

Dokumentasi Proyek: Anime Flutter App

Anime Flutter App adalah aplikasi yang menampilkan daftar anime dari API Jikan (MyAnimeList). Aplikasi ini memungkinkan pengguna untuk melihat daftar anime populer, anime yang akan datang, dan anime musiman. Selain itu, pengguna dapat menandai anime favorit, memfilter berdasarkan genre, dan menyimpan daftar favorit secara permanen menggunakan SharedPreferences.

Fitur Utama:

- Mengambil data anime dari API Jikan (Popular, Upcoming, Seasonal)
- Menampilkan daftar anime dalam UI yang menarik
- Melihat detail anime lengkap dengan sinopsis dan gambar
- Menambahkan/menghapus anime dari daftar favorit
- Menyimpan daftar favorit secara permanen menggunakan SharedPreferences
- Filter berdasarkan genre anime
- Navigasi menggunakan TabBar dan Dropdown untuk filter

API yang Digunakan

Aplikasi ini mengambil data dari **Jikan API**, API publik dari MyAnimeList. Endpoints yang Digunakan:

- **Anime Populer :** <https://api.jikan.moe/v4/top/anime>
- **Anime yang Akan Datang :** <https://api.jikan.moe/v4/seasons/upcoming>
- **Anime Musiman (Seasonal) :** <https://api.jikan.moe/v4/seasons/now>

Untuk menggunakan kode dalam konteks pemrograman Dart, berikut adalah langkah-langkah yang bisa diikuti:

1. Mempersiapkan Lingkungan Pengembangan
 - Menggunakan dartpad yang bisa diakses melalui <https://dartpad.dev/>
2. Tulis Kode Dart
 - Copy code yang ada "main.dart" pada github :
https://github.com/1OneGod1/anime_flutter_app
3. Jalankan kode Dart dengan klik Run

Penjelasan Fitur

- **Mengambil Data Anime Secara Asynchronous**
 - Menggunakan **http** package untuk melakukan request API.
 - Menggunakan **Future.wait** untuk mengambil beberapa kategori anime sekaligus.
- **Menyimpan Anime Favorit Secara Permanen**
 - Menggunakan **SharedPreferences** untuk menyimpan daftar favorit, sehingga tetap ada meskipun aplikasi ditutup.
- **Tampilan UI**
 - **ListView** → Untuk menampilkan daftar anime.
 - **TabBar & TabBarView** → Untuk berpindah antar kategori.
 - **DropdownButton** → Untuk memilih genre anime yang ingin difilter.
- **Penanganan Error**
 - **Menggunakan try-catch** untuk menangani error saat mengambil data API.
 - **Menampilkan pesan error dengan ScaffoldMessenger** jika gagal mengambil data.

Dokumentasi Code:

```

1 import 'dart:convert'; // Digunakan untuk mengelola data JSON dari API
2 import 'package:flutter/material.dart';
3 import 'package:http/http.dart'
4   as http; // Package untuk mengambil data dari API
5 import 'package:shared_preferences/shared_preferences.dart'; // Package untuk menyimpan daftar favorit secara lokal
6
7 // Model class untuk merepresentasikan data anime
8 class Anime {
9   final int id;
10  final String title;
11  final String imageUrl;
12  final String synopsis;
13  final List<String> genres;
14
15  Anime({
16    required this.id,
17    required this.title,
18    required this.imageUrl,
19    required this.synopsis,
20    required this.genres,
21  });
22
23 // Factory method untuk mengubah JSON menjadi objek Anime
24 factory Anime.fromJson(Map<String, dynamic> json) {
25   return Anime(
26     id: json['mal_id'],
27     title: json['title'],
28     imageUrl: json['images'][('jpg')]['image_url'],
29     synopsis: json['synopsis'] ?? "No description available.",
30     genres: (json['genres'] as List).map((g) => g['name'] as String).toList(),
31   );
32 }
33
34 // Method untuk mengubah objek Anime menjadi JSON (berguna untuk SharedPreferences)
35 Map<String, dynamic> toJson() {
36   return {
37     'id': id,
38     'title': title,
39     'imageUrl': imageUrl,
40     'synopsis': synopsis,
41     'genres': genres,
42   };
43 }

