



Es hora de que pongas en práctica todo lo aprendido. 

En este ejercicio realizaremos la implementación de una arquitectura de mensajería asíncrona.

Más adelante conseguirás la resolución para que valides tus respuestas y puedas monitorear tu progreso. 

¡Manos a la obra!

1. **Desafío** : Implementa un prototipo de una arquitectura de mensajería asíncrona, basado en los conceptos y patrones estudiados en el manual. Tu ejercicio debe incluir los siguientes aspectos:

a. Descripción del Flujo:

- ⇒ Explica el rol de los productores, consumidores y la cola de mensajes en tu solución.
- ⇒ Describe, mediante un diagrama, el flujo de mensajes desde que son generados hasta que son procesados.

b. Implementación Práctica:

- ⇒ Crea un módulo (puede ser en Python o pseudocódigo) que simule un productor que envía mensajes a una cola.
- ⇒ Implementa una cola de mensajes (puede ser simulada con una estructura de datos simple).
- ⇒ Desarrolla un módulo consumidor que reciba y procese los mensajes de la cola.

c. Análisis Comparativo:

- ⇒ Elabora un cuadro comparativo que resuma las ventajas y desventajas de la mensajería asíncrona frente a la mensajería síncrona, basado en lo estudiado en el manual.

2. **¿Dónde se lleva a cabo?** 

Realiza la actividad en Visual Studio Code. Puedes usar AWS Academy y/o

SQLiteOnline para simular o complementar tu entorno de pruebas si lo consideras necesario.

3. Tiempo de dedicación

Se estima que la resolución tome entre 1 y 2 horas.

4. Recursos

- Puedes utilizar recursos complementarios de AWS Academy para consultar ejemplos de servicios de mensajería en la nube.
- Si deseas, añade prints de pantalla, diagramas o imágenes que respalden el desarrollo de tu solución.

5. Plus

- Opcionalmente, integra y experimenta con AWS SQS para trabajar con una solución real en la nube.
- Comparte el enlace a tu proyecto en GitHub para que otros bootcampers puedan revisar y comentar tu trabajo.

6. Condición

Esta práctica no requiere ser entregada ni evaluada por el mentor. Sin embargo, se recomienda compartir tus resultados con el resto de los bootcampers para construir conocimiento en conjunto.

Resolución del ejercicio

- Documenta tu resolución paso a paso, incluyendo el código fuente, capturas de pantalla y diagramas.
- Compara tus resultados con la solución esperada para asegurarte de que has comprendido todos los conceptos fundamentales sobre arquitecturas basadas en mensajería asíncrona.

1. Descripción del Flujo

En esta solución se plantea una arquitectura básica en la que:

- **Productores:** Generan y envían mensajes a una cola.
- **Cola de mensajes:** Actúa como intermediario para almacenar temporalmente los mensajes.
- **Consumidores:** Extraen y procesan los mensajes de la cola.
- **Diagrama conceptual (simplificado):**

```
[Productor] → [Cola de Mensajes] → [Consumidor]
```

El productor envía mensajes de forma asíncrona a la cola, donde estos quedan almacenados hasta que el consumidor los lee y procesa.

2. Implementación Práctica

A continuación se muestra un ejemplo en Python (o pseudocódigo) que simula la arquitectura:

Working Time 🧐

L7: Arquitecturas básicas orientadas a mensajes

```
# Simulación de una cola de mensajes usando una lista
class MessageQueue:
    def __init__(self):
        self.queue = []

    def send_message(self, message):
        self.queue.append(message)
        print(f"Mensaje enviado: {message}")

    def receive_message(self):
        if self.queue:
            message = self.queue.pop(0)
            print(f"Mensaje recibido: {message}")
            return message
        else:
            print("No hay mensajes en la cola.")
            return None

# Módulo Productor
def productor(queue, messages):
    for msg in messages:
        queue.send_message(msg)

# Módulo Consumidor
def consumidor(queue):
    while True:
        message = queue.receive_message()
        if message is None:
            break

        # Procesamiento del mensaje (aquí se simula con un print)
        print(f"Procesando mensaje: {message}")

# Ejecución de la simulación
if __name__ == "__main__":
    cola = MessageQueue()
    lista_de_mensajes = ["Mensaje 1", "Mensaje 2", "Mensaje 3"]

    print("=== Ejecución del Productor ===")
    productor(cola, lista_de_mensajes)

    print("\n=== Ejecución del Consumidor ===")
    consumidor(cola)
```

Explicación del código:

- La clase `MessageQueue` simula una cola con métodos para enviar y recibir mensajes.
- La función `productor` recorre una lista de mensajes y los envía a la cola.

- La función `consumidor` extrae los mensajes de la cola y simula su procesamiento.
- Se ejecutan ambos módulos de forma secuencial para demostrar el flujo asíncrono.

3. **Cuadro Comparativo:** Mensajería Asíncrona vs. Mensajería Síncrona

Aspecto	Mensajería Asíncrona	Mensajería Síncrona
Acoplamiento	Bajo (componentes desacoplados)	Alto (llamadas directas entre componentes)
Respuesta	Eventual (no inmediata)	Inmediata
Escalabilidad	Alta, permite procesar en paralelo	Limitada, depende del tiempo de respuesta
Tolerancia a fallos	Mejor manejo de errores y picos de carga	Menor tolerancia a fallos

Este ejercicio ilustra cómo implementar y simular una arquitectura basada en mensajería asíncrona, permitiendo comprender la separación de roles y la capacidad de procesar mensajes de forma distribuida.