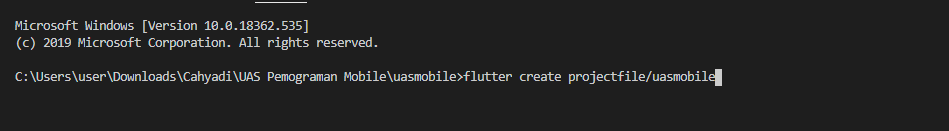
**Membuat Layout**

Aplikasi yang akan di buat :

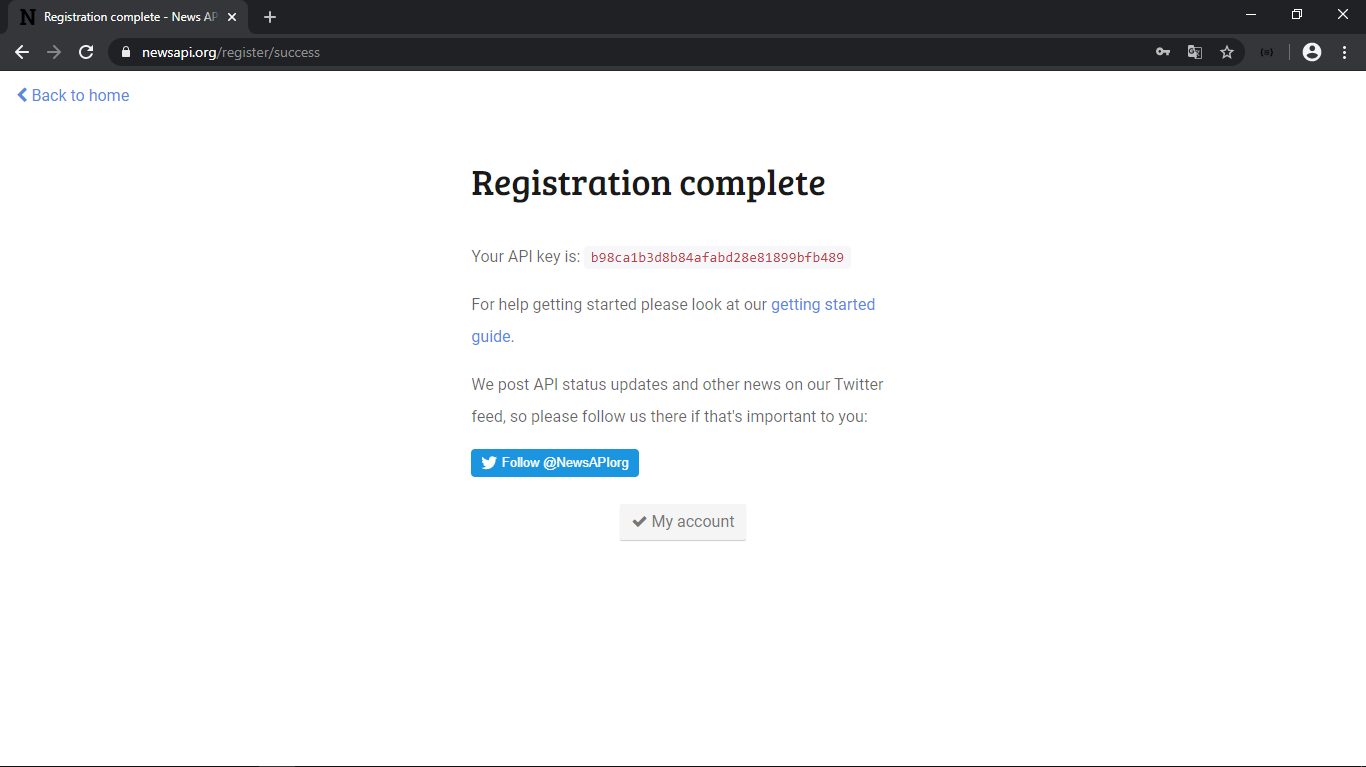


**Step 0 : Buat Project baru**

* + - 1. Buat Basic Project Flutter.
      2. Buka Visual Studio Code dan Jalankan “ Flutter Create projectfile/uasmobile” pada terminal.

