Atividade Flutter — City Explorer & Pet Rescue

Valendo 10 pontos (2 exercícios de 5 pontos cada)

Objetivo geral. Construir dois mini-apps com *tema escuro (Material Design)*, rotas nomeadas, passagem de parâmetros, estado via `setState`, uso de `GestureDetector`, condicionais `if/else`, **assets de imagens** e os widgets: `Image.network`, `Container`, `Text`, `Row`, `Column`, `Padding`, `SingleChildScrollView`, `Scaffold`, `StatefulWidget`, `ElevatedButton`. Além disso, utilizar os pacotes do pub.dev: `google_fonts`, `shimmer`, `animated_text_kit`. Desafio opcional: `youtube_player_flutter` e `url_launcher`.

1. Tutorial: usando pacotes do `pub.dev` (primeira vez)

Passo 1 — Editar `pubspec.yaml` (adicione as dependências e a pasta de assets):

```
dependencies:
  flutter:
    sdk: flutter
  google_fonts: ^6.2.1
  shimmer: ^3.0.0
  animated_text_kit: ^4.2.2
  url_launcher: ^6.3.0 # desafio (abrir links externos)
  youtube_player_flutter: ^9.0.0 # desafio (vdeos YouTube)

flutter:
  assets:
    - assets/images/ # inclua ao menos 1 imagem local aqui
```

Passo 2 — Instalar dependências. No terminal (pasta do projeto): `flutter pub get`

Passo 3 — Imports típicos no código.

```
import 'package:google_fonts/google_fonts.dart';
import 'package:animated_text_kit/animated_text_kit.dart';
import 'package:shimmer/shimmer.dart';
// Desafios:
import 'package:url_launcher/url_launcher.dart';
import 'package:youtube_player_flutter/youtube_player_flutter.dart';
```

Passo 4 — Android (Internet). Se algo falhar em *release*, verifique no arquivo `android/app/src/main/AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Abrir links externos (`url_launcher`). Em alguns cenários, pode ser preciso declarar `queries`:

```
<queries>
  <intent>
    <action android:name="android.intent.action.VIEW" />
    <data android:mimeType="*/*" />
  </intent>
</queries>
```

iOS. Para abrir apps específicos (ex.: WhatsApp), ajuste esquemas no `Info.plist`. Para sites HTTPS comuns, geralmente não é necessário.

Passo 5 — Tema escuro e rotas nomeadas (base para os dois exercícios).

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

void main() => runApp(const MyApp());

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Atividade Flutter',
      theme: ThemeData(
        brightness: Brightness.dark,
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color(0xFF00BCD4),
          brightness: Brightness.dark,
        ),
        useMaterial3: true,
      ),
      initialRoute: '/',
      routes: {
        '/': (_) => const HomePage(),
        '/detail': (_) => const DetailPage(),
      },
    );
  }
}

// Exemplo de Home/Detail mnimos (substitua pelo contedo dos exerccios)
class HomePage extends StatefulWidget {
  const HomePage({super.key});
  @override
  State<HomePage> createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
```

```
      body: const Center(child: Text('Implementar lista aqui')),
    );
  }
}
class DetailPage extends StatelessWidget {
  const DetailPage({super.key});
  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments; // seu objeto
    return Scaffold(
      appBar: AppBar(title: const Text('Detalhe')),
      body: const Center(child: Text('Detalhes...')),
    );
  }
}
```

2. Exercício 1 (5 pontos) — City Explorer: "Descubra Ariquemes"

Ideia. Mini-app que lista pontos de interesse (Places) de Ariquemes. Ao tocar no card, abrir a tela de detalhes. Implementar favorito, contador de visitas e botões de ação.

Requisitos mínimos (entregar todos):

R1. Rotas nomeadas + parâmetros: usar `Navigator.pushNamed` com `arguments`.

R2. Estado com `setState`: alternar favorito e contador de visitas.

R3. `GestureDetector`: em imagem ou container (ex.: like ao tocar ou expandir).

R4. Layout: `Row`, `Column`, `Padding`, `Container`, `SingleChildScrollView` sem overflow.

R5. Imagens: ao menos uma `Image.network` e uma `Image.asset`.

R6. Pacotes: `google_fonts` (tipografia), `animated_text_kit` (título animado), `shimmer` (skeleton de imagem).

R7. Ação: um `ElevatedButton` que aciona `url_launcher` (desafio) para abrir site/rota no Maps.

Pontuação (5,0):

- Rotas nomeadas + parâmetros funcionando **1,2**
- `setState` (favorito + contador) & `GestureDetector` **1,2**
- Layout limpo (Row/Column/Padding/Container/Scroll) + tema escuro **1,2**
- Imagens (asset + network) com `shimmer` **0,8**
- `google_fonts` + `animated_text_kit` + `ElevatedButton` (`url_launcher` opcional) **0,6**

Trechos úteis (exemplos):

Modelo de dado (opcional)

```
class Place {
  final String title;
  final String coverUrl; // para Image.network
  final String description;
  final String localAssetMap; // ex.: 'assets/images/map.png'
  const Place({
    required this.title,
    required this.coverUrl,
    required this.description,
    required this.localAssetMap,
  });
}
```

Navegação com argumentos

```
Navigator.pushNamed(
  context,
  '/detail',
  arguments: place, // instncia de Place
);
```

Receber argumentos no detalhe

