TRABAJO DE GRADO
Opción Seminario-Diplomado.

Aplicación Mobile Recetario

Corporación Universitaria Remington.
Facultad de Ingeniería.
Ingeniería de sistemas.

Juan Jose Sierra Giraldo
Jonatan Stick Campos Nuñez
Opción de Trabajo de grado Seminario-Diplomado.
2025

Dedicatoria

A mis padres, quienes con su amor, paciencia y apoyo incondicional han sido la base de mi formación personal y profesional. A mi madre, por enseñarme la importancia de la constancia y la sensibilidad humana; y a mi padre, por inspirarme con su esfuerzo y ejemplo de perseverancia. Este logro es también suyo, fruto del camino que hemos recorrido juntos.

Agradecimientos

A la universidad, por brindarme las herramientas académicas y el espacio para crecer en conocimiento y experiencia.

A mis docentes, por su compromiso y orientación a lo largo del proceso formativo.

A mis compañeros, por su apoyo, amistad y colaboración, que hicieron de este camino una experiencia enriquecedora y motivadora.

Tabla de Contenidos

Resumen

Flutter es un framework de desarrollo multiplataforma creado por Google, diseñado para crear aplicaciones nativas de alto rendimiento desde una única base de código. Su enfoque basado en widgets, la flexibilidad en el diseño y su capacidad de integración con servicios externos lo convierten en una de las tecnologías más utilizadas para el desarrollo móvil actual. El presente trabajo aborda la instalación y configuración del entorno, la construcción de interfaces mediante widgets fundamentales y de disposición, la aplicación de temas y estilos, la gestión del estado bajo principios de arquitectura limpia, el consumo de servicios API REST, el almacenamiento local y la autenticación de usuarios. Como resultado, se implementó una aplicación de recetario que demuestra la aplicabilidad de estos conceptos en un proyecto real.

Palabras clave

Flutter, widgets, estado, arquitectura limpia, API REST, almacenamiento local, autenticación, desarrollo móvil.

Objetivo general

Analizar e implementar las principales características y funcionalidades de Flutter para el desarrollo de una aplicación móvil multiplataforma, integrando conceptos de diseño, gestión de estado, consumo de servicios y seguridad.

Objetivos específicos

- Instalar y configurar el entorno de desarrollo Flutter en conjunto con sus dependencias.
- Identificar y aplicar los widgets fundamentales para la construcción de interfaces interactivas.
- Implementar diseños visuales mediante widgets de disposición, temas y estilos.
- Establecer una arquitectura limpia y aplicar técnicas de gestión de estado en la aplicación.
- Desarrollar la comunicación con servicios externos a través de API REST.
- Implementar almacenamiento local para persistencia de datos en el dispositivo.
- Incorporar un sistema de autenticación de usuarios que garantice la seguridad y el acceso controlado.
- Construir una aplicación móvil de tipo recetario como resultado práctico del trabajo.

Pregunta orientadora de la búsqueda

¿Cómo utilizar Flutter para desarrollar aplicaciones móviles multiplataforma que integren interfaces modernas, gestión de datos, comunicación con servicios externos y autenticación de usuarios, bajo buenas prácticas de arquitectura de software?

Metodología de búsqueda de la información

La información presentada en este trabajo se recopiló principalmente de la documentación oficial de Flutter, considerada la fuente más confiable y actualizada para el aprendizaje y desarrollo con este framework.

Adicionalmente, se complementa con contenidos de un curso especializado de la academia Platzi, lo cual permitió reforzar conceptos clave y brindar ejemplos prácticos que facilitaron la comprensión de temas avanzados.

Finalmente, el aprendizaje se fortaleció con los temas abordados en el seminario académico, el cual sirvió de guía estructural para orientar la investigación y la práctica, permitiendo articular el marco teórico con la aplicación práctica desarrollada.

