TRABAJO DE GRADO
Opción Seminario-Diplomado.

Seminario programación Mobile

Corporación Universitaria Remington.
Facultad de Ingeniería.
Ingeniería en Sistemas.

Juan Esteban Cuellar Gaviria
Jonatan Stick Campos.
Opción de Trabajo de grado Seminario-Diplomado.
2025.

Tabla de contenido

Resumen

la implementación de un sistema para el buen funcionamiento y gestión de los productos en una empresa en desarrollo o micro empresas casi que obligatorio ya que de esto dependería la optimización de tiempo y gastos en esta, flutter siendo un framework de código abierto permite crear app móviles en iOS y Android , además de aplicativos multiplataforma usando el lenguaje de programación Dart , al ser compilado de manera nativa este es más eficaz tanto para desarrolladores como para el usuario final.

Palabras Claves

Framework, Dart, Flutter, programación móvil, micro empresa

Pregunta orientadora de la búsqueda

¿Cómo puede el uso de Flutter y Dart ayudar a desarrollar una aplicación de inventario móvil eficiente y accesible para microempresas, impulsando la transformación digital y agilizando los procesos de gestión?

Contexto teórico

Es evidente que actualmente se está viviendo una transición tecnológica y que ahora más que nunca se debe hacer uso de estas herramientas innovadoras para no quedarse atrás, poder adaptarse y seguir operando los próximos años de manera efectiva y eficaz, a raíz de esto es muy importante para cada pequeña empresa tener un control asertivo y total de sus productos porque de esto depende el crecimiento de la empresa, cada producto que se pierda, cada reconteo y cada gasto innecesario puede marcar la diferencia entre progresar o no, por eso al tener una app en el teléfono que permita saber que productos y que cantidad hay en tiempo real optimiza así gastos, tiempo y además la satisfacción del cliente al cumplir con los requerimientos de las remisiones.

De acuerdo con Statista (2023), más del 85% de las personas en América Latina accede a internet principalmente desde un dispositivo móvil, lo que hace de las aplicaciones móviles una herramienta idónea para la digitalización de procesos en pequeñas empresas.

Metodología de búsqueda de la información

Para recopilar información, aplicamos una estrategia de recopilación de documentos basada en datos académicos como Google Académico.

Las palabras clave utilizadas fueron: framework Flutter, Dart, desarrollo app móvil, inventario y microempresa.

Sustentación teórica de la pregunta

Desarrollo movil con Flutter y Dart

Flutter, creado por Google, es un framework de código abierto para desarrollar aplicaciones multiplataforma a partir de un único código fuente. Su lenguaje principal es Dart, que se caracteriza por su compilación orientada a objetos, tipada y eficiente (Google, 2024).

Estudios recientes han demostrado que Flutter permite reducir hasta un 40% el tiempo de desarrollo comparado con proyectos nativos (Almeida et al., 2022). Además, ofrece widgets personalizables, una arquitectura flexible y una curva de aprendizaje moderada.

Desarrollo de la app

La aplicación desarrollada en este trabajo implementa las siguientes funcionalidades:
- Registro y categorización de productos.
- Control de entradas y salidas de inventario.
- Reportes básicos en tiempo real.
- Persistencia de datos mediante base de datos.

App móvil de inventario

Las aplicaciones de inventario permiten registrar, consultar y analizar información sobre existencias, mejorando la toma de decisiones en las empresas (García & Torres, 2019). En el caso de microempresas, estas aplicaciones deben ser ligeras, intuitivas y de bajo costo.

Digitalización en empresas en desarrollo

La implementación de tecnologías digitales en pequeñas organizaciones mejora la productividad, reduce costos y fomenta la competitividad (Martínez & López, 2020). Una de las áreas críticas es la gestión del inventario, donde el uso de aplicaciones móviles resulta estratégico.

Rol de las microempresas en la económica

Las microempresas representan más del 80% de las unidades productivas en América Latina, constituyéndose en actores fundamentales para el empleo y el desarrollo económico (CEPAL, 2021). Sin embargo, enfrentan retos de digitalización y gestión administrativa.

Figuras y tablas

Figura 1.Flutter

Segun Google (2018),Flutter es un framework de desarrollo de aplicaciones multiplataforma creado por Google, lanzado en 2017. Está diseñado para construir interfaces de usuario de alta calidad desde una única base de código, permitiendo compilar aplicaciones nativas para Android, iOS, web y escritorio. Su lenguaje de programación base es Dart, también desarrollado por Google, que se caracteriza por su rapidez, facilidad de aprendizaje y capacidad de compilar a código nativo.

Flutter consta principalmente de:

Widgets: componentes reutilizables que permiten construir interfaces visuales de manera flexible y personalizada.

Motor de renderizado (Skia): encargado de dibujar los elementos gráficos directamente en la pantalla, lo que ofrece gran rendimiento y fluidez.

Hot Reload: característica que permite visualizar cambios en el código de manera inmediata sin necesidad de recompilar toda la aplicación.

SDK (Software Development Kit): herramientas y librerías que facilitan la integración de APIs, manejo de bases de datos, pruebas y despliegue de aplicaciones.

Ventajas de la programación móvil actualmente

La programación móvil es una de las áreas de mayor crecimiento en la ingeniería de software, debido al aumento exponencial de dispositivos inteligentes. Sus principales ventajas son:

Accesibilidad y masificación: millones de personas usan smartphones diariamente, lo que convierte a las aplicaciones móviles en un canal directo con el usuario.

Multiplataforma: frameworks como Flutter permiten desarrollar aplicaciones para varios sistemas operativos desde un único código, reduciendo costos y tiempos.
Rendimiento cercano al nativo: gracias a motores como Skia en Flutter o compilaciones en tiempo real, las apps son rápidas y fluidas.

Ecosistema digital: las apps se integran con servicios en la nube, IoT, inteligencia artificial y sistemas empresariales, aumentando la competitividad de las empresas.
Innovación constante: actualizaciones frecuentes en sistemas operativos y librerías promueven la creación de soluciones modernas, interactivas y seguras.

Normatividad en el desarrollo de apps móviles

De acuerdo con ISO/IEC (2011) y como lo señala el Parlamento Europeo (2016), el desarrollo móvil debe regirse por buenas prácticas y normativas internacionales, que garantizan la seguridad, calidad y accesibilidad de las aplicaciones:
Normas ISO/IEC 25010 – Modelo de calidad de software que evalúa aspectos como usabilidad, seguridad, mantenibilidad y rendimiento.

Norma ISO/IEC 27001 – Relacionada con la seguridad de la información, protege datos sensibles en aplicaciones móviles.

