# GUIDE. 3

## Capaian

1. Mahasiswa mampu membuat fungsi fungsi CRUD dengan dilengkapi error handling
2. Mahasiswa mampu membuat komponen UI yang biasa digunakan untuk melakukan operasi CRUD database

Dokumen ini bertujuan untuk menjadi dokumen pengantar mataeri pembelajaran Pemrograman Flutter dengan Database SQlite. Mahasiswa diharapkan mampu membuat widget komponen form dan menghubungkan operasi CRUD ke widget.

## Membuat Class ContactForm pada File contact_form.dart

1. Tambahkan package material, dan import class DbHelper dan model Contact

```
import 'package:flutter/material.dart';
import 'package:simple_database/utils/dbhelper.dart';
import '../utils/contact.dart';
```

2. Buat class ContactForm pada file contact_form.dart yang memiliki variable ‘contact’ bertipe data ‘Contact’ dan ditandai sebagai nullable. Class ContactForm memiliki parameter contact.
3. Buat konstruktor kelas ContactForm menggunakan parameter kontak opsional. Parameter kontak ditetapkan ke variabel anggota kontak menggunakan sintaks singkatan {this.contact}.
4. Tambahkan anotasi @override. Anotasi ini menunjukkan bahwa metode berikut menggantikan metode dari superclass. Dalam hal ini, metode createState dari kelas StatefulWidget sedang diganti.
5. Buat metode createState yang mengembalikan instance kelas ContactFormState.

```
class ContactForm extends StatefulWidget{
  final Contact? contact;
  const ContactForm({this.contact});
  @override
  ContactFormState createState() => ContactFormState();
}
```

**Membuat Class ContactFormState pada File contact_form.dart**

1. Buat class ContactFormState pada file yang sama. Class ini mewarisi kelas State dan dikaitkan dengan kelas ContactForm.
2. Tambahkan beberapa variabel berikut di class ContactFormState

| Tipe data, Kata kunci | Nama Variabel | Value |
|---|---|---|
| DbHelper | db | DbHelper() |
| TextEditingController? | nameController<br>numberController<br>emailController<br>companyController | -<br>-<br>-<br>- |
| final | _form | GlobalKey<FormState>() |

Berikut contoh deklarasi variabel diatas:

```
DbHelper db = DbHelper();
TextEditingController? nameController;
 …
 …
 …
final _form = GlobalKey<FormState>();
```

3. Tambahkan anotasi @override setelah deklarasi dan inisialisasi variabel
4. Buat metode initState(). metode ini biasanya digunakan di Flutter untuk menginisialisasi status widget saat pertama kali dibuat.
5. Di dalam metode initState(), inisialisasi 4 object TextEditingController yaitu **nameController, numberController, emailController, dan companyController** dan menetapkan nilai awal berdasarkan objek ‘widget.contact’. Seperti dibawah ini

```
nameController = TextEditingController(
      text: widget.contact == null ? '' :
widget.contact!.name);
```

6. Setelah penambahan beberapa object diatas, tambahkan ‘super.initState();’. Kode ini memanggil metode ‘initState()’ dari superclass, yang biasanya diperlukan untuk memastikan bahwa logika inisialisasi kelas dasar dijalankan dengan benar.
7. Buat metode ‘build’ yang mewarisi dari metode ‘build’ superclass menggunakan anotasi @override. Beikut cuplikan kode method build
8. Di dalam metode ‘build’, kode mengembalikan widget ‘Scaffold’. Scaffold merupakan widget yang umum digunakan untuk struktur dasar membuat aplikasi desain material. Tambahkan informasi berikut pada widget Scaffold:
    a. key: Key(‘scaffold_contactform’)

9. Tambahkan widget AppBar dengan informasi berikut ini

| Parameter | Widget | Value |
|---|---|---|
| key | Key | contact_form_appbar |
| title | Text | Contact Form,<br><br>key:<br>Key(‘titlte_appbar_contactform’) |

