() async {

    await \_storage.delete(key: 'access\_token');

    await \_storage.delete(key: 'refresh\_token');

  }

  // ------------------ WRAPPERS DE ALTO NIVEL ------------------

  Future<Response<dynamic>> get(

    String path, {

    Map<String, dynamic>? query,

  }) {

    return dio.get(\_withApiPrefix(path), queryParameters: query);

  }

  Future<Response<dynamic>> post(

    String path,

    dynamic data, {

    Map<String, dynamic>? query,

  }) {

    return dio.post(\_withApiPrefix(path), data: data, queryParameters: query);

  }

  Future<Response<dynamic>> patch(

    String path,

    dynamic data, {

    Map<String, dynamic>? query,

  }) {

    return dio.patch(\_withApiPrefix(path), data: data, queryParameters: query);

  }

  Future<Response<dynamic>> delete(

    String path, {

    Map<String, dynamic>? query,

  }) {

    return dio.delete(\_withApiPrefix(path), queryParameters: query);

  }

  // ------------------ REFRESH TOKEN ------------------

  Future<bool> \_refresh() async {

    final refresh = await \_storage.read(key: 'refresh\_token');

    if (refresh == null || refresh.isEmpty) return false;

    try {

      // Ojo: aquí usamos path con prefijo /api

      final res = await dio.post(

        \_withApiPrefix('/auth/refresh/'),

        data: {'refresh': refresh},

      );

      final newAccess = res.data['access'] as String?;

      if (newAccess == null || newAccess.isEmpty) return false;

      await \_storage.write(key: 'access\_token', value: newAccess);

      return true;

    } catch (\_) {

      // Si no se puede refrescar, limpiamos y devolvemos false

      await \_storage.deleteAll();

      return false;

    }

  }

  // ------------------ HELPERS ------------------

  /// Asegura que todos los paths lleven `/api` al frente,

  /// pero sin duplicar si ya viene con `/api`.

  String \_withApiPrefix(String path) {

    // normaliza dobles barras, quita espacios

    final p = path.trim();

    if (p.startsWith('http://') || p.startsWith('https://')) {

      // Es URL absoluta; no tocamos

      return p;

    }

    if (p.startsWith('$\_apiPrefix/')) {

      return p; // ya trae /api

    }

    if (p == \_apiPrefix) {

      return p; // exactamente "/api"

    }

    if (p.startsWith('/')) {

      return '$\_apiPrefix$p';

    }

    return '$\_apiPrefix/$p';

  }

}

”

Profile\_repository.dart “import 'api\_client.dart';

class ProfileRepository {

  final \_api = ApiClient.I;

  Future<Map<String, dynamic>> me() async {

    final resp = await \_api.get('/me/');

    return (resp.data as List).first;

  }

  Future<Map<String, dynamic>> updateCliente({

    required int clienteId,

    required Map<String, dynamic> data,

  }) async {

    final resp = await \_api.patch('/clientes/$clienteId/', data);

    return resp.data as Map<String, dynamic>;

  }

}

”

Profile\_Controller “import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import '../../data/profile\_repository.dart';

import '../auth/auth.dart';

class ProfileState {

  final bool loading;

  final Map<String, dynamic>? me; // mismo shape de /me/

  final String? error;

  const ProfileState({this.loading = false, this.me, this.error});

  ProfileState copyWith(

          {bool? loading, Map<String, dynamic>? me, String? error}) =>

      ProfileState(

          loading: loading ?? this.loading, me: me ?? this.me, error: error);

}

class ProfileController extends StateNotifier<ProfileState> {

  ProfileController(this.\_repo, this.\_ref) : super(const ProfileState());

  final ProfileRepository \_repo;

  final Ref \_ref;

  Future<void> load() async {

    state = state.copyWith(loading: true, error: null);

    try {

      final me = await \_repo.me();

      state = state.copyWith(loading: false, me: me);