Políticas de tiendas oficiales (Google Play y App Store): regulaciones sobre privacidad, permisos, accesibilidad y contenido.

RGPD (Reglamento General de Protección de Datos, en Europa) y Leyes de Habeas Data (en países de Latinoamérica, como Colombia):
normativas que aseguran la protección y el uso adecuado de los datos personales de los usuarios.

Accesibilidad (WCAG 2.1): estándares internacionales que buscan que las aplicaciones puedan ser usadas por personas con discapacidad.

Instalacion de flutter

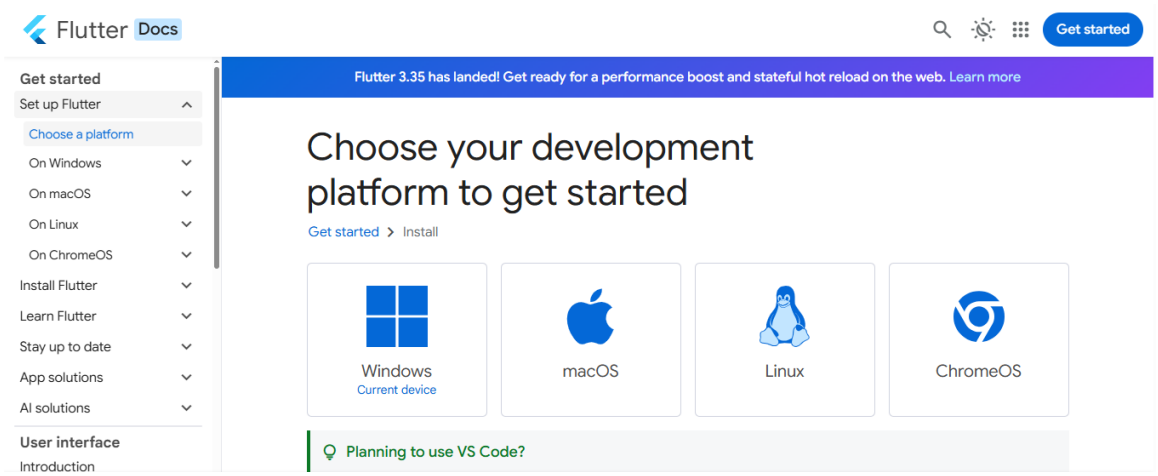


Figura 2.seleccion de plataforma

Ingresamos a la pagina oficial de Flutter tocamos en get started y seleccionamos la plataforma de desarrollo como Windows , macOS, Linux o

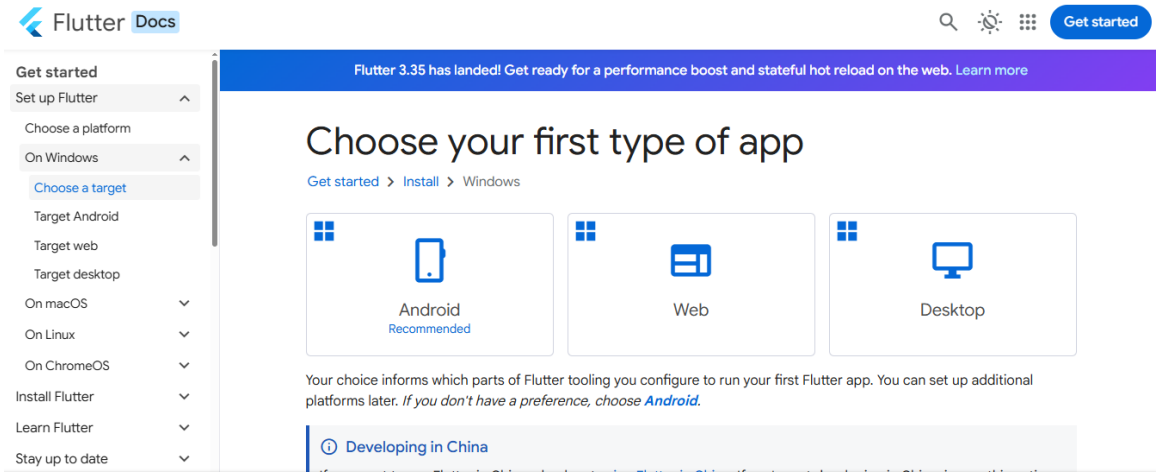


Figura 3.seleccion tipo de app

Aquí vamos a seleccionar el tipo de app que vamos a desarrollar en este caso para android mobile

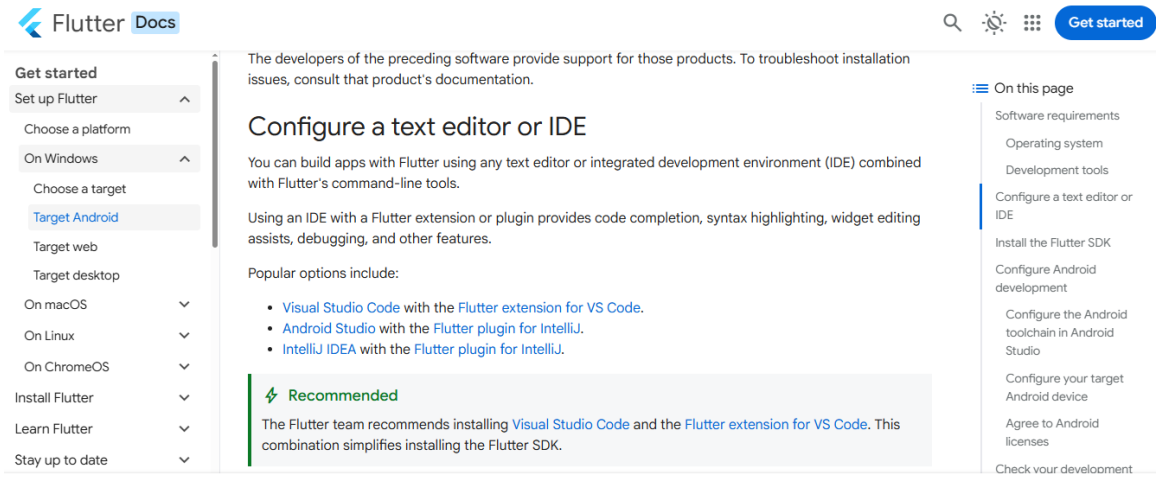


Figura 4.Configure el editor de texto

Vamos a instalar uno de las tres opciones de editor de texto en este caso es recomendable visual studio code con su respectiva extensión para VS Code

