Diseño de layouts

En esta práctica, aprenderás cómo tomar widgets individuales y anidarlos para crear un diseño profesional. Este concepto se conoce como composición y es una gran parte de la creación de aplicaciones móviles Flutter. La mayoría de las veces puedes crear diseños simples o complejos, ya sea usando widgets verticales u horizontales o usando una combinación de ambos.

Una vista de alto nivel del diseño

El objetivo de esta práctica es crear una página de entrada de diario que muestre los detalles de arriba a abajo. La página del diario muestra la imagen del encabezado, el título, los detalles del diario, el clima, la dirección (ubicación del diario), las etiquetas y las imágenes del pie de página. Las secciones de clima, etiquetas e imágenes de pie de página se crean anidando widgets para crear un diseño personalizado (figura 10.1).

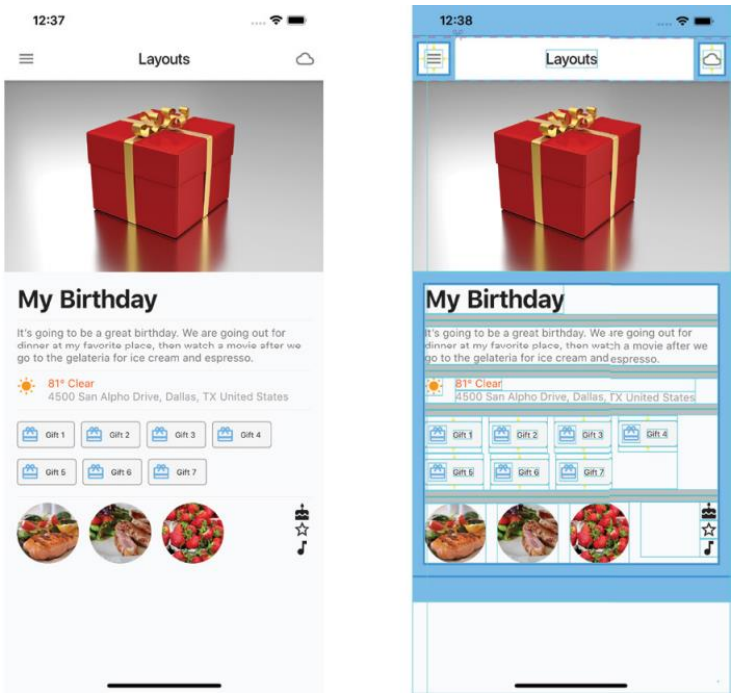


Figura 10.1: Diseño de la página de detalles de la revista.

Comenzando con una vista de alto nivel, analicemos las partes principales del diseño que forma la base. Una excelente manera de comenzar a diseñar la entrada del diario es agregar capas de abajo hacia arriba, de la misma manera que apilas el papel. La figura 10.2 muestra la estructura del diseño de la página de la revista.

1. Dado que los dispositivos móviles están disponibles en diferentes tamaños, el diseño comienza agregando un SingleChildScrollView para manejar automáticamente el desplazamiento de partes de la pantalla que están cortadas por dispositivos más pequeños.

2. A continuación, se utiliza un widget Column para alinear los widgets verticalmente desde la parte superior hacia la parte inferior de la pantalla.

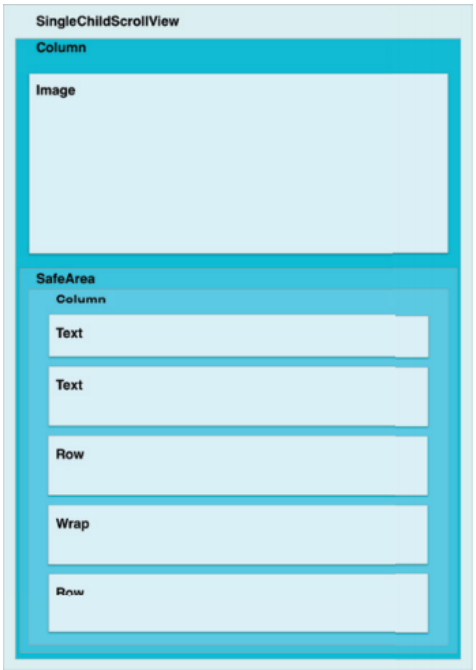


Figura 10.2: Vista de alto nivel.