    } catch (e) {

      state = state.copyWith(loading: false, error: e.toString());

    }

  }

  Future<void> saveCliente(Map<String, dynamic> data) async {

    final clienteId = (\_ref.read(authProvider).user?['cliente\_id']) as int?;

    if (clienteId == null) throw Exception('No se encontró cliente\_id.');

    state = state.copyWith(loading: true, error: null);

    try {

      await \_repo.updateCliente(clienteId: clienteId, data: data);

      await load(); // refresca /me/

    } catch (e) {

      state = state.copyWith(loading: false, error: e.toString());

      rethrow;

    }

  }

}

final profileRepoProvider = Provider((\_) => ProfileRepository());

final profileProvider =

    StateNotifierProvider<ProfileController, ProfileState>((ref) {

  return ProfileController(ref.read(profileRepoProvider), ref);

});

”

Profile\_Screen.dart “import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import '../auth/auth.dart';

import 'profile\_controller.dart';

class ProfileScreen extends ConsumerStatefulWidget {

  const ProfileScreen({super.key});

  @override

  ConsumerState<ProfileScreen> createState() => \_ProfileScreenState();

}

class \_ProfileScreenState extends ConsumerState<ProfileScreen> {

  bool editing = false;

  final telCtrl = TextEditingController();

  final dirCtrl = TextEditingController();

  final ocupCtrl = TextEditingController();

  final ingresosCtrl = TextEditingController();

  final fechaNacCtrl = TextEditingController();

  @override

  void initState() {

    super.initState();

    Future.microtask(() => ref.read(profileProvider.notifier).load());

  }

  @override

  void dispose() {

    telCtrl.dispose();

    dirCtrl.dispose();

    ocupCtrl.dispose();

    ingresosCtrl.dispose();

    fechaNacCtrl.dispose();

    super.dispose();

  }

  void \_fill(Map<String, dynamic>? cliente) {

    telCtrl.text = cliente?['telefono'] ?? '';

    dirCtrl.text = cliente?['direccion'] ?? '';

    ocupCtrl.text = cliente?['ocupacion'] ?? '';

    ingresosCtrl.text = cliente?['ingresos\_mensuales']?.toString() ?? '';

    fechaNacCtrl.text = cliente?['fecha\_nacimiento'] ?? '';

  }

  Future<void> \_save() async {

    final body = {

      'telefono': telCtrl.text.trim(),

      'direccion': dirCtrl.text.trim(),

      'ocupacion': ocupCtrl.text.trim(),

      'ingresos\_mensuales': ingresosCtrl.text.trim().isEmpty

          ? null

          : num.tryParse(ingresosCtrl.text.trim()),

      'fecha\_nacimiento': fechaNacCtrl.text.trim().isEmpty

          ? null

          : fechaNacCtrl.text.trim(), // YYYY-MM-DD

    };

    await ref.read(profileProvider.notifier).saveCliente(body);

    if (!mounted) return;

    setState(() => editing = false);

  }

  Future<void> \_refresh() => ref.read(profileProvider.notifier).load();

  @override

  Widget build(BuildContext context) {

    final auth = ref.watch(authProvider);

    final st = ref.watch(profileProvider);

    final me = st.me;

    final user = me ?? auth.user; // fallback

    final cliente = me?['cliente\_info'];

    final ui = me?['username'] ?? user?['username'] ?? '';

    final name = (me?['first\_name'] ?? user?['first\_name'] ?? '')

                 .toString()

                 .trim();

    final last = (me?['last\_name'] ?? user?['last\_name'] ?? '')

                 .toString()

