UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE INGENIERÍA

ÁREA DE CIENCIAS DE LA COMPUTACIÓN

FUNDAMENTOS DE DESARROLLO MÓVIL

MANUAL DEL PROGRAMADOR

ESTRADA VELÁZQUEZ FRANCISCO EVERARDO

ACOSTA TORRES JORGE ALFREDO

01 DE JUNIO DE 2023

Índice

Contenido

Código Main.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'flutter_flow/flutter_flow_theme.dart';
import 'flutter_flow/flutter_flow_util.dart';
import 'flutter_flow/internationalization.dart';
import 'flutter_flow/nav/nav.dart';
import 'index.dart';
```

Se importan los paquetes y las librerías que serán utilizadas para el desarrollo del proyecto.

Función main()

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await FlutterFlowTheme.initialize();

  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  await deleteProductosCarrito();

  runApp(MyApp());
}
```

En esta función se interactúa con el motor de Flutter, se inicializa Firebase, se manda llamar a la función para borrar los productos que existen en el carrito de compras en un inicio y, por último, se ejecuta la aplicación.

Clase MyApp{}

```
class MyApp extends StatefulWidget {
  // This widget is the root of your application.
  @override
  State<MyApp> createState() => _MyAppState();

  static _MyAppState of(BuildContext context) =>
      context.findAncestorStateOfType<_MyAppState>()!;
}
```

En esta clase se crea un estado que será la raíz de la aplicación con la llamada a _MyAppState();

Clase _MyAppState {}

```
class _MyAppState extends State<MyApp> {
  Locale? _locale;
  ThemeMode _themeMode = FlutterFlowTheme.themeMode;

  late AppStateNotifier _appStateNotifier;
  late GoRouter _router;

  @override
  void initState() {
    super.initState();
    _appStateNotifier = AppStateNotifier();
    _router = createRouter(_appStateNotifier);
  }

  void setLocale(String language) {
    setState(() => _locale = createLocale(language));
  }

  void setThemeMode(ThemeMode mode) => setState(() {
        _themeMode = mode;
        FlutterFlowTheme.saveThemeMode(mode);
      });

  @override
  Widget build(BuildContext context) {
    return MaterialApp.router(
      title: 'Dulceria DIANA 3',
      localizationsDelegates: [
        FFLocalizationsDelegate(),
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      locale: _locale,
      supportedLocales: const [Locale('en', '')],
      theme: ThemeData(brightness: Brightness.light),
      darkTheme: ThemeData(brightness: Brightness.dark),
      themeMode: _themeMode,
      routeInformationParser: _router.routeInformationParser,
      routerDelegate: _router.routerDelegate,
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Esta clase extiende de un estado de la clase “MyApp {}” y se inicializa el tema de la aplicación con FlutterFlow.

Función initState()
En esta función se inicializa el estado de la aplicación y crea las rutas necesarias.

Función setLocale()
En esta función se crea el lenguaje local con el que se tendrá interacción.

Función setThemeMode()
En esta función se establece el tema de fondo de la aplicación.

Función build()
En esta función se construyen las rutas y los temas de la aplicación que serán utilizados.

Clase NavBarPage {}

```
class NavBarPage extends StatefulWidget {
  NavBarPage({Key? key, this.initialPage, this.page}) : super(key: key);

  final String? initialPage;
  final Widget? page;

  @override
  _NavBarPageState createState() => _NavBarPageState();
}
```

En esta clase se crea el estado de la página principal (Inicio) con su respectivo nombre y Widget

Clase _NavBarPageState

```
/// This is the private State class that goes with NavBarPage.
class _NavBarPageState extends State<NavBarPage> {
  String _currentPageName = 'HomePage';
  late Widget? _currentPage;

  @override
  void initState() {
    super.initState();
    _currentPageName = widget.initialPage ?? _currentPageName;
    _currentPage = widget.page;
  }

  @override
  Widget build(BuildContext context) {
    final tabs = {
      'HomePage': HomePageWidget(),
      'AcercaDe': AcercaDeWidget(),
      'Productos': ProductosWidget(),
      'Carrito': CarritoWidget(),
      'Sesion': SesionWidget(),
    };
    final currentIndex = tabs.keys.toList().indexOf(_currentPageName);

    return Scaffold(
      body: _currentPage ?? tabs[_currentPageName],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: currentIndex,
        onTap: (i) => setState(() {
          _currentPage = null;
          _currentPageName = tabs.keys.toList()[i];
        }),
        backgroundColor: Color(0xFFFFF930),
        selectedItemColor: Colors.black,
```

```
        unselectedItemColor: Color(0x8A000000),
        showSelectedLabels: true,
        showUnselectedLabels: true,
        type: BottomNavigationBarType.fixed,
        items: <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(
              Icons.home_outlined,
              size: 24.0,
            ),
            label: 'Inicio',
            tooltip: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.announcement_outlined,
              size: 24.0,
            ),
            label: 'Acerca de',
            tooltip: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.fastfood,
              size: 24.0,
            ),
            label: 'Productos',
            tooltip: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.shopping_cart,
              size: 24.0,
            ),
            label: 'Carrito',
            tooltip: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.person,
              size: 24.0,
            ),
            label: 'Sesión',
            tooltip: '',
          )
        ],
      ),
    );
  }
}
```

Esta clase extiende de la clase NavBarPage {} y se le asigna el nombre de la pagina en la que se encuentra sobre el navbar.

Función initState()
Esta función inicializa los estados del navbar de cada página.

Función build()
Esta función manda a llamar a las páginas que se encuentran dentro del navbar. Además, regresa Widget que será mostrado en la aplicación y fungirá como nuestro navbar donde se tiene un botón de tipo navbar, un ícono y una etiqueta para darle a conocer al usuario sobre que trata esa página cuando la selecciona.

Código Index.dart

```dart
// Export pages
export '/pages/home_page/home_page_widget.dart' show HomePageWidget;
export '/pages/acerca_de/acerca_de_widget.dart' show AcercaDeWidget;
export '/pages/productos/productos_widget.dart' show ProductosWidget;
export '/pages/carrito/carrito_widget.dart' show CarritoWidget;
export '/pages/sesion/sesion_widget.dart' show SesionWidget;
export '/pages/dulces/dulces_widget.dart' show DulcesWidget;
export '/pages/chocolates/chocolates_widget.dart' show ChocolatesWidget;
export '/pages/frituras/frituras_widget.dart' show FriturasWidget;
export '/pages/plasticos_y_desechables/plasticos_y_desechables_widget.dart'
    show PlasticosYDesechablesWidget;
export '/pages/registrarte/registrarte_widget.dart' show RegistrarteWidget;
export '/pages/compra_realizada/compra_realizada_widget.dart'
    show CompraRealizadaWidget;
```

Se exportan todas las páginas a utilizar y las muestra.

Código firebase_options.dart

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
```

```
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyBr9coxWVJfhbzYdI2Q4cineh9RwKNuILA',
    appId: '1:384732469315:web:97b66fb6df0ec09b481cf6',
    messagingSenderId: '384732469315',
    projectId: 'ejemplo-2023-i',
    authDomain: 'ejemplo-2023-i.firebaseapp.com',
    storageBucket: 'ejemplo-2023-i.appspot.com',
  );

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyCZgenOU4CqS2DAkKSx4WDRFSz8THgZHME',
    appId: '1:384732469315:android:84ab77fd1ab59db4481cf6',
    messagingSenderId: '384732469315',
    projectId: 'ejemplo-2023-i',
    storageBucket: 'ejemplo-2023-i.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyDZw15ciycw_mKRblEw9Aw2o1Jz_5f8Srg',
    appId: '1:384732469315:ios:a040fc7aeab52f22481cf6',
    messagingSenderId: '384732469315',
    projectId: 'ejemplo-2023-i',
    storageBucket: 'ejemplo-2023-i.appspot.com',
    iosClientId: '384732469315-
fsncho42n06it92o2m0g2mkeet46u99g.apps.googleusercontent.com',
    iosBundleId: 'com.example.dulceriaDiana2',
  );

  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyDZw15ciycw_mKRblEw9Aw2o1Jz_5f8Srg',
    appId: '1:384732469315:ios:a040fc7aeab52f22481cf6',
    messagingSenderId: '384732469315',
    projectId: 'ejemplo-2023-i',
    storageBucket: 'ejemplo-2023-i.appspot.com',
    iosClientId: '384732469315-
```

```
fsncho42n06it92o2m0g2mkeet46u99g.apps.googleusercontent.com',
    iosBundleId: 'com.example.dulceriaDiana2',
  );
}
```

Este código se genera cuando se crea la conexión entre el proyecto de Flutter y Firebase.

Código servicios_firebase.dart

Import

```
import 'package:cloud_firestore/cloud_firestore.dart';
```

Se importa el lugar de almacenamiento donde se encuentra nuestra Firebase.

Variables globales

```
FirebaseFirestore bd = FirebaseFirestore.instance;

bool sesionIniciada = false;
```

La variable “bd” servirá para obtener las colecciones de nuestro firebase y así poder manipularlos.

La variable “sesionIniciada” servirá para saber si el usuario ya inicio sesión.

Función getProductosPrincipales()

```
Future<List> getProductosPrincipales() async {
  List productos = [];

  CollectionReference coleccionPaginaPrincipal =
bd.collection('paginaPrincipal');

  QuerySnapshot queryProductos = await coleccionPaginaPrincipal.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaPrincipal”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función getProductosDulces()

```
Future<List> getProductosDulces() async {
  List productos = [];
```

```
  CollectionReference coleccionPaginaDulces = bd.collection('paginaDulces');

  QuerySnapshot queryProductos = await coleccionPaginaDulces.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaDulces”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función getProductosChocolates()

```
Future<List> getProductosChocolates() async {
  List productos = [];

  CollectionReference coleccionPaginaChocolates =
bd.collection('paginaChocolates');

  QuerySnapshot queryProductos = await coleccionPaginaChocolates.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaChocolates”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función getProductosFrituras()

```
Future<List> getProductosFrituras() async {
  List productos = [];

  CollectionReference coleccionPaginaFrituras =
bd.collection('paginaFrituras');

  QuerySnapshot queryProductos = await coleccionPaginaFrituras.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaFrituras”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función getProductosPlasticosYDesechables()

```
Future<List> getProductosPlasticosYDesechables() async {
  List productos = [];

  CollectionReference coleccionPaginaPlasticosYDesechables =
bd.collection('paginaPlasticosYDesechables');

  QuerySnapshot queryProductos = await
coleccionPaginaPlasticosYDesechables.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaPlasticosYDesechables”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función getProductosCarrito()

```
Future<List> getProductosCarrito() async {
  List productos = [];

  CollectionReference coleccionPaginaCarrito =
bd.collection('paginaCarrito');

  QuerySnapshot queryProductos = await coleccionPaginaCarrito.get();

  queryProductos.docs.forEach((documento) {
    productos.add(documento.data());
  });

  return productos;
}
```

En esta función se obtienen todos los productos de la colección “paginaCarrito”, para ello se crea una lista, se declara la referencia de la colección a utilizar, se obtienen todos los documentos que se encuentran dentro de esta colección y se hace un ciclo para agregar cada producto en la lista. Por último, se retorna la lista.

Función addProductoEnCarrito()

```
Future<void> addProductoEnCarrito(String nombre, String precio) async {
  await bd.collection('paginaCarrito').add({
    'nombre': nombre,
    'precio': precio,
  });
}
```

En esta función se añaden productos en la colección “paginaCarrito” con los atributos “nombre” y “precio”.

Función realizaCompra()

```
Future<bool> realizaCompra() async {
  if(sesionIniciada) {
    QuerySnapshot queryProductos = await
bd.collection('paginaCarrito').get();

    if(queryProductos.size == 0) {
      return false;
    }

    for(var doc in queryProductos.docs) {
      final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

      final producto = {
        'uid': doc.id,
        'nombre': data['nombre'],
      };

      await updateInventario(producto['nombre']);

      bd.collection('paginaCarrito').doc(producto['uid']).delete();
    }

    return true;
  } else {
    return false;
  }
}
```

En esta función se verifica si el usuario ya inició sesión, en caso de que si se obtienen los documentos de la colección indicada, se verifica que la colección tenga documentos, se itera sobre cada documento se crea un producto para almacenar la información obtenida, se manda llamar a la función “updateInventario()” y, por último, se borran todos los documentos de la colección.

Función updateInventario()

```
Future<void> updateInventario(String nombre) async {
  QuerySnapshot queryProductos = await
bd.collection('paginaPrincipal').get();

  for(var doc in queryProductos.docs) {
```

```
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'imagen': data['imagen'],
      'nombre': data['nombre'],
      'descripcion': data['descripcion'],
      'precio': data['precio'],
      'inventario': data['inventario'],
    };

    if(producto['nombre'] == nombre) {
      String nuevoInventario;

      int viejoInventario = int.parse(data['inventario']);
      int resta = viejoInventario - 1;

      nuevoInventario = resta.toString();

      await bd.collection('paginaPrincipal').doc(producto['uid']).set({
        'imagen': data['imagen'],
        'nombre': data['nombre'],
        'descripcion': data['descripcion'],
        'precio': data['precio'],
        'inventario': nuevoInventario,
      });
    }
  }

  QuerySnapshot queryProductosDulces = await
bd.collection('paginaDulces').get();

  for(var doc in queryProductosDulces.docs) {
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'imagen': data['imagen'],
      'nombre': data['nombre'],
      'descripcion': data['descripcion'],
      'precio': data['precio'],
      'inventario': data['inventario'],
    };

    if(producto['nombre'] == nombre) {
      String nuevoInventario;

      int viejoInventario = int.parse(data['inventario']);
      int resta = viejoInventario - 1;

      nuevoInventario = resta.toString();

      await bd.collection('paginaDulces').doc(producto['uid']).set({
        'imagen': data['imagen'],
        'nombre': data['nombre'],
        'descripcion': data['descripcion'],
        'precio': data['precio'],
```

```
        'inventario': nuevoInventario,
      });
    }
  }

  QuerySnapshot queryProductosChocolates = await
bd.collection('paginaChocolates').get();

  for(var doc in queryProductosChocolates.docs) {
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'imagen': data['imagen'],
      'nombre': data['nombre'],
      'descripcion': data['descripcion'],
      'precio': data['precio'],
      'inventario': data['inventario'],
    };

    if(producto['nombre'] == nombre) {
      String nuevoInventario;

      int viejoInventario = int.parse(data['inventario']);
      int resta = viejoInventario - 1;

      nuevoInventario = resta.toString();

      await bd.collection('paginaChocolates').doc(producto['uid']).set({
        'imagen': data['imagen'],
        'nombre': data['nombre'],
        'descripcion': data['descripcion'],
        'precio': data['precio'],
        'inventario': nuevoInventario,
      });
    }
  }

  QuerySnapshot queryProductosFrituras = await
bd.collection('paginaFrituras').get();

  for(var doc in queryProductosFrituras.docs) {
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'imagen': data['imagen'],
      'nombre': data['nombre'],
      'descripcion': data['descripcion'],
      'precio': data['precio'],
      'inventario': data['inventario'],
    };

    if(producto['nombre'] == nombre) {
      String nuevoInventario;

      int viejoInventario = int.parse(data['inventario']);
```

```
      int resta = viejoInventario - 1;

      nuevoInventario = resta.toString();

      await bd.collection('paginaFrituras').doc(producto['uid']).set({
        'imagen': data['imagen'],
        'nombre': data['nombre'],
        'descripcion': data['descripcion'],
        'precio': data['precio'],
        'inventario': nuevoInventario,
      });
    }
  }

