Flutter

Navegando entre pantallas.

Carmelo Villegas Cruz

Navegando entre pantallas

Hasta ahora solo hemos vistos App donde disponemos de una única pantalla pero esto no es lo más habitual así que es hora de adentrarnos en la navegación entre pantallas.

Para la configuración y definición del routing en Flutter vamos a utilizar principalmente dos Widgets, **Navigator** y **Route** que iremos introduciendo en las posteriores diapositivas.

Navigator & Route

Navigator

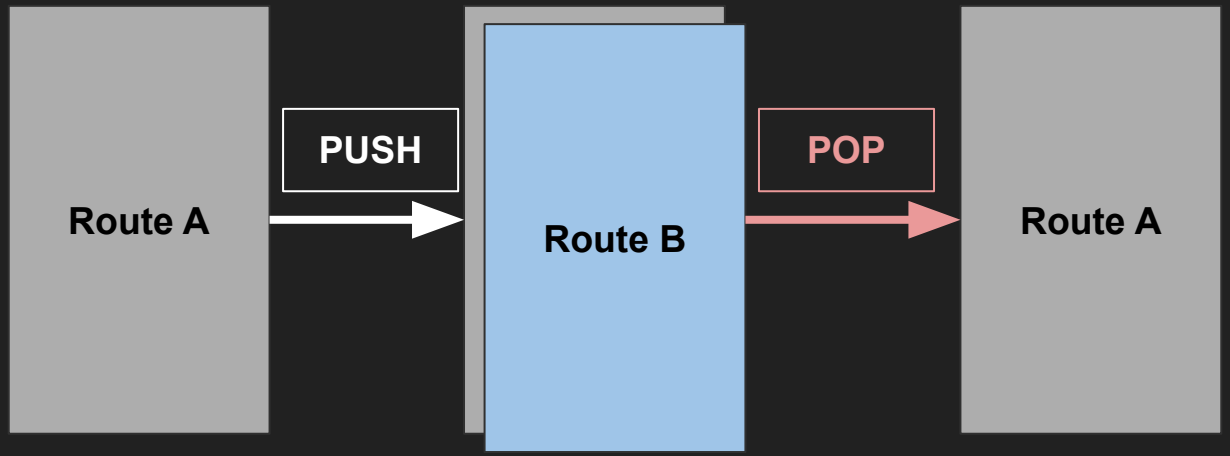
- Este widget se encarga de manejar la pila (stack) de objetos Route.

Route

- Es controlado por un **Navigator** y representa una pantalla. Normalmente es implementado por clases como **MaterialPageRoute** o **CupertinoPageRoute**.
- Si hacemos una comparación con Android una Activity podría equivaler a un Route.

Navigator & Routes

Routes son apiladas (**pushed**) y desapiladas (**popped**) en la pila de **Navigator** bien con **rutas nombradas** o **ruta anónimas**.



Anonymous Routes

La mayoría de las App apilan las pantallas, es decir muestra una encima de otras, esto es muy fácil de lograr haciendo uso de **Navigator**.

Tanto MaterialApp como CupertinoApp ya usan **Navigator** bajo cuerdas. La forma de acceder a la pila de navegación es haciendo uso de **Navigator.of**(), añadiendo una nueva ventana **Navigator.push**() o desapilando otra **Navigator.pop**()

Veamos un pequeño ejemplo para aclarar todo esto y ver el porqué se llaman ‘anonymous Routes’

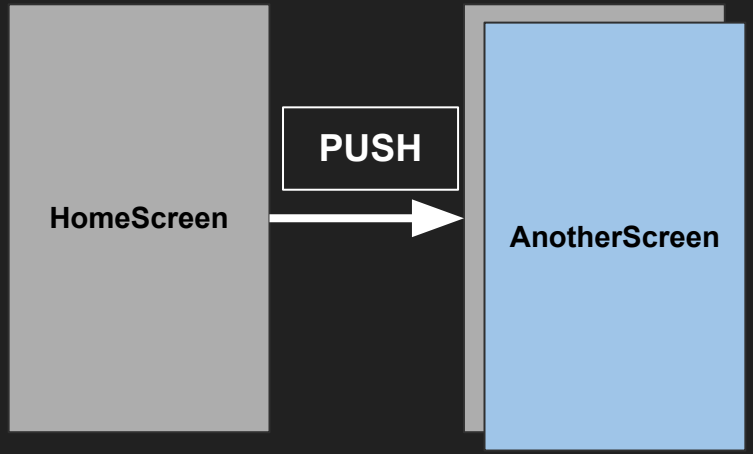
Anonymous Routes

```
//Tenemos en nuestra App un botón que nos va permitir navegar hacia
//otra ventana haciendo uso de anonymous routes.

TextButton(
     child: Text('Go to AnotherScreen'),
     onPressed: () {
        Navigator.push(context,
         MaterialPageRoute(
            builder: (context) {
                     return AnotherScreen();
                            }),
       )})
```

