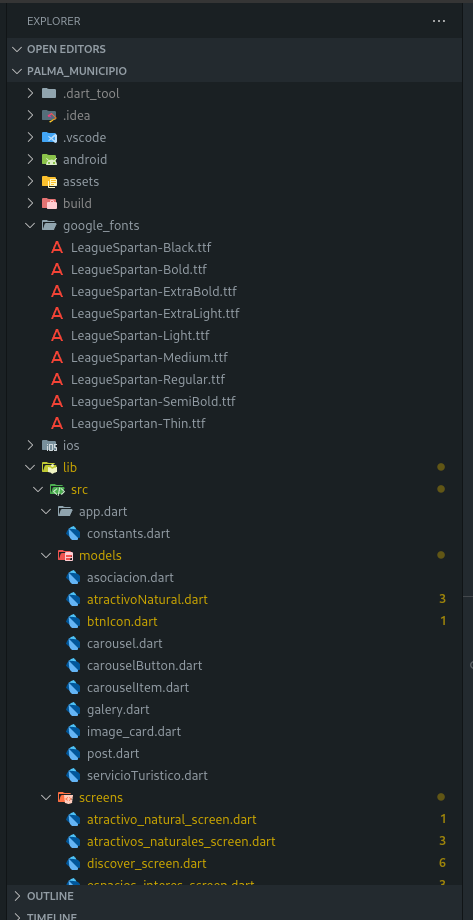
**Documentación**

**LA PALMA APP**

Estructura de carpetas

En el siguiente apartado se va a describir la estructura de carpetas de la aplicación Flutter que se muestra en la imagen para asegurarnos de que quede documentado.



### Estructura de Carpetas

**1. Carpetas Generales del Proyecto**

- .dart\_tool, .idea, .vscode, android, ios, build:

Estas carpetas son parte del entorno de Flutter y Android Studio:

- .dart\_tool: Contiene herramientas del lenguaje Dart necesarias para el desarrollo del proyecto. Incluye cachés y otros archivos generados.

- .idea: Configuraciones específicas del proyecto para el IDE (IntelliJ o Android Studio).

- .vscode: Configuraciones específicas para Visual Studio Code.

- android y ios: Estas carpetas contienen configuraciones específicas para las plataformas Android e iOS, respectivamente. Se utilizan para ajustar configuraciones nativas de la aplicación.

- build: Generada automáticamente cuando se compila el proyecto. Contiene artefactos y versiones construidas de la aplicación.

**2. assets y google\_fonts**

- google\_fonts:

Aquí se han colocado archivos de fuentes personalizadas. En este caso, todos son estilos de la fuente LeagueSpartan, cada uno con un grosor distinto (Bold, Regular, Light, etc.). Esto permite que la aplicación use esta tipografía para garantizar consistencia visual en toda la interfaz.

- assets:

Esta carpeta normalmente se utiliza para imágenes, íconos, archivos JSON, y otros recursos. Aunque en la imagen no se muestran los contenidos, aquí se almacenan todos los archivos que no son código pero que se usan en la UI (como imágenes de fondo, íconos personalizados, etc.).

**3. lib (librerías de la app)**

La carpeta lib es el núcleo del desarrollo de Flutter y contiene todo el código fuente de la aplicación. Dentro de lib, vemos la subcarpeta src que, a su vez, tiene una estructura lógica para la organización del código:

**- src:**

La carpeta src se ha utilizado para mantener bien estructurado el proyecto, dividiendo el código en módulos específicos.

**- app.dart:**

Este archivo parece ser el punto de entrada o el archivo principal de configuración de la app. Podría incluir configuraciones generales como temas, rutas, etc.

**- constants.dart:**

Este archivo se usa para definir constantes que son utilizadas en toda la aplicación. Este enfoque ayuda a mantener el código limpio y facilita la actualización de valores, como colores, padding, o URLs, en un solo lugar.