Sustentación teórica de la pregunta

Durante mi proceso de formación comprendí que las aplicaciones móviles se han convertido en herramientas esenciales de aprendizaje. Hoy en día, desde pedir comida a domicilio, organizar tareas, monitorear la salud o registrar finanzas personales, casi todas las actividades cotidianas pueden gestionarse a través de una app. Este panorama me hizo reflexionar sobre cómo la tecnología no solo facilita la vida de las personas, sino que también abre un mundo de oportunidades para quienes desean crear soluciones innovadoras.

Frente a esta realidad surgió en mí el interés de aprender a desarrollar mis propias aplicaciones y realizar el seminario, no únicamente como un ejercicio académico, sino como una manera de transformar ideas en proyectos concretos que respondan a necesidades reales. Fue en ese contexto que descubrí **Flutter**, un framework moderno creado por Google que me ofrecía una alternativa práctica y poderosa para entrar al mundo del desarrollo móvil.

Lo que más me llamó la atención desde el inicio fue la posibilidad de **programar una sola vez y desplegar la aplicación en diferentes plataformas** como Android e iOS. Esto eliminaba la necesidad de escribir código duplicado para cada sistema operativo, algo que tradicionalmente hacía más costoso y complejo el proceso de desarrollo. Esta característica me motivó a elegirlo como la base de mi proyecto personal: una aplicación de recetario donde pudiera guardar, organizar y consultar recetas de cocina.

Figura 1. (Representación del desarrollo híbrido)

La imagen ilustra el principio del desarrollo multiplataforma que ofrece Flutter. Un mismo código base permite generar aplicaciones tanto para iOS como para Android, lo cual se refleja en los dos íconos que simbolizan ambos sistemas operativos. Esta característica constituye una de las principales ventajas del framework, ya que reduce el esfuerzo de programación duplicada y facilita la creación de soluciones que llegan a una audiencia más amplia.

Además de su capacidad multiplataforma, otro aspecto que me cautivó fue el uso de **widgets**, que constituyen la esencia de Flutter. Al entender que todo en Flutter es un widget, desde un simple texto hasta estructuras completas de interfaz, descubrí una manera más intuitiva de pensar en la construcción de pantallas y funciones. Esta filosofía de diseño me permitió visualizar mi aplicación como un conjunto de piezas modulares que podía ir ensamblando y personalizando según mis necesidades.

En el desarrollo del proyecto utilicé diferentes herramientas que me facilitaron el proceso. Entre ellas, **Visual Studio Code** como editor principal, que me ofreció extensiones específicas para Flutter con autocompletado, depuración y ejecución rápida. También aproveché el **iOS Simulator**, gracias a que trabajé en macOS, lo que me permitió ver en tiempo real cómo lucía y funcionaba mi app en un dispositivo Apple. A esto se sumó el uso de **CocoaPods** para manejar dependencias, algo indispensable cuando integré librerías externas.

Con el paso de los días comprendí que Flutter no es solo un framework de programación, sino una plataforma completa que combina simplicidad con potencia. Desde el "hot reload" —que me permitió visualizar cambios en la app casi de inmediato— hasta la integración con servicios externos como Firebase para la autenticación de usuarios, cada detalle me demostró que estaba utilizando una herramienta pensada para acelerar la curva de aprendizaje y a la vez entregar resultados profesionales.

En definitiva, mi decisión de trabajar con Flutter para el desarrollo de un recetario no fue casual. Fue la combinación de su filosofía basada en widgets, su versatilidad multiplataforma, las herramientas que ofrece para el trabajo en iOS y la oportunidad de aplicar lo aprendido en un proyecto concreto lo que convirtió esta experiencia en un paso clave dentro de mi formación.

1. Instalación y configuración del entorno

El proceso de instalación lo realicé en mi equipo personal con **macOS**, esto presentaba una configuración diferente y un poco más compleja pero que fue interesante y satisfactorio al

lograrla. El primer paso fue descargar el **SDK de Flutter** desde la página oficial y descomprimirlo en el sistema. Después configuré la variable de entorno para que el sistema reconociera el comando flutter desde la terminal.