3. Para la imagen de regalo envuelta, el widget Image se agrega como el primer elemento secundario de la primera columna, lo que permite que la imagen ocupe todo el ancho del dispositivo.

4. La primera columna secundaria es un widget SafeArea para manejar la muesca del dispositivo para el contenido de la entrada del diario.

5. Agrega al hijo SafeArea una segunda columna con widgets secundarios compuestos por un widget de texto para el título de la entrada del diario y un widget de texto para los detalles de la entrada del diario.

6. Continúa agregando a los hijos de la segunda columna un widget de fila que contendrá el icono del clima, la temperatura del clima y la dirección de ubicación de la entrada del diario. En la sección “Diseño de la sección meteorológica” de esta práctica, aprenderás a agregar widgets para crear el diseño detallado.

7. Continúa agregando a los hijos de la segunda columna un widget Wrap que muestre los widgets Chip. Aprenderás a agregar widgets de diseño en la sección “Diseño de etiquetas”.

8. Por último, agregarás a la segunda columna un widget de fila para mostrar imágenes e íconos, y aprenderás cómo agregar widgets de diseño en la sección “Diseño de imágenes de pie de página” de esta prácticao.

Disposición de la sección meteorológica

Cada entrada del diario registra el clima, la temperatura y la ubicación en el momento de la entrada para recordar los detalles en un momento posterior. Para proporcionar esa información, incluirás una sección sobre el clima en la entrada del diario.

Al usar una fila, agrega dos widgets de columna y un widget de SizedBox.

La primera columna contiene un icono para mostrar el símbolo del tiempo. La segunda columna contiene dos widgets de fila. La primera fila tiene un texto que muestra la temperatura y la descripción del clima. La segunda fila tiene un texto que muestra la dirección de ubicación de la entrada del diario (figura 10.3).

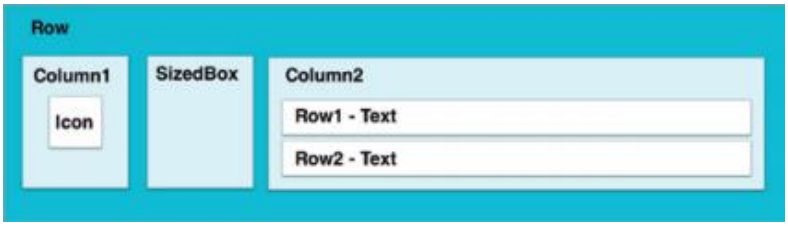


Figura 10.3: Sección meteorológica.

Diseño de etiquetas

Para organizar cada entrada del diario y facilitar la búsqueda, utiliza etiquetas para agregar categorías a la entrada.

Las etiquetas son elementos como película, familia, cumpleaños, vacaciones, etc. La sección de etiquetas usa un widget Wrap con una lista secundaria de widgets Chip. Cuando tienes una lista de elementos que pueden tener diferentes longitudes y un número desconocido de elementos, al anidarlos en un widget Wrap se distribuyen automáticamente los elementos secundarios de acuerdo con el espacio disponible (figura 10.4).

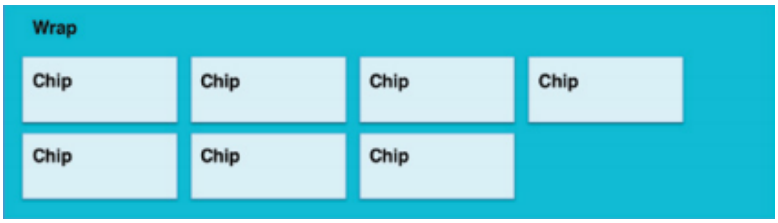


Figura 10.4: Sección de etiquetas.

El widget Chip es una excelente manera de agrupar información y personalizar la apariencia de la presentación. Establecer la propiedad de la etiqueta (label) es el único requisito, pero la mayoría de las veces se usa estableciendo la propiedad del avatar con un ícono o un widget de imagen. Por defecto, el widget Chip es una forma de estadio gris (rectángulo con grandes semicírculos en los extremos en lados opuestos) pero puedes personalizarlo usando la propiedad shape y la propiedad backgroundColor.

