

Configuración, instalación y administración del SDK

Autores

Cristian Acalo

Leonardo Obando

Ariel Guevara

Universidad de las Fuerzas Armadas - ESPE

cjacalo@espe.edu.ec

jlobando2@espe.edu.ec

Adguevara7@espe.edu.ec

Resumen

La aplicación móvil desarrollada en Flutter permite calcular:

- el pago por consumo de energía eléctrica (KW)
- el precio final de un artículo con descuento del 20% e IVA del 15%

Se implementó una interfaz intuitiva con TextField (para entrada de datos), botones de cálculo (ElevatedButton) y tarjetas (Card) para organizar las secciones. Los resultados se muestran dinámicamente usando setState(), garantizando actualización en tiempo real.

I. Introducción

El presente laboratorio se centra en dar los primeros pasos en el desarrollo móvil con Flutter. Se va a explicar el proceso de instalación de un entorno de desarrollo funcional para el desarrollo móvil, para posteriormente explicar el desarrollo de un aplicativo con dos funcionalidades básicas con el fin de entender conceptos fundamentales.

En primer lugar se tiene el cálculo de consumo eléctrico, y por otro lado se tiene el cálculo de precios con descuento e impuestos.

II. Trabajos Relacionados

El libro Flutter for Dummies [1] destaca cómo widgets como TextField y Card agilizan el desarrollo de interfaces para cálculos matemáticos.

En el artículo Mobile UIs for Energy Apps [2], se comparan frameworks y se valida la eficiencia de Flutter para apps de servicios básicos.

III. Materiales y Métodos

Item	Especificaciones
Computador	Windows 11, 16GB RAM
IDE	Android Studio
Flutter Windows	Versión 3.29.3
Emulador	BlueStacks 5
Dart	Versión 3.7.3

Procedimiento:

1. Configuración inicial:

- Crear proyecto Flutter con `flutter create pago_consumo_electrico`.
- Reemplazar el código de `main.dart` con la solución propuesta.

2. Implementación:

- Diseñar dos secciones independientes usando `Column` y `Card`.
- Agregar `TextEditingController` para manejar inputs de usuario.
- Programar funciones `_calcularPagoLuz()` y `_calcularPrecioArticulo()` con fórmulas

IV. Desarrollo

Instalación de Entorno de Desarrollo

Para el desarrollo del presente Laboratorio, se realizó varias instalaciones. En primer lugar se realizó la instalación de Dart. Desde la página oficial de Dart, se descarga el instalador de acuerdo con el sistema operativo:

Announcing Dart 3.7! Find out about updates to the language, analyzer, pub.dev, and more, in the [blog post](#).

Dart Overview Docs Community Try Dart Get Dart

Language
Core libraries
Effective Dart
Packages
Development
Interoperability
Tools & techniques
Resources
Related sites

Dart SDK archive

Use this archive to download [specific versions](#) of the Dart SDK and the Dart API documentation.

Want to install Dart with your OS's package manager? [Get Dart](#).

Notice
Dart tools might send usage metrics and crash reports to Google. If you download the Dart SDK, you agree to the [Google Terms of Service](#). To learn how Dart handles this data, consult the [Google Privacy Policy](#).

To toggle data collection, use the following options on the [dart](#) tool:

- To enable anonymous analytics, run `dart --enable-analytics`.
- To disable anonymous analytics, run `dart --disable-analytics`.

Stable channel
Stable channel builds are tested and approved for production use.

Version: OS:

Version	OS	Architecture	Release date	Downloads
3.7.3 (ref 633eb6bb)	Windows	x64	Apr 17, 2025	Dart SDK (SHA-256)
3.7.3 (ref 633eb6bb)	Windows	IA32	Apr 17, 2025	Dart SDK (SHA-256)
3.7.3 (ref 633eb6bb)	Windows	ARM64	Apr 17, 2025	Dart SDK (SHA-256)
3.7.3 (ref 633eb6bb)	---	---	Apr 17, 2025	API Docs

dart.dev uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic. [Learn more](#) [OK, got it](#)

La instalación consiste en descomprimir el zip en alguna dirección del equipo. En este caso se realizó la instalación en C:\Program Files\dart-sdk

También se requirió de la instalación de flutter. Desde la página oficial de flutter se descarga un zip:

Flutter Docs Homepage Community Packages API reference [Get started](#)

Get started
Set up Flutter
Choose a platform
On Windows
Choose a target
Target Android
Target web
Target desktop
On macOS
On Linux
On ChromeOS
Learn Flutter
Tutorials and samples
Stay up to date
App solutions

User interface
Introduction
Widget catalog
Layout
Adaptive & responsive design
Design & theming
Interactivity
Assets & media

Use VS Code to install [Download and install](#)

Download then install Flutter

To install Flutter, download the Flutter SDK bundle from its archive, move the bundle to where you want it stored, then extract the SDK.

