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 cálcilo 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

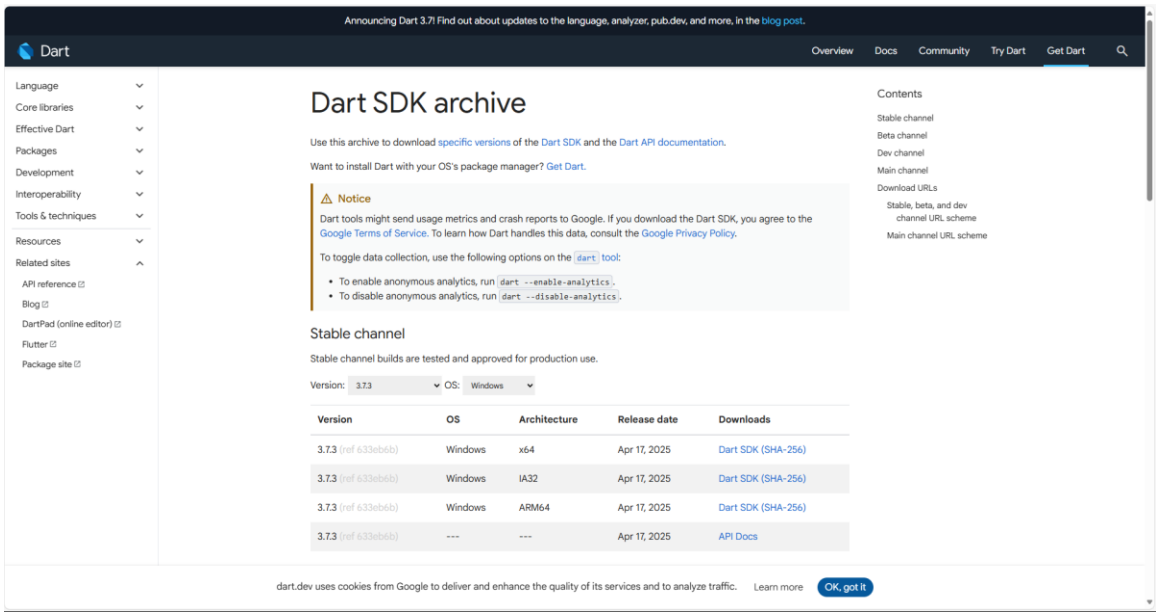
Procedimiento:

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

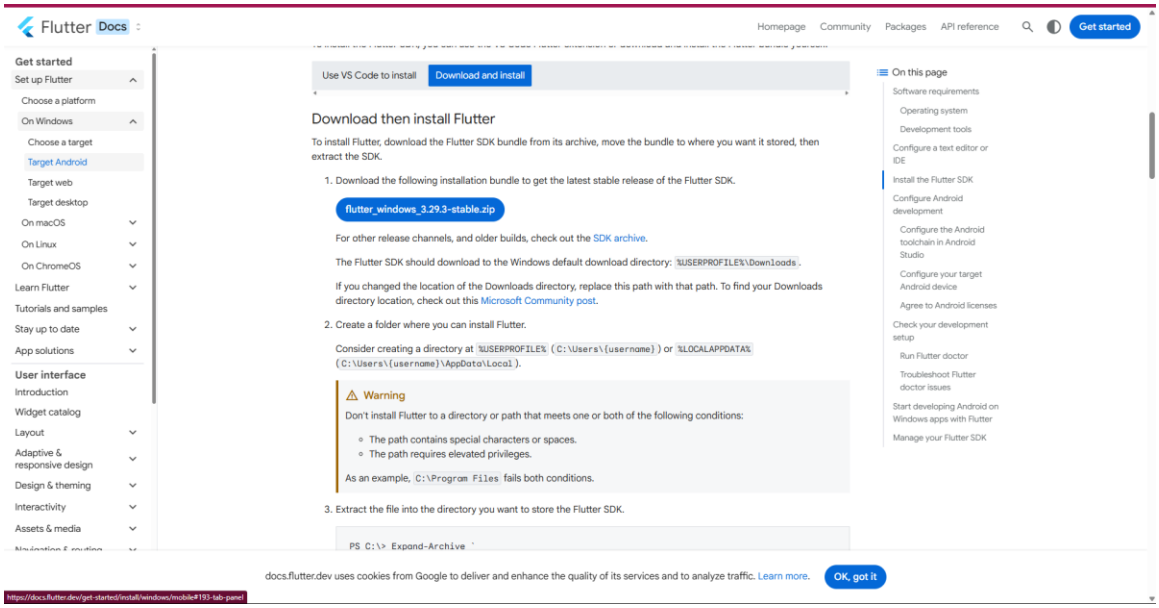
Instalación de Entorno de Desarrollo

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:



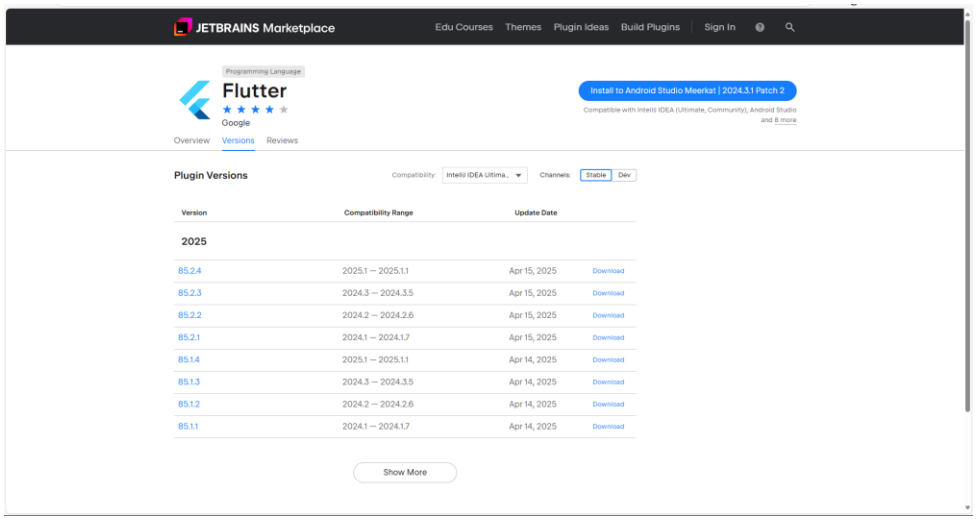
La instalación consiste en descomprimir el zip en alguna dirección del equipo. En este caso se realizó la instalación en C:\Program Files\dart-sdk

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

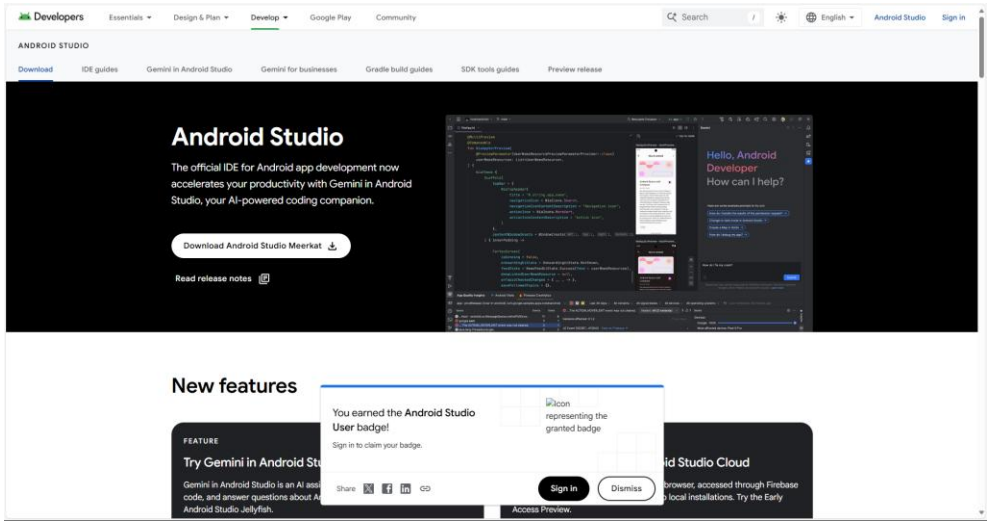


La instalación consiste en extraer el zip descargado en algún directorio que no contenga espacios en su nombre y no requiera de permisos de administrador para ser ejecutado. En este caso se realizó la instalación en C:\Users\PERSONAL\Desktop\Cris\flutter