**Step 1 : Dapatkan API Key**

1. Buka Halaman newsapi.org
2. Daftar dengan email.



**Step 2 : Input library yang di butuhkan**

1. dependencies:
2. flutter:
3. sdk: flutter
4. flutter\_webview\_plugin: ^0.3.10+1
5. cupertino\_icons: ^0.1.2
6. dio: ^3.0.3
7. flutter\_bloc: ^0.21.0
8. json\_serializable: ^3.2.3
9. intl: ^0.16.0
10. url\_launcher: ^5.2.0
11. cached\_network\_image: ^2.0.0-rc

**cuppertino Icon -> agar dapat menggunakan cuppertino icon pada aplikasi**

**dio -> Klien Http untuk Dart yang mendukung interceptor, sebagai request manager dan dapat juga menghandle download.**

**Flutter webview pluggin -> agar dapat menjalankan url langsung tanpa menggunakan browser.**

**Flutter\_bloc -> mempermudah dalam pengimplementasian pattern bloc.**

**Json\_serializable -> secara otomatis menghasilkan kode untuk mengkonversi ke dan dari JSON pada class-class Dart.**

**Intl -> agar dapat menampilkan date, tanggal, format tahun, dan lainnya.**

**Chached network images -> agar dapat memuat dan menyimpan gambar jaringan. Bisa juga digunakan dengan placeholder.**

**Step 3 : Desain Layout**

Pada aplikasi di atas Berurutan dari atas, terdapat Title, subtitle, Category, Title, subtitle, ListView Berita. Untuk itu kita akan melakukan Layouting sesuai dengan apa yang diinginkan.

1. Title
2. child: Padding(
3. padding: const EdgeInsets.symmetric(horizontal: 16.0),
4. child: RichText(
5. text: TextSpan(
6. children: [
7. TextSpan(
8. text: 'Berita Terbaru \n',
9. style: Theme.of(context).textTheme.title.merge(
10. TextStyle(
11. color: Color(0xFF325384),
12. ),
13. ),
14. ),
15. TextSpan(
16. text: beritaBaru,
17. style: Theme.of(context).textTheme.caption.merge(
18. TextStyle(
19. color: Color(0xFF325384).withOpacity(0.8),
20. fontSize: 10.0,
21. ),
22. ),
23. ),
24. ],
25. ),
26. ),
27. ),
28. );
29. }
30. }

Untuk membuat title Perhatikan, Posisi, padding, Margin. Setelah itu baru Warna, jenis dan Style lainnya.

1. Subtitle atau Date

var today = DateFormat().add\_yMMMMd().format(DateTime.now());