- Download the following installation bundle to get the latest stable release of the Flutter SDK.
[flutter_windows_2.29.3-stable.zip](#)
For other release channels, and older builds, check out the [SDK archive](#).
The Flutter SDK should download to the Windows default download directory: `%USERPROFILE%\Downloads`.
If you changed the location of the Downloads directory, replace this path with that path. To find your Downloads directory location, check out this [Microsoft Community post](#).
- Create a folder where you can install Flutter.
Consider creating a directory at `%USERPROFILE% (C:\Users\{username})` or `%LOCALAPPDATA% (C:\Users\{username}\AppData\Local)`.
- Extract the file into the directory you want to store the Flutter SDK.

Warning
Don't install Flutter to a directory or path that meets one or both of the following conditions:

- The path contains special characters or spaces.
- The path requires elevated privileges.

As an example, `C:\Program Files` fails both conditions.

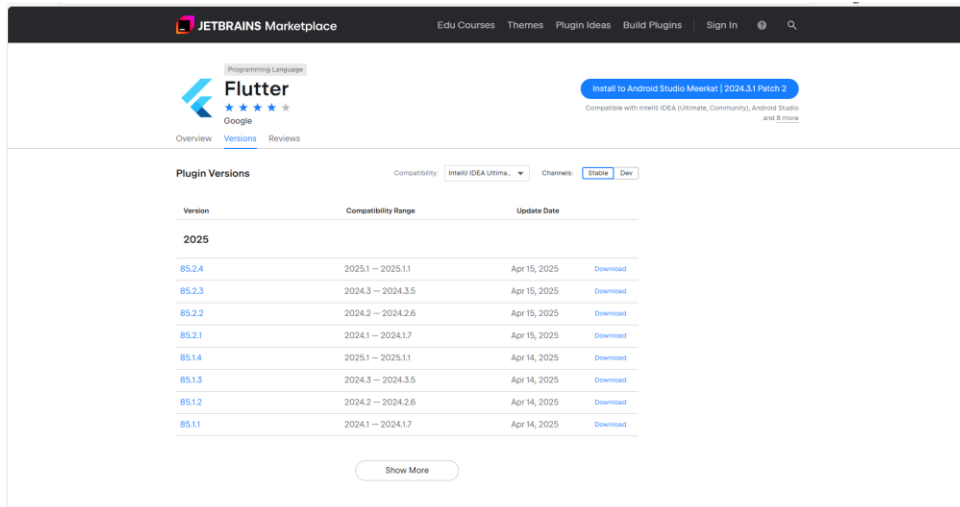
PS C:\> Expand-Archive

docs.flutter.dev uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic. [Learn more](#) [OK, got it](#)

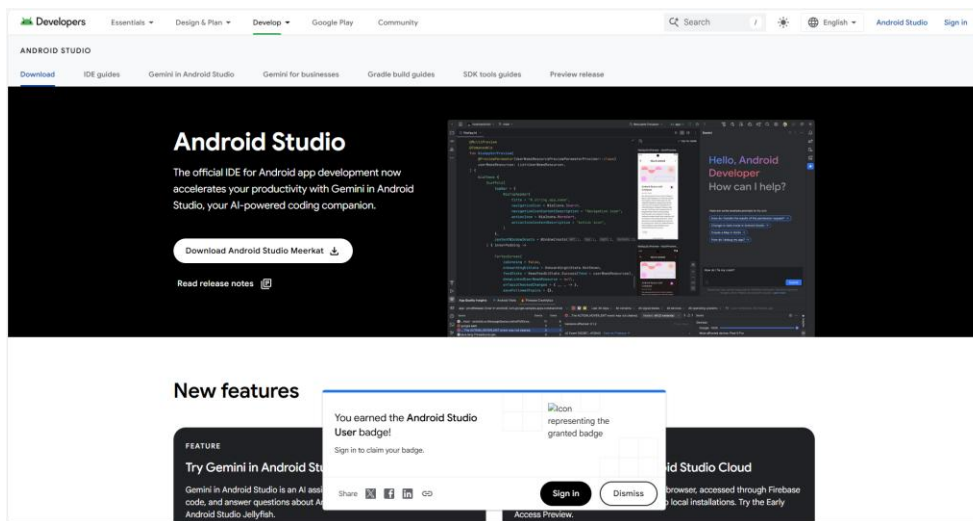
<https://docs.flutter.dev/get-started/install/windows/mobile#193-tab-panel>

