Flutter Adventure:

Programação descomplicada Para Jovens

E-Book destinado ao desmitificar o conhecimento básico para a linguagem de programação

FLUTTER, linguagem desenvolvida pela gigante Google, para criação de Aplicativos Android e IOs

LUCK
GARCIA

info@

Adventure: Instale, Codifique, Jogue.

Bem-vindo ao "Flutter Adventure: Instale, Codifique, Jogue"!
Neste eBook, vamos guiá-lo passo a passo na criação de um Jogo
da Velha simples usando Flutter. Abordaremos desde a instalação
do ambiente de desenvolvimento até a implementação do jogo.
Vamos dividir o conteúdo em capítulos para facilitar o aprendizado..

01

Preparando o Ambiente de Desenvolvimento.

01 Start

Instalando o Flutter e o VSCode.

Passo 1: Instalar o Flutter

Baixe o Flutter SDK:

- Acesse flutter.dev e baixe o SDK para o seu sistema operacional.
- Extraia o arquivo para uma pasta de sua preferência.

Adicione o Flutter ao Path:

- No Windows, adicione flutter/bin ao PATH nas Variáveis de Ambiente.
- No macOS/Linux, adicione export PATH="$PATH:/path/to/flutter/bin" ao arquivo .bashrc ou .zshrc.

Verifique a Instalação:

- Abra o terminal e execute flutter doctor.
- Siga as instruções para resolver qualquer problema encontrado.

01 / Steps

Instalando o Flutter e o VSCode.

Passo 2: Instalar o Visual Studio Code

Baixe o VSCode:

- Acesse code.visualstudio.com e instale o VSCode para o seu sistema
- operacional.

Instale a Extensão Flutter:

- Abra o VSCode.
- Vá para Extensões (ou pressione Ctrl+Shift+X).
- Pesquise por "Flutter" e instale as extensões Flutter e Dart.

Criar o Projeto:

- Abra o terminal.
- Execute ‘flutter create jogo_da_velha’.
- Navegue até a pasta do projeto com ‘cd jogo_da_velha’.

Abrir o Projeto no VSCode:

- Abra o VSCode.
- Vá em File > Open Folder e selecione a pasta do projeto.

02

Estrutura do Projeto e Configuração Inicial.

02 Next

Entendendo a Estrutura do Projeto.

O Flutter cria uma estrutura de
pastas com arquivos importantes.
Aqui estão os principais:

- lib/main.dart: Ponto de entrada do aplicativo.
- pubspec.yaml: Arquivo de configuração do projeto, onde gerenciamos dependências

Configurando o Main.dart.

Vamos começar configurando nosso ponto de entrada no arquivo main.dart. Crie o arquivo main.dart na pasta lib e adicione o seguinte código:

main.dart --> iniciando

Explicar

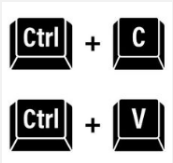
```dart
import 'package:flutter/material.dart';
import 'home_screen.dart';

void main() => runApp(JogoDaVelhaApp());

class JogoDaVelhaApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Jogo da Velha',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomeScreen(),
    );
  }
}
```

Dica 1

O que o código faz:.

O que o código faz:
Este é o ponto de entrada da aplicação Flutter. Aqui, inicializamos a aplicação e configuramos a tela inicial.

Por que:
A função main é necessária para iniciar a execução do aplicativo.
A classe JogoDaVelhaApp define o tema e a tela inicial da aplicação.

