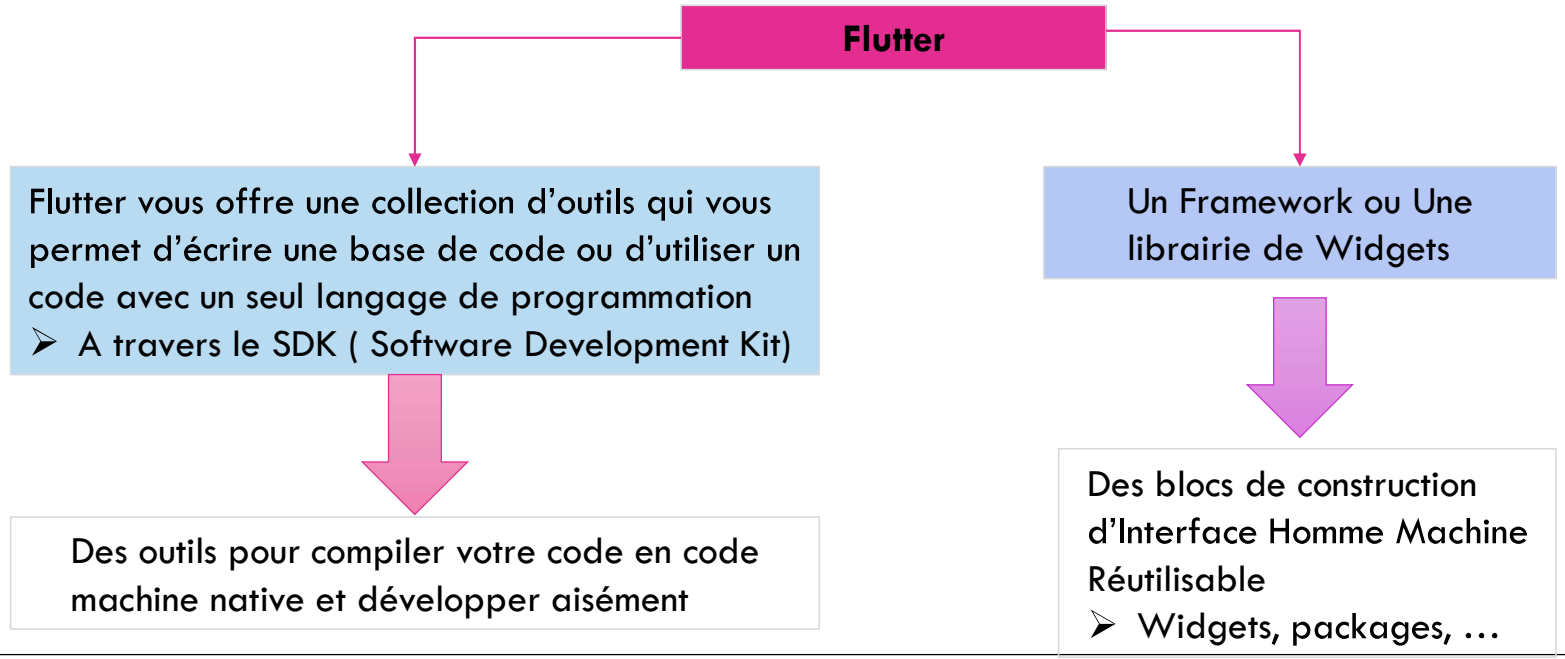
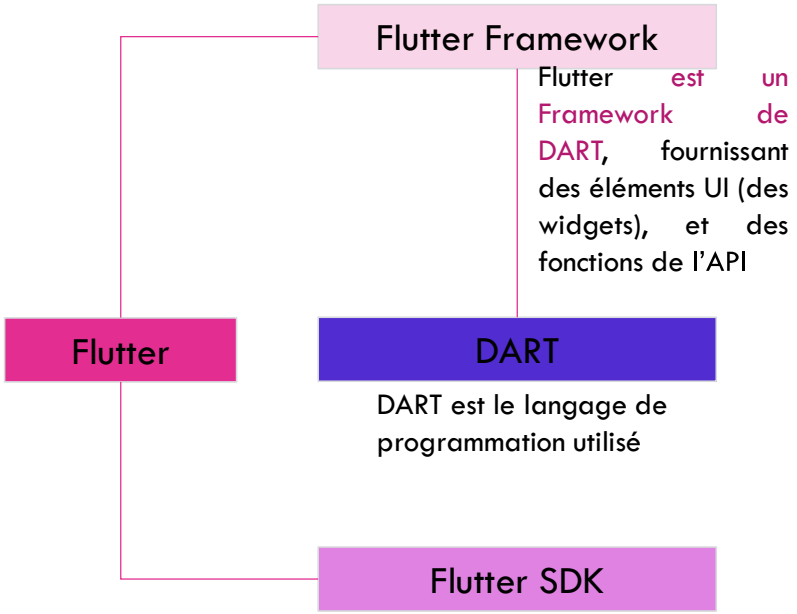
Qu'est-ce que Flutter ?

