DGT2812 - Desenvolvimento de Aplicativos Móveis com Flutter

João Guilherme Fernandes Borba - 202208515835

Polo Ipiranga Porto Alegre - RS
Desenv. de Aplicativos Móveis c/ Flutter – 9001 – 7
Link: GitHub
DGT2812 - Desenvolvimento de Aplicativos Móveis com Flutter

Objetivo da Prática

O objetivo central desta prática foi aplicar os conhecimentos fundamentais do desenvolvimento de aplicativos móveis com Flutter para construir a interface de um aplicativo para uma agência de viagens fictícia, a "Explore Mundo". A atividade visou consolidar as seguintes competências:

- Estruturação de um projeto Flutter: Utilizar os widgets essenciais como MaterialApp, Scaffold e AppBar para criar a base de um aplicativo funcional.

- Construção de Layouts Complexos: Empregar widgets de layout como Row e Column para organizar elementos visuais de forma hierárquica e responsiva, seguindo um design pré-definido.

- Modularização de Código: Organizar a interface em componentes reutilizáveis e criar funções auxiliares para otimizar a construção de elementos repetitivos, como os botões de ação.

- Uso de Listas Roláveis: Aplicar o widget ListView para garantir que a interface do usuário seja funcional e

acessível em dispositivos com telas menores, permitindo a rolagem do conteúdo.

- Gerenciamento de Recursos (Assets): Incluir e exibir imagens locais no aplicativo, configurando corretamente o arquivo pubspec.yaml para o gerenciamento de assets.

A implementação do aplicativo seguiu uma abordagem estruturada, "de baixo para cima" (bottom-up), conforme sugerido pelo material de orientação. Essa estratégia consistiu em construir primeiro os componentes menores e mais isolados da interface para, em seguida, integrá-los em uma estrutura maior e coesa. Os passos para alcançar o resultado final foram os seguintes:

1. Configuração Inicial: O projeto foi iniciado com a estrutura básica de um aplicativo Flutter, contendo um MaterialApp e um Scaffold, que serviram como o esqueleto para os demais componentes visuais.

2. Desenvolvimento da Seção de Título: O primeiro componente criado foi a seção de título. Utilizou-se um Row para alinhar os elementos horizontalmente: uma coluna com os textos de título e subtítulo, o ícone de estrela e o número de avaliações. O widget Expanded foi crucial para garantir que a coluna de texto ocupasse todo o espaço disponível, empurrando os outros elementos para a direita, conforme o design.

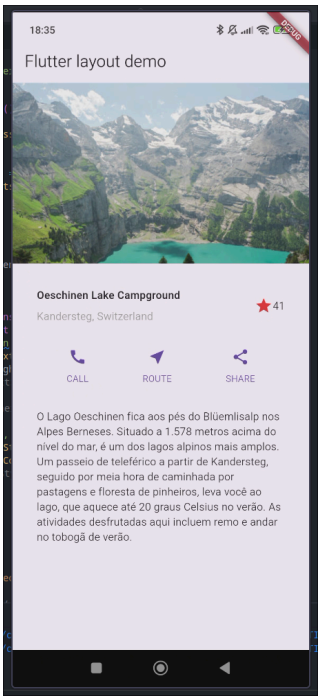
3. Desenvolvimento da Seção de Botões: Para a linha de botões de ação ("CALL", "ROUTE", "SHARE"), foi adotada uma abordagem modular. Uma função auxiliar _buildButtonColumn foi criada para gerar cada botão, que é composto por um Icon e um Text dentro de uma Column.

Isso evitou a repetição de código e tornou a construção da Row principal mais limpa e legível.

4. Desenvolvimento da Seção de Texto: O bloco de texto descritivo foi implementado dentro de um Container para aplicar espaçamento (padding) e utilizou o widget Text com a propriedade softWrap habilitada, garantindo que o texto quebrasse a linha automaticamente, adaptando-se à largura da tela.

