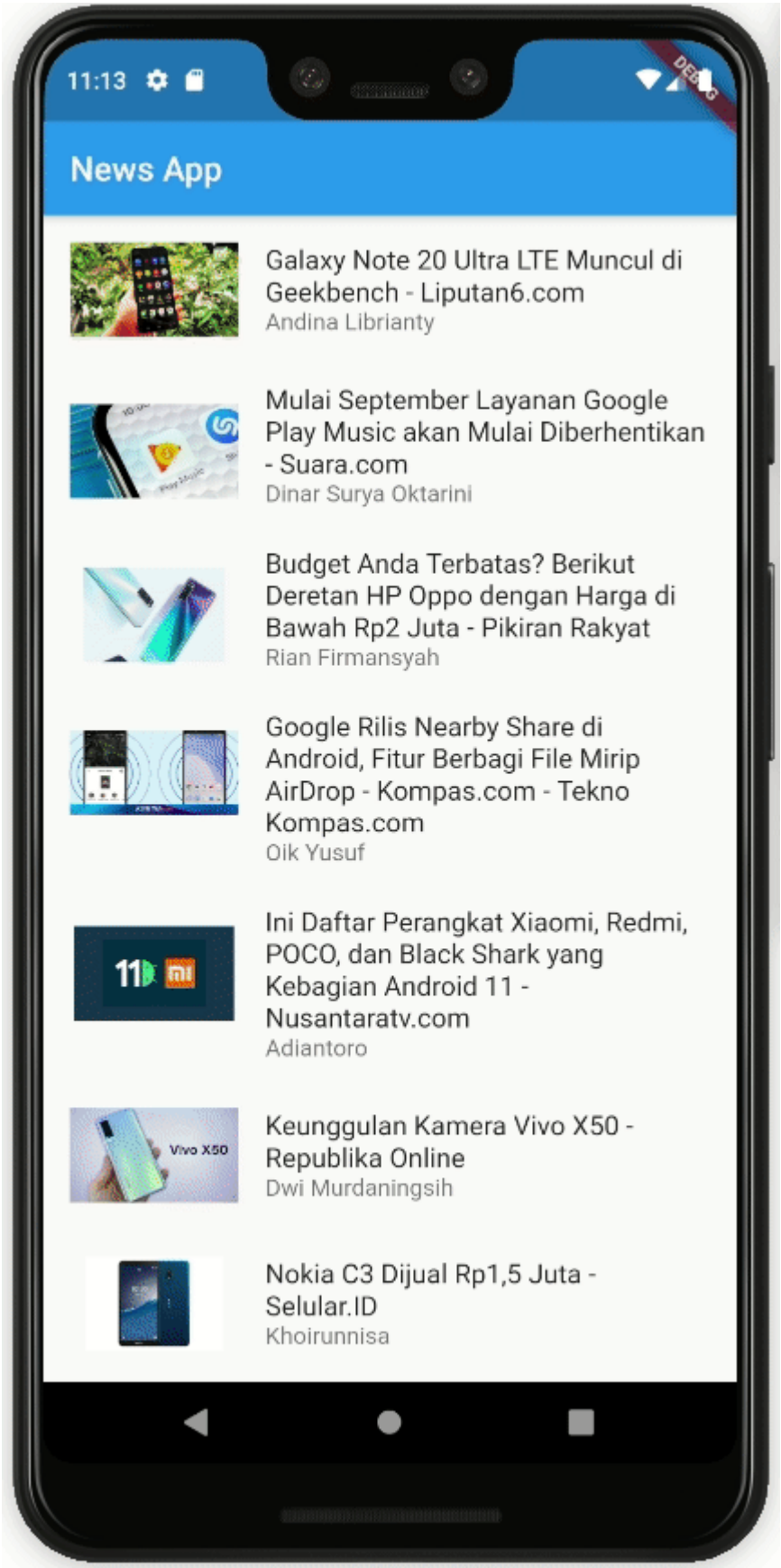
Latihan : News App

Setelah mempelajari tentang widget dan navigasi, sekarang kita akan memulai praktik mengembangkan aplikasi pertama kita di kelas ini. Aplikasi yang akan kita buat adalah sebuah aplikasi portal berita. Aplikasi ini akan menampilkan beberapa berita dari berbagai sumber. Beberapa poin yang akan dipelajari pada materi ini, antara lain:

- Mengambil data dari berkas JSON lalu ditampilkan sebagai *list*.
- Mengimplementasikan *named routes* untuk navigasi halaman.
- Menambahkan *packagewebview* untuk menampilkan konten web.