La instalación consiste en extraer el zip descargado en algún directorio que no contenga espacios en su nombre y no requiera de permisos de administrador para ser ejecutado. En este caso se realizó la instalación en C:\Users\PERSONAL\Desktop\Cris\flutter

Adicionalmente se realizó la descarga de un plugin de flutter para Android Studio. Este se descarga desde el Marketplace de JetBrains

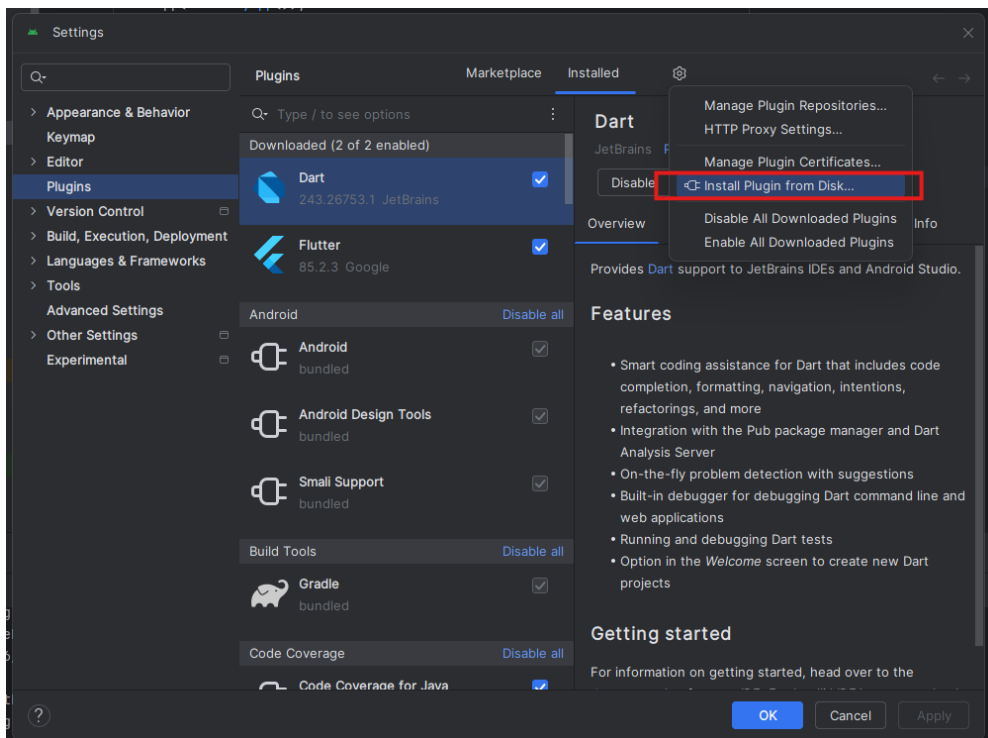


Posteriormente se realizó la descarga de Android Studio. Esto se consigue desde su página oficial:



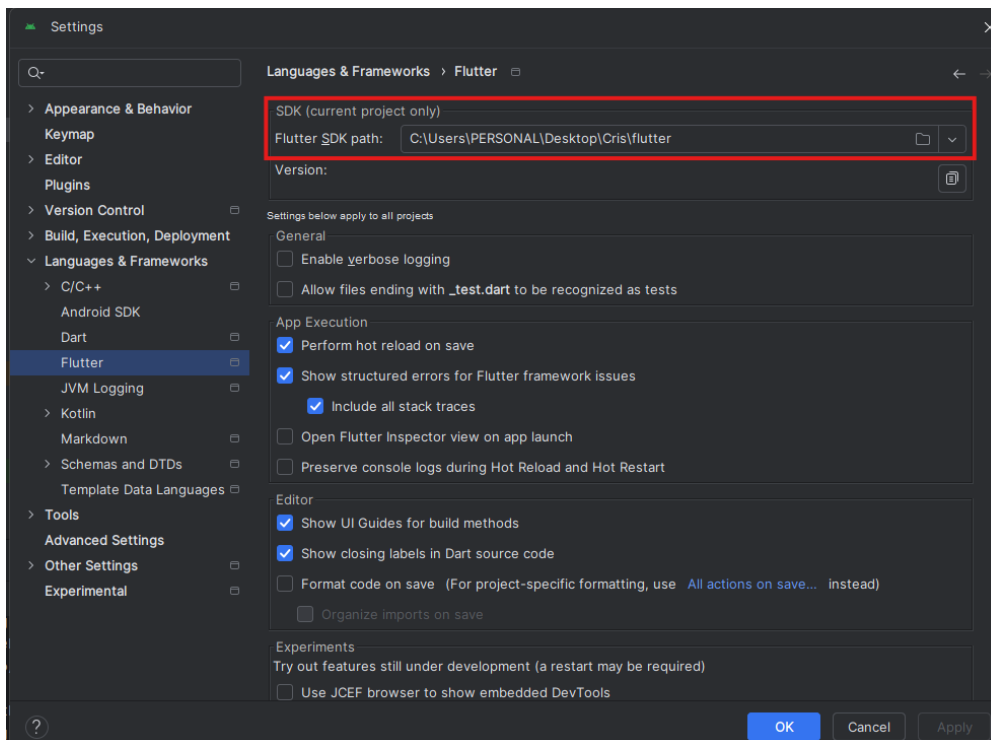
La instalación consiste en una instalación normal con un archivo .exe.

Una vez Instalado, antes de poder usarlo con flutter, es necesario importar el plugin para flutter que se descargó anteriormente.

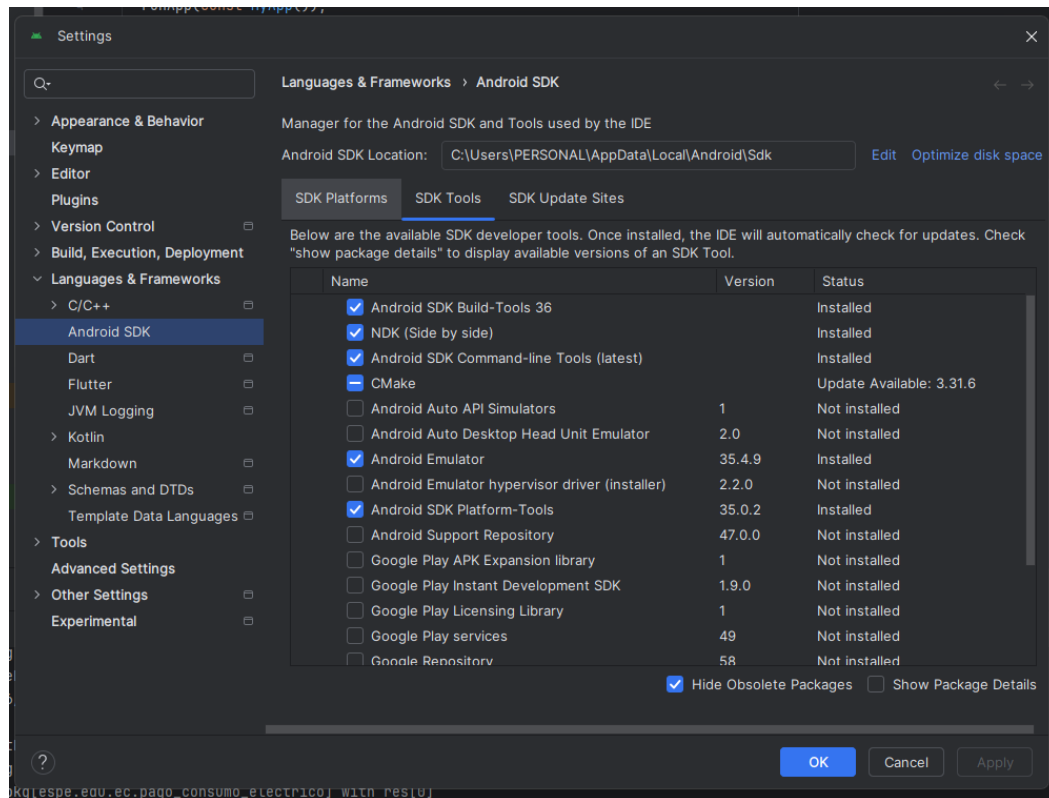


Se selecciona el zip y la instalación se concreta.