**- models:**

Esta carpeta contiene las clases que definen la estructura de datos usada en la aplicación. Por ejemplo:

- asociacion.dart, atractivoNatural.dart, btnIcon.dart, galery.dart, etc.

Cada uno de estos archivos contiene modelos que probablemente representan diferentes entidades dentro de la aplicación. Documentemos algunos ejemplos:

- atractivoNatural.dart: Probablemente contiene la definición del modelo para un atractivo natural, incluyendo atributos como title, image, description, etc.

- btnIcon.dart: Podría definir un widget reutilizable para representar un botón con ícono.

**- screens:**

Esta carpeta organiza las diferentes pantallas que componen la aplicación. Cada archivo representa una pantalla distinta:

- atractivo\_natural\_screen.dart, atractivos\_naturales\_screen.dart, discover\_screen.dart: Cada archivo define una pantalla específica de la aplicación, por ejemplo:

- atractivo\_natural\_screen.dart: pantalla atractivo natural.

- discover\_screen.dart: pantalla descubrenos.

* **utils**: Esta carpeta contiene utilidades generales que son usadas a lo largo de la aplicación. utils.dart podría tener funciones auxiliares comunes, en este caso estamos almacenando ahi toda la informacion acerca de los atractivos nataurales,atractivos religiosos,semana santa etc, flora, fauna, gastronomia.
* **widgets**: La carpeta **widgets** contiene componentes reutilizables que se utilizan en diferentes pantallas de la aplicación. Algunos ejemplos son:
  + carousel\_component.dart, custom\_button.dart, image\_btn.dart, etc.: Estos widgets ayudan a mantener el código modular y reutilizable.
  + **Ejemplos**:
    - carousel\_component.dart: Podría ser un widget reutilizable para mostrar un carrusel de imágenes o elementos.
    - custom\_button.dart: Podría definir un botón personalizado que se usa en varias pantallas.

Screens:

IndexScreen:

### Explicación del Código

#### Estructura General

El código define la clase IndexScreen, que extiende StatelessWidget. La pantalla principal incluye un logo, un título y una lista de botones principales que se utilizan para navegar a diferentes pantallas de la aplicación.

Este código define la pantalla de índice de la aplicación, que actúa como una especie de "menú principal" para acceder a las diferentes secciones de la aplicación. Utiliza varios widgets personalizados, como CustomButton e ImageBtn, que ayudan a garantizar consistencia en el estilo y facilitan la reutilización del código.

El uso de widgets personalizados, organizados en la carpeta widgets, ayuda a mantener la estructura de la aplicación limpia y modular. Esto no solo facilita la lectura y el mantenimiento del código, sino que también promueve una experiencia de usuario consistente en toda la aplicación.

* **Imports**: Se importan diferentes pantallas y widgets que se utilizarán dentro de la clase IndexScreen, como CustomButton y ImageBtn. Estos widgets personalizados se encuentran en la carpeta widgets, según las rutas proporcionadas en las importaciones.

#### Scaffold y SafeArea

* **Scaffold**: Contiene la estructura básica de la interfaz, que incluye el AppBar, el Body y otros elementos comunes en una pantalla de Flutter.
* **SafeArea**: Garantiza que el contenido esté dentro de las áreas seguras de la pantalla (evitando superposición con la barra de notificación, notch, etc.).

#### Logo y Título

* **Logo**: Se utiliza un Container para contener una imagen del logo, que se carga desde los activos (assets/images/logo.png).
* **Título**: El título se extrae de AppTexts.titleDiscover y se presenta con un Text widget, que tiene un estilo definido para darle énfasis.

