Universitas	: Universitas Logistik dan Bisnis Internasional (ULBI)
Program Studi	: D4 Teknik Informatika
Semester	: 5 (Lima)
Mata Kuliah	: Pemrograman IV (Pemrograman Mobile)
Topik	: Praktikum – Rest API Lanjutan (Real Case – Contact API)

DIO (Flutter Package)

Dio adalah paket HTTP client yang populer dan powerful yang digunakan untuk melakukan permintaan HTTP dari aplikasi Flutter ke server RESTful.

Dio menyediakan berbagai fitur termasuk pengiriman permintaan HTTP, menangani respons dari server, mengatur header, manajemen timeout, interceptors untuk penanganan otentikasi atau logging, dll.

Dio dirancang untuk sederhana namun kuat, memudahkan pengguna untuk melakukan berbagai operasi HTTP dengan mudah dan efisien.

IMPLEMENTASI (PRAKTIKUM)

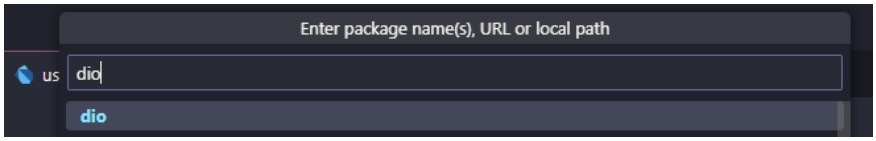
A. Buat project baru dengan nama dio_contact

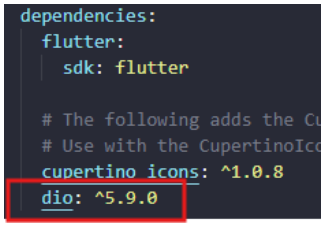
B. Download file yang ada di folder lib DISINI . Buatlah struktur folder di dalam lib seperti berikut (sama seperti pada github di link tersebut)

- **model**
  - contact_model.dart
- **services**
  - api_services.dart
- **view**
  - **screen**
    - home_page.dart
- main.dart

C. Tambahkan depedencies dio

Kalian bisa menambahkan package dengan ketik Ctrl + Shift + P → Dart: Add Depedency → Cari dio dan pilih dengan klik kiri → Kemudian dio akan ada pada file pubspec.yaml kalian.





D. API yang Digunakan

Untuk praktikum kali ini kita akan menggunakan Real API (Contact) yang bisa kalian liat DISINI. Di praktikum ini kita akan membuat Get All, Get Single, Post, Put, dan Delete. Untuk base url API yang digunakan adalah https://contactsapi-production.up.railway.app/.

Disclaimer : Proses akses atau permintaan (request) ke API dapat membutuhkan waktu beberapa saat. Hal ini disebabkan oleh keterbatasan server. Mohon bersabar saat melakukan pengujian.

E. Get Data

1. Buka file api_services.dart

Buka file api_services.dart.

Pada file ini, kita akan membuat instance Dio yang digunakan sebagai HTTP client untuk berkomunikasi dengan server API.

Langkah-langkah yang dilakukan:

- Menambahkan import Dio

  Digunakan agar aplikasi dapat melakukan permintaan HTTP seperti GET, POST, PUT, dan DELETE.

- Membuat class ApiServices

  Class ini berfungsi sebagai service layer yang mengatur seluruh komunikasi dengan API.

- Membuat instance Dio dengan BaseOptions

  BaseOptions digunakan untuk menyimpan konfigurasi dasar yang akan dipakai oleh semua request, sehingga tidak perlu ditulis berulang kali.

Kode berikut digunakan untuk menginisialisasi Dio:

```dart
import 'package:dio/dio.dart';
import 'package:dio_contact/model/contact_model.dart';

Windsurf: Refactor | Explain
class ApiServices {
  final Dio dio = Dio(
    BaseOptions(
      baseUrl: 'https://contactsapi-production.up.railway.app',
      connectTimeout: const Duration(seconds: 15),
      receiveTimeout: const Duration(seconds: 15),
      headers: {'Content-Type': 'application/json'},
    ), // BaseOptions
  ); // Dio
```

Masih pada file api_services.dart tambahkan kode berikut pada method getAllContact

```
Future<List<ContactsModel>?> getAllContact() async {
    try {
      final response = await dio.get('/contacts');

      if (response.statusCode == 200) {
        final list = (response.data['data'] as List)
            .map((e) => ContactsModel.fromJson(e))
            .toList();
        return list;
      }
      return null;
    } on DioException catch (e) {
      debugPrint('Dio error: ${e.response?.statusCode} -
${e.message}');
      return null;
    }
  }
```

Jangan lupa import berikut

```
import 'package:flutter/material.dart';
```

Method getAllContact() digunakan untuk mengambil seluruh data kontak dari REST API dengan melakukan permintaan HTTP GET ke endpoint /contacts menggunakan package Dio. Method ini bersifat asynchronous (async) dan mengembalikan nilai berupa Future<List<ContactsModel>?>, yang berarti hasilnya berupa daftar data kontak atau null jika terjadi kegagalan.

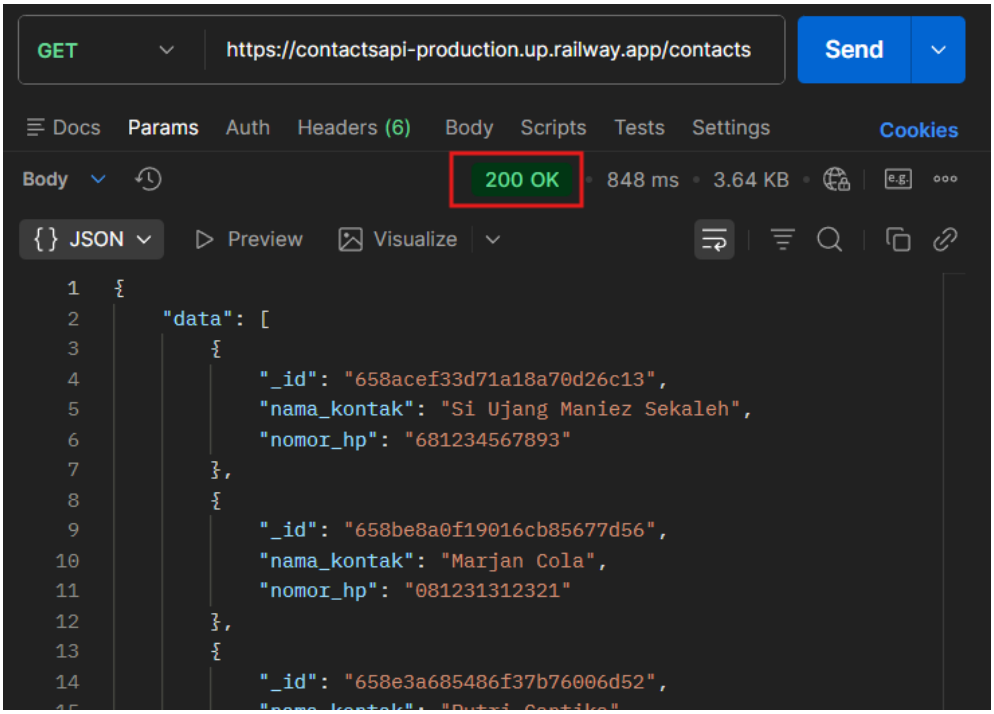
Di dalam method ini, permintaan HTTP dikirim menggunakan dio.get('/contacts'). Karena sebelumnya telah didefinisikan baseUrl pada konfigurasi Dio, maka endpoint yang dipanggil akan digabungkan secara otomatis dengan URL dasar API. Proses ini menggunakan mekanisme async–await sehingga aplikasi akan menunggu hingga server memberikan respons tanpa menghambat proses lain di aplikasi.

