salesians
SARRIÀ

Programación multimedia y dispositivos móviles.

Flutter multiplataforma + UI Builder con FlutterFlow.

Alumno: Damian Altamirano Ontiveros Rivero.

Índice

0. Introducción

En este trabajo exploraremos el funcionamiento de Flutter como entorno de desarrollo multiplataforma, comprendiendo cómo el lenguaje Dart permite crear aplicaciones que se ejecutan en diferentes sistemas operativos a partir de un mismo código fuente.

Además, pondremos en práctica estos conceptos compilando una aplicación para Windows y diseñando interfaces gráficas con FlutterFlow.io, una herramienta visual que facilita la creación de estructuras y pantallas sin necesidad de programar directamente. El objetivo es entender tanto la base técnica de Flutter como su enfoque en la experiencia de usuario (UX).

1. Flutter como herramienta multiplataforma

Flutter es un framework multiplataforma desarrollado por Google que permite crear aplicaciones para móviles, ordenadores y la web usando un único lenguaje: Dart.

Esto significa que el mismo código puede ejecutarse en Android, iOS, Windows, macOS, Linux o navegadores web, sin necesidad de reescribirlo para cada sistema. Esta característica es posible gracias a los distintos tipos de compilación que ofrece Dart (nativa y web) y a los motores gráficos integrados en Flutter, que renderizan la interfaz de manera uniforme en todas las plataformas.

Esto a grandes rasgos supone un ahorro de tiempo y recursos gigante, ya que podremos desarrollar una misma aplicación capaz de poder ejecutarse en cualquier sistema o interfaz operativa.

1.1 Dart Native vs Dart Web

Dart posee dos variantes de compilación y ejecución que varían en función de la plataforma en la que se realice. Estas son Dart Native y Dart Web.

1.1.1 Dart Native

Es la forma en la que Dart se compila directamente a código nativo, como Anddroid, iOS, macOS, Windows o Linux. Este usa la máquina virtual de Dart (Dart VM) para ejecutar el código durante el desarrollo, el cual se compila AOT a binarios nativos, lo que mejora su rendimiento. Es usado mayormente en Flutter para apps móviles y de escritorio.

1.1.2 Dart Web

Es la forma en la que Dart se compila a JavaScript para ejecutarse en navegadores web, lo que permite crear interfaces web modernas usando el mismo lenguaje que se usa en Flutter. Se usa en Flutter Web y también en proyectos Dart puros para frontend.

1.2.3 Diferencias

Las pondremos en una tabla para poder visualizarlas mejor:

Característica	Dart Native	Dart Web
Plataforma destino	Móviles (iOS/Android), escritorio	Navegadores web
Compilación	AOT a binario nativo	Transpilación a JavaScript
Rendimiento	Alto (ejecución nativa)	Depende del motor JS del navegador
Uso principal	Flutter apps móviles/escritorio	Flutter Web, apps web interactivas
Acceso al sistema	Acceso completo (archivos, red, etc)	Limitado por el entorno del navegador

1.2 JIT

JIT (Just–In-Time), significa “compilación en tiempo de ejecución”. Eso quiere decir que el código fuente proveniente de Dart se compila mientras la aplicación se ejecute. Esto permite que, durante las fases de desarrollo, pueda modificarse el código y de esta manera se vea reflejado los cambios inmediatamente sin la necesidad de recompilar, acelerando de esta manera esta fase del desarrollo. Sin embargo, el rendimiento de esta no es tan alto comparada con el producto final.

1.3 AOT

AOT (Ahead-Of-Time), significa “compilación anticipada”. En este caso, el código se compila antes de ejecutar la aplicación, generando código máquina nativo optimizado para cada plataforma (.exe en Windows, . apk en Android, etc...). De esta manera la aplicación se ejecuta en menor tiempo y consume menos recursos, ya que nada se compilará durante la ejecución. Esta herramienta se utiliza para crear la versión final de una app o distribución.

2. Aplicación Windows con Flutter.

Ahora, lo que haremos será comprobar que una aplicación tipo contador que ya tenemos hecha es capaz de funcionar en un ordenador Windows.

2.1 Requisitos

Anticipadamente, deberemos de tener unos requisitos necesarios para que la compilación se efectúe correctamente. Para ello, primero abriremos la barra de comandos de nuestro ordenador y ejecutaremos el comando

|| $ flutter doctor

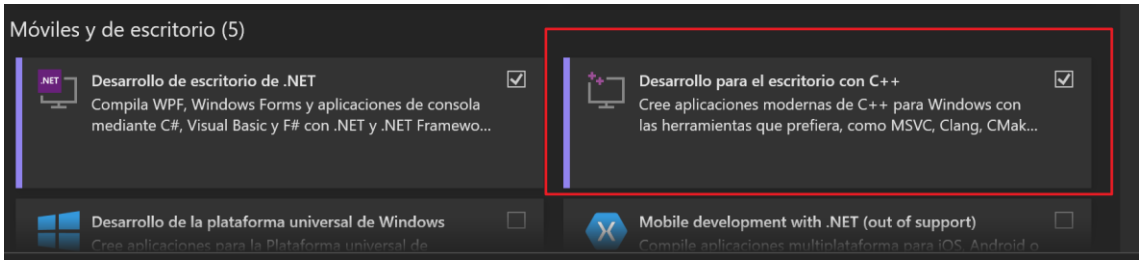
Y nos retornará algo unos resultados tal que así:

```
PS C:\Users\dalta> flutter doctor

  A new version of Flutter is available!

  To update to the latest version, run "flutter upgrade".

Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.35.5, on Microsoft Windows [Versi¢n 10.0.26100.6584], locale es-ES)
[✓] Windows Version (Windows 11 or higher, 24H2, 2009)
[✓] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2019 16.11.51)
[✓] Android Studio (version 2025.1.3)
[✓] VS Code (version 1.105.0)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!
```

Este comando lanza un diagnóstico que evalúa si el sistema tiene los requerimientos necesarios para poder hacer ciertas compilaciones. En nuestro caso, queremos que nuestro programa tenga esta opción en “check”:

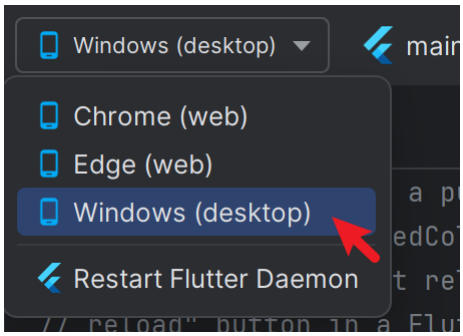
```
[✓] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2019 16.11.51)
[✓] Android Studio (version 2025.1.3)
[✓] VS Code (version 1.105.0)
```