#### Lista de Botones Principales

* **Expanded**: Envuelve la lista de botones para asegurar que se ajuste al espacio disponible y permita el desplazamiento.
* **ListView**: Se utiliza para mostrar los botones principales, permitiendo que se pueda desplazar la lista si hay muchos elementos.
* **CustomButton Widget**:
  + Cada botón es un CustomButton, que es un widget personalizado que permite crear botones con estilo uniforme en toda la aplicación. Los CustomButton se encuentran en la carpeta widgets y ayudan a mantener la consistencia visual.
  + Los CustomButton reciben varios parámetros:
    - **text**: Define el texto del botón (extraído de AppTexts para una gestión centralizada de textos).
    - **onPressed**: Define la función a ejecutar cuando se presiona el botón. En este caso, se utiliza la función getNavigator para navegar a diferentes pantallas de la aplicación.
    - **leadingImage**: Utiliza el widget ImageBtn para agregar una imagen junto al texto del botón. ImageBtn también es un widget personalizado definido en la carpeta widgets y facilita la adición de íconos.

#### Navegación

* La navegación entre pantallas se realiza a través de la función getNavigator, que se utiliza para pasar al contexto de una nueva pantalla de forma segura.
* Las pantallas de destino incluyen DiscoverScreen, AtractivosNaturales, TurismoReligiosoScreen, entre otras. Cada botón permite al usuario acceder a una sección específica de la aplicación.

#### Espaciado

* **SizedBox**: Se utiliza SizedBox entre cada CustomButton para agregar un espaciado consistente (16.0 unidades de altura) entre cada botón, mejorando la apariencia visual y la legibilidad.

Discover\_screen

La clase DiscoverScreen es una pantalla clave en la aplicación que presenta información sobre el municipio de La Palma, incluyendo detalles culturales, naturales y turísticos. Contiene una serie de componentes visuales como carruseles de imágenes, botones de navegación a otras pantallas, e información relevante sobre la región.

#### Estructura del Código

* **Imports**: La pantalla comienza con varios import, que traen funcionalidades de Flutter, modelos, pantallas adicionales, y widgets reutilizables. Algunos widgets importantes son:
  + CarouselComponent, ImageCard, CardContact: Todos estos se encuentran en la carpeta /widgets y son componentes reutilizables que ayudan a estructurar la interfaz.
  + CustomButton y ImageBtn: Widgets personalizados que se utilizan para crear botones con imágenes y acciones específicas.
* **Lista de Botones de Carrusel (categories)**: Se define una lista categories que contiene objetos de tipo Carouselbutton. Cada uno de estos botones representa una categoría distinta que se puede explorar (e.g., atractivosNaturales, turismoReligioso). Estos objetos contienen:
  + button\_text: El texto que aparecerá en el botón.
  + method: La acción que se ejecutará al presionar el botón (en este caso, la navegación a otra pantalla).
  + carousel: Una lista de elementos de carrusel (CarouselItem) que muestran imágenes relacionadas con la categoría.
* **Clase DiscoverScreen**: Es una subclase de StatelessWidget, lo que significa que no tiene estado que cambie durante su vida.
  + **build(BuildContext context)**: Este método construye la interfaz de usuario de la pantalla.
    - **Scaffold**: Proporciona la estructura básica de la pantalla, incluyendo el body.
    - **SingleChildScrollView**: Permite que el contenido de la pantalla sea desplazable, ideal para pantallas largas como esta.
    - **CarouselComponent**: Se utiliza para mostrar un carrusel de imágenes destacado al inicio de la pantalla, con títulos y subtítulos que representan lugares en La Palma.
    - **Padding y Column**: Los widgets Padding y Column se utilizan para organizar el contenido de la pantalla, dando espacio adecuado entre los elementos y alineándolos de manera ordenada.
* **Contenido del children**: Este arreglo contiene los widgets que se muestran en la pantalla principal.
  + **Text**: Varias instancias del widget Text se usan para mostrar el título y las descripciones de La Palma. Se utiliza la fuente LeagueSpartan para mantener la consistencia visual.
  + **StyledImage y MapImageWithButton**: Son widgets personalizados para mostrar imágenes y mapas interactivos.
  + **getCategories(context)**: Este método devuelve un Column que contiene todos los botones de carrusel definidos en categories. Cada botón permite al usuario navegar a una sección específica (e.g., Turismo Religioso, Oferta Gastronómica).
  + **getImageCards()**: Este método devuelve una lista de widgets ImageCard, cada uno representando una tarjeta con información sobre un punto de interés. Cada tarjeta incluye un título, subtítulo e imagen, y permite navegar a una pantalla detallada al ser presionada.
  + **CardContact**: Se utiliza para mostrar información de contacto sobre instituciones relevantes como la Gobernación de Cundinamarca y la Alcaldía Municipal de La Palma.