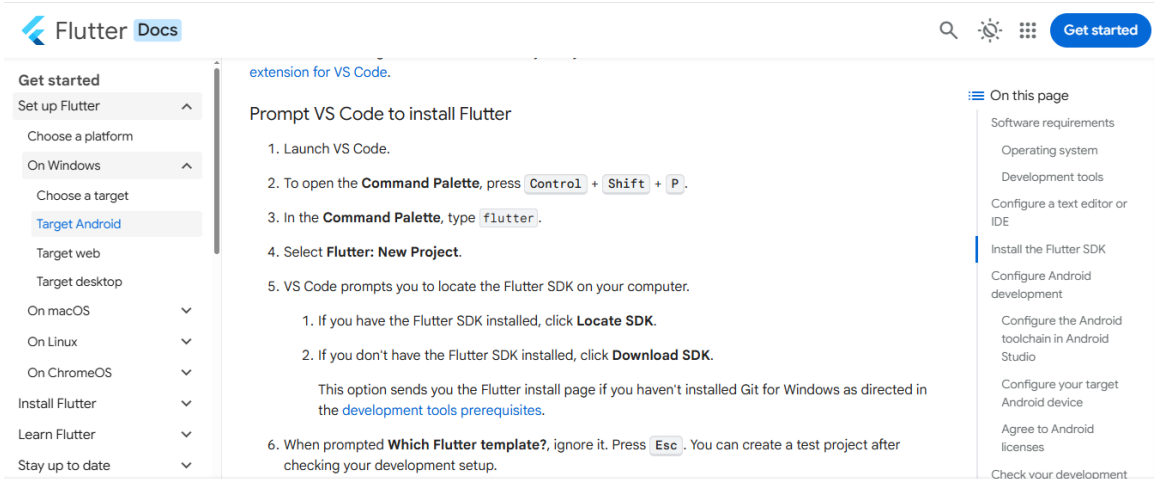
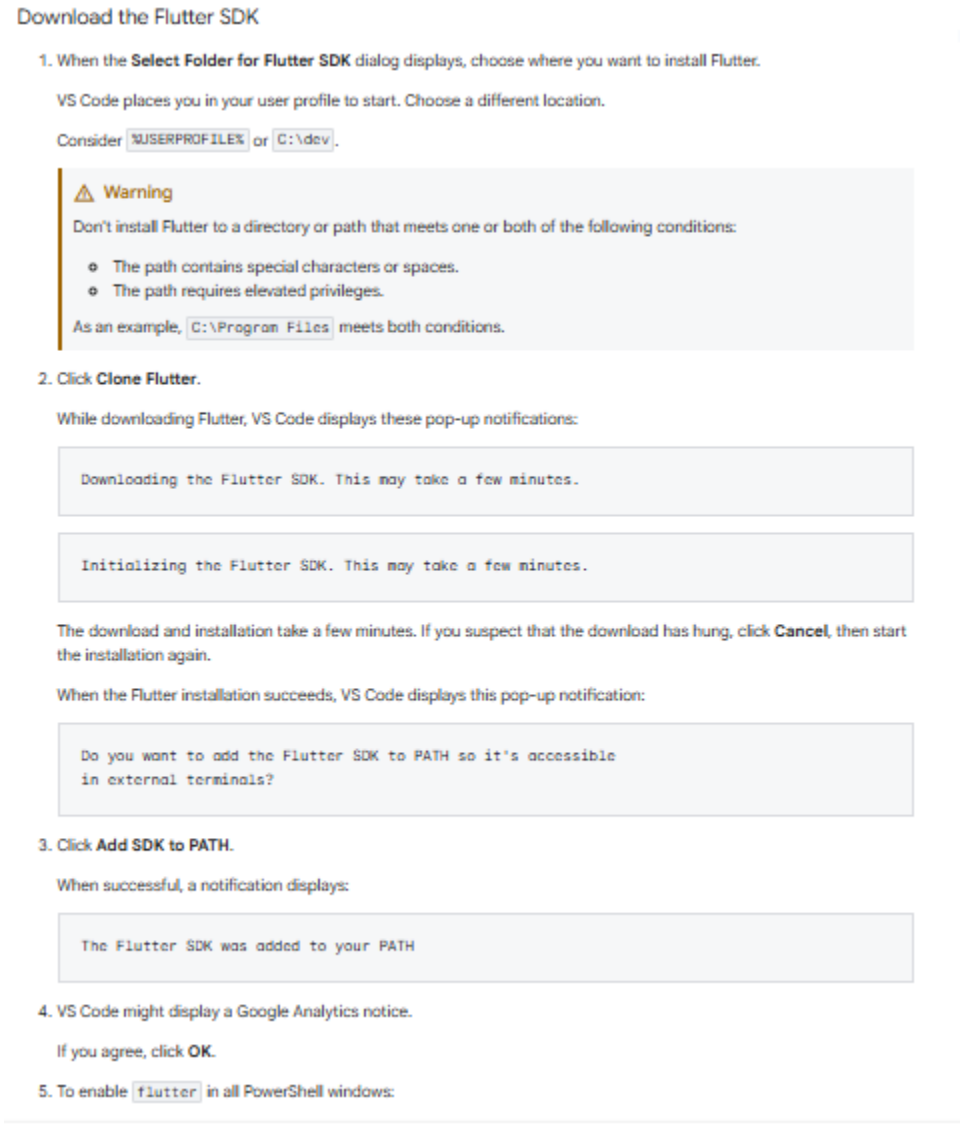


Figura 5.Solicita VS Code para instalar Flutter

Existen dos caminos el primero es simplemente desde VS Code hacer la instalación mediante los pasos en la imagen o hacerlo descargando el sdk e ingresando al PATH



Download the Flutter SDK

1. When the **Select Folder for Flutter SDK** dialog displays, choose where you want to install Flutter.

   VS Code places you in your user profile to start. Choose a different location.

   Consider `%USERPROFILE%` or `C:\dev`.

   > ⚠ 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` meets both conditions.

2. Click **Clone Flutter**.

   While downloading Flutter, VS Code displays these pop-up notifications:

   ```
   Downloading the Flutter SDK. This may take a few minutes.
   ```

   ```
   Initializing the Flutter SDK. This may take a few minutes.
   ```

   The download and installation take a few minutes. If you suspect that the download has hung, click **Cancel**, then start the installation again.

