Flutter

Training Assignments

CRUD API with Flutter Cubit

Overview

In In this article, I will discuss how to create a CRUD application using Cubit state management. I thought to make a CRUD application with Pinch it was quite easy to make but in fact there are still some of our friends who have asked me to write them.

.API Documentation


```
[
 {
  "name": "Ky Le",
  "email": "kyle@r2s.com.vn",
  "age": "20",
  "id": "1"
 },
 {
  "name": "dang",
  "email": "dang@gmail.com",
  "age": "1234",
  "id": "2"
 }
]
```


Tasks

1. Create a Model Class: The next step is that we need to create a model class with the name **profile_data.dart**.

```dart
class ProfileData {
  final String? id;
  final String? name;
  final String? email;
  final String? age;

  const ProfileData({this.id, this.name, this.email, this.age});

  factory ProfileData.fromJson(Map<String, dynamic> json) {
    return ProfileData(
        id: json['id'],
        name: json['name'],
        email: json['email'],
        age: json['age']);
  }
}
```

2. Create a ProfileRepository Class: The next step is we need to create a class with the named **profile_repository.dart** which this class will later be responsible for all endpoint calls that we will call.

```dart
class ProfileRepository {
  static const String urlRead =
      'https://60db1a79801dcb0017290e61.mockapi.io/profileData';
  static const String urlCreate =
      'https://60db1a79801dcb0017290e61.mockapi.io/profileData';

  static const int statusCode200 = 200;
  static const int statusCode201 = 201;

  // Get all profile
  Future<List<ProfileData>> getAllProfiles() async {
    final uri = Uri.parse(urlRead);
    final response = await http.get(uri);
    if (response.statusCode == statusCode200) {
      return parseData(response.body);
    }

    throw Exception('Failed to load Profile data ${response.statusCode}');
  }

  // Parse profile from JSON
  List<ProfileData> parseData(String body) {
    final parsed = jsonDecode(body).cast<Map<String, dynamic>>();
    return parsed
        .map<ProfileData>((json) => ProfileData.fromJson(json))
        .toList();
  }

  // Create a Profile
  Future<ProfileData> createProfile(ProfileData profileData) async {
    final uri = Uri.parse(urlCreate);
    final body = {
      'name': profileData.name,
      'email': profileData.email,
      'age': profileData.age
    };

    final response = await http.post(uri, body: body);
    if (response.statusCode == statusCode201) {
      return ProfileData.fromJson(jsonDecode(response.body));
    } else {
      throw Exception('Failed to create profile ${response.statusCode}');
    }
  }
}
```

3. Create Class Profile State: The next step is that we need to create a with the named **profile_state.dart**. This class serves as the state of the pinch that we will create later.

```dart
abstract class ProfileState {}

class InitialProfileState extends ProfileState {}

class LoadingProfileState extends ProfileState {}

class FailureProfileState extends ProfileState {
  final String errorMessage;
  FailureProfileState(this.errorMessage);
}

class SuccessLoadAllProfilesState extends ProfileState {
  final List<ProfileData> listProfiles;
  SuccessLoadAllProfilesState(this.listProfiles);
}

class SuccessSubmitProfileState extends ProfileState {
  final ProfileData profileData;
  SuccessSubmitProfileState(this.profileData);
}
```

4. Create a Pinch Profile Class: Then we create with the named **profile_cubit.dart**.

```dart
class ProfileCubit extends Cubit<ProfileState> {
  final ProfileRepository _repository;

  ProfileCubit(this._repository) : super(InitialProfileState());

  Future<void> getAllProfiles() async {
    emit(LoadingProfileState());
    try {
      var result = await _repository.getAllProfiles();
      emit(SuccessLoadAllProfilesState(result));
    } catch (e) {
      emit(FailureProfileState(e.toString()));
    }
  }

  Future<void> createProfile(ProfileData profileData) async {
    emit(LoadingProfileState());
    try {
      var result = await _repository.createProfile(profileData);
      emit(SuccessSubmitProfileState(result));
    } catch (e) {
      emit(FailureProfileState(e.toString()));
    }
  }
}
```