Al ejecutar flutter doctor, el sistema me indicó los componentes faltantes. La principal dependencia fue **Xcode**, indispensable para compilar y ejecutar aplicaciones en dispositivos Apple. Instalé Xcode desde la App Store y lo vinculé con Flutter. Este paso también habilitó el uso del **iOS Simulator**, que me permitió probar la aplicación directamente en un simulador de iPhone sin necesidad de un dispositivo físico.

Otro detalle importante fue instalar **CocoaPods**, que se utiliza para manejar dependencias en proyectos iOS. Sin esta herramienta, muchas librerías externas de Flutter no podrían integrarse correctamente. Configuré CocoaPods mediante la terminal y realicé las pruebas iniciales de compilación.

Con esta preparación logré correr la primera aplicación de prueba (“Hello World”) en el simulador de iOS. Verla funcionando en una pantalla de iPhone fue un momento clave, porque me confirmó que todo estaba configurado de manera correcta y que estaba listo para empezar el desarrollo del recetario.

2. Widgets fundamentales

Una de las enseñanzas más importantes del seminario fue entender que **en Flutter todo es un widget**. Al principio me costaba asimilar la idea, pero con la práctica descubrí que pensar en widgets es como armar una maqueta pieza por pieza.

Para mi app de recetas utilicé widgets básicos como:

- **Text** para mostrar los nombres e instrucciones.
- **Image** para cargar fotos de cada plato.
- **Icon** para botones de acción como “favorito”.
- **Scaffold** para la estructura general de la pantalla.

También aprendí a diferenciar entre **StatelessWidget** y **StatefulWidget**. Por ejemplo, al mostrar una receta estática utilicé Stateless, pero cuando quería que un botón cambiara el estado (como marcar una receta como favorita), tuve que usar Stateful. Esa práctica me ayudó a interiorizar el concepto de estado en Flutter.

3. Widgets de disposición visual y diseño

Uno de los retos fue organizar los elementos en pantalla. En mis primeros intentos la interfaz se veía desordenada, pero poco a poco aprendí a usar widgets como **Row** y **Column** para alinear el contenido.

Para la pantalla de recetas decidí usar un **GridView**, porque me permitía mostrar varias tarjetas (Cards) con imágenes y títulos de manera más atractiva. Cuando veía que en algunas pantallas el contenido no encajaba bien, recurrí a widgets como **Expanded** o **Flexible** para ajustar proporciones.

Fue interesante descubrir que no necesitaba escribir código separado para cada tipo de dispositivo, sino que Flutter se encargaba de adaptar la interfaz.

Figura 2. (Ejemplo de disposición con GridView en Flutter)

La imagen muestra el uso del widget **GridView**, el cual permite organizar elementos en forma de cuadrícula dentro de una aplicación. Esta disposición facilita la visualización estructurada de múltiples componentes, como en este caso, una serie de ítems distribuidos en filas y columnas. En el contexto del proyecto, este recurso fue utilizado para presentar las recetas dentro del recetario digital, ofreciendo una interfaz ordenada, clara y fácil de navegar para el usuario.

4. Temas y estilos

Después de tener las pantallas funcionales, me concentré en el aspecto visual. Aprendí a trabajar con **ThemeData** para definir colores y tipografías.

Quise que mi aplicación transmitiera un estilo cálido, propio de un recetario casero. Configuré una paleta de tonos amarillos y blancos, y agregué una tipografía más amigable usando Google Fonts.