```

Kode ini adalah sebuah model Anime dalam bahasa Dart yang digunakan untuk merepresentasikan data anime yang diambil dari API Jikan (MyAnimeList). Model ini memiliki atribut id, title (judul anime), imageUrl (URL gambar anime), synopsis (deskripsi anime), dan genres (daftar genre dalam bentuk list string). Kelas ini memiliki constructor untuk menginisialisasi objek anime. Selain itu, terdapat factory method fromJson yang digunakan untuk mengonversi data JSON dari API menjadi objek Anime, dengan pengecekan apabila synopsis tidak tersedia maka akan diisi dengan teks default "No description available.". Pada bagian genres, JSON yang berupa list akan di-mapping menjadi daftar string. Kode ini juga memiliki method toJson, yang digunakan untuk mengubah objek Anime kembali menjadi format JSON, sehingga bisa disimpan dalam SharedPreferences atau dikirim ke server. Fungsi ini memastikan bahwa data tetap terstruktur dengan baik saat diambil dari atau disimpan ke dalam penyimpanan lokal.

```
46 // Fungsi utama untuk menjalankan aplikasi
47 void main() {
48   runApp(const AnimeApp());
49 }
50
51 // StatefulWidget untuk menangani state aplikasi
52 class AnimeApp extends StatefulWidget {
53   const AnimeApp({Key? key}) : super(key: key);
54
55   @override
56   State<AnimeApp> createState() => _AnimeAppState();
57 }
58
59 class _AnimeAppState extends State<AnimeApp> {
60   List<Anime> popularAnime = [];
61   List<Anime> upcomingAnime = [];
62   List<Anime> seasonalAnime = [];
63   List<Anime> favoriteAnime = [];
64   String selectedGenre =
65     "All"; // Filter genre default (semua anime ditampilkan)
66
67   @override
68   void initState() {
69     super.initState();
70     fetchAnime(); // Memuat data anime dari API
71     loadFavorites(); // Memuat daftar favorit dari local storage
72   }
73
74   // Fungsi untuk mengambil data dari API secara asynchronous
75   Future<void> fetchAnime() async {
76     final urls = {
77       'popular': 'https://api.jikan.moe/v4/top/anime',
78       'upcoming': 'https://api.jikan.moe/v4/seasons/upcoming',
79       'seasonal': 'https://api.jikan.moe/v4/seasons/now',
80     };
81
82     try {
83       // Mengambil semua data sekaligus dengan Future.wait
84       final responses = await Future.wait(
85         urls.values.map((url) => http.get(Uri.parse(url))),
86       );
87
88       // Jika semua request berhasil, simpan data ke dalam state
89       if (responses.every((res) => res.statusCode == 200)) {
90         setState(() {
91           popularAnime =
92             (jsonDecode(responses[0].body)['data'] as List)
93               .map((e) => Anime.fromJson(e))
94               .toList();
95           upcomingAnime =
96             (jsonDecode(responses[1].body)['data'] as List)
97               .map((e) => Anime.fromJson(e))
98               .toList();
99           seasonalAnime =
100             (jsonDecode(responses[2].body)['data'] as List)
101               .map((e) => Anime.fromJson(e))
102               .toList();
103           favoriteAnime =
104             (jsonDecode(responses[3].body)['data'] as List)
105               .map((e) => Anime.fromJson(e))
106               .toList();
107         });
108       }
109     } catch (e) {
110       print('Error fetching anime data: $e');
111     }
112   }
113
114   void loadFavorites() {
115     // Implementasi untuk memuat daftar favorit dari local storage
116   }
117 }
```

Kode ini merupakan bagian utama dari aplikasi Flutter yang menangani pengambilan data anime dari API dan pengelolaan state menggunakan StatefulWidget. Fungsi main() digunakan untuk menjalankan aplikasi dengan runApp(const AnimeApp()), di mana AnimeApp adalah StatefulWidget yang memiliki State bernama _AnimeAppState.

Di dalam _AnimeAppState, terdapat beberapa list data seperti popularAnime, upcomingAnime, seasonalAnime, dan favoriteAnime, yang digunakan untuk menyimpan daftar anime berdasarkan kategori. Selain itu, terdapat variabel selectedGenre yang digunakan untuk menyaring anime berdasarkan genre dengan nilai awal "All".

Pada metode initState(), dua fungsi dipanggil secara otomatis saat aplikasi dijalankan:

1. fetchAnime() → Mengambil data anime dari API Jikan secara asynchronous.
2. loadFavorites() → Memuat daftar anime favorit yang telah disimpan sebelumnya dari local storage (SharedPreferences).