5. Create UI List Profile: The next step is that we will create a UI that serves to display a list of profiles. We create with the named **profile_page.dart**

```dart
class ProfilePage extends StatelessWidget {
  const ProfilePage({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: _HomePage(),
    );  // MaterialApp
  }
}

class _HomePage extends StatefulWidget {
  const _HomePage({super.key});

  @override
  State<_HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<_HomePage> {
  final nameController = TextEditingController();
  final emailController = TextEditingController();
  final ageController = TextEditingController();
  final profileCubit = ProfileCubit(ProfileRepository());

  @override
  void initState() {
    super.initState();
    profileCubit.getAllProfiles();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Profile'),
      ),  // AppBar
      body: BlocProvider.value(
        value: profileCubit,
        child: BlocBuilder<ProfileCubit, ProfileState>(
          builder: (context, state) {
            if (state is InitialProfileState || state is LoadingProfileState) {
              return const Center(child: CircularProgressIndicator());
            } else if (state is SuccessLoadAllProfilesState) {
              final profiles = state.listProfiles;
              return ListView.builder(
                itemCount: profiles.length,
                itemBuilder: (context, index) {
                  final profile = profiles[index];
```

```
            return Padding(
              padding: const EdgeInsets.all(8.0),
              child: Card(
                child: ListTile(
                  leading: Text('Name: ${profile.name}'),
                  title: Text('Email: ${profile.email}'),
                  subtitle: Text('Age: ${profile.age}'),
                  ),  // ListTile
                ),  // Card
              );  // Padding
            },
          );  // ListView.builder
        } else if (state is FailureProfileState) {
          return Center(child: Text(state.errorMessage));
        }
        return Text(state.toString());
      },
    ),  // BlocBuilder
  ),  // BlocProvider.value
  floatingActionButton: FloatingActionButton.extended(
      onPressed: () async {
        await _showDialog(context, null);
        profileCubit.getAllProfiles();
      },
      label: const Text('Add Profile')),  // FloatingActionButton.extended
);  // Scaffold
}

Future<void> _showDialog(
    BuildContext context, ProfileData? profileData) async {
  return showDialog(
    context: context,
    builder: (context) {
      return StatefulBuilder(builder: (context, setState) {
        return Dialog(
          insetPadding: EdgeInsets.zero,
          child: Scaffold(
            appBar: AppBar(
              title: const Text('Create new profile'),
              leading: IconButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
                icon: const Icon(Icons.close),
              ),  // IconButton
              actions: [
                TextButton(
                    onPressed: () {
                      final profile = ProfileData(
                          name: nameController.text,
                          email: emailController.text,
                          age: ageController.text);  // ProfileData
```

```
                    // Create new profile
                    profileCubit.createProfile(profile);
                  },
                  child: const Text('Save',
                      style: TextStyle(color: Colors.white))),  // Text, Tex
            ],
          ),  // AppBar
          body: BlocProvider.value(
            value: profileCubit,
            child: BlocListener<ProfileCubit, ProfileState>(
              listener: (_, state) {
                if (state is SuccessSubmitProfileState) {
                  ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                      content: Text('Profile adds successfully')));  // Snac
                  setState(() {
                    nameController.clear();
                    emailController.clear();
                    ageController.clear();
                  });
                } else if (state is FailureProfileState) {
                  ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text(state.errorMessage)));
                }
              },
              child: Stack(
                children: [_buildWidgetForm(), _buildWidgetLoading()],
              ),  // Stack
            ),  // BlocListener
          ),  // BlocProvider.value
        ),  // Scaffold
      );  // Dialog
    });  // StatefulBuilder
  },
);
}
```

```dart
  @override
  void dispose() {
    super.dispose();
    nameController.dispose();
    emailController.dispose();
    ageController.dispose();
  }

  Widget _buildWidgetForm() {
    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextFormField(
            decoration: const InputDecoration(
                hintText: 'Enter your name',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.people)),  // InputDecoration
            controller: nameController,
          ),  // TextFormField
        ),  // Padding
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextFormField(
            decoration: const InputDecoration(
                hintText: 'Enter your email',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.email)),  // InputDecoration
            controller: emailController,
          ),  // TextFormField
        ),  // Padding
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: TextFormField(
            controller: ageController,
            decoration: const InputDecoration(
                hintText: 'Enter your age',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.calendar_month)),  // InputDecoration
            keyboardType: TextInputType.number,
          ),  // TextFormField
        ),  // Padding
      ],
    );  // Column
  }
```

```
  Widget _buildWidgetLoading() {
    return BlocBuilder<ProfileCubit, ProfileState>(builder: (_, state) {
      if (state is LoadingProfileState) {
        return const Center(
          child: CircularProgressIndicator(),
        );  // Center
      } else {
        return Container();
      }
    });  // BlocBuilder
  }
}
```