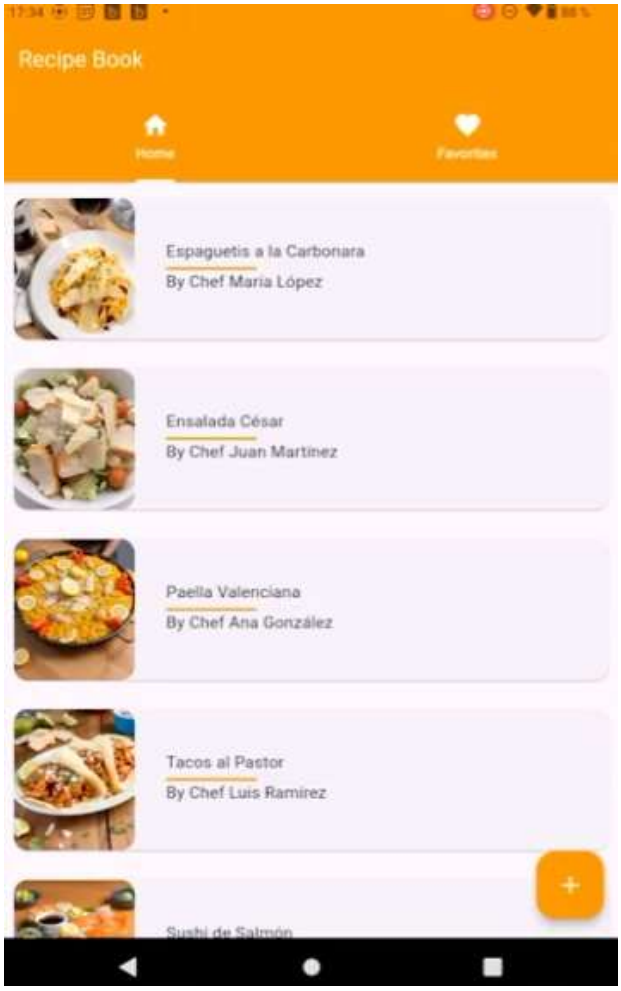

Figura 3. (interfaz principal de la aplicación de recetario desarrollada en Flutter)

La captura de pantalla presenta la vista general de la aplicación construida durante el proyecto. En ella se observa la integración de varios conceptos de diseño aplicados con Flutter: el uso de listas dinámicas para mostrar recetas, la disposición visual mediante widgets de organización, la personalización de colores y estilos para la barra superior y

botones, así como la inclusión de imágenes que acompañan cada elemento. Esta interfaz refleja la materialización práctica de los aprendizajes adquiridos, mostrando un diseño funcional, atractivo y centrado en la experiencia del usuario.

5. Gestión de estado y Clean Architecture

A medida que la aplicación fue creciendo, comprendí la importancia de mantener un control organizado sobre los datos y las interacciones del usuario. No bastaba con mostrar información en pantalla, sino que era necesario establecer una lógica clara que permitiera actualizar la interfaz de forma consistente y sin errores.

En este proceso, aprendí que existen diferentes enfoques para manejar el estado de una aplicación en Flutter. Algunos son más simples y funcionan en proyectos pequeños, mientras que otros permiten mayor escalabilidad y organización en aplicaciones más complejas. Lo esencial fue entender que una buena gestión del estado evita duplicación de código y facilita la comunicación entre las distintas partes de la interfaz.

De la mano con esto, también descubrí la importancia de aplicar principios de **arquitectura limpia (Clean Architecture)**. Al dividir la aplicación en capas (presentación, dominio y datos), logré mantener una estructura más ordenada. La capa de presentación se centró en los widgets y la interfaz de usuario; la capa de dominio en las reglas de negocio, como agregar o eliminar recetas; y la capa de datos en el acceso a información almacenada localmente o consultada desde servicios externos.

Gracias a esta organización, el proyecto se volvió más claro y fácil de mantener, y sobre todo me permitió comprender cómo estructurar aplicaciones que puedan crecer en el futuro sin convertirse en código difícil de manejar.

6. Almacenamiento local

También implementé el **almacenamiento local**, porque quería que el usuario pudiera acceder a sus recetas incluso sin internet. Probé varias opciones y finalmente usé **Hive**, ya que era rápido y fácil de usar.

