Trabajo de Grado

Aplicación Mobile Que Hambre Burgers

Corporación Universitaria Remington.

Facultad de Ingeniería.

Ingeniería de Sistemas.

Jennifer Yurianny Sánchez Holguín.

Jhair Steven García Torres.

Tutor: Jonathan Stick Campos Núñez.

Opción de Trabajo de grado Seminario-Diplomado

2025

Contenido

Indice de Imagenes

Resumen

Flutter es un framework de código abierto desarrollado por Google que permite a los desarrolladores crear aplicaciones multiplataforma, incluyendo móviles (iOS y Android), web y de escritorio, usando una única base de código y el lenguaje de programación Dart. Al compilarse a código nativo, Flutter ofrece un rendimiento superior y una experiencia de usuario consistente, reduciendo el tiempo y el costo de desarrollo.

Palabras Claves

Flutter, Dart, Aplicación móvil.

Glosario

API (Application Programming Interface): Conjunto de funciones, métodos y definiciones que permiten la comunicación entre diferentes aplicaciones o sistemas. Es clave para integrar servicios externos como pasarelas de pago o mapas en aplicaciones móviles.

(Pressman, 2014).

Carrito de compras: Funcionalidad en aplicaciones de comercio electrónico que permite a los usuarios almacenar y gestionar los productos seleccionados antes de realizar el pago.

(Turban, 2018).

Dart: Lenguaje de programación creado por Google, orientado a objetos, con tipado opcional y soporte para programación asíncrona, utilizado principalmente en Flutter.

(Google, 2025).

Flutter: Framework de código abierto desarrollado por Google para crear aplicaciones móviles, web y de escritorio de forma nativa, utilizando un único código base.

(Google, 2025).

Framework: Entorno de trabajo que proporciona componentes predefinidos, librerías y herramientas que agilizan el desarrollo de aplicaciones.

(Sommerville, 2011).

Hot Reload: Funcionalidad de Flutter que permite recargar los cambios en el código de una aplicación sin necesidad de detener la ejecución, optimizando el desarrollo y las pruebas.

(Google, 2025).

Interfaz de usuario (UI): Medio a través del cual los usuarios interactúan con una aplicación. En Flutter, la interfaz está compuesta por widgets.

(Shneiderman, 2017).

Login: Proceso de autenticación que permite a un usuario acceder a una aplicación introduciendo sus credenciales (usuario y contraseña).

(Stallings, 2018).

Mobile Commerce (m-commerce): Modalidad de comercio electrónico que se realiza a través de dispositivos móviles, como smartphones o tablets.

(Turban, 2018).

Navegación: Proceso que permite al usuario desplazarse entre diferentes pantallas o secciones dentro de una aplicación. En Flutter, se gestiona con el widget Navigator.

(Google, 2025).

StatelessWidget: Tipo de widget en Flutter que no mantiene estado interno y se utiliza para elementos estáticos en la interfaz.

(Google, 2025).

StatefulWidget: Tipo de widget en Flutter que puede cambiar su estado durante la ejecución, adecuado para elementos dinámicos e interactivos.

(Google, 2025).

Widget: Unidad básica de construcción en Flutter, todo elemento visible en la aplicación (botones, imágenes, textos) se define como widget.

(Google, 2025).

Objetivo General

Desarrollar una aplicación móvil multiplataforma para la gestión de pedidos de comida rápida utilizando Flutter, con el fin de optimizar la experiencia de compra de los usuarios y apoyar a pequeños negocios gastronómicos en su proceso de digitalización.

Objetivos Específicos

- Diseñar una interfaz de usuario amigable e intuitiva empleando los componentes visuales de Flutter.
- Implementar un sistema de inicio de sesión para garantizar el acceso personalizado de los usuarios.
- Desarrollar un catálogo digital de productos con descripción, imágenes y precios.
- Incorporar un módulo de carrito de compras que permita gestionar pedidos en tiempo real.
- Validar la respectiva funcionalidad de la aplicación mediante pruebas de usabilidad y retroalimentación de usuarios potenciales.

¿Qué es una Aplicación Móvil?

Una aplicación móvil se define como un programa de software diseñado para ejecutarse en dispositivos portátiles como teléfonos inteligentes y tabletas, con el fin de proporcionar al usuario acceso a información, servicios y herramientas específicas de manera rápida y eficiente. Estas aplicaciones representan una de las principales innovaciones de la era digital, pues permiten trasladar procesos tradicionalmente realizados en computadores de escritorio a dispositivos que acompañan al usuario en todo momento (Turban, 2018).

Las aplicaciones móviles se caracterizan por su portabilidad, ya que pueden ser utilizadas en cualquier lugar; por su interfaz intuitiva, diseñada para pantallas táctiles de reducido tamaño; y por su capacidad de conectividad, pues la mayoría requieren acceso a internet para garantizar la interacción en tiempo real entre usuario y servidor. Además, muchas incluyen opciones de personalización, lo que incrementa la satisfacción del usuario (Shneiderman, 2017).

Aplicación móvil como un recurso para facilitar el comercio y las ventas.

El m-commerce o comercio móvil se ha consolidado como una tendencia creciente, debido a la facilidad con la que los usuarios pueden acceder a catálogos de productos, realizar compras en línea y efectuar pagos seguros desde sus dispositivos móviles. Esto no solo agiliza los procesos de compra-venta, sino que también mejora la experiencia del cliente, al permitir transacciones rápidas y seguras en cualquier momento y lugar (López & García, 2020).

Aplicación móvil de ventas.

Las aplicaciones móviles han transformado los procesos de comercialización al convertirse en una herramienta estratégica para empresas de todos los tamaños. Gracias a su accesibilidad, portabilidad y facilidad de uso, han permitido que las ventas se realicen de manera más rápida, eficiente y personalizada. (López & García, 2020).

¿Por qué utilizar una aplicación móvil?

El uso de aplicaciones móviles se ha convertido en una tendencia global debido a su capacidad para ofrecer soluciones rápidas, accesibles y personalizadas a las necesidades de los usuarios y de las empresas. Su implementación responde a varios factores que explican

por qué son hoy un recurso indispensable en diversos ámbitos de la vida social, económica y educativa.

En primer lugar, las aplicaciones móviles destacan por su portabilidad y accesibilidad. Al estar instaladas en dispositivos que los usuarios llevan consigo en todo momento, facilitan el acceso inmediato a productos, servicios e información desde cualquier lugar y en cualquier hora.