Anonymous Routes

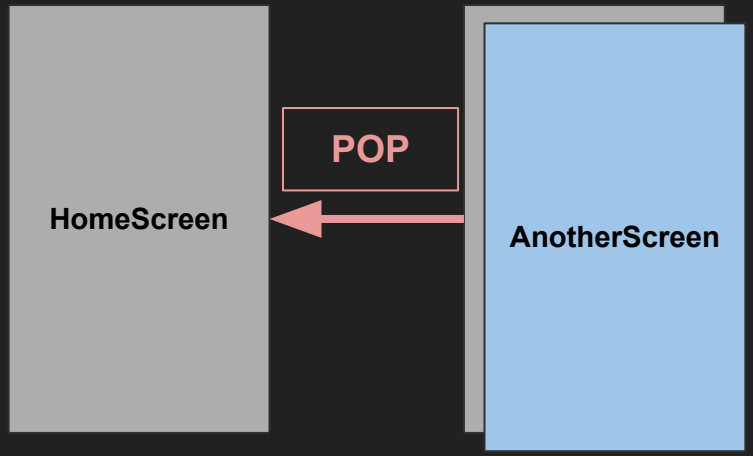
Después de hacer el push, HomeScreen sigue formando parte del árbol de widgets por lo que cualquier objeto State asociado permanecerá presente mientras AnotherScreen está presente.



Anonymous Routes

Para volver desde AnotherScreen a la pantalla anterior, es decir para desapilar tendríamos que hacer un pop.

```
Navigator.pop(context)
```



Named Routes

Flutter nos permite definir nombres a las diferentes rutas de nuestra App para así mantener un código más centralizado y ordenado.

¿Dónde se definen?
Pues dentro de los Widgets MaterialApp o CupertinoApp

¿Cómo se definen?
Veamos un ejemplo

Named Routes

```
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     routes: {
       '/': (context) => HomeScreen(),
       '/another' : (context) => AnotherScreen(),
     },
   );
 }
}
```

Named Routes

```
class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
    ….
   Scaffold(
     body: TextButton(
         child: Text('Go to Another'),
         onPressed: () {
           Navigator.pushNamed(context,' /another');
         }),
}
```

Named Routes

```
class AnotherScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: TextButton(
         child: Text('Go back),
         onPressed: () {
           Navigator.pop(context);
         }),
}
```

Named Routes

Para pasar argumentos de la pantalla origen a la destino cuando usamos Navigator.pushNamed podemos hacer uso de un parámetro de dicha función, arguments.

Podemos enviar cualquier objeto,
Navigator.pushNamed(context, ‘/path’, arguments: object);

En la pantalla destino dentro de build debemos obtenemos los datos de la siguiente forma:

```
Object object = ModalRoute.of(context)?.settings.arguments as Object;
```

Advanced Named Routes

Estas rutas pueden ser predefinidas de una forma aún más flexible usando **onGenerateRoute.**

Se define a igual que la propiedad Route en **MaterialApp**.

Este propiedad nos permite manejar todos las rutas posibles inclusos aquellas que sean dinámicas.

Advanced Named Routes

```
class MyApp extends StatelessWidget{
    @override
    Widget build(BuildContext context){
        return MaterialApp{
            onGenerateRoute: (settings){
            if(settings.name == '/' ){
                return MaterialPageRoute(
                    builder: (context) => HomeScreen());
                }
            }
    … (ver ejemplo completo en los adjuntos del tema.
7_Flutter_Routing_1.dart)
```