Si no fuera así, deberemos abrir el instalador de Visual Studio y modificarlo, marcando esta opción y dándole a instalar:

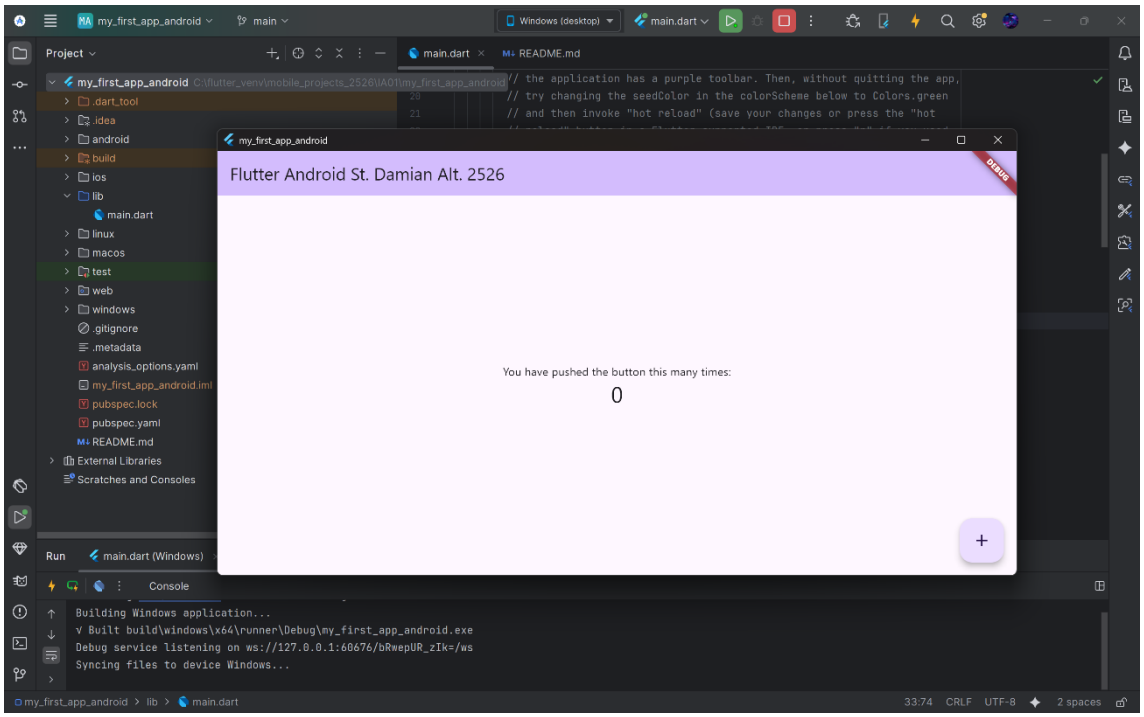


2.2 Validación

Hecho esto, nos vamos al IDE de android studio, y abrimos nuestro programa. Una vez abierto daremos click a la opción **FlutterDeviceSelection > Windows (desktop)**.



Daremos click en **Run** y si todo esta bien configurado se compilara nuestro archivo y podremos interactuar con nuestro programa:



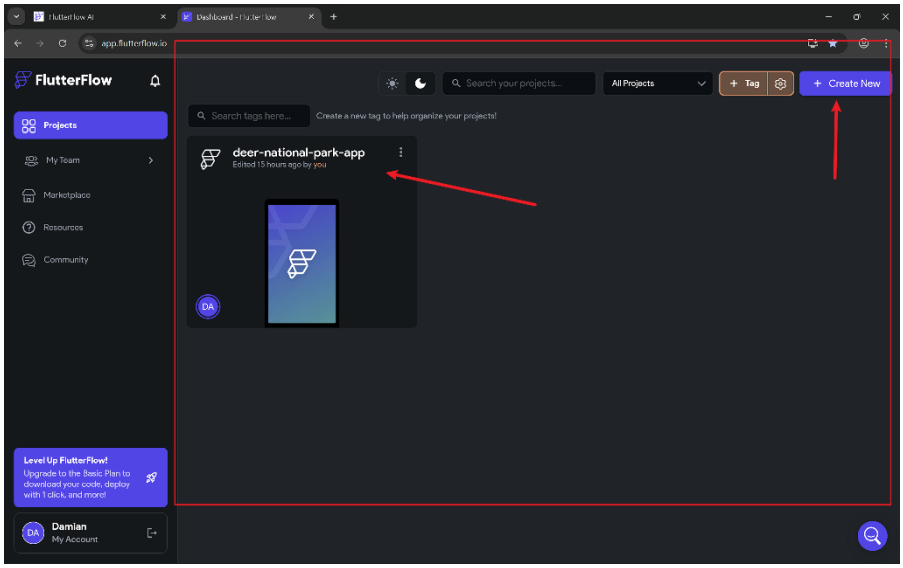
3. Interfaz de usuario con FlutterFlow.io

FlutterFlow.io es una plataforma visual de desarrollo que permite crear aplicaciones móviles y web utilizando Flutter sin necesidad de escribir código desde cero. Su interfaz intuitiva facilita el diseño de interfaces, la integración de bases de datos y la lógica de negocio, acelerando el proceso de prototipado y despliegue. Es ideal para desarrolladores que buscan eficiencia sin sacrificar personalización ni escalabilidad.

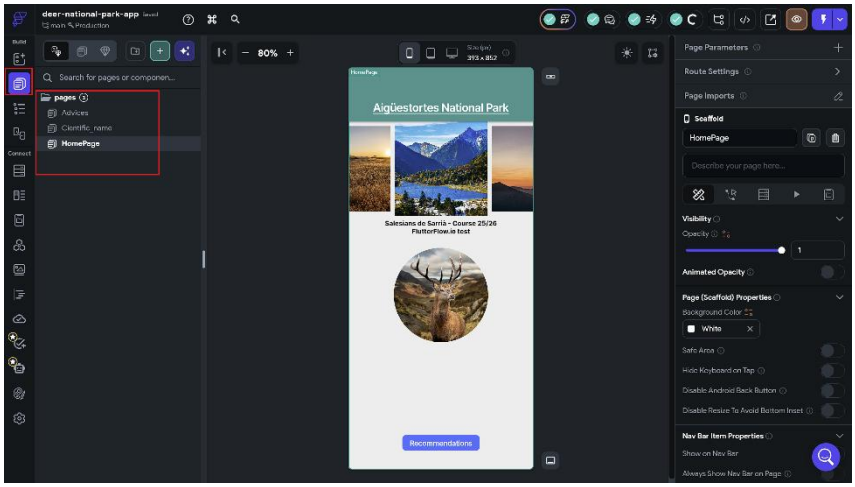
En este apartado crearemos un primer proyecto con tres “screens”. Seguiremos un flujo básico he iremos viendo herramienta y conceptos fundamentales para usar esta herramienta.