En segundo lugar, representan un medio para mejorar la experiencia del usuario, ya que ofrecen interfaces intuitivas, procesos simplificados y opciones de personalización. Esto genera mayor satisfacción en el consumidor, promueve la fidelización y fomenta la interacción constante entre clientes y empresas.

Otro aspecto fundamental es su impacto en la productividad y competitividad de los negocios. Las aplicaciones móviles permiten ampliar el alcance del mercado, gestionar inventarios en tiempo real, implementar pasarelas de pago seguras y analizar datos sobre hábitos de consumo. Todo esto contribuye a optimizar procesos internos y a fortalecer la toma de decisiones estratégicas (López & García, 2020).

¿Qué tipo de aplicaciones móvil existen?

Las aplicaciones móviles se clasifican en diferentes tipos según su modo de desarrollo, compatibilidad y las necesidades que buscan cubrir. Entre las principales categorías se encuentran las siguientes:

Aplicaciones nativas: Son aquellas diseñadas y desarrolladas para un sistema operativo específico, como Android (utilizando Java o Kotlin) o iOS (utilizando Swift u Objective-C). Se caracteriza n por ofrecer un alto rendimiento, acceso completo a las funcionalidades del dispositivo (cámara, GPS, sensores) y una experiencia de usuario optimizada. Sin embargo, requieren mayor tiempo y costo de desarrollo, ya que es necesario crear una versión distinta para cada sistema operativo (Pressman, 2014).

Aplicaciones web: (Web apps) Funcionan a través de un navegador móvil y no necesitan instalación en el dispositivo. Su principal ventaja es la multiplataforma, ya que son accesibles desde cualquier sistema operativo con conexión a internet. No obstante, presentan

limitaciones en el acceso a las funciones del hardware y dependen en gran medida de la conectividad (Turban, 2018).

Aplicaciones híbridas o multiplataforma: Combinan características de las nativas y la web. Se desarrollan utilizando frameworks como Flutter, React Native o Ionic, lo que permite escribir un solo código y ejecutarlo en múltiples plataformas. Ofrecen un desarrollo más rápido y económico, con interfaces modernas y buena integración con el hardware. Su principal reto es que, en algunos casos, el rendimiento puede ser inferior al de las aplicaciones nativas (Google, 2025).

Aplicaciones progresivas (PWA, Progressive Web Apps): Son aplicaciones web avanzadas que pueden instalarse en el dispositivo móvil y funcionar de manera similar a las nativas. Permiten notificaciones push, acceso offline limitado y una experiencia de usuario mejorada. Representan una alternativa de bajo costo para empresas que desean presencia móvil sin desarrollar aplicaciones nativas completas (López & García, 2020).

Flutter

imagen 1 logo flutter

Flutter es un kit de desarrollo de software (SDK) creado por Google que permite la construcción de aplicaciones multiplataforma a partir de un único código base. Este framework posibilita el desarrollo de aplicaciones móviles para Android e iOS, así como el despliegue de aplicaciones en la web y en sistemas de escritorio como Windows, macOS y Linux (Google, 2023).

Una de las características más relevantes de Flutter es su capacidad de ofrecer un rendimiento cercano al nativo, dado que no depende de componentes intermedios para comunicarse con las interfaces gráficas del sistema operativo. En lugar de utilizar elementos nativos, Flutter emplea su propio motor gráfico llamado Skia, lo que le permite renderizar widgets personalizados y garantizar consistencia visual y fluidez en distintos dispositivos (Google, 2023).

Importancia de la programación móvil

La programación móvil se ha convertido en un eje central del desarrollo tecnológico contemporáneo, debido al creciente uso de dispositivos móviles como teléfonos inteligentes y tabletas en la vida cotidiana. Estos dispositivos no solo se utilizan para la comunicación, sino también como herramientas de productividad, educación, comercio y entretenimiento, lo que ha generado una demanda constante de aplicaciones móviles innovadoras y funcionales (Google, 2023).

En términos sociales, la programación móvil permite el desarrollo de soluciones que facilitan la interacción entre individuos y comunidades, fomentando nuevas formas de comunicación y acceso a la información. Gracias a las aplicaciones móviles, se ha logrado democratizar el acceso a servicios de salud, educación en línea, transporte, banca digital y comercio electrónico, impactando de manera positiva en la calidad de vida de las personas.

Desde una perspectiva económica, la industria de las aplicaciones móviles representa un sector de alto crecimiento. Empresas de todos los tamaños han identificado en las aplicaciones móviles un canal estratégico para la generación de ingresos, fidelización de clientes y expansión de mercados. En este sentido, la programación móvil se convierte en un recurso indispensable para la competitividad empresarial, al permitir la creación de productos digitales capaces de adaptarse a las necesidades de los usuarios y a las tendencias de consumo.

Aporte de las aplicaciones móvil en la modernidad

H En la actualidad, las aplicaciones móviles constituyen uno de los principales motores de transformación digital, debido a su impacto en los ámbitos social, económico, educativo y cultural. La modernidad se caracteriza por una sociedad interconectada, en la que los dispositivos móviles y sus aplicaciones se han convertido en herramientas indispensables para la vida cotidiana (Google, 2023).

Uno de los aportes más significativos de las aplicaciones móviles es la facilitación del acceso a la información y la comunicación. Hoy en día, los usuarios pueden acceder a contenidos en tiempo real, mantenerse informados sobre acontecimientos globales y comunicarse de manera instantánea con personas en cualquier lugar del mundo. Esto ha generado una

sociedad más informada y participativa, donde la inmediatez y la interacción son elementos clave.

En el ámbito económico, las aplicaciones móviles han revolucionado la forma en que las empresas ofrecen sus productos y servicios. Plataformas de comercio electrónico, banca digital y servicios de entrega a domicilio han ampliado las oportunidades de negocio, al mismo tiempo que han transformado los hábitos de consumo. En este sentido, las aplicaciones móviles no solo incrementan la eficiencia empresarial, sino que también impulsan la economía digital global.

Desde una perspectiva educativa, las aplicaciones móviles han contribuido a democratizar el acceso al conocimiento, al permitir que estudiantes y profesionales accedan a cursos en línea, bibliotecas digitales y entornos de aprendizaje interactivos. Este aporte resulta fundamental en un mundo donde la educación continua es clave para el desarrollo personal y profesional.