- Flutter est un « outil » qui permet de construire des applications (iOS, Android native cross-platform avec **un seul langage de programmation** et **une seule base de code**



Le framework Flutter & Dart

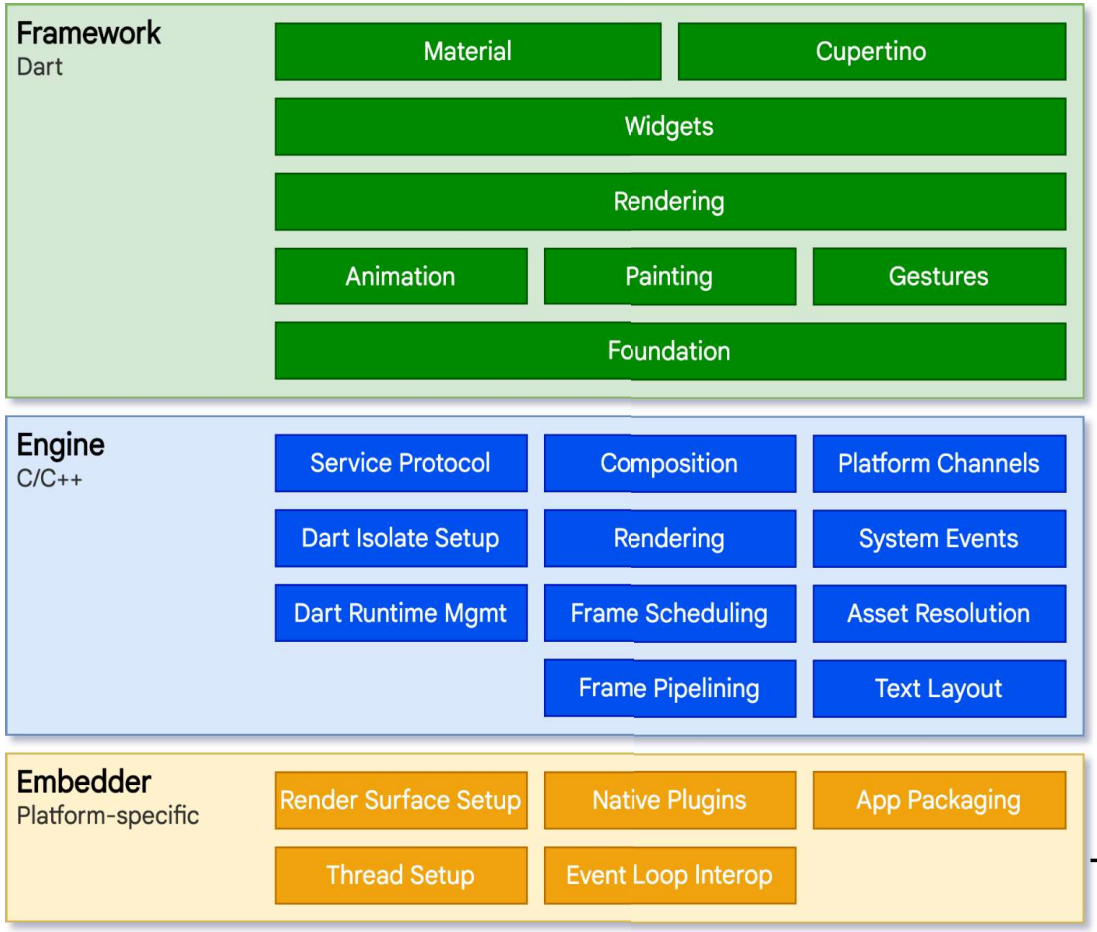
- Flutter utilise un langage de programmation nommé dart
- Dart est un langage de programmation qui se focalise sur le Front-end (mobile apps , web), user interface (UI) development
- DART est un langage orienté objet, fortement typé et sa syntaxe est soit disant un mixte entre le Javascript, Java et C#.
- Dans ce cours, nous allons surtout nous concentrer sur comment Flutter utilise DART



Architecture de Flutter

- Flutter est conçu comme un système extensible en couches.
- Flutter a une architecture modulaire et extensible en couches.
  - Cela vous permet d'écrire votre logique d'application une seule fois et d'avoir un comportement cohérent sur toutes les plates-formes, même si le code du moteur sous-jacent diffère selon la plate-forme.
- Il existe sous la forme d'une série de bibliothèques indépendantes qui dépendent chacune de la couche sous-jacente.
- L'architecture en couches expose également différents points de personnalisation et de remplacement, si nécessaire.

