PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

ANTARMUKA PENGGUNA LANJUTAN – MODUL 5

DISUSUN OLEH :

MUHAMMAD FARIZ NUR HIDAYAT

2211104069

SE06-2

TELKOM UNIVERSITY PURWOKERTO

S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

2024

ANTARMUKA PENGGUNA LANJUTAN

1. ListView.builder

Widget ListView jenis ini cocok digunakan ketika memiliki data list yang lebih besar. ListView.builder membutuhkan itemBuilder dan itemCount. Parameter itemBuilder merupakan fungsi yang mengembalikan widget untuk ditampilkan. Sedangkan itemCount kita isi dengan jumlah seluruh item yang ingin ditampilkan.

2. ListView.Separated

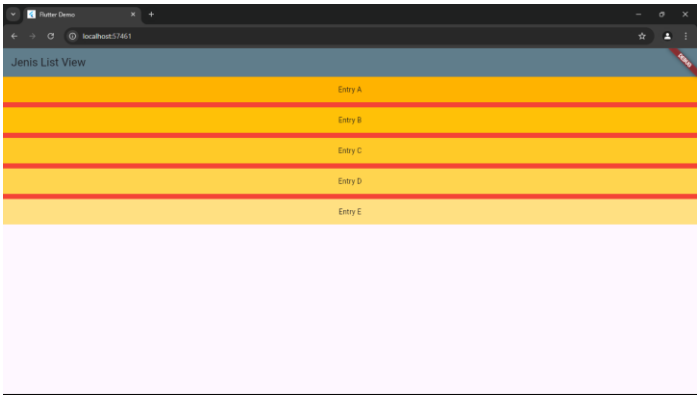
ListView jenis ini akan menampilkan daftar item yang dipisahkan dengan separator. Penggunaan ListView.separated mirip dengan builder, yang membedakan adalah terdapat satu parameter tambahan wajib yaitu separatorBuilder yang mengembalikan Widget yang akan berperan sebagai separator.

Bukti Praktikum

Source Code :

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

class JenisListView extends StatelessWidget {
  const JenisListView({super.key});

  @override
  Widget build(BuildContext context) {
    final List<String> entries = <String>['A', 'B', 'C', 'D', 'E'];
    final List<int> colorCodes = <int>[600, 500, 400, 300, 200, 100];