Setelah respons diterima, method akan memeriksa status code dari server. Jika status code bernilai 200 (OK), maka data respons yang berada pada properti data akan diambil dan dikonversi dari bentuk JSON List menjadi List<ContactsModel> menggunakan fungsi map() dan fromJson(). Hasil konversi tersebut kemudian dikembalikan sebagai nilai dari method.

Apabila status code yang diterima bukan 200, maka method akan mengembalikan nilai null, yang menandakan bahwa pengambilan data tidak berhasil. Selain itu, method ini juga dilengkapi dengan penanganan error menggunakan try-catch. Jika terjadi kesalahan saat proses request, seperti masalah koneksi atau server tidak merespons,

maka error akan ditangkap sebagai DioException, ditampilkan melalui debugPrint, dan method akan mengembalikan null.

Berikut adalah contoh response GetAll Contact jika menggunakan Postman menghasilkan status code 200 jika berhasil.



2. Buka File home_page.dart

Yang dilakukan pertama kali adalah membuat instance dari class ApiServices dan menyimpannya dalam variabel _dataService. Dan juga embuat instance dari class User dan menyimpannya dalam variabel _users. Variabel _users adalah sebuah list (daftar) yang bertipe List<User>. List tersebut dirancang untuk menyimpan sejumlah objek dari kelas User. Awalnya, variabel ini dideklarasikan tanpa nilai awal, sehingga pada saat dideklarasikan, list ini kosong ([]). Fungsi utama dari variabel _users adalah untuk menyimpan dan mengelola kumpulan data pengguna (User) dalam bentuk daftar di dalam aplikasi.

```
import 'package:dio_contact/model/contact_model.dart';
import 'package:dio_contact/services/api_services.dart';
```

```
class _HomePageState extends State<HomePage> {
  final _formKey = GlobalKey<FormState>();
  final _nameCtl = TextEditingController();
  final _numberCtl = TextEditingController();
  String _result = '-';
  final ApiServices _dataService = ApiServices();
  List<ContactsModel> _contactMdl = [];
```

Masih pada file home_page.dart, tambahkan method berikut

```
Future<void> refreshContactList() async {
    final users = await _dataService.getAllContact();
    setState(() {
      if (_contactMdl.isNotEmpty) _contactMdl.clear();
      if (users != null){
     // Konversi Iterable ke List, kemudian gunakan reversed
      _contactMdl.addAll(users .toList().reversed);
      }
    });
  }
```

Fungsi refreshContactList() mengambil daftar kontak dari _dataService dan berfungsi untuk memperbarui tampilan informasi terbaru. Pertama, fungsi ini meminta daftar kontak menggunakan metode getAllContact(). Selanjutnya, jika daftar kontak yang diambil tidak kosong (isNotEmpty), fungsi akan menghapus kontak yang sudah ada sebelumnya (_contactMdl). Kemudian, fungsi akan menambahkan kontak-kontak baru ke dalam _contactMdl dengan menggunakan metohod reversed untuk membalikkan urutan data sehingga kontak terbaru muncul di atas. Akhirnya, fungsi memperbarui tampilan dengan informasi terbaru menggunakan setState().

Kemudian tambahkan kode berikut pada properti onPressed button “Refresh Data”

```
child: ElevatedButton(
  onPressed: () async {
    await refreshContactList();
    setState(() {});
  },
  child: const Text('Refresh Data'),
), // ElevatedButton
```

Pada saat tombol ditekan akan memanggil method refreshContactList() yang sudah dibuat sebelumnya.

Selanjutnya kita buat widget ListView untuk menampilkan data yang akan kita GET. Buat method baru seperti berikut

```
Widget _buildListContact() {
    return ListView.separated(
        itemBuilder: (context, index) {
          final ctList = _contactMdl[index];
          return Card(
            child: ListTile(
              // leading: Text(user.id),
              title: Text(ctList.namaKontak),
              subtitle: Text(ctList.nomorHp),
              trailing: Row(
```

```
                mainAxisSize: MainAxisSize.min,
                children: [
                  IconButton(
                    onPressed: () async {},
                    icon: const Icon(Icons.edit),
                  ),
                  IconButton(
                    onPressed: () {},
                    icon: const Icon(Icons.delete),
                  ),
                ],
              ),
            ),
          );
        },
        separatorBuilder: (context, index) => const SizedBox(height:
10.0),
        itemCount: _contactMdl.length);
  }
```

Fungsi _buildListContact() merupakan sebuah widget yang membangun tampilan daftar kontak dalam bentuk ListView yang dapat kita scroll. Setiap item dalam ListView direpresentasikan sebagai sebuah Card yang berisi informasi kontak seperti nama dan nomor telepon dari objek _contactMdl. Setiap item Card terdiri dari ListTile yang menampilkan informasi kontak dan memiliki ikon edit dan hapus (dalam bentuk IconButton) di bagian kanan, tetapi saat ini fungsionalitas untuk tombol-tombol tersebut belum diimplementasikan. ListView juga memiliki separator antara setiap item menggunakan SizedBox dengan tinggi 10.0 dan jumlah item yang ditampilkan sesuai dengan panjang dari list _contactMdl.