Fungsi fetchAnime() bekerja dengan mengambil data dari tiga endpoint API (popular, upcoming, seasonal). Untuk mengoptimalkan prosesnya, digunakan Future.wait() yang memungkinkan multiple API calls dijalankan secara paralel. Jika semua request berhasil (status kode 200), maka data yang diterima dari API diubah menjadi list objek Anime menggunakan Anime.fromJson(), lalu disimpan ke dalam state aplikasi menggunakan setState() agar UI diperbarui secara otomatis.

Jika terjadi kesalahan saat mengambil data dari API, kode ini menangani error menggunakan try-catch, sehingga aplikasi tidak crash dan dapat menampilkan pesan error kepada pengguna.

```

111 // Fungsi untuk memuat daftar favorit dari SharedPreferences
112 Future<void> loadFavorites() async {
113   final prefs = await SharedPreferences.getInstance();
114   final favoriteData = prefs.getStringList('favoriteAnime') ?? [];
115
116   setState(() {
117     favoriteAnime =
118       favoriteData.map((data) => Anime.fromJson(jsonDecode(data))).toList();
119   });
120 }
121
122 // Fungsi untuk menyimpan daftar favorit ke SharedPreferences
123 Future<void> saveFavorites() async {
124   final prefs = await SharedPreferences.getInstance();
125   final favoriteData =
126     favoriteAnime.map((anime) => jsonEncode(anime.toJson())).toList();
127   await prefs.setStringList('favoriteAnime', favoriteData);
128 }
129
130 // Fungsi untuk menambah/menghapus anime dari daftar favorit
131 void toggleFavorite(Anime anime) {
132   setState(() {
133     if (favoriteAnime.any((fav) => fav.id == anime.id)) {
134       favoriteAnime.removeWhere((fav) => fav.id == anime.id);
135     } else {
136       favoriteAnime.add(anime);
137     }
138   });
139 }
140   saveFavorites();
141 }
142
143 // Fungsi untuk memfilter anime berdasarkan genre yang dipilih
144 List<Anime> getFilteredAnime(List<Anime> animeList) {
145   if (selectedGenre == "All") return animeList;
146   return animeList
147     .where((anime) => anime.genres.contains(selectedGenre))
148     .toList();
149 }
150
151 @override
152 Widget build(BuildContext context) {
153   return MaterialApp(
154     debugShowCheckedModeBanner: false,
155     home: Scaffold(
156       appBar: AppBar(
157         title: const Text("Anime List"),
158         backgroundColor: Colors.indigo,
159       ),
160       body: Center(
161         child: Container(
162           width: 300,
163           height: 300,
164           color: Colors.purple,
165         ),
166       ),
167     ),
168   );
169 }
170
171 
```

Kode ini berfokus pada pengelolaan daftar anime favorit menggunakan SharedPreferences, serta fitur penyaringan anime berdasarkan genre. Fungsi loadFavorites() bertugas untuk memuat daftar anime favorit dari penyimpanan lokal (SharedPreferences). Data disimpan dalam bentuk string list JSON, sehingga perlu dikonversi kembali menjadi objek Anime menggunakan jsonDecode() dan Anime.fromJson(), lalu dimasukkan ke dalam list favoriteAnime.

Fungsi saveFavorites() digunakan untuk menyimpan daftar anime favorit ke SharedPreferences. Proses ini dilakukan dengan mengonversi setiap objek Anime ke format JSON menggunakan toJson(), lalu disimpan dalam list String menggunakan jsonEncode(). Fungsi toggleFavorite(Anime anime) digunakan untuk menambahkan atau menghapus anime dari daftar favorit. Jika anime sudah ada di dalam daftar (favoriteAnime), maka akan dihapus dengan removeWhere(). Jika belum ada, anime akan ditambahkan ke dalam list. Setelah perubahan dilakukan dengan setState(), daftar favorit langsung disimpan ke SharedPreferences dengan memanggil saveFavorites().