    return Scaffold(
        appBar: AppBar(
          title: const Text('Jenis List View'),
          backgroundColor: Colors.blueGrey,
        ),
        body: ListView.separated(
          itemBuilder: (BuildContext context, int index) {
            return Container(
              height: 50,
              color: Colors.amber[colorCodes[index]],
              child: Center(
                child: Text("Entry ${entries[index]}"),
              ),
            );
          },
          itemCount: entries.length,
          separatorBuilder: (BuildContext context, int index) {
            return Container(
              height: 10,
              color: Colors.red,
            );
          },
        ));
  }
}

```

Output :



Deskripsi Program :

Program tersebut adalah sebuah aplikasi Flutter sederhana yang menampilkan daftar menggunakan `ListView.separated` dengan nama tampilan "Jenis List View." Berikut penjelasan lebih detail mengenai komponen-komponen dalam program:

1. Scaffold
Program menggunakan `Scaffold` sebagai kerangka utama aplikasi yang menyediakan struktur standar seperti app bar dan body.

2. AppBar
Pada bagian atas aplikasi terdapat `AppBar` dengan judul **'Jenis List View'** dan latar belakang berwarna `Colors.blueGrey`.

3. ListView.separated
Komponen utama yang ditampilkan adalah sebuah `ListView` dengan pemisah antar item. `ListView.separated` menampilkan daftar item yang dipisahkan oleh widget separator yang ditentukan.

- `itemBuilder`: Digunakan untuk membangun setiap item dalam daftar. Di sini, program membangun container setinggi 50 dengan warna `Colors.amber` dan intensitas warna yang berbeda-beda berdasarkan nilai yang ada dalam list `colorCodes`. Setiap item menampilkan teks "Entry X", di mana `X` adalah huruf dari list `entries` (`A`, `B`, `C`, dll.).

- `separatorBuilder`: Digunakan untuk membangun pemisah antar item. Pemisah berupa container setinggi 10 dengan warna `Colors.red`.

- `itemCount`: Menentukan jumlah total item yang akan ditampilkan, diambil dari panjang list `entries` yang berjumlah 5.

4. Daftar (List)
- `entries`: List berisi elemen-elemen berupa string yang menampung huruf dari 'A' hingga 'E' yang akan digunakan sebagai teks dalam daftar.
- `colorCodes`: List berisi nilai integer yang mengindikasikan kode warna untuk item yang akan ditampilkan dengan berbagai intensitas warna amber.

3. Flexible dan Expanded

a) Flexible

Flexible digunakan ketika Anda ingin memberikan ruang fleksibel kepada widget di dalam kolom atau baris. Dengan Flexible, widget dapat mengambil ruang yang tersisa di dalam layout, tetapi tetap memiliki batasmaksimal yang disesuaikan dengan kebutuhan ruangnya.

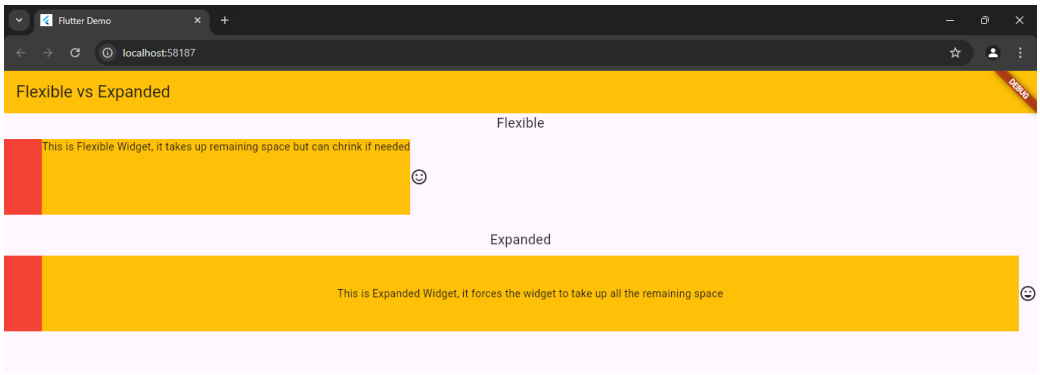
b) Expanded

Expanded adalah turunan dari Flexible yang secara otomatis mengisi semua ruang yang tersisa di dalam kolom atau baris. Ketika Anda menggunakan Expanded, widget tersebut akan mengambil ruangsebanyak mungkin tanpa memperhatikan kebutuhan ruang minimum.

Bukti Praktikum :
Source Code :

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

class FlexibleExpandedScreen extends StatelessWidget {
  const FlexibleExpandedScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Flexible vs Expanded"),
        backgroundColor: Colors.amber,
      ),
      body: Column(
        children: [
          Text(
            "Flexible",
            style: TextStyle(fontSize: 18),
          ),
          SizedBox(
            height: 8,
          ),
          Row(
            children: [
              Container(
                width: 50,
                height: 100,
                color: Colors.red,
              ),
              Flexible(
                child: Container(
                  height: 100,
                  color: Colors.amber,
                  child: Text(

"This is Flexible kidget, it takes up remaining space but can chrink if needed"
),
                ),
              ),
              Icon(Icons.sentiment satisfied alt rounded)
            ],
          ),
          SizedBox(
            height: 20,
          ),
          // Expanded Section
          Text(
            "Expanded",
            style: textStyle(fontSize: 18),
          ),
          SizedBox(
            height: 8,
          ),
          Row(
            children: [
              Container(
                width: 50,
                height: 100,
                color: Colors.red,
              ),
              Expanded(
                child: Container(
                  height: 100,
                  color: Colors.amber,
                  padding: EdgeInsets.all(8),
                  child: Center(
                    child: Text(

"This is Expanded Widget, it forces the widget to take up all the remaining spac
e"
,
                      textAlign: textAlign.center,
                    ),
                  ),
                ),
              ),
              Icon(Icons.sentiment very satisfied rounded)
            ],
          ),
        ],
      ),
    );
  }
}
```

Output :



Deskripsi Program :

Program ini adalah sebuah aplikasi Flutter sederhana yang menunjukkan perbedaan antara penggunaan widget `Flexible` dan `Expanded` di dalam sebuah `Row`. Aplikasi ini menggunakan tata letak berbasis kolom (Column) dan baris (Row) untuk mendemonstrasikan bagaimana kedua widget ini berperilaku dalam membagi ruang di dalam tata letak horizontal.