Masih di home_page.dart kalian cari widget Expanded di paling bawah yang memiliki Widget Text yang memiliki nilai _result.

```
const SizedBox(height: 8.0),
Expanded(
  child: Text(_result),
),
const SizedBox(
  height: 20,
),
```

Kemudian ubah menjadi seperti gambar di bawah

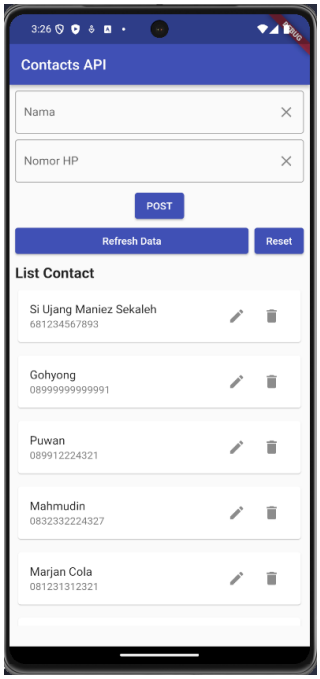
```
Expanded(
  child:
      _contactMdl.isEmpty ? Text(_result) : _buildListContact(),
),
```

Expanded memiliki kondisi yang mengecek apakah variabel **_contactMdl** kosong atau tidak. Jika **_contactMdl** kosong, maka akan ditampilkan teks dari variabel **_result**

menggunakan widget **Text**. Namun, jika **_contactMdl** tidak kosong, akan ditampilkan daftar pengguna yang telah diformat dengan menggunakan widget **_buildListContact()**, yang merupakan ListView yang menampilkan informasi data kontak dalam card terpisah. Dengan menggunakan kondisi ini, tampilan antarmuka aplikasi akan menyesuaikan tata letaknya berdasarkan apakah terdapat data pengguna yang dapat ditampilkan atau tidak.

3. **Jalankan aplikasi**

   Coba jalankan aplikasi kemudian tekan tombol Refresh Data, seharusnya hasilnya akan seperti berikut



F. Post Data

1. **Buka file api_services.dart**

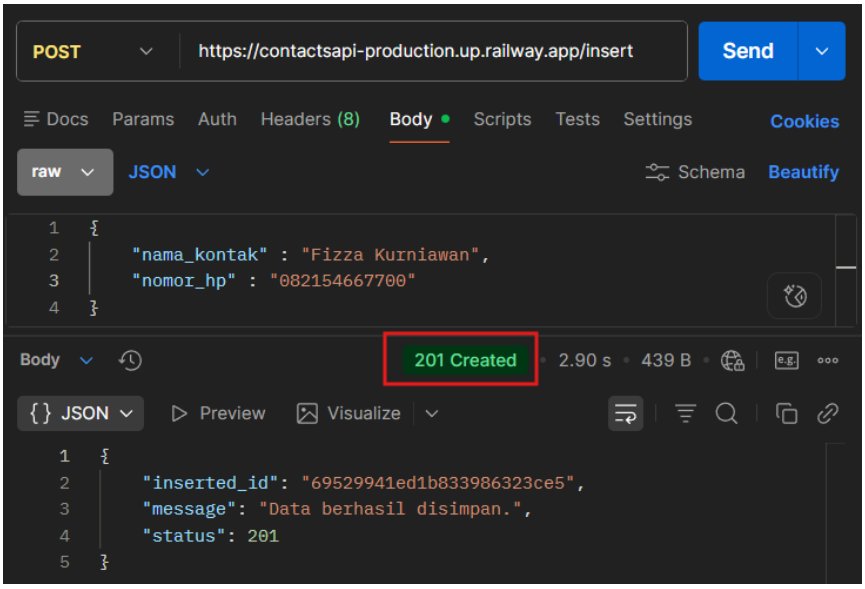
   Pada file api_services.dart tambahkan kode berikut pada method postContact

```
Future<ContactResponse?> postContact(ContactInput ct) async {
    try {
      final response = await dio.post('/insert', data: ct.toJson());
      if (response.statusCode == 201) {
        return ContactResponse.fromJson(response.data);
      }
      return null;
    } on DioException catch (e) {
      debugPrint('Dio error: ${e.response?.statusCode} -
${e.message}');
      return null;
    }
  }
```

Method postContact() digunakan untuk menambahkan data kontak baru ke server dengan melakukan permintaan HTTP POST ke endpoint /insert menggunakan package Dio. Data kontak dikirim dalam bentuk JSON yang berasal dari objek ContactInput melalui method toJson(). Proses pengiriman data dilakukan secara asynchronous dengan mekanisme async–await.

Jika server memberikan respons dengan status code 201 (Created), maka data respons akan dikonversi menjadi objek ContactResponse menggunakan method fromJson() dan dikembalikan sebagai hasil fungsi. Namun, jika terjadi kesalahan selama proses permintaan atau status code yang diterima bukan 201, maka error akan ditangani melalui DioException dan fungsi akan mengembalikan nilai null.

Kenapa status code 201? Karena response pada Backend jika berhasil adalah 201.

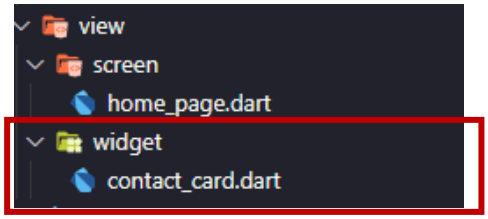


2. **Buka File home_page.dart**

Pada file home_page.dart, yang pertama dilakukan adalah membuat instance dari class ContactResponse dan menyimpannya dalam variabel ctRes. ctRes ini digunakan untuk menyimpan response ke dalam model ContactResponse yang ada pada folder model.

```
class _HomePageState extends State<HomePage> {
  final ApiServices _dataService = ApiServices();
  final _formKey = GlobalKey<FormState>();
  final _nameCtl = TextEditingController();
  final _numberCtl = TextEditingController();
  String _result = '-';
  List<ContactsModel>  contactMdl = [];
  ContactResponse? ctRes;
```

Kemudian kita akan menyimpan response tersebut ke dalam interface Widget. Jadi kita buat dahulu interface baru di dalam folder view→**widget→contact_card.dart**




```dart
import 'package:dio_contact/model/contact_model.dart';
import 'package:flutter/material.dart';

class ContactCard extends StatelessWidget {
  final ContactResponse ctRes;

  const ContactCard({Key? key, required this.ctRes}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(15),
      width: double.infinity,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(8),
        color: Colors.lightBlue[200],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          _buildDataRow('ID', ctRes.insertedId),
          _buildDataRow('Message', ctRes.message),
          _buildDataRow('Status', ctRes.status),
        ],
      ),
    );
  }

  Widget _buildDataRow(String label, dynamic value) {
    return Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(
          width: 60,
          child: Text(
            label,
            style: const TextStyle(
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
        const SizedBox(width: 10), // Jarak antara label dan titik dua
        Expanded(
```

```
          child: Text(
            ': $value',
            softWrap: true,
          ),
        ),
      ],
    );
  }
}
```

Selanjutnya kita buat widget Column untuk menampilkan ContactCard yang sudah dibuat di atas pada saat data diinputkan (POST). Buat method baru seperti berikut pada home_page.dart

```
Widget hasilCard(BuildContext context) {
    return Column(children: [
      if (ctRes != null)
        ContactCard(
          ctRes: ctRes!,
        )
      else
        const Text(''),
    ]);
  }
```

Jangan lupa import ContactCard

```
import 'package:dio_contact/view/widget/contact_card.dart';
```

Panggil method hasilCard di bawah Button Post dan di atas Button Refresh Data.