  QuerySnapshot queryProductosPyD = await
bd.collection('paginaPlasticosYDesechables').get();

  for(var doc in queryProductosPyD.docs) {
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'imagen': data['imagen'],
      'nombre': data['nombre'],
      'descripcion': data['descripcion'],
      'precio': data['precio'],
      'inventario': data['inventario'],
    };

    if(producto['nombre'] == nombre) {
      String nuevoInventario;

      int viejoInventario = int.parse(data['inventario']);
      int resta = viejoInventario - 1;

      nuevoInventario = resta.toString();

      await
bd.collection('paginaPlasticosYDesechables').doc(producto['uid']).set({
        'imagen': data['imagen'],
        'nombre': data['nombre'],
        'descripcion': data['descripcion'],
        'precio': data['precio'],
        'inventario': nuevoInventario,
      });
    }
  }
}
```

En esta función, para cada colección, se obtienen sus respectivos documentos, se itera sobre cada uno de ellos, se crea un producto para almacenar la información, se verifica que el nombre del producto obtenido del documento sea igual al que recibimos por parámetro, en caso de que si, se calcula el nuevo inventario que tendrá ese producto y, por último, se actualiza el producto de la colección en la que se encuentra.

Función deleteProductosCarrito()

```
Future<void> deleteProductosCarrito() async {
  QuerySnapshot queryProductos = await bd.collection('paginaCarrito').get();

  for(var doc in queryProductos.docs) {
    final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;

    final producto = {
      'uid': doc.id,
      'nombre': data['nombre'],
    };

    bd.collection('paginaCarrito').doc(producto['uid']).delete();
  }
}
```

En esta función se obtienen los documentos de la colección definida, se itera sobre cada uno de ellos, se obtiene su ID y se borran de la colección.

Función getUsuario()

```
Future<bool> getUsuario(String correo, String contrasena) async {
  QuerySnapshot queryUsuarios = await bd.collection('usuarios').get();

  for(var doc in queryUsuarios.docs) {
    final Map<String, dynamic> data =  doc.data() as Map<String, dynamic>;

    final usuario = {
      'uid': doc.id,
      'nombre': data['nombre'],
      'correo': data['correo'],
      'contrasena': data['contrasena'],
    };

    if(usuario['correo'] == correo && usuario['contrasena'] == contrasena) {
      sesionIniciada = true;

      return true;
    }
  }

  return false;
}
```

En esta función se obtienen los documentos que existen en la colección definida, se itera sobre cada uno de ellos, se crea un usuario para guardar su información, se verifica que el correo y la contraseña sea iguales (correctos) y se cambia el valor de la variable “sesionIniciada” a un true.

Función addUsuarios()

```
Future<bool> addUsuarios(String nombre, String correo, String contrasena)
async {
  if(!sesionIniciada) {
```

```
    await bd.collection('usuarios').add({
      'nombre': nombre,
      'correo': correo,
      'contrasena': contrasena,
    });

    Future<bool> res = getUsuario(correo, contrasena);

    if(await res) {
      sesionIniciada = true;

      return true;
    } else {
      return false;
    }
  } else {
    return false;
  }
}
```

En esta función se verifica que no se tenga la sesión iniciada, en caso de que no, se añade un nuevo documento (usuario) a la colección indicada y se cambia el valor de la variable “sesionIniciada” a un true.

Código acerca_de_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías que serán empleados en el código.

Clase AcercaDeModel {}

```
class AcercaDeModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

En esta clase se inicializa el estado de la página Acerca de

Código acerca_de_widget.dart

Imports

```
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
```

```dart
import 'package:flutter/material.dart';
import 'acerca_de_model.dart';
export 'acerca_de_model.dart';
```

Se importan los paquetes y las librerías a utilizar.

Clase AcercaDeWidget {}

```dart
class AcercaDeWidget extends StatefulWidget {
  const AcercaDeWidget({Key? key}) : super(key: key);

  @override
  _AcercaDeWidgetState createState() => _AcercaDeWidgetState();
}
```

Esta clase crea el estado de la página Acerca de

Clase _AcercaDeWidgetState

```dart
class _AcercaDeWidgetState extends State<AcercaDeWidget> {
  late AcercaDeModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => AcercaDeModel());
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Align(
            alignment: AlignmentDirectional(0.0, 0.0),
            child: Text(
              'Dulcería DIANA',
              textAlign: TextAlign.start,
              style: FlutterFlowTheme.of(context).headlineMedium.override(
                    fontFamily: 'Poppins',
```

```
                    color: Colors.black,
                    fontSize: 22.0,
                    fontWeight: FontWeight.bold,
                  ),
              ),
            ),
            actions: [],
            centerTitle: false,
            elevation: 2.0,
          ),
          body: SafeArea(
            top: true,
            child: SingleChildScrollView(
              child: Column(
                mainAxisSize: MainAxisSize.max,
                children: [
                  Align(
                    alignment: AlignmentDirectional(0.0, 0.0),
                    child: Text(
                      'Sobre nosotros...',
                      textAlign: TextAlign.start,
                      style: FlutterFlowTheme.of(context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 30.0,
                            fontWeight: FontWeight.bold,
                          ),
                    ),
                  ),
                  Padding(
                    padding: EdgeInsetsDirectional.fromSTEB(10.0, 0.0, 10.0,
0.0),
                    child: Text(
                      'Bienvenido(a), esta es una aplicación de ventas para una
dulcería que permite realizar compras a los visitantes que accedan a ella.',
                      textAlign: TextAlign.justify,
                      style: FlutterFlowTheme.of(context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 19.0,
                          ),
                    ),
                  ),
                  Padding(
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 20.0, 0.0,
0.0),
                    child: Image.network(
                      'https://firebasestorage.googleapis.com/v0/b/ejemplo-
2023-i.appspot.com/o/Dulces.jpg?alt=media&token=706456cc-0fb2-4e3e-85c4-
27f5380838af',
                      width: 300.0,
                      height: 200.0,
                      fit: BoxFit.contain,
                    ),
                  ),
                  Align(
                    alignment: AlignmentDirectional(0.0, 0.0),
                    child: Padding(
                      padding:
```

```
                      EdgeInsetsDirectional.fromSTEB(0.0, 20.0, 0.0, 0.0),
                  child: Text(
                    '¿Cómo se usa?',
                    textAlign: TextAlign.start,
                    style:
FlutterFlowTheme.of(context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 30.0,
                            fontWeight: FontWeight.bold,
                          ),
                    ),
                  ),
                ),
                Padding(
                  padding: EdgeInsetsDirectional.fromSTEB(10.0, 0.0, 10.0,
0.0),
                  child: Text(
                    '1. En la pantalla de inicio podrás navegar la página
para ver los productos.\n\n2. En la pestaña de \"productos\" podrás ver las
categorías de los productos y navegar entre ellos.\n\n3. Para poder realizar
la compra de los productos, deberás de tener sesión iniciada, en caso de no
contar con una cuenta, deberás crearte una.\n',
                    textAlign: TextAlign.justify,
                    style: FlutterFlowTheme.of(context).bodyMedium.override(
                          fontFamily: 'Poppins',
                          fontSize: 19.0,
                        ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Esta clase crea el widget que visualizamos en la aplicación.

Función initState()
Inicializa el estado de la página.

Función dispose()
Crea el modelo de la página.

Función build()
Crea el widget que se visualiza. Crea un AppBar que contiene como título el nombre de la tienda, crea un body que contiene una columna que a su vez contiene un alineado para el texto de título, un rellenado para el texto, otro para una imagen que lee de Firebase y otros 2 para más texto.

Código carrito_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías que serán empleados.

Clase CarritoModel {}

```
class CarritoModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

En esta clase se inicializa el estado de la página Carrito

Código carrito_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'carrito_model.dart';
export 'carrito_model.dart';
```

Paquetes y librerías que serán empleados.

Clase CarritoWidget {}

```
class CarritoWidget extends StatefulWidget {
  const CarritoWidget({Key? key}) : super(key: key);

  @override
  _CarritoWidgetState createState() => _CarritoWidgetState();
}
```

Crea el estado de la página carrito.

Clase _CarritoWidgetState {}

```
class _CarritoWidgetState extends State<CarritoWidget> {
  late CarritoModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
```

```dart
final _unfocusNode = FocusNode();

@override
void initState() {
  super.initState();
  _model = createModel(context, () => CarritoModel());
}

@override
void dispose() {
  _model.dispose();

  _unfocusNode.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      appBar: AppBar(
        backgroundColor: Color(0xFFFFF930),
        automaticallyImplyLeading: false,
        title: Align(
          alignment: AlignmentDirectional(0.0, 0.0),
          child: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
                  fontSize: 22.0,
                  fontWeight: FontWeight.bold,
                ),
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosCarrito(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
```

```
                            10.0, 20.0, 10.0, 0.0
                        ),
                        child: Row(
                          mainAxisSize: MainAxisSize.max,
                          children: [
                            Expanded(
                              child: Padding(
                                padding: EdgeInsetsDirectional.fromSTEB(
                                    10.0, 0.0, 10.0, 0.0
                                ),
                                child: Text(
                                  snapshot.data?[index]['nombre'],
                                  style:
                                  FlutterFlowTheme.of(
                                      context).bodyMedium.override(
                                    fontFamily: 'Poppins',
                                    fontSize: 20.0,
                                  ),
                                ),
                              ),
                            ),
                            Text(
                              ': ' + snapshot.data?[index]['precio'] + '
MXN',
                              style: FlutterFlowTheme.of(
                                  context).bodyMedium.override(
                                fontFamily: 'Poppins',
                                fontSize: 20.0,
                              ),
                            ),
                          ],
                        ),
                      ),
                    ),
                  ],
                );
              },
            );
          } else {
            return const Center(
              child: CircularProgressIndicator(),
            );
          }
        }),
      ),
    ),
    floatingActionButton: Align(
      alignment: AlignmentDirectional(0.0, 0.0),
      child: Padding(
        padding: EdgeInsetsDirectional.fromSTEB(20.0, 760.0, 0.0, 0.0),
        child: FFButtonWidget(
          onPressed: () async {
            Future<bool> res = realizaCompra();
            if(await res) {
              context.pushNamed('CompraRealizada');
            } else {
              showSnackbar(
                  context,
```

```
                    'Falta agregar productos al carrito o iniciar sesión'
                );
              }
            },
            text: 'Realizar pedido',
            options: FFButtonOptions(
              width: 130.0,
              height: 40.0,
              padding:
              EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
              iconPadding:
              EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
              color: Color(0xFF1DF94C),
              textStyle:
              FlutterFlowTheme.of(context).titleSmall.override(
                fontFamily: 'Poppins',
                color: Colors.black,
                fontSize: 20.0,
              ),
              borderSide: BorderSide(
                color: Colors.transparent,
                width: 1.0,
              ),
              borderRadius: BorderRadius.circular(8.0),
            ),
          ),
        ),
      ),
    ),
  );
 }
}
```

Función initState()
Inicializa el estado de la página.

Función dispose()
Crea el modelo de la página.

Función build()
Crea el widget que se visualiza, contiene un AppBar que a su vez contiene el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosCarrito()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con un botón que al dar clic en él, llama a la función “realizaCompra()” y cambia a la página que lleva por nombre “CompraRealizada”

Código chocolates_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase ChocolatesModel {}

```
class ChocolatesModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código chocolates_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'chocolates_model.dart';
export 'chocolates_model.dart';
```

Paquetes y librerías para utilizar.

Clase ChocolatesWidget {}

```
class ChocolatesWidget extends StatefulWidget {
  const ChocolatesWidget({Key? key}) : super(key: key);

  @override
  _ChocolatesWidgetState createState() => _ChocolatesWidgetState();
}
```

Se crea el estado de la página

Clase _ChocolatesWidgetState {}

```
class _ChocolatesWidgetState extends State<ChocolatesWidget> {
  late ChocolatesModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
```

```dart
final _unfocusNode = FocusNode();

@override
void initState() {
  super.initState();
  _model = createModel(context, () => ChocolatesModel());
}

@override
void dispose() {
  _model.dispose();

  _unfocusNode.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      appBar: AppBar(
        backgroundColor: Color(0xFFFFF930),
        automaticallyImplyLeading: false,
        title: Align(
          alignment: AlignmentDirectional(0.0, 0.0),
          child: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
                  fontSize: 22.0,
                  fontWeight: FontWeight.bold,
                ),
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosChocolates(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
```

```
                            0.0, 20.0, 0.0, 0.0
                        ),
                        child: Image.network(
                          snapshot.data?[index]['imagen'],
                          width: 200.0,
                          height: 225.0,
                          fit: BoxFit.contain,
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['nombre'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['descripcion'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          'Precio: ' + snapshot.data?[index]['precio'] + '
MXN',
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Align(
                        alignment: AlignmentDirectional(0.0, 0.0),
                        child: Padding(
                          padding: EdgeInsetsDirectional.fromSTEB(
                              0.0, 5.0, 0.0, 20.0
```

```
                            ),
                            child: FFButtonWidget(
                              onPressed: () {
                                print('Button pressed ...');

if(int.parse(snapshot.data?[index]['inventario']) > 0) {
                                addProductoEnCarrito(
                                    snapshot.data?[index]['nombre'],
                                    snapshot.data?[index]['precio']);
                                showSnackbar(
                                    context,
                                    'El producto se ha agregado al carrito'
                                );
                              } else {
                                showSnackbar(context, 'Inventario
agotado');
                              }
                            },
                            text: 'Agregar al carrito',
                            options: FFButtonOptions(
                              width: 130.0,
                              height: 40.0,
                              padding:
                              EdgeInsetsDirectional.fromSTEB(
                                  0.0, 0.0, 0.0, 0.0
                              ),
                              iconPadding:
                              EdgeInsetsDirectional.fromSTEB(
                                  0.0, 0.0, 0.0, 0.0
                              ),
                              color: Color(0xFF1DF94C),
                              textStyle:

FlutterFlowTheme.of(context).titleSmall.override(
                                fontFamily: 'Poppins',
                                color: Colors.black,
                                fontSize: 20.0,
                              ),
                              borderSide: BorderSide(
                                color: Colors.transparent,
                                width: 1.0,
                              ),
                              borderRadius: BorderRadius.circular(8.0),
                            ),
                          ),
                        ),
                      ),
                    ],
                  );
                },
              );
            } else {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
          }),
```

```
          ),
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosChocolates()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con con una columna que contiene una imagen leída de Firebase, el nombre del producto, la descripción, el precio y un botón que al dar clic en él llama a la función “addProductoEnCarrito()”.