Hasil dari aplikasi yang dibuat akan jadi seperti ini:


11:13
DEBUG
News App
Galaxy Note 20 Ultra LTE Muncul di Geekbench - Liputan6.com
Andina Librianty
Mulai September Layanan Google Play Music akan Mulai Diberhentikan - Suara.com
Dinar Surya Oktarini
Budget Anda Terbatas? Berikut Deretan HP Oppo dengan Harga di Bawah Rp2 Juta - Pikiran Rakyat
Rian Firmansyah
Google Rilis Nearby Share di Android, Fitur Berbagi File Mirip AirDrop - Kompas.com - Tekno Kompas.com
Oik Yusuf
Ini Daftar Perangkat Xiaomi, Redmi, POCO, dan Black Shark yang Kebagian Android 11 - Nusantaratv.com
Adiantoro
Vivo X50
Keunggulan Kamera Vivo X50 - Republika Online
Dwi Murdaningsih
Nokia C3 Dijual Rp1,5 Juta - Selular.ID
Khoirunnisa

Note : Semua kode dalam kelas ini akan ditulis menggunakan fitur null safety dari Dart versi 2.12. Untuk menggunakannya, pastikan Anda sudah menginstal versi Flutter terbaru lalu mengubah versi minimal sdk yang digunakan pada berkas **pubspec.yaml**.

```
environment:
  sdk: '>=2.12.0 <3.0.0'
```

Cara lainnya adalah menggunakan perintah dart migrate melalui terminal. Perintah ini akan berguna jika Anda sudah memiliki sebuah project lalu ingin memigrasikan ke null safety. Selengkapnya mengenai migrasi null safety, dapat Anda baca pada dokumentasi null safety.

1. Mari buat project Flutter baru dan berikan nama misalnya **dicoding_news_app**.
2. Sebelum mulai menyusun *layout* aplikasi, kita akan menyiapkan data yang akan ditampilkan terlebih dulu. Saat ini kita masih akan menggunakan data lokal yang disimpan dalam bentuk json. Untuk berkas *json*-nya sendiri dapat Anda unduh pada tautan berikut.
3. Tambahkan berkas yang telah Anda unduh ke dalam *project*. Buat folder baru bernama ***assets*** pada direktori utama lalu letakkan berkas json pada folder tersebut.
4. Selanjutnya daftarkan aset json tersebut pada berkas **pubspec.yaml**.

   ```
   # To add assets to your application, add an assets section, like this:
   assets:
     - assets/articles.json
   ```

   Jangan lupa perhatikan indentasi karena merupakan hal yang penting dalam berkas yaml.

5. Lalu bagaimana kita membaca berkas json tersebut? Pertama, kita perlu membuat class yang akan menjadi *blueprint* dari objek artikel. Buat berkas baru bernama **article.dart**. Kemudian tambahkan beberapa propertinya. Sesuaikan properti ini dengan berkas json Anda.