Asimismo, en la vida social y cultural, las aplicaciones móviles favorecen la creación de comunidades virtuales, el intercambio cultural y la preservación de tradiciones a través de plataformas digitales. De esta forma, se convierten en instrumentos de cohesión social que permiten tanto la interacción global como el fortalecimiento de identidades locales.

En conclusión, las aplicaciones móviles representan un pilar esencial de la modernidad, ya que permiten mejorar la comunicación, optimizar procesos económicos, democratizar la educación y fortalecer la interacción social. Su aporte es indiscutible en la construcción de un mundo más dinámico, interconectado y orientado hacia la innovación tecnológica (Google, 2023).

Normativa de las aplicaciones móvil

Las aplicaciones móviles, al estar directamente relacionadas con la interacción digital y el manejo de información personal y comercial, se encuentran sujetas a diferentes normativas nacionales e internacionales que buscan proteger los derechos de los usuarios y garantizar la transparencia en el uso de la tecnología.

En primer lugar, destacan las normas de protección de datos personales, que constituyen la base de la regulación para la mayoría de las aplicaciones móviles. A nivel internacional, el Reglamento General de Protección de Datos (GDPR) de la Unión Europea es una de las legislaciones más estrictas, pues exige el consentimiento informado de los usuarios, la minimización de los datos recolectados y el derecho al olvido digital. En el contexto colombiano, la Ley 1581 de 2012 regula el tratamiento de datos personales, estableciendo obligaciones para los responsables y encargados del manejo de la información (Google, 2023).

En segundo lugar, la normativa incluye las políticas de propiedad intelectual y licenciamiento de software. Las aplicaciones móviles son productos protegidos por derechos de autor, lo que

implica que los desarrolladores deben respetar licencias de código abierto, registrar marcas y evitar la reproducción no autorizada de contenidos.

Por otra parte, las aplicaciones que realizan actividades comerciales deben cumplir con la legislación en comercio electrónico y protección al consumidor, la cual regula las compras en línea, la validez de los contratos electrónicos, el derecho al retracto y la transparencia en la información sobre productos y servicios.

A nivel técnico, las tiendas oficiales de aplicaciones como Google Play y App Store también imponen normativas específicas para la publicación de software. Estas políticas incluyen lineamientos de seguridad, protección contra malware, requisitos de accesibilidad y prohibiciones en relación con contenidos ilegales o engañosos.

Finalmente, es necesario considerar las normas en ciberseguridad, que sancionan prácticas como el acceso no autorizado, el fraude electrónico o la manipulación indebida de datos. Estas normativas obligan a los desarrolladores a implementar medidas técnicas de protección, como el cifrado de la información y la verificación de identidad de los usuarios.

Marco teórico

Instalación de flutter:

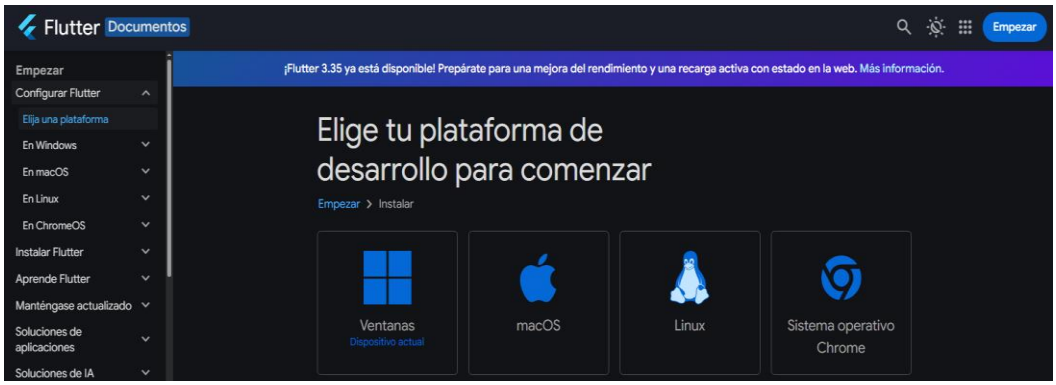

imagen 2 selección del sistema operativo

Primero se ingresa a la pagina de flutter y seleccionamos el sistema operativo en el cual vamos a instalar.

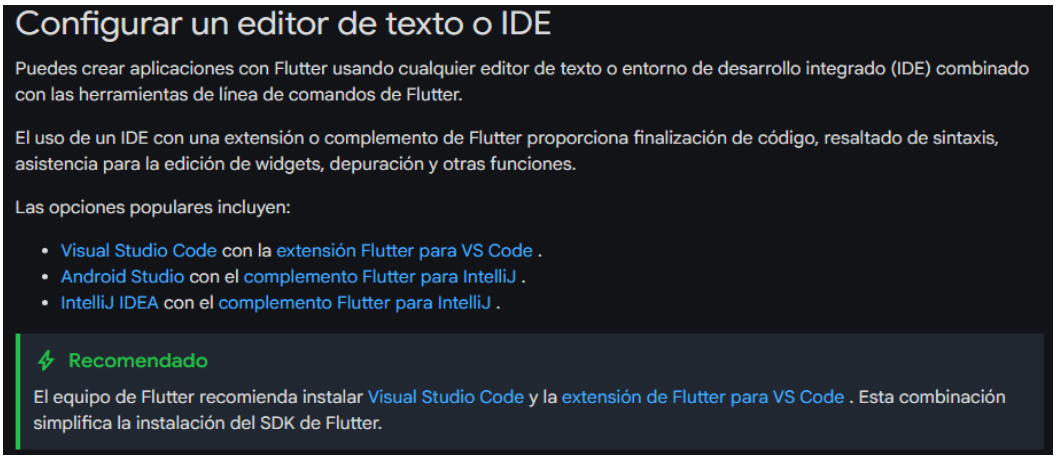

imagen 3 recomendaciones para instalación del editor de texto

En la imagen nos da unas recomendaciones para la instalación de un editor de texto los cuales pueden variar entre visual estudio code, Android estudio e intellid IDEA, En la cual nos recomienda usar visual estudio code con la extensión Flutter para VS Code.

imagen 4 Solicitar a VS Code que instale Flutter

En la anterior imagen muestra el paso a paso que debemos seguir después de haber instalado visual estudio code para la instalación de flutter como un complemento del mismo.

imagen 5 descarga del SDK de flutter

Esta es la forma de descarga de SDK que es el conjunto de herramientas de desarrollo creado por Google que permite diseñar, compilar y ejecutar aplicaciones multiplataforma a partir de un único código base. Un SDK en términos generales es un paquete de software que contiene librerías, compiladores, entornos de ejecución, documentación y utilidades, todo lo necesario para que los desarrolladores puedan crear aplicaciones de manera eficiente.

