4. Comandos do Flutter para Criação de Aplicativos e Possibilidades(2h)

Os comandos do Flutter para criar e configurar aplicativos oferecem flexibilidade para iniciar novos projetos com diferentes objetivos e configurações. Eles ajudam a definir o ambiente inicial do projeto e permitem personalizar o suporte para plataformas como Web, iOS, Android, Windows, macOS, e Linux.

Comando de Criação de Aplicativos

O principal comando para criar aplicativos no Flutter é o **`flutter create`**.

Exemplo Básico:

```
flutter create meu_app
```

Esse comando cria uma estrutura básica de projeto com pastas e arquivos essenciais, incluindo:

- **lib/main.dart:** O ponto de entrada da aplicação.
- **pubspec.yaml:** Gerenciamento de dependências e configurações do projeto.
- **android/** e **ios/**: Arquivos nativos para Android e iOS.
- **web/**: Suporte para aplicativos web (se ativado).
- **macos/ win/ linux/:** Arquivos nativos para Windows, MacOs e Linux;
- **test/**: Arquivos de teste para garantir a qualidade do código.

Customizando a Criação de Aplicativos

1. **Habilitar Suporte a Plataformas Específicas:**

Se quiser criar um projeto somente para uma plataforma específica (exemplo: Web ou Windows), utilize o parâmetro `--platforms`:

```
flutter create --platforms=android meu_app_web
```

   - Plataformas suportadas: `android`, `ios`, `web`, `windows`, `macos`, `linux`.

2. **Definir um Pacote ou Módulo:**

Para criar um módulo reutilizável (como uma biblioteca que será usada em outros aplicativos):

```
flutter create --template=module meu_modulo
```

Para criar um pacote (útil para bibliotecas ou ferramentas sem interface gráfica):

```
flutter create --template=package meu_pacote
```

3. **Configuração Inicial de Linguagem:**

Por padrão, o Flutter cria projetos para Android com Kotlin e para iOS com Swift. É possível forçar o uso de Java ou Objective-C:

```
flutter create --android-language java --ios-language objc meu_app
```

O que São Dependências no Flutter?

Dependências no Flutter são pacotes ou bibliotecas que você pode adicionar ao seu projeto para expandir suas funcionalidades. Elas podem incluir:

- **Bibliotecas de Interface de Usuário:** Para criar componentes prontos, como `fluttertoast` ou `carousel_slider`.
- **APIs e Comunicação com Back-end:** Como `http` para chamadas REST ou `firebase_core` para integrar Firebase.
- **Utilitários Diversos:** Como `intl` para formatação de datas e números.

O Flutter usa o **Pub** (Package Manager do Dart) para gerenciar essas dependências.

Como Instalar Dependências no Flutter

As dependências são declaradas e gerenciadas no arquivo `pubspec.yaml`, que é o coração do gerenciamento de pacotes no Flutter.

Passo a Passo para Instalar Dependências

1. **Abrir o Arquivo `pubspec.yaml`:**
    - Esse arquivo contém todas as informações de dependências e configurações do projeto.
2. **Adicionar a Dependência:**

Na seção `dependencies`, adicione o nome do pacote e a versão desejada:

```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.15.0
  provider: ^6.0.5
```

Ou

use o comando

flutter pub add nome_da_dependencia

3. **Baixar as Dependências:**

Depois de adicionar a dependência, execute o comando:

```
flutter pub get
```

- Isso baixa as dependências e as registra no arquivo `pubspec.lock`, garantindo que as versões instaladas sejam consistentes.

Atualizar Dependências Existentes

Para atualizar para a versão mais recente permitida pelas restrições do `pubspec.yaml`, use:

```
flutter pub upgrade
```

Se quiser verificar dependências desatualizadas:

```
flutter pub outdated
```

Outras Possibilidades e Configurações no Gerenciamento de Dependências

1. **Versão Fixa ou Sem Restrições:**

Para usar uma versão específica, defina assim:

```
dependencies:
  provider: 6.0.5
```

        ○

Para sempre usar a versão mais recente:

```
dependencies:
  provider: any
```

2. **Adicionar Dependências Diretas de Repositórios Git:**

   Para usar um pacote diretamente do GitHub:

   ```
   dependencies:
     meu_pacote_personalizado:
       git:
         url: https://github.com/seu_usuario/repositorio.git
         ref: main
   ```

3. **Adicionar Dependências Locais:**

   Se você tem um pacote no seu computador, pode referenciá-lo:

   ```
   dependencies:
     meu_pacote_local:
       path: ../caminho/do/pacote
   ```

Exemplo Prático

Vamos supor que queremos adicionar uma funcionalidade de exibição de mensagens (Toast) no Flutter usando o pacote `fluttertoast`.

Adicionar ao `pubspec.yaml`:

```
dependencies:
  fluttertoast: ^8.2.1
```

ou

flutter pub add fluttertoast

Baixar as Dependências:

```
flutter pub get
```

Usar no Código: No arquivo `main.dart`:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Exemplo com Fluttertoast')),
        body: Center(
          child: ElevatedButton(
           onPressed: () {
              Fluttertoast.showToast(
                msg: "Olá, Mundo!",
                toastLength: Toast.LENGTH_SHORT,
                gravity: ToastGravity.CENTER,
              );
```

```
            },
            child: Text("Mostrar Mensagem"),
          ),
        ),
      ),
    );
  }
}
```