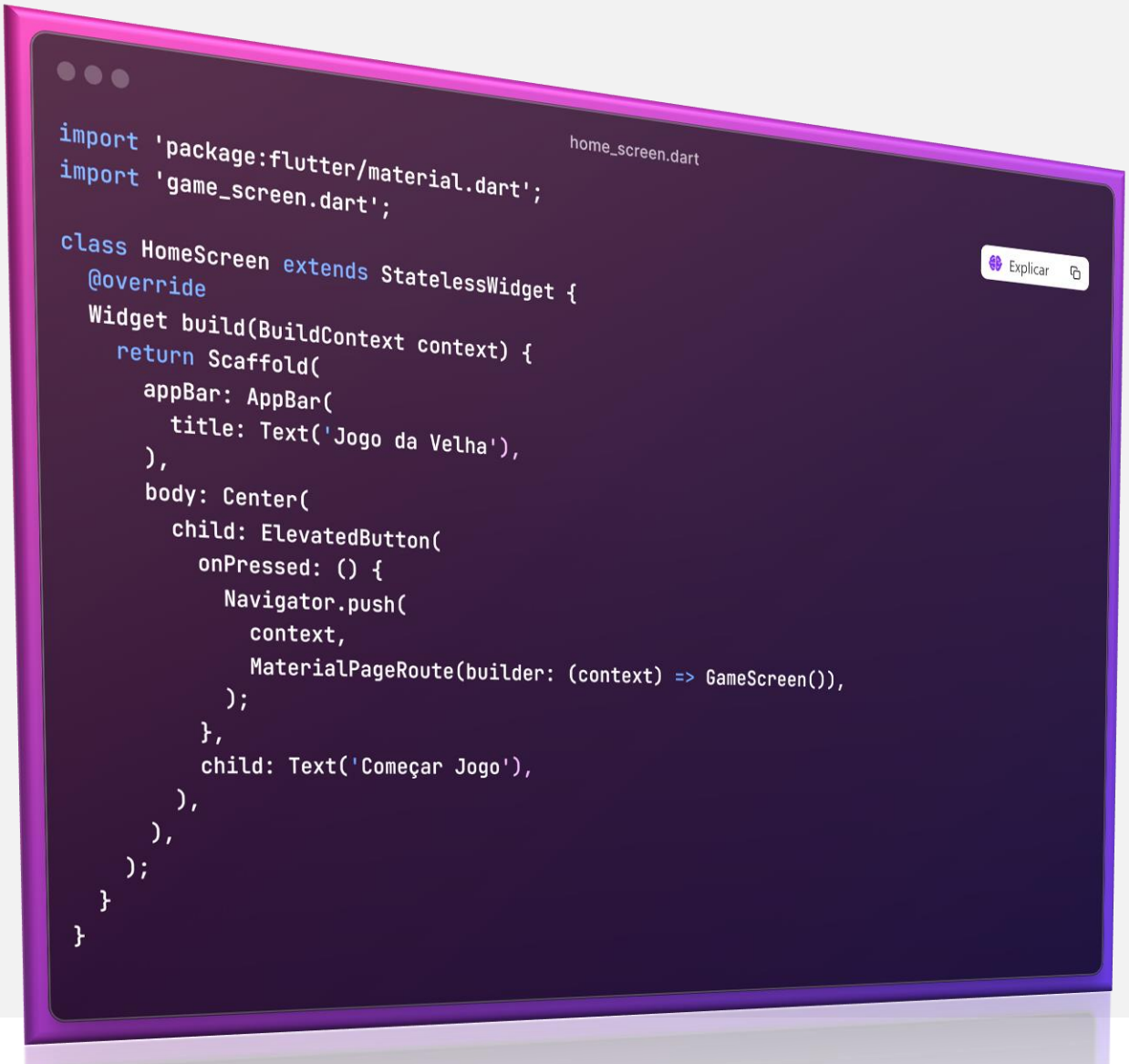



```
import 'package:flutter/material.dart'; // Importa o pacote Flutter Material
import 'home_screen.dart'; // Importa a tela inicial

void main() => runApp(JogoDaVelhaApp()); // Ponto de entrada da aplicação

class JogoDaVelhaApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Jogo da Velha', // Título da aplicação
      theme: ThemeData(
        primarySwatch: Colors.blue, // Define o tema da aplicação
      ),
      home: HomeScreen(), // Define a tela inicial da aplicação
    );
  }
}
```

02 e meio

Arquivo home_screen.dart

Crie o arquivo home_screen.dart na pasta lib e adicione o seguinte código:

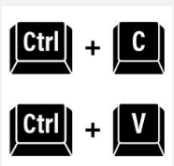


Dica 2

Arquivo home_screen.dart

O que o código faz:
Esta tela inicial exibe um botão que, ao ser pressionado, navega para a tela do jogo.