Fungsi getFilteredAnime() digunakan untuk memfilter anime berdasarkan genre yang dipilih pengguna. Jika selectedGenre bernilai "All", maka semua anime akan ditampilkan tanpa filter. Jika tidak, maka hanya anime yang memiliki genre yang sesuai (anime.genres.contains(selectedGenre)) yang akan ditampilkan. Terakhir, dalam metode build(), aplikasi dibuat menggunakan MaterialApp dengan Scaffold sebagai struktur utama, serta AppBar sebagai judul halaman utama aplikasi.

```

202     ),
203     // Menampilkan daftar anime berdasarkan tab yang dipilih
204     Expanded(
205       child: TabBarView(
206         children: [
207           AnimeList(
208             animes: getFilteredAnime(popularAnime),
209             onFavorite: toggleFavorite,
210             favorites: favoriteAnime,
211           ),
212           AnimeList(
213             animes: getFilteredAnime(upcomingAnime),
214             onFavorite: toggleFavorite,
215             favorites: favoriteAnime,
216           ),
217           AnimeList(
218             animes: getFilteredAnime(seasonalAnime),
219             onFavorite: toggleFavorite,
220             favorites: favoriteAnime,
221           ),
222           AnimeList(
223             animes: favoriteAnime,
224             onFavorite: toggleFavorite,
225             favorites: favoriteAnime,
226           ),
227         ],
228       ),
229     ),
230   ],
231 ),
232 ),
233 );
234 );
235 );
236 }
237
238 // Widget untuk menampilkan daftar anime
239 class AnimeList extends StatelessWidget {
240   final List<Anime> animes;
241   final Function(Anime) onFavorite;
242   final List<Anime> favorites;
243
244   const AnimeList({
245     Key? key,
246     required this.animes,
247     required this.onFavorite,
248     required this.favorites,
249   }) : super(key: key);
250
251   @override
252   Widget build(BuildContext context) {

```

Kode ini bertanggung jawab untuk menampilkan daftar anime berdasarkan kategori yang dipilih menggunakan TabBarView, yang mencakup anime populer, yang akan datang, musiman, dan favorit. Setiap kategori menggunakan widget AnimeList, yang berfungsi untuk menampilkan daftar anime dalam bentuk ListView. Widget ini menerima tiga parameter utama: animes (data anime yang ditampilkan), onFavorite (fungsi untuk menambah atau menghapus anime dari favorit), dan favorites (daftar anime yang telah ditandai sebagai favorit). Dengan pendekatan ini, kode menjadi lebih modular dan terorganisir, sehingga tampilan dan logika data terpisah dengan baik.