   When the Flutter installation succeeds, VS Code displays this pop-up notification:

   ```
   Do you want to add the Flutter SDK to PATH so it's accessible
   in external terminals?
   ```

3. Click **Add SDK to PATH**.

   When successful, a notification displays:

   ```
   The Flutter SDK was added to your PATH
   ```

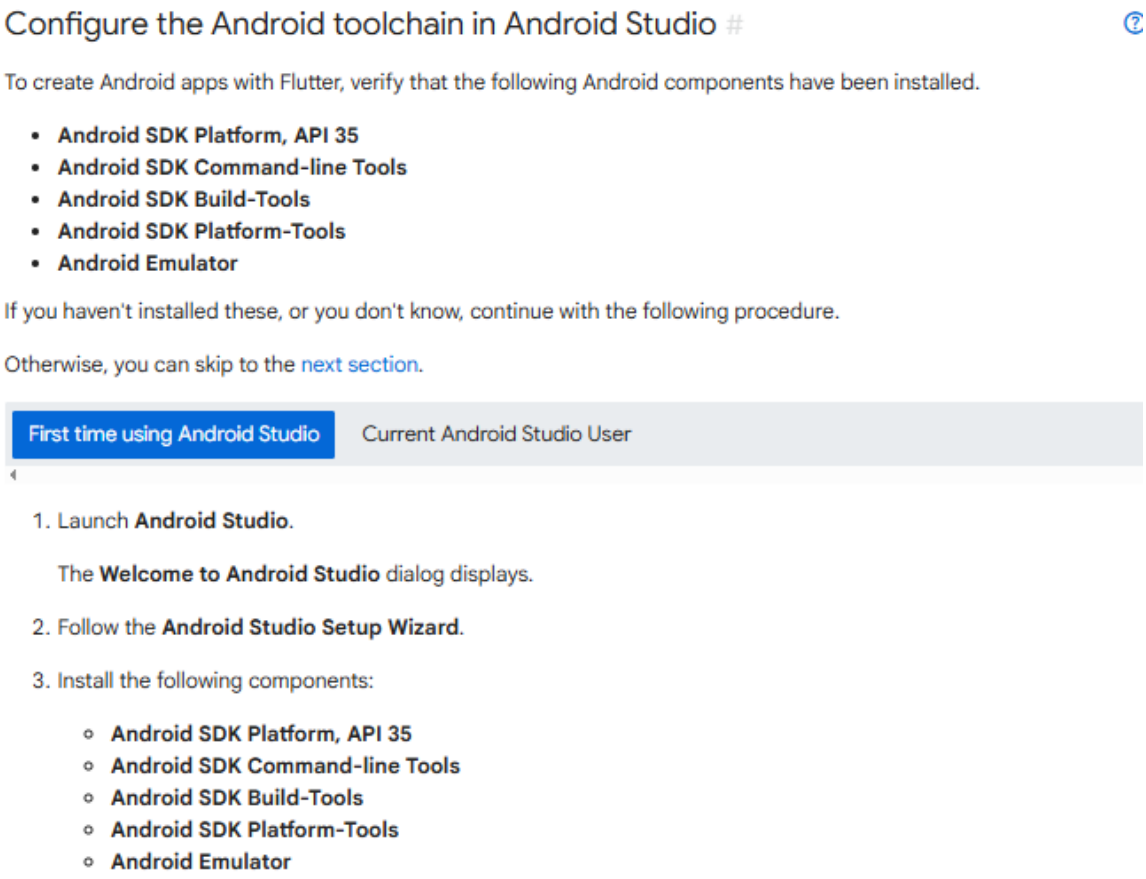
4. VS Code might display a Google Analytics notice.

   If you agree, click **OK**.

5. To enable `flutter` in all PowerShell windows:

Figura 6.Instalacion de SDK

Solo sigue el paso a paso desde VS Code para realizar la instalación del SDK con el cual obtendrás las librerías, podrás compilar el código y ejecutar entornos de desarrollo



Configure the Android toolchain in Android Studio

To create Android apps with Flutter, verify that the following Android components have been installed.

- **Android SDK Platform, API 35**
- **Android SDK Command-line Tools**
- **Android SDK Build-Tools**
- **Android SDK Platform-Tools**
- **Android Emulator**

If you haven't installed these, or you don't know, continue with the following procedure.

Otherwise, you can skip to the next section.

First time using Android Studio | Current Android Studio User

1. Launch **Android Studio**.

   The **Welcome to Android Studio** dialog displays.

2. Follow the **Android Studio Setup Wizard**.

3. Install the following components:

   - **Android SDK Platform, API 35**
   - **Android SDK Command-line Tools**
   - **Android SDK Build-Tools**
   - **Android SDK Platform-Tools**
   - **Android Emulator**

Figura 7.Configuracion de desarrollo

El Android toolchain es el ecosistema de herramientas que permite transformar el código fuente escrito por el programador en una aplicación ejecutable en dispositivos móviles o emuladores.

