



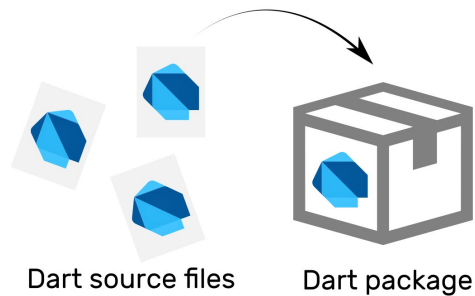
Flutter Avançado - Aula 1

Exercícios 1 a 3



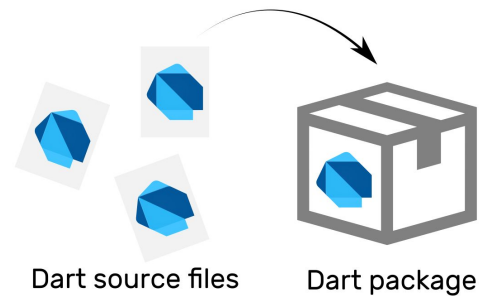
- 24 apps famosos criados com Flutter [Ver mais](#)

```
dependencies:  
  flutter:  
    sdk: flutter  
  intl: ^0.17.0  
  bubble: ^1.2.1  
  chips_choice: ^2.0.1  
  flutter_rating: ^0.0.2  
  image_picker: ^0.7.3  
  http: ^0.13.0  
  json_annotation: ^4.0.1  
  requests: ^3.3.0  
  fluttertoast: ^8.0.3  
  random_string: ^2.1.0  
  firebase_auth_web: ^1.0.3  
  firebase_core: ^1.0.2  
  cloud_firestore: ^1.0.3  
  firebase_storage: ^8.0.1  
  firebase_storage_web: ^1.0.0  
  firebase_core_platform_interface: ^4.0.0  
  firebase_auth: ^1.0.1  
  page_transition: ^1.1.7+6  
  flutter_spinkit: ^5.0.0  
  geolocator: ^7.0.1  
  google_maps_flutter: ^2.0.1  
  flutter_google_places: ^0.2.8  
  location: ^4.1.1  
  geocoder: ^0.2.1  
  modal_progress_hud: ^0.1.3
```

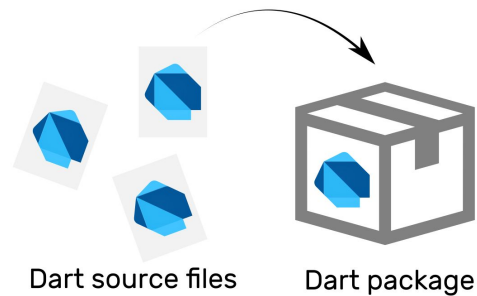




- Um pacote Dart é um diretório contendo um arquivo pubspec.yaml
- Esse pacote também pode conter dependências
- Pode conter bibliotecas Dart
- Apps, Resources, Testes, Imagens, Fontes, Exemplos...

