

LAPORAN TUGAS BESAR
PEMROGRAMAN MOBILE
PROGRAM APLIKASI TOKO ATK



Disusun Oleh:

Ahmad Ariza Alghandi (201910370311178)

Amrul Maulidi (201910370311190)

FAKULTAS TEKNIK
JURUSAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

2021

KATA PENGANTAR

Puji syukur Penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya penyusunan laporan tugas besar “Pemrograman Mobile” dapat diselesaikan dengan baik. Penulis menyadari bahwa dalam proses penulisan laporan praktikum ini banyak mengalami kendala, namun kendala-kendala dapat diatasi.

Laporan tugas besar ini dibuat dalam rangka memenuhi tugas Pemrograman Mobile selain itu laporan ini dibuat untuk dokumentasi mengenai implementasi yang dilakukan kami selaku mahasiswa jurusan informatika Universitas Muhammadiyah Malang..

Penulis menyadari betul sepenuhnya bahwa Laporan tugas besar ini masih jauh dari sempurna. Oleh karena itu, dengan segala kerendahan hati Penulis berharap saran dan kritik demi perbaikan-perbaikan lebih lanjut. Penulis berharap semoga Laporan ini dapat memberikan manfaat bagi pembaca

BAB 1

PENDAHULUAN

Latar Belakang

Seiring dengan kemajuan dan perkembangan zaman di era globalisasi ini, manusia dituntut mengikuti perkembangan ilmu pengeahuan dan teknologi. Semakin banyak perangkat lunak (Software), dan semakin meningkatnya kecanggihan perangkat keras (Hardware), maka perangkat kompuer sebagai alat bantu menjadi semakin berperan dalam menyelesaikan suatu pekerjaan Tersedianya banyak Bahasa-bahasa pemrograman komputer mendorong dari segala kalangan dari anak kecil sampai dengan dewasa saat ini bisa mempelajarinya karena dimudahkan dalam mempelajarinya dengan disuguhi interface yang menarik dan mudah untuk dipahami, apalagi pada saat ini ilmu bisa kita gali dengan bebas di internet, youtube dan sebagainya. Contoh kecil pada saat ini yang memudahkan para developer yaitu dibuatnya framework aplikasi mobile yang bernama Flutter yang diciptakan oleh Google. Flutter ini digunakan dalam pengembangan aplikasi untuk system operasi Android, iOS, Windows, Linux, MacOS, serta menjadi metode utama untuk membuat aplikasi Google Fuchia.

Jika kita ingin membangun sebuah Toko ATK langsung pada kehidupan nyata pasti sangat mahal biayanya, banyak yang harus kita siapkan, akan tetapi jika kita membangun sebuah Toko virtual / online kita tidak membutuhkan biaya yang sangat banyak, karena kita cukup mempunyai tempat untuk menampung barang dan semua barang kita bisa di promosikan pada Toko virtual yang kita buat. Karena itulah Flutter hadir menggunakan Bahasa Dart untuk memudahkan manusia dalam pembuatan aplikasi mobile salah satunya aplikasi Toko ATK.

Tujuan

Tujuan dari pembuatan Aplikasi Toko ATK

- Implementasi dari matakuliah Pemrograman Mobile
- Menyelesaikan Tugas Besar

BAB II

LANDASAN TEORI

Pembuatan Aplikasi Toko ATK menggunakan framework flutter :

1. Widget flutter

Widget Flutter dibuat menggunakan kerangka kerja modern yang mengambil inspirasi dari [React](#) . Ide utamanya adalah Anda membangun UI dari widget. Widget menjelaskan seperti apa tampilan mereka jika diberikan konfigurasi dan status saat ini.

Aplikasi Flutter minimal hanya memanggil *runApp()*

The runApp() Fungsi mengambil diberikan Widget dan membuat akar pohon widget. Dalam contoh ini, pohon widget terdiri dari dua widget, Centerwidget dan anaknya, Textwidget. Kerangka kerja memaksa widget root untuk menutupi layar, yang berarti teks "Halo, dunia" berakhir di tengah layar. Arah teks perlu ditentukan dalam contoh ini; ketika MaterialAppwidget digunakan, ini diurus untuk Anda, seperti yang ditunjukkan nanti. Sebuah SafeAreawidget juga digunakan untuk benar pad teks sehingga muncul di bawah layar di bagian atas layar.

Saat menulis aplikasi, Anda biasanya akan membuat widget baru yang merupakan subkelas dari salah satu StatelessWidgetatau StatefulWidget, bergantung pada apakah widget Anda mengelola status apa pun. Tugas utama widget adalah mengimplementasikan build()fungsi, yang menjelaskan widget dalam kaitannya dengan widget tingkat rendah lainnya. Kerangka kerja membangun widget-widget tersebut secara bergiliran hingga proses berakhir di widget yang mewakili yang mendasarinya RenderObject, yang menghitung dan menjelaskan geometri widget.