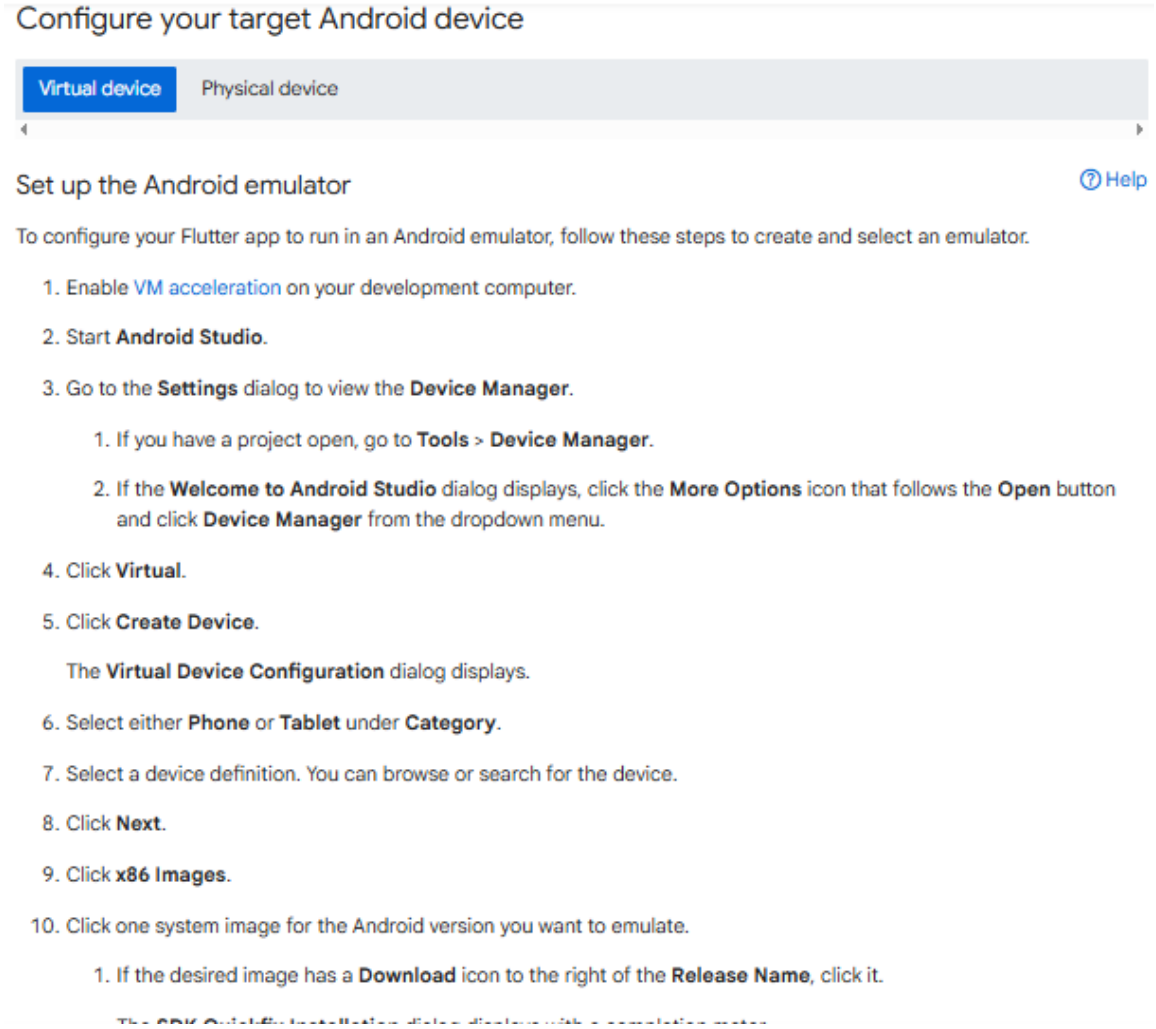


Figura 7.Configuracion de dispositivo

Existen dos formas de realizar esta configuración, con un dispositivo físico o uno virtual, es preferible realizarlo con un equipo fisco para esto es necesario tener las opciones de desarrollador activas y activar la depuración por USB.

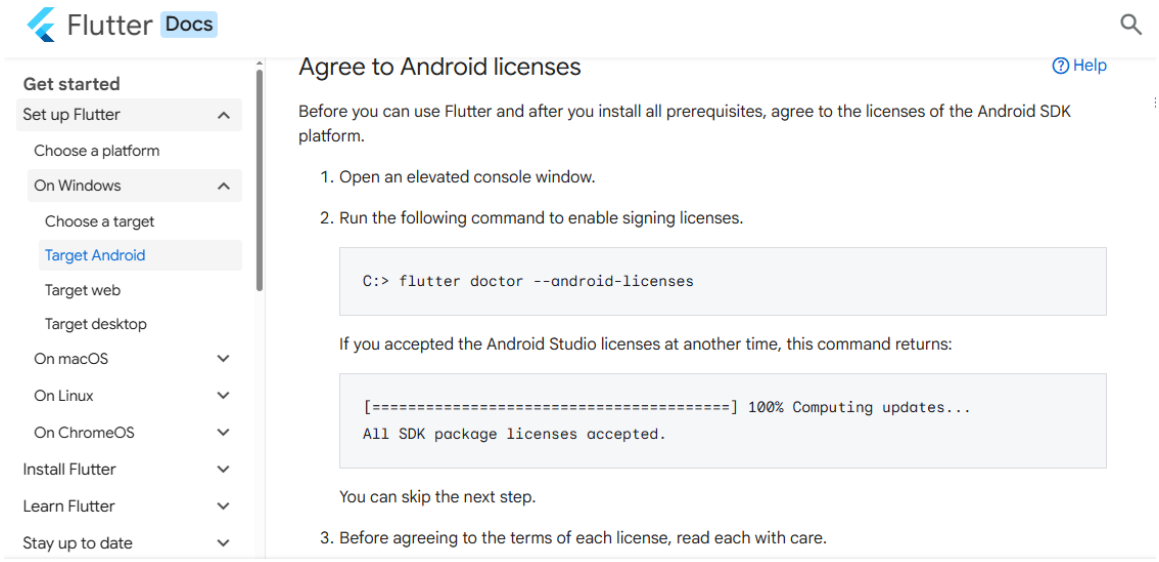


Figura 8.Acepta licencias de Android

Básicamente son acuerdos legales establecidos por Google que regulan el uso del software, las bibliotecas y las APIs de Android, se aceptan estando dentro de la terminal de comandos en VS Code.

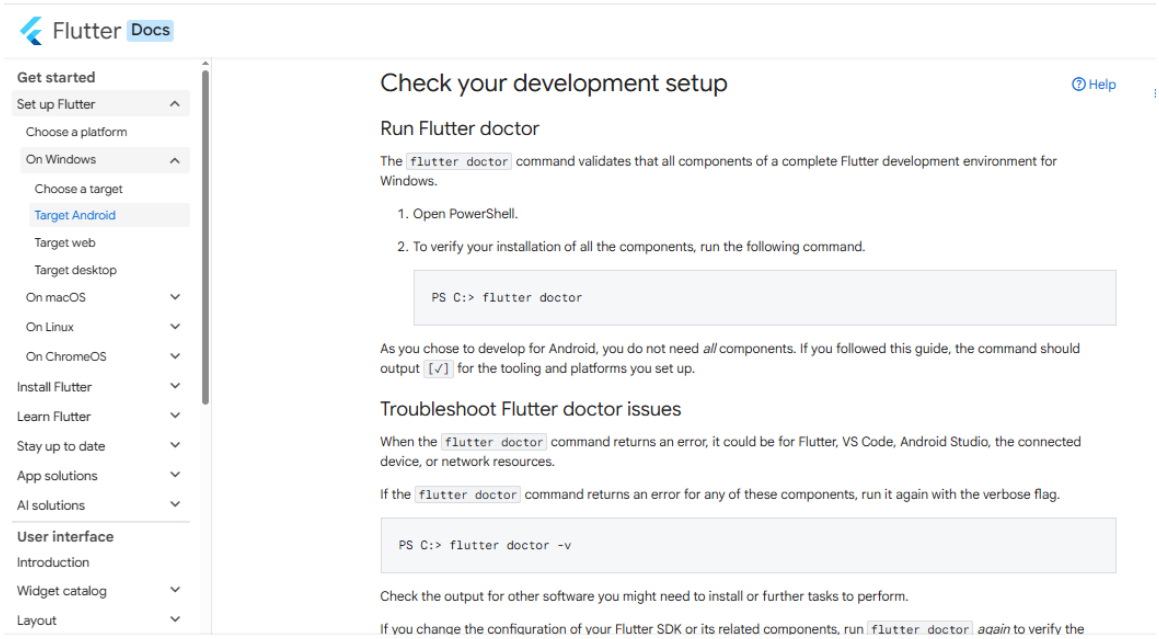


Figura 9.Chequea entorno de desarrollo

Aquí podrás verificar si todo quedo bien instalado o si falta alguna instalación ejecutando en la terminal de comandos el comando flutter doctor podrás ver si aun falta algo o si todo salió bien

