Flutter

MINI APUNTES FLUTTER 2

Desarrollo de Interfaces

13 DE NOVIEMBRE DE 2024

ERIK AMO TOQUERO

Contenido

Introducción a la Navegación

La navegación en Flutter consiste en la transición de distintas pantallas. Hay que tener en cuenta que **cada pantalla es un objeto diferente**.

¿Para qué se usa esto? En un proyecto un poco grande, normalmente va a haber más de una pantalla, y no es plato de buen gusto hacer todo en una misma pantalla y cambiar visibilidades, solapándose todo con todo y tener un popurrí de cosas sin sentido.

Hay varias formas de realizar la navegación, las cuales se van a tratar a lo largo del documento.

Navegación por pantallas

Es la más sencilla: consiste en el cambio de pantalla principal por medio de un widget que, al accionarse, llama a un Navegador. Este navegador usa el contexto principal de la aplicación para moverse entre las distintas pantallas, pasándolas dicho contexto para crearlas.

El Navegador ordena el acceso a las pantallas en LIFO (Last-In-First-Out), y se manejan de la siguiente forma:

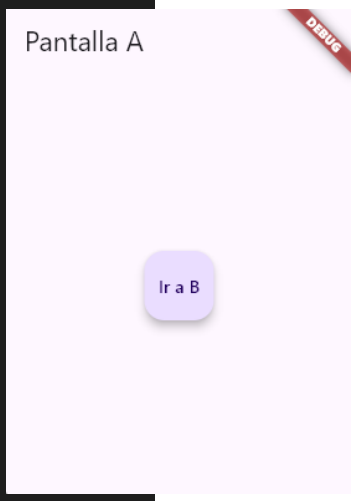
- **Push:** Añade una nueva pantalla
- **Pop:** Quita la pantalla actual y vuelve en la anterior

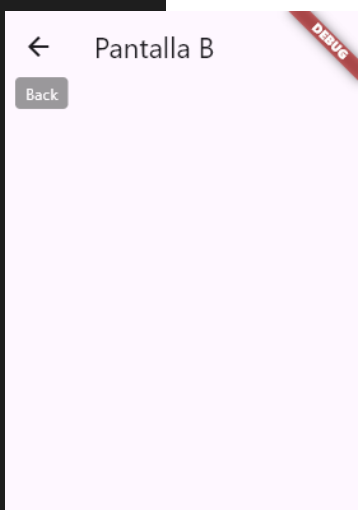
Métodos de Navegación

Navigator.push() – Introducir objeto

Este utiliza objetos, además de un objeto de tipo MaterialPageRoute, encargado de hacer la ruta a la nueva pantalla.

```
class MainApp extends StatelessWidget {
  const MainApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: PantallaA(), //IMPORTANTE ESTO
    );
  }
}
class PantallaA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla A"),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a B"),
          onPressed: (){
          Navigator.push(context, MaterialPageRoute(builder: (context) =>
PantallaB()));
        }),
      ),
    );
  }
}
class PantallaB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla B"),
      ),
    );
  }
}
```





IMPORTANTE: En la clase principal, el home de MaterialApp solo debe tener un objeto. De no ser así, dará un error relacionado a que el contexto no tiene navegadores.

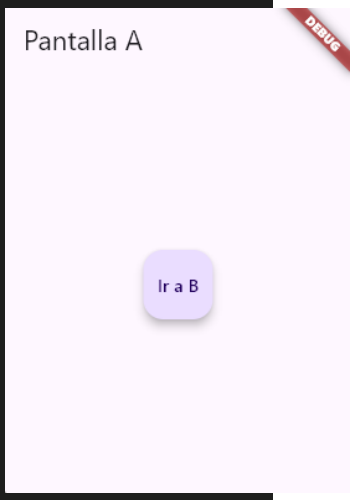
Navigator.pushNamed() – Introducir ruta

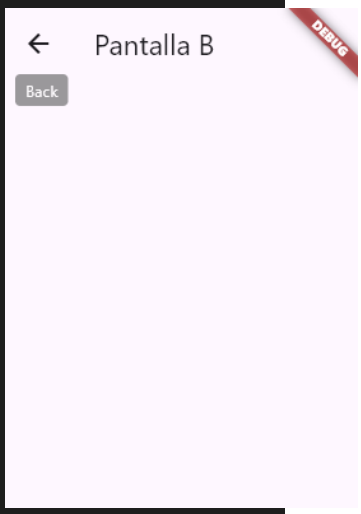
La estructura de la aplicación principal cambia un poco, ya que MaterialApp pierde el atributo “home” y pasa a tener las **rutas**, Strings que acceden a objetos. Navigator.pushNamed() accede al string de la ruta que se marque por parámetro.