Flutter hadir dengan rangkaian widget dasar yang kuat, yang biasanya digunakan sebagai berikut:

Text

The Textwidget memungkinkan Anda membuat lari dari teks bergaya dalam aplikasi Anda.

Row, Column

Widget fleksibel ini memungkinkan Anda membuat tata letak yang fleksibel dalam arah horizontal (Row) dan vertikal (Column). Desain objek ini didasarkan pada model tata letak flexbox web.

Stack

Alih-alih berorientasi linier (baik horizontal atau vertikal), Stackwidget memungkinkan Anda menempatkan widget di atas satu sama lain dalam urutan cat. Anda kemudian dapat menggunakan Positionedwidget pada anak-anak a Stackuntuk memposisikannya relatif terhadap tepi atas, kanan, bawah, atau kiri tumpukan. Tumpukan didasarkan pada model tata letak pemosisian absolut web.

Container

The Containerwidget memungkinkan Anda membuat elemen visual persegi panjang. Wadah dapat didekorasi dengan BoxDecoration, seperti latar belakang, batas, atau bayangan. A Containerjuga dapat memiliki margin, padding, dan batasan yang diterapkan

pada ukurannya. Selain itu, a Container dapat ditransformasikan dalam ruang tiga dimensi menggunakan matriks.

2. Menggunakan Komponen Material

Flutter menyediakan sejumlah widget yang membantu Anda membangun aplikasi yang mengikuti Desain Material. Aplikasi Material dimulai dengan *MaterialApp* widget, yang membuat sejumlah widget berguna di akar aplikasi Anda, termasuk *Navigator*, yang mengelola tumpukan widget yang diidentifikasi oleh string, juga dikenal sebagai "route". The *Navigator* memungkinkan Anda transisi lancar antara layar aplikasi Anda. Menggunakan *MaterialApp* widget sepenuhnya opsional tetapi praktik yang baik.

Sekarang kode telah beralih dari *MyApp* dan *MyScaffold* ke *App* dan *Scaffold* widget, dan dari *material.dart*, aplikasi mulai terlihat sedikit lebih Material. Misalnya, bilah aplikasi memiliki bayangan dan teks judul mewarisi gaya yang benar secara otomatis. Tombol aksi mengambang juga ditambahkan.

Perhatikan bahwa widget diteruskan sebagai argumen ke widget lain. The *Scaffold* widget mengambil sejumlah widget yang berbeda sebagai argumen bernama, yang masing-masing ditempatkan di *Scaffold* tata letak di tempat yang tepat. Demikian pula, *AppBar* widget memungkinkan Anda lulus dalam widget untuk *leading* widget, dan *actions* dari *title* widget. Pola ini berulang di seluruh kerangka kerja dan merupakan sesuatu yang mungkin Anda pertimbangkan saat mendesain widget Anda sendiri.

3. Keys

Gunakan tombol untuk mengontrol widget mana yang cocok dengan kerangka kerja dengan widget lain saat widget dibuat ulang. Secara default, framework mencocokkan widget di build saat ini dan sebelumnya sesuai dengan widget *runtimeType* dan urutan kemunculannya. Dengan kunci, kerangka mensyaratkan bahwa dua widget memiliki sama *key* serta sama *runtimeType*.

Tanpa kunci, entri pertama di build saat ini akan selalu disinkronkan dengan entri pertama di build sebelumnya, meskipun, secara semantik, entri pertama dalam daftar baru saja di-scroll dari layar dan tidak lagi terlihat di viewport.

Dengan menetapkan setiap entri dalam daftar sebagai kunci "semantik", daftar tak terbatas dapat lebih efisien karena kerangka menyinkronkan entri dengan kunci semantik yang cocok dan oleh karena itu tampilan visual yang serupa (atau identik). Selain itu, menyinkronkan entri secara semantik berarti bahwa status yang dipertahankan dalam widget anak stateful tetap melekat pada entri semantik yang sama daripada entri dalam posisi numerik yang sama di viewport.

Konstruktor

Kunci (Nilai string)

Bangun `ValueKey<String>` dengan String yang diberikan . [...]

konstan

pabrik

Kunci.kosong ()

Konstruktor default, digunakan oleh subclass. [...]

konstan

Properti

hashCode → int

Kode hash untuk objek ini. [...]

hanya-baca, diwariskan

runtimeType → Ketik

Sebuah representasi dari jenis runtime objek.

hanya-baca, diwariskan

Metode

noSuchMethod (Doa doa) → dinamis →

Dipanggil ketika metode atau properti yang tidak ada diakses. [...]

diwariskan

toString () → String

Representasi string dari objek ini. [...]

diwariskan

Operator

operator == (Obyek lain) → bool

Operator kesetaraan. [...]

4. Global keys

Gunakan kunci global untuk mengidentifikasi widget anak secara unik. Kunci global harus unik secara global di seluruh hierarki widget, tidak seperti kunci lokal yang hanya perlu unik di antara saudara kandung. Karena mereka unik secara global, kunci global dapat digunakan untuk mengambil status yang terkait dengan widget.