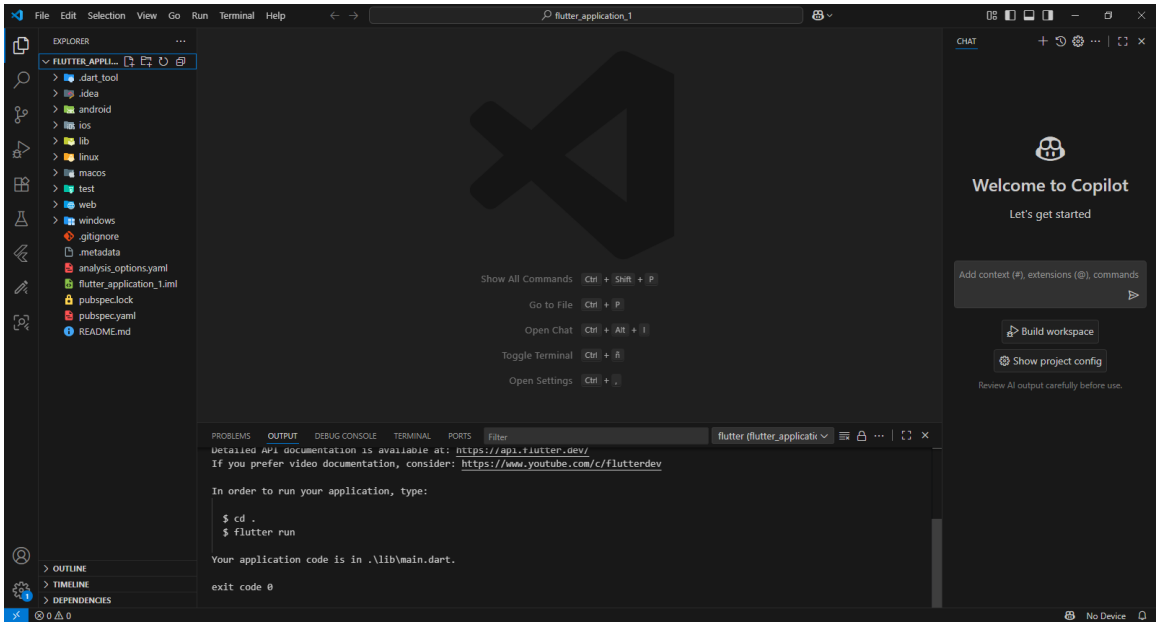


Figura 10.Entorno de desarollo

Así deberá verse el entorno de desarrollo listo para programar con sus respectivas carpetas

Inventario YA

A lo largo de mi experiencia laboral he notado que la mayoría de micro empresas o empresas en desarrollo no toman muy en cuenta el tema de tener total control y registro de todos sus productos generando así compras innecesarias, re conteos , reprocesos y un pequeño drenaje que a simple vista no es perceptible pero que puede marcar la diferencia en llegar al existo pronto o mantenerse en el tiempo , el orden y control es clave para determinar la rotación y fluyo de cada producto y maximizar tiempo y reducir gastos ,por eso un inventario interno que permita llevar el control y registro de todo en tiempo real al alcance de un par de clics y desde el teléfono pude hacer la diferencia y facilitar muchas tareas para el empresario y sus colaboradores.

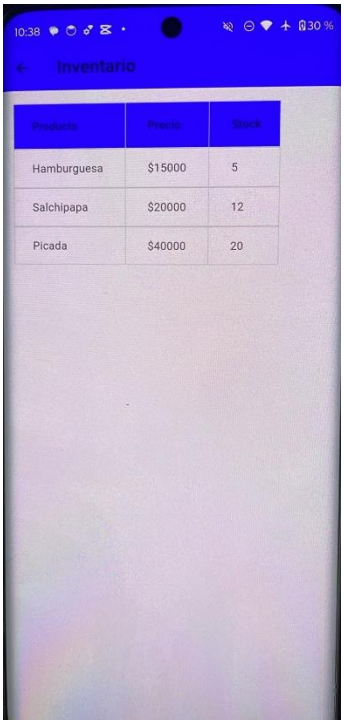


Figura 11.Inventario

En este apartado es en donde se podrá solo visualizar los productos y cantidad que hay permitiendo solo dar permisos para que lo colaboradores de cocina puedan verlos

Figura 11.Productos

Desde este apartado es en donde se podrá agregar, eliminar o editar cualquier producto

```dart
// PANTALLA DE INVENTARIO
class InventarioPage extends StatelessWidget {
  const InventarioPage({super.key});
  @override
  Widget build(BuildContext context) {
    // Lista simulada de productos
    final List<Map<String, dynamic>> productos = [
      {
        "nombre": "Hamburguesa",
        "precio": 15000,
        "cantidad": 5,
      },
      {
        "nombre": "Salchipapa",
        "precio": 20000,
        "cantidad": 12,
      },
      {
        "nombre": "Picada",
        "precio": 40000,
        "cantidad": 20,
      },
    ];

    return Scaffold(
      appBar: AppBar(
        title: const Text("Inventario",style: TextStyle(fontWeight: FontWeight.bold
        backgroundColor: const Color.fromARGB(255, 0, 4, 255),
      ), // AppBar
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: DataTable(
```

Figura 12.Codigo de inventario

Básicamente para la creación de la pantalla del inventario y que se vea en forma de tabla se hace uso de lo siguiente:

-class InventarioPage extends StatelessWidget:

Define un widget que no mantiene estado.

-const InventarioPage({super.key}):

Constructor const

-Widget build(BuildContext context):

Método que devuelve la estructura visual (árbol de widgets) de esta pantalla.

-final List<Map<String, dynamic>> productos = [...]:

Lista local con 3 productos. Cada producto es un Map con claves "nombre", "precio" y "cantidad".