Cuando se crea un nuevo proyecto, se debe seleccionar la dirección de instalación de flutter. Esta se puede cambiar más adelante en las configuraciones del proyecto:



Finalmente, para asegurarse de que la aplicación pueda ser ejecutada exitosamente, hay que asegurarse que los siguientes paquetes de SDK estén instalados correctamente:



Ahora bien, para la emulación de un dispositivo Android no se tuvo éxito con Android Studio. Por lo que se decidió recurrir a BlueStacks. Este se descarga desde la página oficial de BlueStacks y se trata de una instalación normal.

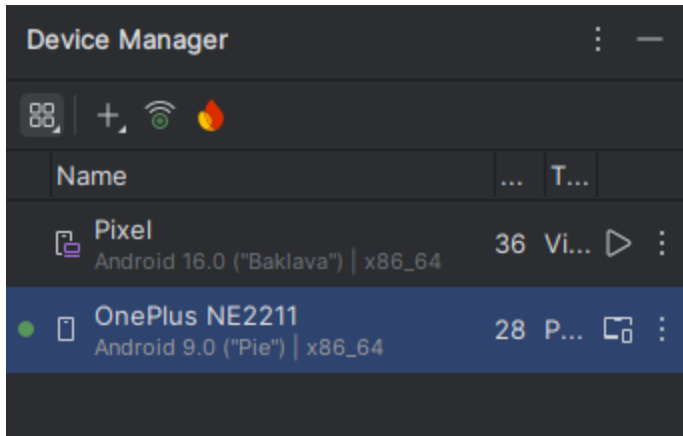
Una vez ejecutado BlueStacks, para que sea detectable para Android Studio hay que ejecutar los siguientes comandos:

- **cd C:\Users\PERSONAL\AppData\Local\Android\Sdk\platform-tools>**
- **adb connect 127.0.0.1:5555**

Y podemos revisar si se detecta correctamente con el comando **adb devices**

```
C:\Users\PERSONAL\AppData\Local\Android\Sdk\platform-tools>adb devices
List of devices attached
127.0.0.1:5555 device
```

Y finalmente será visible por Android Studio



Desarrollo de Aplicativo

Estructura Principal

La aplicación se desarrolló en **Flutter**, utilizando el paradigma de widgets. El archivo main.dart contiene toda la lógica y la interfaz de usuario.

Componentes Clave

Clase MyApp (Punto de Entrada)

- **Propósito:** Configuración inicial de la aplicación.
- **Código relevante:**

```
7  class MyApp extends StatelessWidget {  
8    const MyApp({super.key});  
9  
10   @override  
11   Widget build(BuildContext context) {  
12     return MaterialApp(  
13       title: 'Calculadora CLS',  
14       theme: ThemeData(  
15         primarySwatch: Colors.blue,  
16       ), // ThemeData  
17       home: const HomeScreen(),  
18     ); // MaterialApp  
19   }  
20 }
```

Explicación:

- Define el tema global (Colors.blue).
- Establece HomeScreen como la pantalla inicial.

Clase HomeScreen (Lógica e Interfaz)

- **Propósito:** Contiene los campos de entrada, botones y cálculos.

Variables y Controladores

```
29 class _HomeScreenState extends State<HomeScreen> {
30     // Variables para el cálculo de energía eléctrica
31     double _kwConsumidos = 0.0;
32     double _precioPorKW = 0.0;
33     double _totalPago = 0.0;
34
35     // Variables para el cálculo de artículo con descuento
36     double _precioArticulo = 0.0;
37     double _precioConDescuento = 0.0;
38     double _precioFinal = 0.0;
39
40     // Controladores para los campos de texto
41     final TextEditingController _kwController = TextEditingController();
42     final TextEditingController _precioKWController = TextEditingController();
43     final TextEditingController _articuloController = TextEditingController();
```

Explicación:

- _kwController, _precioKWController, _articuloController: Gestionan la entrada de texto.
- Variables como _totalPago almacenan resultados.

Funciones de Cálculo

```
53 void _calcularPagoLuz() {
54     setState(() {
55         _kwConsumidos = double.tryParse(_kwController.text) ?? 0.0;
56         _precioPorKW = double.tryParse(_precioKWController.text) ?? 0.0;
57         _totalPago = _kwConsumidos * _precioPorKW;
58     });
59 }
60
61 void _calcularPrecioArticulo() {
62     setState(() {
63         _precioArticulo = double.tryParse(_articuloController.text) ?? 0.0;
64         _precioConDescuento = _precioArticulo * 0.8; // 20% de descuento
65         _precioFinal = _precioConDescuento * 1.15; // 15% de IVA
66     });
67 }
```

Explicación:

- setState(): Actualiza la interfaz cuando los valores cambian.
- double.tryParse(): Convierte texto a número (evita errores con entradas no válidas).