GlobalKey<T extends State<StatefulWidget>> class

Kunci yang unik di seluruh aplikasi.

Kunci global secara unik mengidentifikasi elemen. Kunci global menyediakan akses ke objek lain yang terkait dengan elemen tersebut, seperti BuildContext. Untuk StatefulWidget, kunci global juga menyediakan akses ke State.

Widget yang memiliki kunci global membuat ulang subpohonnya saat dipindahkan dari satu lokasi di pohon ke lokasi lain di pohon. Untuk me-reparent subpohonnya, widget harus tiba di lokasi barunya di pohon dalam bingkai animasi yang sama dengan tempat widget itu dihapus dari lokasi lamanya di pohon.

Reparenting sebuah Elemen menggunakan kunci global relatif mahal, karena operasi ini akan memicu panggilan ke `State.deactivate` pada `State` terkait dan semua turunannya; kemudian paksa semua widget yang bergantung pada `InheritedWidget` untuk dibangun kembali.

Jika Anda tidak memerlukan salah satu fitur yang tercantum di atas, pertimbangkan untuk menggunakan `Kunci`, `ValueKey`, `ObjectKey`, atau `UniqueKey`.

5. Menggunakan `Navigator.push`

Untuk beralih ke rute baru, gunakan `Navigator.push()` metode. The `push()`Metode menambahkan Route ke tumpukan rute dikelola oleh `Navigator`. Dari mana Route asalnya? Anda dapat membuat sendiri, atau menggunakan `MaterialPageRoute`, yang berguna karena transisi ke rute baru menggunakan animasi khusus platform.

6. Menggunakan `Navigator.pop()`

Bagaimana Anda menutup rute kedua dan kembali ke yang pertama? Dengan menggunakan `Navigator.pop()`metode. The `pop()`Metode menghilangkan arus Route dari tumpukan rute dikelola oleh `Navigator`.

Untuk menerapkan kembali ke rute asli, perbarui `onPressed()` panggilan balik di `SecondRouteWidget`:

7. Menggunakan Parse JSON di latar belakang

1. Tambahkan http paket

Pertama, tambahkan `http` paket ke proyek Anda. The `http` paket membuatnya lebih mudah untuk melakukan permintaan jaringan, seperti mengambil data dari endpoint JSON.

```
dependencies:  
  http: <latest_version>
```

2. Buat permintaan jaringan

Contoh ini mencakup cara mengambil dokumen JSON besar yang berisi daftar 5000 objek foto dari JSONPlaceholder REST API, menggunakan `http.get()`metode.

```
Masa Depan < http . Respon > fetchPhotos ( http . Klien klien ) async { klien kembali  
  . dapatkan ( Uri . parse ( 'https://jsonplaceholder.typicode.com/photos' )); }
```

3. Parsing dan ubah JSON menjadi daftar foto

Buat `Photo` kelas

Pertama, buat `Photo` kelas yang berisi data tentang foto. Sertakan `fromJson()`metode pabrik untuk memudahkan pembuatan `Photo` awal dengan objek JSON.

Ubah tanggapan menjadi daftar foto

Sekarang, gunakan instruksi berikut untuk memperbarui `fetchPhotos()` fungsi sehingga mengembalikan `Future<List<Photo>>`:

1. Buat `parsePhotos()` fungsi yang mengubah badan respons menjadi `List<Photo>`.
2. Gunakan `parsePhotos()` fungsi dalam `fetchPhotos()` fungsi.

4. Pindahkan pekerjaan ini ke isolate terpisah

Jika Anda menjalankan `fetchPhotos()` fungsi pada perangkat yang lebih lambat, Anda mungkin melihat aplikasi berhenti sejenak saat mem-parsing dan mengonversi JSON. Ini jank, dan Anda ingin menyingkirkannya.

Anda dapat menghapus jank dengan memindahkan penguraian dan konversi ke isolat latar belakang menggunakan `compute()` fungsi yang disediakan oleh Flutter. The `compute()` Fungsi berjalan fungsi mahal dalam isolat latar belakang dan mengembalikan hasilnya. Dalam hal ini, jalankan `parsePhotos()` fungsi di latar belakang.

Catatan tentang bekerja dengan isolate

Isolat berkomunikasi dengan menyampaikan pesan bolak-balik. Pesan ini dapat berupa nilai primitif, seperti `null`, `num`, `bool`, `double`, atau `String`, atau objek sederhana seperti `List<Photo>` dalam contoh ini.

Anda mungkin mengalami kesalahan jika mencoba melewati objek yang lebih kompleks, seperti `a Future` atau `http.Response` antara isolat.