   ```
   class Article {
     late String author;
     late String title;
     late String description;
   ```

```
  late String url;
  late String urlToImage;
  late String publishedAt;
  late String content;

    Article({
      required this.author,
      required this.title,
      required this.description,
      required this.url,
      required this.urlToImage,
      required this.publishedAt,
      required this.content,
    });
  }
```

6. Selanjutnya masih di dalam kelas Article tambahkan juga *named constructor* untuk mengonversi format json menjadi bentuk object Article.

```
Article.fromJson(Map<String, dynamic> article) {
  author = article['author'];
  title = article['title'];
  description = article['description'];
  url = article['url'];
  urlToImage = article['urlToImage'];
  publishedAt = article['publishedAt'];
  content = article['content'];
}
```

7. Setelah *object* article siap, sekarang saatnya kita mulai menyusun tampilan aplikasi. Hapus atau ganti seluruh kode aplikasi Counter dan sisakan kode berikut:

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'News App',
        theme: ThemeData(
          primarySwatch: Colors.blue,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        initialRoute: NewsListPage.routeName,
        routes: {
          NewsListPage.routeName: (context) => NewsListPage(),
        },
      );
    }
```

```
  }
```

8. Buat *stateless widget* baru bernama NewsListPage. Di sini kita menggunakan tampilan standar dengan Scaffold dan AppBar. Untuk menampilkan daftar artikel ke bagian body, kita akan menggunakan widget ListView. Namun, sebelum itu kita perlu membaca aset json-nya bukan? Lantas bagaimana caranya? Karena proses pembacaan aset dilakukan secara *asynchronous*, di sini kita akan memanfaatkan widget FutureBuilder. Widget ini membutuhkan 2 parameter, yaitu *future* yang berisi objek Future yang akan ditampilkan lalu parameter *builder*untuk menjalankan suatu proses ketika nilai Future telah didapatkan.

```
class NewsListPage extends StatelessWidget {
  static const routeName = '/article_list';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('News App'),
      ),
        body: FutureBuilder<String>(
          future:
              DefaultAssetBundle.of(context).loadString('asse
ts/articles.json'),
          builder: (context, snapshot) {},
        ),
      );
    }
  }
```

DefaultAssetBundle pada dasarnya juga merupakan sebuah widget. Widget ini akan membaca String dari berkas aset yang kita tentukan.

9. Karena Future kita mengembalikan String, maka kita perlu mengonversinya menjadi objek yang kita siapkan. Proses konversi ini juga dikenal dengan ***json parsing***. Mari buat fungsi baru di dalam berkas **article.dart**.

```
List<Article> parseArticles(String? json) {
  if (json == null) {
    return [];
  }


  final List parsed = jsonDecode(json);
```

```
  return parsed.map((json) => Article.fromJson(json)).toList();
}
```

10. Kemudian kita akan panggil fungsi `parseArticles()` tersebut dari dalam fungsi *builder*.