Por que:
Ter uma tela inicial melhora a experiência do usuário, fornecendo um ponto de partida para o jogo.




```
import 'package:flutter/material.dart'; // Importa o pacote Flutter Material
import 'game_screen.dart'; // Importa a tela do jogo

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Jogo da Velha'), // Título da barra de navegação
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => GameScreen()), // Navega para a tela do jogo
            );
          },
          child: Text('Começar Jogo'), // Texto do botão
        ),
      ),
    );
  }
}
```

Curiosidades!!!

Sobre o Flutter

**Flutter começou como um projeto interno no Google e rapidamente se tornou uma das ferramentas mais populares para o desenvolvimento
de aplicativos. Sua capacidade de criar experiências nativas de alta qualidade a partir de um único código base, combinada com uma comunidade ativa e um ecossistema em expansão, garante que o Flutter continuará a ser uma escolha popular para desenvolvedores no futuro.**

03

Implementando o Jogo da Velha.

03 DOWN

Criando o Tabuleiro

Crie o arquivo ‘game_screen.dart’ na pasta lib e adicione o seguinte código:

game_screen.dart --> Inicio

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

class GameScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Jogo da Velha'),
      ),
      body: Center(
        child: Tabuleiro(),
      ),
    );
  }
}
```

Continuando com ‘game_screen.dart’ crie as duas classe de tabuleiro:

game_screen.dart --> classes do tabuleiro

```dart
class Tabuleiro extends StatefulWidget {
  @override
  _TabuleiroState createState() => _TabuleiroState();
}

class _TabuleiroState extends State<Tabuleiro> {
  List<List<String>> _tabuleiro;
  String _jogadorAtual;

  @override
  void initState() {
    super.initState();
    _resetarJogo();
  }

  void _resetarJogo() {
    _tabuleiro = List.generate(3, (_) => List.filled(3, ''));
    _jogadorAtual = 'X';
  }
```

03 e meio

Ainda Tabuleiro

Atenção para não se perder

03 < 04

QUASE LÁ

game_screen.dart --> terminando .....

Explicar

```dart
void _mostrarDialogoVitoria(String vencedor) {
  showDialog(
    context: context,
    builder: (_) => AlertDialog(
      title: Text('Vitória!'),
      content: Text('$vencedor venceu o jogo!'),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.of(context).pop();
            _resetarJogo();
          },
          child: Text('Jogar Novamente'),
        ),
      ],
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: List.generate(3, (linha) {
      return Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: List.generate(3, (coluna) {
          return GestureDetector(
            onTap: () => _jogar(linha, coluna),
            child: Container(
              width: 100,
              height: 100,
              decoration: BoxDecoration(
                border: Border.all(),
              ),
              child: Center(
                child: Text(
                  _tabuleiro[linha][coluna],
                  style: TextStyle(fontSize: 32),
                ),
              ),
            ),
          );
        }),
      );
    }),
  );
}
}
```

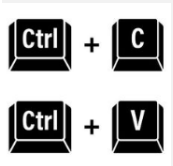
Eita!!!!, código grande

Dica 3

Arquivo game_screen.dart

O que o código faz:
Esta é a tela principal do jogo, onde o tabuleiro do Jogo da Velha é exibido e os jogadores podem interagir.

Por que:
O game_screen.dart contém toda a lógica do jogo, desde a exibição do tabuleiro até a verificação do vencedor.

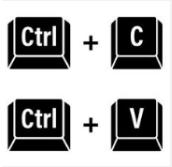



```
import 'package:flutter/material.dart'; // Importa o pacote Flutter Material

class GameScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Jogo da Velha'), // Título da barra de navegação
      ),
      body: Center(
        child: Tabuleiro(), // Exibe o tabuleiro do jogo
      ),
    );
  }
}

class Tabuleiro extends StatefulWidget {
  @override
  _TabuleiroState createState() => _TabuleiroState();
}
```

Continuando

Arquivo game_screen.dart

Continuando a parte do copia & cola, da pagina anterior.

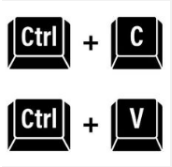



```
class _TabuleiroState extends State<Tabuleiro> {
 List<List<String>> _tabuleiro; // Armazena o estado do tabuleiro
 String _jogadorAtual; // Armazena o jogador atual (X ou O)