Penjelasan Program:

Bagian 1: Flexible

- Judul: Teks berjudul "Flexible" dengan ukuran font 18 diikuti oleh sedikit jarak vertikal (`SizedBox` dengan tinggi 8).
- Row: Di dalam `Row`, terdapat tiga elemen:
  1. Container pertama berwarna merah dengan lebar 50 dan tinggi 100.
  2. Flexible: Mengandung sebuah `Container` berwarna amber dengan tinggi 100 dan teks yang menjelaskan tentang sifat `Flexible`. Teks ini menyatakan bahwa widget ini mengambil ruang yang tersisa namun bisa mengecil jika dibutuhkan (contoh, ketika ruang yang tersedia tidak cukup).
  3. Icon: Ikon `Icons.sentiment_satisfied_alt_rounded` yang ditempatkan di ujung `Row`.

Bagian 2: Expanded

- Judul: Teks berjudul "Expanded" dengan ukuran font 18 diikuti oleh jarak vertikal (`SizedBox` dengan tinggi 8).
- Row: Sama seperti di bagian `Flexible`, namun widget yang digunakan adalah `Expanded`. Di dalam `Row`, terdapat tiga elemen:
  1. Container pertama berwarna merah dengan lebar 50 dan tinggi 100.
  2. Expanded: Mengandung sebuah `Container` berwarna amber dengan tinggi 100 dan teks yang menyatakan bahwa widget ini memaksa dirinya untuk mengambil semua ruang yang tersisa dalam baris, tanpa menyusut.
  3. Icon: Ikon `Icons.sentiment_very_satisfied_rounded` yang ditempatkan di ujung `Row`.

Perbedaan Antara Flexible dan Expanded:

- Flexible: Menyesuaikan ukurannya dengan ruang yang tersisa di dalam `Row` tetapi bisa menyusut jika diperlukan. Dalam hal ini, ia akan membatasi dirinya jika ruang yang tersedia terbatas.

- Expanded: Secara paksa mengambil semua ruang yang tersisa dalam `Row` tanpa menyusut.

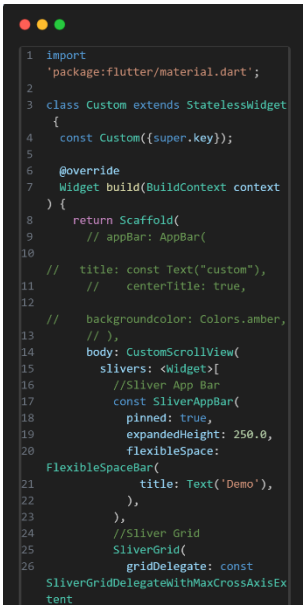
Program ini berfungsi sebagai ilustrasi sederhana dari cara kerja `Flexible` dan `Expanded` di Flutter.

4. CustomScrollView

Widget ini memungkinkan membuat efek pada list, grid, maupun header yang lebar. Misalnya, ketika ingin membuat scroll view yang berisi app bar yang lebar yang meliputi list dan grid secara bersamaan, maka bisa menggunakan 3 widget sliver, yaitu SliverAppBar, SliverList, dan SliverGrid.

Bukti Praktikum

Source Code:



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

class Custom extends StatelessWidget {
  const Custom({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // appBar: AppBar(

//   title: const Text("custom"),
      //    centerTitle: true,

//    backgroundcolor: Colors.amber,
      // ),
      body: CustomScrollView(
        slivers: <Widget>[
          //Sliver App Bar
          const SliverAppBar(
            pinned: true,
            expandedHeight: 250.0,
            flexibleSpace: FlexibleSpaceBar(
              title: Text('Demo'),
            ),
          ),
          //Sliver Grid
          SliverGrid(
            gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent
```

```dart
              maxCrossAxisExtent: 200.0,
              mainAxisSpacing: 10.0,
              crossAxisSpacing: 10.0,
              childAspectRatio: 4.0,
            ),
            delegate: SliverChildBuilderDelegate(
              (BuildContext context, int index) {
                return Container(
                  alignment: Alignment.center,
                  color: Colors.amber[100 * (index % 9)],
                  child: Column(
                    children: [
                      Icon(
                        Icons.access_alarm,
                        size: 15,
                      ),
                      Text('Grid Item $index'),
                    ],
                  ),
                );
              },
              childCount: 20,
            ),
          ),
          //Sliver Fixed List
          // SliverFixedExtentList(
          //   itemExtent: 50.0,