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

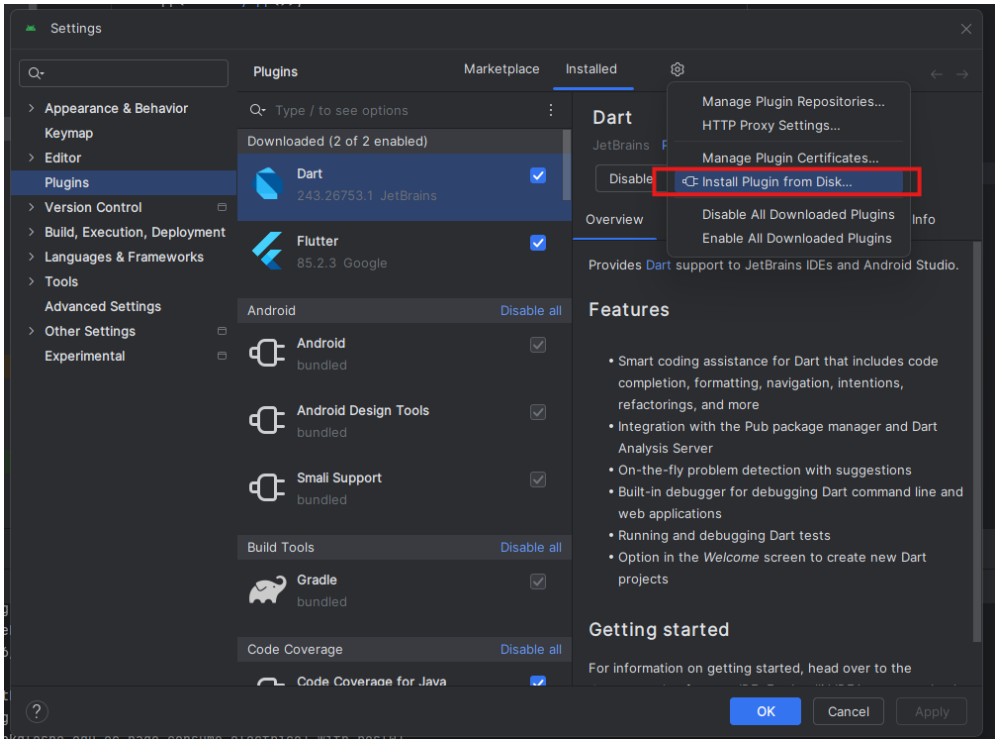


Posteriormente se realizó la descarga de Android Studio. Esto se consigue desde su página oficial:



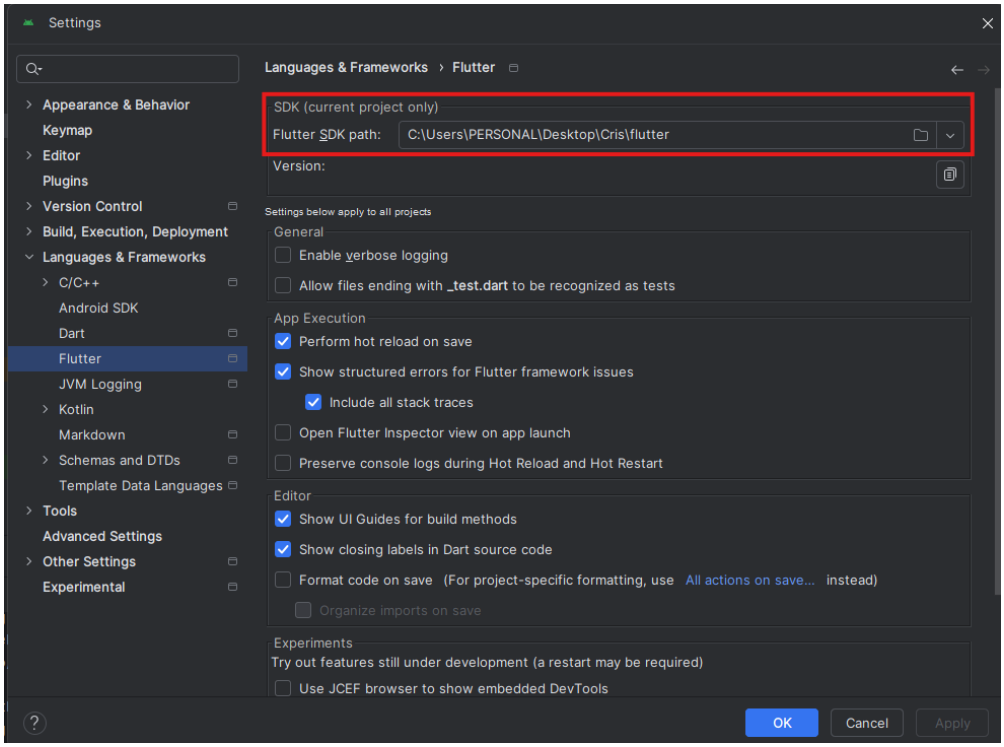
La instalación consiste en una instalación normal con un archivo .exe.