```
final place = ModalRoute.of(context)!.settings.arguments as Place;
```

Shimmer simples para imagem em loading

```
Widget skeleton(double h) {
  return Shimmer.fromColors(
    baseColor: Colors.grey.shade800,
    highlightColor: Colors.grey.shade700,
    child: Container(height: h, color: Colors.grey.shade900),
  );
}
```

AnimatedTextKit no cabeçalho

```
AnimatedTextKit(
  animatedTexts: [
    TypewriterAnimatedText(
      'City Explorer - Ariquemes',
      speed: Duration(milliseconds: 80),
    ),
  ],
  repeatForever: true,
);
```

Abrir link externo (desafio)

```
Future<void> openLink(String url) async {
  final uri = Uri.parse(url);
  await launchUrl(uri, mode: LaunchMode.externalApplication);
}
```

YouTube embutido (desafio)

```
late YoutubePlayerController yt;
@override
void initState(){
  super.initState();
  yt = YoutubePlayerController(
    initialVideoId: YoutubePlayer.convertUrlToId(
      'https://www.youtube.com/watch?v=dQw4w9WgXcQ'
    )!,
    flags: const YoutubePlayerFlags(autoPlay:false),
  );
}
@override
void dispose(){ yt.dispose(); super.dispose(); }
```

3. Exercício 2 (5 pontos) — Pet Rescue: "Adote um Amigo"

> **Ideia.** Lista de pets para adoção (cards com foto, nome, temperamento). Detalhe com perfil, galeria e botão "Quero Adotar".

Requisitos mínimos (entregar todos):

R1. Rotas nomeadas + parâmetros: objeto `Pet` em `arguments`.

R2. Estado com `setState`: marcar interesse e filtrar por porte (`if/else`: pequeno, médio, grande).

R3. `GestureDetector`: alternar a imagem entre `Image.network` e `Image.asset`.

R4. Layout: `Row`, `Column`, `Padding`, `Container`, `SingleChildScrollView`.

R5. Imagens: 1 de rede + 1 asset. Tema escuro aplicado.

R6. Pacotes: `google_fonts`, `animated_text_kit`, `shimmer`. Botão `ElevatedButton`.

R7. Desafio: `url_launcher` para Instagram/WhatsApp da ONG; `youtube_player_flutter` para vídeo institucional.

Pontuação (5,0):

- Rotas nomeadas + parâmetros funcionando **1,2**
- `setState` (interesse + filtro) & `GestureDetector` **1,2**
- Layout limpo + tema escuro **1,2**
- Imagens (asset + network) com `shimmer` **0,8**
- `google_fonts` + `animated_text_kit` + `ElevatedButton` (`url_launcher`/YouTube opcional) **0,6**

Trechos úteis (exemplos):

Modelo de dado

```
class Pet {
  final String name;
  final String photoUrl;
  final String localAssetAlt; // ex.: 'assets/images/pet.png'
  final String size; // 'pequeno' | 'medio' | 'grande'
  final String temperament;
  const Pet({
    required this.name,
    required this.photoUrl,
    required this.localAssetAlt,
    required this.size,
    required this.temperament,
  });
}
```

Filtro por porte (if/else)

```
List<Pet> filterBySize(List<Pet> all, String selected){
  if(selected == 'todos') return all;
  return all.where((p) => p.size == selected).toList();
}
```

Alternar imagem com GestureDetector

```
bool showNetwork = true;
GestureDetector(
  onTap: () => setState(() => showNetwork = !showNetwork),
  child: showNetwork
    ? Image.network(pet.photoUrl, height: 180, fit: BoxFit.cover)
    : Image.asset(pet.localAssetAlt, height: 180, fit: BoxFit.cover),
);
```

4. Estrutura sugerida de pastas

```
lib/
  main.dart
  pages/
    home_page.dart
    detail_page.dart
assets/
  images/
    ari_map.png
    pet_placeholder.png
```

5. Checklist de entrega

- Rotas nomeadas criadas e testadas (`Home` -> `Detail`).
- Passagem de parâmetros funcionando (`arguments`).

- Dois usos de `setState` em interações distintas.
- Um `GestureDetector` sensível ao toque.
- Layout com `Row/Column/Padding/Container/SingleChildScrollView` sem overflow.
- Pelo menos 1 `Image.network` e 1 `Image.asset` por app.
- Tema escuro configurado no `MaterialApp`.
- `google_fonts`, `animated_text_kit`, `shimmer` em uso.
- (Desafio) `url_launcher` e/ou `youtube_player_flutter` funcionando.
- README curto explicando onde cada requisito foi implementado + 2 prints (Home e Detalhe).

6. Dicas importantes

- Use `const` onde possível para melhorar performance.
- Mesmo que não consiga concluir a atividade por completo, tente ir o mais longe possível, irei pontuar seu esforço e dedicação.
- O uso de IA é liberado APENAS para o entendimento do códigos contidos nesse documento e estudos, PROIBIDO o uso para geração de códigos.
- Separe widgets em arquivos (ex.: `PlaceCard`, `PetCard`) para manter o código limpo (agrega nota, mas opcional).
- Antes do desafio (YouTube/links), garanta que as telas e rotas estão 100% funcionais.

Avaliação (resumo). Total 10 pontos: 5 (City Explorer) + 5 (Pet Rescue). O desafio pode fortalecer a nota dentro do limite de 5 pontos de cada exercício.