1. Categoty
   1. Untuk Category, dapat dengan membuat class baru, statefulWidget karena aka nada perupahan pada statenya. Pertama kita membuat list dari Category yang di inginkan, perhatikan juga assets yang akan di gunakan apakah dengan menginput di pupspec.yaml
   2. Selanjutnya dapat mendesain tampilan dari List Category sesuai dengan yang di inginkan.
   3. Set state, dimana aka nada perubahan Category ketika user melakukan perubahan pilihan categoty.
2. class WidgetCategory extends StatefulWidget {
3. @override
4. \_WidgetCategoryState createState() => \_WidgetCategoryState();
5. }
6. class \_WidgetCategoryState extends State<WidgetCategory> {
7. final listCategories = [
8. Category('', 'All'),
9. Category('assets/images/bisnis.png', 'Bisnis'),
10. Category('assets/images/hiburan.png', 'Hiburan'),
11. Category('assets/images/kesehatan.png', 'Kesehatan'),
12. Category('assets/images/sains.png', 'Sains'),
13. Category('assets/images/olagraga.png', 'Olahraga'),
14. Category('assets/images/technology.png', 'Technology'),
15. ];
16. int indexSelectedCategory = 0;
17. @override
18. void initState() {
19. final homeBloc = BlocProvider.of<HomeBloc>(context);
20. homeBloc.dispatch(DataEvent(listCategories[indexSelectedCategory].title));
21. super.initState();
22. }
23. @override
24. Widget build(BuildContext context) {
25. final homeBloc = BlocProvider.of<HomeBloc>(context);
26. return Container(
27. height: 74,
28. child: ListView.builder(
29. shrinkWrap: false,
30. scrollDirection: Axis.horizontal,
31. itemBuilder: (context, index) {
32. Category itemCategory = listCategories[index];
33. return Padding(
34. padding: EdgeInsets.only(
35. left: 16.0,
36. right: index == listCategories.length - 1 ? 16.0 : 0.0,
37. ),
38. child: Column(
39. children: <Widget>[
40. GestureDetector(
41. onTap: () {
42. setState(() {
43. indexSelectedCategory = index;
44. homeBloc.dispatch(DataEvent(
45. listCategories[indexSelectedCategory].title));
46. });
47. },
48. child: index == 0
49. ? Container(
50. width: 45.0,
51. height: 45.0,
52. decoration: BoxDecoration(
53. shape: BoxShape.circle,
54. color: Color(0xFFDDCDDE),
55. border: indexSelectedCategory == index
56. ? Border.all(
57. color: Colors.white,
58. width: 5.0,
59. )
60. : null,
61. ),
62. child: Icon(
63. Icons.library\_books,
64. color: Colors.white,
65. ),
66. )
67. : Container(
68. width: 45.0,
69. height: 45.0,
70. decoration: BoxDecoration(
71. shape: BoxShape.circle,
72. image: DecorationImage(
73. image: AssetImage(itemCategory.image),
74. fit: BoxFit.cover,
75. ),
76. border: indexSelectedCategory == index
77. ? Border.all(
78. color: Colors.white,
79. width: 5.0,
80. )
81. : null,
82. ),
83. ),
84. ),
85. SizedBox(height: 8.0),
86. Text(
87. itemCategory.title,
88. style: TextStyle(
89. fontSize: 14,
90. color: Color(0xFF325384),
91. fontWeight: indexSelectedCategory == index
92. ? FontWeight.bold
93. : FontWeight.normal,
94. ),
95. ),
96. ],
97. ),
98. );
99. },
100. itemCount: listCategories.length,
101. ),
102. );
103. }
104. }
105. List Berita
     1. Buat class Stateful untuk berita
     2. Setting padding dan margin