```
builder: (context, snapshot) {
  final List<Article> articles = parseArticles(snapshot.data);
},
```

11. Saat ini kita telah mendapatkan daftar Article. Artinya, kita dapat menampilkannya ke layar. Tampilkan widget `ListView`Anda dengan item dari list Article.

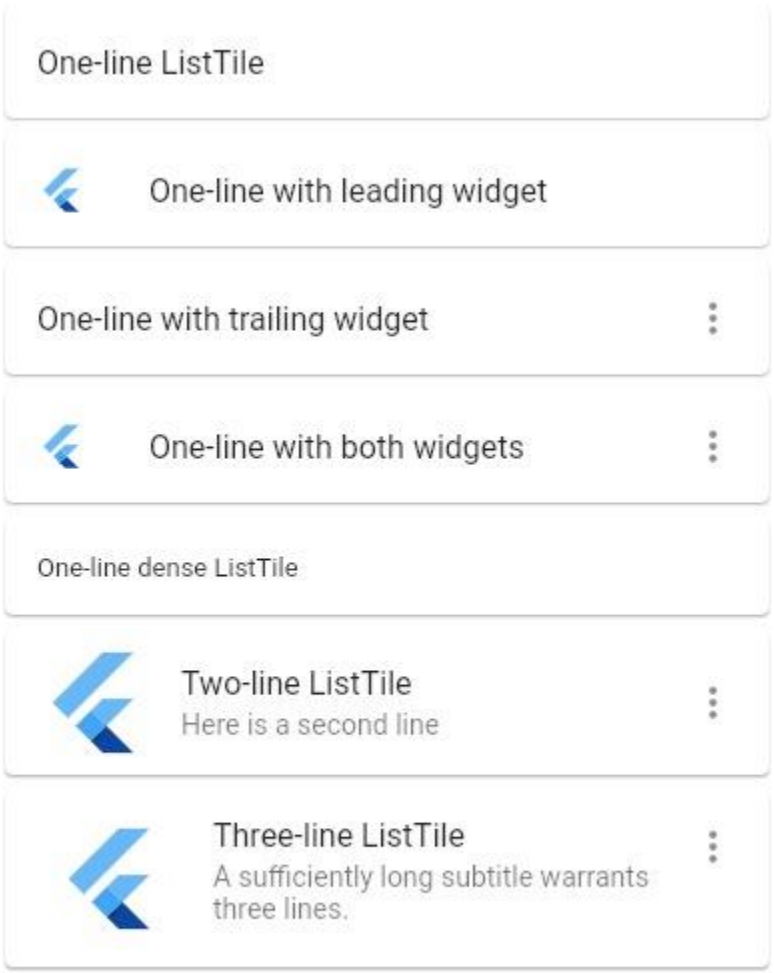
```
builder: (context, snapshot) {
  final List<Article> articles = parseArticles(snapshot.data);
  return ListView.builder(
    itemCount: articles.length,
    itemBuilder: (context, index) {
      return _buildArticleItem(context, articles[index]);
    },
  );
},
```

12. Jika kode Anda mengalami eror, buat method baru bernama `_buildArticleItem()`. Method ini akan menampilkan widget dari masing-masing artikel.

```
Widget _buildArticleItem(BuildContext context, Article article) {
  return ListTile(
    contentPadding:
        const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
    leading: Image.network(
      article.urlToImage,
      width: 100,
    ),
    title: Text(article.title),
       subtitle: Text(article.author),
     );
  }
```

Di sini kita menggunakan widget `ListTile`. Widget ini merupakan widget *built-in* dari Flutter yang pada dasarnya merupakan susunan tiga kolom dan bisa diatur sedemikian rupa. Beberapa kustomisasi yang mungkin dilakukan dengan ListTile adalah

seperti berikut:



13. Sekarang jalankan aplikasi Anda. Seharusnya daftar artikel sudah berhasil ditampilkan.
14. Fitur terakhir yang akan kita kembangkan selanjutnya adalah menampilkan detail dari artikel yang dipilih. Buat berkas baru bernama **detail_page.dart**. Di berkas ini kita akan membuat widget baru yang berperan sebagai halaman detail. Anda dapat berkreasi sendiri untuk menyusun halaman detail atau menggunakan kode berikut:

```
1. class ArticleDetailPage extends StatelessWidget {
2.   static const routeName = '/article_detail';
3.
4.   final Article article;
5.
6.   const ArticleDetailPage({required this.article});
7.
8.   @override
```