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

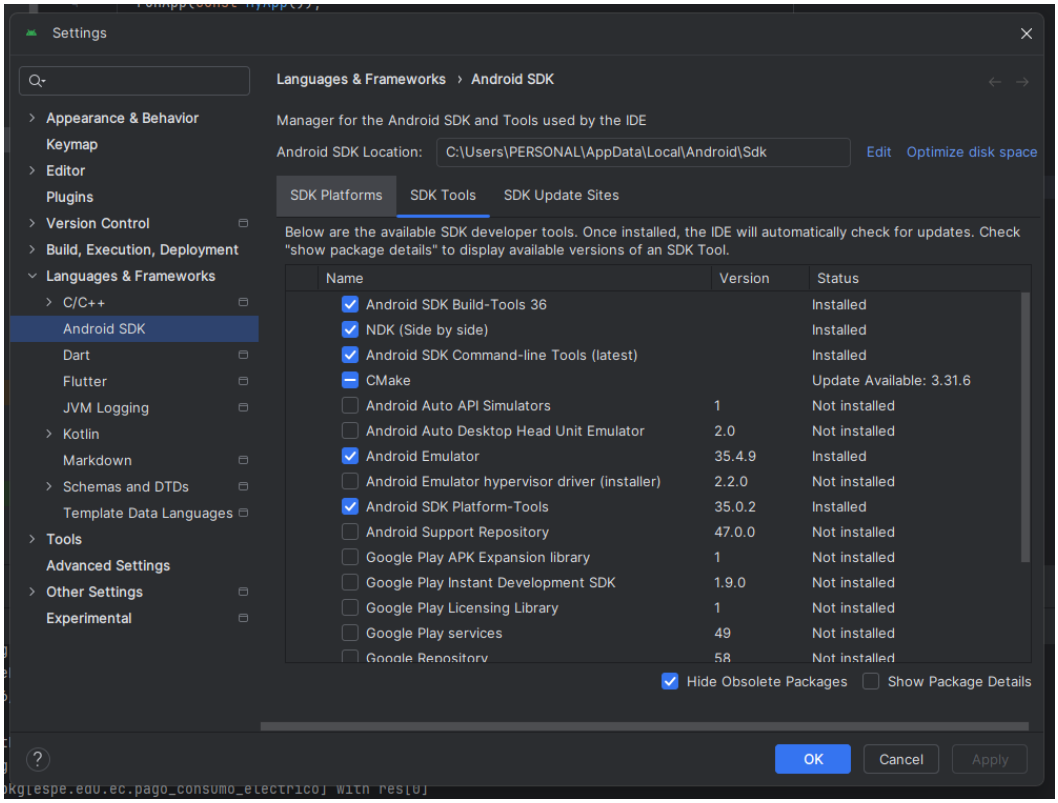


Se selecciona el zip y la instalación se concreta.

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.