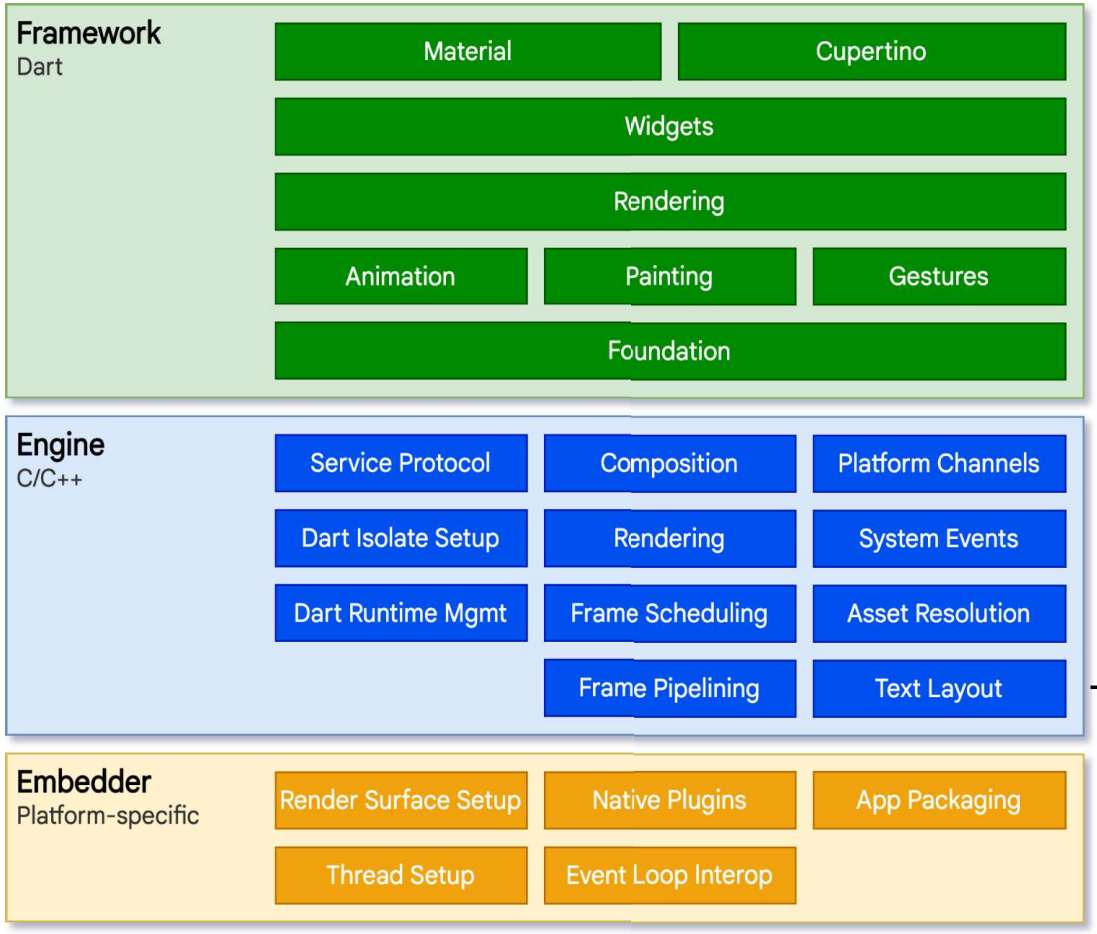
Architecture de Flutter



- Un **embedder** est spécifique à la plate-forme fournit un point d'entrée ; se coordonne avec le système d'exploitation sous-jacent pour l'accès aux services tels que les surfaces de rendu, l'accessibilité et l'entrée; et gère la boucle d'événement de message.

- L'**embedder** est écrit dans un langage adapté à la plateforme : actuellement Java et C++ pour Android, Objective-C/Objective-C++ pour iOS et macOS, et C++ pour Windows et Linux.