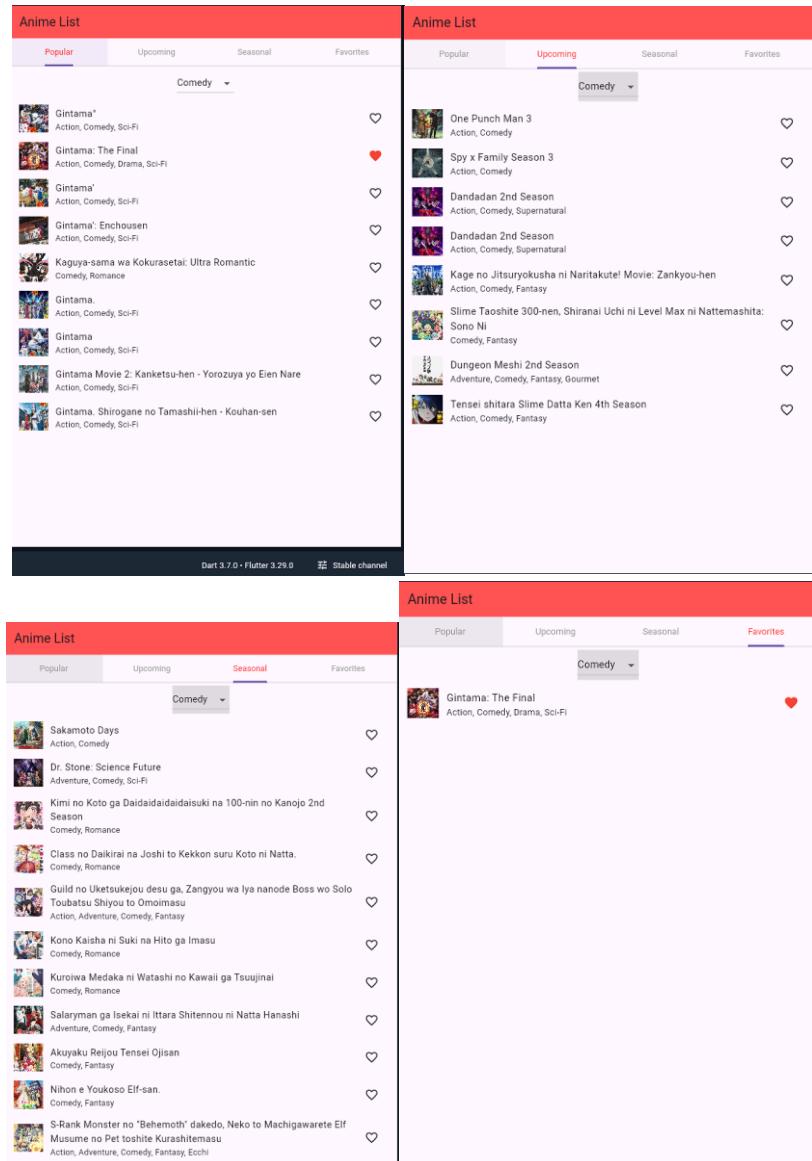
```

251 @override
252 Widget build(BuildContext context) {
253   return animes.isEmpty
254     ? const Center(
255       child: CircularProgressIndicator(),
256     ) // Tampilkan loading jika data belum tersedia
257     : ListView.builder(
258       itemCount: animes.length,
259       itemBuilder: (context, index) {
260         final anime = animes[index];
261         final isFavorite = favorites.any((fav) => fav.id == anime.id);
262
263         return ListTile(
264           leading: Image.network(
265             anime.imageUrl,
266             width: 50,
267             height: 70,
268             fit: BoxFit.cover,
269           ),
270           title: Text(anime.title),
271           subtitle: Text(anime.genres.join(", ")),
272           trailing: IconButton(
273             icon: Icon(
274               isFavorite ? Icons.favorite : Icons.favorite_border,
275               color: isFavorite ? Colors.red : null,
276             ),
277             onPressed: () => onFavorite(anime),
278           ),
279           onTap: () => Navigator.push(
280             context,
281             MaterialPageRoute(
282               builder: (_) => AnimeDetail(anime: anime),
283             ),
284           ),
285         );
286       },
287     );
288   );
289 }
290 }
291
292 class AnimeDetail extends StatelessWidget {
293   final Anime anime;
294
295   const AnimeDetail({Key? key, required this.anime}) : super(key: key);
296
297   @override
298   Widget build(BuildContext context) {
299     return Scaffold(
300       appBar: AppBar(
301         title: Text(anime.title),
302         backgroundColor: Colors.redAccent,

```

Kode ini berfungsi untuk menampilkan daftar anime dalam bentuk ListView dan memberikan tampilan detail anime saat item diklik. Jika data anime belum tersedia, maka akan ditampilkan CircularProgressIndicator sebagai indikator loading. Setiap item anime ditampilkan menggunakan ListTile, yang berisi gambar, judul, dan genre anime. Terdapat IconButton untuk menandai anime sebagai favorit, di mana ikon berubah antara ikon hati kosong dan hati merah tergantung apakah anime sudah masuk ke daftar favorit atau belum. Saat pengguna mengklik item anime, aplikasi akan berpindah ke halaman AnimeDetail menggunakan Navigator.push(). AnimeDetail adalah widget yang menampilkan informasi lengkap tentang anime dalam Scaffold, dengan judul anime di dalam AppBar berwarna merah.

Aplikasi (UI Preview)



Gambar ini menunjukkan tampilan aplikasi Anime Flutter App yang telah berhasil dijalankan. Aplikasi ini menampilkan daftar anime berdasarkan kategori yang dipilih melalui TabBar di bagian atas, yaitu Popular, Upcoming, Seasonal, dan Favorites. Saat ini, tab Popular sedang aktif, dengan daftar anime yang dapat difilter berdasarkan genre menggunakan dropdown, yang saat ini disetel ke Comedy. Setiap anime ditampilkan dengan gambar, judul, dan daftar genre dalam bentuk ListView. Pengguna dapat menandai anime sebagai favorit dengan menekan ikon hati di sebelah kanan. Jika anime sudah masuk dalam daftar favorit, ikon hati akan berubah menjadi berwarna merah, seperti yang terlihat pada "Gintama: The Final".