Código compra_realizada_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase CompraRealizadaModel {}

```
class CompraRealizadaModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código compra_realizada_widget.dart

Imports

```
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
```

```dart
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'compra_realizada_model.dart';
export 'compra_realizada_model.dart';
```

Paquetes y librerías para utilizar.

Clase CompraRealizadaWidget {}

```dart
class CompraRealizadaWidget extends StatefulWidget {
  const CompraRealizadaWidget({Key? key}) : super(key: key);

  @override
  _CompraRealizadaWidgetState createState() => _CompraRealizadaWidgetState();
}
```

Se crea el estado de la página

Clase _CompraRealizadaWidgetState {}

```dart
class _CompraRealizadaWidgetState extends State<CompraRealizadaWidget> {
  late CompraRealizadaModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => CompraRealizadaModel());
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Align(
            alignment: AlignmentDirectional(0.0, 0.0),
            child: Text(
              'Dulcería DIANA',
              style: FlutterFlowTheme.of(context).headlineMedium.override(
                    fontFamily: 'Poppins',
```

```
                    color: Colors.black,
                    fontSize: 22.0,
                    fontWeight: FontWeight.bold,
                  ),
              ),
            ),
            actions: [],
            centerTitle: false,
            elevation: 2.0,
          ),
          body: SafeArea(
            top: true,
            child: Column(
              mainAxisSize: MainAxisSize.max,
              children: [
                Align(
                  alignment: AlignmentDirectional(0.0, 0.0),
                  child: Text(
                    '¡PEDIDO REALIZADO!',
                    style: FlutterFlowTheme.of(context).bodyMedium.override(
                          fontFamily: 'Poppins',
                          fontSize: 30.0,
                          fontWeight: FontWeight.bold,
                        ),
                  ),
                ),
                Padding(
                  padding: EdgeInsetsDirectional.fromSTEB(10.0, 30.0, 10.0,
0.0),
                  child: Text(
                    '¡Bien hecho! Tu pedido se ha realizado con éxito',
                    style: FlutterFlowTheme.of(context).bodyMedium.override(
                          fontFamily: 'Poppins',
                          fontSize: 20.0,
                        ),
                  ),
                ),
                Padding(
                  padding: EdgeInsetsDirectional.fromSTEB(0.0, 30.0, 0.0, 0.0),
                  child: FFButtonWidget(
                    onPressed: () async {
                      context.pushNamed('HomePage');
                    },
                    text: 'Inicio',
                    options: FFButtonOptions(
                      width: 130.0,
                      height: 40.0,
                      padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                      iconPadding:
                          EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                      color: Color(0xFF1DF94C),
                      textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                            fontFamily: 'Poppins',
                            color: Colors.black,
                            fontSize: 20.0,
```

```
                      ),
                  elevation: 3.0,
                  borderSide: BorderSide(
                    color: Colors.transparent,
                    width: 1.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra una columna con 2 textos y un botón que al dar clic en él se llama a la página “HomePage”.

Código dulces_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase DulcesModel {}

```
class DulcesModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código dulces_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'dulces_model.dart';
export 'dulces_model.dart';
```

Paquetes y librerías para utilizar.

Clase DulcesWidget {}

```
class DulcesWidget extends StatefulWidget {
  const DulcesWidget({Key? key}) : super(key: key);

  @override
  _DulcesWidgetState createState() => _DulcesWidgetState();
}
```

Se crea el estado de la página

Clase _DulcesWidgetState {}

```
class _DulcesWidgetState extends State<DulcesWidget> {
  late DulcesModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => DulcesModel());
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
```

```
        automaticallyImplyLeading: false,
        title: Align(
          alignment: AlignmentDirectional(0.0, 0.0),
          child: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
                  fontSize: 22.0,
                  fontWeight: FontWeight.bold,
                ),
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosDulces(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            0.0, 20.0, 0.0, 0.0
                        ),
                        child: Image.network(
                          snapshot.data?[index]['imagen'],
                          width: 200.0,
                          height: 225.0,
                          fit: BoxFit.contain,
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['nombre'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
```

```
                                10.0, 0.0, 10.0, 0.0
                            ),
                            child: Text(
                              snapshot.data?[index]['descripcion'],
                              textAlign: TextAlign.center,
                              style: FlutterFlowTheme.of(
                                  context).bodyMedium.override(
                                fontFamily: 'Poppins',
                                fontSize: 20.0,
                              ),
                            ),
                          ),
                          Padding(
                            padding: EdgeInsetsDirectional.fromSTEB(
                                10.0, 0.0, 10.0, 0.0
                            ),
                            child: Text(
                              'Precio: ' + snapshot.data?[index]['precio'] + '
MXN',
                              textAlign: TextAlign.center,
                              style: FlutterFlowTheme.of(
                                  context).bodyMedium.override(
                                fontFamily: 'Poppins',
                                fontSize: 20.0,
                              ),
                            ),
                          ),
                          Align(
                            alignment: AlignmentDirectional(0.0, 0.0),
                            child: Padding(
                              padding: EdgeInsetsDirectional.fromSTEB(
                                  0.0, 5.0, 0.0, 20.0
                              ),
                              child: FFButtonWidget(
                                onPressed: () {
                                  print('Button pressed ...');

if(int.parse(snapshot.data?[index]['inventario']) > 0) {
                                    addProductoEnCarrito(
                                        snapshot.data?[index]['nombre'],
                                        snapshot.data?[index]['precio']);
                                    showSnackbar(
                                        context,
                                        'El producto se ha agregado al carrito'
                                    );
                                  } else {
                                    showSnackbar(context, 'Inventario
agotado');

                                  }
                                },
                                text: 'Agregar al carrito',
                                options: FFButtonOptions(
                                  width: 130.0,
                                  height: 40.0,
                                  padding:
                                  EdgeInsetsDirectional.fromSTEB(
                                      0.0, 0.0, 0.0, 0.0
```

```
                                ),
                                iconPadding:
                                EdgeInsetsDirectional.fromSTEB(
                                    0.0, 0.0, 0.0, 0.0
                                ),
                                color: Color(0xFF1DF94C),
                                textStyle:
                                FlutterFlowTheme.of(
                                    context).titleSmall.override(
                                  fontFamily: 'Poppins',
                                  color: Colors.black,
                                  fontSize: 20.0,
                                ),
                                borderSide: BorderSide(
                                  color: Colors.transparent,
                                  width: 1.0,
                                ),
                                borderRadius: BorderRadius.circular(8.0),
                              ),
                            ),
                          ),
                        ),
                      ),
                    ],
                  );
                },
              );
            } else {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
          }),
        ),
      ),
    ),
  );
 }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosDulces()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con con una columna que contiene una imagen leída de Firebase, el nombre del producto, la descripción, el precio y un botón que al dar clic en él llama a la función “addProductoEnCarrito()”.

Código frituras_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase FriturasModel {}

```
class FriturasModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código frituras_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'frituras_model.dart';
export 'frituras_model.dart';
```

Paquetes y librerías para utilizar.

Clase FriturasWidget {}

```
class FriturasWidget extends StatefulWidget {
  const FriturasWidget({Key? key}) : super(key: key);

  @override
  _FriturasWidgetState createState() => _FriturasWidgetState();
}
```

Se crea el estado de la página

Clase _FriturasWidgetState {}

```
class _FriturasWidgetState extends State<FriturasWidget> {
  late FriturasModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
```

```dart
final _unfocusNode = FocusNode();

@override
void initState() {
  super.initState();
  _model = createModel(context, () => FriturasModel());
}

@override
void dispose() {
  _model.dispose();

  _unfocusNode.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      appBar: AppBar(
        backgroundColor: Color(0xFFFFF930),
        automaticallyImplyLeading: false,
        title: Align(
          alignment: AlignmentDirectional(0.0, 0.0),
          child: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
                  fontSize: 22.0,
                  fontWeight: FontWeight.bold,
                ),
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosFrituras(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
```

```
                            0.0, 20.0, 0.0, 0.0
                        ),
                        child: Image.network(
                          snapshot.data?[index]['imagen'],
                          width: 200.0,
                          height: 225.0,
                          fit: BoxFit.contain,
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['nombre'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['descripcion'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          'Precio: ' + snapshot.data?[index]['precio'] + '
MXN',
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Align(
                        alignment: AlignmentDirectional(0.0, 0.0),
                        child: Padding(
                          padding: EdgeInsetsDirectional.fromSTEB(
                              0.0, 5.0, 0.0, 20.0
```

```
                            ),
                            child: FFButtonWidget(
                              onPressed: () {
                                print('Button pressed ...');

if(int.parse(snapshot.data?[index]['inventario']) > 0) {
                                addProductoEnCarrito(
                                    snapshot.data?[index]['nombre'],
                                    snapshot.data?[index]['precio']);
                                showSnackbar(
                                    context,
                                    'El producto se ha agregado al carrito'
                                );
                              } else {
                                showSnackbar(context, 'Inventario
agotado');
                              }
                            },
                            text: 'Agregar al carrito',
                            options: FFButtonOptions(
                              width: 130.0,
                              height: 40.0,
                              padding:
                              EdgeInsetsDirectional.fromSTEB(
                                  0.0, 0.0, 0.0, 0.0
                              ),
                              iconPadding:
                              EdgeInsetsDirectional.fromSTEB(
                                  0.0, 0.0, 0.0, 0.0
                              ),
                              color: Color(0xFF1DF94C),
                              textStyle:
                              FlutterFlowTheme.of(
                                  context).titleSmall.override(
                                fontFamily: 'Poppins',
                                color: Colors.black,
                                fontSize: 20.0,
                              ),
                              borderSide: BorderSide(
                                color: Colors.transparent,
                                width: 1.0,
                              ),
                              borderRadius: BorderRadius.circular(8.0),
                            ),
                          ),
                        ),
                      ),
                    ],
                  );
                },
              );
            } else {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
          }),
```

```
          ),
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosFrituras()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con con una columna que contiene una imagen leída de Firebase, el nombre del producto, la descripción, el precio y un botón que al dar clic en él llama a la función “addProductoEnCarrito()”.

Código home_page_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase HomePageModel {}

```
class HomePageModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código home_page_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
```

```dart
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'home_page_model.dart';
export 'home_page_model.dart';
```

Paquetes y librerías para utilizar.

Clase HomePageWidget {}

```dart
class HomePageWidget extends StatefulWidget {
  HomePageWidget({Key? key}) : super(key: key);

  @override
  _HomePageWidgetState createState() => _HomePageWidgetState();
}
```

Se crea el estado de la página

Clase _HomePageWidgetState {}

```dart
class _HomePageWidgetState extends State<HomePageWidget> {
  late HomePageModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => HomePageModel());
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
```

```
              fontSize: 22.0,
              fontWeight: FontWeight.bold,
            ),
        ),
        actions: [],
        centerTitle: true,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosPrincipales(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            0.0, 20.0, 0.0, 0.0
                        ),
                        child: Image.network(
                          snapshot.data?[index]['imagen'],
                          width: 200.0,
                          height: 225.0,
                          fit: BoxFit.contain,
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['nombre'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
                          ),
                        ),
                      ),
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            10.0, 0.0, 10.0, 0.0
                        ),
                        child: Text(
                          snapshot.data?[index]['descripcion'],
                          textAlign: TextAlign.center,
                          style: FlutterFlowTheme.of(
                              context).bodyMedium.override(
                            fontFamily: 'Poppins',
                            fontSize: 20.0,
```

```
                              ),
                            ),
                          ),
                          Padding(
                            padding: EdgeInsetsDirectional.fromSTEB(
                                10.0, 0.0, 10.0, 0.0
                            ),
                            child: Text(
                              'Precio: ' + snapshot.data?[index]['precio'] + '
MXN',
                              textAlign: TextAlign.center,
                              style: FlutterFlowTheme.of(
                                  context).bodyMedium.override(
                                fontFamily: 'Poppins',
                                fontSize: 20.0,
                              ),
                            ),
                          ),
                          Align(
                            alignment: AlignmentDirectional(0.0, 0.0),
                            child: Padding(
                              padding: EdgeInsetsDirectional.fromSTEB(
                                  0.0, 5.0, 0.0, 20.0
                              ),
                              child: FFButtonWidget(
                                onPressed: () {
                                  print('Button pressed ...');

if(int.parse(snapshot.data?[index]['inventario']) > 0) {
                                    addProductoEnCarrito(
                                        snapshot.data?[index]['nombre'],
                                        snapshot.data?[index]['precio']);
                                    showSnackbar(
                                        context,
                                        'El producto se ha agregado al carrito'
                                    );
                                  } else {
                                    showSnackbar(context, 'Inventario
agotado');
                                  }
                                },
                                text: 'Agregar al carrito',
                                options: FFButtonOptions(
                                  width: 130.0,
                                  height: 40.0,
                                  padding:
                                  EdgeInsetsDirectional.fromSTEB(
                                      0.0, 0.0, 0.0, 0.0
                                  ),
                                  iconPadding:
                                  EdgeInsetsDirectional.fromSTEB(
                                      0.0, 0.0, 0.0, 0.0
                                  ),
                                  color: Color(0xFF1DF94C),
                                  textStyle:
                                  FlutterFlowTheme.of(
                                      context).titleSmall.override(
```

```
                                  fontFamily: 'Poppins',
                                  color: Colors.black,
                                  fontSize: 20.0,
                                ),
                                borderSide: BorderSide(
                                  color: Colors.transparent,
                                  width: 1.0,
                                ),
                                borderRadius: BorderRadius.circular(8.0),
                              ),
                            ),
                          ),
                        ),
                      ],
                    );
                  },
                );
              } else {
                return const Center(
                  child: CircularProgressIndicator(),
                );
              }
            }),
          ),
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosPrincipales()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con con una columna que contiene una imagen leída de Firebase, el nombre del producto, la descripción, el precio y un botón que al dar clic en él llama a la función “addProductoEnCarrito()”.

Código plásticos_y_desechables_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase PlasticosYDesechablesModel {}

```
class PlasticosYDesechablesModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código plásticos_y_desechables_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'plasticos_y_desechables_model.dart';
export 'plasticos_y_desechables_model.dart';
```

Paquetes y librerías para utilizar.

Clase PlasticosYDesechablesWidget {}

```
class PlasticosYDesechablesWidget extends StatefulWidget {
  const PlasticosYDesechablesWidget({Key? key}) : super(key: key);

  @override
  _PlasticosYDesechablesWidgetState createState() =>
      _PlasticosYDesechablesWidgetState();
}
```

Se crea el estado de la página

Clase _PlasticosYDesechablesWidgetState {}

```
class _PlasticosYDesechablesWidgetState
    extends State<PlasticosYDesechablesWidget> {
  late PlasticosYDesechablesModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => PlasticosYDesechablesModel());
```

```dart
}

@override
void dispose() {
  _model.dispose();

  _unfocusNode.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
    child: Scaffold(
      key: scaffoldKey,
      backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
      appBar: AppBar(
        backgroundColor: Color(0xFFFFF930),
        automaticallyImplyLeading: false,
        title: Align(
          alignment: AlignmentDirectional(0.0, 0.0),
          child: Text(
            'Dulcería DIANA',
            style: FlutterFlowTheme.of(context).headlineMedium.override(
                  fontFamily: 'Poppins',
                  color: Colors.black,
                  fontSize: 22.0,
                  fontWeight: FontWeight.bold,
                ),
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: FutureBuilder<List>(
          future: getProductosPlasticosYDesechables(),
          builder: ((context, snapshot) {
            if(snapshot.hasData) {
              return ListView.builder(
                padding: EdgeInsets.zero,
                scrollDirection: Axis.vertical,
                itemCount: snapshot.data?.length,
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Padding(
                        padding: EdgeInsetsDirectional.fromSTEB(
                            0.0, 20.0, 0.0, 0.0
                        ),
                        child: Image.network(
                          snapshot.data?[index]['imagen'],
                          width: 200.0,
                          height: 225.0,
```