#### Widgets Personalizados Utilizados

* **CustomButton**: Este widget se utiliza para crear botones con un diseño personalizado que incluyen imágenes (íconos) a la izquierda del texto. Esto facilita la navegación a diferentes secciones de la aplicación de una manera visualmente atractiva y cómoda para el usuario.
* **ImageBtn**: Un widget que muestra una imagen y se utiliza como líder en los CustomButton. Proporciona una representación visual de cada sección, ayudando al usuario a identificarlas rápidamente.
* **CarouselComponent**: Un widget reutilizable que presenta un carrusel de imágenes con opciones de autoplay. Se usa para resaltar diferentes lugares en La Palma.
* **StyledImage y MapImageWithButton**: Son widgets que permiten mostrar imágenes con un estilo consistente y mapas interactivos, respectivamente. MapImageWithButton incluye la capacidad de navegar al mapa en Google Maps.
* **CardContact**: Este widget está diseñado para mostrar información de contacto de manera estructurada, incluyendo el título, la dirección, el horario de atención y más.

AtractivosNaturales

La pantalla **AtractivosNaturales**, que es una pantalla de la aplicación Flutter destinada a mostrar los atractivos naturales de La Palma. La estructura del código se compone de varios widgets y lógica que permiten una experiencia de usuario rica e interactiva.

### Estructura General del Código

Esta pantalla utiliza el widget **StatelessWidget** de Flutter para representar la interfaz de usuario. A continuación, se describe el código de la pantalla **AtractivosNaturales** para entender cómo se construye la estructura de la aplicación, cómo se organiza la información, y cómo se utilizan otros widgets personalizados.

### Importaciones

* flutter/material.dart: Importa la biblioteca de Flutter para diseñar la interfaz de usuario.
* google\_fonts/google\_fonts.dart: Utilizada para implementar tipografías personalizadas en la aplicación, en este caso **League Spartan**.
* Varias importaciones como constants.dart, carouselItem.dart, image\_card.dart, etc., que son archivos internos del proyecto y permiten estructurar la aplicación de una manera modular, reutilizando componentes y modelos.

### Definición de AtractivosNaturales

Esta pantalla muestra información sobre los atractivos naturales de La Palma, incluyendo descripciones, imágenes y carrousels de fotos que representan los diferentes paisajes y elementos naturales.

class AtractivosNaturales extends StatelessWidget {

const AtractivosNaturales({super.key});

@override

Widget build(BuildContext context) {

...

return Scaffold(

body: SingleChildScrollView(

child: Column(

children: [

CarouselComponent(

items: carouselItems,

height: 200.0,

autoPlay: true,

),

Padding(

padding: EdgeInsets.all(16.0),

child: Column(

children: children,

),

),

],

),

),

);

}

}

* Scaffold: Proporciona la estructura básica de la interfaz de usuario con una barra de aplicación y un contenedor principal para el contenido.
* SingleChildScrollView: Permite desplazarse por el contenido cuando éste supera el tamaño de la pantalla.
* CarouselComponent: Un componente personalizado que se utiliza para crear un carrusel de imágenes. En este caso, se pasa una lista de elementos (carouselItems) y se configura la altura y la opción de auto-reproducción.

