**نظرة عامة على الطبقات**

* **core**: أساسات عامة للتطبيق (ألوان + Theme + إعدادات مشتركة). لا تعتمد على أي feature.
* **data**: الوصول للبيانات (API client، موديلات، Repositories). ما فيهاش UI.
* **features**: كل ميزة (auth, explore, profile…) في مجلد مستقل يحتوي UI + State الخاص بيها.
* **shared**: Widgets/Utilities قابلة لإعادة الاستخدام بين الميزات.
* **app**: نقطة تشغيل التطبيق (Theme, Providers, Router).

**شرح كل مجلد/ملف**

**lib/main.dart**

* **نقطة البداية** للتطبيق.
* يستدعي PyramidsTouristApp من app/app.dart.
* ما فيهوش منطق—مجرد bootstrapping.

**lib/app/app.dart**

* يبني **MaterialApp** ويطبّق **Theme** من core/theme/app\_theme.dart.
* يجهّز **مزودي الحالة** (Providers) العامة؛ مثل AuthController.
* يحدد **home** (عادةً SplashPage).

لاحقًا: لو استخدمت GoRouter حط خطوة التهيئة هنا أيضًا.

**lib/core/constants/colors.dart**

* تعريف ثوابت الألوان (kGold, kDark) لتوحيد الهوية البصرية.

**lib/core/theme/app\_theme.dart**

* دالة buildAppTheme() بترجع ThemeData موحّد (Dark + Gold).
* أي تعديل على الشكل (CardTheme, InputDecorationTheme…) يكون هنا فقط، فينعكس على كل الشاشات.

**lib/data/api/api\_client.dart**

* **الطبقة الأدنى** للوصول للـ API (HTTP).
* مسئول عن:
  + إرسال الطلبات (http).
  + **إدارة التوكن** (حفظ/قراءة من SharedPreferences).
  + بناء الهيدرز.
  + فك الاستجابة ومعالجة الأخطاء.
* لا يعرف حاجة عن الـ UI أو الـ Models؛ بيرجع خرائط/قوائم JSON.

**lib/data/models/\*.dart**

* **نماذج البيانات** (مثل User, Poi).
* تحول JSON إلى كائنات Dart typed (و العكس لو احتجت).
* تحافظ على أنواع وحقول واضحة وتقلل أخطاء runtime.

**lib/data/repositories/\*.dart**

* **واجهة نظيفة** بين الـ UI والـ API.
* تجمع استدعاءات متعددة وتطبّق منطق بسيط (مثال: بعد login اطلب me).
* ترجع **Models** بدل JSON:
  + AuthRepository.login() → User
  + PoiRepository.list() → List<Poi>
* مكان مناسب لاحقًا لإضافة:
  + **Caching**، **Pagination**، **Retry**، دمج مصادر بيانات متعددة.

**lib/features/*/presentation/*.dart**

* شاشات/Widgets لكل ميزة على حدة:
  + splash\_page.dart: يتحقق من الجلسة ويحول.
  + auth\_page.dart: فورم الدخول/التسجيل.
  + app\_shell.dart: Bottom Navigation + التبديل بين التابات.
  + صفحات explore/plan/itineraries/favorites/profile … إلخ.
* استخدم Widgets صغيرة وقابلة لإعادة الاستخدام من shared/widgets.

**lib/features/auth/state/auth\_controller.dart**

* **إدارة حالة ميزة** (ChangeNotifier بسيط الآن).
* يمسك User? user، ويوفر:
  + tryRestore(), signIn(), signUp(), refreshProfile(), signOut().
* يتواصل مع AuthRepository فقط (مش مع API مباشرة).
* UI يقرأ منه عن طريق Provider.

**lib/shared/widgets/\***

* **Atoms/Building Blocks**: أزرار، Chips، خلفية موحّدة.
* ده يمنع تكرار الكود ويحافظ على ستايل ثابت.

**lib/shared/demo/demo\_data.dart**

* بيانات ديمو بديلة لو الـ API مارجّعش بيانات (أو أثناء التطوير).

**ليه التقسيم ده عملي؟**

* **فصل المسؤوليات**: UI ≠ Data. أي تغيير في API مش هيكسر UI والعكس.
* **قابلية التوسع**: كل Feature في جزيرتها. تضيف/تحذف Feature بدون ما تلمّس الباقي.
* **اختبار أسهل**: تقدر تعمل Mock للـ Repositories وتختبر Controllers بدون UI.
* **إعادة استخدام**: Widgets مشتركة، Theme موحّد، Repositories مشتركة.