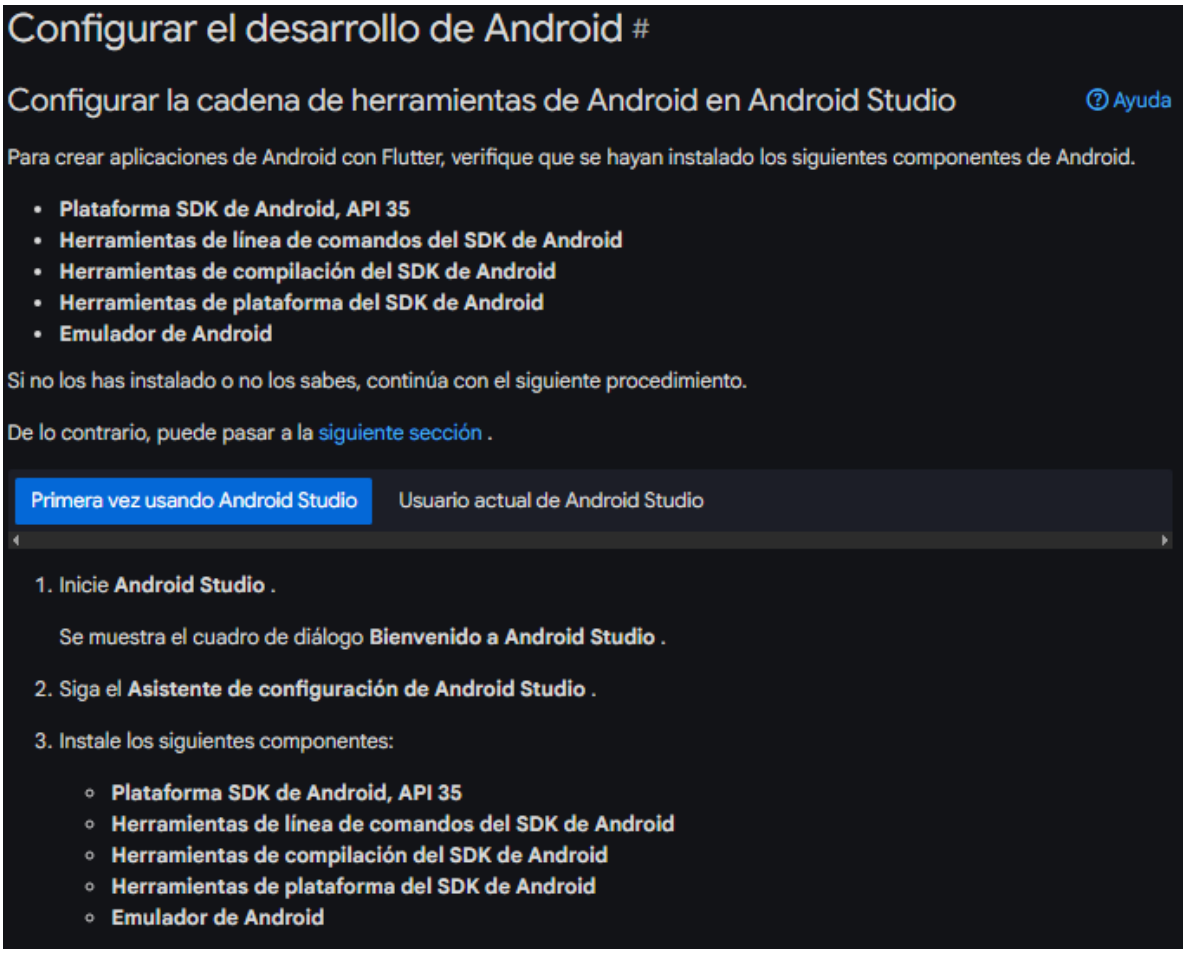

imagen 6 Configurar el desarrollo de Android

La cadena de herramientas de Android (Android toolchain) se refiere al conjunto de programas, librerías y utilidades que proporciona Android Studio para desarrollar, compilar, depurar y ejecutar aplicaciones en el sistema operativo Android. Es decir, es el ecosistema de herramientas que permite transformar el código fuente escrito por el programador en una aplicación ejecutable en dispositivos móviles o emuladores.