8. Update data over the internet

1. Add the http package

To install the [http](#) package, add it to the dependencies section of the `pubspec.yaml` file. You can find the latest version of the [http package](#) on pub.dev.

dependencies:

```
http: <latest_version>
```

Import the [http](#) package.

```
import 'package:http/http.dart' as http;
```

2. Updating data over the internet using the http package

This recipe covers how to update an album title to the [JSONPlaceholder](#) using the `http.put()` method.

The `http.put()` method returns a [Future](#) that contains a [Response](#).

- [Future](#) is a core Dart class for working with async operations. A [Future](#) object represents a potential value or error that will be available at some time in the future.
- The [http.Response](#) class contains the data received from a successful http call.
- The `updateAlbum()` method takes an argument, `title`, which is sent to the server to update the [Album](#).

3. Convert the http.Response to a custom Dart object

While it's easy to make a network request, working with a raw `Future<http.Response>` isn't very convenient. To make your life easier, convert the [http.Response](#) into a Dart object.

Create an Album class

First, create an `Album` class that contains the data from the network request. It includes a factory constructor that creates an `Album` from JSON.

Converting JSON by hand is only one option. For more information, see the full article on [JSON and serialization](#).

4. Get the data from the internet

Get the data from internet before you can update it. For a complete example, see the [Fetch data](#) recipe.

Ideally, you will use this method to set `_futureAlbum` during `initState` to fetch the data from the internet.

5. Update the exiting title from user input

Create a `TextField` to enter a title and a `ElevatedButton` to update the data on server. Also define a `TextEditingController` to read the user input from a `TextField`.

When the `ElevatedButton` is pressed, the `_futureAlbum` is set to the value returned by `updateAlbum()` method.

On pressing the **Update Data** button, a network request sends the data in the `TextField` to the server as a `POST` request. The `_futureAlbum` variable is used in the next step.

Display the response on screen

To display the data on screen, use the `FutureBuilder` widget.

The `FutureBuilder` widget comes with Flutter and makes it easy to work with async data sources. You must provide two parameters:

1. The `Future` you want to work with. In this case, the future returned from the `updateAlbum()` function.
2. A `builder` function that tells Flutter what to render, depending on the state of the `Future`: loading, success, or error.

Note that `snapshot.hasData` only returns `true` when the snapshot contains a non-null data value. This is why the `updateAlbum` function should throw an exception even in the case of a “404 Not Found” server response.

If `updateAlbum` returns `null` then `CircularProgressIndicator` will display indefinitely.

BAB III

PEMBAHASAN

Algoritma

Start Aplikasi

Output Tampilan Awal Aplikasi

Klik tambahkan barang

Masukkan nama barang, foto barang, dan harga barang kemudian tambahkan

Output masuk ke tampilan awal mengenai barang baru yang barusaja ditambahkan

Klik foto barang di tampilan awal untuk melihat detail informasi barang

Program Selesai

Source Code

Flutter

1.main.dart

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

```
import 'screens/homepage.dart';
```

```
void main() {  
  runApp(MyApp());  
}
```

```
class MyApp extends StatelessWidget {
```

```
  const MyApp({  
    Key key,  
  }) : super(key: key);
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(  
      title: 'TOKO',  
      home: HomePage(),  
    );
```

```
}  
}
```

2. p.detail.dart

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

```
class ProductDetail extends StatelessWidget {  
  final Map product;
```

```
  ProductDetail({ @required this.product});
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Product Detail"),  
      ),  
      body: Column(  
        children: [  
          Container(  
            child: Image.network(product['image_url']),  
          ),  
          Padding(  
            padding: const EdgeInsets.all(8.0),  
            child: Row(  
              mainAxisAlignment: MainAxisAlignment.spaceBetween,  
              children: [  
                Text(  
                  product['price'],  
                  style: TextStyle(fontSize: 22),  
                ),  
                Row(  
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,  
                  children: [  
                    Text(  
                      product['description'],  
                      style: TextStyle(fontSize: 16),  
                    ),  
                    Text(  
                      product['category'],  
                      style: TextStyle(fontSize: 16),  
                    ),  
                  ],  
                ),  
              ],  
            ),  
          ],  
        ),  
      ),  
    );  
  }
```

```

        children: [Icon(Icons.edit), Icon(Icons.delete)],
      ),
    ],
  ),
),
Padding(
  padding: EdgeInsets.fromLTRB(0, 0, 0, 88),
  child: Text(product['description']),
),
Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    Container(
      alignment: Alignment.center,
      child: Text(
        "Beli",
        style: TextStyle(fontWeight: FontWeight.w300, fontSize: 20.0),
      ),
      height: 60,
      width: 120,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(10),
        color: Colors.blue),
    ),
  ],
)
],
),
);
}
}