3.1 Deer National Park

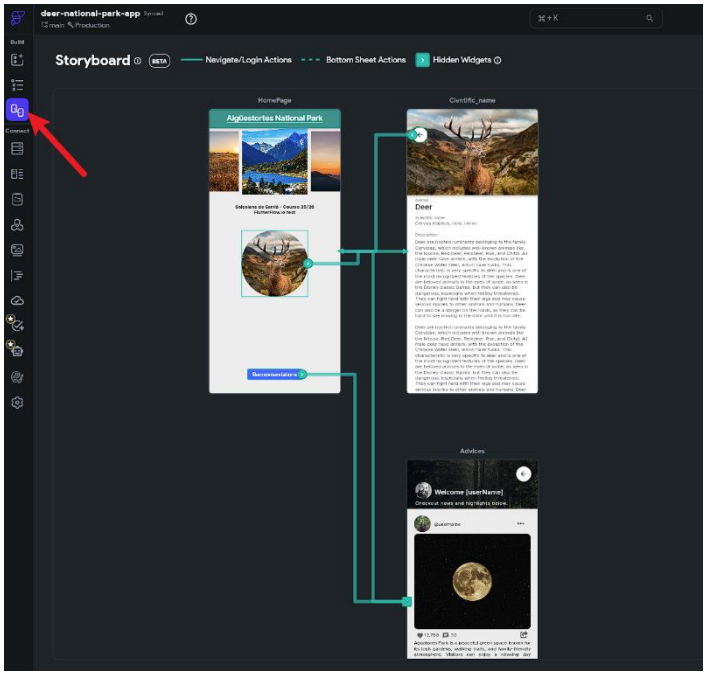
Una vez nos hayamos registrado, crearemos un nuevo proyecto en la interfaz de **Projects**.



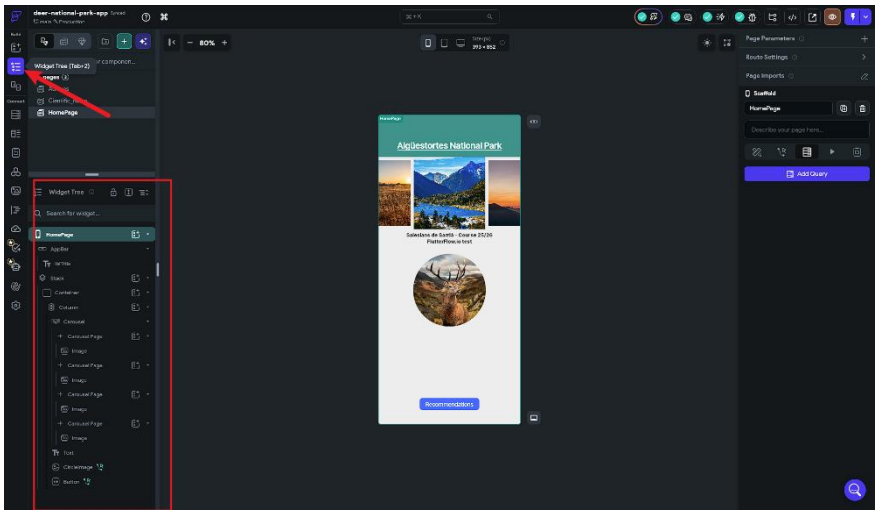
Ahora, echaremos un vistazo a la interfaz general de FlutterFlow.io. Una vez dentro del proyecto, dentro de la barra lateral nos iremos a **Page Selector**, donde allí podremos ver todas nuestras páginas:



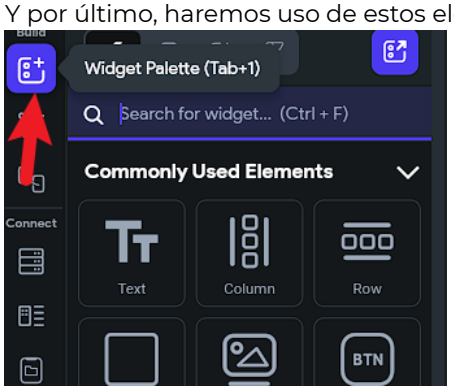
Desde la sección **Storyboard** podremos ver el flujo de nuestro proyecto:



Encontraremos nuestro espacio de gestión de herramientas y componentes la encontraremos en la sección **Widget tree**:



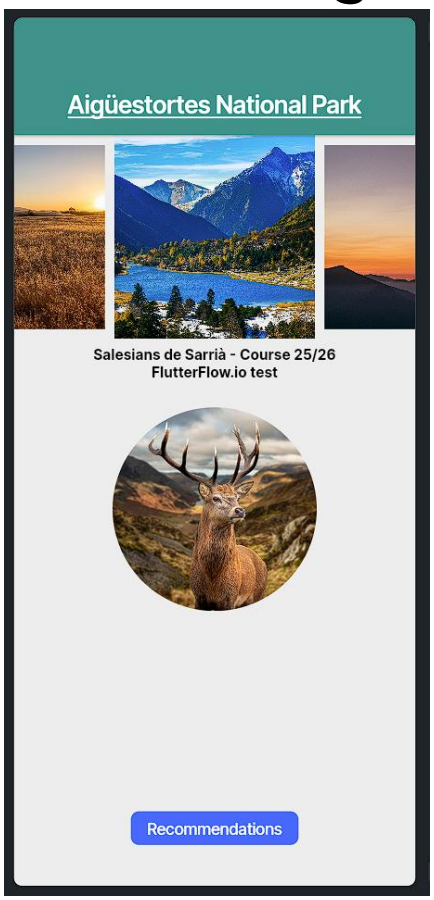
Y por último, haremos uso de estos elementos desde la sección **Widget Palette**, donde



tendremos elementos de todo tipo para usar e integrar directamente desde esta pestaña, además de contar con un pequeñó browser para buscar cosas en específico.

Con todo esto visto, empezarémos a analizar los componentes y elementos que usamos para desarrollar este proyecto.

3.2 HomePage



3.2.1 AppBar

Es la barra superior de una pantalla que muestra el título, íconos de navegación, acciones (como búsqueda o ajustes) y puede incluir estilos personalizados. Es útil para orientar al usuario y ofrecer accesos rápidos dentro de la app.