Selanjutnya pada button POST pada onPressed buatlah seperti kode di bawah

```
ElevatedButton(
      onPressed: () async {
        if (_nameCtl.text.isEmpty ||
            _numberCtl.text.isEmpty) {
          displaySnackbar('Semua field harus diisi');
          return;
        }
        final postModel = ContactInput(
          namaKontak: _nameCtl.text,
          nomorHp: _numberCtl.text,
        );

        ContactResponse? res;
        res = await _dataService.postContact(postModel);

        setState(() {
          ctRes = res;
        });
        _nameCtl.clear();
        _numberCtl.clear();
        await refreshContactList();
      },
      child: const Text('POST'),
    ),
```

1. Terdapat pengecekan kondisi, jika **_nameCtl** atau **_numberCtl** kosong, maka akan ditampilkan pesan notifikasi menggunakan **displaySnackbar('Semua field harus diisi')**, dan fungsi ini akan berhenti untuk mencegah eksekusi lebih lanjut.
2. Selanjutnya, dibuat objek **postModel** menggunakan informasi yang diambil dari **_nameCtl** (nama kontak) dan **_numberCtl** (nomor telepon).
3. Dilakukan pemanggilan async function **_dataService.postContact(postModel)**, yang berfungsi untuk mengirim data kontak baru ke server dengan menggunakan objek **postModel** yang telah dibuat sebelumnya. Hasil dari pemanggilan ini disimpan dalam variabel **res**, yang merupakan objek **ContactResponse?**.
4. Menggunakan **setState()**, nilai dari variabel **ctRes** diperbarui dengan nilai dari **res** yang diperoleh dari pemanggilan **_dataService.postContact(postModel)**. Yang dimana akan menyebabkan perubahan pada tampilan sesuai dengan nilai yang diperbarui.
5. Menggunakan .clear() untuk mengosongkan nilai dari **_nameCtl** dan **_numberCtl** saat data sudah disubmit.

6. Terakhir, dipanggil **await refreshContactList()** untuk memperbarui daftar kontak setelah menambahkan kontak baru ke dalamnya.

Selanjutnya sekarang buat validasi form nama dan nomor hp terlebih dahulu. Pada file home_page.dart tambahkan kode berikut

```
String? _validateName(String? value) {
    if (value != null && value.length < 4) {
      return 'Masukkan minimal 4 karakter';
    }
    return null;
  }

  String? _validatePhoneNumber(String? value) {
    if (!RegExp(r'^[0-9]+$').hasMatch(value!)) {
      return 'Nomor HP harus berisi angka';
    }
    return null;
  }
```

Kemudian pada widget TexFormField nama dan nomor hp tambahkan properties validator

```
children: [
  TextFormField(
    controller: _nameCtl,
    validator: _validateName,
    decoration: InputDecoration(
      border: const OutlineInputBorder(),
      labelText: 'Nama',
      suffixIcon: IconButton(
        onPressed: _nameCtl.clear,
        icon: const Icon(Icons.clear),
      ), // IconButton
    ), // InputDecoration
  ), // TextFormField
  const SizedBox(height: 8.0),
  TextFormField(
    controller: _numberCtl,
    validator: _validatePhoneNumber,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      border: const OutlineInputBorder(),
      labelText: 'Nomor HP',
```

Kemudian tambahkan kode berikut pada widget ElevatedButton POST

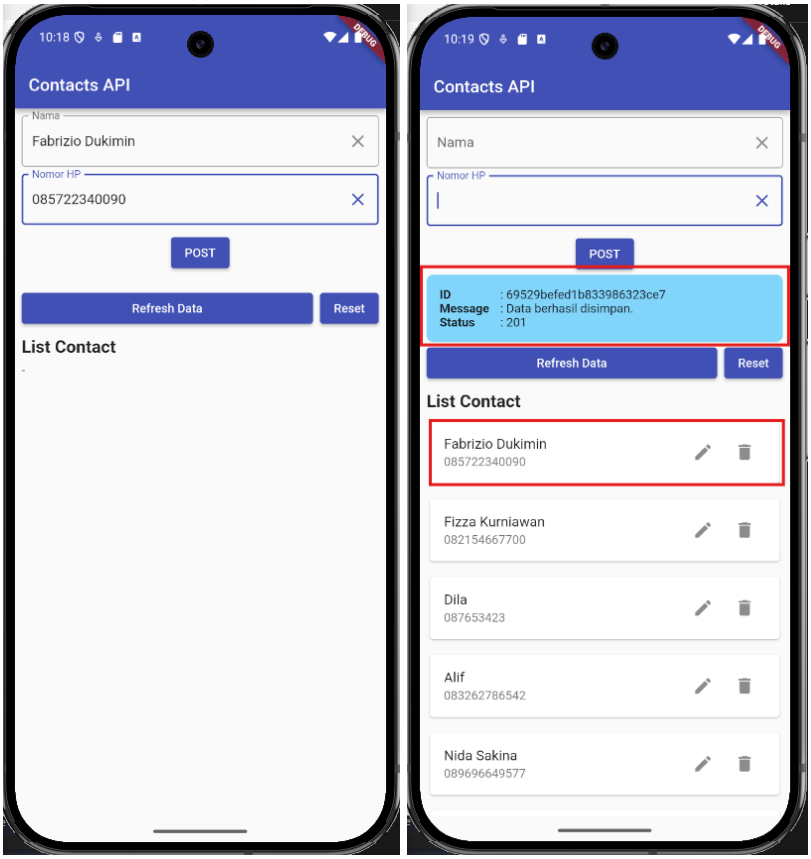
```
ElevatedButton(
  onPressed: () async {
    final isValidForm = _formKey.currentState?.validate();
    if (_nameCtl.text.isEmpty ||
        _numberCtl.text.isEmpty) {
      displaySnackbar('Semua field harus diisi');
      return;
    } else if (!isValidForm!) {
      displaySnackbar("Isi form dengan benar");
      return;
    }

    final postModel = ContactInput(
```

Kita menggunakan _formKey.currentState!.validate() untuk memvalidasi semua field dalam form.

