Programación IV

Profesora Cynthia Estrada

Fecha: 26/08

“Asincronía y Promesas”

Integrantes:

- “Díaz Rossini Juan José”,
- “Gallo Genaro”
- “Navarro Victor Leandro”

---

1. **¿Cuáles son las herramientas que se utilizan en la asincronía?**
2. **¿Qué son las promesas?**
3. **¿Cómo sería el funcionamiento de una promesa?**
4. **¿Qué pasaría al ejecutar una función asíncrona?**
5. **¿Qué significado tiene "consumir promesas"?**

¿Cuáles son las herramientas que se utilizan en la asincronía?

En JavaScript, la asincronía se maneja principalmente con callbacks, promesas y la sintaxis de async/await. Los callbacks fueron la primera herramienta, pero suelen generar problemas de legibilidad cuando se anidan demasiado (lo que se conoce como *callback hell*). Las promesas mejoraron esto al ofrecer una forma más clara de encadenar operaciones. Finalmente, `async/await` es la forma más moderna y legible, ya que permite escribir código asíncrono con una sintaxis muy similar a la del código síncrono.

¿Qué son las promesas?

Una promesa es un objeto que representa el resultado eventual de una operación asíncrona. Puede estar en tres estados: pendiente (aún no se resolvió), resuelta (la operación terminó correctamente) o rechazada (hubo un error). Gracias a las promesas, es posible manejar operaciones que tardan en completarse, como llamadas a una API, sin detener la ejecución del resto del programa.

¿Cómo sería el funcionamiento de una promesa?

Cuando se crea una promesa, dentro de ella se ejecuta una tarea que puede tardar (por ejemplo, consultar una base de datos). Si la operación es exitosa, se llama a la función `resolve()` y la promesa pasa al estado resuelta; si ocurre un error, se llama a `reject()`, cambiando al estado rechazada. Posteriormente, quien consume la promesa puede manejar estos casos con `.then()` para el éxito y `.catch()` para los errores.

¿Qué pasaría al ejecutar una función asíncrona?

Al ejecutar una función marcada con `async`, esta devuelve automáticamente una promesa, incluso si no se escribe explícitamente. Dentro de esa función, se puede usar la palabra clave `await`, que hace que el código espere a que otra promesa se resuelva antes de continuar, sin bloquear

el hilo principal. De esta forma, el flujo parece "secuencial", aunque realmente sigue siendo asíncrono.

¿Qué significado tiene "consumir promesas"?

Consumir una promesa significa utilizar su resultado una vez que termina la operación asíncrona. En la práctica, esto se logra de dos maneras: encadenando métodos como `.then()` y `.catch()`, o usando `async/await` para esperar la resolución dentro de una función asíncrona. Es decir, consumir una promesa es obtener y trabajar con el valor que devuelve, o manejar el error si falla.