

## Proyecto Asincronía

ASDO - 27111993

Instructor:

Edison Sandoval

SENA

Integrantes:

Santiago Vallejo Rodriguez  
Valentina Gonzalez Carmona  
Juan Gabriel Cataño Cataño  
Esteban Cardona Osorio  
Duvan Mendez Henao

Cómo se implementa la asincronía en Dart:

En Dart, la asincronía se implementa principalmente utilizando Futures y Streams.

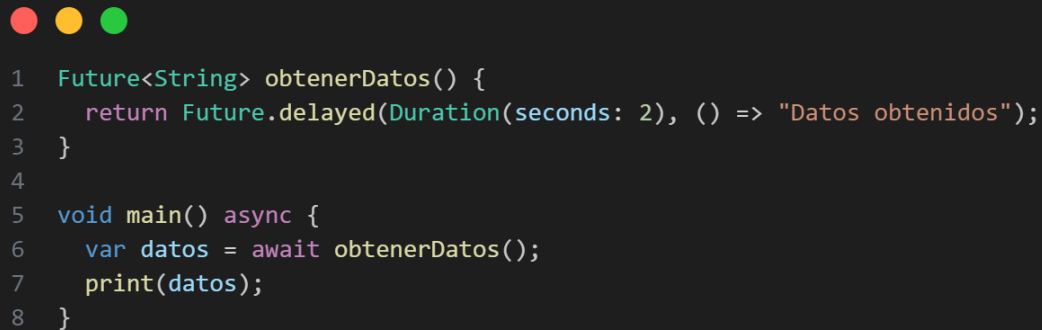
### Futures:

Un **Future** representa un valor o un error que estará disponible en el futuro. Se utiliza para operaciones asíncronas que devuelven un solo valor. Puedes esperar un **Future** para obtener su valor usando **then()** o **async/await** para esperar de manera asíncrona.

Ejemplo utilizando **then()**:

```
1 Future<String> obtenerDatos() {  
2   return Future.delayed(Duration(seconds: 2), () => "Datos obtenidos");  
3 }  
4  
5 void main() {  
6   obtenerDatos().then((datos) {  
7     print(datos);  
8   });  
9 }
```

Ejemplo utilizando **async / await**:



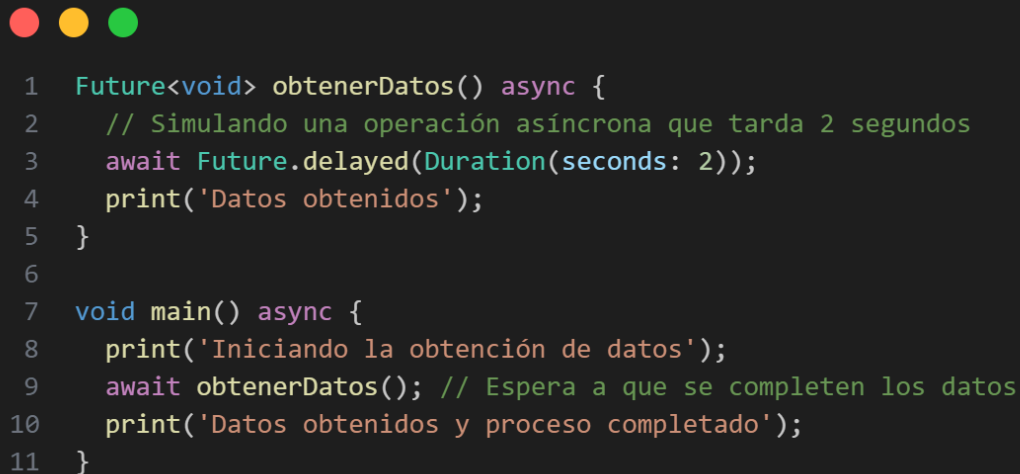
```
1 Future<String> obtenerDatos() {
2     return Future.delayed(Duration(seconds: 2), () => "Datos obtenidos");
3 }
4
5 void main() async {
6     var datos = await obtenerDatos();
7     print(datos);
8 }
```

Que es Await Async y cómo se implementan:

En Dart, `async` y `await` son palabras clave que se utilizan para trabajar con código asíncrono de manera más fácil y legible.

- `async`: La palabra clave `async` se utiliza para marcar una función como asíncrona. Esto significa que la función puede contener operaciones asíncronas y puede usar `await` dentro de ella para esperar que esas operaciones se completen sin bloquear el hilo principal de ejecución.
- `await`: La palabra clave `await` se utiliza dentro de una función marcada con `async` para esperar la finalización de una operación asíncrona. Cuando se encuentra `await`, la ejecución de la función se detiene temporalmente hasta que la operación asíncrona se completa y devuelve un resultado. Mientras tanto, el hilo de ejecución puede ocuparse con otras tareas.