El siguiente código de ejemplo muestra un widget Chip personalizado que muestra la etiqueta y el avatar en forma rectangular con pequeñas esquinas redondeadas. La clase RoundedRectangleBorder devuelve el borde rectangular con esquinas redondeadas.

```
Chip(
    label: Text('Vacation'),
    avatar: Icon(Icons.local_airport),
    shape: RoundedRectangleBorder(
       borderRadius: BorderRadius.circular(4.0),
       side: BorderSide(color: Colors.grey),
    ),
    backgroundColor: Colors.grey.shade100,
);
```

Diseño de imágenes de pie de página

Se dice que una imagen vale más que mil palabras, y la sección de pie de página te permite agregar fotos a cada entrada del diario para traer recuerdos. Las secciones de pie de página usan una fila con un widget CircleAvatar que muestra diferentes imágenes. Al final de la fila, se usa un SizedBox para espaciar la columna secundaria hasta el final. La columna muestra iconos alineados verticalmente (Figura 10.5).

Disposición final (Final layout)

Observaste cómo diseñar cada sección de la página de detalles de la revista. Al anidar widgets, creas diseños personalizados o complejos conocidos como composición. El poder de los widgets anidados para crear hermosas interfaces de usuario está limitado solo por tu imaginación. La figura 10.6 muestra la página de detalles de la revista y las tres secciones principales personalizadas para el clima, las etiquetas y las imágenes de pie de página.

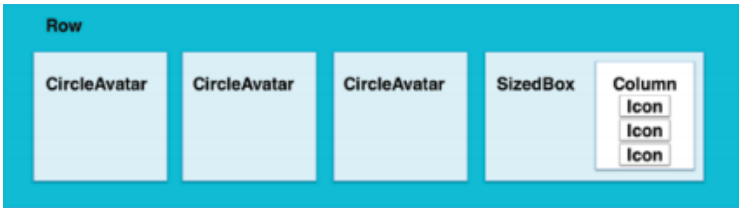


Figura 10.5: Sección de pie de página.

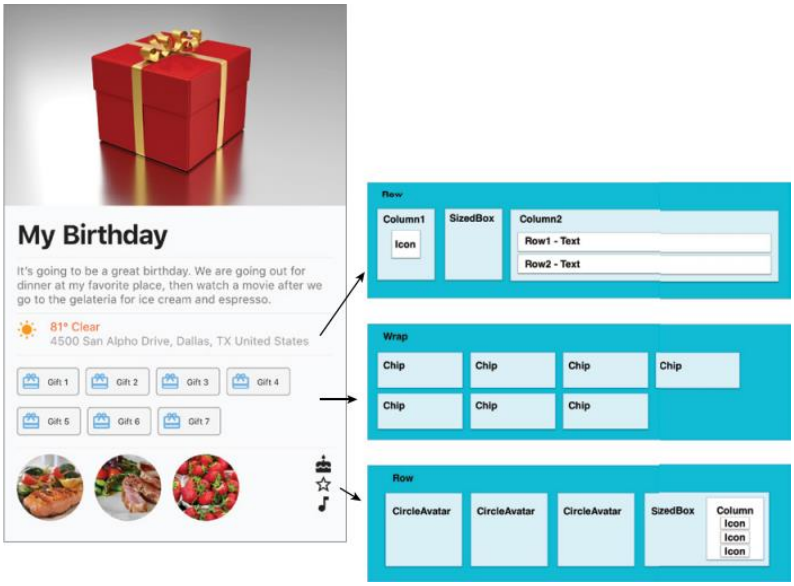


Figura 10.6: Diseño final.

Creando el diseño (Layout)

Al crear el diseño, es bueno comenzar desde una vista de alto nivel y luego avanzar hasta cada sección detallada. Al tomar cada sección de la página, comienzas a analizar los requisitos y el formato según sea necesario. Por ejemplo, si una sección en particular colocas los elementos horizontalmente, comienzas con una Fila; si el diseño de la sección es vertical, comienzas con una columna. Luego, observas los requisitos de visualización y comienzas a dividir los datos en sus propias secciones anidando widgets.

Creación de la aplicación de diseño

En nuestro ejemplo, el cuerpo principal contiene SingleChildScrollView con el elemento secundario como Columna. La lista Columna de widgets contiene una imagen de encabezado seguida de un SafeArea con Padding como hijo. La imagen se ajusta a todo el ancho del dispositivo, pero las entradas del diario están contenidas en SafeArea con un relleno para formatear la entrada.