### Uso de Componentes Personalizados

Esta pantalla utiliza una variedad de widgets personalizados y modelos que se encuentran en la carpeta **widgets** del proyecto, ayudando a crear un código modular y reutilizable. Algunos de estos widgets incluyen:

1. **CarouselComponent**
   * Utilizado para mostrar un carrusel de imágenes. En este caso, se utilizan tres carrousels diferentes para mostrar los atractivos, la fauna y la flora de la región. Cada **CarouselItem** tiene una imagen y puede tener un título y subtítulo opcionales.
2. **StyledImage**
   * Widget personalizado que se usa para mostrar una imagen estilizada con un borde redondeado. En este caso, se utiliza para mostrar una de las imágenes de los atractivos naturales con un borde de 16px de radio.
3. **TitleDivider**
   * Widget utilizado para dividir secciones con un título. En este archivo, se utiliza para dividir las distintas secciones de la pantalla (“La Palma”, “Fauna”, “Flora”, etc.).
4. **LinkImageCard**
   * Widget para mostrar tarjetas de imágenes con un enlace asociado. Estas tarjetas llevan al usuario a contenido relacionado con realidad aumentada.
5. **ImageCard**
   * Se utiliza para mostrar tarjetas con títulos, subtítulos e imágenes. Estas tarjetas permiten a los usuarios acceder a más detalles sobre los atractivos naturales al navegar a una pantalla específica para cada uno.

### Descripción de la Fauna y Flora

La pantalla incluye textos descriptivos que detallan las riquezas naturales de La Palma:

* **Descripciones Textuales**:
  + description1 y description2: Describen los paisajes y actividades disponibles en La Palma.
  + faunaDescription y floraDescription: Proveen detalles específicos sobre la fauna y flora locales, enriqueciendo la experiencia de los visitantes.
* Estas descripciones están representadas mediante **Text Widgets** estilizados con la fuente **League Spartan**, lo cual ayuda a mantener la consistencia visual en toda la aplicación.

### Uso de getImageCards()

El método **getImageCards** devuelve una lista de widgets **ImageCard**, cada uno representando un atractivo natural diferente. Utiliza el widget **ImageCard** que incluye un título, subtítulo e imagen, y una acción para navegar a la pantalla detallada de ese atractivo natural.

List<Widget> getImageCards() {

List<ImageCardModel> cardData = [

ImageCardModel(

title: "PUENTE DE ORO",

subtitle: "",

image: "assets/images/Puente-de-oro-.jpg",

method: (context) => Navigator.push(

context,

MaterialPageRoute(

builder: (context) => AtractivoNaturalScreen(

atractivonatural: getDataAtractivoNatural("puete-oro"))))),

// ... otros ImageCards

];

return cardData.map((data) {

return Padding(

padding: const EdgeInsets.symmetric(vertical: 8.0),

child: ImageCard(

title: data.title!,

subtitle: data.subtitle!,

image: data.image!,

onTap: data.method,

),

);

}).toList();

}

* **ImageCardModel**: Cada instancia representa un atractivo con un título, imagen, y un método que permite navegar a una pantalla con más información.
* **Navigator.push**: Se utiliza para navegar hacia la pantalla detallada de cada atractivo natural, pasando la información correspondiente.

AtractivoNatural

La pantalla **AtractivoNaturalScreen** presenta una interfaz interactiva para explorar detalles específicos de los atractivos naturales en **La Palma**. Los elementos visuales y las descripciones detalladas permiten a los usuarios conectar con la riqueza natural del lugar. Además, la organización modular mediante widgets reutilizables mejora la mantenibilidad del código y facilita futuras modificaciones o expansiones del proyecto.

A continuacion explicaremos la estructura del codigo:

#### Importaciones

El código comienza importando paquetes y componentes que se utilizarán para definir la pantalla:

* **Material.dart**: Biblioteca básica de Flutter que proporciona widgets y temas.
* **Google Fonts**: Se usa para personalizar las fuentes, en este caso, usando **League Spartan**.
* **Módulos internos**: Se incluyen diferentes modelos, widgets y constantes que forman parte de la aplicación, tales como:
  + **Widgets**: carousel\_component, image\_component, link\_image\_card, openMapGoogle, recomendaciones. Estos se encuentran en la carpeta de widgets y son componentes reutilizables que agregan modularidad al código.
  + **Modelos**: atractivoNatural y carouselItem son utilizados para estructurar los datos y los elementos visuales que se presentan.