     3. Buat widgetcontent untuk menampilkan data dari newsapi.org
     4. Buat juga indicator loading, Notifikasi error dan dan notifikasi lainnya.
     5. Selanjutnya setting layout sesuai dengan yang di inginkan.
106. class WidgetBerita extends StatefulWidget {
107. WidgetBerita();
108. @override
109. \_WidgetBeritaState createState() => \_WidgetBeritaState();
110. }
111. class \_WidgetBeritaState extends State<WidgetBerita> {
112. @override
113. Widget build(BuildContext context) {
114. MediaQueryData mediaQuery = MediaQuery.of(context);
115. final HomeBloc homeBloc = BlocProvider.of<HomeBloc>(context);
116. return Padding(
117. padding: EdgeInsets.only(
118. left: 16.0,
119. top: 8.0,
120. right: 16.0,
121. bottom: mediaQuery.padding.bottom + 16.0,
122. ),
123. child: BlocListener<HomeBloc, DataState>(
124. listener: (context, state) {
125. if (state is DataFailed) {
126. Scaffold.of(context).showSnackBar(
127. SnackBar(content: Text(state.errorMessage)),
128. );
129. }
130. },
131. child: BlocBuilder(
132. bloc: homeBloc,
133. builder: (BuildContext context, DataState state) {
134. return \_widgetContent(state, mediaQuery);
135. },
136. ),
137. ),
138. );
139. }
140. Widget \_widgetContent(
141. DataState state, MediaQueryData mediaQuery) {
142. if (state is DataLoading) {
143. return Center(
144. child: Platform.isAndroid
145. ? CircularProgressIndicator()
146. : CupertinoActivityIndicator(),
147. );
148. } else if (state is DataSuccess) {
149. ResponseC data = state.data;
150. return ListView.separated(
151. padding: EdgeInsets.zero,
152. itemCount: data.articles.length,
153. separatorBuilder: (context, index) {
154. return Divider();
155. },
156. itemBuilder: (context, index) {
157. Article itemArticle = data.articles[index];
158. if (index == 0) {
159. return Stack(
160. children: <Widget>[
161. ClipRRect(
162. child: CachedNetworkImage(
163. imageUrl: itemArticle.urlToImage,
164. height: 192.0,
165. width: mediaQuery.size.width,
166. fit: BoxFit.cover,
167. placeholder: (context, url) => Platform.isAndroid
168. ? CircularProgressIndicator()
169. : CupertinoActivityIndicator(),
170. errorWidget: (context, url, error) => Image.asset(
171. 'assets/images/img\_not\_found.jpg',
172. fit: BoxFit.cover,
173. ),
174. ),
175. borderRadius: BorderRadius.all(
176. Radius.circular(8.0),
177. ),
178. ),
179. GestureDetector(
180. onTap: () async {
181. if (await canLaunch(itemArticle.url)) {
182. var urln = itemArticle.url;
183. Navigator.push(
184. context,
185. new MaterialPageRoute(
186. builder: (BuildContext context) => new ArticlePage(urln),
187. ));
188. } else {
189. scaffoldState.currentState.showSnackBar(SnackBar(
190. content: Text('Error'),
191. ));
192. }
193. },
194. child: Container(
195. width: mediaQuery.size.width,
196. height: 192.0,
197. decoration: BoxDecoration(
198. borderRadius: BorderRadius.all(
199. Radius.circular(8.0),
200. ),
201. gradient: LinearGradient(
202. colors: [
203. Colors.black.withOpacity(0.8),
204. Colors.black.withOpacity(0.0),
205. ],
206. begin: Alignment.topCenter,
207. end: Alignment.bottomCenter,
208. stops: [
209. 0.0,
210. 0.7,
211. ],
212. ),
213. ),
214. ),
215. ),