En donde podremos crear de manera local nuestros productos que simularían la base de datos a usar, luego de esto daremos forma a nuestra tabla de la siguiente manera:

```
class InventarioPage extends StatelessWidget {
  Widget build(BuildContext context) {
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: DataTable(
        headingRowColor: WidgetStateProperty.all(const Color.fromARGB(255, 4, 0,
        border: TableBorder.all(color: Colors.grey.shade300, width: 1),
        columns: const [
          DataColumn(
            label: Text(
              "Producto",
              style: TextStyle(fontWeight: FontWeight.bold),
            ), // Text
          ), // DataColumn
          DataColumn(
            label: Text(
              "Precio",
              style: TextStyle(fontWeight: FontWeight.bold),
            ), // Text
          ), // DataColumn
          DataColumn(
            label: Text(
              "Stock",
              style: TextStyle(fontWeight: FontWeight.bold),
            ), // Text
          ), // DataColumn
        ],
        rows: productos.map((producto) {
          return DataRow(
            cells: [
              DataCell(Text(producto["nombre"])),
              DataCell(Text("\$${producto["precio"]}")),
```

Figura 13.Codigo de inventario1.1

- Scaffold(...)

Estructura general de la pantalla: contiene appBar y body.

- AppBar(title: const Text("Inventario"), backgroundColor: ...)

Barra superior con título y color.

- SingleChildScrollView(padding: ..., child: DataTable(...))El SingleChildScrollView permite scroll si la tabla ocupa más espacio vertical del disponible.

- DataTable(...) Widget que renderiza la tabla.

- Border: TableBorder.all(color: ..., width: 1) dibuja bordes alrededor de la tabla.

- Columns: const [DataColumn(label: Text("Producto")), ...] define las columnas (cada etiqueta es un widget).

- Rows: productos.map((producto) { return DataRow(cells: [DataCell(...), ...]); }).toList() Mapea cada Map de productos a una DataRow con DataCells. Cada DataCell puede contener texto, iconos y tiene la opción onTap para interacción.

Formato de celdas:

- Text("\$${producto["precio"]}"):

interpolación para mostrar el precio con signo $.

- Producto["cantidad"].toString()_

convierte la cantidad a String.

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

  @override
  Widget build(BuildContext context) {
    // Lista simulada de productos
    final List<Map<String, dynamic>> productos = [
      {
        "nombre": "hamburguesa",
        "precio": 15000,
        "cantidad": 5,
        "imagen":
            "https://cdn-icons-png.flaticon.com/512/3075/3075977.png"
      },
      {
        "nombre": "salchipapa",
        "precio": 20000,
        "cantidad": 12,
        "imagen":
            "https://cdn-icons-png.flaticon.com/512/1046/1046784.png"
      },
      {
        "nombre": "picada",
        "precio": 40000,
        "cantidad": 20,
        "imagen":
            "https://cdn-icons-png.flaticon.com/512/1046/1046784.png"
      },
    ];

return Scaffold(
  appBar: AppBar(
```

Figura 14.Codigo de productos

Es una lista (List) de mapas (Map<String, dynamic>).

Cada producto tiene:

- nombre: texto.
- precio: número entero.
- cantidad: stock disponible.
- imagen: URL a una imagen.

```
return Scaffold(
  appBar: AppBar(
    title: const Text(
      "Productos",
      style: TextStyle(fontWeight: FontWeight.bold),
    ), // Text
    backgroundColor: const Color.fromARGB(255, 0, 38, 255),
  ), // AppBar
  body: Container(
    color: Colors.grey[100],
    child: ListView.builder(
      padding: const EdgeInsets.all(12),
      itemCount: productos.length,
      itemBuilder: (context, index) {
        final producto = productos[index];
        return Card(
          elevation: 6,
          margin: const EdgeInsets.symmetric(vertical: 10),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(16),
          ), // RoundedRectangleBorder
          child: Padding(
            padding: const EdgeInsets.all(12),
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                // Imagen del producto
                Container(
                  width: 80,
                  height: 80,
```

Figura 15.Codigo de productos1.1

Este es el cuerpo pricipal de nuestra pantalla de productos la cual consta de:

- Scaffold da la estructura básica de la pantalla.

- AppBar tiene un título y un color azul.

Container con fondo gris.

- ListView.builder: crea una lista de forma dinámica y eficiente.

- itemCount: número de productos.

- itemBuilder: función que construye cada tarjeta de producto.
- Card: tarjeta con sombra, bordes redondeados y margen vertical.
- Dentro hay un Row (fila horizontal) con 3 partes:
- Imagen del producto.

- Información.
-
- Botones de acción.

```
class ProductosPage extends StatelessWidget {
  Widget build(BuildContext context) {
                height: 80,
                decoration: BoxDecoration(
                  color: Colors.grey[200],
                  borderRadius: BorderRadius.circular(12),
                  image: DecorationImage(
                    image: NetworkImage(producto["imagen"]),
                    fit: BoxFit.contain,
                  ), // DecorationImage
                ), // BoxDecoration
              ), // Container
              const SizedBox(width: 16),