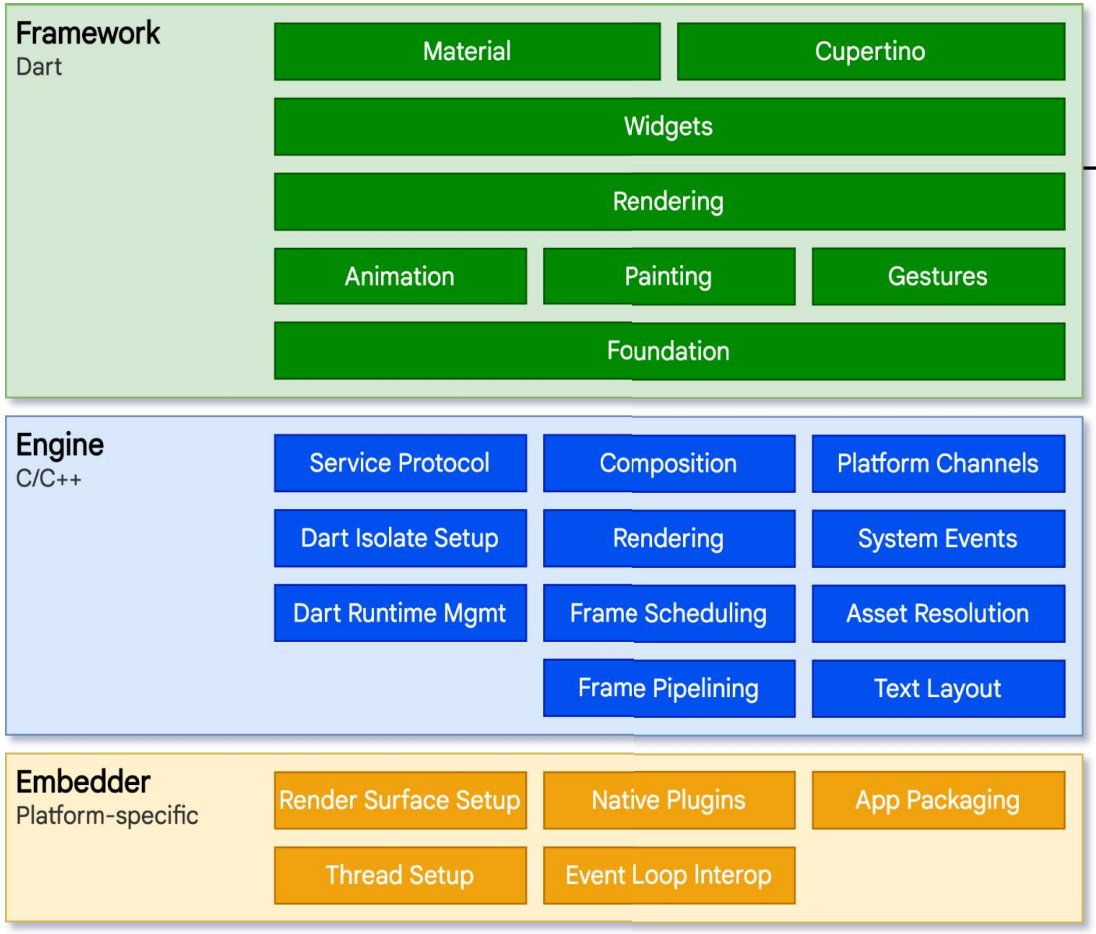
Architecture de Flutter



- Le **moteur Flutter (Flutter Engine)**, qui est principalement écrit en C++ et prend en charge les primitives nécessaires pour prendre en charge toutes les applications Flutter.

- Le **moteur** est responsable de la pixélisation des scènes composées chaque fois qu'une nouvelle image doit être peinte. Il fournit l'implémentation de bas niveau de l'API principale de Flutter:
  - les graphiques (via Skia),
  - la mise en page du texte,
  - les E/S de fichiers et de réseau,
  - la prise en charge de l'accessibilité
  - …..

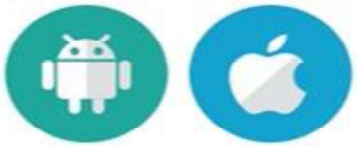
Architecture de Flutter



- Contient l'ensemble des classes de l'API.
- La couche de rendu fournit une abstraction pour traiter la mise en page.
  - ➢ création d'arborescence d'objets pouvant être rendus.
  - ➢ Manipulation des objets de manière dynamique, l'arborescence mettant automatiquement à jour la mise en page pour refléter vos modifications.
- La couche ***widgets*** est une abstraction de composition.
  - ➢ La couche widgets vous permet de définir des combinaisons de classes que vous pouvez réutiliser.
- Les bibliothèques Matérielles et Cupertino offrent des ensembles complets des Widgets et des méthodes pour les maniupler