El elemento secundario Padding es una columna con una lista de widgets que desglosan cada sección de la entrada del diario separada por un widget Divider. Cada sección separada es llamada por los métodos _buildJournalHeaderImage(), _buildJournalEntry(), _buildJournalWeather(), _buildJournalTags() y _buildJournalFooterImages().

Estás creando una página de entrada de diario que muestra los detalles de arriba a abajo. La página del diario muestra la imagen del encabezado, el título, los detalles del diario, el clima, la dirección, las etiquetas y las imágenes del pie de página. Las secciones de clima, etiquetas e imágenes de pie de página se crean anidando widgets para crear un diseño personalizado.

En este ejemplo, para mantener el árbol de widgets poco profundo, usarás métodos en lugar de clases de widgets. Este es un gran ejemplo de cómo utilizar la técnica adecuada para cada situación.

El propósito de la entrada del diario es ver los detalles y no requiere cambios, por lo que usas métodos para mantener el árbol de widgets poco profundo. Pero si esta página requiere actualizar partes de la pantalla en función de cambios de datos externos, entonces usar clases de widgets podría ser la mejor solución, porque solo se reconstruye la parte de la pantalla que cambia.

1. Crea un nuevo proyecto de Flutter y asígnale el nombre layouts. Puedes seguir las instrucciones de la práctica 4. Para este proyecto, necesitas crear solo las páginas y las carpetas de assets/images. Crea la clase de inicio como un widget sin estado, ya que los datos no requieren cambios.

2. Abre el archivo pubspec.yaml para agregar recursos. En la sección de activos, agrega la carpeta assets/images/.

```
# To add assets to your application, add an assets section, like this:
 assets:
 - assets/images/
```

3. Haz clic en el botón Save, y dependiendo del Editor que estés usando, automáticamente ejecutarás los paquetes de flutter get, y una vez terminado, mostrará un mensaje de Proceso terminado con el código de salida 0. Si no ejecuta automáticamente el comando por ti mismo, abre la ventana de Terminal (ubicada en la parte inferior de su editor) y escribe **flutter packages get**.

4. Agrega los recursos de la carpeta y las imágenes de la subcarpeta en la raíz del proyecto y luego copia los archivos present.jpg, salmon.jpg, asparagus.jpg y strawberries.jpg en la carpeta de imágenes.

Dado que este ejemplo se centra en cómo diseñar una pantalla, las imágenes se incluyen en la carpeta de assets/images, pero en una aplicación real, el usuario elegiría las imágenes y, en su lugar, se guardarían en el dispositivo.

5. Abre el archivo home.dart y agrega al cuerpo el método _buildBody(). Para este proyecto, personaliza el widget AppBar cambiando las propiedades backgroundColor, iconTheme, brightness, leading y actions, como se muestra en el siguiente código.

   Hacer los cambios de AppBar es puramente cosmético para que la aplicación se vea genial. En la práctica 6, aprendiste que el widget Icon se dibuja con un glifo de una fuente descrita en IconData. Tienes disponible una lista completa de iconos de la fuente MaterialIcons.

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(
        'Layouts',
        style: TextStyle(color: Colors.black87),
      ),
      backgroundColor: Colors.white,
      iconTheme: IconThemeData(color: Colors.black54),
      brightness: Brightness.light,
      leading: IconButton(icon: Icon(Icons.menu), onPressed: () {}),
      actions: <Widget>[
        IconButton(icon: Icon(Icons.cloud_queue), onPressed: () {})
      ],
    ),
    body: _buildBody(),
  );
}
```

6. Agrega el método del widget _buildBody() después de la compilación del widget (BuildContext context) {...}.

7. Devolver SingleChildScrollView con el elemento secundario como Column. La lista de widgets de Column child está llamando a todos los métodos para crear cada sección de la página. Ten en cuenta que el primer método, _buildJournalHeaderImage(), se coloca antes de SafeArea y Padding, lo que permite que la imagen ocupe todo el ancho del dispositivo.

8. Agrega una columna como elemento secundario del Padding y luego llama a los métodos _buildJournalEntry(), _buildJournalWeather(), _buildJournalTags() y _buildJournalFooterImages().