3. **Jalankan aplikasi**

Coba jalankan aplikasi kemudian isi form tekan tombol post, seharusnya hasilnya akan seperti berikut



G. Reset Form dan Hapus Response

1. **Reset Form**

Untuk melakukan reset form menggunakan button Reset kita hanya perlu menambahkan kode berikut pada bagian onPressed Button Reset di home_page.dart

```
onPressed: () {
                    setState(() {
                      _result = '-';
                      _contactMdl.clear();
                      ctRes = null;
                    });
                  },
```

Sekarang coba tekan tombol Reset.

2. **Hapus Response**

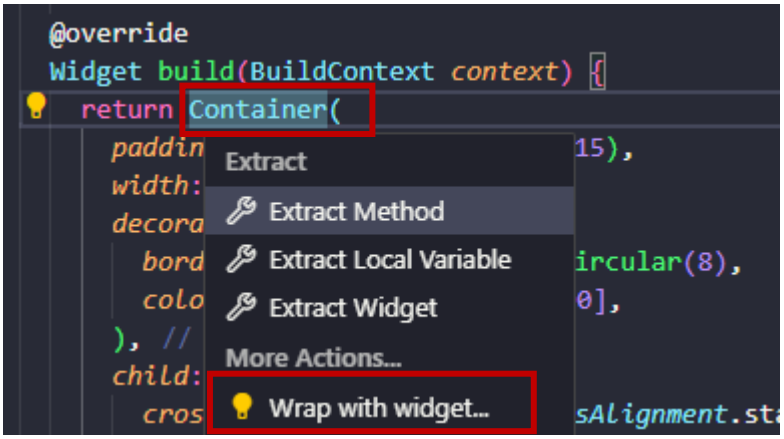
Selanjutnya kita akan buat menghapus response berikut dengan cara menggeser ke kanan atau ke kiri saja

Yang pertama harus dilakukan adalah buka file contact_card.dart di dalam folder view→widget. Kemudian tambahkan propery onDismissed dan juga panggil di dalam constructor.

```
class ContactCard extends StatelessWidget {
  final ContactResponse ctRes;
  final Function() onDismissed;

  const ContactCard({Key? key, required this.ctRes, required this.onDismissed})
      : super(key: key);
```

Kemudian lihat pada Widget Container, kita akan lakukan refactor (shortcut **: Ctrl + .**) wrap widget Container dengan Widget Dismissible



Sehingga menjadi seperti ini

```
Widget build(BuildContext context) {
  return Dismissible(
    child: Container(
      padding: const EdgeInsets.all(15),
      width: double.infinity,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(8),
        color: Colors.lightBlue[200],
      ), // BoxDecoration
```

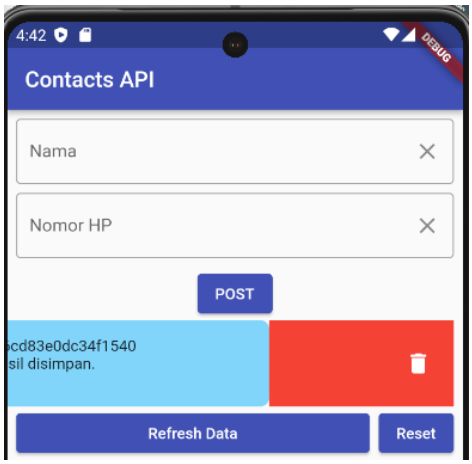
Namun widget Dismissible masih error, tambahkan kode berikut sehingga tidak terjadi error

```
@override
  Widget build(BuildContext context) {
    return Dismissible(
      key: Key(ctRes.message),
      onDismissed: (direction) {
        onDismissed();
      },
      background: Container(
        color: Colors.red,
        alignment: Alignment.centerRight,
        padding: const EdgeInsets.only(right: 20.0),
        child: const Icon(Icons.delete, color: Colors.white),
      ),
      child: Container(
        padding: const EdgeInsets.all(15),
        width: double.infinity,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(8),
          color: Colors.lightBlue[200],
        ),
```

Karena tadi kita menambah 1 nilai di constructor, pada file home_page.dart pasti terdapat error, oleh karena itu kita buka file home_page.dart kemudian tambah properti onDismissed untuk menghapus Response.

```
Widget hasilCard(BuildContext context) {
    return Column(children: [
      if (ctRes != null)
        ContactCard(
          ctRes: ctRes!,
          onDismissed: () {
            setState(() {
              ctRes = null;
            });
          },
        )
      else
        const Text(''),
    ]);
  }
```

Sekarang coba kita hapus Card hasil response dengan cara geser ke kiri atau kanan



H. Put Data

1. Buka file api_services.dart

Sebelum melakukan put tentunya kita harus bisa mendapatkan id dari data yang akan di edit, oleh karena itu kita akan buat dulu pemanggilan data berdasarkan Contact yang dipilih. Pada file api_services.dart tambahkan kode berikut pada method getSingleContact.

```
Future<ContactsModel?> getSingleContact(String id) async {
    try {
      final response = await dio.get('/contacts/$id');

      if (response.statusCode == 200) {
        return ContactsModel.fromJson(response.data);
      }
      return null;
    } on DioException catch (e) {
      debugPrint('Dio error: ${e.response?.statusCode} -
${e.message}');
      return null;
    }
  }
```

Kode di atas kurang lebih sama dengan service getAllContact, namun pada service getSingleContact data yang dikembalikan hanya 1 data saja.

2. Buka File home_page.dart

Selanjutnya kita inisialisasi dan deklarasi variabel isEdit dan idContact.

```dart
class _HomePageState extends State<HomePage> {
  final _formKey = GlobalKey<FormState>();
  final _nameCtl = TextEditingController();
  final _numberCtl = TextEditingController();
  String _result = '-';
  final ApiServices _dataService = ApiServices();
  List<ContactsModel> _contactMdl = [];
  ContactResponse? ctRes;
  bool isEdit = false;
  String idContact = '';
```

isEdit adalah sebuah boolean yang mengindikasikan apakah aplikasi saat ini sedang dalam mode pengeditan atau tidak. idContact adalah sebuah string yang digunakan untuk menyimpan ID dari kontak yang sedang diedit.