                 .trim();

    final fullName = ([name, last]..removeWhere((e)=>e.isEmpty)).join(' ');

    final email = me?['email'] ?? user?['email'] ?? '';

    if (!editing) \_fill(cliente);

    return Scaffold(

      appBar: AppBar(

        title: const Text('Mi perfil'),

        actions: [

          IconButton(

            icon: Icon(editing ? Icons.close : Icons.edit),

            onPressed: () => setState(() => editing = !editing),

            tooltip: editing ? 'Cancelar' : 'Editar',

          ),

        ],

      ),

      body: RefreshIndicator(

        onRefresh: \_refresh,

        child: ListView(

          padding: const EdgeInsets.all(16),

          children: [

            // Header

            Container(

              padding: const EdgeInsets.all(16),

              decoration: \_card,

              child: Row(

                children: [

                  CircleAvatar(

                    radius: 28,

                    child: Text(

                      (fullName.isNotEmpty ? fullName[0] : ui.isNotEmpty ? ui[0] : '?').toUpperCase(),

                      style: const TextStyle(fontSize: 24, fontWeight: FontWeight.bold),

                    ),

                  ),

                  const SizedBox(width: 12),

                  Expanded(

                    child: Column(

                      crossAxisAlignment: CrossAxisAlignment.start,

                      children: [

                        Text(fullName.isEmpty ? ui : fullName,

                            style: const TextStyle(fontSize: 18, fontWeight: FontWeight.w700)),

                        const SizedBox(height: 2),

                        Text(email, style: const TextStyle(color: Colors.grey)),

                      ],

                    ),

                  ),

                ],

              ),

            ),

            const SizedBox(height: 12),

            // Resumen documento

            if (cliente != null) Container(

              padding: const EdgeInsets.all(16),

              decoration: \_card,

              child: Wrap(

                spacing: 16,

                runSpacing: 8,

                children: [

                  \_pill('Documento', '${cliente['tipo\_documento'] ?? ''} ${cliente['numero\_documento'] ?? ''}'.trim()),

                  \_pill('Puntaje', '${cliente['puntuacion\_crediticia'] ?? 0}'),

                  \_pill('Preferencial', (cliente['es\_cliente\_preferencial'] ?? false) ? 'Sí' : 'No'),

                ],

              ),

            ),

            const SizedBox(height: 12),

            // Form cliente

            Container(

              decoration: \_card,

              padding: const EdgeInsets.all(16),

              child: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  const Text('Datos del cliente',

                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.w700)),

                  const SizedBox(height: 10),

                  \_field('Teléfono', telCtrl, enabled: editing, keyboard: TextInputType.phone),

                  \_field('Dirección', dirCtrl, enabled: editing),

                  \_field('Ocupación', ocupCtrl, enabled: editing),

                  \_field('Ingresos mensuales', ingresosCtrl, enabled: editing, keyboard: TextInputType.number),

                  \_field('Fecha nac. (YYYY-MM-DD)', fechaNacCtrl, enabled: editing, keyboard: TextInputType.datetime),

                  if (st.error != null) Padding(

                    padding: const EdgeInsets.only(top: 8),

                    child: Text(st.error!, style: const TextStyle(color: Colors.red)),

                  ),

                  const SizedBox(height: 8),

                  if (editing)

                    FilledButton.icon(

                      onPressed: st.loading ? null : \_save,

                      icon: const Icon(Icons.save),

                      label: const Text('Guardar cambios'),

                    )

                ],

              ),

            ),

          ],

        ),

      ),

    );

  }

  BoxDecoration get \_card => BoxDecoration(

    color: Colors.white,

    borderRadius: BorderRadius.circular(14),

    boxShadow: const [BoxShadow(color: Colors.black12, blurRadius: 8, offset: Offset(0,2))]

  );

  Widget \_pill(String k, String v) {

    return Container(

      padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 8),

      decoration: BoxDecoration(

        color: const Color(0xFFF2F5F8),

        borderRadius: BorderRadius.circular(999),

      ),

      child: Row(mainAxisSize: MainAxisSize.min, children: [

        Text('$k: ', style: const TextStyle(color: Colors.black54)),

        Text(v, style: const TextStyle(fontWeight: FontWeight.w600)),

      ]),

    );