//   delegate: SliverChildBuilderDelegate(

//     (BuildContext context, int index) {
          //       return Container(

//         alignment: Alignment.center,

//         color: Colors.amber[100 * (index % 9)],

//         child: Text('List Item $index'),
          //       );
          //     },
          //   ),
          // ),
        ],
      ),
    );
  }
}
```

Output :

Bagian 1: SliverAppBar

- SliverAppBar adalah header yang muncul di bagian atas tampilan dan dapat di-scroll. SliverAppBar ini bersifat **pinned**, artinya akan tetap terlihat di bagian atas saat pengguna menggulir ke bawah.
- expandedHeight: Menetapkan tinggi saat AppBar dalam keadaan diperluas hingga 250 piksel.
- FlexibleSpaceBar: Menyediakan ruang fleksibel untuk judul "Demo" yang akan diubah ukurannya saat pengguna menggulir.

Bagian 2: SliverGrid

- SliverGrid menampilkan daftar item dalam tata letak grid. Grid ini memiliki beberapa pengaturan yang membuat tampilan lebih fleksibel:
    - **gridDelegate**: Menggunakan SliverGridDelegateWithMaxCrossAxisExtent untuk menentukan ukuran dan tata letak grid. Ini memungkinkan grid untuk memiliki ukuran maksimal pada sumbu horizontal sebesar 200 piksel, dengan jarak antar item sebesar 10 piksel di kedua sumbu (horizontal dan vertikal).
    - **childAspectRatio**: Menentukan rasio aspek dari anak-anak grid (4:1, lebih lebar daripada tinggi).
- **SliverChildBuilderDelegate**: Digunakan untuk membangun elemen grid secara dinamis. Ini memanfaatkan loop berdasarkan indeks item (index) untuk membangun item satu per satu.
    - Setiap elemen grid adalah sebuah Container dengan warna amber yang berubah sesuai dengan indeks (index % 9), berisi ikon kecil Icons.access_alarm dan teks Grid Item $index.
- **childCount**: Menentukan jumlah total item dalam grid, yaitu 20.

Bagian 3 (dikomentari): SliverFixedExtentList

- Bagian ini dikomentari, tetapi jika diaktifkan, akan menambahkan daftar statis yang menggunakan SliverFixedExtentList dengan tinggi setiap item ditentukan secara tetap, yaitu 50 piksel.

- **SliverChildBuilderDelegate**: Sama seperti di SliverGrid, tetapi untuk membangun item dalam daftar (bukan grid). Setiap item adalah Container berwarna amber dengan teks List Item $index.

Tampilan Akhir:

1. **SliverAppBar**: Header yang dapat di-scroll dengan judul "Demo". Saat pengguna menggulir ke atas, AppBar akan mengecil tetapi tetap terlihat di bagian atas layar.