#### Clase AtractivoNaturalScreen

La clase **AtractivoNaturalScreen** extiende **StatelessWidget** y recibe un objeto de tipo **Atractivonatural** como parámetro, lo cual permite mostrar información dinámica basada en los datos del atractivo natural.

class AtractivoNaturalScreen extends StatelessWidget {

final Atractivonatural atractivonatural;

const AtractivoNaturalScreen({super.key, required this.atractivonatural});

Este patrón asegura que cada instancia de la pantalla muestre un atractivo natural específico al usuario.

#### Método build

El método build define el código que renderiza la interfaz de usuario. El diseño se construye utilizando una combinación de Scaffold, SingleChildScrollView y Column para organizar los elementos de la pantalla.

* **Scaffold**: Es el contenedor principal de la pantalla.
* **SingleChildScrollView**: Permite hacer que la pantalla sea desplazable si su contenido excede la altura visible.
* **Column**: Organiza el contenido verticalmente.

#### Definición del Contenido (Variable children)

La variable children contiene una lista de widgets que forman el contenido de la pantalla. Se utilizan varios widgets para construir la interfaz, como se muestra a continuación:

1. **SizedBox**: Se usa para agregar espaciado entre los elementos.
2. **Text**: Utilizado para mostrar el título y la descripción del atractivo natural.
3. **Botones de Acción**: atractivonatural.btns despliega una lista de botones definidos en el objeto **Atractivonatural**.
4. **Galerías y Carruseles**: Se muestra una galeria de imágenes usando el widget CarouselComponent para ofrecer una mejor experiencia visual al usuario. Se utiliza el método getCarousel para construir cada carrusel de imágenes.
5. **Mapa y Ubicación**: Se usa el widget MapImageWithButton para mostrar una imagen del mapa y un botón que lleva al usuario a la ubicación en Google Maps.
6. **Recomendaciones**: Se presenta una lista de recomendaciones relevantes con el widget Recomendaciones, si existen recomendaciones disponibles en el objeto **Atractivonatural**.

#### Métodos Auxiliares

* **getCarousel**: Este método crea un widget CarouselComponent a partir de una lista de imágenes (tipo String).
  + Se recorre la lista de imágenes y se convierte cada una en un CarouselItem.
  + Se configura el carrusel con un radio de borde (border\_radius\_) y una altura (height).

CarouselComponent getCarousel(List<String> galery, double border\_radius\_) {

List<CarouselItem> items = [];

for (var el in galery) {

items.add(CarouselItem(image: el));

}

return CarouselComponent(

items: items,

border\_radius: border\_radius\_,

height: 200,

);

}

#### Widget drawEstatua

Este método define una sección especial que solo se muestra si el título del atractivo es "Capilla de Santa Bárbara".

El método drawEstatua construye una columna que muestra una descripción sobre la estatua de **Fray Juan Martín de la Palma**, así como una imagen y un enlace para ver la estatua en realidad aumentada.

Widget drawEstatua() {

return Column(

children: [

Text(

'Estatua de Fray Juan Martín de la Palma',

style: GoogleFonts.leagueSpartan(

fontSize: 25,

fontWeight: FontWeight.w700,

fontStyle: FontStyle.normal,

color: AppColors.primaryColor,

),

),

Text(

description1,

style: TextStyle(fontSize: 16),

),

LinkImageCard(

imageUrl: "assets/images/icons/vr.jpg",

label: "Realidad Aumentada",

link:

"https://www.lookiar.com/models/046aca96462638dc19c4bf8531f18c2a/view",

),

StyledImage(

imageUrl:

'assets/images/Estatua-Fray-Martin-La-Palma.jpg',

borderRadius: 16.0,

),

],

);