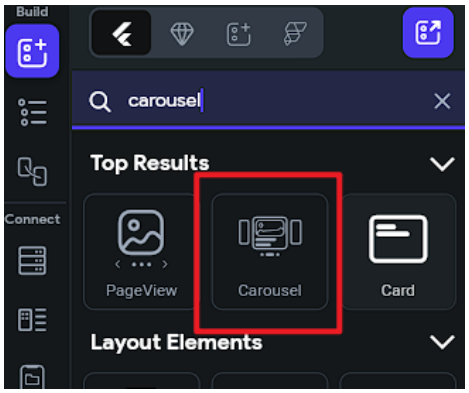
Empezamos destacando que esta página tendrá una App Bar donde contendrá por default el nombre de nuestra aplicación.

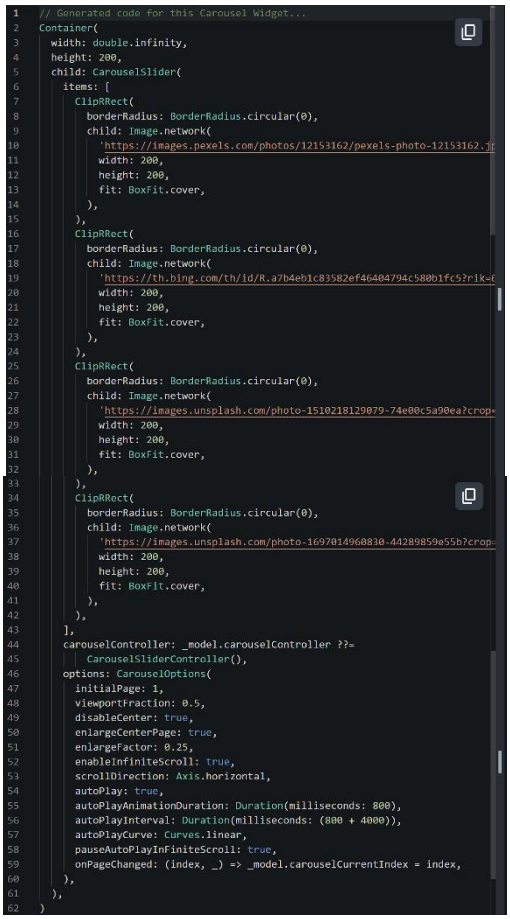
```
// Generated code for this AppBar Widget...
AppBar(
  backgroundColor: Color(0xFF3F938B),
  automaticallyImplyLeading: false,
  title: Align(
    alignment: AlignmentDirectional(0, 0),
    child: Padding(
      padding: EdgeInsetsDirectional.fromSTEB(0, 25, 0, 0),
      child: Text(
        'AigÃ¼estortes National Park\n',
        textAlign: TextAlign.center,
        style: FlutterFlowTheme.of(context).headlineSmall.override(
              font: GoogleFonts.interTight(
                fontWeight:
                    FlutterFlowTheme.of(context).headlineSmall.fontWeight,
                fontStyle:
                    FlutterFlowTheme.of(context).headlineSmall.fontStyle,
              ),
              color: Colors.white,
              letterSpacing: 0.0,
              fontWeight:
                  FlutterFlowTheme.of(context).headlineSmall.fontWeight,
              fontStyle: FlutterFlowTheme.of(context).headlineSmall.fontStyl
              decoration: TextDecoration.underline,
            ),
      ),
    ),
  ),
  actions: [],
  centerTitle: false,
  elevation: 2,
)
```

3.2.2 Carousel

Es un componente visual que permite mostrar varias imágenes, tarjetas o widgets en forma de carrusel deslizante. Los usuarios pueden desplazarse horizontalmente para ver diferentes elementos, lo que lo hace ideal para destacar productos, fotos o contenido destacado en una app.

Aquí la usaremos para almacenar las fotos de la reserva natural.

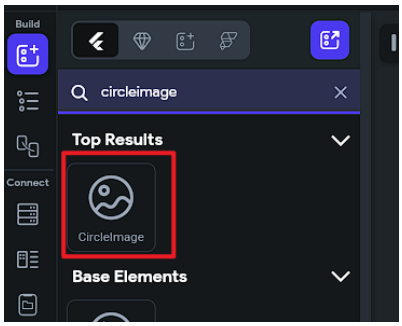




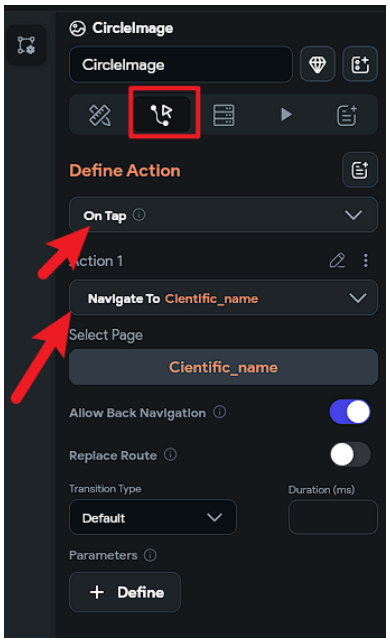
```
// Generated code for this Carousel Widget...
Container(
  width: double.infinity,
  height: 200,
  child: CarouselSlider(
    items: [
      ClipRRect(
        borderRadius: BorderRadius.circular(0),
        child: Image.network(
          'https://images.pexels.com/photos/12153162/pexels-photo-12153162.j
          width: 200,
          height: 200,
          fit: BoxFit.cover,
        ),
      ),
      ClipRRect(
        borderRadius: BorderRadius.circular(0),
        child: Image.network(
          'https://th.bing.com/th/id/R.a7b4eb1c83582ef46404794c580b1fc5?rik=
          width: 200,
          height: 200,
          fit: BoxFit.cover,
        ),
      ),
      ClipRRect(
        borderRadius: BorderRadius.circular(0),
        child: Image.network(
          'https://images.unsplash.com/photo-1510218129079-74e00c5a90ea?crop=
          width: 200,
          height: 200,
          fit: BoxFit.cover,
        ),
      ),
      ClipRRect(
        borderRadius: BorderRadius.circular(0),
        child: Image.network(
          'https://images.unsplash.com/photo-1697014960830-44289859a55b?crop=
          width: 200,
          height: 200,
          fit: BoxFit.cover,
        ),
      ),
    ],
    carouselController: _model.carouselController ??=
        CarouselSliderController(),
    options: CarouselOptions(
      initialPage: 1,
      viewportFraction: 0.5,
      disableCenter: true,
      enlargeCenterPage: true,
      enlargeFactor: 0.25,
      enableInfiniteScroll: true,
      scrollDirection: Axis.horizontal,
      autoPlay: true,
      autoPlayAnimationDuration: Duration(milliseconds: 800),
      autoPlayInterval: Duration(milliseconds: (800 + 4000)),
      autoPlayCurve: Curves.linear,
      pauseAutoPlayInFiniteScroll: true,
      onPageChanged: (index, _) => _model.carouselCurrentIndex = index,
    ),
  ),
)
```

3.2.3 CircleImage

Es un widget que muestra una imagen dentro de un marco circular. Es ideal para fotos de perfil, íconos o elementos visuales que necesitan un estilo redondeado y limpio. Puedes personalizar su tamaño, borde y fuente de imagen (local o URL).



