Roteiro de Aula Prática: Navegação entre Telas e Consumo de APIs com Flutter

Ambiente: DartPad
Duração Estimada: 1h40m

Objetivos de Aprendizagem

- Compreender o conceito de navegação entre telas no Flutter
- Implementar passagem de dados entre telas
- Criar modelos de dados em Dart
- Consumir APIs REST em aplicações Flutter
- Trabalhar com imagens e layouts responsivos

Parte 1: Navegação Básica entre Telas

1.1 Código Base da Navegação

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'App de Navegação',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
```

```dart
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tela Inicial'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => SecondScreen(data: 'Olá da Tela Inicial'),
              ),
            );
          },
          child: Text('Ir para a Segunda Tela'),
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  final String data;

  SecondScreen({required this.data});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Segunda Tela'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              data,
              style: TextStyle(fontSize: 24),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context);
              },
              child: Text('Voltar para a Tela Inicial'),
            ),
          ],
```

```
        ),
      ),
    );
  }
}
```

1.2 Exercício Prático

- Modificar o texto passado entre as telas
- Adicionar um campo de texto na primeira tela para enviar dados personalizados
- Implementar um contador na segunda tela

Parte 2: Lista de Filmes

2.1 Código da Aplicação de Filmes

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

class Movie {
  final String title;
  final String director;
  final String synopsis;

  Movie({
    required this.title,
    required this.director,
    required this.synopsis,
  });
}

class MovieListScreen extends StatelessWidget {
  final List<Movie> movies = [
    Movie(
      title: 'O Poderoso Chefão',
      director: 'Francis Ford Coppola',
      synopsis: 'Uma saga sobre a máfia italiana...',
    ),
    Movie(
      title: 'Forrest Gump',
      director: 'Robert Zemeckis',
      synopsis: 'A história de um homem que vive várias aventuras...',
    ),
    Movie(
      title: 'Interestelar',
```

```
      director: 'Christopher Nolan',
      synopsis: 'Uma jornada épica pelo espaço e tempo...',
    ),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de Filmes'),
      ),
      body: ListView.builder(
        itemCount: movies.length,
        itemBuilder: (context, index) {
          final movie = movies[index];
          return ListTile(
            title: Text(movie.title),
            subtitle: Text(movie.director),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => MovieDetailScreen(movie: movie),
                ),
              );
            },
          );
        },
      ),
    );
  }
}

class MovieDetailScreen extends StatelessWidget {
  final Movie movie;

  MovieDetailScreen({required this.movie});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(movie.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
```

```
            Text(
              'Diretor: ${movie.director}',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 16),
            Text(
              'Sinopse:',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 8),
            Text(
              movie.synopsis,
              style: TextStyle(fontSize: 16),
            ),
          ],
        ),
      ),
    );
  }
}
```

2.2 Exercício Prático

- Adicionar um campo para ano do filme
- Implementar um sistema de classificação (estrelas)
- Adicionar uma imagem do poster do filme

Parte 3: Consumo de API do League of Legends

3.1 Código da Aplicação com Consumo de API

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter/material.dart';

class Champion {
  final String id;
  final String name;
  final String title;
  final String blurb;
  final String imageUrl;

  Champion({
    required this.id,
```

```
    required this.name,
    required this.title,
    required this.blurb,
    required this.imageUrl,
  });

  factory Champion.fromJson(Map<String, dynamic> json) {
    return Champion(
      id: json['id'] ?? '',
      name: json['name'] ?? 'Nome não disponível',
      title: json['title'] ?? 'Título não disponível',
      blurb: json['blurb'] ?? 'Descrição não disponível',
      imageUrl: 'https://ddragon.leagueoflegends.com/cdn/13.7.1/img/champion/${json[
    );
  }
}

class ApiService {
  final String apiUrl = 'https://ddragon.leagueoflegends.com/cdn/13.7.1/data/en_US/c

  Future<List<Champion>> fetchChampions() async {
    final response = await http.get(Uri.parse(apiUrl));

    if (response.statusCode == 200) {
      Map<String, dynamic> data = json.decode(response.body);
      List<dynamic> championsData = data['data'].values.toList();
      return championsData.map((json) => Champion.fromJson(json)).toList();
    } else {
      throw Exception('Falha ao carregar campeões');
    }
  }
}

class ChampionListScreen extends StatefulWidget {
  @override
  _ChampionListScreenState createState() => _ChampionListScreenState();
}

class _ChampionListScreenState extends State<ChampionListScreen> {
  late Future<List<Champion>> futureChampions;
  final ApiService apiService = ApiService();

  @override
  void initState() {
    super.initState();
    futureChampions = apiService.fetchChampions();
  }

  @override
```

```dart
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Lista de Campeões'),
      ),
      body: FutureBuilder<List<Champion>>(
        future: futureChampions,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Erro: ${snapshot.error}'));
          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
            return Center(child: Text('Nenhum campeão disponível'));
          } else {
            final champions = snapshot.data!;
            return ListView.builder(
              itemCount: champions.length,
              itemBuilder: (context, index) {
                final champion = champions[index];
                return ListTile(
                  leading: CircleAvatar(
                    backgroundImage: NetworkImage(champion.imageUrl),
                  ),
                  title: Text(champion.name),
                  subtitle: Text(champion.title),
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => ChampionDetailScreen(champion: champio
                      ),
                    );
                  },
                );
              },
            );
          }
        },
      ),
    );
  }
}

class ChampionDetailScreen extends StatelessWidget {
  final Champion champion;

  ChampionDetailScreen({required this.champion});
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(champion.name),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Center(
              child: Image.network(
                champion.imageUrl,
                width: 150,
                height: 150,
              ),
            ),
            SizedBox(height: 16),
            Text(
              champion.title,
              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 16),
            Text(
              'Descrição:',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 8),
            Text(
              champion.blurb,
              style: TextStyle(fontSize: 16),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.2 Exercício Prático

1. Implementar uma barra de pesquisa para filtrar campeões
2. Adicionar um sistema de favoritos usando SharedPreferences
3. Implementar um grid view como alternativa à lista
4. Adicionar informações adicionais do campeão (função, dificuldade)

Avaliação e Feedback

- Você deverá:
    1. Implementar a navegação básica
    2. Criar e manipular modelos de dados
    3. Consumir a API e tratar erros adequadamente
    4. Implementar interfaces responsivas e atraentes

Dicas Importantes

1. Utilize o hot reload para testar mudanças rapidamente
2. Mantenha o código organizado em arquivos separados
3. Sempre trate os erros possíveis ao consumir APIs
4. Use const widgets quando possível para melhor performance
5. Siga as boas práticas de programação Flutter

Desafios Extras

1. Implementar tema escuro/claro com provider
2. Adicionar animações nas transições
3. Implementar cache de imagens
4. Criar um sistema de busca avançado

Recursos Adicionais

- Documentação oficial do Flutter
- API do League of Legends
- Material Design Guidelines