Tutorial 4: Gerenciamento de Estado no Flutter

Neste tutorial, vamos criar uma aplicação Flutter que utiliza gerenciamento de estado para uma lista de tarefas (to-do list) utilizando controles de estado da aplicação, como `setState()` e `StatefulWidget`.

Estrutura do Projeto

1. **Configuração do Ambiente**
2. **Criação do projeto Flutter**
3. **Criação da estrutura básica do app**
4. **Implementação do gerenciamento de estado**
5. **Conclusão e execução**

1. Configuração do Ambiente

Antes de começar, certifique-se de que você tenha o Flutter instalado em seu sistema. Para verificar, abra o terminal do Visual Studio Code (Ctrl+Shift+') e execute:

```
flutter doctor
```

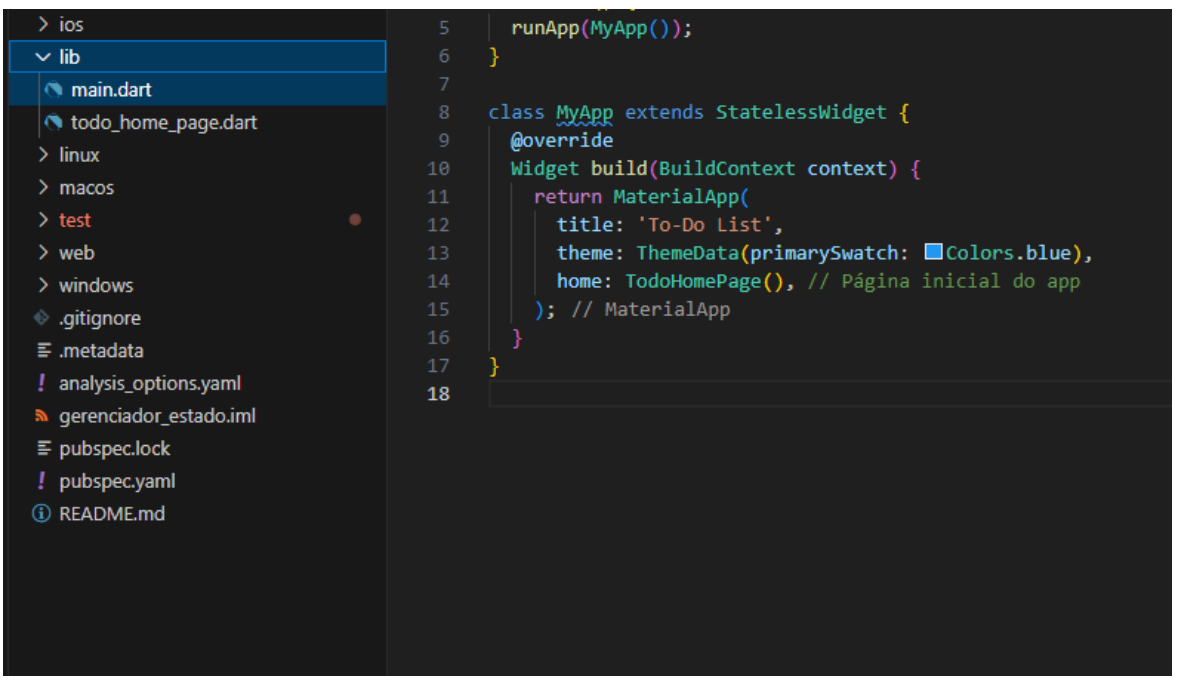
2. Criação do projeto Flutter

Crie um novo projeto Flutter utilizando o terminal:

```
flutter create todo_app
cd todo_app
```

3. Criação da estrutura básica do app

No diretório `lib`, substitua o conteúdo do `main.dart` pelo código:



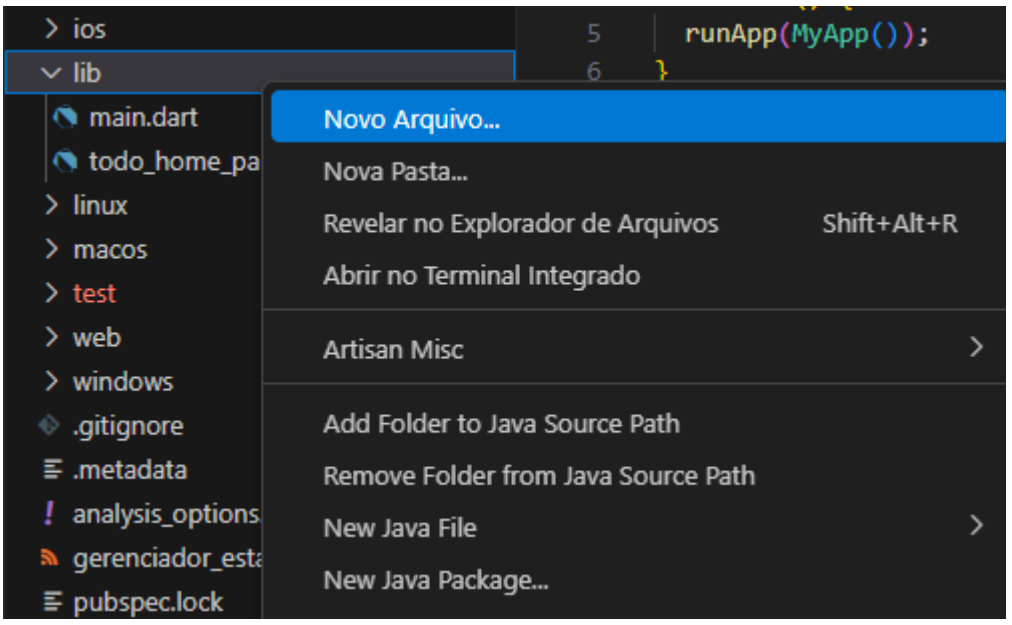

```
import 'package:flutter/material.dart';
import 'package:gerenciador_estado/todo_home_page.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'To-Do List',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: TodoHomePage(), // Página inicial do app
    );
  }
}
```

4. Implementação do gerenciamento de estado

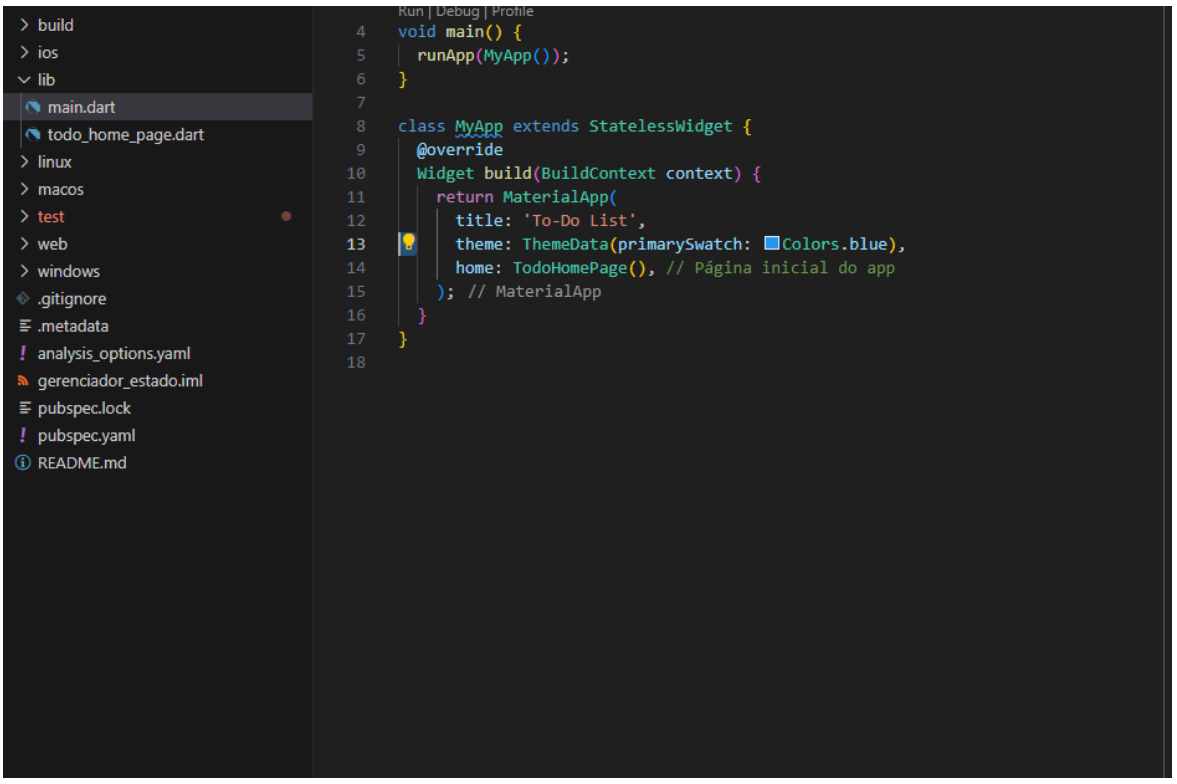
Agora, vamos construir a interface do aplicativo no arquivo `todo_home_page.dart`. Crie um novo arquivo com este nome e adicione o seguinte código:

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

class TodoHomePage extends StatefulWidget {
  @override
  _TodoHomePageState createState() => _TodoHomePageState();
}

class _TodoHomePageState extends State<TodoHomePage> {
  List<String> _tasks = []; // Lista de tarefas
  final TextEditingController _controller =
      TextEditingController(); // Controlador para o campo de texto