Cari widget IconButton Edit pada widget _buildListContact

```dart
children: [
  IconButton(
    onPressed: () async {},
    icon: const Icon(Icons.edit),
  ), // IconButton
  IconButton(onPressed: () {}, icon: const Icon(Icons.delete)),
],
```

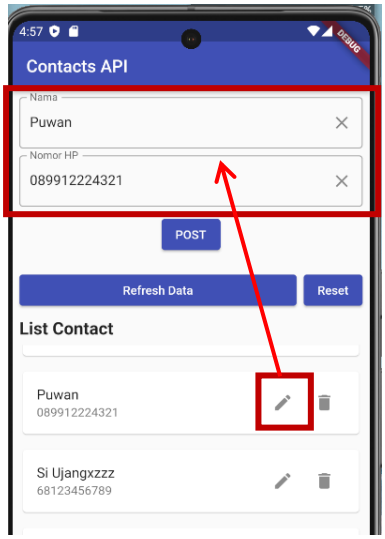
Isi property onPressed seperti di bawah

```dart
onPressed: () async {
                    final contacts =
                        await _dataService.getSingleContact(ctList.id);
                    setState(() {
                      if (contacts != null) {
                        _nameCtl.text = contacts.namaKontak;
                        _numberCtl.text = contacts.nomorHp;
                        isEdit = true;
                        idContact = contacts.id;
                      }
                    });
                  },
```

Fungsi kode di atas pada property onPressed, yaitu:

1. Menerima data kontak dari server menggunakan **_dataService.getSingleContact(ctList.id)**, dimana **ctList.id** mengacu pada ID dari kontak yang dipilih.
2. Jika data kontak berhasil diperoleh (**contacts != null**), nilai-nilai dari nama kontak dan nomor telepon dari data kontak tersebut dimasukkan ke dalam input field yang sesuai (**_nameCtl.text** dan **_numberCtl.text**).
3. Selain itu, variabel **isEdit** diatur menjadi **true**, yang menunjukkan bahwa aplikasi berada dalam mode pengeditan, dan **idContact** diisi dengan ID dari kontak yang sedang diubah.

Sekarang coba klik button Edit, maka data kontak yang dipilih akan ditampilkan pada input field di atasnya



Kemudian sekarang selanjutnya kita akan membuat Button Post berubah menjadi Update jika kondisi aplikasi sedang berada dalam mode edit, cari Widget Text untuk button POST

```
child: const Text('POST'),
```

Kemudian ubah menjadi

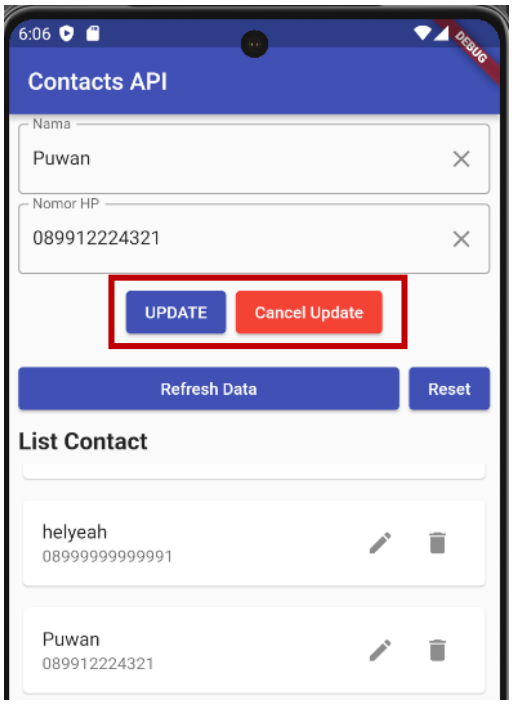
```
child: Text(isEdit ? 'UPDATE' : 'POST'),
```

Sekarang ketika klik button edit maka Button akan berubah menjadi Update.

Ingat…. masih belum bisa update karena kita belum buat servicenya. Selanjutnya adalah kita akan membuat button untuk cancel update jikalau kita tidak jadi untuk mengubah data. Tambah kode berikut di bawah Elevated Button POST atau UPDATE

```
child: Text(isEdit ? 'UPDATE' : 'POST'),
                    ),
                    if (isEdit)
                      ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.red,
                        ),
                        onPressed: () {
                          _nameCtl.clear();
                          _numberCtl.clear();
                          setState(() {
                            isEdit = false;
                          });
                        },
                        child: const Text('Cancel Update'),
                      ),
```

Kode di atas mengecek apakah aplikasi sedang dalam mode pengeditan (isEdit = true). Jika kondisi tersebut terpenuhi, tombol ElevatedButton ditampilkan dengan teks "Cancel Update" yang ketika ditekan akan mengosongkan input field untuk nama dan nomor telepon (_nameCtl dan _numberCtl), dan mengubah variabel isEdit menjadi false. Proses ini bertujuan untuk memberikan pengguna opsi untuk membatalkan proses pengeditan yang sedang berlangsung, membersihkan input field, dan mengembalikan aplikasi ke mode penambahan kontak baru.



3. **Buka file api_services.dart**

Selanjutnya kita akan buat service untuk update. Tambahkan kode berikut pada method putContact.

```
Future<ContactResponse?> putContact(String id, ContactInput ct) async {
    try {
      final response = await dio.put('/update/$id', data: ct.toJson());
      if (response.statusCode == 200) {
        return ContactResponse.fromJson(response.data);
      }
      return null;
    } on DioException catch (e) {
      debugPrint('Dio error: ${e.response?.statusCode} -
${e.message}');
      return null;
    }
  }
```

Kode di atas kurang lebih sama dengan service postContact, namun pada service ini kita menambahkan 1 parameter id yang digunakan untuk mengupdate data secara spesifik sesuai id yang dipilih.