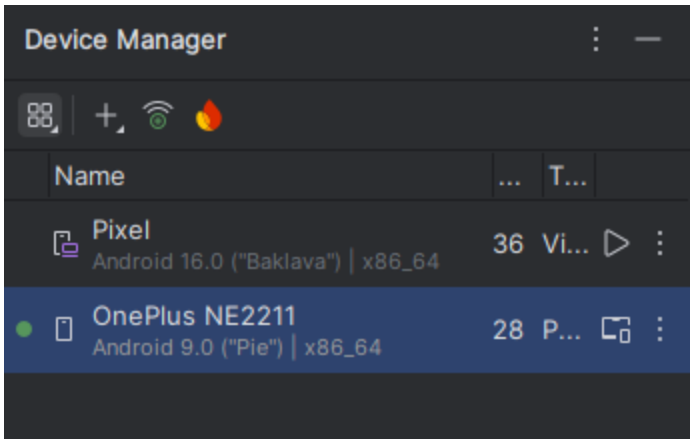
Una vez ejecutado BlueStacks, para que sea detectable para Android Studio hay que ejecutar los siguientes comandos:

- **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**:

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Calculadora CLS',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),  // ThemeData
      home: const HomeScreen(),
    );  // MaterialApp
  }
}
```

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

```dart
class _HomeScreenState extends State<HomeScreen> {
  // Variables para el cálculo de energía eléctrica
  double _kwConsumidos = 0.0;
  double _precioPorKW = 0.0;
  double _totalPago = 0.0;

  // Variables para el cálculo de artículo con descuento
  double _precioArticulo = 0.0;
  double _precioConDescuento = 0.0;
  double _precioFinal = 0.0;

  // Controladores para los campos de texto
  final TextEditingController _kwController = TextEditingController();
  final TextEditingController _precioKWController = TextEditingController();
  final TextEditingController _articuloController = TextEditingController();
```

Explicación:

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

Funciones de Cálculo

```dart
  void _calcularPagoLuz() {
    setState(() {
      _kwConsumidos = double.tryParse(_kwController.text) ?? 0.0;
      _precioPorKW = double.tryParse(_precioKWController.text) ?? 0.0;
      _totalPago = _kwConsumidos * _precioPorKW;
    });
  }

  void _calcularPrecioArticulo() {
    setState(() {
      _precioArticulo = double.tryParse(_articuloController.text) ?? 0.0;
      _precioConDescuento = _precioArticulo * 0.8; // 20% de descuento
      _precioFinal = _precioConDescuento * 1.15; // 15% de IVA
    });
  }
```

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)

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Calculadora CLS'),
    ),  // AppBar
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(20.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          // Sección de cálculo de energía eléctrica
          Card(...),  // Card

          const SizedBox(height: 20.0),

          // Sección de cálculo de artículo con descuento
          Card(...),  // Card
        ],
      ),  // Column
    ),  // SingleChildScrollView
  );  // Scaffold
}
}
```

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

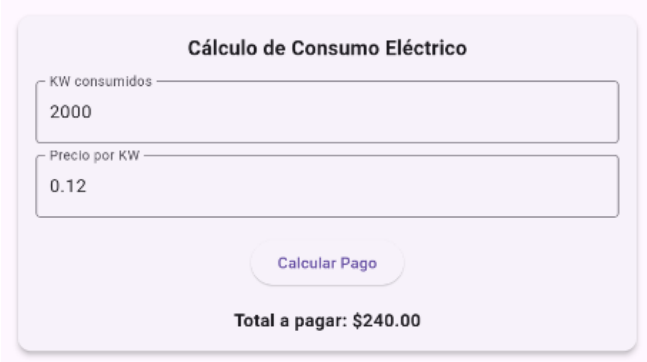
Cálculo de Artículo con Descuento e IVA

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

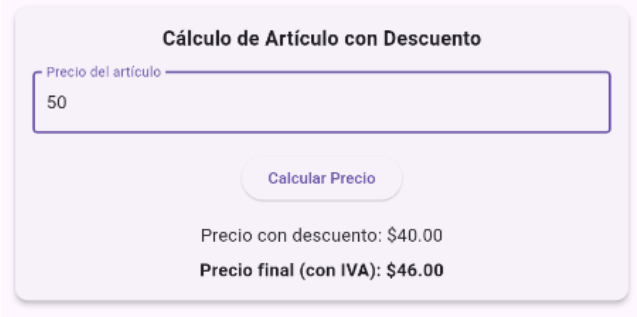
Interfaces

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

Cálculo de Consumo Eléctrico:



Cálculo de precio con descuento e impuestos



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 fn 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.