              // Info del producto
              Expanded(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      producto["nombre"],
                      style: const TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                        color: Colors.black87,
                      ), // TextStyle
                    ), // Text
                    const SizedBox(height: 6),
                    Text(
                      "Precio: \$${producto["precio"]}",
                      style: TextStyle(
                        fontSize: 15,
                        color: Colors.green[700]
```

Figura 16.Codigo de productos1.2

Para dar las propiedades y forma a las tarjetas o contenedores usamos lo siguiente:

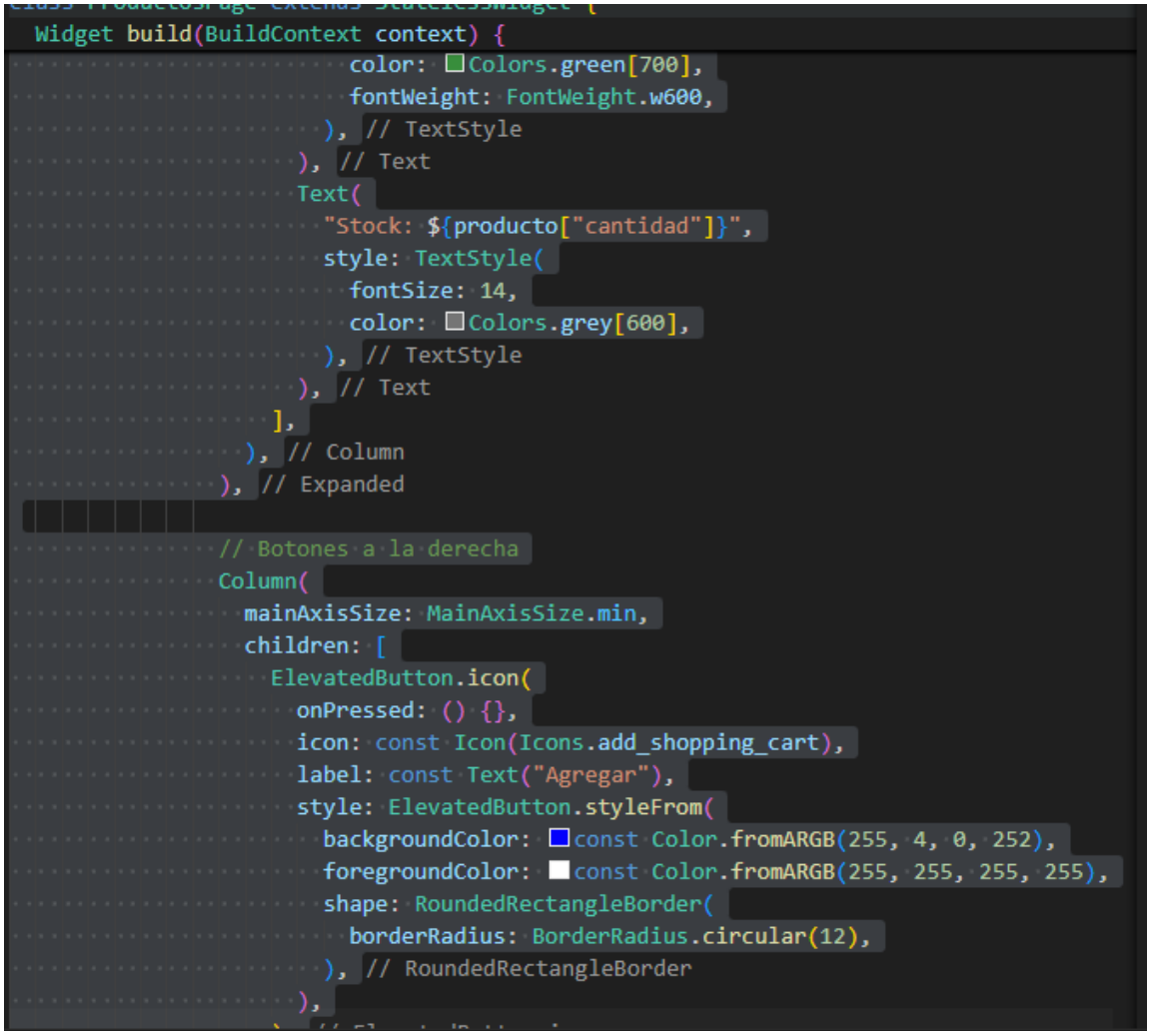
- Container cuadrado con fondo gris y esquinas redondeadas.

- Muestra una imagen desde internet (NetworkImage).

- fit: BoxFit.contain asegura que la imagen encaje sin recortarse.
- `Expanded`: ocupa todo el espacio disponible entre la imagen y los botones.
- `Column`: coloca nombre, precio y stock en vertical.

Estilos aplicados:

- Nombre → grande y en negrita.
- Precio → color verde y resaltado.
- Stock → texto gris claro.



```
  Widget build(BuildContext context) {
                        color: Colors.green[700],
                        fontWeight: FontWeight.w600,
                      ), // TextStyle
                    ), // Text
                    Text(
                      "Stock: ${producto["cantidad"]}",
                      style: TextStyle(
                        fontSize: 14,
                        color: Colors.grey[600],
                      ), // TextStyle
                    ), // Text
                  ],
                ), // Column
              ), // Expanded

              // Botones a la derecha
              Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  ElevatedButton.icon(
                    onPressed: () {},
                    icon: const Icon(Icons.add_shopping_cart),
                    label: const Text("Agregar"),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: const Color.fromARGB(255, 4, 0, 252),
                      foregroundColor: const Color.fromARGB(255, 255, 255, 255),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(12),
                      ), // RoundedRectangleBorder
                    ),
```

Figura 17.Codigo de productos1.3

Aqui finalmente colocaremos y ordenaremos todos los elementos que van dentro de el contenedor o tarjeta usando los siguiente:

- Columna con 3 botones verticales.
- mainAxisSize: MainAxisSize.min → la columna ocupa el espacio justo, no todo el alto.
- Todos los botones tienen el mismo estilo (azul con texto blanco).
- Actualmente, los onPressed están vacíos ({}), por lo que no hacen nada.

- Los 3 botones usan el mismo ícono (Icons.add_shopping_cart), aunque lo ideal sería:
- Agregar
- Eliminar
- Editar

Asi finalmente realizariamos la construccion en una primera instancia de nuestro inventario usando widgets y celdas vistas en la asesoria paso a paso y explicando el funcionamiento de cada linea de codigo alli.

Conclusiones.

Actualmente es claro que el uso de las nuevas tecnologías es totalmente necesario para así poder tener mayor control y optimización de todos los procesos dentro de una empresa, en este caso el uso del framework flutter facilita el desarrollo e implementación de nuevos métodos los cuales permiten dar un mejor funcionamiento dentro las empresas dejando en evidencia que estamos en una era tecnológica en la cual debemos adaptarnos para persistir en el tiempo.

Con el desarrollo de esta app se busca principalmente poder brindar una mejor organización de todos los insumos de una empresa en desarrollo ya que son estas las que

no se fijan en ese pequeño detalle tan importante y puede dar mucho mas control de todo al empresario dándole mayor tranquilidad, mayor manejo y mejor experiencia al usuario final siendo esto posible desde cualquier lugar.

En conclusión, al automatizar cualquier proceso se consigue la reducción de gastos, tiempo y deja como resultado un desarrollo y consolidación más prematuro en las empresas que están en pro de desarrollo. Las empresas las cuales se resistan al cambio o mejoramiento se irán quedando atrás y perdiendo en el tiempo o simplemente se quedarán en un punto de estancamiento.

Referencias

Almeida, F., Silva, L., & Gomes, P. (2022). *Cross-platform mobile development: An analysis of Flutter framework*. Journal of Software Engineering, 14(2), 45–57.

CEPAL. (2021). *El papel de las microempresas en América Latina*. Naciones Unidas.

García, J., & Torres, P. (2019). *Sistemas de información aplicados a la gestión de inventarios*. Revista de Innovación Empresarial, 8(3), 55–70.

Google. (2024). *Flutter Documentation*. Recuperado de https://flutter.dev

Martínez, R., & López, S. (2020). *Transformación digital en pymes: Retos y oportunidades*. Revista Iberoamericana de Tecnología, 15(1), 33–49.