```
                            fit: BoxFit.contain,
                          ),
                        ),
                        Padding(
                          padding: EdgeInsetsDirectional.fromSTEB(
                              10.0, 0.0, 10.0, 0.0
                          ),
                          child: Text(
                            snapshot.data?[index]['nombre'],
                            textAlign: TextAlign.center,
                            style: FlutterFlowTheme.of(
                                context).bodyMedium.override(
                              fontFamily: 'Poppins',
                              fontSize: 20.0,
                            ),
                          ),
                        ),
                        Padding(
                          padding: EdgeInsetsDirectional.fromSTEB(
                              10.0, 0.0, 10.0, 0.0
                          ),
                          child: Text(
                            snapshot.data?[index]['descripcion'],
                            textAlign: TextAlign.center,
                            style: FlutterFlowTheme.of(
                                context).bodyMedium.override(
                              fontFamily: 'Poppins',
                              fontSize: 20.0,
                            ),
                          ),
                        ),
                        Padding(
                          padding: EdgeInsetsDirectional.fromSTEB(
                              10.0, 0.0, 10.0, 0.0
                          ),
                          child: Text(
                            'Precio: ' + snapshot.data?[index]['precio'] + '
MXN',
                            textAlign: TextAlign.center,
                            style: FlutterFlowTheme.of(
                                context).bodyMedium.override(
                              fontFamily: 'Poppins',
                              fontSize: 20.0,
                            ),
                          ),
                        ),
                        Align(
                          alignment: AlignmentDirectional(0.0, 0.0),
                          child: Padding(
                            padding: EdgeInsetsDirectional.fromSTEB(
                                0.0, 5.0, 0.0, 20.0
                            ),
                            child: FFButtonWidget(
                              onPressed: () {
                                print('Button pressed ...');

if(int.parse(snapshot.data?[index]['inventario']) > 0) {
```

```
                                    addProductoEnCarrito(
                                        snapshot.data?[index]['nombre'],
                                        snapshot.data?[index]['precio']);
                                    showSnackbar(
                                        context,
                                        'El producto se ha agregado al carrito'
                                    );
                                  } else {
                                    showSnackbar(context, 'Inventario
agotado');
                                  }
                                },
                                text: 'Agregar al carrito',
                                options: FFButtonOptions(
                                  width: 130.0,
                                  height: 40.0,
                                  padding:
                                  EdgeInsetsDirectional.fromSTEB(
                                      0.0, 0.0, 0.0, 0.0
                                  ),
                                  iconPadding:
                                  EdgeInsetsDirectional.fromSTEB(
                                      0.0, 0.0, 0.0, 0.0
                                  ),
                                  color: Color(0xFF1DF94C),
                                  textStyle:
                                  FlutterFlowTheme.of(
                                      context).titleSmall.override(
                                    fontFamily: 'Poppins',
                                    color: Colors.black,
                                    fontSize: 20.0,
                                  ),
                                  borderSide: BorderSide(
                                    color: Colors.transparent,
                                    width: 1.0,
                                  ),
                                  borderRadius: BorderRadius.circular(8.0),
                                ),
                              ),
                            ),
                          ),
                        ],
                      );
                    },
                  );
                } else {
                  return const Center(
                    child: CircularProgressIndicator(),
                  );
                }
              }),
            ),
          ),
        ),
      );
    }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un FutureBuilder que servirá para ciclar los siguientes elementos (hijos) y construir de manera dinámica la página, llama a la función “getProductosPlasticosYDesechables()” y se construye un ListView con los documentos obtenidos de la función. Cuenta con con una columna que contiene una imagen leída de Firebase, el nombre del producto, la descripción, el precio y un botón que al dar clic en él llama a la función “addProductoEnCarrito()”.

Código productos_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase ProductosModel {}

```
class ProductosModel extends FlutterFlowModel {
  /// Initialization and disposal methods.

  void initState(BuildContext context) {}

  void dispose() {}

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página.

Código productos_widget.dart

Imports

```
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'productos_model.dart';
export 'productos_model.dart';
```

Paquetes y librerías para utilizar.

Clase ProductosWidget {}

```
class ProductosWidget extends StatefulWidget {
  const ProductosWidget({Key? key}) : super(key: key);

  @override
  _ProductosWidgetState createState() => _ProductosWidgetState();
}
```

Se crea el estado de la página

Clase _ProductosWidgetState {}

```
class _ProductosWidgetState extends State<ProductosWidget> {
  late ProductosModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => ProductosModel());
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Align(
            alignment: AlignmentDirectional(0.0, 0.0),
            child: Text(
              'Dulcería DIANA',
              style: FlutterFlowTheme.of(context).headlineMedium.override(
                    fontFamily: 'Poppins',
                    color: Colors.black,
                    fontSize: 22.0,
                    fontWeight: FontWeight.bold,
                  ),
            ),
          ),
```

```
          actions: [],
          centerTitle: false,
          elevation: 2.0,
        ),
        body: SafeArea(
          top: true,
          child: Column(
            mainAxisSize: MainAxisSize.max,
            children: [
              Align(
                alignment: AlignmentDirectional(0.0, 0.0),
                child: Text(
                  'Categorías',
                  textAlign: TextAlign.center,
                  style: FlutterFlowTheme.of(context).bodyMedium.override(
                        fontFamily: 'Poppins',
                        fontSize: 30.0,
                        fontWeight: FontWeight.bold,
                      ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(10.0, 10.0, 10.0,
20.0),
                child: FFButtonWidget(
                  onPressed: () async {
                    context.pushNamed('Dulces');
                  },
                  text: 'Dulces',
                  options: FFButtonOptions(
                    width: double.infinity,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(10.0, 0.0, 10.0,
20.0),
                child: FFButtonWidget(
                  onPressed: () async {
                    context.pushNamed('Chocolates');
```

```
                  },
                  text: 'Chocolates',
                  options: FFButtonOptions(
                    width: double.infinity,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(10.0, 0.0, 10.0,
20.0),
                child: FFButtonWidget(
                  onPressed: () async {
                    context.pushNamed('Frituras');
                  },
                  text: 'Frituras',
                  options: FFButtonOptions(
                    width: double.infinity,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(10.0, 0.0, 10.0,
0.0),
```

```
                child: FFButtonWidget(
                  onPressed: () async {
                    context.pushNamed('PlasticosYDesechables');
                  },
                  text: 'Plásticos y Desechables',
                  options: FFButtonOptions(
                    width: double.infinity,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra un texto “Categorías” y 4 botones con el nombre de cada categoría, según el botón al que se le dé clic, es a la página que te mandará.

Código registrarte_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase RegistrarteModel {}

```
class RegistrarteModel extends FlutterFlowModel {
  ///  State fields for stateful widgets in this page.

  // State field(s) for TextField widget.
  TextEditingController? textController1;
  String? Function(BuildContext, String?)? textController1Validator;
  // State field(s) for TextField widget.
  TextEditingController? textController2;
  String? Function(BuildContext, String?)? textController2Validator;
  // State field(s) for TextField widget.
  TextEditingController? textController3;
  late bool passwordVisibility;
  String? Function(BuildContext, String?)? textController3Validator;

  /// Initialization and disposal methods.

  void initState(BuildContext context) {
    passwordVisibility = false;
  }

  void dispose() {
    textController1?.dispose();
    textController2?.dispose();
    textController3?.dispose();
  }

  /// Additional helper methods are added here.

}
```

Se inicializan los controladores del texto y el estado de la página.

Código registrarte_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'registrarte_model.dart';
export 'registrarte_model.dart';
```

Paquetes y librerías para utilizar.

Clase RegistrarteWidget {}

```
class RegistrarteWidget extends StatefulWidget {
  const RegistrarteWidget({Key? key}) : super(key: key);

  @override
```

```
  _RegistrarteWidgetState createState() => _RegistrarteWidgetState();
}
```

Se crea el estado de la página

Clase _RegistrarteWidgetState {}

```
class _RegistrarteWidgetState extends State<RegistrarteWidget> {
  late RegistrarteModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  TextEditingController nombreController = TextEditingController(text: '');
  TextEditingController correoController = TextEditingController(text: '');
  TextEditingController contrasenaController = TextEditingController(text:
'');

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => RegistrarteModel());

    _model.textController1 ??= TextEditingController();
    _model.textController2 ??= TextEditingController();
    _model.textController3 ??= TextEditingController();
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Align(
            alignment: AlignmentDirectional(0.0, 0.0),
            child: Text(
              'Dulcería DIANA',
              style: FlutterFlowTheme.of(context).headlineMedium.override(
                    fontFamily: 'Poppins',
                    color: Colors.black,
                    fontSize: 22.0,
                    fontWeight: FontWeight.bold,
                  ),
```

```
          ),
        ),
        actions: [],
        centerTitle: false,
        elevation: 2.0,
      ),
      body: SafeArea(
        top: true,
        child: Column(
          mainAxisSize: MainAxisSize.max,
          children: [
            Align(
              alignment: AlignmentDirectional(0.0, 0.0),
              child: Text(
                'Registrarte',
                style: FlutterFlowTheme.of(context).bodyMedium.override(
                      fontFamily: 'Poppins',
                      fontSize: 30.0,
                      fontWeight: FontWeight.bold,
                    ),
              ),
            ),
            Padding(
              padding: EdgeInsetsDirectional.fromSTEB(0.0, 20.0, 0.0, 0.0),
              child: Text(
                'Nombre completo:',
                style: FlutterFlowTheme.of(context).bodyMedium.override(
                      fontFamily: 'Poppins',
                      fontSize: 20.0,
                    ),
              ),
            ),
            Padding(
              padding: EdgeInsetsDirectional.fromSTEB(8.0, 0.0, 8.0, 0.0),
              child: Container(
                width: MediaQuery.of(context).size.width * 0.9,
                child: TextFormField(
                  controller: nombreController,
                  autofocus: true,
                  obscureText: false,
                  decoration: InputDecoration(
                    hintText: 'Ingresa tu nombre',
                    hintStyle:
                        FlutterFlowTheme.of(context).labelMedium.override(
                              fontFamily: 'Poppins',
                              fontSize: 14.0,
                              fontWeight: FontWeight.normal,
                            ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0x00000000),
                        width: 2.0,
                      ),
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: BorderSide(
```

```
                    color: FlutterFlowTheme.of(context).primary,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: FlutterFlowTheme.of(context).error,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: FlutterFlowTheme.of(context).error,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                filled: true,
                fillColor: FlutterFlowTheme.of(context).lineColor,
              ),
              style: FlutterFlowTheme.of(context).bodyMedium,
              validator:
                  _model.textController1Validator.asValidator(context),
            ),
          ),
        ),
        Padding(
          padding: EdgeInsetsDirectional.fromSTEB(0.0, 10.0, 0.0, 0.0),
          child: Text(
            'Correo electrónico',
            style: FlutterFlowTheme.of(context).bodyMedium.override(
                  fontFamily: 'Poppins',
                  fontSize: 20.0,
                ),
          ),
        ),
        Padding(
          padding: EdgeInsetsDirectional.fromSTEB(8.0, 0.0, 8.0, 0.0),
          child: Container(
            width: MediaQuery.of(context).size.width * 0.9,
            child: TextFormField(
              controller: correoController,
              autofocus: true,
              obscureText: false,
              decoration: InputDecoration(
                hintText: 'Ingresa tu correo',
                hintStyle:
                    FlutterFlowTheme.of(context).labelMedium.override(
                          fontFamily: 'Poppins',
                          fontSize: 14.0,
                          fontWeight: FontWeight.normal,
                        ),
                enabledBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0x00000000),
```

```
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: FlutterFlowTheme.of(context).primary,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: FlutterFlowTheme.of(context).error,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: FlutterFlowTheme.of(context).error,
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(8.0),
                ),
                filled: true,
                fillColor: FlutterFlowTheme.of(context).lineColor,
              ),
              style: FlutterFlowTheme.of(context).bodyMedium,
              keyboardType: TextInputType.emailAddress,
              validator:
                  _model.textController2Validator.asValidator(context),
            ),
          ),
        ),
        Padding(
          padding: EdgeInsetsDirectional.fromSTEB(0.0, 10.0, 0.0, 0.0),
          child: Text(
            'Contraseña',
            style: FlutterFlowTheme.of(context).bodyMedium.override(
                  fontFamily: 'Poppins',
                  fontSize: 20.0,
                ),
          ),
        ),
        Padding(
          padding: EdgeInsetsDirectional.fromSTEB(8.0, 0.0, 8.0, 0.0),
          child: Container(
            width: MediaQuery.of(context).size.width * 0.9,
            child: TextFormField(
              controller: contrasenaController,
              autofocus: true,
              obscureText: !_model.passwordVisibility,
              decoration: InputDecoration(
                hintText: 'Ingresa tu contraseña',
                hintStyle:
                    FlutterFlowTheme.of(context).labelMedium.override(
```

```
                          fontFamily: 'Poppins',
                          fontSize: 14.0,
                          fontWeight: FontWeight.normal,
                        ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0x00000000),
                        width: 2.0,
                      ),
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: FlutterFlowTheme.of(context).primary,
                        width: 2.0,
                      ),
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: FlutterFlowTheme.of(context).error,
                        width: 2.0,
                      ),
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    focusedErrorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: FlutterFlowTheme.of(context).error,
                        width: 2.0,
                      ),
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    filled: true,
                    fillColor: FlutterFlowTheme.of(context).lineColor,
                    suffixIcon: InkWell(
                      onTap: () => setState(
                        () => _model.passwordVisibility =
                            !_model.passwordVisibility,
                      ),
                      focusNode: FocusNode(skipTraversal: true),
                      child: Icon(
                        _model.passwordVisibility
                            ? Icons.visibility_outlined
                            : Icons.visibility_off_outlined,
                        size: 20.0,
                      ),
                    ),
                  ),
                  style: FlutterFlowTheme.of(context).bodyMedium,
                  validator:
                      _model.textController3Validator.asValidator(context),
                ),
              ),
            ),
            Padding(
              padding: EdgeInsetsDirectional.fromSTEB(0.0, 30.0, 0.0, 0.0),
              child: FFButtonWidget(
```

```
                  onPressed: () async {
                    print('Button pressed ...');
                    Future<bool> res = addUsuarios(nombreController.text,
correoController.text, contrasenaController.text);
                    if(await res) {
                      showSnackbar(
                          context,
                          'Se ha agregado correctamente y se ha iniciado
sesión'
                      );
                    } else {
                      showSnackbar(
                          context,
                          'Se ha producido un error, intentelo más tarde');
                    }
                  },
                  text: 'Registrarte',
                  options: FFButtonOptions(
                    width: 130.0,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    elevation: 3.0,
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Función initState()
Se inicializa el estado de la página y de los controladores.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra varios textos y 3 TextFormField para que el usuario pueda ingresar sus datos, también que cuenta con un botón que al darle clic llama a la función “addUsuarios()”.