}

#### Explicación de los Widgets Utilizados

* **CarouselComponent**: Este widget se utiliza para crear carruseles de imágenes de los diferentes atractivos. Permite mostrar múltiples imágenes de forma atractiva y facilita la navegación entre ellas.
* **MapImageWithButton**: Este widget se usa para mostrar una imagen con un botón que permite abrir un enlace a Google Maps. Es ideal para mostrar ubicaciones geográficas y facilitar el acceso al mapa del atractivo natural.
* **LinkImageCard**: Widget utilizado para mostrar una tarjeta con una imagen y un enlace, en este caso, para acceder a experiencias en realidad aumentada. Mejora la experiencia del usuario proporcionando información interactiva y enriquecida.
* **StyledImage**: Muestra imágenes con un estilo específico, en este caso con bordes redondeados, para hacer que la experiencia visual sea más atractiva.
* **Recomendaciones**: Muestra una lista de recomendaciones sobre cómo visitar y comportarse en el atractivo natural, ayudando a educar a los visitantes sobre cómo disfrutar el lugar de manera responsable.

EspaciosInteres

La clase EspaciosInteresScreen representa una pantalla que presenta varios espacios de interés dentro del municipio de La Palma. Utiliza el paquete de Flutter Material para construir la interfaz de usuario, junto con varios componentes personalizados para enriquecer la experiencia del usuario.

La pantalla EspaciosInteresScreen está diseñada para ofrecer una experiencia visual rica e informativa, presentando los espacios más importantes del municipio de La Palma. Los diferentes widgets personalizados permiten al usuario explorar la información de una manera visualmente atractiva, fomentando la interacción y el conocimiento sobre la comunidad. Cada sección proporciona detalles importantes para que los visitantes puedan disfrutar de lo mejor que ofrece el municipio.

#### Descripción de los Componentes y Funcionalidad Principal

* **Atributos**:
  + La pantalla utiliza varios carruseles (carouselItems) para mostrar imágenes de diferentes lugares de interés, incluyendo parques, la plaza de mercado, museos, artesanías y la biblioteca municipal.
  + Cada carrusel está compuesto por instancias de la clase CarouselItem, que contiene la imagen, título y subtítulo del espacio correspondiente.
* **Métodos y Widgets**:
  + build(BuildContext context): Este es el método principal que construye la interfaz de usuario para la pantalla.
    - Utiliza un SingleChildScrollView para permitir que el contenido sea desplazable.
    - El contenido principal se construye usando un Column que organiza todos los elementos visuales de la pantalla.
    - Se utilizan múltiples componentes personalizados para mostrar galerías de imágenes, texto descriptivo y botones interactivos.
* **Widgets Personalizados Utilizados**:
  + **CarouselComponent**: Utilizado para mostrar galerías de imágenes de los diferentes espacios de interés. Este componente mejora la experiencia del usuario al presentar las imágenes de manera atractiva y dinámica.
  + **CustomButton**: Botón personalizado utilizado para enlazar con la pantalla de eventos. Contiene una imagen como icono que acompaña el texto del botón.
  + **TitleDivider**: Este widget muestra un título con un estilo dividido que ayuda a organizar visualmente el contenido en secciones.
  + **StyledImage**: Utilizado para mostrar imágenes con bordes redondeados, mejorando la presentación visual.

#### Descripción Detallada de la Pantalla

1. **Ferias y Fiestas**:
   * Se muestra un título estilizado con GoogleFonts.leagueSpartan.
   * Incluye descripciones detalladas de las festividades de La Palma, junto con una imagen representativa y un botón que enlaza a la pantalla de eventos del municipio.
2. **Parques**:
   * Seccionado en subespacios como **Parque Diana Turbay** y **Parques Deportivos**.
   * Cada sección incluye un carrusel de imágenes y una descripción sobre los parques, su historia y utilidad para la comunidad.
   * Incluye información adicional, como coordenadas geográficas y ubicaciones específicas.
3. **Plaza de Mercado**:
   * Descripción detallada sobre la importancia comercial y cultural de la plaza de mercado de La Palma.
   * Carrusel con imágenes de la plaza para dar una idea visual de su atmósfera vibrante.
4. **Museo**:
   * Contiene información sobre el Museo de la Virgen en La Palma, con una descripción detallada y un carrusel de imágenes.
   * Se proporciona información adicional, como dirección y teléfono.
5. **Artesanías**:
   * Se muestra la información sobre las artesanías de La Palma, resaltando el trabajo local y el uso de materiales naturales como el amero.
   * Un contenedor estilizado resalta las artesanías de la Sra. Yolanda, proporcionando información de contacto y una galeria de imágenes.
6. **Biblioteca Municipal**:
   * Describe el papel de la biblioteca municipal como centro cultural y educativo.
   * Incluye un carrusel de imágenes de la biblioteca y detalles como ubicación y coordenadas.