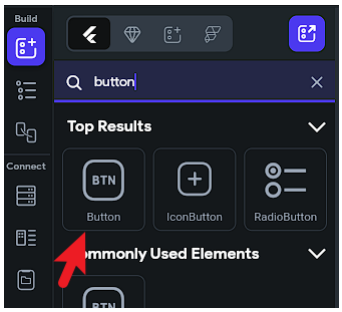

```dart
// Generated code for this CircleImage Widget...
Align(
  alignment: AlignmentDirectional(0, 0),
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(0, 20, 0, 0),
    child: InkWell(
      splashColor: Colors.transparent,
      focusColor: Colors.transparent,
      hoverColor: Colors.transparent,
      highlightColor: Colors.transparent,
      onTap: () async {
        context.pushNamed(ScientificInfoWidget.routeName);
      },
      child: Container(
        width: 200,
        height: 200,
        clipBehavior: Clip.antiAlias,
        decoration: BoxDecoration(
          shape: BoxShape.circle,
        ),
        child: Image.network(
          'https://media.istockphoto.com/id/1196316447/es/foto/majestuoso-pai
          fit: BoxFit.cover,
        ),
      ),
    ),
  ),
)
```

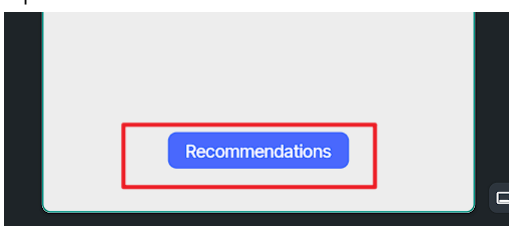


Además, a este widget tendremos asociado un evento de Acción, en el que, a la hora de pulsar sobre este, no redirigirá a otra pestaña de nuestro proyecto. Esta pestaña la podremos encontrar en el panel derecho, en la sección **Actions**:

3.2.4 Button

Es un componente interactivo que permite al usuario ejecutar una acción al tocarlo, como navegar a otra pantalla, enviar un formulario o activar una función. Puedes personalizar su texto, color, forma, ícono y comportamiento fácilmente desde el editor visual.

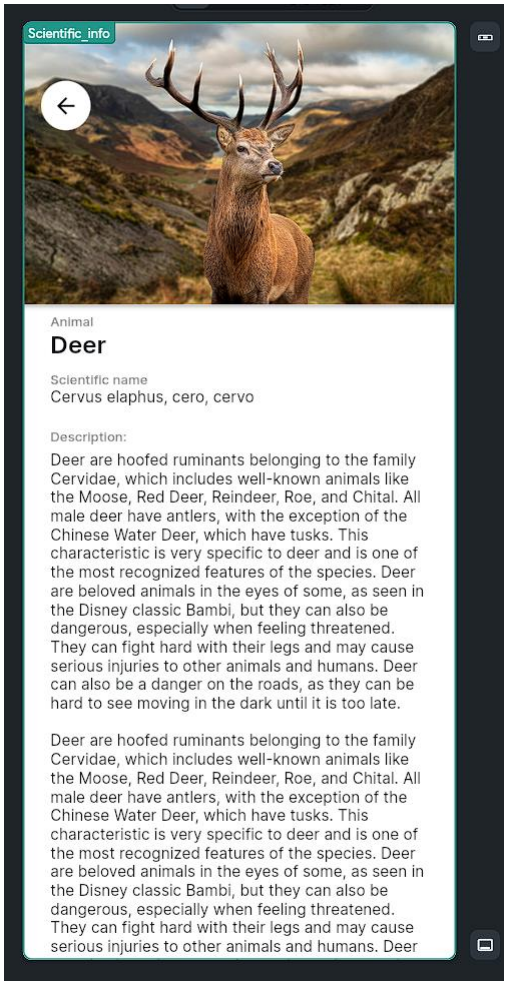




En nuestro caso, al igual que el anterior widget, tendremos enlazado un evento de acción que se desencadenará cuando pulsemos sobre él, y nos dirigirá a otra página de nuestro proyecto.

```dart
// Generated code for this Button Widget...
Align(
  alignment: AlignmentDirectional(0, 1),
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(0, 190, 0, 40),
    child: FFButtonWidget(
      onPressed: () async {
        context.pushNamed(AdvicesWidget.routeName);
      },
      text: 'Recommendations',
      options: FFButtonOptions(
        height: 33,
        padding: EdgeInsetsDirectional.fromSTEB(16, 0, 16, 0),
        iconPadding: EdgeInsetsDirectional.fromSTEB(0, 0, 0, 0),
        color: Color(0xFF4968FD),
        textStyle: FlutterFlowTheme.of(context).titleSmall.override(
              font: GoogleFonts.interTight(
                fontWeight: FontWeight.w500,
                fontStyle: FlutterFlowTheme.of(context).titleSmall.fontStyle,
              ),
              color: Colors.white,
              letterSpacing: 0.0,
              fontWeight: FontWeight.w500,
              fontStyle: FlutterFlowTheme.of(context).titleSmall.fontStyle,
            ),
        elevation: 0,
        borderRadius: BorderRadius.circular(8),
      ),
    ),
  ),
```