Código sesión_model.dart

Imports

```
import '/flutter_flow/flutter_flow_util.dart';
import 'package:flutter/material.dart';
```

Paquetes y librerías para utilizar.

Clase SesionModel {}

```
class SesionModel extends FlutterFlowModel {
  ///  State fields for stateful widgets in this page.

  // State field(s) for TextField widget.
  TextEditingController? textController1;
  String? Function(BuildContext, String?)? textController1Validator;
  // State field(s) for TextField widget.
  TextEditingController? textController2;
  late bool passwordVisibility;
  String? Function(BuildContext, String?)? textController2Validator;

  /// Initialization and disposal methods.

  void initState(BuildContext context) {
    passwordVisibility = false;
  }

  void dispose() {
    textController1?.dispose();
    textController2?.dispose();
  }

  /// Additional helper methods are added here.

}
```

Se inicializa el estado de la página y los controladores.

Código sesión_widget.dart

Imports

```
import 'package:dulceria_d_i_a_n_a_3/servicios/servicios_firebase.dart';
import '/flutter_flow/flutter_flow_theme.dart';
import '/flutter_flow/flutter_flow_util.dart';
import '/flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
```

```dart
import 'sesion_model.dart';
export 'sesion_model.dart';
```

Paquetes y librerías para utilizar.

Clase SesionWidget {}

```dart
class SesionWidget extends StatefulWidget {
  const SesionWidget({Key? key}) : super(key: key);

  @override
  _SesionWidgetState createState() => _SesionWidgetState();
}
```

Se crea el estado de la página.

Clase _SesionWidgetState {}

```dart
class _SesionWidgetState extends State<SesionWidget> {
  late SesionModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();
  final _unfocusNode = FocusNode();

  TextEditingController correoController = TextEditingController(text: '');
  TextEditingController contrasenaController = TextEditingController(text:
'');

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => SesionModel());

    _model.textController1 ??= TextEditingController();
    _model.textController2 ??= TextEditingController();
  }

  @override
  void dispose() {
    _model.dispose();

    _unfocusNode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).requestFocus(_unfocusNode),
      child: Scaffold(
        key: scaffoldKey,
        backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
        appBar: AppBar(
          backgroundColor: Color(0xFFFFF930),
          automaticallyImplyLeading: false,
          title: Align(
```

```
            alignment: AlignmentDirectional(0.0, 0.0),
            child: Text(
              'Dulcería DIANA',
              style: FlutterFlowTheme.of(context).headlineMedium.override(
                    fontFamily: 'Poppins',
                    color: Colors.black,
                    fontSize: 22.0,
                    fontWeight: FontWeight.bold,
                  ),
            ),
          ),
          actions: [],
          centerTitle: false,
          elevation: 2.0,
        ),
        body: SafeArea(
          top: true,
          child: Column(
            mainAxisSize: MainAxisSize.max,
            children: [
              Align(
                alignment: AlignmentDirectional(0.0, 0.0),
                child: Text(
                  'Iniciar sesión',
                  style: FlutterFlowTheme.of(context).bodyMedium.override(
                        fontFamily: 'Poppins',
                        fontSize: 30.0,
                        fontWeight: FontWeight.bold,
                      ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(0.0, 20.0, 0.0, 0.0),
                child: Text(
                  'Correo electrónico',
                  style: FlutterFlowTheme.of(context).bodyMedium.override(
                        fontFamily: 'Poppins',
                        fontSize: 20.0,
                      ),
                ),
              ),
              Container(
                width: MediaQuery.of(context).size.width * 0.9,
                child: TextFormField(
                  //controller: _model.textController1,
                  controller: correoController,
                  autofocus: true,
                  obscureText: false,
                  decoration: InputDecoration(
                    hintText: 'Ingresa tu correo',
                    hintStyle:
FlutterFlowTheme.of(context).bodySmall.override(
                          fontFamily: 'Poppins',
                          fontSize: 14.0,
                        ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: BorderSide(
```

```
                    color: Color(0x00000000),
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(15.0),
                ),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0xFF4B39EF),
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(15.0),
                ),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0xFFE21C3D),
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(15.0),
                ),
                focusedErrorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0xFFE21C3D),
                    width: 2.0,
                  ),
                  borderRadius: BorderRadius.circular(15.0),
                ),
                filled: true,
                fillColor: FlutterFlowTheme.of(context).lineColor,
              ),
              style: FlutterFlowTheme.of(context).bodyMedium.override(
                    fontFamily: 'Poppins',
                    fontSize: 14.0,
                  ),
              keyboardType: TextInputType.emailAddress,
              validator:
                  _model.textController1Validator.asValidator(context),
            ),
          ),
          Padding(
            padding: EdgeInsetsDirectional.fromSTEB(0.0, 10.0, 0.0, 0.0),
            child: Text(
              'Contraseña',
              style: FlutterFlowTheme.of(context).bodyMedium.override(
                    fontFamily: 'Poppins',
                    fontSize: 20.0,
                  ),
            ),
          ),
          Container(
            width: MediaQuery.of(context).size.width * 0.9,
            child: TextFormField(
              //controller: _model.textController2,
              controller: contrasenaController,
              autofocus: true,
              obscureText: !_model.passwordVisibility,
              decoration: InputDecoration(
                hintText: 'Ingresa tu contraseña',
```

```
                    hintStyle:
FlutterFlowTheme.of(context).bodySmall.override(
                          fontFamily: 'Poppins',
                          fontSize: 14.0,
                        ),
                    enabledBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0x00000000),
                        width: 1.0,
                      ),
                      borderRadius: BorderRadius.circular(12.0),
                    ),
                    focusedBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0xFF4B39EF),
                        width: 1.0,
                      ),
                      borderRadius: BorderRadius.circular(12.0),
                    ),
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0xFFE21C3D),
                        width: 1.0,
                      ),
                      borderRadius: BorderRadius.circular(12.0),
                    ),
                    focusedErrorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Color(0xFFE21C3D),
                        width: 1.0,
                      ),
                      borderRadius: BorderRadius.circular(12.0),
                    ),
                    filled: true,
                    fillColor: FlutterFlowTheme.of(context).lineColor,
                    suffixIcon: InkWell(
                      onTap: () => setState(
                        () => _model.passwordVisibility =
                            !_model.passwordVisibility,
                      ),
                      focusNode: FocusNode(skipTraversal: true),
                      child: Icon(
                        _model.passwordVisibility
                            ? Icons.visibility_outlined
                            : Icons.visibility_off_outlined,
                        size: 20.0,
                      ),
                    ),
                  ),
                  style: FlutterFlowTheme.of(context).bodyMedium,
                  validator:
                      _model.textController2Validator.asValidator(context),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(0.0, 30.0, 0.0, 0.0),
                child: FFButtonWidget(
```

```
                  onPressed: () async {
                    print('Button pressed ...');
                    Future<bool> res = getUsuario(correoController.text,
contrasenaController.text);
                    if(await res) {
                      showSnackbar(context, 'Sesión iniciada');
                    } else {
                      showSnackbar(context, 'El correo o la contraseña son
incorrectos');
                    }
                  },
                  text: 'Iniciar sesión',
                  options: FFButtonOptions(
                    width: 130.0,
                    height: 40.0,
                    padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                    iconPadding:
                        EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                    color: Color(0xFF1DF94C),
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                          fontSize: 20.0,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ),
              Padding(
                padding: EdgeInsetsDirectional.fromSTEB(0.0, 50.0, 0.0, 0.0),
                child: Text(
                  '¿Nuevo aquí?',
                  style: FlutterFlowTheme.of(context).bodyMedium.override(
                        fontFamily: 'Poppins',
                        fontSize: 15.0,
                      ),
                ),
              ),
              FFButtonWidget(
                onPressed: () async {
                  context.pushNamed('Registrarte');
                },
                text: 'Registrarte',
                options: FFButtonOptions(
                  width: 130.0,
                  height: 40.0,
                  padding: EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0,
0.0),
                  iconPadding:
                      EdgeInsetsDirectional.fromSTEB(0.0, 0.0, 0.0, 0.0),
                  color: Color(0xFF1DF94C),
```

```
                    textStyle:
FlutterFlowTheme.of(context).titleSmall.override(
                          fontFamily: 'Poppins',
                          color: Colors.black,
                        ),
                    borderSide: BorderSide(
                      color: Colors.transparent,
                      width: 1.0,
                    ),
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                ),
              ],
            ),
          ),
        ),
      );
  }
}
```

Función initState()
Se inicializa el estado de la página y de los controladores.

Función dispose()
Se crea el modelo de la página.

Función build()
Crea el widget que se visualiza de la página. Contiene un AppBar con el nombre de la tienda, también cuenta con un body que en su interior se encuentra varios textos y 2 TextFormField para que el usuario pueda ingresar sus datos, también se cuenta con 2 botones, uno para iniciar sesión que llama a la función “getUsuario()” y otro para mandarte a la página de registrarte.

Código flutter_flow_model.dart

Imports

```
import 'package:collection/collection.dart';
import 'package:flutter/material.dart';
import 'package:flutter/scheduler.dart';
import 'package:provider/provider.dart';
```

Paquetes y librerías que será utilizadas.

```
Widget wrapWithModel<T extends FlutterFlowModel>({
  required T model,
  required Widget child,
  required VoidCallback updateCallback,
  bool updateOnChange = false,
}) {
  // Set the component to optionally update the page on updates.
  model.setOnUpdate(
    onUpdate: updateCallback,
```

```
    updateOnChange: updateOnChange,
  );
  // Models for components within a page will be disposed by the page's
model,
  // so we don't want the component widget to dispose them until the page is
  // itself disposed.
  model.disposeOnWidgetDisposal = false;
  // Wrap in a Provider so that the model can be accessed by the component.
  return Provider<T>.value(
    value: model,
    child: child,
  );
}

T createModel<T extends FlutterFlowModel>(
  BuildContext context,
  T Function() defaultBuilder,
) {
  final model = context.read<T?>() ?? defaultBuilder();
  model._init(context);
  return model;
}

abstract class FlutterFlowModel {
  // Initialization methods
  bool _isInitialized = false;
  void initState(BuildContext context);
  void _init(BuildContext context) {
    if (!_isInitialized) {
      initState(context);
      _isInitialized = true;
    }
  }

  // Dispose methods
  // Whether to dispose this model when the corresponding widget is
  // disposed. By default this is true for pages and false for components,
  // as page/component models handle the disposal of their children.
  bool disposeOnWidgetDisposal = true;
  void dispose();
  void maybeDispose() {
    if (disposeOnWidgetDisposal) {
      dispose();
    }
  }

  // Whether to update the containing page / component on updates.
  bool updateOnChange = false;
  // Function to call when the model receives an update.
  VoidCallback _updateCallback = () {};
  void onUpdate() => updateOnChange ? _updateCallback() : () {};
  FlutterFlowModel setOnUpdate({
    bool updateOnChange = false,
    required VoidCallback onUpdate,
  }) =>
      this
        .._updateCallback = onUpdate
```

```
        ..updateOnChange = updateOnChange;
  // Update the containing page when this model received an update.
  void updatePage(VoidCallback callback) {
    callback();
    _updateCallback();
  }
}

class FlutterFlowDynamicModels<T extends FlutterFlowModel> {
  FlutterFlowDynamicModels(this.defaultBuilder);

  final T Function() defaultBuilder;
  final Map<String, T> _childrenModels = {};
  final Map<String, int> _childrenIndexes = {};
  Set<String>? _activeKeys;

  T getModel(String uniqueKey, int index) {
    _updateActiveKeys(uniqueKey);
    _childrenIndexes[uniqueKey] = index;
    return _childrenModels[uniqueKey] ??= defaultBuilder();
  }

  List<S> getValues<S>(S? Function(T) getValue) {
    return _childrenIndexes.entries
        // Sort keys by index.
        .sorted((a, b) => a.value.compareTo(b.value))
        .where((e) => _childrenModels[e.key] != null)
        // Map each model to the desired value and return as list. In order
        // to preserve index order, rather than removing null values we
provide
        // default values (for types with reasonable defaults).
        .map((e) => getValue(_childrenModels[e.key]!) ??
_getDefaultValue<S>()!)
        .toList();
  }

  S? getValueAtIndex<S>(int index, S Function(T) getValue) {
    final uniqueKey =
        _childrenIndexes.entries.firstWhereOrNull((e) => e.value ==
index)?.key;
    return getValueForKey(uniqueKey, getValue);
  }

  S? getValueForKey<S>(String? uniqueKey, S Function(T) getValue) {
    final model = _childrenModels[uniqueKey];
    return model != null ? getValue(model) : null;
  }

  void dispose() => _childrenModels.values.forEach((model) =>
model.dispose());

  void _updateActiveKeys(String uniqueKey) {
    final shouldResetActiveKeys = _activeKeys == null;
    _activeKeys ??= {};
    _activeKeys!.add(uniqueKey);

    if (shouldResetActiveKeys) {
```

```
      // Add a post-frame callback to remove and dispose of unused models
after
      // we're done building, then reset `_activeKeys` to null so we know to
do
      // this again next build.
      SchedulerBinding.instance.addPostFrameCallback((_) {
        _childrenIndexes.removeWhere((k, _) => !_activeKeys!.contains(k));
        _childrenModels.keys
            .toSet()
            .difference(_activeKeys!)
            // Remove and dispose of unused models since they are  not being
used
            // elsewhere and would not otherwise be disposed.
            .forEach((k) => _childrenModels.remove(k)?.dispose());
        _activeKeys = null;
      });
    }
  }
}

T? _getDefaultValue<T>() {
  switch (T) {
    case int:
      return 0 as T;
    case double:
      return 0.0 as T;
    case String:
      return '' as T;
    case bool:
      return false as T;
    default:
      return null as T;
  }
}

extension TextValidationExtensions on String? Function(BuildContext,
String?)? {
  String? Function(String?)? asValidator(BuildContext context) =>
      this != null ? (val) => this!(context, val) : null;
}
```

Código que se utiliza para crear los modelos, iniciar los modelos y cambiar los modelos.