```
Widget _buildBody() {
  return SingleChildScrollView(
    child: Column(
      children: <Widget>[
        _buildJournalHeaderImage(),
        SafeArea(
          child: Padding(
            padding: EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                _buildJournalEntry(),
                Divider(),
                _buildJournalWeather(),
                Divider(),
                _buildJournalTags(),
                Divider(),
                _buildJournalFooterImages(),
              ],
            ),
          ),
        ),
      ],
    ),
  );
}
```

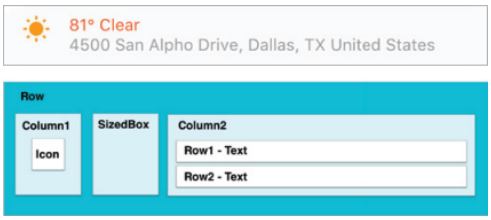
9. Crea el método _buildJournalHeaderImage(), que devuelve una imagen. La propiedad de la imagen usa AssetImage present.jpg con ajuste establecido en BoxFit.cover, lo que permite que la imagen ocupe todo el ancho del dispositivo.

```
Image _buildJournalHeaderImage() {
  return Image(
    image: AssetImage('assets/images/present.jpg'),
    fit: BoxFit.cover,
  );
}
```

10. Crea el método _buildJournalEntry(), que devuelve una columna. La lista de widgets secundarios de Columna contiene dos widgets de Texto y un widget Divider().

```
Column _buildJournalEntry() {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      Text(
        'My Birthday',
        style: TextStyle(
          fontSize: 32.0,
          fontWeight: FontWeight.bold,
        ),
      ),
      Divider(),
      Text(
        'It's going to be a great birthday. We are going out for dinner at my
favorite place, then watch a movie after we go to the gelateria for ice cream and
espresso.',
        style: TextStyle(color: Colors.black54),
      ),
    ],
  );
}
```

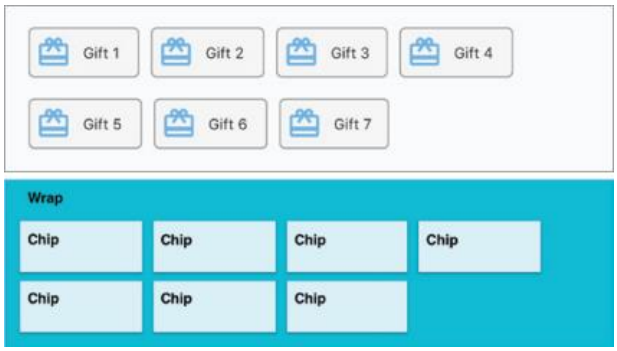
11. Crea el método _buildJournalWeather(), que devuelve una fila. La lista de widgets secundarios de Row contiene una Column, un SizedBox y otra Column. La lista de widgets de la segunda columna contiene dos widgets de fila.




```
Row _buildJournalWeather() {
  return Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Icon(
              Icons.wb_sunny,
              color: Colors.orange,
            ),
          ],
        ),
        SizedBox(width: 16.0,),
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Row(
              children: <Widget>[
                Text(
                  '81° Clear',
                  style: TextStyle(color: Colors.deepOrange),
                ),
              ],
            ),
            Row(
              children: <Widget>[
                Text(
                  '4500 San Alpho Drive, Dallas, TX United States',
                  style: TextStyle(color: Colors.grey),
                ),
              ],
            ),
          ],
        ),
      ],
    );
}
```

12. Crea el método _buildJournalTags(), que devuelve un Wrap. Los hijos Wrap usan el constructor List.generate para construir la lista de datos de muestra para mostrar siete valores de etiqueta de muestra.

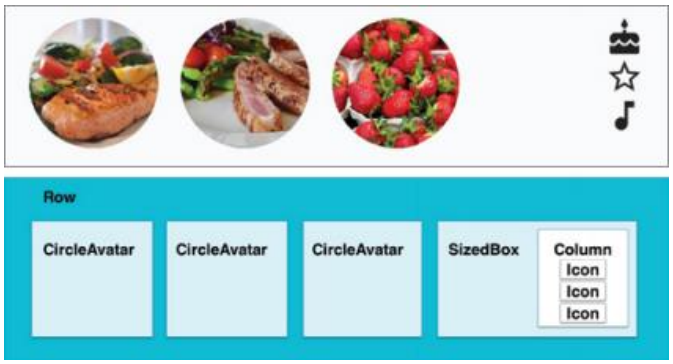
    El constructor toma dos argumentos, la longitud de la lista y el índice.