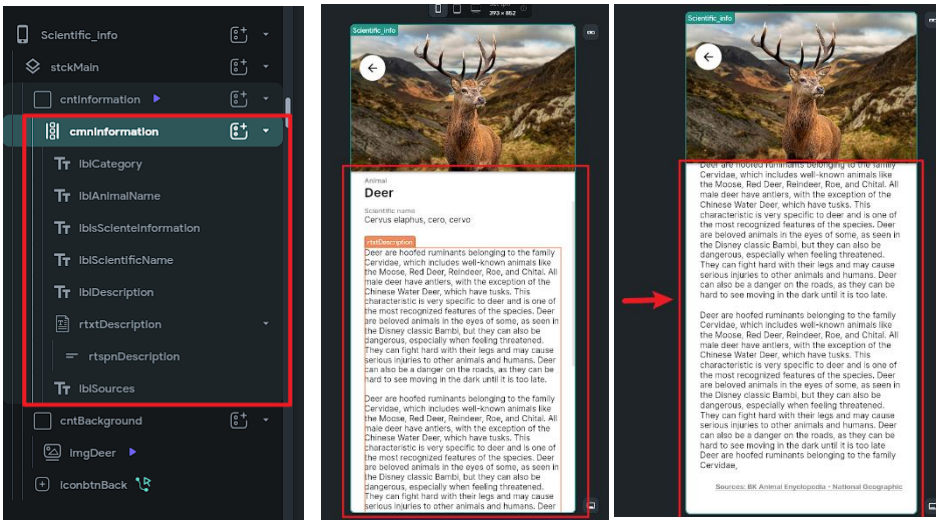
3.3 Scientific Name



3.3.1 Column

Es un widget que organiza sus hijos en una disposición vertical alineada de arriba abajo, permitiendo controlar alineación, espaciado y tamaño de cada elemento dentro de la

columna, y la función de scroll añade desplazamiento a una pantalla o a un contenedor cuando su contenido supera el espacio disponible para que el usuario pueda desplazarse vertical u horizontalmente (Scrolling) según la configuración, manteniendo la experiencia fluida y evitando desbordes de contenido.

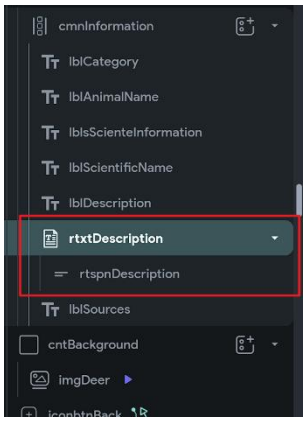


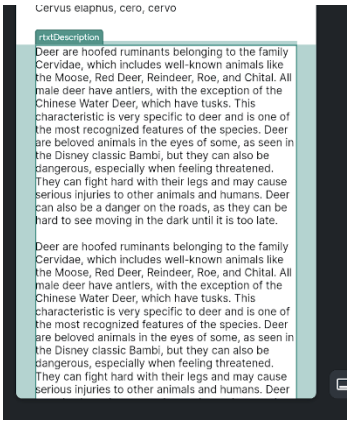
Hemos anidado todos los textos he información dentro de nuestra columna y habilitado la opción de “scrolling”. De esta manera podemos desplazarnos entre todos los elementos hijos sin obstruir otros elementos, como la fotografía.

```dart
// Generated code for this cmnInformation Widget...
SingleChildScrollView(
  child: Column(
    mainAxisSize: MainAxisSize.min,
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Align(
        alignment: AlignmentDirectional(-1, 0),
        child: Padding(
          padding: EdgeInsetsDirectional.fromSTEB(25, 15, 0, 0),
          child: Text(
            'Animal',
            style: FlutterFlowTheme.of(context).bodyMedium.override(
                  font: GoogleFonts.inter(
                    fontWeight: FontWeight.w500,
                    fontStyle:
                        FlutterFlowTheme.of(context).bodyMedium.fontStyle,
                  ),
                  color: Color(0xFF8F8F8F),
                  fontSize: 12,
                  letterSpacing: 0.0,
                  fontWeight: FontWeight.w500,
                  fontStyle:
                      FlutterFlowTheme.of(context).bodyMedium.fontStyle,
                ),
          ),
        ),
      ),
```

3.3.2 RichText

Es un widget que permite mostrar texto con múltiples estilos dentro del mismo bloque, combinando diferentes fuentes, tamaños, colores, pesos y enlaces en segmentos independientes para crear cabeceras mixtas, frases con énfasis o texto con hipervínculos sin necesitar widgets.



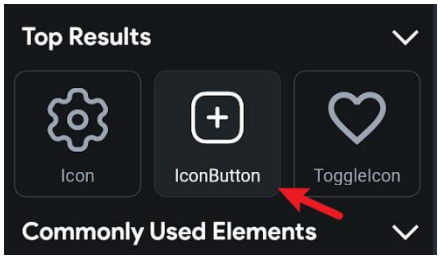


En mi caso, lo he utilizado para almacenar toda la descripción del cervatillo.

```
// Generated code for this rtxtDescription Widget...
Align(
  alignment: AlignmentDirectional(0, -1),
  child: Padding(
    padding: EdgeInsetsDirectional.fromSTEB(25, 5, 25, 20),
    child: RichText(
      textScaler: MediaQuery.of(context).textScaler,
      text: TextSpan(
        children: [
          TextSpan(
            text:
                'Deer are hoofed ruminants belonging to the family Cervidae,
            style: FlutterFlowTheme.of(context).bodyMedium.override(
                  font: GoogleFonts.inter(
                    fontWeight:
                        FlutterFlowTheme.of(context).bodyMedium.fontWeight,
                    fontStyle:
                        FlutterFlowTheme.of(context).bodyMedium.fontStyle,
                  ),
                  letterSpacing: 0.0,
                  fontWeight:
                      FlutterFlowTheme.of(context).bodyMedium.fontWeight,
                  fontStyle:
                      FlutterFlowTheme.of(context).bodyMedium.fontStyle,
                ),
          )
        ]
```

3.3.3 IconButton

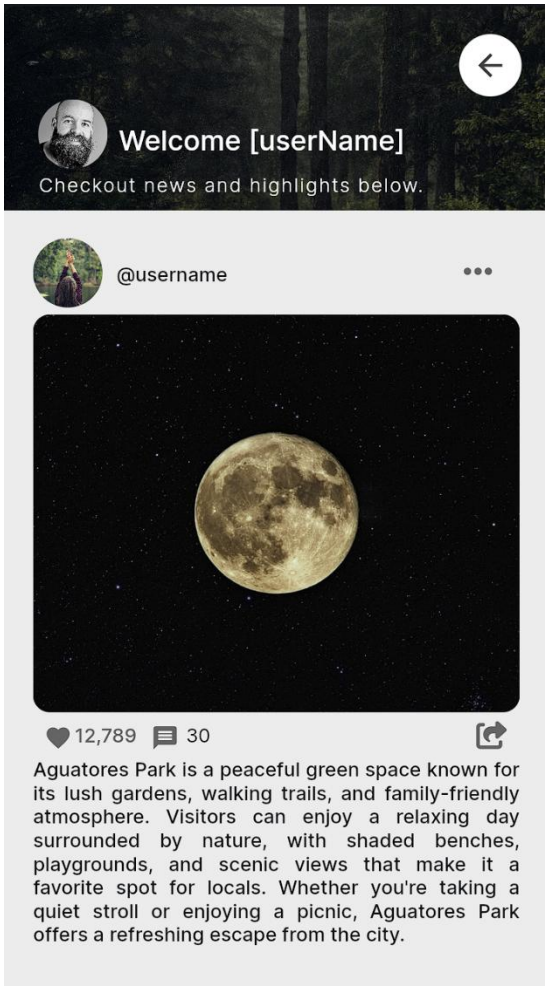
Es un widget interactivo que muestra solo un icono tocable para ejecutar una acción (navegar, abrir un modal, activar una acción personalizada), permite personalizar tamaño, color,