Con Hive logré guardar las recetas favoritas en el dispositivo, de manera que cuando el usuario cerraba y volvía a abrir la aplicación, las recetas seguían disponibles. Esta característica le dio más valor práctico a la app.

7. Reflexión final

El desarrollo de esta aplicación fue una experiencia enriquecedora. No se trató solo de aprender qué es Flutter, sino de poner en práctica cada concepto paso a paso: instalar el entorno, crear interfaces, manejar el estado, consumir servicios, guardar datos y proteger la información del usuario.

El resultado fue una aplicación funcional que demuestra cómo la teoría se puede transformar en una herramienta útil. Más allá del recetario en sí, el aprendizaje me permitió

comprender que Flutter no es solo un framework, sino una manera diferente de pensar y construir aplicaciones. Finalmente anexo algunos pantallazos de lo que fue mi aplicación.

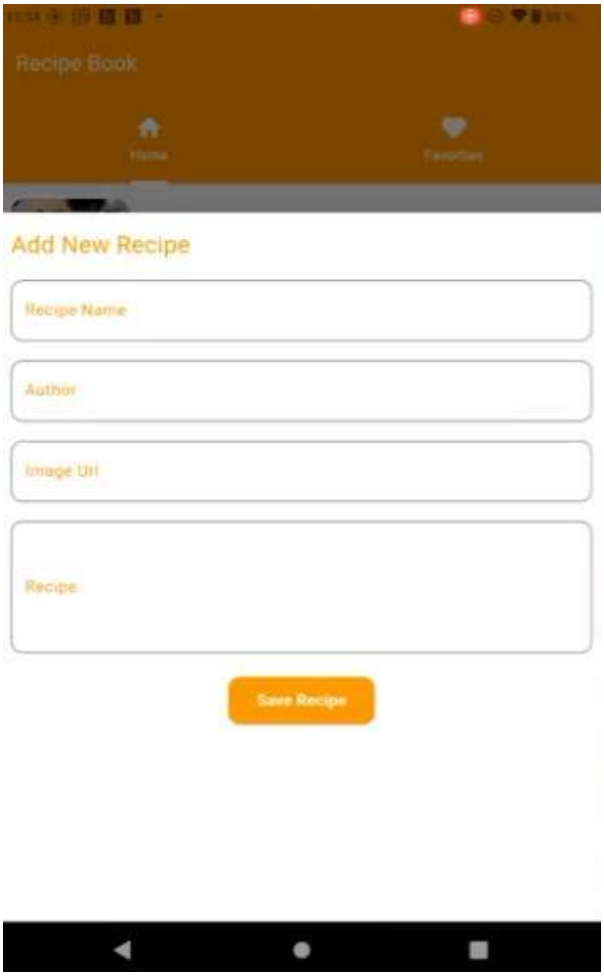

Figura 4. (Formulario de creación de una nueva receta)

La imagen muestra la interfaz destinada al registro de nuevas recetas dentro del recetario digital. El formulario incluye campos para ingresar el nombre de la receta, el autor, la URL de la imagen y la descripción detallada de la preparación. Finalmente, un botón de acción permite guardar la información en la base de datos de la aplicación. Esta funcionalidad

representa un paso esencial en el flujo de uso, ya que posibilita que cada usuario pueda personalizar y ampliar su propio recetario con contenidos propios o de su preferencia.

Figura 5. (Vista ilustrativa e informativa individual de recetas)

En esta pantalla se presenta la información completa de una receta previamente registrada en el recetario. Se observa la imagen del plato como elemento principal, acompañada del título, el autor y los pasos detallados para su preparación. Además, en la parte superior se mantiene la barra de navegación, lo que permite volver a la lista general o marcar la receta

como favorita. Esta vista representa el núcleo de la experiencia de usuario, ya que concentra los datos que hacen útil la aplicación: la consulta clara y ordenada de cada receta con todos sus elementos visuales y textuales.