Código flutter_flow_theme.dart

Imports

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

Paquetes y librerías que serán utilizadas.

```
const kThemeModeKey = '__theme_mode__';
SharedPreferences? _prefs;
```

```dart
abstract class FlutterFlowTheme {
  static Future initialize() async =>
      _prefs = await SharedPreferences.getInstance();
  static ThemeMode get themeMode {
    final darkMode = _prefs?.getBool(kThemeModeKey);
    return darkMode == null
        ? ThemeMode.system
        : darkMode
            ? ThemeMode.dark
            : ThemeMode.light;
  }

  static void saveThemeMode(ThemeMode mode) => mode == ThemeMode.system
      ? _prefs?.remove(kThemeModeKey)
      : _prefs?.setBool(kThemeModeKey, mode == ThemeMode.dark);

  static FlutterFlowTheme of(BuildContext context) {
    return Theme.of(context).brightness == Brightness.dark
        ? DarkModeTheme()
        : LightModeTheme();
  }

  @Deprecated('Use primary instead')
  Color get primaryColor => primary;
  @Deprecated('Use secondary instead')
  Color get secondaryColor => secondary;
  @Deprecated('Use tertiary instead')
  Color get tertiaryColor => tertiary;

  late Color primary;
  late Color secondary;
  late Color tertiary;
  late Color alternate;
  late Color primaryText;
  late Color secondaryText;
  late Color primaryBackground;
  late Color secondaryBackground;
  late Color accent1;
  late Color accent2;
  late Color accent3;
  late Color accent4;
  late Color success;
  late Color warning;
  late Color error;
  late Color info;

  late Color lineColor;
  late Color primaryBtnText;
  late Color customColor1;

  @Deprecated('Use displaySmallFamily instead')
  String get title1Family => displaySmallFamily;
  @Deprecated('Use displaySmall instead')
  TextStyle get title1 => typography.displaySmall;
  @Deprecated('Use headlineMediumFamily instead')
  String get title2Family => typography.headlineMediumFamily;
```

```
  @Deprecated('Use headlineMedium instead')
  TextStyle get title2 => typography.headlineMedium;
  @Deprecated('Use headlineSmallFamily instead')
  String get title3Family => typography.headlineSmallFamily;
  @Deprecated('Use headlineSmall instead')
  TextStyle get title3 => typography.headlineSmall;
  @Deprecated('Use titleMediumFamily instead')
  String get subtitle1Family => typography.titleMediumFamily;
  @Deprecated('Use titleMedium instead')
  TextStyle get subtitle1 => typography.titleMedium;
  @Deprecated('Use titleSmallFamily instead')
  String get subtitle2Family => typography.titleSmallFamily;
  @Deprecated('Use titleSmall instead')
  TextStyle get subtitle2 => typography.titleSmall;
  @Deprecated('Use bodyMediumFamily instead')
  String get bodyText1Family => typography.bodyMediumFamily;
  @Deprecated('Use bodyMedium instead')
  TextStyle get bodyText1 => typography.bodyMedium;
  @Deprecated('Use bodySmallFamily instead')
  String get bodyText2Family => typography.bodySmallFamily;
  @Deprecated('Use bodySmall instead')
  TextStyle get bodyText2 => typography.bodySmall;

  String get displayLargeFamily => typography.displayLargeFamily;
  TextStyle get displayLarge => typography.displayLarge;
  String get displayMediumFamily => typography.displayMediumFamily;
  TextStyle get displayMedium => typography.displayMedium;
  String get displaySmallFamily => typography.displaySmallFamily;
  TextStyle get displaySmall => typography.displaySmall;
  String get headlineLargeFamily => typography.headlineLargeFamily;
  TextStyle get headlineLarge => typography.headlineLarge;
  String get headlineMediumFamily => typography.headlineMediumFamily;
  TextStyle get headlineMedium => typography.headlineMedium;
  String get headlineSmallFamily => typography.headlineSmallFamily;
  TextStyle get headlineSmall => typography.headlineSmall;
  String get titleLargeFamily => typography.titleLargeFamily;
  TextStyle get titleLarge => typography.titleLarge;
  String get titleMediumFamily => typography.titleMediumFamily;
  TextStyle get titleMedium => typography.titleMedium;
  String get titleSmallFamily => typography.titleSmallFamily;
  TextStyle get titleSmall => typography.titleSmall;
  String get labelLargeFamily => typography.labelLargeFamily;
  TextStyle get labelLarge => typography.labelLarge;
  String get labelMediumFamily => typography.labelMediumFamily;
  TextStyle get labelMedium => typography.labelMedium;
  String get labelSmallFamily => typography.labelSmallFamily;
  TextStyle get labelSmall => typography.labelSmall;
  String get bodyLargeFamily => typography.bodyLargeFamily;
  TextStyle get bodyLarge => typography.bodyLarge;
  String get bodyMediumFamily => typography.bodyMediumFamily;
  TextStyle get bodyMedium => typography.bodyMedium;
  String get bodySmallFamily => typography.bodySmallFamily;
  TextStyle get bodySmall => typography.bodySmall;

  Typography get typography => ThemeTypography(this);
}
```

```
class LightModeTheme extends FlutterFlowTheme {
  @Deprecated('Use primary instead')
  Color get primaryColor => primary;
  @Deprecated('Use secondary instead')
  Color get secondaryColor => secondary;
  @Deprecated('Use tertiary instead')
  Color get tertiaryColor => tertiary;

  late Color primary = const Color(0xFF4B39EF);
  late Color secondary = const Color(0xFF39D2C0);
  late Color tertiary = const Color(0xFFEE8B60);
  late Color alternate = const Color(0xFFFF5963);
  late Color primaryText = const Color(0xFF101213);
  late Color secondaryText = const Color(0xFF57636C);
  late Color primaryBackground = const Color(0xFFF1F4F8);
  late Color secondaryBackground = const Color(0xFFFFFFFF);
  late Color accent1 = const Color(0xFF616161);
  late Color accent2 = const Color(0xFF757575);
  late Color accent3 = const Color(0xFFE0E0E0);
  late Color accent4 = const Color(0xFFEEEEEE);
  late Color success = const Color(0xFF04A24C);
  late Color warning = const Color(0xFFFCDC0C);
  late Color error = const Color(0xFFE21C3D);
  late Color info = const Color(0xFF1C4494);

  late Color lineColor = Color(0xFFE0E3E7);
  late Color primaryBtnText = Color(0xFFFFFFFF);
  late Color customColor1 = Color(0xFF2FB73C);
}

abstract class Typography {
  String get displayLargeFamily;
  TextStyle get displayLarge;
  String get displayMediumFamily;
  TextStyle get displayMedium;
  String get displaySmallFamily;
  TextStyle get displaySmall;
  String get headlineLargeFamily;
  TextStyle get headlineLarge;
  String get headlineMediumFamily;
  TextStyle get headlineMedium;
  String get headlineSmallFamily;
  TextStyle get headlineSmall;
  String get titleLargeFamily;
  TextStyle get titleLarge;
  String get titleMediumFamily;
  TextStyle get titleMedium;
  String get titleSmallFamily;
  TextStyle get titleSmall;
  String get labelLargeFamily;
  TextStyle get labelLarge;
  String get labelMediumFamily;
  TextStyle get labelMedium;
  String get labelSmallFamily;
  TextStyle get labelSmall;
  String get bodyLargeFamily;
  TextStyle get bodyLarge;
```

```
  String get bodyMediumFamily;
  TextStyle get bodyMedium;
  String get bodySmallFamily;
  TextStyle get bodySmall;
}

class ThemeTypography extends Typography {
  ThemeTypography(this.theme);

  final FlutterFlowTheme theme;

  String get displayLargeFamily => 'Poppins';
  TextStyle get displayLarge => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 57.0,
      );
  String get displayMediumFamily => 'Poppins';
  TextStyle get displayMedium => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 45.0,
      );
  String get displaySmallFamily => 'Poppins';
  TextStyle get displaySmall => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 36.0,
      );
  String get headlineLargeFamily => 'Poppins';
  TextStyle get headlineLarge => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 32.0,
      );
  String get headlineMediumFamily => 'Poppins';
  TextStyle get headlineMedium => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 28.0,
      );
  String get headlineSmallFamily => 'Poppins';
  TextStyle get headlineSmall => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
        fontWeight: FontWeight.normal,
        fontSize: 24.0,
      );
  String get titleLargeFamily => 'Poppins';
  TextStyle get titleLarge => GoogleFonts.getFont(
        'Poppins',
        color: theme.primaryText,
```

```
      fontWeight: FontWeight.w500,
      fontSize: 22.0,
    );
String get titleMediumFamily => 'Poppins';
TextStyle get titleMedium => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.w500,
      fontSize: 16.0,
    );
String get titleSmallFamily => 'Poppins';
TextStyle get titleSmall => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.w500,
      fontSize: 14.0,
    );
String get labelLargeFamily => 'Poppins';
TextStyle get labelLarge => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.w500,
      fontSize: 14.0,
    );
String get labelMediumFamily => 'Poppins';
TextStyle get labelMedium => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.w500,
      fontSize: 12.0,
    );
String get labelSmallFamily => 'Poppins';
TextStyle get labelSmall => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.w500,
      fontSize: 11.0,
    );
String get bodyLargeFamily => 'Poppins';
TextStyle get bodyLarge => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.normal,
      fontSize: 16.0,
    );
String get bodyMediumFamily => 'Poppins';
TextStyle get bodyMedium => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.normal,
      fontSize: 14.0,
    );
String get bodySmallFamily => 'Poppins';
TextStyle get bodySmall => GoogleFonts.getFont(
      'Poppins',
      color: theme.primaryText,
      fontWeight: FontWeight.normal,
```

```
        fontSize: 12.0,
      );
}

class DarkModeTheme extends FlutterFlowTheme {
  @Deprecated('Use primary instead')
  Color get primaryColor => primary;
  @Deprecated('Use secondary instead')
  Color get secondaryColor => secondary;
  @Deprecated('Use tertiary instead')
  Color get tertiaryColor => tertiary;

  late Color primary = const Color(0xFF4B39EF);
  late Color secondary = const Color(0xFF39D2C0);
  late Color tertiary = const Color(0xFFEE8B60);
  late Color alternate = const Color(0xFFFF5963);
  late Color primaryText = const Color(0xFFFFFFFF);
  late Color secondaryText = const Color(0xFF95A1AC);
  late Color primaryBackground = const Color(0xFF101213);
  late Color secondaryBackground = const Color(0xFF1A1F24);
  late Color accent1 = const Color(0xFFEEEEEE);
  late Color accent2 = const Color(0xFFE0E0E0);
  late Color accent3 = const Color(0xFF757575);
  late Color accent4 = const Color(0xFF616161);
  late Color success = const Color(0xFF04A24C);
  late Color warning = const Color(0xFFFCDC0C);
  late Color error = const Color(0xFFE21C3D);
  late Color info = const Color(0xFF1C4494);

  late Color lineColor = Color(0xFF22282F);
  late Color primaryBtnText = Color(0xFFFFFFFF);
  late Color customColor1 = Color(0xFF452FB7);
}

extension TextStyleHelper on TextStyle {
  TextStyle override({
    String? fontFamily,
    Color? color,
    double? fontSize,
    FontWeight? fontWeight,
    double? letterSpacing,
    FontStyle? fontStyle,
    bool useGoogleFonts = true,
    TextDecoration? decoration,
    double? lineHeight,
  }) =>
      useGoogleFonts
          ? GoogleFonts.getFont(
              fontFamily!,
              color: color ?? this.color,
              fontSize: fontSize ?? this.fontSize,
              letterSpacing: letterSpacing ?? this.letterSpacing,
              fontWeight: fontWeight ?? this.fontWeight,
              fontStyle: fontStyle ?? this.fontStyle,
              decoration: decoration,
              height: lineHeight,
            )
```

```
          : copyWith(
              fontFamily: fontFamily,
              color: color,
              fontSize: fontSize,
              letterSpacing: letterSpacing,
              fontWeight: fontWeight,
              fontStyle: fontStyle,
              decoration: decoration,
              height: lineHeight,
            );
}
```

Código empleado para utilizar en el tema de la aplicación.

Código flutter_flow_util.dart

Imports

```
import 'dart:io';
import 'package:flutter/foundation.dart' show kIsWeb;
import 'package:flutter/material.dart';
import 'package:from_css_color/from_css_color.dart';
import 'package:intl/intl.dart';
import 'package:json_path/json_path.dart';
import 'package:timeago/timeago.dart' as timeago;
import 'package:url_launcher/url_launcher.dart';
import '../main.dart';
export 'lat_lng.dart';
export 'place.dart';
export 'uploaded_file.dart';
export 'flutter_flow_model.dart';
export 'dart:math' show min, max;
export 'dart:typed_data' show Uint8List;
export 'dart:convert' show jsonEncode, jsonDecode;
export 'package:intl/intl.dart';
export 'package:page_transition/page_transition.dart';
export 'nav/nav.dart';
```