padding y borde, ofrece estado deshabilitado y tooltip para accesibilidad, y se puede conectar a flujos de acciones, visibilidad condicional y diseños responsivos dentro de la interfaz visual.

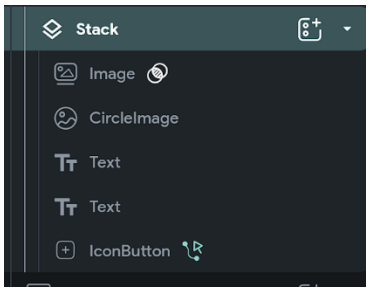
```dart
// Generated code for this iconbtnBack Widget...
Align(
  alignment: AlignmentDirectional(-0.91, -0.87),
  child: FlutterFlowIconButton(
    borderRadius: 25,
    buttonSize: 45,
    fillColor: Colors.white,
    icon: Icon(
      Icons.arrow_back,
      color: Colors.black,
      size: 24,
    ),
    onPressed: () async {
      context.pushNamed(
        HomePageWidget.routeName,
        extra: <String, dynamic>{
          kTransitionInfoKey: TransitionInfo(
            hasTransition: true,
            transitionType: PageTransitionType.fade,
            duration: Duration(milliseconds: 200),
          ),
        },
      );
```

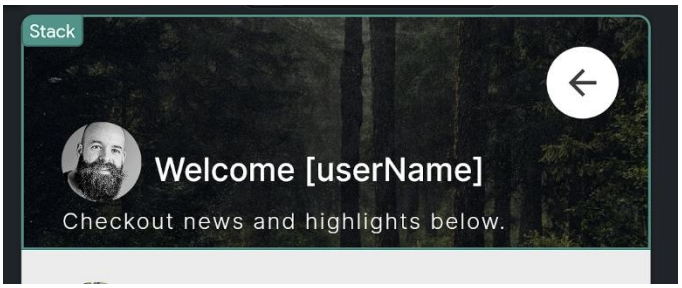
3.4 Recommendations



3.4.1 Stack

Es un contenedor que apila sus hijos en capas permitiendo posicionarlos unos sobre otros para crear layouts superpuestos, fondos con contenido encima, o elementos flotantes; admite controlar la alineación y el orden Z de los elementos, usar widgets Positioned para ubicar hijos con coordenadas relativas dentro del Stack, manejar gestos y visibilidad condicional en capas específicas, y combinarse con contenedores responsivos y constraints para mantener un comportamiento correcto en diferentes tamaños de pantalla.





En nuestro caso, usamos este elemento para poder organizar y superponer diferentes elementos de visualización de perfil dentro de un mismo espacio.

```dart
// Generated code for this Stack Widget...
Stack(
  alignment: AlignmentDirectional(0, 0),
  children: [
    Opacity(
      opacity: 0.6,
      child: Align(
        alignment: AlignmentDirectional(0, 0),
        child: ClipRRect(
          borderRadius: BorderRadius.circular(0),
          child: Image.network(
            'https://images.unsplash.com/photo-1545486332-9e0999c535b2?crop=e
            width: double.infinity,
            height: 185.4,
            fit: BoxFit.cover,
          ),
        ),
      ),
    ),
  ],
),
```

3.4.2 Container

Es un widget flexible que sirve como caja para otros widgets, permite definir tamaño, padding, margin, color de fondo, bordes y sombras, soporta alineación y transformaciones, y se usa para agrupar y dar estilo a elementos, controlar su comportamiento responsivo y aplicar decoración visual o gestos sobre áreas específicas de la interfaz.


```dart
// Generated code for this Container Widget...
Align(
  alignment: AlignmentDirectional(0, 0),
  child: Container(
    width: double.infinity,
    height: 560.1,
    decoration: BoxDecoration(
      color: Color(0xFFECECEC),
    ),
```

4. Divertirse con FlutterFlow.io

Dentro de este apartado crearemos un proyecto libre donde usaremos todas las herramientas que nos proporciona la plataforma de FlutterFlow.io para crear nuestra propia app. Además, realizaremos una pequeña demo demostración para el funcionamiento de esta en acción.

4.1 HiveStock Concepto

Esta idea nace de la necesidad de que tenemos por querer obtener un mejor control y gestión del núcleo principal de nuestros hogares, la cocina. Para ello, hemos creado HiveStock, un panalgestor inteligente que busca facilitar le administración de nuestros recursos y alimentos, así como las finanzas de estos.

Contará con un chef “inteligente” BuzzAI, que revisará lo que tengas en tu despensa en el momento, y creará recetas personalizadas en base a tus recursos, preferencias, objetivos, etc.

Y, al igual que el concepto colmena que sigue nuestra app, buscará conectar a todos los integrantes de la familia, gestionada por roles, para todos puedan hacer uso de las funciones de “BuzzAI”. Y no solo podrás crear y configurar tu “panal”, sino que ser parte de una comunidad entera de “Colmenas”, donde podrás compartir tus estrategias de ahorro, recetas, comidas, etc.

4.2 Interfaz UI

4.2.1 HomePage

Esta pantalla sirve como el "Núcleo de la Colmena", el punto central donde el usuario aterriza y visualiza un resumen de las actividades más importantes y un *Call to Action (CTA)* basado en el contexto. El saludo personalizado (*"Hello Damian"*) refuerza el concepto de hogar digital.

