Conceptos Básicos De Flutter

Presentado Por:

Andrés Calderón

Servicio Nacional De Aprendizaje: Sena

Centro Agroempresarial Y Desarrollo Pecuario Del Huila

Tecnólogo En Análisis Y Desarrollo De Software

No. De Ficha 2502629

Garzón Huila

2023

Estructura de un proyecto en Flutter

- .dart_tool: se utiliza para almacenar información y configuraciones relacionadas con las herramientas de desarrollo de Dart y Flutter.

- .**idea/:** Esta carpeta es generada por el entorno de desarrollo, como Android Studio o IntelliJ IDEA. Contiene configuraciones específicas del proyecto y no debe ser modificada manualmente.

- build/: Esta carpeta contiene archivos generados durante el proceso de construcción de la aplicación. Puedes encontrar aquí los archivos compilados y los recursos generados para ejecutar la aplicación.

- linux/, macos/, web/, windows/: Estas carpetas contienen configuraciones específicas para la compilación de tu aplicación para plataformas específicas. Por ejemplo, la carpeta "linux/" contendría los archivos necesarios para compilar la aplicación para Linux.

- .metadata: Este archivo se utiliza para almacenar información sobre el proyecto y las dependencias. No se debe modificar manualmente.

- .packages: Este archivo contiene información sobre las dependencias de Dart que se utilizan en el proyecto.

- analysis_options.yaml: Este archivo se utiliza para configurar las reglas de análisis estático de código usando herramientas como Dart Analyzer o pedantic.

- pubspec.lock: Este archivo es generado automáticamente por el sistema de gestión de paquetes de Dart (Pub). Contiene información detallada sobre las versiones exactas de las dependencias que se deben utilizar en el proyecto.

- pubspec.yaml: Este archivo es fundamental en un proyecto Flutter. En él, defines las dependencias de tu proyecto, la configuración del proyecto, el nombre de la aplicación y otra información relevante.

- README.md: Este es un archivo de marcado (Markdown) que se utiliza para proporcionar documentación sobre el proyecto. Es comúnmente utilizado para describir el propósito del proyecto, cómo configurarlo y ejecutarlo, y otras instrucciones importantes para los desarrolladores o colaboradores. –

- .gitignore: Es un archivo para ayudar a git a ignorar todo lo que no necesita flutter. Señala los archivos que git no les va dar seguimiento.

- pubspec.lock: Dice como fue construido cada una de las dependencias de nuestro proyecto

- android/: Esta carpeta contiene el código fuente específico de Android para tu aplicación. Si necesitas personalizar aspectos de tu aplicación para Android, como permisos o configuraciones específicas, aquí es donde lo harías.

- ios/: Similar a la carpeta "android/", esta carpeta contiene el código fuente específico de iOS para tu aplicación. Puedes personalizar la configuración de iOS, como permisos y recursos, aquí.

- lib/: Esta es una de las carpetas más importantes en tu proyecto Flutter. Contiene el código fuente de tu aplicación Flutter. El archivo "main.dart" suele ser el punto de entrada de tu

aplicación, donde se define la función main() que se ejecuta al iniciar la aplicación. Puedes organizar tu código en subcarpetas dentro de "lib/" para mantenerlo ordenado y modular.

- test/: En esta carpeta, puedes escribir pruebas unitarias y de integración para tu aplicación. Flutter incluye un marco de pruebas que facilita la escritura y ejecución de pruebas.

- assets/: Aquí puedes almacenar archivos estáticos, como imágenes, fuentes o archivos de configuración, que tu aplicación pueda necesitar. Estos archivos se pueden acceder desde tu código Flutter.

- pubspec.yaml: Este archivo es esencial para gestionar las dependencias y configuraciones de tu proyecto. Puedes especificar las dependencias de Flutter y los paquetes de terceros que tu aplicación necesita en este archivo. También puedes definir información sobre tu aplicación, como el nombre, la versión y los recursos.

Glosario

Void: Void no es un término específico de Flutter, sino un tipo de retorno que se usa en muchos lenguajes de programación, incluido Dart (el lenguaje de programación de Flutter). Se utiliza para indicar que una función no devuelve ningún valor.

Class: Una "class" (clase en español) es una estructura fundamental en la programación orientada a objetos (POO). En Flutter, las clases se utilizan para definir objetos y componentes reutilizables, como widgets y modelos de datos.

Extends: La palabra clave "extends" se utiliza en la herencia de clases. En Flutter, se usa para crear una nueva clase que hereda propiedades y métodos de una clase existente (superclase). Esto se utiliza comúnmente al crear widgets personalizados.

@override: Es un decorador que se utiliza en Dart para indicar que un método en una clase deriva de una superclase y se está sobrescribiendo en la subclase. En Flutter, se utiliza comúnmente en métodos como build() para indicar que se está sobrescribiendo un método heredado.

Stateless Widget: Un "Stateless Widget" es un tipo de widget en Flutter que representa una parte de la interfaz de usuario que no cambia durante su ciclo de vida. Estos widgets son inmutables y se utilizan para crear componentes estáticos.

Const: "Const" es una palabra clave que se usa en Dart para declarar constantes. En Flutter, se usa para crear objetos inmutables y optimizar el rendimiento al declarar valores constantes que no cambian durante la ejecución de la aplicación. Widget: Un "Widget" es un componente fundamental en Flutter que define la interfaz de usuario de la aplicación. Los widgets son los bloques de construcción básicos para crear la interfaz de usuario y pueden ser tanto "Stateless" como "Stateful".

Widget build: "Widget build" es un método que se encuentra en las clases de widget en Flutter, especialmente en las clases que heredan de "StatelessWidget" y "StatefulWidget". Este método se utiliza para construir la parte visual del widget y se llama automáticamente cuando es necesario redibujar el widget en la pantalla.

Return: "Return" es una palabra clave utilizada en funciones o métodos para devolver un valor. En Flutter, se usa dentro del método "build()" para devolver el widget que representa la interfaz de usuario que se mostrará en la pantalla.

Child: "Child" se utiliza comúnmente en Flutter para indicar el contenido de un contenedor o un widget que contiene otros widgets. Por ejemplo, un "Container" puede tener un único "child" que define su contenido.

Children: Al igual que "Child", "Children" se usa para indicar varios elementos hijos en un widget que admite múltiples hijos, como "Column" o "Row". Stateful

Widget: Un "Stateful Widget" es un tipo de widget en Flutter que puede cambiar su estado durante su ciclo de vida. Estos widgets tienen un objeto de estado asociado que permite que los datos cambien y se redibuje la interfaz de usuario en función de esos cambios. Final: "Final" es una palabra clave que se usa en Dart para declarar una variable o propiedad como inmutable. En Flutter, a menudo se utiliza en combinación con "Stateless" para declarar propiedades que no cambian después de su inicialización.