```

3. homepage.dart

```
import 'dart:convert';
```

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

```
import 'package:http/http.dart' as http;
```

```
import 'product.dart';
```

```
import 'p.detail.dart';
```

```
class HomePage extends StatelessWidget {
```

```
  const HomePage({
```

```
    Key key,
```

```
  }) : super(key: key);
```

```
  final String url = 'http://127.0.0.1:8000/api/products';
```

```
  Future getProducts() async {
```

```
    var response = await http.get(Uri.parse(url));
```

```
    print(json.decode(response.body));
```

```
    return json.decode(response.body);
```

```
  }
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
  return Scaffold(
```

```
    floatingActionButton: FloatingActionButton(
```

```
      onPressed: () {
```

```
        Navigator.push(
```

```
          context, MaterialPageRoute(builder: (context) => AddProduct()),
```

```
        ),
```

```
        child: Icon(Icons.add),
```

```
      ),
```

```

appBar: AppBar(
  title: Text(' TOKO ATK '),
),
body: FutureBuilder(
  future: getProducts(),
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return ListView.builder(
        itemCount: snapshot.data['data'].length,
        itemBuilder: (context, index) {
          return Container(
            height: 180,
            child: Card(
              elevation: 5,
              child: Row(
                children: [
                  GestureDetector( //gambar
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => ProductDetail(
                            product: snapshot.data['data']
                              [index],
                          )),
                    ),
                  Container(
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(15.0),
                    ),
                    padding: EdgeInsets.all(5),
                    height: 120,

```

```
width: 120,
child: Image.network(
  snapshot.data['data'][index]['image_url'],
  fit: BoxFit.cover,
),
),
),
Expanded(
  child: Container(
    padding: EdgeInsets.all(10.0),
    child: Column(
      mainAxisAlignment:
        MainAxisAlignment.spaceAround,
      children: [
        Align(
          alignment: Alignment.topLeft,
          child: Text(
            snapshot.data['data'][index]
              ['name'],
            style: TextStyle(
              fontSize: 20.0,
              fontWeight: FontWeight.bold),
          ),
        ),
        Align(
          alignment: Alignment.topLeft,
          child: Text(
            snapshot.data['data'][index]
              ['description'],
          ),
        ),
      ],
    ),
  ),
  Row(
```

```

        mainAxisAlignment:
            MainAxisAlignment.spaceBetween,
        children: [
            Icon(Icons.star_half_rounded),
            Icon(Icons.message),
            Text(
                snapshot.data['data'][index]
                    ['price'],
            ),
        ],
    )
],
)))

],
),
),
);
});
} else {
    return Text('Data error');
}
}));
}
}

```

4. product.dart

```

import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'product.dart';
import 'p.detail.dart';

```



```

class HomePage extends StatelessWidget {
  const HomePage({
    Key key,
  }) : super(key: key);

  final String url = 'http://127.0.0.1:8000/api/products';

  Future getProducts() async {
    var response = await http.get(Uri.parse(url));
    print(json.decode(response.body));
    return json.decode(response.body);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context, MaterialPageRoute(builder: (context) => AddProduct()));
        },
        child: Icon(Icons.add),
      ),
      appBar: AppBar(
        title: Text(' TOKO ATK '),
      ),
      body: FutureBuilder(
        future: getProducts(),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            return ListView.builder(

```

```

itemCount: snapshot.data['data'].length,
itemBuilder: (context, index) {
  return Container(
    height: 180,
    child: Card(
      elevation: 5,
      child: Row(
        children: [
          GestureDetector( //gambar
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ProductDetail(
                    product: snapshot.data['data']
                      [index],
                  )),
            ),
          Container(
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(15.0),
            ),
            padding: EdgeInsets.all(5),
            height: 120,
            width: 120,
            child: Image.network(
              snapshot.data['data'][index]['image_url'],
              fit: BoxFit.cover,
            ),
          ),
        ],
      ),
    ),
  );
}
Expanded(

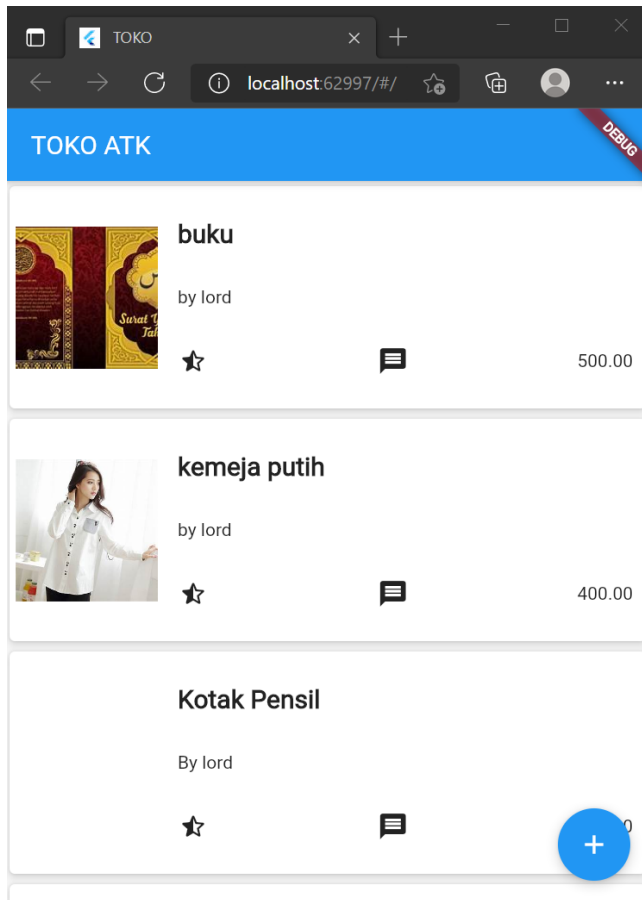
```

```
child: Container(  
  padding: EdgeInsets.all(10.0),  
  child: Column(  
    mainAxisAlignment:  
      MainAxisAlignment.spaceAround,  
    children: [  
      Align(  
        alignment: Alignment.topLeft,  
        child: Text(  
          snapshot.data['data'][index]  
            ['name'],  
          style: TextStyle(  
            fontSize: 20.0,  
            fontWeight: FontWeight.bold),  
          ),  
        ),  
      Align(  
        alignment: Alignment.topLeft,  
        child: Text(  
          snapshot.data['data'][index]  
            ['description'],  
          ),  
        ),  
      Row(  
        mainAxisAlignment:  
          MainAxisAlignment.spaceBetween,  
        children: [  
          Icon(Icons.star_half_rounded),  
          Icon(Icons.message),  
          Text(  
            snapshot.data['data'][index]  
              ['price'],
```