2. **SliverGrid**: Menampilkan grid dengan 20 item, masing-masing dengan ikon alarm kecil dan teks yang dinamis, misalnya "Grid Item 0", "Grid Item 1", dan seterusnya.
3. **(Optional)** SliverFixedExtentList (dikomentari): Jika diaktifkan, menambahkan daftar item dengan tinggi tetap dan teks yang sesuai dengan indeks mereka.

Kesimpulan:

Program ini mendemonstrasikan penggunaan CustomScrollView dan berbagai jenis Sliver untuk membuat tata letak yang lebih dinamis dan responsif, seperti kombinasi grid dan daftar yang dapat di-scroll.

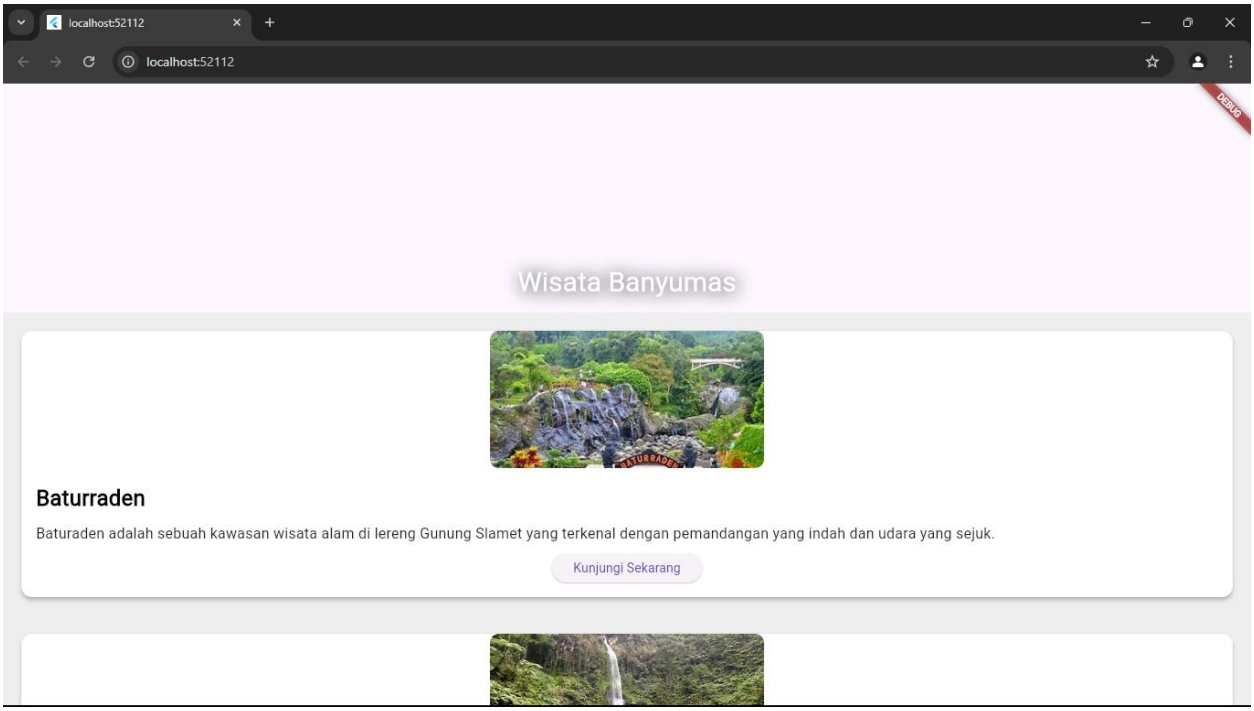
Tugas Mandiri (Unguided)

1. Modifikasi project Rekomendasi Wisata pada Tugas Unguided 04 modul Antarmuka Pengguna dengan mengimplementasikan widget CustomScrollView, SliverAppBar, dan SliverList untuk merekomendasikan beberapa tempat wisata yang ada di Banyumas disertai foto, nama wisata, dan deskripsi singkat! (buatlah se kreatif mungkin).

\

Source Code :

Output :



Deskripsi Program :

Program di atas adalah aplikasi Flutter yang menampilkan daftar rekomendasi tempat wisata di daerah Banyumas. Aplikasi ini menggunakan beberapa komponen UI seperti `CustomScrollView`, `SliverAppBar`, dan `SliverList` untuk menyajikan informasi wisata secara interaktif.

- SliverAppBar: Komponen ini berfungsi sebagai header yang berisi judul "Wisata Banyumas" dan gambar latar belakang yang dapat di-scroll. Saat pengguna menggulir halaman, `SliverAppBar` akan mengecil namun tetap terlihat di bagian atas.

- CustomScrollView: Komponen ini memungkinkan pengguna untuk menggulir halaman secara vertikal. Di dalamnya terdapat daftar tempat wisata yang ditampilkan menggunakan `SliverList`.

- SliverList: Digunakan untuk menampilkan daftar tempat wisata. Setiap item dalam daftar ini berisi gambar, nama, deskripsi, dan tombol aksi untuk mengunjungi tempat wisata. Gambar ditampilkan dengan ukuran kecil yang terpusat di layar, dan deskripsi tempat wisata menggunakan teks yang dirapikan agar mudah dibaca.

Program juga menangani pemuatan gambar dengan indikator loading (`CircularProgressIndicator`) ketika gambar sedang diunduh dari internet. Seluruh tampilan aplikasi diberikan latar belakang abu-abu muda, sedangkan teks dan elemen interaktif seperti tombol menggunakan warna yang lebih kontras untuk meningkatkan keterbacaan dan pengalaman pengguna.