Flutter Architecture

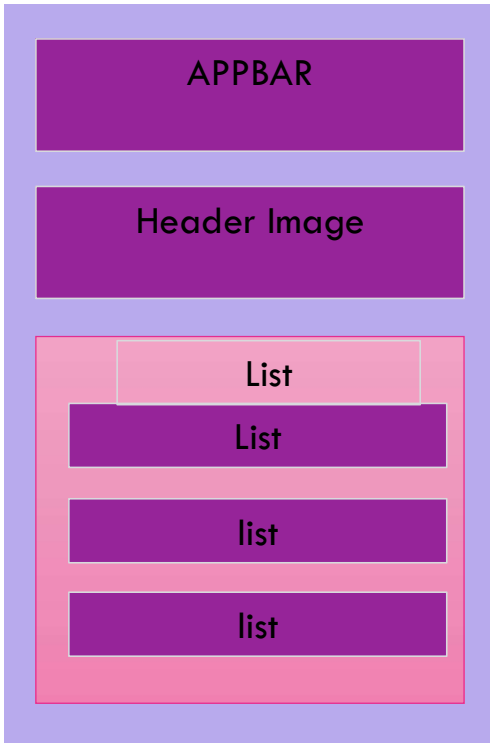
UI as Code: Build a Widget Tree



Embrace Platform Differences

One Codebase

Tout est Widgets

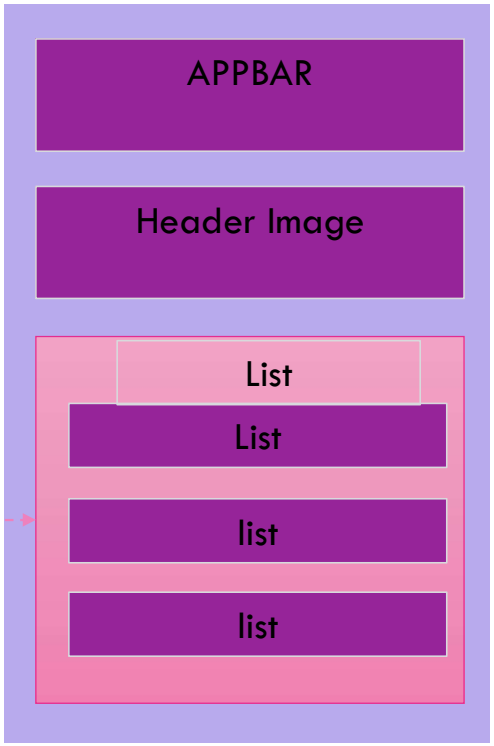


Toute application Flutter que vous construisez est un tas de widgets

Un **widget** c'est le bloc d'interfaces utilisateur (UI) construits et que vous voyez sur votre écran, exemples de widgets :

- AppBar
- HeaderImage
- List
- ….

Tout est Widgets



Toute application Flutter que vous construisez est un tas de widgets

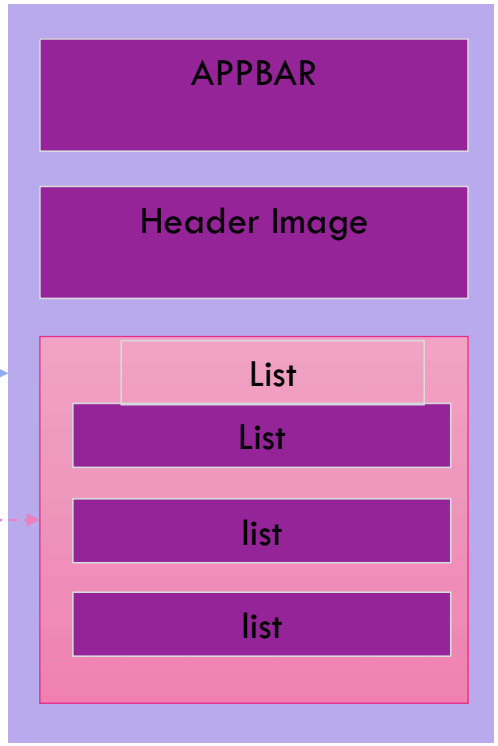
Un **widget** c'est le bloc d'interfaces utilisateur (UI) construits et que vous voyez sur votre écran, exemples de widgets :

- AppBar
- HeaderImage
- List
- ….

Un widget peut contenir d'autres widget, on dit qu'on est entrain de composer **un arbre de widget**

Tout est Widgets

Même la page qui contient tous les widgets sur votre écran est un widget
- **Scaffold widget**



Toute application Flutter que vous construisez est un tas de widgets

Un **widget** c'est le bloc d'interfaces utilisateur (UI) construits et que vous voyez sur votre écran, exemples de widgets :
- AppBar
- HeaderImage
- List
- ….