Paquetes y librerías que serán utilizadas.

```
T valueOrDefault<T>(T? value, T defaultValue) =>
    (value is String && value.isEmpty) || value == null ? defaultValue :
value;

String dateTimeFormat(String format, DateTime? dateTime, {String? locale}) {
  if (dateTime == null) {
    return '';
  }
  if (format == 'relative') {
    return timeago.format(dateTime, locale: locale);
  }
  return DateFormat(format, locale).format(dateTime);
}

Future launchURL(String url) async {
```

```
  var uri = Uri.parse(url).toString();
  try {
    await launch(uri);
  } catch (e) {
    throw 'Could not launch $uri: $e';
  }
}

Color colorFromCssString(String color, {Color? defaultColor}) {
  try {
    return fromCssColor(color);
  } catch (_) {}
  return defaultColor ?? Colors.black;
}

enum FormatType {
  decimal,
  percent,
  scientific,
  compact,
  compactLong,
  custom,
}

enum DecimalType {
  automatic,
  periodDecimal,
  commaDecimal,
}

String formatNumber(
  num? value, {
  required FormatType formatType,
  DecimalType? decimalType,
  String? currency,
  bool toLowerCase = false,
  String? format,
  String? locale,
}) {
  if (value == null) {
    return '';
  }
  var formattedValue = '';
  switch (formatType) {
    case FormatType.decimal:
      switch (decimalType!) {
        case DecimalType.automatic:
          formattedValue = NumberFormat.decimalPattern().format(value);
          break;
        case DecimalType.periodDecimal:
          formattedValue =
NumberFormat.decimalPattern('en_US').format(value);
          break;
        case DecimalType.commaDecimal:
          formattedValue =
NumberFormat.decimalPattern('es_PA').format(value);
          break;
```

```
      }
      break;
    case FormatType.percent:
      formattedValue = NumberFormat.percentPattern().format(value);
      break;
    case FormatType.scientific:
      formattedValue = NumberFormat.scientificPattern().format(value);
      if (toLowerCase) {
        formattedValue = formattedValue.toLowerCase();
      }
      break;
    case FormatType.compact:
      formattedValue = NumberFormat.compact().format(value);
      break;
    case FormatType.compactLong:
      formattedValue = NumberFormat.compactLong().format(value);
      break;
    case FormatType.custom:
      final hasLocale = locale != null && locale.isNotEmpty;
      formattedValue =
          NumberFormat(format, hasLocale ? locale : null).format(value);
  }

  if (formattedValue.isEmpty) {
    return value.toString();
  }

  if (currency != null) {
    final currencySymbol = currency.isNotEmpty
        ? currency
        : NumberFormat.simpleCurrency().format(0.0).substring(0, 1);
    formattedValue = '$currencySymbol$formattedValue';
  }

  return formattedValue;
}

DateTime get getCurrentTimestamp => DateTime.now();
DateTime dateTimeFromSecondsSinceEpoch(int seconds) {
  return DateTime.fromMillisecondsSinceEpoch(seconds * 1000);
}

extension DateTimeConversionExtension on DateTime {
  int get secondsSinceEpoch => (millisecondsSinceEpoch / 1000).round();
}

extension DateTimeComparisonOperators on DateTime {
  bool operator <(DateTime other) => isBefore(other);
  bool operator >(DateTime other) => isAfter(other);
  bool operator <=(DateTime other) => this < other ||
isAtSameMomentAs(other);
  bool operator >=(DateTime other) => this > other ||
isAtSameMomentAs(other);
}

dynamic getJsonField(
  dynamic response,
```

```
  String jsonPath, [
  bool isForList = false,
]) {
  final field = JsonPath(jsonPath).read(response);
  if (field.isEmpty) {
    return null;
  }
  if (field.length > 1) {
    return field.map((f) => f.value).toList();
  }
  final value = field.first.value;
  return isForList && value is! Iterable ? [value] : value;
}

Rect? getWidgetBoundingBox(BuildContext context) {
  try {
    final renderBox = context.findRenderObject() as RenderBox?;
    return renderBox!.localToGlobal(Offset.zero) & renderBox.size;
  } catch (_) {
    return null;
  }
}

bool get isAndroid => !kIsWeb && Platform.isAndroid;
bool get isiOS => !kIsWeb && Platform.isIOS;
bool get isWeb => kIsWeb;

const kBreakpointSmall = 479.0;
const kBreakpointMedium = 767.0;
const kBreakpointLarge = 991.0;
bool isMobileWidth(BuildContext context) =>
    MediaQuery.of(context).size.width < kBreakpointSmall;
bool responsiveVisibility({
  required BuildContext context,
  bool phone = true,
  bool tablet = true,
  bool tabletLandscape = true,
  bool desktop = true,
}) {
  final width = MediaQuery.of(context).size.width;
  if (width < kBreakpointSmall) {
    return phone;
  } else if (width < kBreakpointMedium) {
    return tablet;
  } else if (width < kBreakpointLarge) {
    return tabletLandscape;
  } else {
    return desktop;
  }
}

const kTextValidatorUsernameRegex = r'^[a-zA-Z][a-zA-Z0-9_-]{2,16}$';
const kTextValidatorEmailRegex =
    r"^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,253}[a-
zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,253}[a-zA-Z0-9])?)*$";
const kTextValidatorWebsiteRegex =
    r'(https?:\/\/)?(www\.)[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,10}\b([-a-
```

```
zA-Z0-9@:%_\+.~#?&//=]*)|(https?:\/\/)?(www\.)?(?!ww)[-a-zA-Z0-
9@:%._\+~#=]{2,256}\.[a-z]{2,10}\b([-a-zA-Z0-9@:%_\+.~#?&//=]*)';

extension FFTextEditingControllerExt on TextEditingController? {
  String get text => this == null ? '' : this!.text;
  set text(String newText) => this?.text = newText;
}

extension IterableExt<T> on Iterable<T> {
  List<S> mapIndexed<S>(S Function(int, T) func) => toList()
      .asMap()
      .map((index, value) => MapEntry(index, func(index, value)))
      .values
      .toList();
}

void setAppLanguage(BuildContext context, String language) =>
    MyApp.of(context).setLocale(language);

void setDarkModeSetting(BuildContext context, ThemeMode themeMode) =>
    MyApp.of(context).setThemeMode(themeMode);

void showSnackbar(
  BuildContext context,
  String message, {
  bool loading = false,
  int duration = 4,
}) {
  ScaffoldMessenger.of(context).hideCurrentSnackBar();
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Row(
        children: [
          if (loading)
            Padding(
              padding: EdgeInsetsDirectional.only(end: 10.0),
              child: Container(
                height: 20,
                width: 20,
                child: const CircularProgressIndicator(
                  color: Colors.white,
                ),
              ),
            ),
          Text(message),
        ],
      ),
      duration: Duration(seconds: duration),
    ),
  );
}

extension FFStringExt on String {
  String maybeHandleOverflow({int? maxChars, String replacement = ''}) =>
      maxChars != null && length > maxChars
          ? replaceRange(maxChars, null, replacement)
          : this;
```

```
}

extension ListFilterExt<T> on Iterable<T?> {
  List<T> get withoutNulls => where((s) => s != null).map((e) =>
e!).toList();
}
```

Código para obtener toda la información relacionada al dispositivo (hora y fecha, lenguaje, tema, etc.).

Código flutter_flow_widgets.dart

Imports

```
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:flutter/material.dart';
import 'package:auto_size_text/auto_size_text.dart';
```

Paquetes y librerías que serán utilizados.

```
class FFButtonOptions {
  const FFButtonOptions({
    this.textStyle,
    this.elevation,
    this.height,
    this.width,
    this.padding,
    this.color,
    this.disabledColor,
    this.disabledTextColor,
    this.splashColor,
    this.iconSize,
    this.iconColor,
    this.iconPadding,
    this.borderRadius,
    this.borderSide,
    this.hoverColor,
    this.hoverBorderSide,
    this.hoverTextColor,
    this.hoverElevation,
  });

  final TextStyle? textStyle;
  final double? elevation;
  final double? height;
  final double? width;
  final EdgeInsetsGeometry? padding;
  final Color? color;
  final Color? disabledColor;
  final Color? disabledTextColor;
  final Color? splashColor;
  final double? iconSize;
  final Color? iconColor;
  final EdgeInsetsGeometry? iconPadding;
  final BorderRadius? borderRadius;
```

```dart
  final BorderSide? borderSide;
  final Color? hoverColor;
  final BorderSide? hoverBorderSide;
  final Color? hoverTextColor;
  final double? hoverElevation;
}

class FFButtonWidget extends StatefulWidget {
  const FFButtonWidget({
    Key? key,
    required this.text,
    required this.onPressed,
    this.icon,
    this.iconData,
    required this.options,
    this.showLoadingIndicator = true,
  }) : super(key: key);

  final String text;
  final Widget? icon;
  final IconData? iconData;
  final Function()? onPressed;
  final FFButtonOptions options;
  final bool showLoadingIndicator;

  @override
  State<FFButtonWidget> createState() => _FFButtonWidgetState();
}

class _FFButtonWidgetState extends State<FFButtonWidget> {
  bool loading = false;

  @override
  Widget build(BuildContext context) {
    Widget textWidget = loading
        ? Center(
            child: Container(
              width: 23,
              height: 23,
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(
                  widget.options.textStyle!.color ?? Colors.white,
                ),
              ),
            ),
          )
        : AutoSizeText(
            widget.text,
            style: widget.options.textStyle?.withoutColor(),
            maxLines: 1,
            overflow: TextOverflow.ellipsis,
          );

    final onPressed = widget.onPressed != null
        ? (widget.showLoadingIndicator
            ? () async {
                if (loading) {
```

```
                return;
              }
              setState(() => loading = true);
              try {
                await widget.onPressed!();
              } finally {
                if (mounted) {
                  setState(() => loading = false);
                }
              }
            }
          : () => widget.onPressed!())
      : null;

  ButtonStyle style = ButtonStyle(
    shape: MaterialStateProperty.resolveWith<OutlinedBorder>(
      (states) {
        if (states.contains(MaterialState.hovered) &&
            widget.options.hoverBorderSide != null) {
          return RoundedRectangleBorder(
            borderRadius:
                widget.options.borderRadius ?? BorderRadius.circular(8),
            side: widget.options.hoverBorderSide!,
          );
        }
        return RoundedRectangleBorder(
          borderRadius:
              widget.options.borderRadius ?? BorderRadius.circular(8),
          side: widget.options.borderSide ?? BorderSide.none,
        );
      },
    ),
    foregroundColor: MaterialStateProperty.resolveWith<Color?>(
      (states) {
        if (states.contains(MaterialState.disabled) &&
            widget.options.disabledTextColor != null) {
          return widget.options.disabledTextColor;
        }
        if (states.contains(MaterialState.hovered) &&
            widget.options.hoverTextColor != null) {
          return widget.options.hoverTextColor;
        }
        return widget.options.textStyle?.color;
      },
    ),
    backgroundColor: MaterialStateProperty.resolveWith<Color?>(
      (states) {
        if (states.contains(MaterialState.disabled) &&
            widget.options.disabledColor != null) {
          return widget.options.disabledColor;
        }
        if (states.contains(MaterialState.hovered) &&
            widget.options.hoverColor != null) {
          return widget.options.hoverColor;
        }
        return widget.options.color;
      },
```

```
      ),
      overlayColor: MaterialStateProperty.resolveWith<Color?>((states) {
        if (states.contains(MaterialState.pressed)) {
          return widget.options.splashColor;
        }
        return null;
      }),
      padding: MaterialStateProperty.all(widget.options.padding ??
          const EdgeInsets.symmetric(horizontal: 12.0, vertical: 4.0)),
      elevation: MaterialStateProperty.resolveWith<double?>(
        (states) {
          if (states.contains(MaterialState.hovered) &&
              widget.options.hoverElevation != null) {
            return widget.options.hoverElevation!;
          }
          return widget.options.elevation;
        },
      ),
    );

    if (widget.icon != null || widget.iconData != null) {
      return Container(
        height: widget.options.height,
        width: widget.options.width,
        child: ElevatedButton.icon(
          icon: Padding(
            padding: widget.options.iconPadding ?? EdgeInsets.zero,
            child: widget.icon ??
                FaIcon(
                  widget.iconData,
                  size: widget.options.iconSize,
                  color: widget.options.iconColor ??
                      widget.options.textStyle!.color,
                ),
          ),
          label: textWidget,
          onPressed: onPressed,
          style: style,
        ),
      );
    }

    return Container(
      height: widget.options.height,
      width: widget.options.width,
      child: ElevatedButton(
        onPressed: onPressed,
        style: style,
        child: textWidget,
      ),
    );
  }
}

extension _WithoutColorExtension on TextStyle {
  TextStyle withoutColor() => TextStyle(
        inherit: inherit,
```

```
        color: null,
        backgroundColor: backgroundColor,
        fontSize: fontSize,
        fontWeight: fontWeight,
        fontStyle: fontStyle,
        letterSpacing: letterSpacing,
        wordSpacing: wordSpacing,
        textBaseline: textBaseline,
        height: height,
        leadingDistribution: leadingDistribution,
        locale: locale,
        foreground: foreground,
        background: background,
        shadows: shadows,
        fontFeatures: fontFeatures,
        decoration: decoration,
        decorationColor: decorationColor,
        decorationStyle: decorationStyle,
        decorationThickness: decorationThickness,
        debugLabel: debugLabel,
        fontFamily: fontFamily,
        fontFamilyFallback: fontFamilyFallback,
        // The _package field is private so unfortunately we can't set it
here,
        // but it's almost always unset anyway.
        // package: _package,
        overflow: overflow,
      );
}
```

Código donde se declaran todos los atributos y opciones que se le pueden asignar a las clases.

Código form_field_controller.dart

Import

```
import 'package:flutter/foundation.dart';
```

Paquete que será utilizado.

Clase FormFieldController<T> {}

```
class FormFieldController<T> extends ValueNotifier<T?> {
  FormFieldController(this.initialValue) : super(initialValue);

  final T? initialValue;

  void reset() => value = initialValue;
}
```

Clase para darle un valor inicial a los controladores.

Código internationalization.dart

Imports

```
import 'package:flutter/material.dart';
import 'package:flutter/foundation.dart';
import 'package:shared_preferences/shared_preferences.dart';
```

Paquetes que serán utilizados.

```
const _kLocaleStorageKey = '__locale_key__';

class FFLocalizations {
  FFLocalizations(this.locale);

  final Locale locale;

  static FFLocalizations of(BuildContext context) =>
      Localizations.of<FFLocalizations>(context, FFLocalizations)!;

  static List<String> languages() => ['en'];

  static late SharedPreferences _prefs;
  static Future initialize() async =>
      _prefs = await SharedPreferences.getInstance();
  static Future storeLocale(String locale) =>
      _prefs.setString(_kLocaleStorageKey, locale);
  static Locale? getStoredLocale() {
    final locale = _prefs.getString(_kLocaleStorageKey);
    return locale != null && locale.isNotEmpty ? createLocale(locale) : null;
  }

  String get languageCode => locale.toString();
  String? get languageShortCode =>
      _languagesWithShortCode.contains(locale.toString())
          ? '${locale.toString()}_short'
          : null;
  int get languageIndex => languages().contains(languageCode)
      ? languages().indexOf(languageCode)
      : 0;

  String getText(String key) =>
      (kTranslationsMap[key] ?? {})[locale.toString()] ?? '';

  String getVariableText({
    String? enText = '',
  }) =>
      [enText][languageIndex] ?? '';

  static const Set<String> _languagesWithShortCode = {
    'ar',
    'az',
    'ca',
    'cs',
    'da',
    'de',
```

```
    'dv',
    'en',
    'es',
    'et',
    'fi',
    'fr',
    'gr',
    'he',
    'hi',
    'hu',
    'it',
    'km',
    'ku',
    'mn',
    'ms',
    'no',
    'pt',
    'ro',
    'ru',
    'rw',
    'sv',
    'th',
    'uk',
    'vi',
  };
}

class FFLocalizationsDelegate extends LocalizationsDelegate<FFLocalizations>
{
  const FFLocalizationsDelegate();

  @override
  bool isSupported(Locale locale) {
    final language = locale.toString();
    return FFLocalizations.languages().contains(
      language.endsWith('_')
          ? language.substring(0, language.length - 1)
          : language,
    );
  }

  @override
  Future<FFLocalizations> load(Locale locale) =>
      SynchronousFuture<FFLocalizations>(FFLocalizations(locale));

  @override
  bool shouldReload(FFLocalizationsDelegate old) => false;
}

Locale createLocale(String language) => language.contains('_')
    ? Locale.fromSubtags(
        languageCode: language.split('_').first,
        scriptCode: language.split('_').last,
      )
    : Locale(language);
```

```
final kTranslationsMap =
    <Map<String, Map<String, String>>>[].reduce((a, b) => a..addAll(b));
```

Código para seleccionar el lenguaje (cambiar) según la nación.

Código lat_lng.dart

```
class LatLng {
  const LatLng(this.latitude, this.longitude);
  final double latitude;
  final double longitude;

  @override
  String toString() => 'LatLng(lat: $latitude, lng: $longitude)';

  String serialize() => '$latitude,$longitude';

  @override
  int get hashCode => latitude.hashCode + longitude.hashCode;

  @override
  bool operator ==(other) =>
      other is LatLng &&
      latitude == other.latitude &&
      longitude == other.longitude;
}
```

Código para obtener la latitud y la longitud del dispositivo.

Código place.dart

Import

```
import 'lat_lng.dart';
```

Se importa la clase anterior.

```
class FFPlace {
  const FFPlace({
    this.latLng = const LatLng(0.0, 0.0),
    this.name = '',
    this.address = '',
    this.city = '',
    this.state = '',
    this.country = '',
    this.zipCode = '',
  });

  final LatLng latLng;
  final String name;
  final String address;
  final String city;
  final String state;
  final String country;
```

```
  final String zipCode;

  @override
  String toString() => '''FFPlace(
        latLng: $latLng,
        name: $name,
        address: $address,
        city: $city,
        state: $state,
        country: $country,
        zipCode: $zipCode,
      )''';

  @override
  int get hashCode => latLng.hashCode;

  @override
  bool operator ==(other) =>
      other is FFPlace &&
      latLng == other.latLng &&
      name == other.name &&
      address == other.address &&
      city == other.city &&
      state == other.state &&
      country == other.country &&
      zipCode == other.zipCode;
}
```

Código para almacenar la posición (información) del dispositivo.