Interfaz de Usuario (Widget build)

```
69 @override
70 Widget build(BuildContext context) {
71   return Scaffold(
72     appBar: AppBar(
73       title: const Text('Calculadora CLS'),
74     ), // AppBar
75     body: SingleChildScrollView(
76       padding: const EdgeInsets.all(20.0),
77       child: Column(
78         crossAxisAlignment: CrossAxisAlignment.stretch,
79         children: [
80           // Sección de cálculo de energía eléctrica
81           Card(...), // Card
129
130           const SizedBox(height: 20.0),
131
132           // Sección de cálculo de artículo con descuento
133           Card(...), // Card
179         ],
180       ), // Column
181     ), // SingleChildScrollView
182   ); // Scaffold
183 }
184 }
```

Explicación:

- Card: Organiza cada sección en tarjetas independientes.
- TextField: Permite ingresar valores numéricos (keyboardType: TextInputType.number).
- toStringAsFixed(2): Muestra resultados con 2 decimales (ej: \ \$15.00).

Flujo de la Aplicación

- **Entrada de datos:** Usuario ingresa valores en los campos de texto.
- **Procesamiento:**
 - Al presionar un botón, se ejecuta _calcularPagoLuz() o _calcularPrecioArticulo().
 - Los resultados se actualizan con setState().
- **Salida:** Los valores calculados se muestran debajo de cada sección.

V. Resultados

Pruebas

Se ejecutaron dos tipos de cálculos con datos de entrada controlados:

Cálculo de Energía Eléctrica

KW Consumidos	Precio por KW (\$)	Resultado (\$)
---------------	--------------------	----------------

50	0.15	7.50
120	0.20	24.00
75.5	0.10	7.55

Cálculo de Artículo con Descuento e IVA

Precio Original (\$)	Precio con Descuento (\$)	Precio Final (IVA 15%) (\$)
100	80.00	92.00
250	200.00	230.00
45.50	36.40	41.86

Interfaces

Después del desarrollo del código correspondiente a las dos funcionalidades se obtuvieron las siguientes interfaces:

Cálculo de Consumo Eléctrico:

Cálculo de Consumo Eléctrico

KW consumidos

Precio por KW

Calcular Pago

Total a pagar: \$240.00

Cálculo de precio con descuento e impuestos

Cálculo de Artículo con Descuento

Precio del artículo

Calcular Precio

Precio con descuento: \$40.00

Precio final (con IVA): \$46.00

VII. Conclusiones

- La instalación de Flutter, Dart y Android Studio requiere atención a detalles como las rutas del SDK, las variables de entorno y los plugins del IDE. Tras seguir los pasos correctamente, flutter doctor debe mostrar todos los checks en verde, indicando que el entorno está listo para el desarrollo.

- **Importancia del orden en la instalación:** Seguir una secuencia lógica (Flutter → Android Studio → Plugins) evita errores comunes y facilita la resolución de problemas.
- **Variables de entorno críticas:** Configurar correctamente el PATH del sistema y las rutas del SDK (tanto de Flutter como de Android) es esencial para que los comandos funcionen globalmente.
- **flutter doctor como herramienta diagnóstica:** Esta herramienta no solo señala problemas, sino que también guía hacia las soluciones (ej.: instalar licencias o dependencias faltantes).
- **Personalización según el SO:** Los pasos varían entre Windows, macOS y Linux (ej.: instalación de dependencias adicionales en sistemas basados en Unix).
- **Android Studio como aliado:** Más que un IDE, es un gestor centralizado para el SDK de Android, emuladores y plugins necesarios (Flutter/Dart), lo que simplifica el flujo de trabajo.
- **Documentación oficial como referencia:** Ante errores, la documentación de Flutter y Android Studio suele tener respuestas actualizadas, evitando soluciones obsoletas de foros.

Conocer bien las rutas donde se hace la instalación de las tecnologías para poder ubicar bien los path, a fin de evitar inconvenientes al momento de desarrollar.

Fijarse en las especificaciones de la pantalla del dispositivo que se esta emulando a fin de saber con que tipo de resolución se esta trabajando.

Referencias

- [1] B. Birch, *Flutter for Dummies*, 2nd ed., Hoboken: Wiley, 2022.
- [2] C. Lee, "Mobile UIs for Energy Apps", *Journal of Dev Tools*, vol. 12, no. 3, pp. 45-50, Jun. 2023.