Ejemplo de cómo se utilizan `async` y `await` juntos:



```
1 Future<void> obtenerDatos() async {  
2     // Simulando una operación asíncrona que tarda 2 segundos  
3     await Future.delayed(Duration(seconds: 2));  
4     print('Datos obtenidos');  
5 }  
6  
7 void main() async {  
8     print('Iniciando la obtención de datos');  
9     await obtenerDatos(); // Espera a que se completen los datos  
10    print('Datos obtenidos y proceso completado');  
11 }
```

En este ejemplo, la función `obtenerDatos()` está marcada como `async`, lo que permite usar `await` dentro de ella. Dentro de `obtenerDatos()`, utilizamos `await Future.delayed(Duration(seconds: 2))` para simular una operación asíncrona que toma 2 segundos. En `main()`, también está marcada como `async`, lo que nos permite usar `await` para esperar a que `obtenerDatos()` se complete antes de continuar con el siguiente paso.

El uso de `async` y `await` hace que el código asíncrono sea más fácil de entender y de escribir, ya que se asemeja mucho al estilo de programación asíncrona. Sin embargo, detrás de escena, Dart gestiona la concurrencia de manera eficiente sin bloquear el hilo de ejecución principal.

**Ejercicios:**

Santiago



```
1 void main() {  
2     print('Inicio del programa');  
3  
4     operation().then((value){  
5         print("Operacion completada: $value");  
6         print("Fin del programa");  
7     });  
8 }  
9  
10 Future<int> operation(){  
11     return Future.delayed(Duration(seconds: 3), () {  
12         return 5 * 2;  
13     });  
14 }
```

Valentina



```
1 import 'dart:async';
2
3 void main() {
4   print('Iniciando descarga...');
5   descargarImagen('https://miweb.com/imagen1.jpg').then((imagen) {
6     print('Descarga completada: $imagen');
7   }).catchError((error) {
8     print('Error durante la descarga: $error');
9   });
10  print('Descarga en progreso...');
11 }
12
13 Future<String> descargarImagen(String url) async {
14   // Simulamos una espera de 3 segundos para simular la descarga
15   await Future.delayed(Duration(seconds: 3));
16
17   // Simulamos que la descarga es exitosa y devolvemos el nombre de la imagen
18   return 'imagen1.jpg';
19 }
20
```

Duvan

```
1  import 'dart:convert';
2  import 'dart:io';
3
4  Future<String> descargarPaginaWeb(String url) async {
5      HttpClient httpClient = HttpClient();
6      var request = await httpClient.getUrl(Uri.parse(url));
7      var response = await request.close();
8      if (response.statusCode == HttpStatus.ok) {
9          String responseBody = await response.transform(Utf8Decoder()).join();
10         return responseBody;
11     } else {
12         throw Exception('Error al descargar la página: ${response.statusCode}');
13     }
14 }
15
16 void main() async {
17     var url = 'https://www.example.com';
18
19     print('Iniciando descarga de la página web...\n');
20
21     try {
22         await Future.delayed(Duration(seconds: 5));
23
24         print('Contenido de la página web:');
25         var paginaWeb = await descargarPaginaWeb(url);
26         print(paginaWeb);
27     } catch (e) {
28         print('Error: $e');
29     }
30 }
```

## Pocholo



```
1  import 'dart:async';
2
3
4  Future<String> datos() async {
5    await Future.delayed(Duration(seconds: 2));
6    return 'Nombre: Juan Pérez, Edad: 28, Email: juan.perez@example.com';
7  }
8
9  void main() async {
10   print('Obteniendo datos del usuario...');
11   String userData = await datos();
12
13   print('Datos del usuario obtenidos:');
14   print(userData);
15 }
16
```



Esteban



```
1  void main() {
2      print('Inicio del programa');
3
4      tareaAsincrona().then((mensaje) {
5          print(mensaje);
6          print('Fin del programa');
7      }).catchError((error) {
8          print('Ocurrio un error: $error');
9      });
10 }
11
12 Future<String> tareaAsincrona() async{
13     try {
14         await Future.delayed( new Duration(seconds: 2));
15         return 'Tarea asincrona completada';
16     } catch (e) {
17         print ('Error en la tarea asincrona');
18     }
19 }
```