Actividad 1

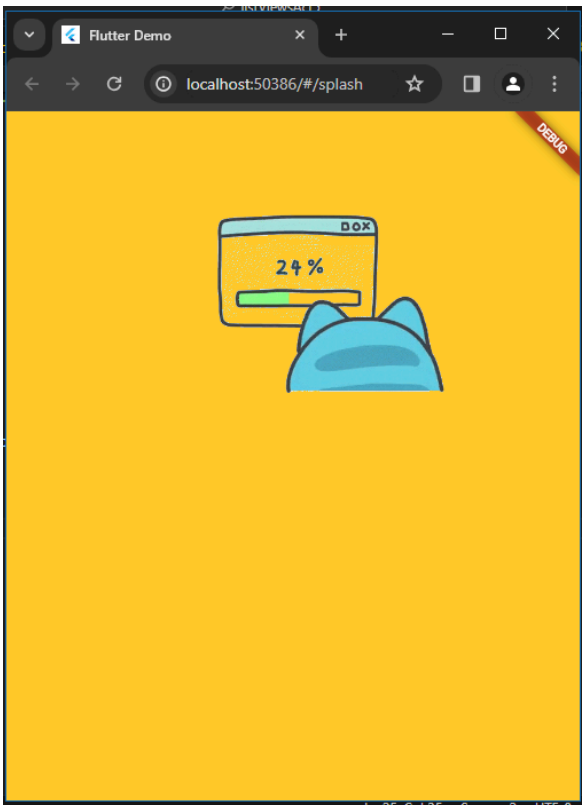
Para realizar el splash vamos a usar y modificar el método **initState()** para que se ejecute, lo que está dentro de la función.

```
class _Splash extends State<Splash> {

  @override
  void initState(){
    super.initState();
    Future.delayed(const Duration(seconds: 10),).then((value) => {
      Navigator.pushReplacementNamed(context, Routes.login)
    });
  }
```

Con el **Future.then()** haremos que el código ignore lo que está dentro de un plazo de 10 segundos. Pasados los segundos se hará la navegación hacia la vista del login. De esta manera conseguiremos el efecto del splash.

```
@override
Widget build(BuildContext context) {

  return Scaffold(
    body: Container(
      color: Colors.amber[400],
      child: Column(
        children: <Widget> [
          const SizedBox(height: 50,),
          SizedBox(
            height: 200,
            child: Stack(
              alignment: AlignmentDirectional.center,
              children: [
                Center(
                  child: SizedBox(
                    width: 600,
                    height: 600,
                    child: Image.asset("assets/cat-loading (2).gif"),
                  ), // SizedBox
                ),  // Center
              ],
            ), // Stack
          ) // SizedBox
        ], // <Widget>[]
      ), // Column
    ) // Container
  ); // Scaffold
```

Este será el cuerpo y contenido del splash, y este el resultado gráfico:



Actividad 2

```
class Routes{
  Routes._();
  static const login = "/login";
  static const listaPaises =  "/listaPaises";
  static const pais = "/pais";
  static const splash = "/splash";
}
```

En esta clase declaramos los nombres de las rutas. Y en la siguiente imagen se declará otra clase que llama a los atributos estáticos de Routes y asocia una instancia de una vista.

```dart
Map<String, Widget Function(BuildContext)> get appRoutes{
  return {
    Routes.splash: (context) => Splash(),
    Routes.login: (context) => Login(title: "Login"),
    Routes.listaPaises: (context) => PagePais(title: "Paises"),
    Routes.pais: (context)  {
      final pais = ModalRoute.of(context)!.settings.arguments as Pais;
      return PaisDescription(pais, title: "Pais");
    },
  };
}
```

En la ruta pais se necesitan pasar valores para inicializar la vista con estos. Para ello utilizo el ModalRoute donde almaceno en una variable los argumentos.

Actividad 3

```dart
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      routes: appRoutes,
      initialRoute: Routes.splash,
    ); // MaterialApp
  }
}
```

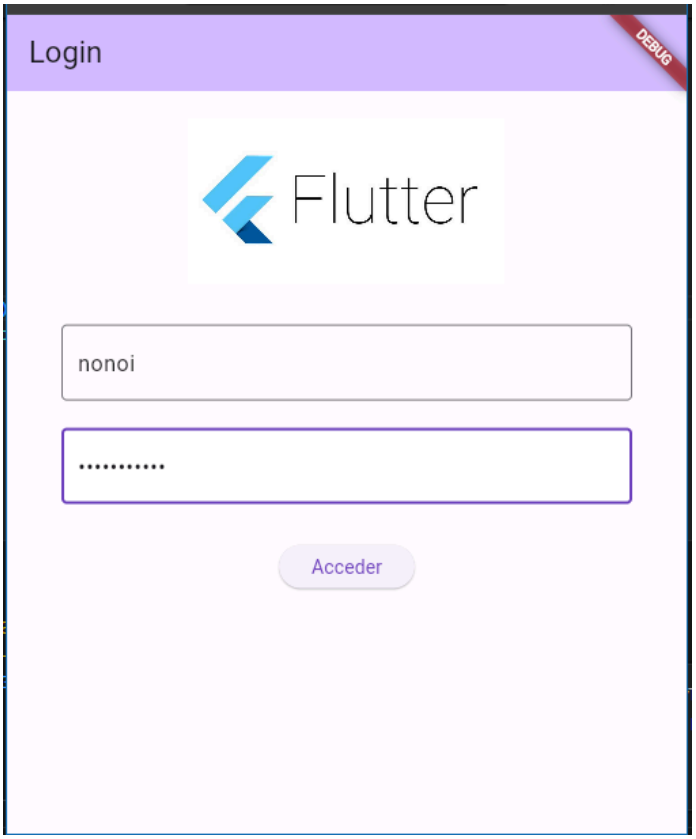
En vez de usar la propiedad “body” del MaterialApp he usado el “initialRoute” para que inicie la aplicación con el splash. También, con la propiedad “routes” le estoy indicando dónde tiene que ir para navegar por las vistas. Este tipo de navegación sustituye al modalRoute.