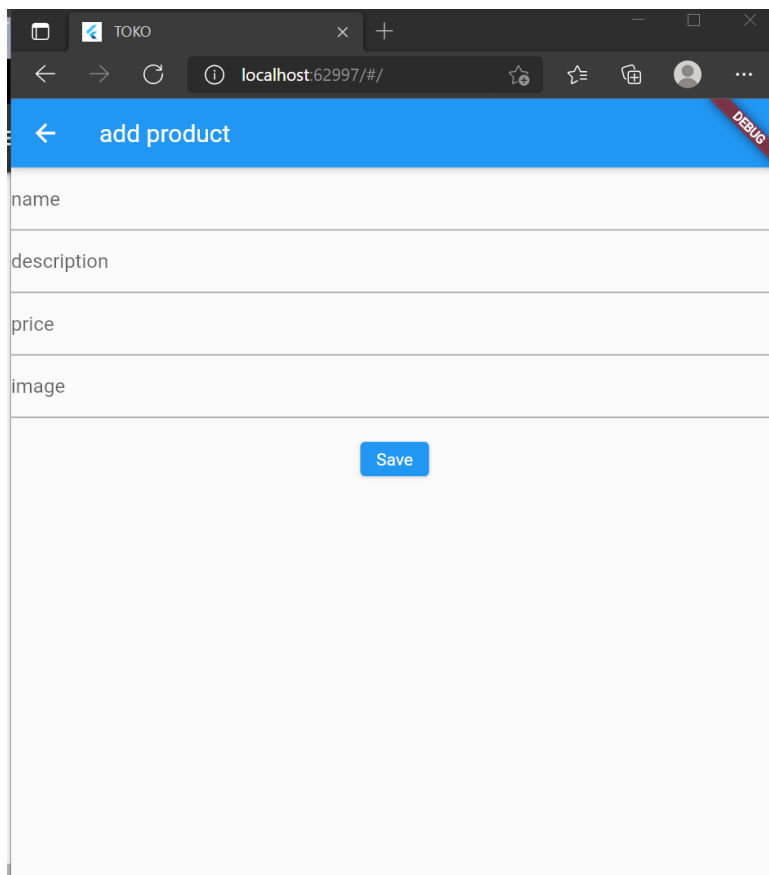
```
        ),  
        ],  
    )  
    ],  
    )))  
    ],  
    ),  
    ),  
    );  
    });  
    } else {  
        return Text('Data error');  
    }  
    }));  
    }  
    }
```