Código uploaded_file.dart

Imports

```
import 'dart:convert';
import 'dart:typed_data' show Uint8List;
```

Paquetes que se ocuparán.

Clase FFUploadedFile {}

```
class FFUploadedFile {
  const FFUploadedFile({
    this.name,
    this.bytes,
    this.height,
    this.width,
    this.blurHash,
  });

  final String? name;
  final Uint8List? bytes;
  final double? height;
  final double? width;
  final String? blurHash;
```

```
  @override
  String toString() =>
      'FFUploadedFile(name: $name, bytes: ${bytes?.length ?? 0}, height:
$height, width: $width, blurHash: $blurHash)';

  String serialize() => jsonEncode(
        {
          'name': name,
          'bytes': bytes,
          'height': height,
          'width': width,
          'blurHash': blurHash,
        },
      );

  static FFUploadedFile deserialize(String val) {
    final serializedData = jsonDecode(val) as Map<String, dynamic>;
    final data = {
      'name': serializedData['name'] ?? '',
      'bytes': serializedData['bytes'] ?? Uint8List.fromList([]),
      'height': serializedData['height'],
      'width': serializedData['width'],
      'blurHash': serializedData['blurHash'],
    };
    return FFUploadedFile(
      name: data['name'] as String,
      bytes: data['bytes'] as Uint8List,
      height: data['height'] as double?,
      width: data['width'] as double?,
      blurHash: data['blurHash'] as String?,
    );
  }

  @override
  int get hashCode => Object.hash(name, bytes, height, width, blurHash);

  @override
  bool operator ==(other) =>
      other is FFUploadedFile &&
      name == other.name &&
      bytes == other.bytes &&
      height == other.height &&
      width == other.width &&
      blurHash == other.blurHash;
}
```

Código para cargar archivos.

Código nav.dart

Imports
```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:page_transition/page_transition.dart';
```

```dart
import '../../index.dart';
import '../../main.dart';
import 'serialization_util.dart';
export 'package:go_router/go_router.dart';
export 'serialization_util.dart';
```

Paquetes y librerías que serán utilizadas.

```dart
const kTransitionInfoKey = '__transition_info__';

class AppStateNotifier extends ChangeNotifier {
  bool showSplashImage = true;

  void stopShowingSplashImage() {
    showSplashImage = false;
    notifyListeners();
  }
}

GoRouter createRouter(AppStateNotifier appStateNotifier) => GoRouter(
      initialLocation: '/',
      debugLogDiagnostics: true,
      refreshListenable: appStateNotifier,
      errorBuilder: (context, _) => NavBarPage(),
      routes: [
        FFRoute(
          name: '_initialize',
          path: '/',
          builder: (context, _) => NavBarPage(),
        ),
        FFRoute(
          name: 'HomePage',
          path: '/homePage',
          builder: (context, params) => params.isEmpty
              ? NavBarPage(initialPage: 'HomePage')
              : HomePageWidget(),
        ),
        FFRoute(
          name: 'AcercaDe',
          path: '/acercaDe',
          builder: (context, params) => params.isEmpty
              ? NavBarPage(initialPage: 'AcercaDe')
              : AcercaDeWidget(),
        ),
        FFRoute(
          name: 'Productos',
          path: '/productos',
          builder: (context, params) => params.isEmpty
              ? NavBarPage(initialPage: 'Productos')
              : ProductosWidget(),
        ),
        FFRoute(
          name: 'Carrito',
          path: '/carrito',
          builder: (context, params) => params.isEmpty
              ? NavBarPage(initialPage: 'Carrito')
```

```
            : CarritoWidget(),
      ),
      FFRoute(
        name: 'Sesion',
        path: '/sesion',
        builder: (context, params) => params.isEmpty
            ? NavBarPage(initialPage: 'Sesion')
            : SesionWidget(),
      ),
      FFRoute(
        name: 'Dulces',
        path: '/dulces',
        builder: (context, params) => DulcesWidget(),
      ),
      FFRoute(
        name: 'Chocolates',
        path: '/chocolates',
        builder: (context, params) => ChocolatesWidget(),
      ),
      FFRoute(
        name: 'Frituras',
        path: '/frituras',
        builder: (context, params) => FriturasWidget(),
      ),
      FFRoute(
        name: 'PlasticosYDesechables',
        path: '/plasticosYDesechables',
        builder: (context, params) => PlasticosYDesechablesWidget(),
      ),
      FFRoute(
        name: 'Registrarte',
        path: '/registrarte',
        builder: (context, params) => RegistrarteWidget(),
      ),
      FFRoute(
        name: 'CompraRealizada',
        path: '/compraRealizada',
        builder: (context, params) => CompraRealizadaWidget(),
      )
    ].map((r) => r.toRoute(appStateNotifier)).toList(),
    urlPathStrategy: UrlPathStrategy.path,
  );

extension NavParamExtensions on Map<String, String?> {
  Map<String, String> get withoutNulls => Map.fromEntries(
        entries
            .where((e) => e.value != null)
            .map((e) => MapEntry(e.key, e.value!)),
      );
}

extension NavigationExtensions on BuildContext {
  void safePop() {
    // If there is only one route on the stack, navigate to the initial
    // page instead of popping.
    if (GoRouter.of(this).routerDelegate.matches.length <= 1) {
      go('/');
```

```
    } else {
      pop();
    }
  }
}

extension _GoRouterStateExtensions on GoRouterState {
  Map<String, dynamic> get extraMap =>
      extra != null ? extra as Map<String, dynamic> : {};
  Map<String, dynamic> get allParams => <String, dynamic>{}
    ..addAll(params)
    ..addAll(queryParams)
    ..addAll(extraMap);
  TransitionInfo get transitionInfo =>
extraMap.containsKey(kTransitionInfoKey)
      ? extraMap[kTransitionInfoKey] as TransitionInfo
      : TransitionInfo.appDefault();
}

class FFParameters {
  FFParameters(this.state, [this.asyncParams = const {}]);

  final GoRouterState state;
  final Map<String, Future<dynamic> Function(String)> asyncParams;

  Map<String, dynamic> futureParamValues = {};

  // Parameters are empty if the params map is empty or if the only parameter
  // present is the special extra parameter reserved for the transition info.
  bool get isEmpty =>
      state.allParams.isEmpty ||
      (state.extraMap.length == 1 &&
          state.extraMap.containsKey(kTransitionInfoKey));
  bool isAsyncParam(MapEntry<String, dynamic> param) =>
      asyncParams.containsKey(param.key) && param.value is String;
  bool get hasFutures => state.allParams.entries.any(isAsyncParam);
  Future<bool> completeFutures() => Future.wait(
        state.allParams.entries.where(isAsyncParam).map(
          (param) async {
            final doc = await asyncParams[param.key]!(param.value)
                .onError((_, __) => null);
            if (doc != null) {
              futureParamValues[param.key] = doc;
              return true;
            }
            return false;
          },
        ),
      ).onError((_, __) => [false]).then((v) => v.every((e) => e));

  dynamic getParam<T>(
    String paramName,
    ParamType type, [
    bool isList = false,
  ]) {
    if (futureParamValues.containsKey(paramName)) {
      return futureParamValues[paramName];
```

```
    }
    if (!state.allParams.containsKey(paramName)) {
      return null;
    }
    final param = state.allParams[paramName];
    // Got parameter from `extras`, so just directly return it.
    if (param is! String) {
      return param;
    }
    // Return serialized value.
    return deserializeParam<T>(
      param,
      type,
      isList,
    );
  }
}

class FFRoute {
  const FFRoute({
    required this.name,
    required this.path,
    required this.builder,
    this.requireAuth = false,
    this.asyncParams = const {},
    this.routes = const [],
  });

  final String name;
  final String path;
  final bool requireAuth;
  final Map<String, Future<dynamic> Function(String)> asyncParams;
  final Widget Function(BuildContext, FFParameters) builder;
  final List<GoRoute> routes;

  GoRoute toRoute(AppStateNotifier appStateNotifier) => GoRoute(
        name: name,
        path: path,
        pageBuilder: (context, state) {
          final ffParams = FFParameters(state, asyncParams);
          final page = ffParams.hasFutures
              ? FutureBuilder(
                  future: ffParams.completeFutures(),
                  builder: (context, _) => builder(context, ffParams),
                )
              : builder(context, ffParams);
          final child = page;

          final transitionInfo = state.transitionInfo;
          return transitionInfo.hasTransition
              ? CustomTransitionPage(
                  key: state.pageKey,
                  child: child,
                  transitionDuration: transitionInfo.duration,
                  transitionsBuilder: PageTransition(
                    type: transitionInfo.transitionType,
                    duration: transitionInfo.duration,
```

```
                    reverseDuration: transitionInfo.duration,
                    alignment: transitionInfo.alignment,
                    child: child,
                  ).transitionsBuilder,
                )
              : MaterialPage(key: state.pageKey, child: child);
        },
        routes: routes,
      );
}

class TransitionInfo {
  const TransitionInfo({
    required this.hasTransition,
    this.transitionType = PageTransitionType.fade,
    this.duration = const Duration(milliseconds: 300),
    this.alignment,
  });

  final bool hasTransition;
  final PageTransitionType transitionType;
  final Duration duration;
  final Alignment? alignment;

  static TransitionInfo appDefault() => TransitionInfo(hasTransition: false);
}
```

Código para crear las rutas, parámetros y las transiciones que tiene el navbar en cada una de sus pestañas.

Código serialization_util.dart

Imports

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:from_css_color/from_css_color.dart';
import '../../flutter_flow/lat_lng.dart';
import '../../flutter_flow/place.dart';
import '../../flutter_flow/uploaded_file.dart';
```

Paquetes y librerías que serán utilizadas.

```
/// SERIALIZATION HELPERS

String dateTimeRangeToString(DateTimeRange dateTimeRange) {
  final startStr = dateTimeRange.start.millisecondsSinceEpoch.toString();
  final endStr = dateTimeRange.end.millisecondsSinceEpoch.toString();
  return '$startStr|$endStr';
}

String placeToString(FFPlace place) => jsonEncode({
      'latLng': place.latLng.serialize(),
      'name': place.name,
      'address': place.address,
```

```
      'city': place.city,
      'state': place.state,
      'country': place.country,
      'zipCode': place.zipCode,
    });

String uploadedFileToString(FFUploadedFile uploadedFile) =>
    uploadedFile.serialize();

String? serializeParam(
  dynamic param,
  ParamType paramType, [
  bool isList = false,
]) {
  try {
    if (param == null) {
      return null;
    }
    if (isList) {
      final serializedValues = (param as Iterable)
          .map((p) => serializeParam(p, paramType, false))
          .where((p) => p != null)
          .map((p) => p!)
          .toList();
      return json.encode(serializedValues);
    }
    switch (paramType) {
      case ParamType.int:
        return param.toString();
      case ParamType.double:
        return param.toString();
      case ParamType.String:
        return param;
      case ParamType.bool:
        return param ? 'true' : 'false';
      case ParamType.DateTime:
        return (param as DateTime).millisecondsSinceEpoch.toString();
      case ParamType.DateTimeRange:
        return dateTimeRangeToString(param as DateTimeRange);
      case ParamType.LatLng:
        return (param as LatLng).serialize();
      case ParamType.Color:
        return (param as Color).toCssString();
      case ParamType.FFPlace:
        return placeToString(param as FFPlace);
      case ParamType.FFUploadedFile:
        return uploadedFileToString(param as FFUploadedFile);
      case ParamType.JSON:
        return json.encode(param);

      default:
        return null;
    }
  } catch (e) {
    print('Error serializing parameter: $e');
    return null;
  }
```

```
}

/// END SERIALIZATION HELPERS

/// DESERIALIZATION HELPERS

DateTimeRange? dateTimeRangeFromString(String dateTimeRangeStr) {
  final pieces = dateTimeRangeStr.split('|');
  if (pieces.length != 2) {
    return null;
  }
  return DateTimeRange(
    start: DateTime.fromMillisecondsSinceEpoch(int.parse(pieces.first)),
    end: DateTime.fromMillisecondsSinceEpoch(int.parse(pieces.last)),
  );
}

LatLng? latLngFromString(String latLngStr) {
  final pieces = latLngStr.split(',');
  if (pieces.length != 2) {
    return null;
  }
  return LatLng(
    double.parse(pieces.first.trim()),
    double.parse(pieces.last.trim()),
  );
}

FFPlace placeFromString(String placeStr) {
  final serializedData = jsonDecode(placeStr) as Map<String, dynamic>;
  final data = {
    'latLng': serializedData.containsKey('latLng')
        ? latLngFromString(serializedData['latLng'] as String)
        : const LatLng(0.0, 0.0),
    'name': serializedData['name'] ?? '',
    'address': serializedData['address'] ?? '',
    'city': serializedData['city'] ?? '',
    'state': serializedData['state'] ?? '',
    'country': serializedData['country'] ?? '',
    'zipCode': serializedData['zipCode'] ?? '',
  };
  return FFPlace(
    latLng: data['latLng'] as LatLng,
    name: data['name'] as String,
    address: data['address'] as String,
    city: data['city'] as String,
    state: data['state'] as String,
    country: data['country'] as String,
    zipCode: data['zipCode'] as String,
  );
}

FFUploadedFile uploadedFileFromString(String uploadedFileStr) =>
    FFUploadedFile.deserialize(uploadedFileStr);

enum ParamType {
  int,
```

```
  double,
  String,
  bool,
  DateTime,
  DateTimeRange,
  LatLng,
  Color,
  FFPlace,
  FFUploadedFile,
  JSON,
}

dynamic deserializeParam<T>(
  String? param,
  ParamType paramType,
  bool isList,
) {
  try {
    if (param == null) {
      return null;
    }
    if (isList) {
      final paramValues = json.decode(param);
      if (paramValues is! Iterable || paramValues.isEmpty) {
        return null;
      }
      return paramValues
          .where((p) => p is String)
          .map((p) => p as String)
          .map((p) => deserializeParam<T>(p, paramType, false))
          .where((p) => p != null)
          .map((p) => p! as T)
          .toList();
    }
    switch (paramType) {
      case ParamType.int:
        return int.tryParse(param);
      case ParamType.double:
        return double.tryParse(param);
      case ParamType.String:
        return param;
      case ParamType.bool:
        return param == 'true';
      case ParamType.DateTime:
        final milliseconds = int.tryParse(param);
        return milliseconds != null
            ? DateTime.fromMillisecondsSinceEpoch(milliseconds)
            : null;
      case ParamType.DateTimeRange:
        return dateTimeRangeFromString(param);
      case ParamType.LatLng:
        return latLngFromString(param);
      case ParamType.Color:
        return fromCssColor(param);
      case ParamType.FFPlace:
        return placeFromString(param);
      case ParamType.FFUploadedFile:
```

```
        return uploadedFileFromString(param);
      case ParamType.JSON:
        return json.decode(param);

      default:
        return null;
    }
  } catch (e) {
    print('Error deserializing parameter: $e');
    return null;
  }
}
```

Código para la información del lugar, fecha y hora del dispositivo y la carga de archivos.