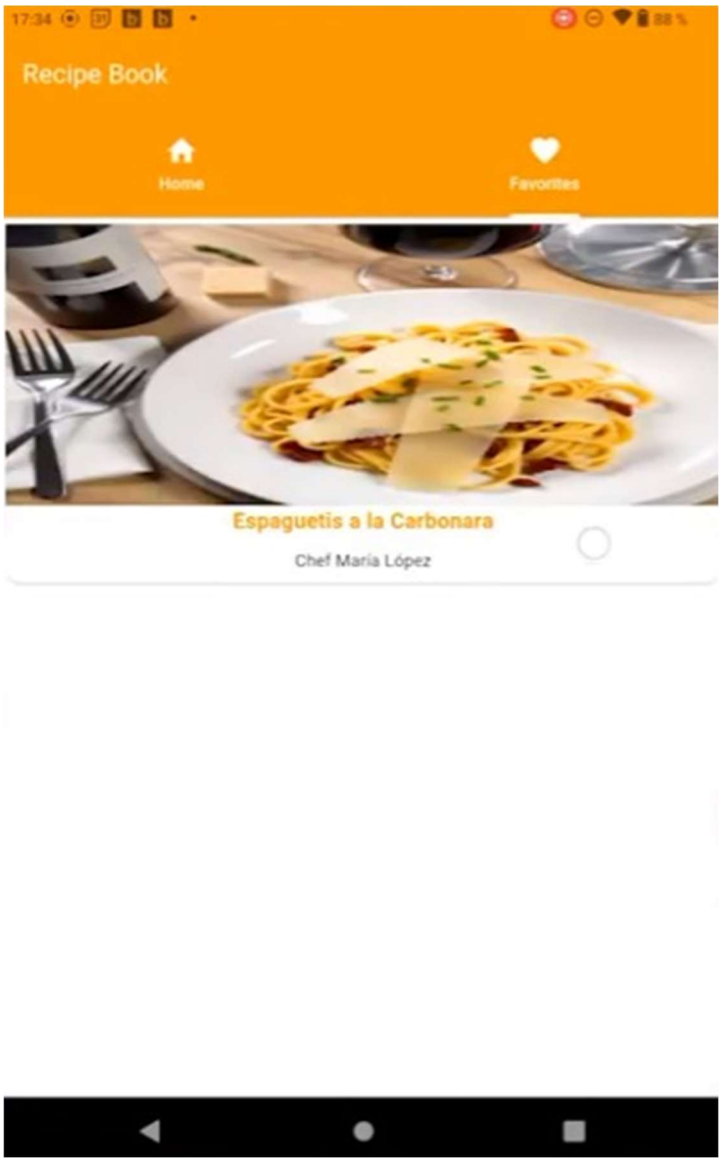

Figura 6. (Vista de recetas marcadas como favortias en la aplicación)

La pantalla muestra la sección destinada a las recetas que el usuario ha marcado como favoritas. En este caso, se visualiza la receta de “Espaguetis a la Carbonara”, almacenada en esta lista especial para un acceso rápido y personalizado. La interfaz conserva la misma coherencia de diseño que la vista principal, con la barra de navegación superior y el ícono de favoritos resaltado. Esta funcionalidad le otorga al usuario la posibilidad de organizar su recetario de manera más práctica, priorizando aquellas preparaciones de mayor interés.

Conclusiones.

- Flutter se consolida como un framework multiplataforma eficiente, capaz de generar aplicaciones nativas desde una única base de código, optimizando tiempos y costos de desarrollo.
- La instalación y configuración del entorno son procesos accesibles que permiten a los desarrolladores integrarse rápidamente al ecosistema de Flutter, facilitando el inicio de proyectos.
- La arquitectura basada en widgets ofrece una manera intuitiva y flexible de construir interfaces de usuario, garantizando modularidad y reutilización de componentes.
- Los widgets de disposición y diseño visual posibilitan la creación de interfaces modernas, adaptables y responsivas, ajustándose a distintas pantallas y dispositivos.