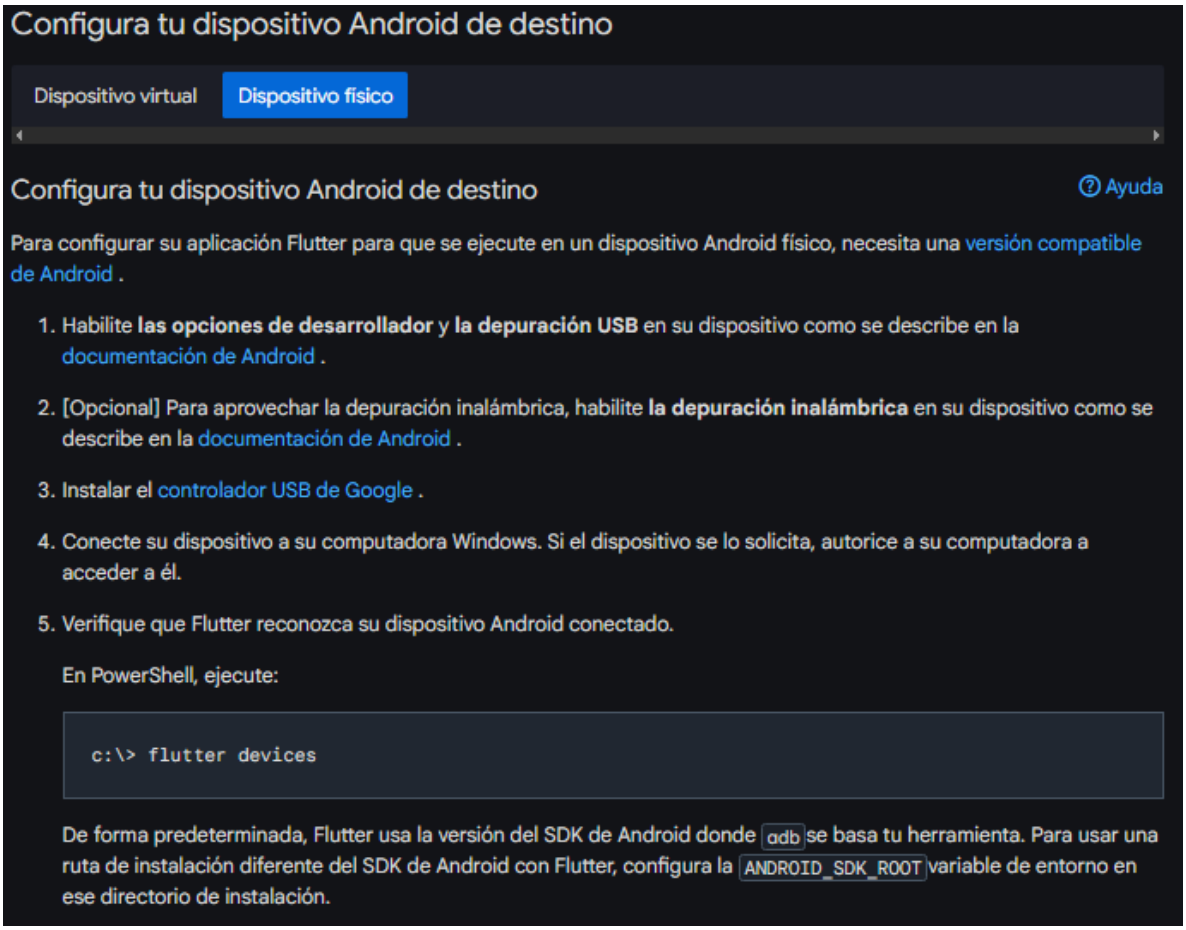

imagen 7 Configurar dispositivo Android de destino

En este caso tenemos dos formas de configurar la visualización para realizar pruebas. Los dispositivos físicos ofrecen la ventaja de ejecutar la aplicación en condiciones reales de hardware, permitiendo evaluar aspectos como el rendimiento, el consumo de batería y la interacción táctil. Sin embargo, requieren conexión mediante cable o configuración de depuración inalámbrica, además de presentar limitaciones relacionadas con la versión de Android del equipo.

Por su parte, los dispositivos virtuales permiten simular diferentes versiones del sistema operativo y tamaños de pantalla, lo cual facilita las pruebas de compatibilidad sin necesidad de disponer de múltiples teléfonos. No obstante, suelen demandar altos recursos del computador, presentan menor precisión en la simulación de hardware y pueden dar una percepción errónea del rendimiento real de la aplicación.

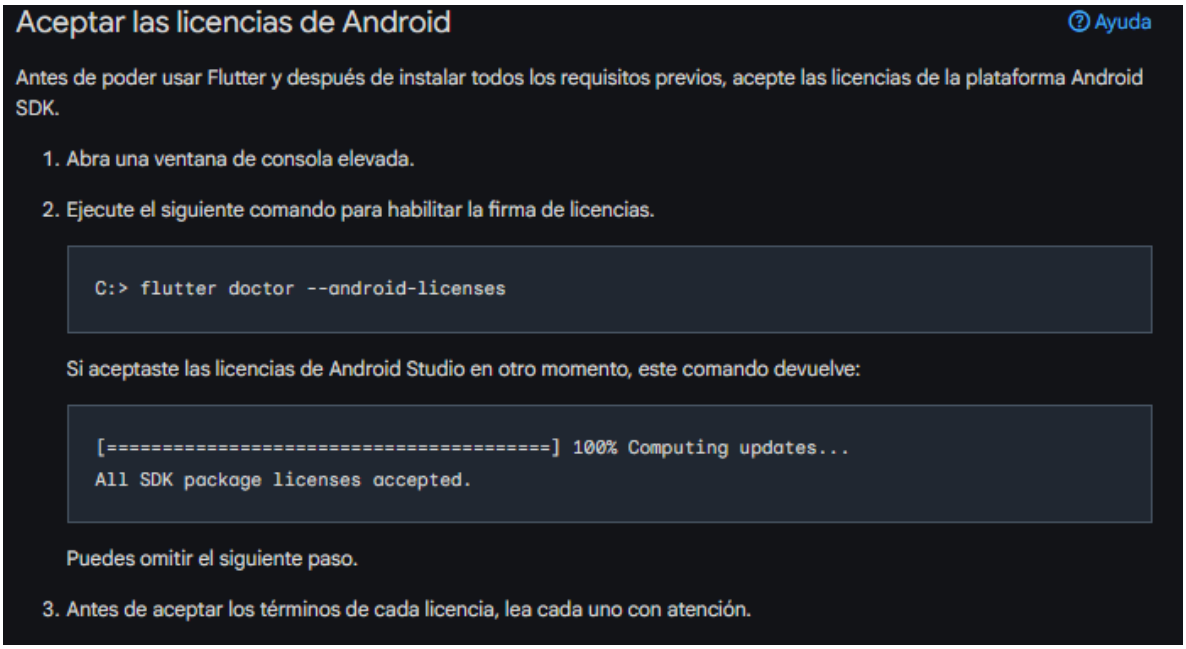

imagen 8 aceptar las licencias de Android

Es necesario aceptar las licencias de Android que acompañan al SDK (Software Development Kit) y demás herramientas de desarrollo. Estas licencias son acuerdos legales establecidos por Google que regulan el uso del software, las bibliotecas y las APIs de Android.