Analisis Program

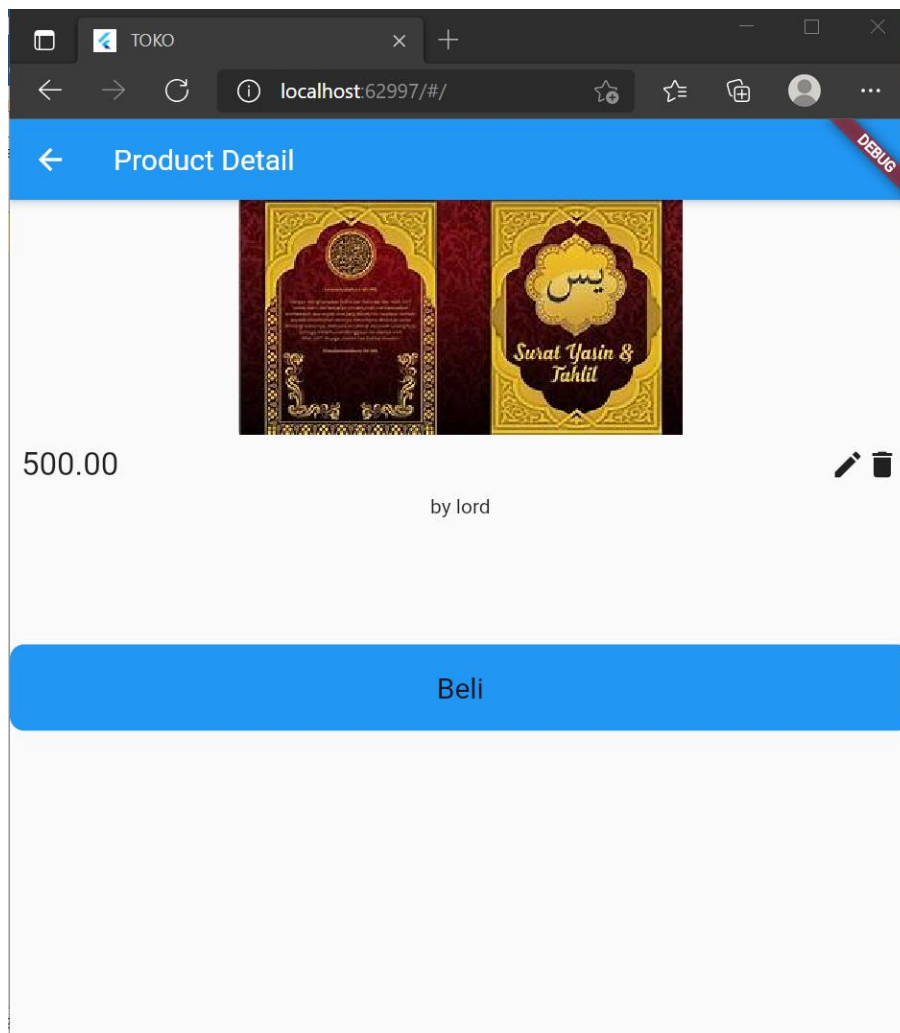
1. Tampilan awal ketika di run



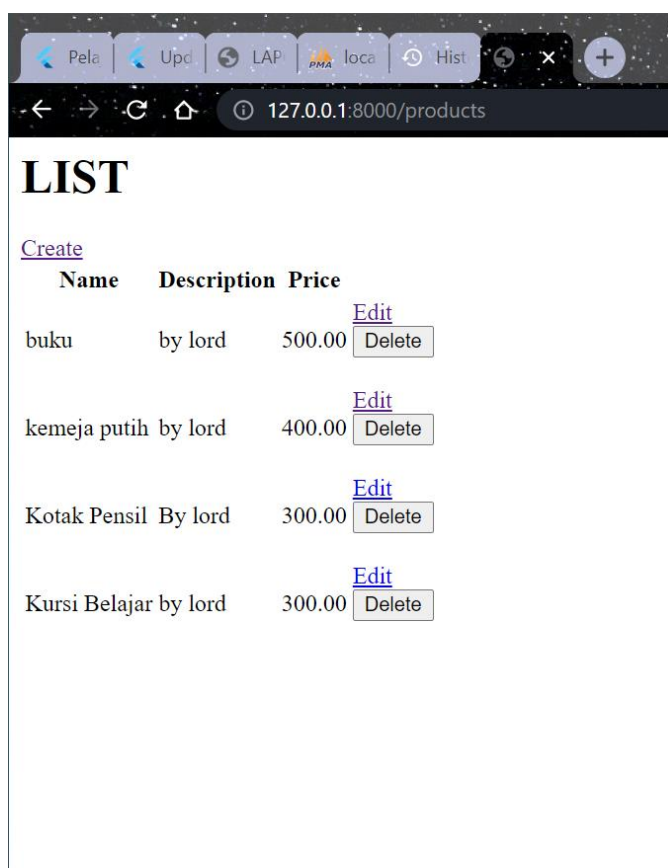
2. Tampilan ketika add product ketika kita klik icon “+” pada tampilan utama



3. Tampilan ketika klik image pada tampilan awal akan menampilkan Product Detail



4. Tampilan Admin Server & disini bisa langsung melakukan delete pada barang



5. Tampilan Create pada Admin Server



The screenshot shows a mobile browser interface with a dark theme. The address bar at the top displays the URL `127.0.0.1:8000/products/create`. Below the address bar, the page title is **Create Product**. The form contains four input fields: **Name :**, **Description :**, **Price :**, and **Image URL :**. A **Save** button is located at the bottom left of the form area.