```
9.   Widget build(BuildContext context) {
10.       return Scaffold(
11.         appBar: AppBar(
12.           title: Text(article.title),
13.         ),
14.         body: SingleChildScrollView(
15.           child: Column(
16.             children: [
17.               Image.network(article.urlToImage),
18.               Padding(
19.                 padding: EdgeInsets.all(10),
20.                 child: Column(
21.                   crossAxisAlignment: CrossAxisAlignment.star
    t,
22.                   children: [
23.                     Text(article.description),
24.                     Divider(color: Colors.grey),
25.                     Text(
26.                       article.title,
27.                       style: TextStyle(
28.                         color: Colors.black,
29.                         fontWeight: FontWeight.bold,
30.                         fontSize: 24,
31.                       ),
32.                     ),
33.                     Divider(color: Colors.grey),
34.                     Text('Date: ${article.publishedAt}'),
35.                     SizedBox(height: 10),
36.                     Text('Author: ${article.author}'),
37.                     Divider(color: Colors.grey),
38.                     Text(
39.                       article.content,
40.                       style: TextStyle(fontSize: 16),
41.                     ),
42.                     SizedBox(height: 10),
43.                     RaisedButton(
44.                       child: Text('Read more'),
45.                       onPressed: () {},
46.                     ),
47.                   ],
48.                 ),
49.               ),
50.             ],
51.           ),
52.         ),
53.       );
54.     }
55.   }
56.
```

Halaman ArticleDetailPage membutuhkan data artikel yang dipilih sehingga perlu kita kirimkan melalui constructor.

15. Jangan lupa untuk menambahkan *route*ke halaman ArticleDetailPage.

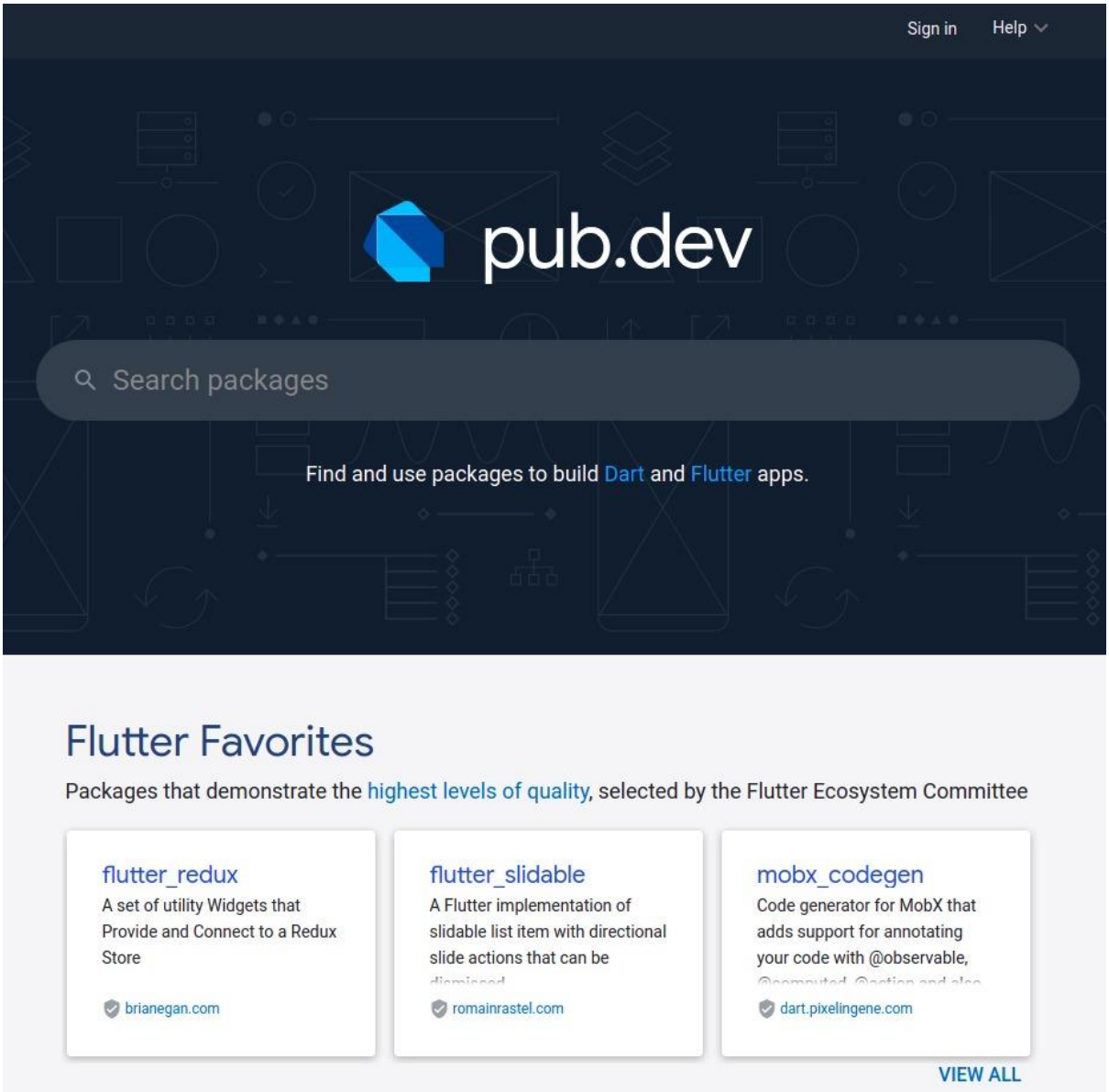
```
routes: {
  NewsListPage.routeName: (context) => NewsListPage(),
  ArticleDetailPage.routeName: (context) => ArticleDetailPage(
        article: ModalRoute.of(context)?.settings.arguments as Article,
      ),
},
```

Selanjutnya buat item yang diklik agar membuka halaman detail. ListTile telah dilengkapi dengan parameter `onTap`untuk menjalankan sebuah fungsi ketika item tersebut diklik.