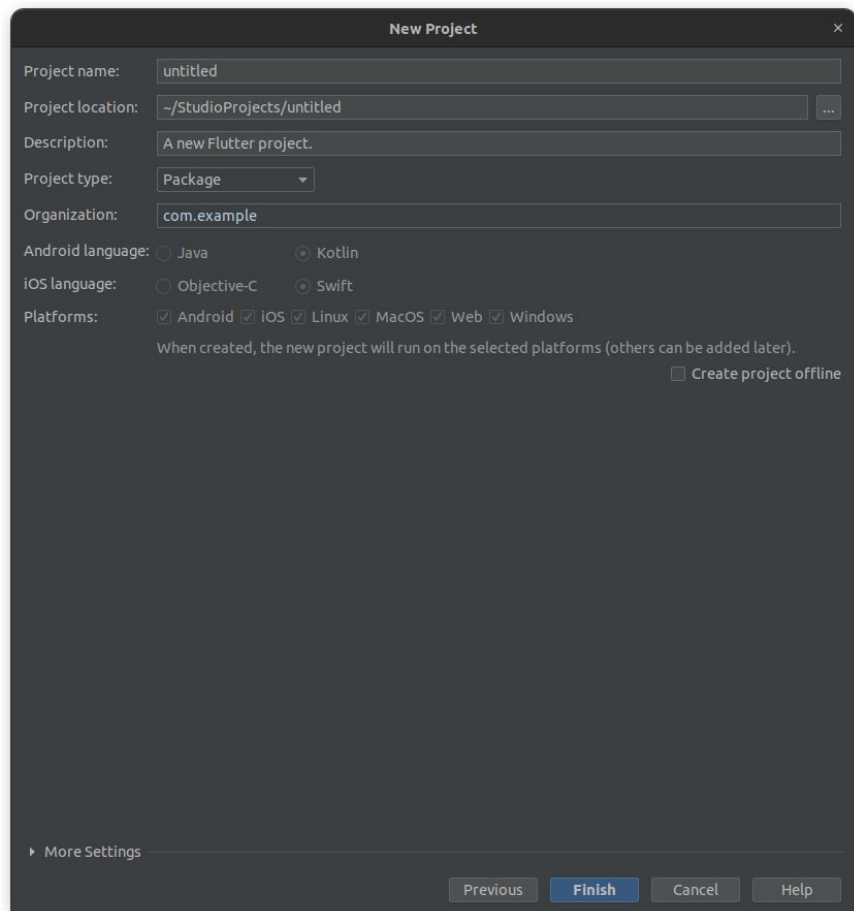


- Um *plugin* é um pacote
- A única diferença entre um pacote e um plugin é que os pacotes possuem somente código Dart
- Plugins possuem código nativo
- Plugins são utilizados para integrações com a plataforma nativa
- Um exemplo de plugin é o `url_launcher`



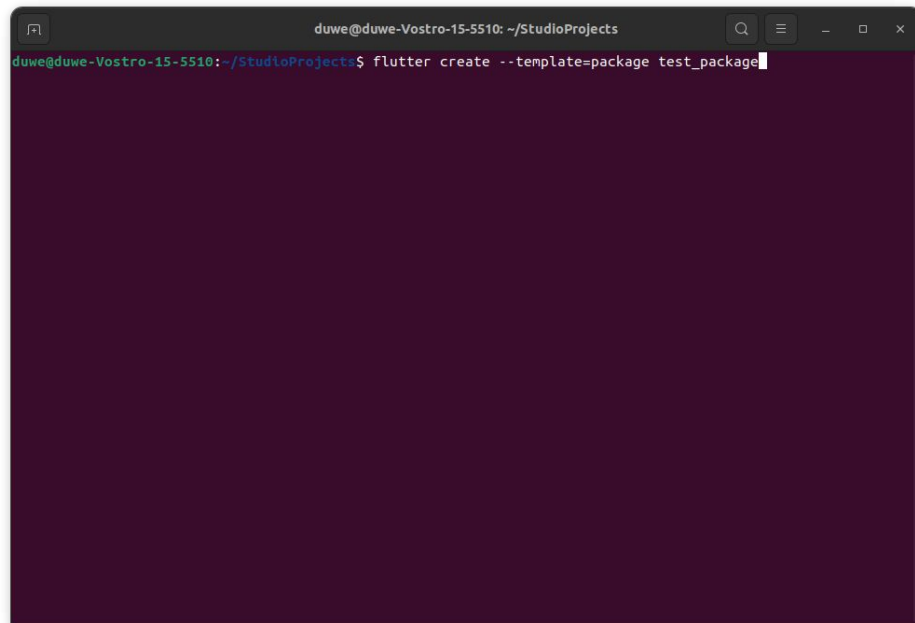
Criando uma nova package

[Ver mais](#)



The 'New Project' dialog box in Flutter IDE. It contains the following fields and options:

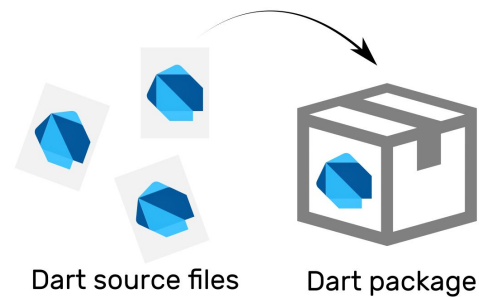
- Project name:
- Project location:
- Description:
- Project type:
- Organization:
- Android language: ☐ Java ☒ Kotlin
- iOS language: ☐ Objective-C ☒ Swift
- Platforms: ☒ Android ☒ iOS ☒ Linux ☒ MacOS ☒ Web ☒ Windows
- When created, the new project will run on the selected platforms (others can be added later).
- ☐ Create project offline
- Bottom bar:
- Link: [More Settings](#)



```
duwe@duwe-Vostro-15-5510: ~/StudioProjects
duwe@duwe-Vostro-15-5510:~/StudioProjects$ flutter create --template=package test_package
```

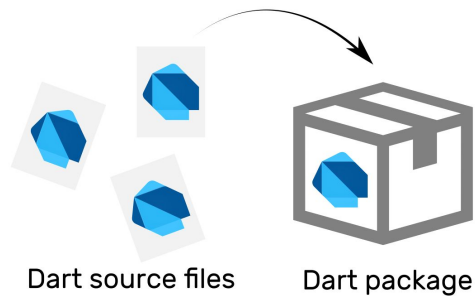


- Arquivo pubspec.yaml
 - Nome
 - Descrição
 - versão
 - homepage
- Pasta lib



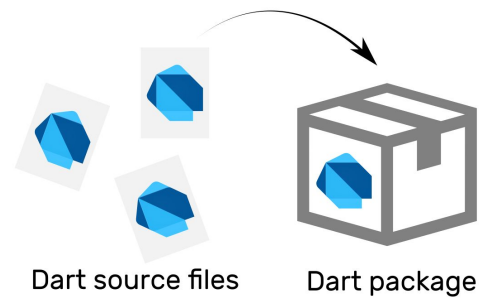


- O código criado dentro de uma package é muito semelhante aos códigos que já escrevemos nas nossas aplicações
- Podemos criar *models*, *repositories*, *widgets*, etc
- Adicionar assets
- Escrever testes





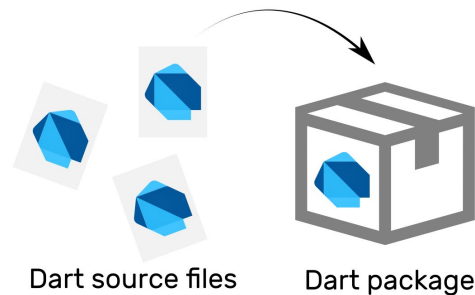
- O código criado





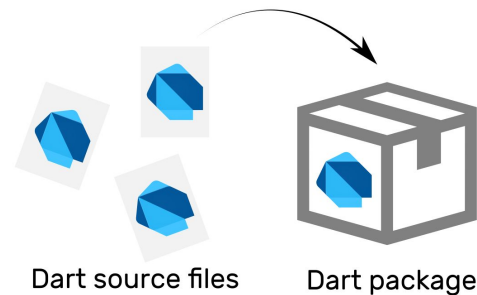
- Dentro da nossa aplicação, podemos criar uma aplicação para testar o pacote que estamos escrevendo
- Dentro da nossa package, podemos criar um projeto via linha de comando

```
flutter create --template=app example
```



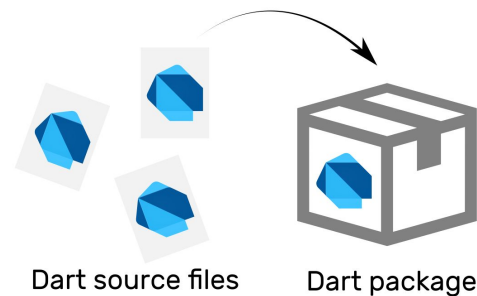
- Para podermos utilizar a nossa package dentro do projeto de teste, precisamos incluí-la no pubspec da aplicação de exemplo
- Como ela ainda não está liberada e queremos testar a package que está localmente em nosso computador, precisamos utilizar a cláusula *path*

```
flutter create --template=app example
```