Un widget peut contenir d'autres widget, on dit qu'on est entrain de composer **un arbre de widget**

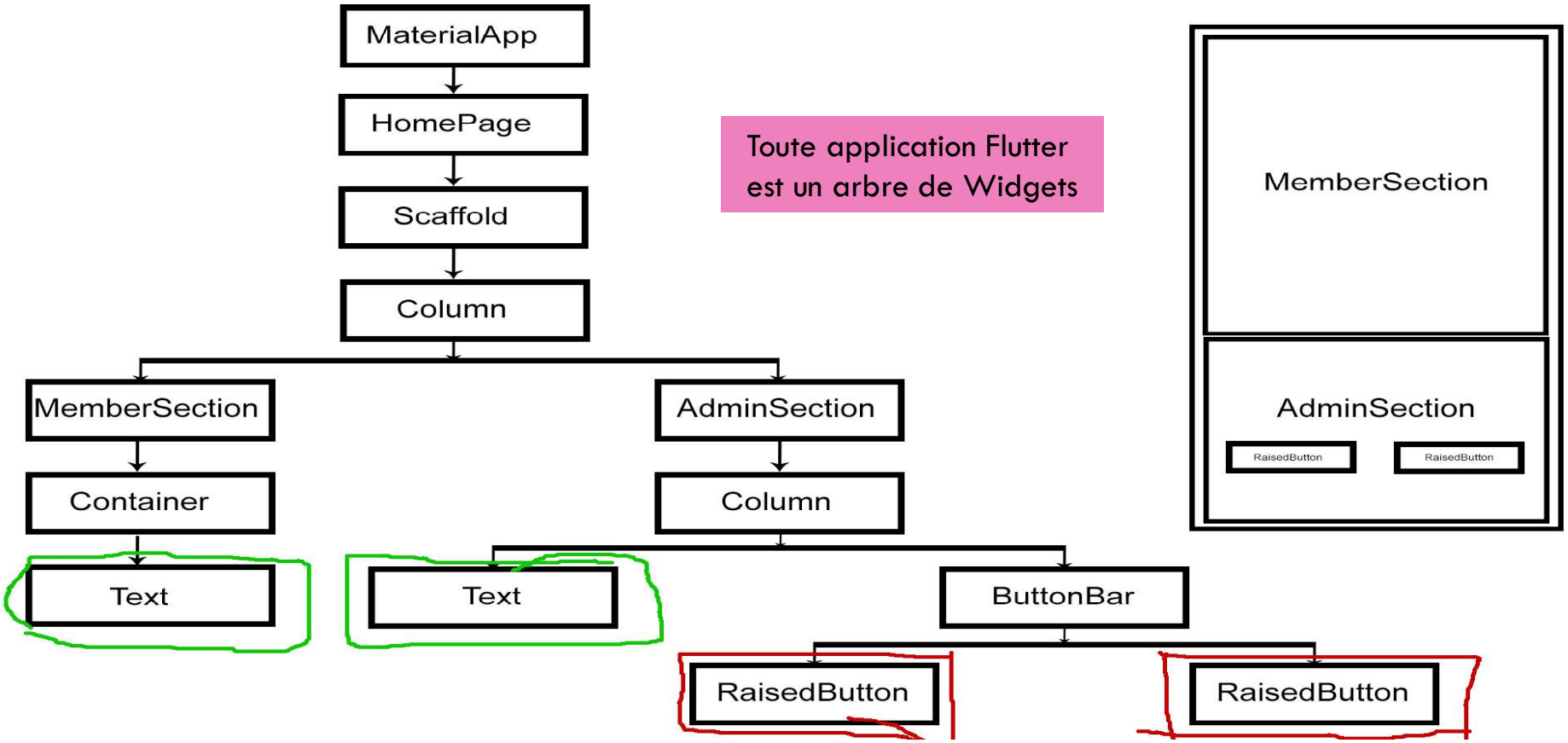
Les Widgets

- Les widgets sont essentiellement des composants d'interface utilisateur utilisés pour créer l'interface utilisateur de l'application.

- Dans Flutter, l'application est elle-même un widget.
  - L'application est le widget de niveau supérieur et son interface utilisateur est construite en utilisant un ou plusieurs enfants (widgets), qui à nouveau sont construits en utilisant ses widgets enfants.

  - Cette fonctionnalité de composabilité nous aide à créer une interface utilisateur de toute complexité.

Votre Application et les Widget



Le Widget MaterialApp

- Il s'agit du widget parent de Flutter UI Widget

- Le widget MaterialApp est une classe Flutter qui fournit une disposition de conception material design. En fait, c'est un composant principal pour d'autres widgets enfants et on l'utilise généralement comme widget de niveau supérieur.
- (En fait, le mot material ou matériel, veut dire qu'il s'inspire d'objets réels.)

- Le Widget MaterialApp possède plusieurs propriétés (home, theme, color, …)

Le Widget Scaffold (1/2)

- Scaffold est une classe Flutter très importante qui **implémente la structure de mise en page visuelle** material design. Ce widget encapsule de nombreux widgets comme **AppBar, Drawer**, **BottomNavigationBar**, **FloatingActionButton**, **SnackBar**, etc.

- C'est un conteneur de niveau supérieur pour MaterialApp .