  }

  Widget \_field(String label, TextEditingController c, {bool enabled=true, TextInputType? keyboard}) {

    return Padding(

      padding: const EdgeInsets.only(bottom: 10),

      child: TextField(

        controller: c,

        enabled: enabled,

        keyboardType: keyboard,

        decoration: InputDecoration(

          labelText: label,

          border: const OutlineInputBorder(),

          isDense: true,

        ),

      ),

    );

  }

}

”

Router.dart “// lib/core/router.dart

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart' as rp;

import 'package:go\_router/go\_router.dart';

import '../features/auth/auth.dart'; // exporta authProvider y AuthState

import '../features/auth/onboarding\_screen.dart';

import '../features/auth/login\_screen.dart';

import '../features/auth/register\_screen.dart';

import '../features/home/home\_screen.dart';

import '../features/solicitudes/solicitud\_new\_screen.dart';

import '../features/solicitudes/solicitudes\_list\_screen.dart';

import '../features/solicitudes/solicitud\_detail\_screen.dart';

import '../features/profile/profile\_screen.dart';

import '../features/notifications/notifications\_screen.dart';

class AppRouter {

  /// Acepta WidgetRef (desde un ConsumerWidget/ConsumerState)

  static GoRouter create(rp.WidgetRef ref) {

    return GoRouter(

      initialLocation: '/onboarding',

      // Se vuelve a evaluar el redirect cuando cambia el authProvider

      refreshListenable: GoRouterRefresh(ref),

      redirect: (context, state) {

        final auth = ref.read(authProvider);

        final path = state.fullPath ?? state.uri.toString();

        final inAuth = path == '/login' || path == '/register';

        // No autenticado y fuera de login/register/onboarding -> a login

        if (!auth.authenticated &&

            path != '/login' &&

            path != '/register' &&

            path != '/onboarding') {

          return '/login';

        }

        // Autenticado y dentro de login/register -> a home

        if (auth.authenticated && inAuth) return '/home';

        return null;

      },

      // ⚠️ ¡Sin const! GoRoute no es const porque usa builder (una función).

      routes: [

        GoRoute(

          path: '/onboarding',

          builder: (\_, \_\_) => const OnboardingScreen(),

        ),

        GoRoute(

          path: '/login',

          builder: (\_, \_\_) => const LoginScreen(),

        ),

        GoRoute(

          path: '/register',

          builder: (\_, \_\_) => const RegisterScreen(),

        ),

        GoRoute(

          path: '/home',

          builder: (\_, \_\_) => const HomeScreen(),

        ),

        GoRoute(

          path: '/solicitudes',

          builder: (\_, \_\_) => const SolicitudesListScreen(),

        ),

        GoRoute(

          path: '/solicitudes/new',

          builder: (\_, \_\_) => const SolicitudNewScreen(),

        ),

        GoRoute(

          path: '/solicitudes/:id',

          builder: (ctx, st) =>

              SolicitudDetailScreen(id: st.pathParameters['id']!),

        ),

        GoRoute(path: '/perfil', builder: (\_, \_\_) => const ProfileScreen()),

        GoRoute(

            path: '/notificaciones',

            builder: (\_, \_\_) => const NotificationsScreen()),

      ],

    );

  }

}

/// Notificador que despierta a GoRouter cuando cambia el provider de auth.

/// Usamos listenManual para obtener una suscripción y cerrarla en dispose.

class GoRouterRefresh extends ChangeNotifier {

  GoRouterRefresh(this.ref) {

    \_sub = ref.listenManual<AuthState>(

      authProvider,

      (prev, next) => notifyListeners(),

      fireImmediately: false, // no dispares al crear

    );

  }

  final rp.WidgetRef ref;

  late final rp.ProviderSubscription<AuthState> \_sub;

  @override

  void dispose() {

    \_sub.close();

    super.dispose();

  }

}

”