- La implementación de temas y estilos centralizados aporta coherencia visual a las aplicaciones, reforzando la identidad de marca y mejorando la experiencia de usuario.
- La gestión del estado es fundamental para mantener aplicaciones reactivas y dinámicas; soluciones como Provider, Riverpod o Bloc permiten una separación clara de responsabilidades.
- La aplicación de principios de Clean Architecture en Flutter asegura escalabilidad y mantenibilidad, favoreciendo el desarrollo colaborativo y la evolución de proyectos a largo plazo.
- La integración de APIs REST mediante librerías como *http* o *Dio* extiende las capacidades de las aplicaciones al permitir el consumo y la actualización de información en tiempo real.
- El almacenamiento local mediante herramientas como SharedPreferences, SQLite o Hive ofrece soluciones diversas según la naturaleza y complejidad de los datos a gestionar.
- La autenticación de usuarios en Flutter, especialmente con servicios como Firebase Authentication, refuerza la seguridad y permite gestionar accesos personalizados en las aplicaciones.
- La experiencia práctica con la aplicación de recetario evidencia que la teoría y las herramientas de Flutter son aplicables de manera exitosa en proyectos reales de uso cotidiano.

- En conjunto, Flutter se perfila como una tecnología versátil y robusta que no solo facilita el desarrollo de aplicaciones móviles, sino que también potencia la innovación en el diseño y la implementación de soluciones digitales.

Anexos

Anexo 1. Código de popUp del formulario para la creación de un nuevo recetario

```
47      Future<void> _showBottom(BuildContext context) {
48        return showModalBottomSheet(
49            isScrollControlled: true,
50            context: context,
51            builder: (contexto) => Container(
52                  width: MediaQuery.of(context).size.width,
53                  height: 600,
54                  color: Colors.white,
55                  child: const RecipeForm(),
56                ));
57      }
58
59      Widget _RecipesCard(BuildContext context, dynamic recipe) {
60        return GestureDetector(
61          onTap: () {
62            Navigator.push(context, MaterialPageRoute(builder: (context) => RecipeDetail(recipesData: recipe
63          )));
64          },
65          child: Padding(
66            padding: const EdgeInsets.all(8.0),
67            child: SizedBox(
68              width: MediaQuery.of(context).size.width,
```

Anexo 2. Configuración de ruta para imagenes del proyecto en la carpeta assets

```
64
65      # To add assets to your application, add an assets section, like this:
66      assets:
67      - assets/images/
68      #   - images/a_dot_ham.jpeg
69
```

Anexo 3. Programación de navegación entre vistas

```
51      Widget build(BuildContext context) {
52        return Scaffold(
53          appBar: AppBar(
54            title: Text(
55              widget.recipesData.name,
56              style: TextStyle(color: Colors.white),
57            ),
58            backgroundColor: Colors.orange,
59            leading: IconButton(
60                icon: const Icon(Icons.arrow_back),
61                color: Colors.white,
62                onPressed: () {
63                  Navigator.pop(context);
64                }),
65            actions: [
66              IconButton(
67                  onPressed: () async {
68                    await Provider.of<RecipesProvider>(context, listen: false)
69                        .toggleFavoriteStatus(widget.recipesData);
70                    setState(() {
71                      isFavorite = !isFavorite;
72                    });
73                  },
74                  icon: ScaleTransition(
75                      scale: _scaleAnimation,
76                      child: Icon(
77                        isFavorite ? Icons.favorite : Icons.favorite_border,
78                        color: Colors.red,
```

Referencias

Google. (2024). *Flutter documentation*. Google Developers. https://docs.flutter.dev

Amazon Web Services. (2024). *¿Qué es Flutter? Explicación de la aplicación Flutter*. https://aws.amazon.com/es/what-is/flutter/

MaureDev. (2022). *Flutter: Cómo crear una app desde cero (para principiantes)* [Video]. YouTube. https://www.youtube.com/watch?v=-pWSQYpkkjk