216. Column(
217. crossAxisAlignment: CrossAxisAlignment.start,
218. children: <Widget>[
219. Padding(
220. padding: const EdgeInsets.only(
221. left: 12.0,
222. top: 12.0,
223. right: 12.0,
224. ),
225. child: Text(
226. itemArticle.title,
227. style: TextStyle(
228. color: Colors.white,
229. ),
230. overflow: TextOverflow.ellipsis,
231. maxLines: 2,
232. ),
233. ),
234. Padding(
235. padding: const EdgeInsets.only(
236. left: 12.0,
237. top: 4.0,
238. right: 12.0,
239. ),
240. child: Wrap(
241. children: <Widget>[
242. Icon(
243. Icons.launch,
244. color: Colors.white.withOpacity(0.8),
245. size: 12.0,
246. ),
247. SizedBox(width: 4.0),
248. Text(
249. '${itemArticle.source.name}',
250. style: TextStyle(
251. color: Colors.white.withOpacity(0.8),
252. fontSize: 11.0,
253. ),
254. ),
255. ],
256. ),
257. ),
258. ],
259. ),
260. ],
261. );
262. } else {
263. return GestureDetector(
264. onTap: () async {
265. if (await canLaunch(itemArticle.url)) {
266. var urln = itemArticle.url;
267. Navigator.push(
268. context,
269. new MaterialPageRoute(
270. builder: (BuildContext context) => new ArticlePage(urln),
271. ));
272. }
273. },
274. child: Container(
275. width: mediaQuery.size.width,
276. child: Row(
277. crossAxisAlignment: CrossAxisAlignment.start,
278. children: <Widget>[
279. Expanded(
280. child: SizedBox(
281. height: 72.0,
282. child: Column(
283. crossAxisAlignment: CrossAxisAlignment.start,
284. mainAxisAlignment: MainAxisAlignment.spaceBetween,
285. children: <Widget>[
286. Text(
287. itemArticle.title,
288. overflow: TextOverflow.ellipsis,
289. maxLines: 3,
290. style: TextStyle(
291. fontSize: 16.0,
292. color: Color(0xFF325384),
293. fontWeight: FontWeight.w400,
294. ),
295. ),
296. Wrap(
297. crossAxisAlignment: WrapCrossAlignment.center,
298. children: <Widget>[
299. Icon(
300. Icons.launch,
301. size: 12.0,
302. color: Color(0xFF325384).withOpacity(0.5),
303. ),
304. SizedBox(width: 4.0),
305. Text(
306. itemArticle.source.name,
307. style: TextStyle(
308. color: Color(0xFF325384).withOpacity(0.5),
309. fontSize: 12.0,
310. ),
311. ),
312. ],
313. ),
314. ],
315. ),
316. ),
317. ),
318. Padding(
319. padding: const EdgeInsets.only(left: 16.0),
320. child: ClipRRect(
321. child: CachedNetworkImage(
322. imageUrl: itemArticle.urlToImage,
323. imageBuilder: (context, imageProvider) {
324. return Container(
325. width: 72.0,
326. height: 72.0,
327. decoration: BoxDecoration(
328. image: DecorationImage(
329. image: imageProvider,
330. fit: BoxFit.cover,
331. ),
332. ),
333. );
334. },
335. placeholder: (context, url) => Container(
336. width: 72.0,
337. height: 72.0,
338. child: Center(
339. child: Platform.isAndroid
340. ? CircularProgressIndicator()
341. : CupertinoActivityIndicator(),
342. ),
343. ),
344. errorWidget: (context, url, error) => Image.asset(
345. 'assets/images/img\_not\_found.jpg',
346. fit: BoxFit.cover,
347. width: 72.0,
348. height: 72.0,
349. ),
350. ),
351. borderRadius: BorderRadius.all(
352. Radius.circular(4.0),
353. ),
354. ),
355. ),
356. ],
357. ),
358. ),
359. );
360. }
361. },
362. );
363. } else {
364. return Container();
365. }
366. }
367. }