RedServicios

La clase RedServiciosScreen es una pantalla que muestra la información sobre los servicios disponibles en La Palma. La pantalla utiliza diferentes componentes visuales para mostrar detalles sobre transporte, asistencia pública, entidades financieras y estaciones de servicio. Cada sección está acompañada de carruseles de imágenes que ilustran los puntos de interés y sus ubicaciones.

La pantalla RedServiciosScreen está diseñada para proporcionar información detallada sobre la red de servicios disponibles en La Palma. Utiliza carruseles de imágenes, contenedores informativos, y divisores de título para organizar la información de manera clara y atractiva. Esto permite al usuario explorar servicios financieros, opciones de transporte, entidades de asistencia pública, y estaciones de servicio de una manera ágil y visualmente agradable.

Cada sección está bien delimitada y contiene la información necesaria, complementada con imágenes ilustrativas para mejorar la experiencia del usuario, facilitando la navegación y la comprensión de los servicios ofrecidos en el municipio.

#### Descripción de los Componentes y Funcionalidad Principal

* **Atributos**:
  + **carouselItems**: Lista principal de elementos del carrusel que representa los servicios generales de la red de servicios en La Palma.
  + **carouselAgrario, carouselCorresponsal, carouselTexaco, carouselBiomax, etc.**: Diferentes listas de elementos del carrusel que representan fotos específicas de entidades y lugares importantes.
  + **children**: Una lista de widgets que componen la estructura de la pantalla, incluyendo textos y contenedores para cada sección.
* **Métodos y Widgets**:
  + **build(BuildContext context)**: Es el método principal que construye la interfaz de usuario de la pantalla. Utiliza un SingleChildScrollView para permitir que el contenido se desplace y un Column para organizar los diferentes elementos visuales.
    - **Texto del Título**: Utiliza GoogleFonts.leagueSpartan para estilizar el título principal de la pantalla: "RED DE SERVICIOS".
    - **TitleDivider**: Widget que se usa para separar las diferentes secciones de la pantalla con un título y una línea divisoria para mejorar la navegación visual.
    - **Secciones con Contenedores**: Cada sección de información está contenida dentro de un Container estilizado con bordes redondeados, sombra, y relleno. Esto incluye información detallada de cómo llegar a La Palma desde diferentes ubicaciones y detalles de servicios locales como el Banco Agrario, estaciones de gasolina, y la policía.
    - **getListStylized**: Este método genera una lista de widgets estilizados que se muestran como filas, cada una con un ícono de flecha y un texto asociado. Es utilizado para mostrar información detallada en listas, como rutas de acceso y detalles de contacto.
* **Componentes Personalizados Utilizados**:
  + **CarouselComponent**: Widget que muestra una galeria de imágenes en un carrusel. Se utiliza para mostrar imágenes de varios servicios, como estaciones de servicio, entidades financieras, y transporte.
  + **TitleDivider**: Widget que muestra un título de sección con una línea divisoria para una mejor organización visual del contenido.
  + **Iconos y Textos en Contenedores**: Cada contenedor de información tiene filas que incluyen íconos (Icons.location\_on, Icons.phone\_android, Icons.bus\_alert) junto a texto explicativo, lo cual mejora la claridad de la información proporcionada al usuario.

#### 