El splash quedaría de la siguiente manera:

Para que el usuario no pueda retroceder a la anterior pantalla se usará la siguiente función:

```
void initState(){
  super.initState();
  Future.delayed(const Duration(seconds: 10),).then((value) => {
    Navigator.pushReplacementNamed(context, Routes.login)
  });
}
```

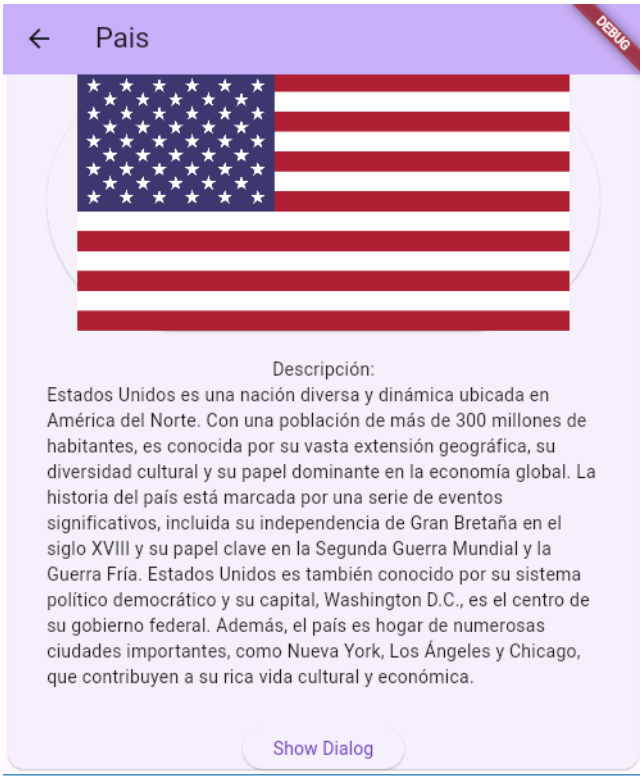
Navigator.pushReplacementNamed() coge el contexto de la vista y la ruta destino. En la siguiente imagen podemos ver el resultado, donde no aparece el botón que sale por defecto para dirigirse a la pantalla anterior.



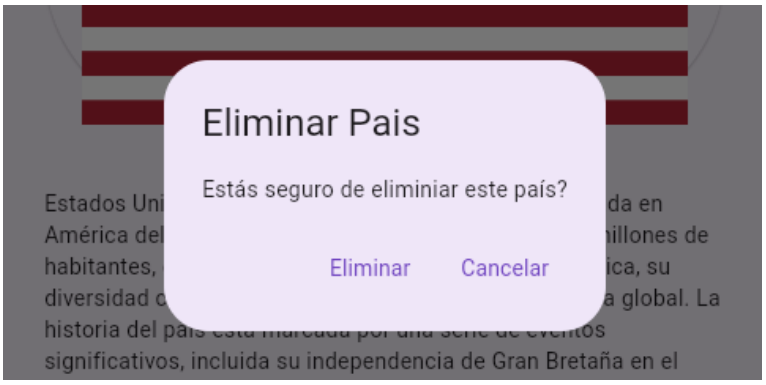
La siguiente imagen igual, no se muestra el botón para la vista que muestra el listado de paises.

Actividad 4

Dentro de la vista “pais” tenemos un botón abajo de todo “show dialog” que permite borrar el país.



Una vez le damos al botón se nos mostrará una ventana emergente donde nos preguntará si deseamos eliminar el país.



Para lograr esta funcionalidad se ha utilizado la función **showDialog**. Dentro de esta función se hace la construcción, usando el context y en el builder crear una función lambda que cree un Widget **AlertDialog()** donde se crea la estructura de la ventana emergente.

```
ElevatedButton(
  onPressed: () => showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: Text("Eliminar Pais"),
      content: Text("Estás seguro de eliminiar este país?"),
      actions: <Widget> [
        TextButton(
          onPressed: (){
            ListaPaises.paises.remove(widget.pais);
            Navigator.pushNamed(context, Routes.listaPaises);
          },
          child: Text("Eliminar")
        ), // TextButton
        TextButton(onPressed: () => Navigator.pop(context),
        child: Text("Cancelar"),
        ), // TextButton
      ], // <Widget>[]
    ) // AlertDialog
  ),
  child: Text("Eliminar Pai"), // ElevatedButton
```

Los botones que se encargan de hacer las acciones están dentro del parámetro **actions.**

Profundizando en el código anterior, este botón siguiente se encargará de eliminar el Widget país en cuestión se hará una navegación a la ruta de la lista de los países.

```
TextButton(
  onPressed: (){
    ListaPaises.paises.remove(widget.pais);
    Navigator.pushNamed(context, Routes.listaPaises);
  },
  child: Text("Eliminar")
), // TextButton
```

El botón que viene a continuación hace una navegación a la página anterior con **pop.** La página anterior sería la propia vista donde se encuentra el país.

```
TextButton(onPressed: () => Navigator.pop(context),
child: Text("Cancelar"),
), // TextButton
```

Actividad 5

Siguiendo la vista anterior, si le damos al botón de eliminar el país dejará de aparecer. En el ejemplo de presentación hemos seleccionado Estados Unidos. A continuación se muestra el resultado de darle a eliminar, que desaparece de la lista de la vista anterior.