```
onTap: () {
  Navigator.pushNamed(context, ArticleDetailPage.routeName,
      arguments: article);
  },
```

Masukkan data artikel yang dipilih sebagai argumen navigasi.

16. Terakhir, kita akan menambahkan fitur agar pengguna dapat membaca berita lebih lengkap dengan mengarahkan ke tautan aslinya. Karena artikel kita berbasis web yang umumnya dibuka menggunakan *browser*, mari memanfaatkan widget `WebView`. Namun, WebView bukan termasuk widget yang secara *default* disediakan oleh Flutter, sehingga kita harus menambahkan *package* WebView. Ada banyak sekali *package* yang dapat Anda gunakan pada aplikasi Anda. Anda dapat mencarinya pada tautan https://pub.dev/.



Tim Flutter telah mengembangkan *package* yang menyediakan widget WebView. Carilah webview pada halaman pub.dev.

Kemudian tambahkan package pada berkas pubspec.yaml.

```
1. dependencies:
2.   webview_flutter: ^2.0.4
```

Jangan lupa untuk menginstal package dengan menjalankan perintah:

```
3. flutter pub get
```

17. Setelah *package* terinstal, buatlah widget baru yang akan menampilkan detail artikel. WebViewadalah sebuah widget. Karena setiap komponen dalam Flutter adalah Widget, maka kita juga dapat melakukan kustomisasi pada halaman web dengan menambahkan widget lain seperti AppBar, FloatingActionButton, dan lainnya.

```
1. class ArticleWebView extends StatelessWidget {
2.   static const routeName = '/article_web';
3.
4.   final String url;
5.
6.   const ArticleWebView({required this.url});
7.
8.   @override
9.   Widget build(BuildContext context) {
10.       return Scaffold(
11.         appBar: AppBar(
12.           title: Text('News App'),
13.         ),
14.         body: WebView(
15.           initialUrl: url,
16.         ),
17.       );
18.     }
19.   }
```

Perhatikan bahwa WebView membutuhkan url dari web yang akan ditampilkan, sehingga Anda perlu mengirimkan nilai url ke ArticleWebView.

18. Tambahkan navigasi ke ArticleWebView ketika tombol “Read more” diklik.

```
1. ElevatedButton(
2.   child: Text('Read more'),
3.   onPressed: () {
4.     Navigator.pushNamed(context, ArticleWebView.routeName,
5.         arguments: article.url);
6.   },
7. ),
```

Jangan lupa untuk mendaftarkan routes untuk navigasi.

```
8. routes: {
9.   NewsListPage.routeName: (context) => NewsListPage(),
10.     ArticleDetailPage.routeName: (context) => ArticleDetailPa
   ge(
```

```
            article: ModalRoute.of(context)?.settings.arguments as Article,
          ),
      ArticleWebView.routeName: (context) => ArticleWebView(
            url: ModalRoute.of(context)?.settings.arguments as String,
          ),
    },
```

19. Simpan perubahan dan biarkan Flutter menjalankan fitur *hot reload*-nya.