**Step 4 : Setting API**

1. Setelah mendapatkan API KEY setelah registrasi di newsapi.org
2. Buat Variabel untuk url API nya.
3. Buat Response untuk mendapatkan data per category dengan menambahkan parameter category pada url API.

import 'package:dio/dio.dart';

import 'package:flutter\_uas/model/response\_category.dart';

import 'dart:async';

class Api {

  final Dio \_dio = Dio();

  final String \_baseUrl =

      'https://newsapi.org/v2/top-headlines?country=id&apiKey=b98ca1b3d8b84afabd28e81899bfb489';

  void printOutError(error, StackTrace stacktrace) {

    print('Exception occured: $error with stacktrace: $stacktrace');

  }

  Future<ResponseC> getAll() async {

    try {

      final response = await \_dio.get(\_baseUrl);

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getBisnis() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=business');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getHiburan() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=entertainment');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getKesehatan() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=health');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getSains() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=science');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getOlahraga() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=sport');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

  Future<ResponseC> getTechnology() async {

    try {

      final response = await \_dio.get('$\_baseUrl&category=technology');

      return ResponseC.fromJson(response.data);

    } catch (error, stacktrace) {

      printOutError(error, stacktrace);

      return ResponseC.withError('$error');

    }

  }

}

**Step 5 : Buat Bloc**

**Untuk menampilkan berita dengan informasi yang lebih Rapih kita dapat menggunakan Bloc, agar berita yang ditampilkan lebih spesifik.**

1. **Bloc akan menghandle category dari Url API yang ada dan menampilkannya berdasarkan interaksi User dengan category di aplikasi.**

 yield DataLoading();

    final apirepo = ApiRepo();

    final categoryLowerCase = event.category.toLowerCase();

    switch (categoryLowerCase) {

      case 'all':

        final data = await apirepo.fetchAll();

        if (data.error == null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed('Failed to fetch data');

        }

        break;

      case 'bisnis':

        final data = await apirepo.fetchBisnis();

        if (data.error == null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      case 'hiburan':

        final data = await apirepo.fetchHiburan();

        if (data.error == null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      case 'kesehatan':

        final data = await apirepo.fetchKesehatan();

        if (data.error == null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      case 'sains':

        final data = await apirepo.fetchSains();

        if (data.error == null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      case 'olahraga':

        final data = await apirepo.fetchOlahraga();

        if (data != null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      case 'technology':

        final data = await apirepo.fetchTechnology();

        if (data != null) {

          yield DataSuccess(data);

        } else {

          yield DataFailed(data.error);

        }

        break;

      default:

        yield DataFailed('category error');

    }

  }

}

**Step 6 : Buat Model**

Buat Model sesuai dengan data dari API yang akan di ambil dalam bentuk json, data json ini selanjutnya akan di konsumsi dan di tampilkan pada UI.

import 'package:json\_annotation/json\_annotation.dart';

Category \_$CategoryFromJson(Map<String, dynamic> json) {

  return Category(

    json['image'] as String,

    json['title'] as String,

  );

}

Map<String, dynamic> \_$CategoryToJson(Category instance) => <String, dynamic>{

      'image': instance.image,

      'title': instance.title,

    };

@JsonSerializable()

class Category {

  String image;

  String title;

  Category(this.image, this.title);

  factory Category.fromJson(Map<String, dynamic> json) => \_$CategoryFromJson(json);

  Map<String, dynamic> toJson() => \_$CategoryToJson(this);

  @override

  String toString() {

    return 'Category{image: $image, title: $title}';

  }

}

Map<String, dynamic> \_$ResponseCToJson(

        ResponseC instance) =>

    <String, dynamic>{

      'status': instance.status,

      'totalResults': instance.totalResults,

      'articles': instance.articles,

    };

Article \_$ArticleFromJson(Map<String, dynamic> json) {

  return Article(

    json['source'] == null

        ? null

        : Source.fromJson(json['source'] as Map<String, dynamic>),

    json['author'] as String,

    json['title'] as String,

    json['description'] as String,

    json['url'] as String,

    json['urlToImage'] as String,

    json['publishedAt'] as String,

    json['content'] as String,

  );

}

**Step 7 : Buat Page untuk tampilkan berita**

**Untuk menampilkan detail berita tanpe membuka browser tetapi langsung di aplikasi dapat menggunakan Webview, yang membuat 1 page full untuk menampilkan website yang di klick, untuk itu perlu parameter untuk meneruskan url yang di pilih ke page selanjutnya. Di sini kita menggunakan urln untuk memuat url yang dipilih.**

return new Scaffold(

            body: new SafeArea(

              child: new WebviewScaffold(

              url: "$urln",

              appBar: Appbar(

                height: 55,

              ),

              withZoom: true,

              withLocalStorage: true,

              hidden: true,

              initialChild: Container(

                color: Colors.white,

                child: const Center(

                  child: Text('Memuat...'),

              ),

          ),

        ),

            ),

    );

  }

}

**Terima Kasih 😊**