6. Run the app

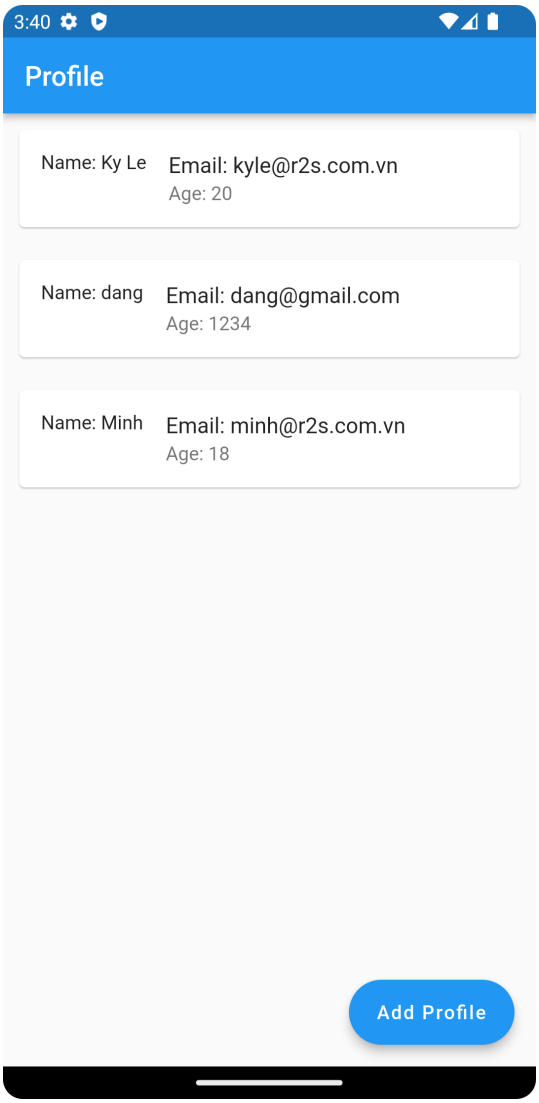


Figure 1: Run

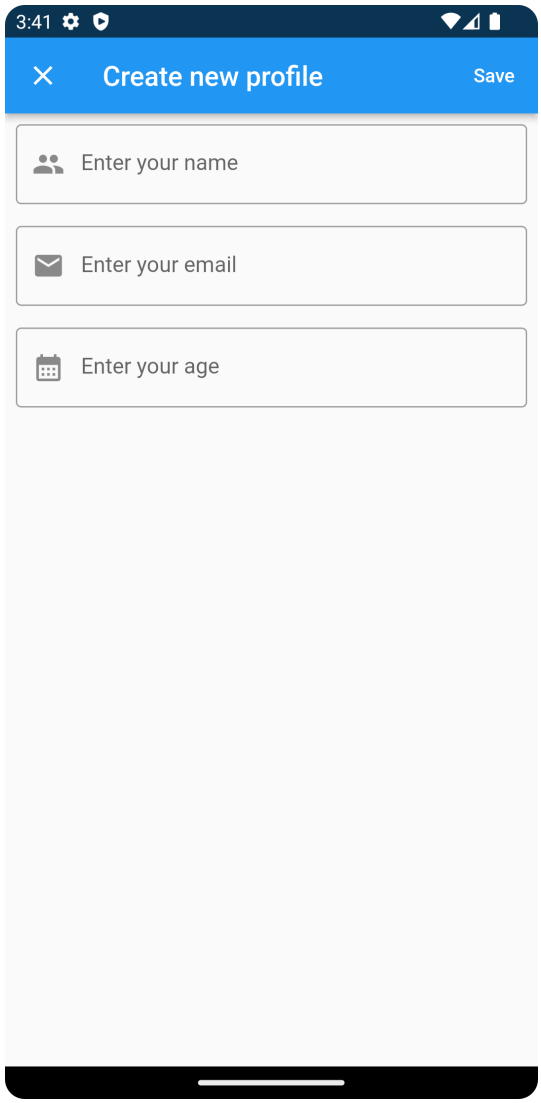
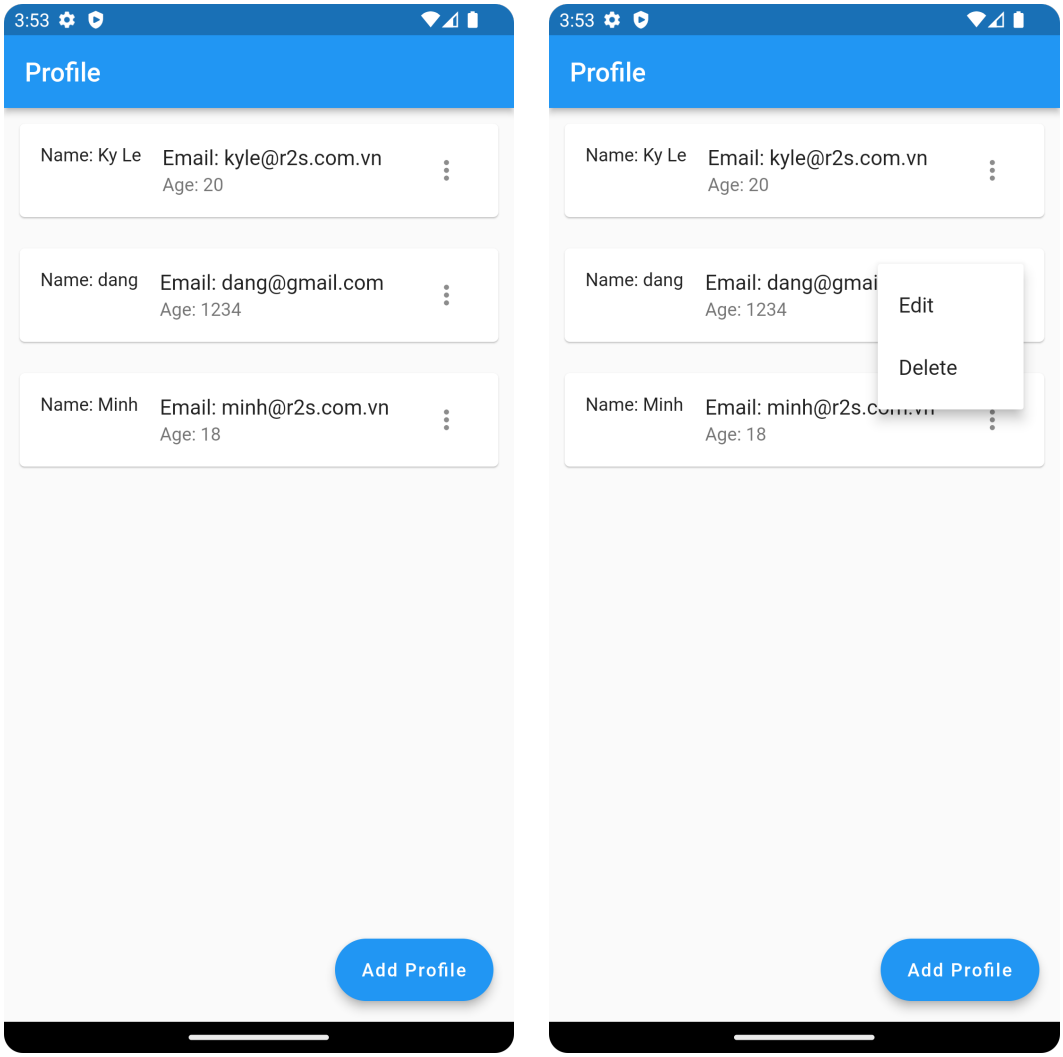


Figure 2: Add profile

Extra tasks

Functional requirements:

1. Create Profile Edit Feature
2. Create a Delete Profile Feature



--THE END—