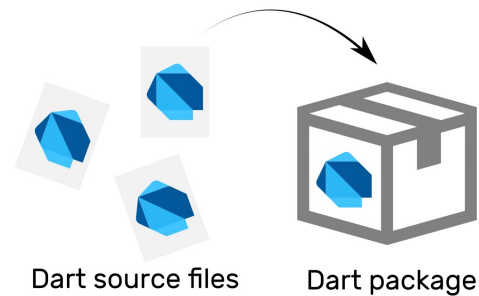


- Vamos criar uma package que implementa uma integração com o site ViaCEP
- Você deverá criar uma nova *package*
- Essa package deverá possuir um *model* da resposta do ViaCEP
- Você deverá criar também um repository que fará a integração entre o ViaCEP e o model
- Você pode testar a lógica usando os testes unitários





- Crie agora um widget dentro do seu pacote que ao ser criado, deve receber um CEP
- Esse CEP, quando o componente for desenhado, deve pesquisar o CEP e exibir as informações em tela
- Utilize um *FutureBuilder* para isso



- Vamos criar agora um projeto de exemplo
- Esse projeto deve utilizar o widget criado no exercício 2
- Ao digitar, a pesquisa deve ser realizada

The screenshot shows a mobile application interface for searching a CEP (Brazilian ZIP code). At the top, the status bar displays the time 7:54 and various icons. Below it, a blue header bar contains the title "Pesquisa de CEP". The main content area has a light gray background. It starts with the prompt "Informe o CEP desejado" in blue text, followed by the input field containing "89041390". Below the input field, the search results are displayed in a table-like format with two columns: labels on the left and values on the right. The labels are: CEP, Bairro, Complemento, DDD, Gia, IBGE, Localidade, Logradouro, and Siafi. The corresponding values are: 89041-390, Velha, 47, 4202404, Blumenau, Rua Thiago Piazza, and 8047. At the bottom of the screen, there is a numeric keypad with buttons for digits 1-9, 0, a comma, a period, a minus sign, an equals sign, a backspace icon, and a checkmark icon. The keypad is styled with rounded buttons and a light gray background.

7:54

Pesquisa de CEP

Informe o CEP desejado

89041390

CEP	89041-390
Bairro	Velha
Complemento	
DDD	47
Gia	
IBGE	4202404
Localidade	Blumenau
Logradouro	Rua Thiago Piazza
Siafi	8047

1 2 3 -

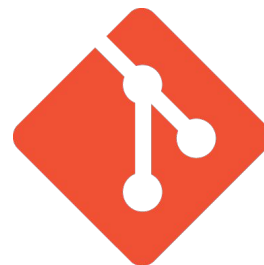
4 5 6 =

7 8 9 ✕

, 0 . ✓



- Para liberar o nosso pacote, inicialmente vamos enviá-lo para o github
- Vamos criar um novo projeto, caso ainda não exista
- Efetuamos o commit e o push





- Para registrar o nosso pacote no pub.dev
- O comando dry run verifica se a package está pronta para ser commitada

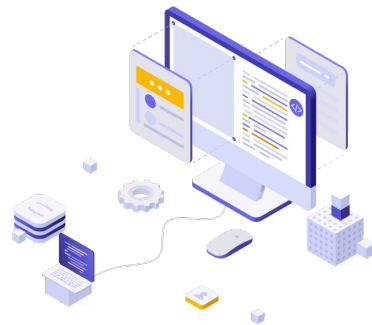
```
flutter pub publish --dry-run
```

- Após ter o Ok, rode o comando sem o dry run para criar a package no pub.dev

```
flutter pub publish
```



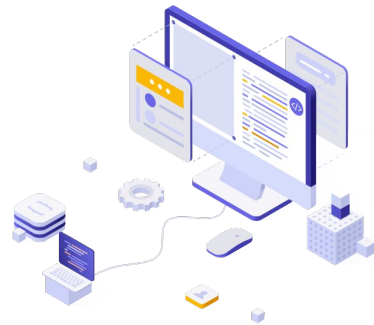

- Indicada somente para apps muito grandes
- Consiste em dividir o app em múltiplos pacotes
- Isso melhora o tempo de build do app
- Dividindo o app em pacotes nós também facilitamos o trabalho em equipes muito grandes