IMPORTANTE: “/” es la pantalla principal (home) del proyecto. Si esta existe y el usuario mete un atributo home en MaterialApp, el proyecto sacará una excepción

```
class MainApp extends StatelessWidget {
  const MainApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      routes: {
        "/": (context) => PantallaA(),
        "/b": (context) => PantallaB(),
      },
    );
  }
}
class PantallaA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla A"),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a B"),
          onPressed: (){
            Navigator.pushNamed(context, "/b");
        }),
      ),
    );
  }

}
class PantallaB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla B"),
      ),
    );
  }
}
```

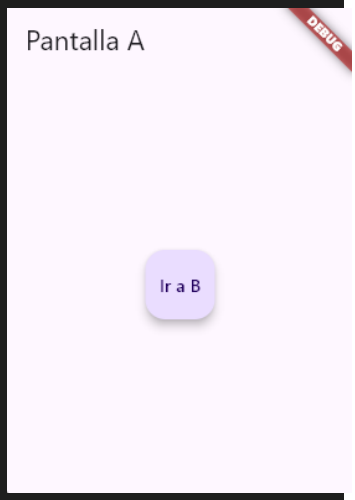


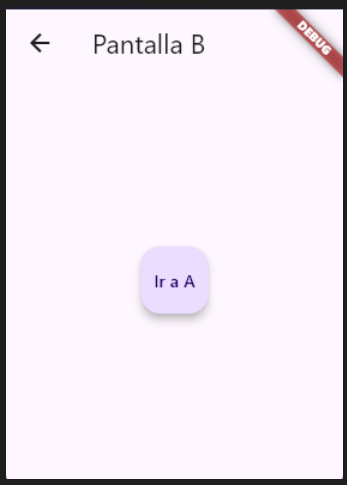


Navigator.pop() – Volver a la anterior

Este método no distingue de si se ha hecho el push por objeto o por rutas, se hace exactamente de la misma forma en ambas.

```
class MainApp extends StatelessWidget {
  const MainApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      routes: {
        "/": (context) => PantallaA(),
        "/b": (context) => PantallaB(),
      },
    );
  }
}
class PantallaA extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla A"),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a B"),
          onPressed: (){
            Navigator.pushNamed(context, "/b");
        }),
      ),
    );
  }}
class PantallaB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla B"),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a A"),
          onPressed: (){
            Navigator.pop(context);
        }
        ))
    );
  }}
```





Navegación sin datos

La navegación sin datos es simplemente hacer un push o un pop, pasando los objetos o el nombre de ruta como único requisito. Es tal cual como se han hecho los ejemplos de antes.

Navegación con datos

Se pueden pasar datos en el proceso de push y pop, **siendo el push para pasar el valor a la nueva pantalla o el pop para pasar los datos de la nueva a la anterior cuando se cierre.**

IMPORTANTE: SE RECOMIENDA USAR ESTADOS PARA VER LOS CAMBIOS, YA QUE LOS DATOS PUEDEN CAMBIAR (no sabia como poner esto)

Push pasando datos:

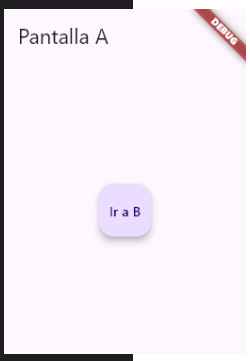
- **Pantalla 1:** Se meten los datos al final del parámetro en el push, tanto si es Named como si es normal,
- **Pantalla 2:** Recibe los parámetros y hace un cambio de estado con **ModalRoute**

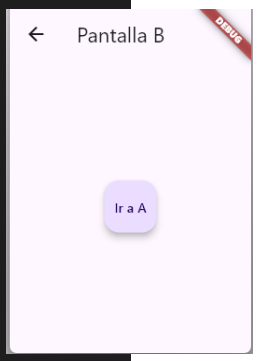
```dart
//En el push del objeto principal:
Navigator.pushNamed(context, "/b", arguments: "Pantalla B");
//PantallaB: Uso de ModalRoute parseándolo al tipo de argumento que usa
class PantallaB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    String text = ModalRoute.of(context)!.settings.arguments as String;
    return Scaffold(
      appBar: AppBar(
        title: Text(text),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a A"),
          onPressed: (){
            Navigator.pop(context);
        }
        ))
    );
  }
}
```

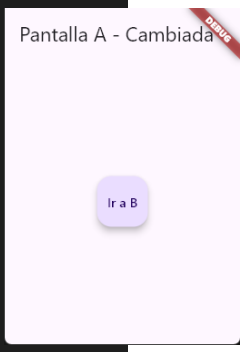
Pop pasando datos:

- **Pantalla 1:** Espera una respuesta con await. La respuesta será del tipo que nos interesa
- **Pantalla 2:** Hace su recorrido normal y corriente, pero pasa como argumentos los datos que quiera pasar

```dart
class PantallaA extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return EstadoA();
  }
}
class EstadoA extends State<PantallaA>{
  String titulo = "Pantalla A";
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(titulo),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a B"),
          onPressed: () async {
            var txt = await Navigator.pushNamed(context, "/b");
            setState(() {titulo = txt.toString();});
        }),
      ),
    );
  }
}
class PantallaB extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Pantalla B"),
      ),
      body: Center(
        child: FloatingActionButton(
          child: Text("Ir a A"),
          onPressed: (){
            Navigator.pop(context, "Pantalla A - Cambiada");
        }
        ))
    );
  }
```



