dependencies:

flutter:

sdk: flutter

riverpod: ^2.0.0

provider: ^6.0.0

dio: ^5.0.0

flutter\_riverpod: ^2.0.0

dev\_dependencies:

flutter\_test:

sdk: flutter

lib/

│

├── core/ # Core utilities

│ ├── error/

│ ├── network/

│

├── features/ # Feature-specific code

│ ├── user/ # User module

│ │ ├── data/ # Data layer (repository, models)

│ │ ├── domain/ # Business logic (use cases)

│ │ └── presentation/ # UI layer (widgets, providers)

│ └── ...

│

└── main.dart

class User {

final String id;

final String name;

final String email;

User({

required this.id,

required this.name,

required this.email,

});

// Convert User to a Map (for API requests)

Map<String, dynamic> toMap() {

return {

'id': id,

'name': name,

'email': email,

};

}

// Convert Map to User

factory User.fromMap(Map<String, dynamic> map) {

return User(

id: map['id'],

name: map['name'],

email: map['email'],

);

}

}

import 'dart:async';

import 'package:dio/dio.dart';

import 'package:your\_app/features/user/data/models/user\_model.dart';

abstract class UserRepository {

Future<User> createUser(User user);

Future<User> getUser(String id);

Future<List<User>> getAllUsers();

Future<User> updateUser(User user);

Future<void> deleteUser(String id);

}

class UserRepositoryImpl implements UserRepository {

final Dio dio;

UserRepositoryImpl({required this.dio});

@override

Future<User> createUser(User user) async {

final response = await dio.post(

'https://your-api.com/users',

data: user.toMap(),

);

return User.fromMap(response.data);

}

@override

Future<User> getUser(String id) async {

final response = await dio.get('https://your-api.com/users/$id');

return User.fromMap(response.data);

}

@override

Future<List<User>> getAllUsers() async {

final response = await dio.get('https://your-api.com/users');

return (response.data as List)

.map((item) => User.fromMap(item))

.toList();

}

@override

Future<User> updateUser(User user) async {

final response = await dio.put(

'https://your-api.com/users/${user.id}',

data: user.toMap(),

);

return User.fromMap(response.data);

}

@override

Future<void> deleteUser(String id) async {

await dio.delete('https://your-api.com/users/$id');

}

}

class CreateUser {

final UserRepository repository;

CreateUser(this.repository);

Future<User> call(User user) {

return repository.createUser(user);

}

}

class GetUser {

final UserRepository repository;

GetUser(this.repository);

Future<User> call(String id) {

return repository.getUser(id);

}

}

class GetAllUsers {

final UserRepository repository;

GetAllUsers(this.repository);

Future<List<User>> call() {

return repository.getAllUsers();

}

}

class UpdateUser {

final UserRepository repository;

UpdateUser(this.repository);

Future<User> call(User user) {

return repository.updateUser(user);

}

}

class DeleteUser {

final UserRepository repository;

DeleteUser(this.repository);

Future<void> call(String id) {

return repository.deleteUser(id);

}

}

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

import 'package:your\_app/features/user/domain/usecases/create\_user.dart';

import 'package:your\_app/features/user/domain/usecases/get\_user.dart';

import 'package:your\_app/features/user/domain/usecases/get\_all\_users.dart';

import 'package:your\_app/features/user/domain/usecases/update\_user.dart';

import 'package:your\_app/features/user/domain/usecases/delete\_user.dart';

import 'package:your\_app/features/user/data/models/user\_model.dart';

final userProvider = StateNotifierProvider<UserNotifier, List<User>>((ref) {

final getAllUsers = ref.read(getAllUsersUseCaseProvider);

return UserNotifier(getAllUsers);

});

final createUserUseCaseProvider = Provider<CreateUser>((ref) {

final repository = ref.read(userRepositoryProvider);

return CreateUser(repository);

});

final getUserUseCaseProvider = Provider<GetUser>((ref) {

final repository = ref.read(userRepositoryProvider);

return GetUser(repository);

});

final updateUserUseCaseProvider = Provider<UpdateUser>((ref) {

final repository = ref.read(userRepositoryProvider);

return UpdateUser(repository);

});

final deleteUserUseCaseProvider = Provider<DeleteUser>((ref) {

final repository = ref.read(userRepositoryProvider);

return DeleteUser(repository);

});

final userRepositoryProvider = Provider<UserRepository>((ref) {

return UserRepositoryImpl(dio: Dio());

});

final getAllUsersUseCaseProvider = Provider<GetAllUsers>((ref) {

final repository = ref.read(userRepositoryProvider);

return GetAllUsers(repository);

});

class UserNotifier extends StateNotifier<List<User>> {

final GetAllUsers getAllUsers;

UserNotifier(this.getAllUsers) : super([]);

Future<void> fetchAllUsers() async {

final users = await getAllUsers.call();

state = users;

}

}

import 'package:flutter/material.dart';

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

import 'package:your\_app/features/user/presentation/providers/user\_provider.dart';

class UserListScreen extends ConsumerWidget {

@override

Widget build(BuildContext context, WidgetRef ref) {

final users = ref.watch(userProvider);

return Scaffold(

appBar: AppBar(title: Text('Users')),

body: users.isEmpty

? Center(child: Text('No users found.'))

: ListView.builder(

itemCount: users.length,

itemBuilder: (context, index) {

final user = users[index];

return ListTile(

title: Text(user.name),

subtitle: Text(user.email),

);

},

),

);

}

}

import 'package:flutter/material.dart';

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

import 'features/user/presentation/widgets/user\_list\_screen.dart';

void main() {

runApp(const ProviderScope(child: MyApp()));

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter User Management',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: UserListScreen(),

);

}

}