- Scaffold s'étend pour remplir l'espace disponible, ce qui signifie qu'il occupe tout l'écran. En fait, nous pouvons utiliser le widget Scaffold sans paramètres.

- Selon la structure de mise en page visuelle de Material Design , l'écran d'une application mobile affiche plusieurs composants. Par exemple,
  - une barre supérieur, qui porte le nom de l'application,
  - un menu de navigation ou une barre de navigation inférieure
  - et un bouton d'appel à l'action flottant.
  - ces composants permettent d'accéder à différents écrans et à d'autres fonctionnalités de l'application.

Types de Widgets

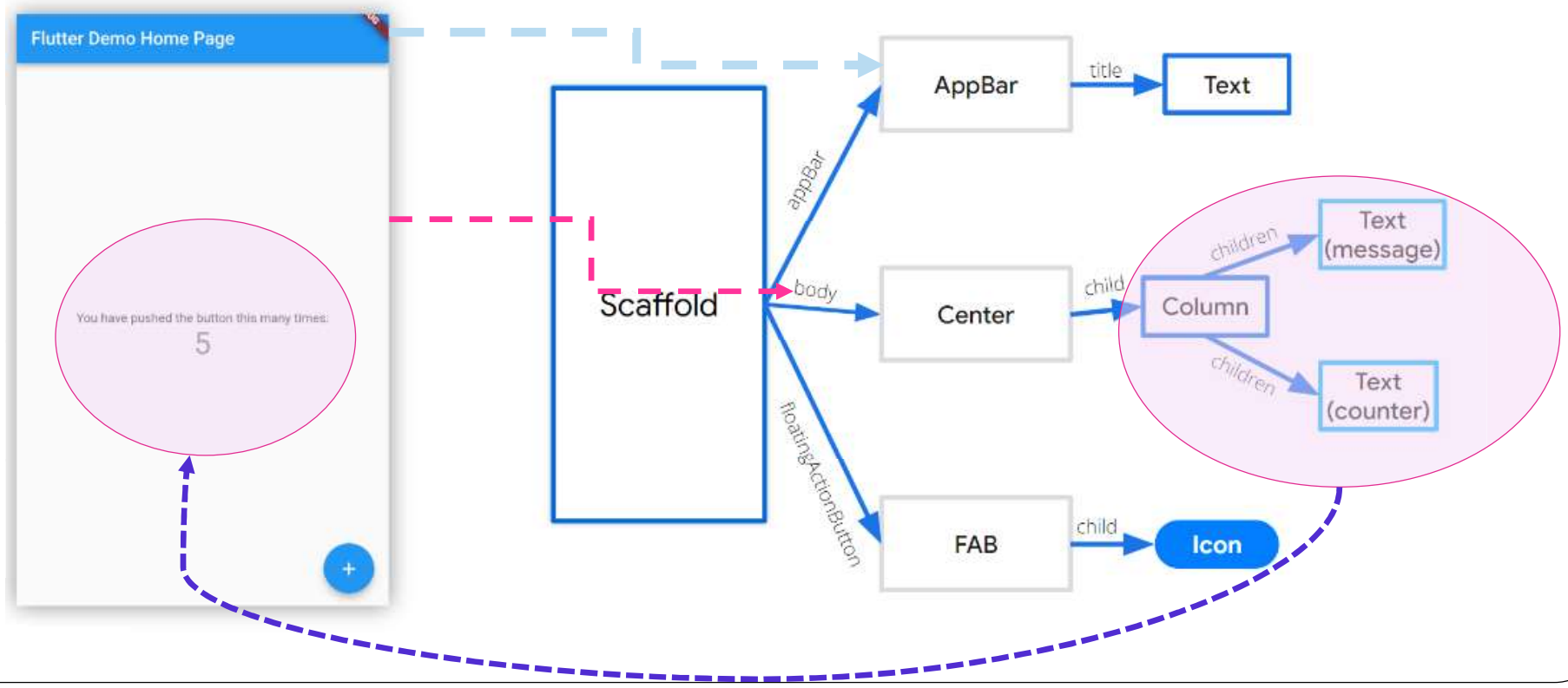
Layout widgets (Widgets de Mise en pages)

- Container
- Column
- Row
- Center
- Align
- ….

Widgets « visible »

- Text
- TextField
- ElevatedButton
- Icon
- Image
- ….