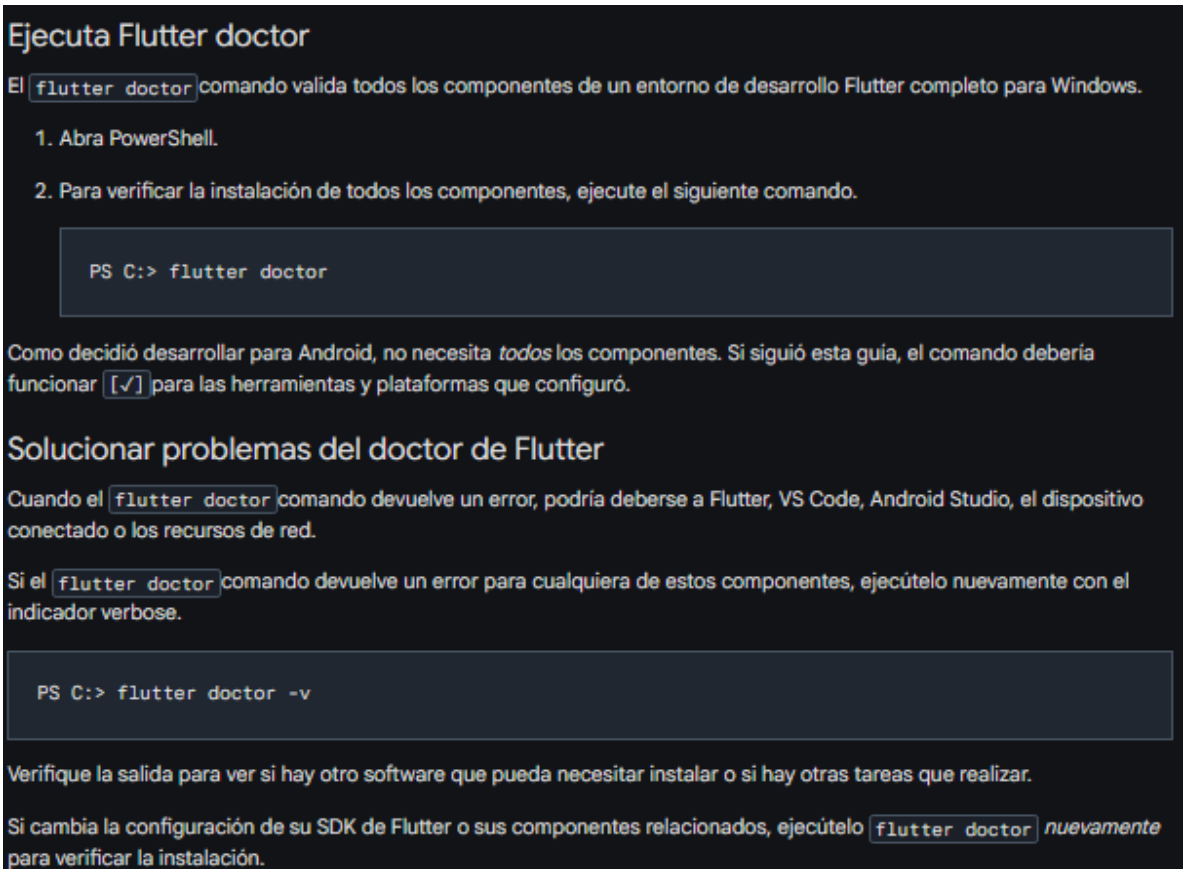

imagen 9 ejecutar flutter doctor

El comando flutter doctor es una herramienta de diagnóstico incluida en el SDK de Flutter, cuyo objetivo es verificar la correcta instalación y configuración del entorno de desarrollo. Al ejecutarlo desde la terminal o PowerShell, el sistema analiza los componentes necesarios para compilar y ejecutar aplicaciones en Flutter y genera un informe detallado sobre su estado.

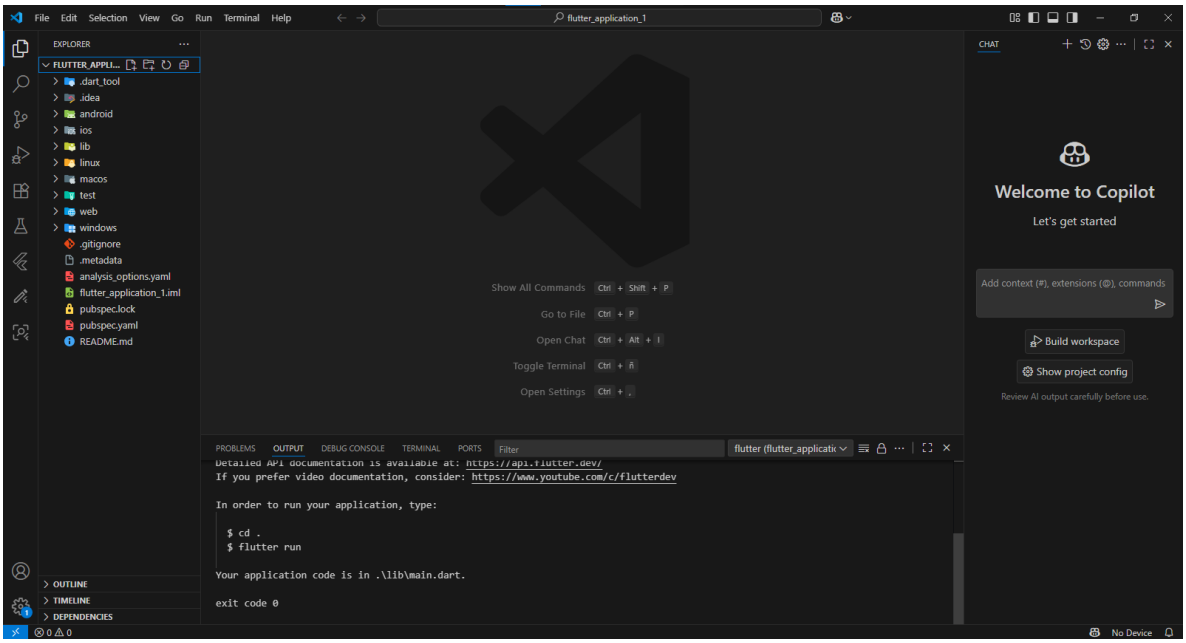

imagen 10 entorno configurado

En este caso vemos el entorno configurado y listo para poder programar nuestra aplicación móvil.

Que Hambre Burgers

Nos inspiramos en la necesidad de ofrecer una solución digital para aquellos negocios que no cuentan con una infraestructura física. A partir de esta idea, se desarrolló un prototipo de aplicación móvil utilizando Canva como herramienta inicial de diseño.

El flujo que presentamos corresponde a la experiencia de un usuario una vez registrado con sus credenciales dentro de la plataforma. La aplicación está organizada en cuatro vistas principales, que se describen a continuación:

imagen 11 pantalla login

Pantalla de inicio de sesión (Login): Permite al usuario ingresar con su nombre de usuario y contraseña. También incluye la opción de restablecer la contraseña en caso de olvido.

imagen 12 pantalla home

Pantalla principal (Home): En esta vista se concentra el acceso al menú principal, compuesto por tres secciones:

- ¿Quiénes somos?: información sobre los orígenes y objetivos de la empresa.
- Número de contacto: datos para comunicarse directamente con la empresa.
- Salir de la aplicación: opción para cerrar sesión.