 @override
 void initState() {
  super.initState();
  _resetarJogo(); // Inicializa o jogo
 }

 void _resetarJogo() {
  // Cria um tabuleiro 3x3 vazio
  _tabuleiro = List.generate(3, (_) => List.filled(3, ''));
  _jogadorAtual = 'X'; // Define o jogador inicial como 'X'
 }

 void _jogar(int linha, int coluna) {
  // Verifica se a posição está vazia
  if (_tabuleiro[linha][coluna] == '') {
   setState(() {
    _tabuleiro[linha][coluna] = _jogadorAtual; // Marca a posição com o jogador atual
    // Verifica se o jogador atual venceu
    if (_verificarVencedor(linha, coluna)) {
     _mostrarDialogoVitoria(_jogadorAtual);
    } else {
     // Alterna o jogador
     _jogadorAtual = _jogadorAtual == 'X' ? 'O' : 'X';
    }
   });
  }
 }
bool _verificarVencedor(int linha, int coluna) {
  // Verifica se uma linha está completa
  if (_tabuleiro[linha].every((marcador) => marcador == _jogadorAtual)) {
   return true;
  }
  // Verifica se uma coluna está completa
  if (_tabuleiro.every((linha) => linha[coluna] == _jogadorAtual)) {
   return true;
  }
  // Verifica se a diagonal principal está completa
  if (_tabuleiro[0][0] == _jogadorAtual &&
     _tabuleiro[1][1] == _jogadorAtual &&
     _tabuleiro[2][2] == _jogadorAtual) {
   return true;
  }
```

Continuando

Arquivo game_screen.dart

Continuando a parte do copia & cola, da pagina anterior.

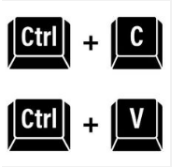



```
// Verifica se a diagonal secundária está completa
   if (_tabuleiro[0][2] == _jogadorAtual &&
       _tabuleiro[1][1] == _jogadorAtual &&
       _tabuleiro[2][0] == _jogadorAtual) {
     return true;
   }
   return false;
 }

 void _mostrarDialogoVitoria(String vencedor) {
   showDialog(
     context: context,
     builder: (_) => AlertDialog(
       title: Text('Vitória!'), // Título do diálogo
       content: Text('$vencedor venceu o jogo!'), // Conteúdo do diálogo
       actions: [
         TextButton(
           onPressed: () {
             Navigator.of(context).pop();
             _resetarJogo(); // Reseta o jogo quando o botão é pressionado
           },
           child: Text('Jogar Novamente'), // Texto do botão
         ),
       ],
     ),
   );
 }
```

Continuando

Arquivo game_screen.dart

Continuando a parte do copia & cola, da pagina anterior.




```
@override
 Widget build(BuildContext context) {
  return Column(
   mainAxisAlignment: MainAxisAlignment.center,
   children: List.generate(3, (linha) {
    return Row(
     mainAxisAlignment: MainAxisAlignment.center,
     children: List.generate(3, (coluna) {
      return GestureDetector(
       onTap: () => _jogar(linha, coluna), // Marca a posição quando pressionada
       child: Container(
        width: 100,
        height: 100,
        decoration: BoxDecoration(
         border: Border.all(), // Adiciona borda ao quadrado
        ),
        child: Center(
         child: Text(
          _tabuleiro[linha][coluna], // Exibe 'X' ou 'O'
          style: TextStyle(fontSize: 32), // Define o estilo do texto
         ),
        ),
       ),
      );
     }),
    );
   }),
  );
 }
}
```

04

Conclusão – Game Over.

04 JOGUE

O Game.

Estrutura Final do Jogo

CONCLUSÃO

Parabéns!
Agora você sabe como criar um Jogo da Velha simples em Flutter. Este eBook cobriu desde a instalação do Flutter e do VSCode até a criação de um jogo funcional. Continue explorando e aprimorando suas habilidades em Flutter.
Boa codificação!

Sobre o autor deste E-Book transcrevido por IA e Diagramado por:

Luciano Garcia

FLUTTER

Adventure: Instale, Codifique, Jogue.