Programación IV
Profesora Cynthia Estrada
Fecha: 25/08

“Asincronía y Sincronía”

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

---

1. **¿Que es una tarea asincrónica y una tarea sincrónica?**
2. **Cómo sería el comportamiento de las tareas sincrónicas cómo así también de las tareas asincrónicas?**
3. **Indique las características que tiene un suceso sincrónico como así también un suceso asincrónico**
4. **Es importante saber cuándo el código es bloqueante o no? ¿Cómo sería?**
5. **¿Pueden existir múltiples tareas asíncronas? Las tareas pueden quedar pendientes de forma infinita o nunca realizarse o fallar?**
6. **Existen mecanismo para controlar estás tareas? Indique de ser así**
7. **Diga ejemplos de tareas asincrónicas y sincrónicas**
8. **¿Qué situación puede ocurrir en caso de complicaciones en tareas asíncronas?**

¿Qué es una tarea asincrónica y una tarea sincrónica?

Una tarea sincrónica es aquella que se ejecuta en orden y bloquea la ejecución hasta que termina. En cambio, una tarea asincrónica permite que el programa continúe con otras operaciones mientras espera que esa tarea se complete, sin detener el flujo principal.

¿Cómo sería el comportamiento de las tareas sincrónicas como así también de las tareas asincrónicas?

Las tareas sincrónicas se ejecutan una tras otra, de manera secuencial. Si una tarea tarda, todo el programa espera. En cambio, las tareas asincrónicas se inician y, mientras están en proceso, el programa puede seguir ejecutando otras instrucciones; cuando la tarea finaliza, se notifica el resultado (éxito o error).

Indique las características que tiene un suceso sincrónico como así también un suceso asincrónico

- **Sincrónico: secuencial, bloqueante, más fácil de razonar pero puede generar lentitud si una tarea demora.**

- **Asincrónico: no bloqueante, permite multitarea, mejora el rendimiento, pero requiere mecanismos de control para manejar resultados y errores.**

¿Es importante saber cuándo el código es bloqueante o no? ¿Cómo sería?

Sí, es muy importante. Un código bloqueante detiene la ejecución del resto del programa hasta que finaliza, lo que puede provocar demoras o congelamientos. Un código no bloqueante, en cambio, deja que otras partes del programa sigan

ejecutándose, lo cual es clave en aplicaciones modernas que necesitan responder rápido al usuario.

¿Pueden existir múltiples tareas asíncronas? ¿Las tareas pueden quedar pendientes de forma infinita o nunca realizarse o fallar?

Sí, pueden existir varias tareas asíncronas ejecutándose en paralelo. Sin embargo, algunas pueden quedar pendientes si nunca se resuelven (por ejemplo, por un error de conexión) o incluso fallar. Por eso es fundamental manejar los posibles errores y definir tiempos de espera.

¿Existen mecanismos para controlar estas tareas? Indique de ser así

Sí. En JavaScript, los mecanismos principales son las promesas (`.then()`, `.catch()`, `.finally()`), la sintaxis async/await, y métodos como `Promise.all`, `Promise.race` o `Promise.any`, que ayudan a manejar varias tareas a la vez. Además, se pueden usar timeouts, try/catch y validaciones para evitar que queden colgadas indefinidamente.

Diga ejemplos de tareas asincrónicas y sincrónicas

- Asincrónicas: petición a una API, lectura de archivos en disco, temporizadores (`setTimeout`), operaciones de red.

- Sincrónicas: operaciones matemáticas simples, recorrer un array con un `for`, asignar valores a variables.

¿Qué situación puede ocurrir en caso de complicaciones en tareas asíncronas?

Si una tarea asincrónica falla o nunca se resuelve, puede provocar errores en el flujo del programa, resultados incompletos o bloqueos indirectos (por ejemplo, que la interfaz quede esperando datos que nunca llegan). Por eso es importante siempre manejar excepciones y prever qué hacer en caso de error.