Además, se incluyen accesos directos:

- Botón "Carrito de compras": muestra los productos seleccionados, el valor total y permite finalizar la compra.
- Botón "Nuestros productos": redirige a la pantalla de catálogo.
- Botón "¿Quiénes somos?": describe los inicios y la esencia de Que Hambre Burgers.

imagen 13 pantalla de productos

Pantalla de productos: Aquí se despliega el listado completo de productos disponibles, acompañado de su descripción, precio y un botón para agregar al carrito, en esta podemos elegir diferentes productos los cuales se van sumando en el carrito y la flecha de retorno la cual nos regresa a la pantalla home.

imagen 14 pantalla carrito

Pantalla de carrito: Muestra los productos seleccionados, incluyendo nombre, cantidad y valor total de la compra, cuando le presionamos el botón finalizar compra nos sale un aviso confirmando la compra con el valor total .

Creación de la aplicación móvil *Que Hambre Burgers*

A continuación, compartimos una parte de nuestro código correspondiente a la **primera pantalla (Login)**.

Posteriormente, explicaremos de forma sencilla cada sección del código y la función que cumple dentro de la aplicación. Al final, se compartirá el enlace al repositorio de **Git**, en el cual se encuentra el proyecto completo y anexado para su consulta (anexos).

Código de flutter Login_page.dart

```dart
import 'package:flutter/material.dart';
import '../design/app_colors.dart';

class LoginPage extends StatefulWidget {
  @override
  State<LoginPage> createState() => _LoginPageState();
}

class   LoginPageState extends State<LoginPage> {
  final   userCtrl = TextEditingController();
  final _passCtrl = TextEditingController();
  final   formKey = GlobalKey<FormState>();
  bool _loading = false;

  // Credenciales dummy
  final String goodUser = 'admin';
  final String goodPass = '1234';

  void _login() async {
    if (!_formKey.currentState!.validate()) return;
    setState(() => _loading = true);
    await Future.delayed(Duration(milliseconds: 700));
    setState(() => _loading = false);

    if (_userCtrl.text == goodUser && _passCtrl.text == goodPass) {
      Navigator.of(context).pushReplacementNamed('/home');
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Credenciales inválidas. Usa admin / 1234')),
      );
    }
  }

  @override
  void dispose() { //
     userCtrl.dispose();
    _passCtrl.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
```

```
return Scaffold(
  body: Container(
    decoration: BoxDecoration(gradient: AppColors.mainGradient),
    padding: EdgeInsets.symmetric(horizontal: 28),
    child: Center(
      child: SingleChildScrollView(
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              SizedBox(height: 40),
              Text(
                  'Qué Hambre Burgers',
                  style: TextStyle(
                  fontSize: 28,
                  fontWeight: FontWeight.bold,
                  color: Colors.white,
                  shadows: [
                 Shadow(
                  offset: Offset(4, 4),
                  blurRadius: 4,
                  color: Colors.black.withOpacity(0.6),
                   ),
                  ],
                ),
               ),
              SizedBox(height: 40),
              TextFormField(
                controller: _userCtrl,
                decoration: InputDecoration(
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.15),
                  hintText: 'Username',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(30),
                    borderSide: BorderSide.none,
                  ),
                  contentPadding: EdgeInsets.symmetric(horizontal: 20, vertical: 16),
                ),
                validator: (v) => (v == null || v.isEmpty) ? 'Ingrese usuario' : null,
              ),
              SizedBox(height: 16),
              TextFormField(
                controller: _passCtrl,
                obscureText: true,
```

```
                decoration: InputDecoration(
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.15),
                  hintText: 'Password',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(30),
                    borderSide: BorderSide.none,
                  ),
                  contentPadding: EdgeInsets.symmetric(horizontal: 20, vertical: 16),
                ),
                validator: (v) => (v == null || v.isEmpty) ? 'Ingrese password' : null,
              ),
              SizedBox(height: 12),
              Align(
                alignment: Alignment.centerRight,
                child: TextButton(
                  onPressed: () {
                    showDialog(context: context, builder: ( ) =>
                      AlertDialog(
                        title: Text('Recuperar contraseña'),
                        content: Text('Recuerda usar: admin / 1234'),
                              actions: [TextButton(onPressed: ()=>Navigator.pop(context), child:
Text('OK'))],
                      )
                    );
                  },
                  child: Center(
                    child: Text(
                      'Restablecer contraseña?',
                      style: TextStyle(color: Colors.white),
                    ),
                  ),
                ),
              ),
              SizedBox(height: 8),
              ElevatedButton(
                onPressed: _loading ? null : _login,
                style: ElevatedButton.styleFrom(
                  padding: EdgeInsets.symmetric(horizontal: 60, vertical: 14),
                  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(30)),
                  backgroundColor: Color.fromARGB(255, 57, 128, 209),
                ),
                child: _loading
                ? CircularProgressIndicator(color: const Color.fromARGB(255, 255, 254, 254))
                  : Text(
```

```
                  'Login',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 16,       // opcional: tamaño
                  fontWeight: FontWeight.bold, // opcional: negrita
                ),
              ),
            ),
            SizedBox(height: 40),
          ],
        ),
      ),
    ),
  ),
),
    );
  }
}
```

Conclusiones

El desarrollo de la aplicación demuestra cómo las herramientas de programación móvil, en este caso Flutter, permiten crear soluciones tecnológicas innovadoras que responden a necesidades reales del mercado. A través de interfaces intuitivas y dinámicas, la aplicación integra funciones esenciales como el inicio de sesión, la visualización de productos, la gestión del carrito de compras y la navegación sencilla, facilitando así la experiencia del usuario.

Este proyecto evidencia que las aplicaciones móviles no solo representan un canal moderno de interacción entre empresas y clientes, sino que también constituyen un recurso estratégico para fortalecer la competitividad en el sector comercial. En este sentido, la aplicación desarrollada contribuye a la digitalización de los procesos de ventas y promoción gastronómica, optimizando tiempos, mejorando la accesibilidad y generando una mayor cercanía con el consumidor.

Finalmente, el trabajo realizado refleja la importancia de la programación multiplataforma en la formación académica y profesional, ya que permite materializar proyectos funcionales con un alto impacto práctico. Por tanto, la aplicación constituye un aporte significativo tanto al aprendizaje en el ámbito del desarrollo móvil como a la modernización de los modelos de negocio en la actualidad.

Referencias bibliográficas.

Asimov, I. (1950). I, Robot. Gnome Press.