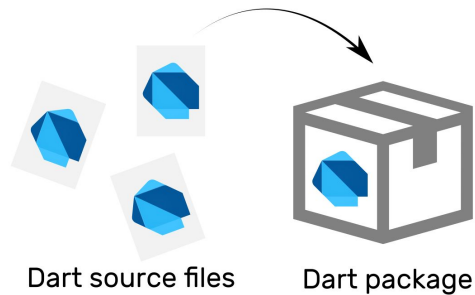
- Referenciando um pacote do git

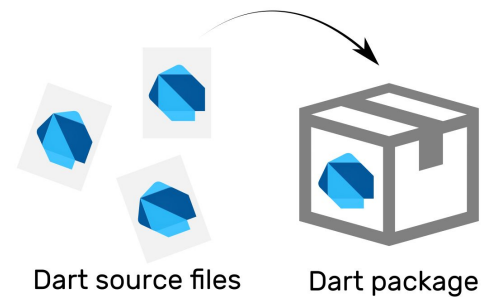
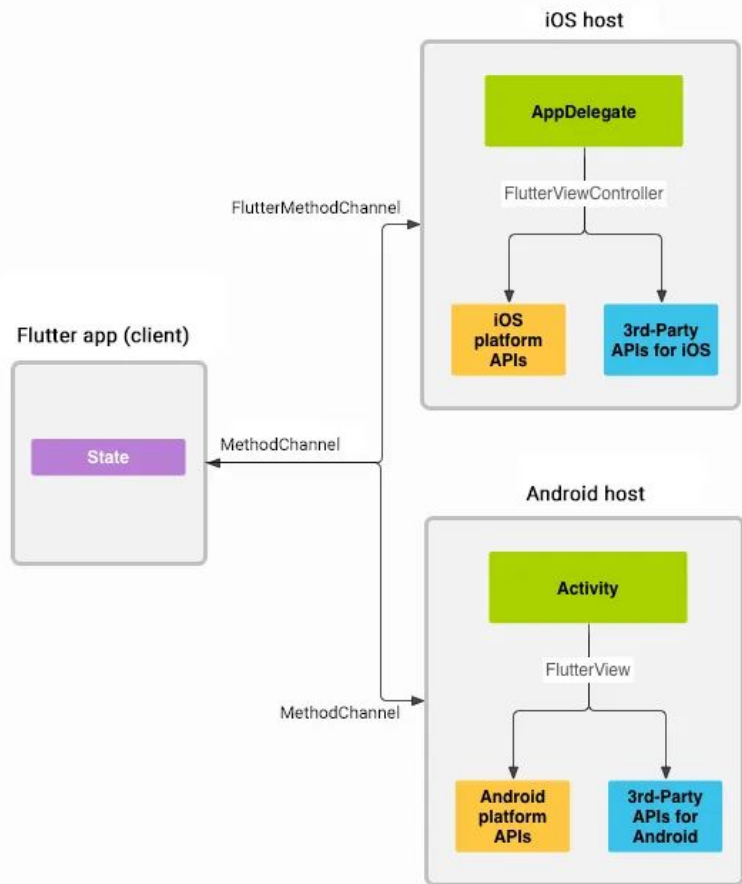
```
carousel_pro:  
  git:  
    url: https://github.com/jlouage/flutter-carousel-pro.git  
    ref: main
```





- Para desenvolver plugins, é necessário conhecimento da linguagem nativa a qual você deseja trabalhar
- Por padrão os projetos Android são criados usando Kotlin
- Projetos iOS, usando Swift
- Mas isso pode ser alterado, no caso de Android para Java
- E no caso de iOS, para Objective-C







- A parte Flutter do aplicativo envia mensagens para seu host, a parte iOS ou Android do aplicativo, através do Platform Channel
- O host possui um listener no através do Platform Channel e recebe a mensagem. Em seguida, ele invoca as APIs específicas da plataforma usando a linguagem de programação nativa e envia uma resposta de volta ao cliente, a parte Flutter do aplicativo