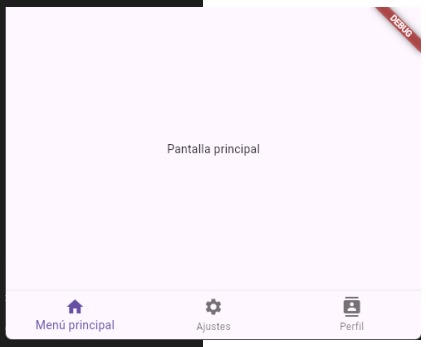


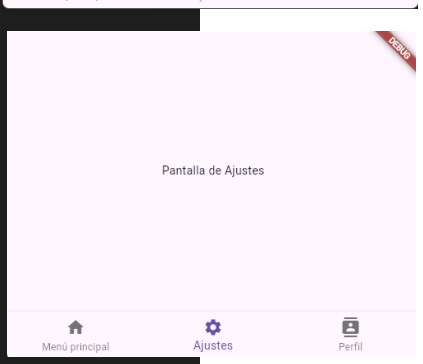


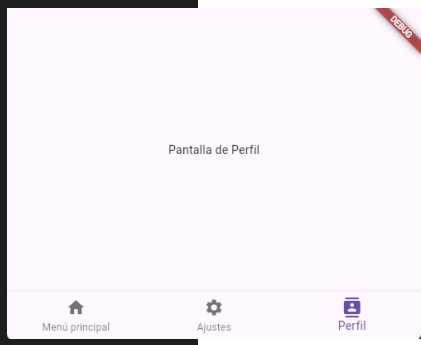
Navegación inferior

La navegación inferior utiliza el widget BottomNavigationBar, el cual (Valga la redundancia), guarda los “botones” de los menús, tantos “botones” como pantallas queramos asignar al menú inferior.
Lo llamo “botones” porque son BottomNavigationBarItem, ítems que pueden ser pulsados y tienen guardado un índice con su posición, ya que se guardan en una lista.
Una buena práctica de la navegación inferior es guardar en un switch en el body cada pantalla en función del índice del botón que queramos. (Tabular código bien si se pega, tenía poco espacio)

```
class MainState extends State<MainApp> {
  List<BottomNavigationBarItem> listaBotones = List.of([ //Lista de "botones"
    const BottomNavigationBarItem(
        icon: Icon(Icons.home), label: "Menú principal"),
    const BottomNavigationBarItem(icon: Icon(Icons.settings), label: "Ajustes"),
    const BottomNavigationBarItem(icon: Icon(Icons.contacts), label: "Perfil")]);
  int currentIndex = 0;
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          bottomNavigationBar: BottomNavigationBar(
            items: listaBotones,
            currentIndex: currentIndex,
            onTap: (value) {setState(() {currentIndex = value;});},),
          body: switch (currentIndex) {
            0 => MainScreen(),
            1 => SettingsScreen(),
            _ => ProfileScreen()
          }),);}}
class MainScreen extends StatelessWidget {
  const MainScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(child: Text("Pantalla principal"));}}
class ProfileScreen extends StatelessWidget {
  const ProfileScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(child: Text("Pantalla de Perfil"));}}
class SettingsScreen extends StatelessWidget {
  const SettingsScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(child: Text("Pantalla de Ajustes"),);
}}
```







Navegación por pestañas

La navegación por pestañas es la navegación en una misma pantalla donde cada “subpantalla” se encuentra abierta y accesible por el menú de tarjetas (TabBar). La TabBar se encuentra en el AppBar, y se compone de una serie de pestañas (Tab). Luego, en el body del Scaffold, aparece de principal TabBarView, un Widget que coge la lista del TabBar y da una pantalla a cada pestaña.

IMPORTANTE: Todo esto tiene que ir dentro de un DefaultTabController para decir cuántas pestañas van a aparecer en la interfaz. La estructura sería MaterialApp-DefaultTabController-Scaffold
(Tabular cuando se use todo esto, falta de sitio)

```
class MainState extends State<MainApp> {
  List<Tab> listaPestanas = List.of([ //Lista con todas las pestañas
    const Tab(icon: Icon(Icons.home)),const Tab(icon: Icon(Icons.settings)),const
Tab(icon: Icon(Icons.contacts))]);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: DefaultTabController(
            length: 3,
            child: Scaffold(
              appBar: AppBar(
                bottom: TabBar(tabs: listaPestanas),
                title: Text("Pestañas"),),
              body: TabBarView(
                  children: [MainScreen(), SettingsScreen(), ProfileScreen()]),
            )));}}
class MainScreen extends StatelessWidget {
  const MainScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text("Pantalla principal"),
    );}}
class ProfileScreen extends StatelessWidget {
  const ProfileScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text("Pantalla de Perfil"),);}}
class SettingsScreen extends StatelessWidget {
  const SettingsScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text("Pantalla de Ajustes"),);}}
```