Google. (2023). Android SDK Terms and Conditions. Android Developers. https://developer.android.com/studio/terms

Google. (2023). Android Studio: The official IDE for Android. Android Developers. https://developer.android.com/studio

Google. (2023). Configure your Android device for development. Android Developers. https://developer.android.com/studio/run/device

Google. (2023). Flutter doctor tool. Flutter Documentation. https://docs.flutter.dev/tools/flutter-doctor

Google. (2023). Flutter SDK overview. Flutter Documentation. https://docs.flutter.dev

TechTarget. (2021). What is mobile application development? TechTarget. https://www.techtarget.com/searchmobilecomputing/definition/mobile-application-development.

Anexo

Explicación del código de Login en Flutter

El siguiente código implementa una pantalla de inicio de sesión (Login) en una aplicación Flutter. Incluye:

- Validación de usuario y contraseña.
- Simulación de carga.
- Redirección a la pantalla Home (Pantalla de inicio) si las credenciales son correctas.
- Mensaje de error si las credenciales son incorrectas.
- Diseño visual con gradiente, campos estilizados y botones personalizados.

1. **Importaciones:**

```
import 'package:flutter/material.dart';
import '../design/app_colors.dart';
```

material.dart: Permite usar los widgets principales de Flutter (Scaffold, TextFormField, Button, etc.).

app_colors.dart: Archivo personalizado donde se define la paleta de colores (en este caso, un gradiente para el fondo).

2. **Clase principal LoginPage:**

```
class LoginPage extends StatefulWidget {
@override
State<LoginPage> createState() => _LoginPageState();
}
```

Es un StatefulWidget porque el formulario necesita manejar estados (como el texto ingresado, la carga y validación).

3. **Estado de la página _LoginPageState:**

Aquí se definen las variables y funciones que controlan el comportamiento del login.

Controladores de texto:

```
final _userCtrl = TextEditingController();
final _passCtrl = TextEditingController();
```

Permiten acceder y manipular el texto ingresado en los campos de usuario y contraseña.

Clave del formulario:

```
final _formKey = GlobalKey<FormState>();
```

Se utiliza para validar los campos dentro de un formulario (Form).

Indicador de carga:

```
bool _loading = false;
```

Controla si se está mostrando el spinner de carga en el botón de login.

Credenciales de prueba;

```
final String goodUser = 'admin';
final String goodPass = '1234';
```

Se usan como credenciales dummy (solo para fines de demostración).

4. **Función de login**

```
void _login() async {
  if (!_formKey.currentState!.validate()) return; // Validación
  setState(() => _loading = true);

  // Simulación de carga
  await Future.delayed(Duration(milliseconds: 700));

  setState(() => _loading = false);
```

```
  // Verificación de credenciales
  if (_userCtrl.text == goodUser && _passCtrl.text == goodPass) {
    Navigator.of(context).pushReplacementNamed('/home'); // Redirige al home
  } else {
   ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Credenciales inválidas. Usa admin / 1234')),
    );
  }
}
```

Valida los campos.
Simula un proceso de autenticación con una breve espera.
Si las credenciales son correctas, redirige a /home.
Si son incorrectas, muestra un mensaje con SnackBar.

5. **Liberación de memoria:**

```
@override
void dispose() {
  _userCtrl.dispose();
  _passCtrl.dispose();
  super.dispose();
}
```

Libera los controladores cuando ya no se usan, evitando fugas de memoria.

6. **Interfaz gráfica (build):**

```
return Scaffold(
  body: Container(
   decoration: BoxDecoration(gradient: AppColors.mainGradient),
    padding: EdgeInsets.symmetric(horizontal: 28),
    child: Center(
      child: SingleChildScrollView(
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              ...
            ],
          ),
        ),
      ),
    ),
  ),
);
```

Usa un Scaffold con un Container de fondo que aplica el gradiente.

El formulario está dentro de un SingleChildScrollView para evitar problemas si el teclado cubre los campos.

7. **Interfaz gráfica (build):**

```
return Scaffold(
  body: Container(
    decoration: BoxDecoration(gradient: AppColors.mainGradient),
    padding: EdgeInsets.symmetric(horizontal: 28),
    child: Center(
      child: SingleChildScrollView(
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              ...
            ],
          ),
        ),
      ),
    ),
  ),
);
```

Usa un Scaffold con un Container de fondo que aplica el gradiente.
El formulario está dentro de un SingleChildScrollView para evitar problemas si el teclado cubre los campos.

Campo de usuario:

```
TextFormField(
  controller: _userCtrl,
  decoration: InputDecoration(
    hintText: 'Username',
    filled: true,
    fillColor: Colors.white.withOpacity(0.15),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(30),
      borderSide: BorderSide.none,
    ),
  ),
  validator: (v) => (v == null || v.isEmpty) ? 'Ingrese usuario' : null,
),
```

Caja de texto con validación que pide el nombre de usuario.

Campo de contraseña:

```
TextFormField(
  controller: _passCtrl,
```

```
  obscureText: true,
  decoration: InputDecoration(
    hintText: 'Password',
    filled: true,
    fillColor: Colors.white.withOpacity(0.15),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(30),
      borderSide: BorderSide.none,
    ),
  ),
  validator: (v) => (v == null || v.isEmpty) ? 'Ingrese password' : null,
),
```

Caja de texto que oculta la contraseña.

Botón de restablecer contraseña:

```
TextButton(
  onPressed: () {
    showDialog(
      context: context,
      builder: (_) => AlertDialog(
        title: Text('Recuperar contraseña'),
        content: Text('Recuerda usar: admin / 1234'),
        actions: [
          TextButton(onPressed: ()=>Navigator.pop(context), child: Text('OK'))
        ],
      )
    );
  },
  child: Text('Restablecer contraseña?', style: TextStyle(color: Colors.white)),
),
```

Abre un AlertDialog con la información de recuperación.

Botón de login:

```
ElevatedButton(
  onPressed: _loading ? null : _login,
  style: ElevatedButton.styleFrom(
    padding: EdgeInsets.symmetric(horizontal: 60, vertical: 14),
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(30)),
    backgroundColor: Color.fromARGB(255, 57, 128, 209),
  ),
  child: _loading
    ? CircularProgressIndicator(color: Colors.white)
```

```
    : Text(
        'Login',
        style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
      ),
),
```

Si _loading está activo, muestra un indicador de progreso.

Si no, muestra el texto Login.

Al presionar, ejecuta la función _login.

Para visualizar el código fuente del proyecto está disponible en GitHub, para quienes deseen revisarlo, estudiarlo o mejorarlo.