Le Widget Scaffold (2/2) : Exemple



UI entant que CODE

Installation Flutter + IDE Intellij

- Lien d'installation
- ➢ https://www.geeksforgeeks.org/how-to-install-and-setup-flutter-in-intellij-idea/

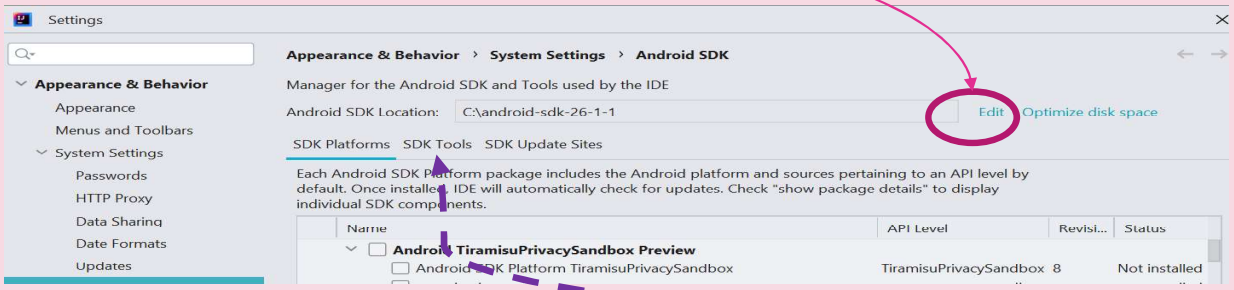
A ne pas oublier :

•Les outils nécessaires pour le fonctionnement de Flutter

➢ Android SDK

(Pour l'installation de ce composant à partir de Intellij, Accéder à :

Files> Settings> Android SDK > SDK platforms > Edit (vous installez la version proposée)



➢ et il faut aussi installer Android Cmd-line tools à (Files> Settings> Android SDK > SDK tools> cmd-Line tools)

Ou voir Le processus d'installation sur cette vidéo :
https://www.youtube.com/watch?v=5FEY5-1m1cs&ab_channel=JohannesMilke

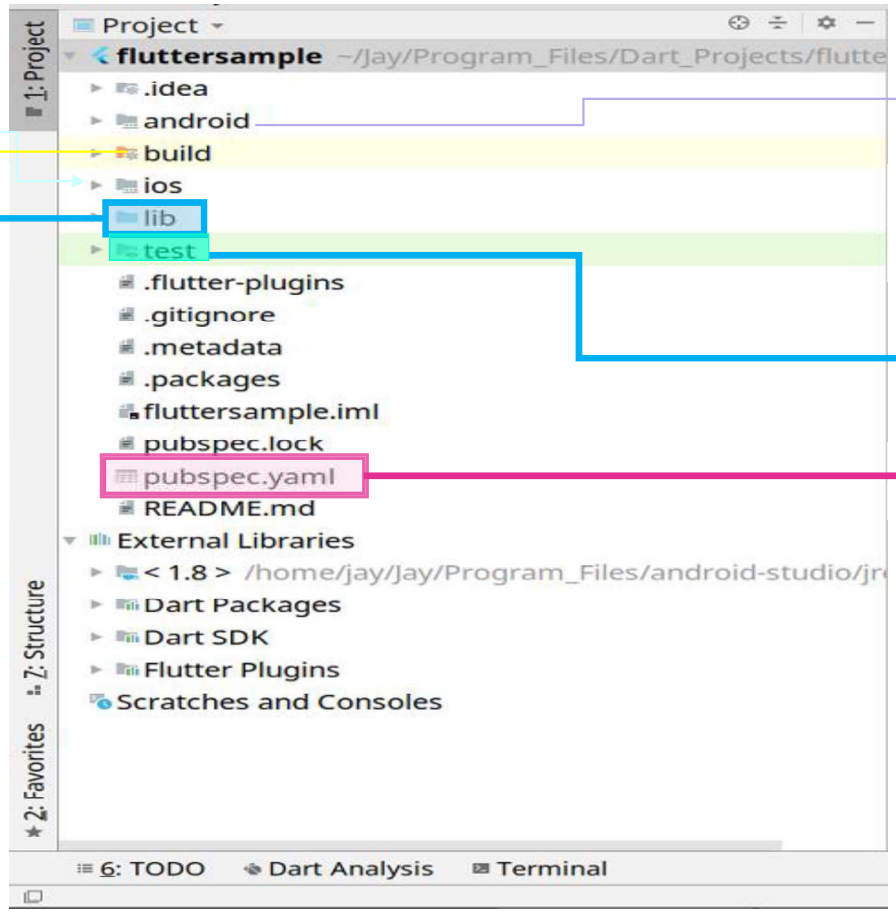
Structure d'un projet Flutter

Code généré automatiquement pour créer et lancer une application IOS.

Ce dossier est géré par Flutter

Build est un dossier généré à la compilation de l'application

Contient les codes source de notre application flutter



Code généré automatiquement pour créer et lancer une application Android.

Ce dossier est géré par Flutter

Contient les codes d'automatisation de test de notre application flutter

Fichier YAML permettant la gestion de dépendance de notre application

Flutter & Dart : Fonctionnement interne