10. Setelah itu tambahakan body widget dengan informasi berikut ini:

| | | | |
|---|---|---|---|
| body | Form() | key<br>child | _form<br>ListView() |

11. Pada ListView widget tambahkan parameter padding untuk mengatur jarak dan isi konten, serta tambahkan paraemter children yag dapat menampung banyak widget. Kode seperti dibawah ini:

```
child: ListView(
    key: Key(‘listview’),
    padding: const
EdgeInsets.all(16.0),
    children: [
        ...
```

12. Pada children buatlah widget Padding yang memiliki parameter child Text. Berikut rincian informasi nya:

| Parameter | Value |
|---|---|
| key | Key(‘padding_name’) |
| padding | EdgeInsets.only(top: 20) |
| child | TextFormField() |

13. Pada TextFormField tambahkan informasi berikut ini:
    a. controller: nameContoller
    b. key: Key(‘name_textformfield’)
    c. decoration: InputDecoration()
    d. validator: (value){}
14. Pada inputDecoration tambahkan parameter berikut:
    a. labelText: ‘Name’

b. border: OutlineInputBorder()

15. Pada OutlineInputBorder tambahkan parameter berikut:

    a. borderRadius: BorderRadius.circular(8)

16. Pada parameter validator di widget TextFormField, tambahkan kondisi jika '`value!.isEmpty`' akan mengembalikan nilai '`Please enter your name`'.

17. Berikut cuplikan kode pada informasi diatas

```
Padding(
        key: Key('padding_name'),
        padding: const EdgeInsets.only(
          top: 20,
        ),
        child: TextFormField(
          key: Key('name_textformfield'),
          controller: …,
          decoration: InputDecoration(
              labelText: …,
              border: OutlineInputBorder(
                borderRadius: …,
              )),
          validator: (value){
            if(…){
              return '…';
            }
          },
```

18. Ulangi lankah 11-16 sebanyak 3 kali. Lakukan penyesuaian pada parameter key, labelText, dan controller untuk data Name, Number, dan Company.

19. Tambahkan 1 widget padding lagi yang memiliki parameter child Elevated Button.

```
Padding(
  padding: const EdgeInsets.only(top: 20),
  key: Key('padding_button'),
child: ElevatedButton(
    key: Key('elevatedbutton_contactform'),
  child: (widget.contact == null)
? const Text('Add',
    key:Key('add_text'),
    style: TextStyle(color: Colors.white),)  // Text
    : const Text('Update',
    key:Key('update_text'),
    style: TextStyle(color: Colors.white),),  // Text
  onPressed: (){
    if(_form.currentState!.validate()){
      _form.currentState!.save();
      upsertContact();
    }
  },
),)  // ElevatedButton, Padding
```

20. Setelah itu buatlah sebuah metode upsertContact() yang dipanggil pada saat menyimpan dan mengedit data.

```
Future<void> upsertContact() async {
    if (widget.contact != null) {
      //update
      await db.updateContact(Contact(
          id: widget.contact!.id,
          name: nameController!.text,
          number: numberController!.text,
          email: emailController!.text,
          company: companyController!.text
      ));
      Navigator.pop(context, 'update');
      ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(
        content: Text('Success update data'),
      ));
    } else {
      //insert
      await db.saveContact(Contact(
        name: nameController!.text,
        number: numberController!.text,
        email: emailController!.text,
        company: companyController!.text,
      ));
      Navigator.pop(context, 'save');
      ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(
        content: Text('Success add data'),
      ));
    }
  }
```

**Testing Project**

1. Unduh code file guide3_1_test.dart dan guide3_2_test.dart, kemudian pindahkan ke folder test pada project anda

2. Jalanka testingnya. Jika berhasil semua maka akan muncul keterangan ‘Test Success’ dan jika ada yang gagal maka akan muncul keterangan ‘Test Fail’ disertai keterangan bagian mana yang fail.