Create Product

Name :

Description :

Price :

Image URL :

6. Tampilan Edit pada Admin Server



Edit Product

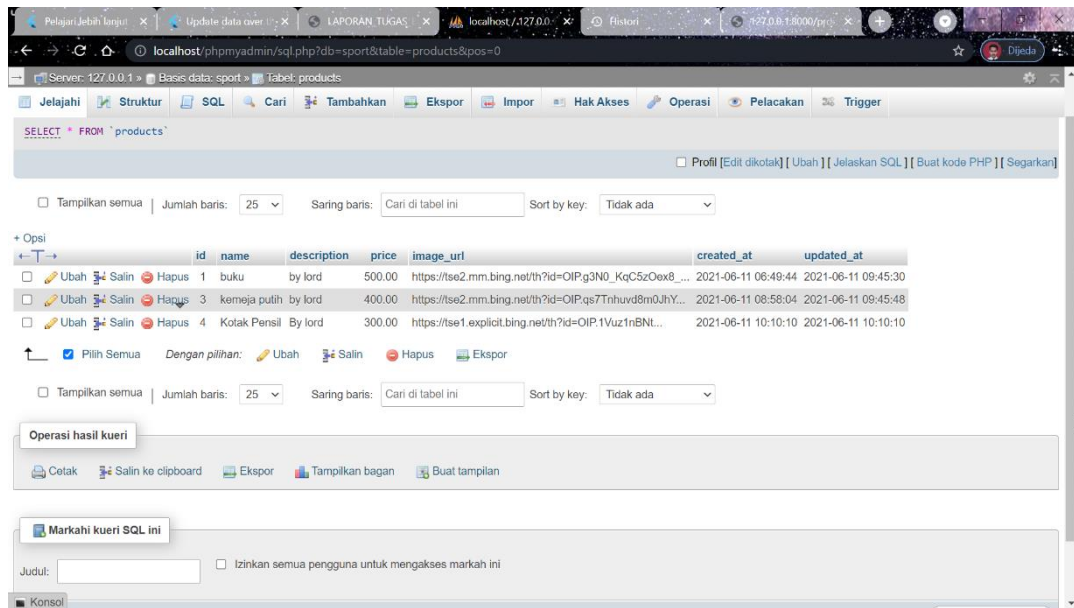
Name :

Description :

Price :

Image URL :

7. Tampilan Database Create, Edit, dan Delete di php my admin



The screenshot displays the phpMyAdmin interface for a database named 'sport'. The 'products' table is selected, and the SQL query editor shows the query: `SELECT * FROM 'products'`. The table view shows 4 records with columns: id, name, description, price, image_uri, created_at, and updated_at.

	id	name	description	price	image_uri	created_at	updated_at
<input type="checkbox"/>	1	buku	by lord	500.00	https://tse2.mm.bing.net/th?id=OIP.g3N0_KqC5zOex8...	2021-06-11 06:49:44	2021-06-11 09:45:30
<input type="checkbox"/>	3	kemeja putih	by lord	400.00	https://tse2.mm.bing.net/th?id=OIP.qs7Tnhuvd8m0JhY...	2021-06-11 08:58:04	2021-06-11 09:45:48
<input type="checkbox"/>	4	Kotak Pensil	By lord	300.00	https://tse1.explicit.bing.net/th?id=OIP.1Vuz1nBNL...	2021-06-11 10:10:10	2021-06-11 10:10:10

Below the table, there are options for 'Operasi hasil kueri' (Print, Salin ke clipboard, Ekspor, Tampilkan bagan, Buat tampilan) and a section for 'Markahi kueri SQL ini' (Judul: [input], Izinkan semua pengguna untuk mengakses markah ini).

BAB IV

PENUTUP

Kesimpulan

Dengan adanya program yang telah dibuat oleh penulis, maka dapat diambil kesimpulan, yaitu;

1. Kita dapat membuat sebuah Aplikasi Toko Online dengan sangat mudah menggunakan Flutter. Interface yang terdapat di Flutter sangat bagus dan mudah untuk dipelajari.
2. Kita lebih mudah mempromosikan barang dagangan kita dengan hanya mengupload gambar, nama barang, dan harga barang.
3. Kita bisa hemat tempat karena tidak harus membangun sebuah Toko di kehidupan nyata, kita hanya harus menampung barang yang ingin kita jual. Dengan ini kita dapat menghemat pengeluaran sehingga biaya yang keluar tidak terlalu besar.

Saran

Program yang dibuat oleh penulis masih dapat dikembangkan menjadi lebih besar, karena penulis menyadari bahwa program yang dibuat masih terdapat banyak kekurangan baik dari keindahan interface yang masih standart, fitur yang kurang lengkap, dan sebagainya.

DAFTAR PUSTAKA

<https://flutter.dev/docs/get-started/codelab>

<https://flutter.dev/docs/cookbook/networking/update-data#4-get-the-data-from-the-internet>

<https://flutter.dev/docs/cookbook/navigation/navigation-basics>

<https://flutter.dev/docs/cookbook/testing/widget/introduction>

<https://flutter.dev/docs/cookbook/networking/background-parsing>

<https://flutter.dev/docs/deployment/android>