```
Wrap _buildJournalTags() {
  return Wrap(
    spacing: 8.0,
    children: List.generate(7, (int index) {
      return Chip(
        label: Text(
          'Gift ${index + 1}',
          style: TextStyle(fontSize: 10.0),
        ),
        avatar: Icon(
          Icons.card_giftcard,
          color: Colors.blue.shade300,
        ),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(4.0),
          side: BorderSide(color: Colors.grey),
        ),
        backgroundColor: Colors.grey.shade100,
      );
    }),
  );
}
```

13. Crea el método _buildJournalFooterImages(), que devuelve una fila. La lista de widgets secundarios de Row contiene tres CircleAvatars y un SizedBox.

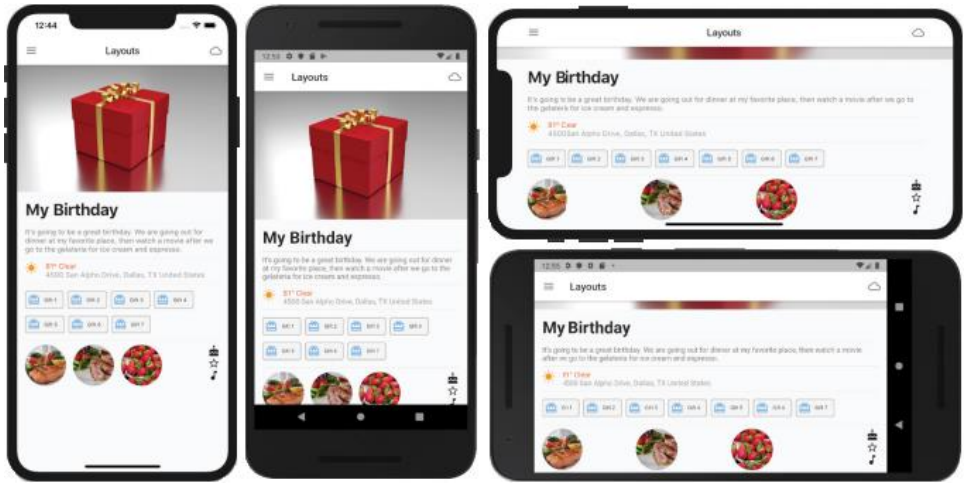
    El elemento secundario SizedBox es una columna con una lista secundaria de widget de tres iconos. El propósito principal de SizedBox es agregar espacio adicional entre el CircleAvatar y los íconos colocados verticalmente.




```
Row _buildJournalFooterImages() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      CircleAvatar(
        backgroundImage: AssetImage('assets/images/salmon.jpg'),
        radius: 40.0,
      ),
      CircleAvatar(
        backgroundImage: AssetImage('assets/images/asparagus.jpg'),
        radius: 40.0,
      ),
      CircleAvatar(
        backgroundImage: AssetImage('assets/images/strawberries.jpg'),
        radius: 40.0,
      ),
      SizedBox(
        width: 100.0,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.end,
          children: <Widget>[
            Icon(Icons.cake),
            Icon(Icons.star_border),
            Icon(Icons.music_note),
            //Icon(Icons.movie),
          ],
        ),
      ),
    ],
  );
}
```

Primero creaste una vista de alto nivel desglosando las secciones principales de la página. En la base, usaste SingleChildScrollView y construiste consecutivamente en la parte superior una Column, Image de encabezado, SafeArea, Padding y una Column con una lista secundaria de Widget que construyen cada sección por separado.

Los dividiste en cuatro secciones separadas y las creaste llamando a los métodos _buildJournalEntry(), _buildJournalWeather(), _buildJournalTags() y _buildJournalFooterImages(). Cada uno de estos métodos crea un diseño personalizado anidando widgets.



Resumen

En esta práctica, aprendiste cómo imaginar un diseño personalizado de alto nivel y dividirlo en sus secciones principales. Luego tomaste cada sección principal y construiste el diseño necesario anidando widgets.

En la próxima práctica, aprenderás a agregar interactividad mediante los widgets GestureDetector, Draggable, DragTarget, InkWell y Dismissable.