Flutter GetX Project Structure

GetX all-in-one toolkit. State management, routing, and DI with minimal boilerplate.

#flutter #dart #mobile #getx #state-management

PNG PDF Copy Prompt

Project Directory

```
myapp/
    pubspec.yaml
    pubspec.lock
    analysis_options.yaml
    .gitignore
  lib/
    main.dart GetMaterialApp
    app/ App-wide config
      bindings/ Initial bindings
        initial_binding.dart
      routes/ GetX routes
        app_pages.dart
        app_routes.dart
      translations/
      theme/
    modules/ Feature modules
      auth/
        bindings/
          auth_binding.dart Inject controll…
        controllers/
          auth_controller.dart GetxController
        views/
          login_view.dart
        widgets/
      home/
        bindings/
        controllers/
        views/
        widgets/
    data/ Data layer
      models/
      providers/ API providers
        api_provider.dart
      repositories/
        auth_repository.dart
    core/
      utils/
        values/ Constants, colo…
  test/
    modules/
      auth/
  android/
  ios/
  assets/
```

Why This Structure?

GetX provides state management, routing, and dependency injection in one package. Minimal boilerplate with `.obs` reactive variables and `Obx` widgets. Fastest development velocity for GetX-familiar teams.

Key Directories

lib/modules/ - Feature modules with MVC pattern

lib/modules/*/controllers/ - GetxController for each module

lib/modules/*/bindings/ - Dependency injection per route

lib/app/routes/ - GetX named routes config

GetX Controller

```dart
// lib/modules/auth/controllers/auth_controller.dart
class AuthController extends GetxController {
  final AuthRepository _repo = Get.find();

  final user = Rxn();
  final isLoading = false.obs;

  Future login(String email, String password) async {
    isLoading.value = true;
    try {
      user.value = await _repo.login(email, password);
      Get.offAllNamed(Routes.HOME);
    } finally {
      isLoading.value = false;
    }
  }
}
```

Getting Started

1. `flutter pub add get`
2. Replace MaterialApp with GetMaterialApp
3. Create module with binding, controller, view
4. Add routes to app_pages.dart
5. Use `Get.to()`, `Get.find()`, `.obs`, `Obx()`

When To Use This

- Want fastest development velocity
- Team familiar with GetX patterns
- Prefer less boilerplate over explicitness
- Need routing + state + DI in one package
- Building medium-sized apps quickly

Trade-offs

Less explicit - Magic can make debugging harder

Tight coupling - Heavy dependency on GetX package

Testing - Requires GetX test utilities

Community opinion - Some consider it not idiomatic Flutter