5. Integração e Montagem Final: Com todas as seções construídas como variáveis de widget separadas (titleSection, buttonSection, textSection), o passo final foi integrá-las. Inicialmente, foi usado um Column, mas, seguindo a orientação final da prática, o widget foi substituído por um ListView. Essa mudança foi fundamental para garantir que o layout não "quebrasse" em telas menores, adicionando a funcionalidade de rolagem vertical. Por fim, a imagem de destaque do destino foi adicionada no topo da ListView, completando a interface visual do aplicativo.

Print 1: Tela Final do Aplicativo



Descrição: A imagem exibe a tela final do aplicativo "Explore Mundo". Nela, todos os componentes desenvolvidos estão integrados e visíveis: a imagem de destaque no topo, seguida pela seção de título com o nome do local e as avaliações, a seção de botões de ação e, por fim, o bloco de texto descritivo. O layout está organizado verticalmente dentro de uma ListView, como proposto pelo roteiro da prática.

Print 2: Estrutura Principal com ListView

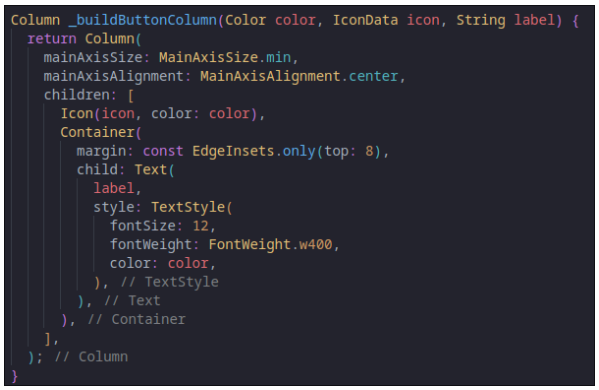
```
@override
Widget build(BuildContext context) {
  Color color = Theme.of(context).primaryColor;

  Widget buttonSection = Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      _buildButtonColumn(color, Icons.call, 'CALL'),
      _buildButtonColumn(color, Icons.near_me, 'ROUTE'),
      _buildButtonColumn(color, Icons.share, 'SHARE'),
    ],
  );

  return MaterialApp(
    title: 'Flutter layout demo',
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Flutter layout demo'),
      ), // AppBar
      body: ListView(
        children: [
          Image.asset(
            'images/lake.jpg',
            width: 600,
            height: 240,
            fit: BoxFit.cover,
          ), // Image.asset
          titleSection,
          buttonSection,
          textSection,
        ],
      ), // ListView
    ), // Scaffold
  ); // MaterialApp
}
```

Descrição: Este trecho de código demonstra a montagem final da interface dentro do método build. O widget ListView é utilizado como o corpo (body) do Scaffold e seus filhos (children) são as variáveis que representam cada seção criada anteriormente (Image.asset, titleSection, buttonSection e textSection). Essa abordagem modular torna o código do layout principal limpo, organizado e fácil de entender.

Print 3: Código da Função Auxiliar de Botões



```
Column _buildButtonColumn(Color color, IconData icon, String label) {
  return Column(
    mainAxisSize: MainAxisSize.min,
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Icon(icon, color: color),
      Container(
        margin: const EdgeInsets.only(top: 8),
        child: Text(
          label,
          style: TextStyle(
            fontSize: 12,
            fontWeight: FontWeight.w400,
            color: color,
          ), // TextStyle
        ), // Text
      ), // Container
    ],
  ); // Column
}
```

Descrição: O código acima mostra a função auxiliar _buildButtonColumn, criada para otimizar a construção da seção de botões. A função recebe uma cor, um ícone e um rótulo como parâmetros e retorna um widget Column completamente estilizado. Essa técnica foi essencial para evitar a repetição de código, demonstrando uma boa prática de desenvolvimento e mantendo a seção de botões organizada e de fácil manutenção.


Estácio


Estácio