4. **Buka File home_page.dart**

Pada file home_page.dart, tambahkan kode berikut pada properti onPressed button UPDATE atau POST

Ganti baris berikut

```
ContactResponse? res;
res = await _dataService.postContact(postModel);
```

Menjadi

```
ContactResponse? res;
if (isEdit) {
  res = await _dataService.putContact(
      idContact, postModel);
} else {
  res = await _dataService.postContact(postModel);
}
```

Kemudian masih di button UPDATE atau POST bisa dilihat pada setState berikut

```
setState(() {
  ctRes = res;
});
_nameCtl.clear();
_numberCtl.clear();
await refreshContactList();
```

Tambahkan kondisi isEdit = false

```
setState(() {
  ctRes = res;
  isEdit = false;
});
_nameCtl.clear();
_numberCtl.clear();
await refreshContactList();
```

Penjelasan dari penambahan kode di atas, sebagai berikut :

a) **if (isEdit)** { **...** }: Kondisi yang mengecek apakah aplikasi sedang dalam mode pengeditan (**isEdit = true**). Jika kondisi ini benar, maka akan melakukan permintaan ke server untuk memperbarui kontak yang telah ada dengan menggunakan metode **putContact**. Data yang dikirimkan adalah **postModel** (informasi kontak yang baru) ke

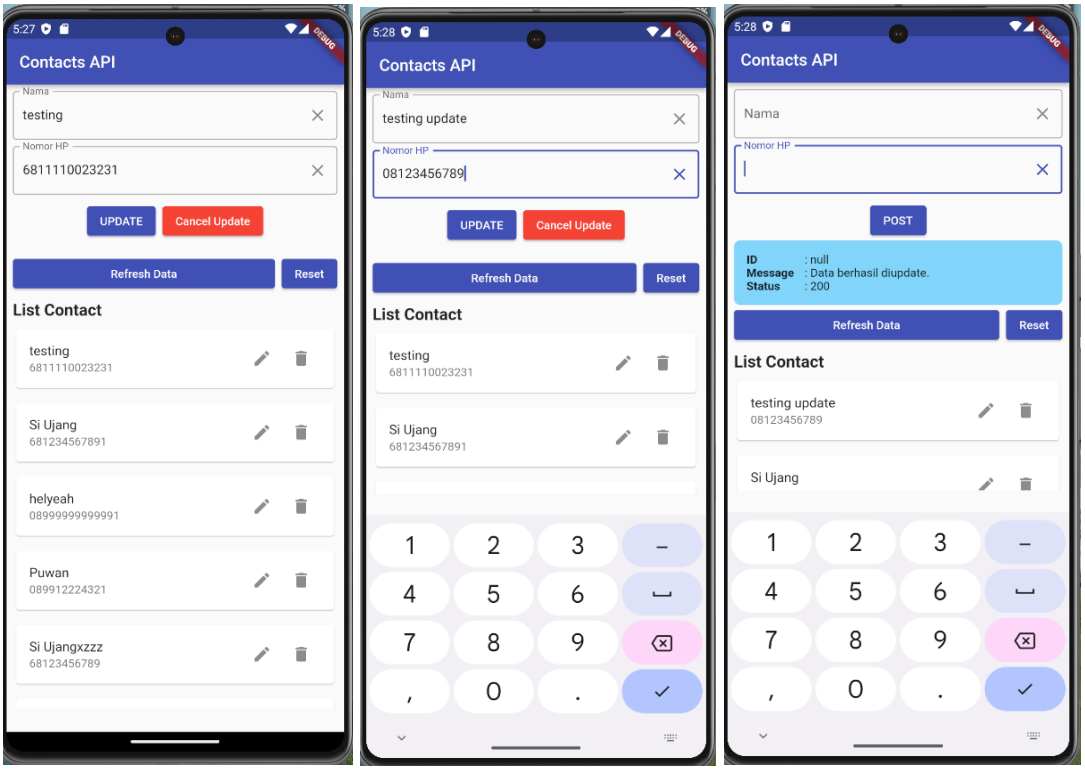
endpoint yang sesuai dengan ID kontak yang sedang diubah (**idContact**). Hasil respons dari server akan disimpan dalam variabel **res**.

b) **else { ... }**: Jika kondisi **isEdit** bernilai **false**, itu berarti aplikasi sedang dalam mode penambahan kontak baru. Dalam kasus ini, aplikasi akan memanggil **await _dataService.postContact(postModel);** untuk menambahkan kontak baru ke server.

c) **isEdit = false;**: Setelah operasi (entah itu pengeditan atau penambahan) selesai, aplikasi mengatur kembali **isEdit** menjadi **false**. Hal ini menunjukkan bahwa aplikasi kembali ke mode penambahan kontak baru, bukan mode pengeditan. Hal ini memastikan bahwa langkah selanjutnya yang diambil oleh pengguna dianggap sebagai penambahan kontak baru.

5. Jalankan aplikasi

Coba jalankan aplikasi kemudian isi form tekan tombol Update, seharusnya hasilnya akan seperti berikut



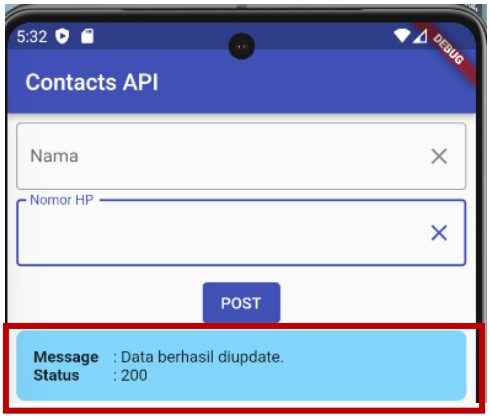
Coba lihat response dari hasil update (gambar paling kanan) ID : null, untuk mengatasi hal tersebut kita coba buka file contact_card.dart tambahkan baris kode if untuk mengecek apakah response insertedId bukan null. Ganti baris berikut

```
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    _buildDataRow('ID', ctRes.insertedId),
    _buildDataRow('Message', ctRes.message),
    _buildDataRow('Status', ctRes.status),
  ],
```

Diganti dengan

```
children: [
  if (ctRes.insertedId != null) _buildDataRow('ID', ctRes.insertedId),
  _buildDataRow('Message', ctRes.message),
  _buildDataRow('Status', ctRes.status),
],
```

Sekarang ketika mengupdate data maka response ID tidak akan ditampilkan



I. Delete Data

1. Buka file api_services.dart

Pada file api_services.dart tambahkan kode berikut pada method deleteContact

```
Future<ContactResponse?> deleteContact(String id) async {
    try {
      final response = await dio.delete('/delete/$id');
      if (response.statusCode == 200) {
        return ContactResponse.fromJson(response.data);
      }
      return null;
    } on DioException catch (e) {
      debugPrint('Dio error: ${e.response?.statusCode} -
${e.message}');
      return null;
    }
  }
```

Method deleteContact() digunakan untuk menghapus data kontak dari server dengan melakukan permintaan HTTP DELETE ke endpoint /delete/{id}, di mana id merupakan

identitas kontak yang akan dihapus. Proses penghapusan dilakukan secara asynchronous menggunakan async–await melalui package Dio.