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('To-Do List'), // Título da barra de navegação
      ),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              itemCount: _tasks.length, // Número de tarefas
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(_tasks[index]), // Tarefa exibida
                  trailing: IconButton(
                    icon: Icon(Icons.delete), // Botão para remover a
                                              tarefa
                    onPressed: () {
                      setState(() {
                        _tasks.removeAt(
                            index); // Remove a tarefa ao pressionar o
                                    botão
                      });
                    },
                  ),
                );
              },
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0), // Espaçamento
            child: Row(
              children: [
                Expanded(
                  child: TextField(
                    controller: _controller, // Controlador do campo de
                                             texto
                    decoration: InputDecoration(
                        labelText: 'Nova Tarefa'), // Rótulo do campo
                  ),
                ),
                IconButton(
                  icon: Icon(Icons.add), // Botão para adicionar nova
                                         tarefa
                  onPressed: () {
                    if (_controller.text.isNotEmpty) {
                      setState(() {
                        _tasks.add(_controller.text); // Adiciona tarefa
                      });
                      _controller.clear(); // Limpa o campo de texto
                    }
                  },
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Você deve criar seu arquivo assim, dando um clique direito na pasta `lib` e criando o arquivo.



Sua estrutura de pastas e arquivos deve estar assim:



5. Conclusão e execução

Agora que o código está completo, você pode executar a aplicação:

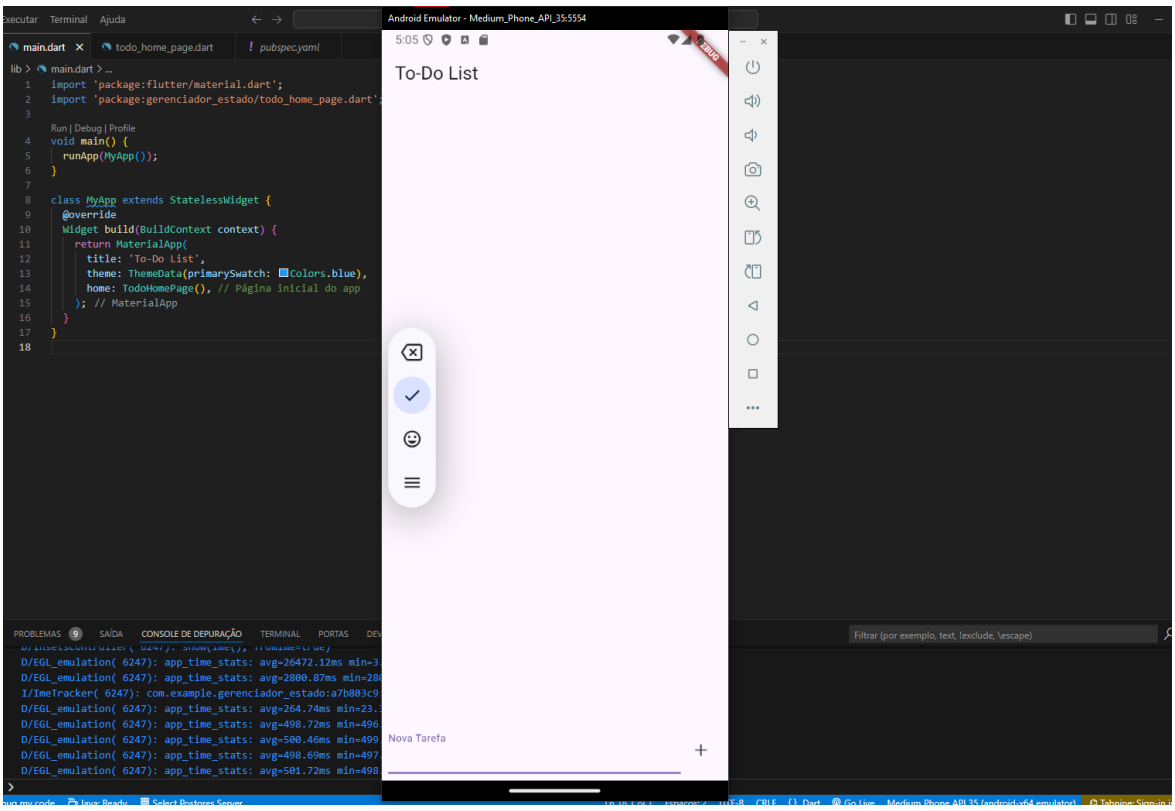
```
flutter run
```

Comentários sobre o Código

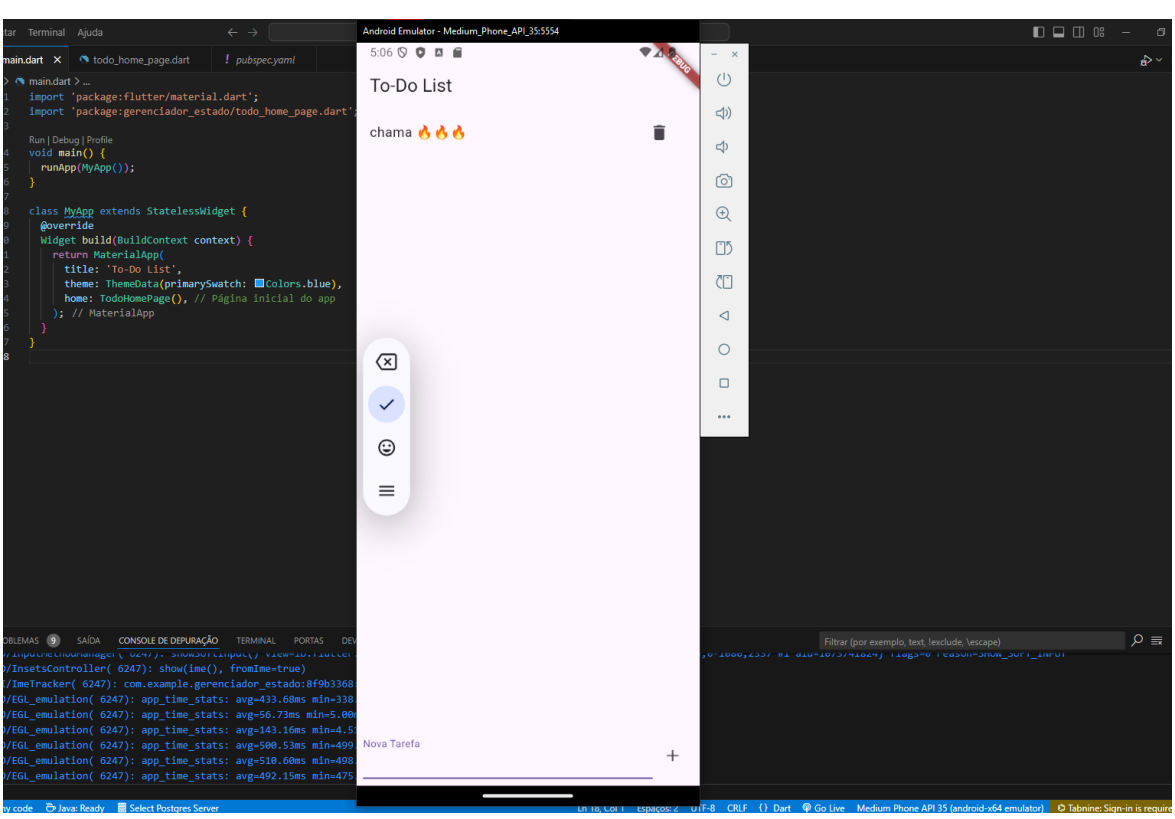
- **ChangeNotifier**: A classe `TodoList` estende `ChangeNotifier`, permitindo que ela notifique os ouvintes sobre alterações.
- **Provider**: O `Provider` gerencia a instância de `TodoList`, tornando-a disponível na árvore de widgets.
- **Consumer**: O `Consumer` escuta alterações na lista de tarefas e atualiza a interface automaticamente.
- **TextEditingController**: Usamos um controlador para gerenciar o texto do campo de entrada.

Testes e execução

Ao iniciar a aplicação, ela estará assim:



Adicione uma tarefa. Isso vai atualizar o estado da aplicação com o `SetState()` e atualizar sua lista automaticamente.



Conclusão

Neste tutorial, aprendemos a criar uma aplicação simples de lista de tarefas em Flutter, utilizando o gerenciamento de estado com `setState()`. Através de um `StatefulWidget`, conseguimos adicionar e remover tarefas, refletindo as mudanças na interface do usuário de forma clara e eficaz.

Considerações Finais

- **Flexibilidade**: Usar `setState()` é uma abordagem direta e fácil de entender, especialmente para pequenos projetos.
- **Escalabilidade**: Para aplicações maiores, considere explorar soluções de gerenciamento de estado mais robustas, como `Provider`, `Riverpod` ou `Bloc`, que ajudam a manter o código organizado e a lógica de negócios separada da interface.

Próximos Passos

- **Melhorias**: Você pode aprimorar a aplicação adicionando funcionalidades como a persistência de dados usando `SharedPreferences` ou `SQLite`.
- **Interface do Usuário**: Considere melhorar a UI com temas personalizados ou animações para tornar a experiência do usuário mais atraente.