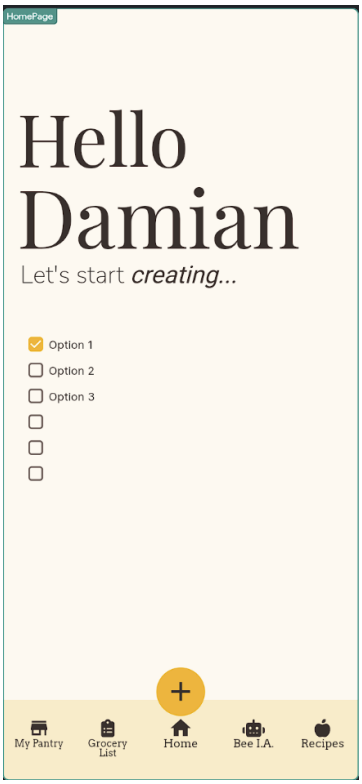
- Función Esencial: Servir como el Dashboard dinámico del ecosistema HiveStock.
- Contenido Dinámico: El subtítulo *(e.g., "Let's start cooking...", "Let's start storing...", "Let's start creating...")* y el contenido principal cambian según la hora del día, el

estado del inventario o las notificaciones pendientes. Por ejemplo, si hay un producto a punto de caducar, la pantalla podría sugerir recetas para usarlo.

- CTA Central (+): El botón flotante central grande es la puerta de entrada rápida a las acciones más comunes, como añadir un producto al HivePantry, iniciar una nueva receta o chatear con Chef BuzzAI.
- Metáfora: Representa la actividad en tiempo real del enjambre.

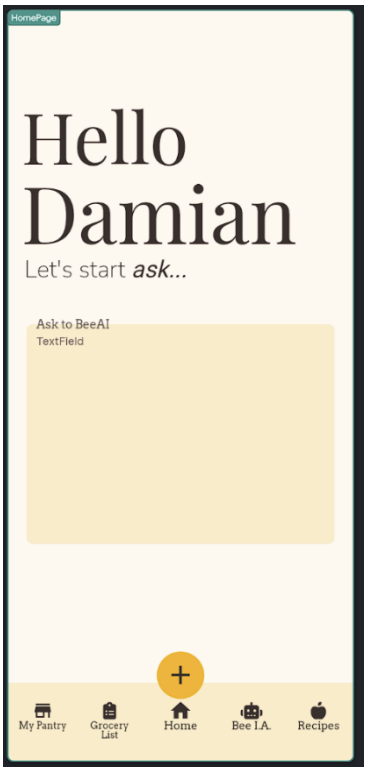
Esta es el "Corazón del Panal" y el módulo ancla de la aplicación. Su función es ofrecer una representación fiel y organizada del inventario físico del usuario, fundamental para la sostenibilidad y el cero desperdicio.

- Función Esencial: Control de Stock en tiempo real y gestión de caducidades.
- Organización Visual: La pantalla principal de HivePantry probablemente presenta tarjetas o iconos grandes para categorizar el inventario (e.g., Congelador, Nevera, Despensa Seca), representados por iconos como la copo de nieve (congelador) y el icono de comida general.
- Alertas Clave: En esta sección se priorizan las alertas de productos a punto de vencer (el "zumbido" de alerta de la colmena), sugiriendo acciones inmediatas para evitar el desperdicio.
- Integración IA: Es la base de datos que alimenta directamente a Chef BuzzAI, permitiéndole sugerir solo recetas que son realmente posibles con lo que hay en casa.



Este módulo es el "Zumbido Logístico del Enjambre". Convierte la necesidad de comprar en una tarea eficiente, inteligente y económica, enlazando directamente con el objetivo de ahorro.

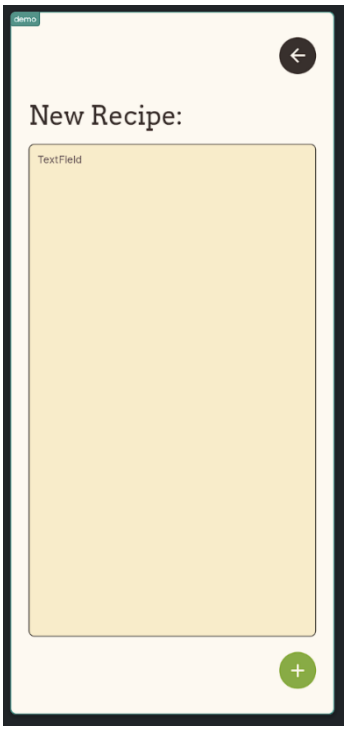
- Función Esencial: Generar listas de compra optimizadas y automatizadas.
- Automatización: La lista se actualiza automáticamente cuando un usuario programa una receta (ingredientes faltantes) o marca un producto como agotado en HivePantry.
- Inteligencia de Precios: Deberá mostrar la información crítica para el mercado español: precios y ofertas en los supermercados locales (APIs de supermercados), permitiendo al usuario ahorrar tiempo y dinero en su ruta de compra.
- Colaboración: Permite la sincronización en tiempo real para que todos los "Abejas Obreras" de la colmena familiar puedan ver, añadir o marcar artículos comprados.



Este es el "Cerebro Colectivo de la Colmena" y la interfaz con tu asistente culinario inteligente, la "Abeja Exploradora".

- Función Esencial: Ofrecer asistencia conversacional para la toma de decisiones culinarias y logísticas.
- Uso Primario: El usuario puede escribir preguntas o peticiones de comida ("Quiero algo dulce con las manzanas que me quedan" o "¿Qué hago con el pollo y la zanahoria?"). Chef BuzzAI analiza la petición contra el HivePantry y sugiere la mejor receta del Recetario o consejos de aprovechamiento.
- Experiencia: La interfaz de chat debe ser cálida y conversacional, reflejando el tono de inteligencia natural de la marca.
- Integración: Es el puente directo entre el inventario (HivePantry) y el Recetario Inteligente.

4.2.2 Recipe Book



Este módulo es el "Laboratorio Culinario del Enjambre". No es solo un lugar para guardar recetas, sino un centro de ideas vinculadas directamente a la acción.

- Función Esencial: Organizar recetas y enlazarlas automáticamente con el inventario.
- Vínculo Inteligente: Al seleccionar una receta, la aplicación descuenta automáticamente los ingredientes de HivePantry o añade los faltantes a la Lista de Compras.
- Contenido: El recetario contendrá las 200+ recetas base, las recetas sugeridas por Chef BuzzAI y las recetas creadas/guardadas por el usuario.
- Fomento de Creación: El formulario de "New Recipe" anima a los usuarios a contribuir a la "sabiduría culinaria" de la colmena.

Recursos

Demo Deer-National-Park-App

Enlace:

- DeerNationalPark_Demo

Demo HiveStock-App

Enlace:

- HiveStock_Demo

Repositorio GitHub

Enlace:

- Mobile_projects_2526