Jika server mengembalikan respons dengan status code 200 (OK), maka data respons akan dikonversi menjadi objek ContactResponse dan dikembalikan sebagai hasil fungsi. Namun, jika terjadi kesalahan selama proses permintaan atau respons tidak berhasil, maka error akan ditangani menggunakan DioException dan fungsi akan mengembalikan nilai null.

2. **Buka File home_page.dart**

Pada file home_page.dart, kita buat sebuah proses penghapusan data sekaligus dengan konfirmasi untuk melakukan penghapusan data menggunakan widget AlertDialog. Buat method berikut

```
void _showDeleteConfirmationDialog(String id, String nama) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Konfirmasi Hapus'),
          content: Text('Apakah Anda yakin ingin menghapus data $nama ?'),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('CANCEL'),
            ),
            TextButton(
              onPressed: () async {
                ContactResponse? res = await _dataService.deleteContact(id);
                setState(() {
                  ctRes = res;
                });
                Navigator.of(context).pop();
                await refreshContactList();
              },
              child: const Text('DELETE'),
            ),
          ],
        );
      },
    );
  }
```

tambahkan kode berikut pada properti onPressed button delete dengan memanggil method yang tadi sudah dibuat di atas

```
IconButton(
                    onPressed: () {
                      _showDeleteConfirmationDialog(
                          ctList.id, ctList.namaKontak);
                    },
                    icon: const Icon(Icons.delete),
                  ),
```

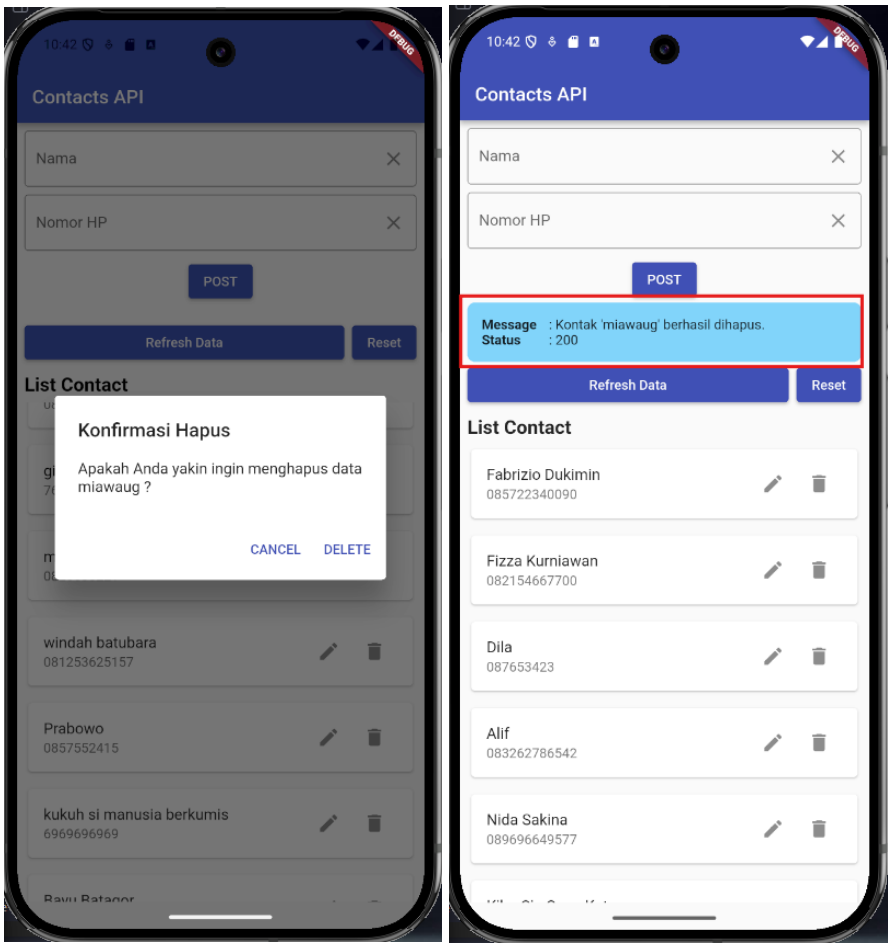
_showDeleteConfirmationDialog() yang digunakan untuk menampilkan dialog konfirmasi penghapusan saat pengguna menekan icon delete pada item dalam list kontak. Ketika icon delete di tekan, fungsi **_showDeleteConfirmationDialog()** dipanggil dengan ID dan NAMA dari item kontak yang ingin dihapus.

Penjelasannya adalah sebagai berikut:

1. Fungsi **_showDeleteConfirmationDialog(String id, String nama)** menginisiasi sebuah dialog AlertDialog yang menampilkan pesan konfirmasi untuk memastikan keinginan pengguna dalam menghapus data. Pesan konfirmasi ini berisi pertanyaan "Apakah Anda yakin ingin menghapus data (nama kontak yang dipilih) ?".
2. Dalam dialog tersebut, terdapat dua pilihan aksi yaitu 'CANCEL' dan 'DELETE', diimplementasikan sebagai dua TextButton di dalam actions dari AlertDialog.
3. Jika pengguna memilih 'CANCEL', dialog ditutup menggunakan **Navigator.of(context).pop();**.
4. Jika pengguna memilih 'DELETE', fungsi **_dataService.deleteContact(id);** dipanggil untuk menghapus kontak yang sesuai dengan ID yang diberikan. Hasil respons dari server kemudian disimpan dalam variabel **res**.
5. Setelah itu, **setState()** digunakan untuk mengupdate variabel **ctRes** dengan nilai dari respons yang didapat dari server.
6. Dialog ditutup menggunakan **Navigator.of(context).pop();**.
7. Terakhir, pemanggilan **await refreshContactList();** digunakan untuk memperbarui daftar kontak setelah kontak yang dipilih dihapus, sehingga perubahan dapat segera terlihat dalam aplikasi.

3. **Jalankan aplikasi**

Coba jalankan aplikasi kemudian isi form tekan tombol delete, seharusnya hasilnya akan seperti berikut



Pengumpulan Hasil :

Buat folder dengan nama **Pertemuan12** kemudian di dalamnya buat 2 folder yaitu **folder Praktikum** dan **Output**.

Isi folder **Praktikum** adalah aplikasi flutter yang sudah dibuat, tetapi sebelum melakukan push ke github wajib jalankan perintah **"flutter clean"** pada VSCode. Isi folder **Output** adalah hasil output dari aplikasi flutter yang sudah dibuat (berupa video dengan format gif).

Setelah melakukan perintah **flutter clean** kalian bisa langsung push ke dalam repository kalian.