

Universidad Laica Eloy Alfaro de Manabí

Facultad de Ciencias De La Vida Y Tecnologías

Carrera: Tecnología de la Información

Asignatura:

Sistemas Distribuidos

Nivel/Paralelo:

8to Nivel “A”

Docente:

Ing. Willian Zamora M.

Integrantes:

Bravo Ponce Darlin alexander

Peralta Rivas Rody Leandro

Mero Quimis José Alexander

PROGRAMACIÓN CONCURRENTE

The background is a dark blue field decorated with various geometric elements. It includes numerous small squares in white, light blue, orange, and pink. Some of these squares are solid, while others are hollow outlines. Additionally, there are several thin, vertical white lines of varying lengths scattered across the composition, creating a modern, minimalist aesthetic.

Índice del contenido

- Introducción
- Desarrollo
- Talleres
- Conclusiones
- Recomendaciones
- Referencias Bibliográficas

Introducción

La programación concurrente se enfoca en programas que puedan realizar múltiples tareas simultáneamente. Abarcaremos la definición de una manera clara para poder entender la programación concurrente, presentaremos sus ventajas y desventajas, la manera en la que la podemos aplicar, en como la tecnología java tiene sus propios mecanismos para soportarla, mostrando ejemplos prácticos en código para entender el funcionamiento de la concurrencia.

The background is a solid dark blue. It is decorated with several thin white vertical lines of varying lengths and small squares in white, light blue, and pink. Some squares are solid, while others are just outlines. The word "DESARROLLO" is centered in a large, white, sans-serif font.

DESARROLLO

¿Qué es programación secuencial?

Ejecuta las tareas de una aplicación de manera lineal y una por una. Hasta que no se finaliza una tarea no se comienza con la siguiente tarea.

¿Qué es concurrencia?

Es la capacidad de ejecutar múltiples actividades en paralelo o simultáneamente.

La podemos encontrar en:

- Varios navegadores accediendo a la misma pagina
- Impresión de documentos mientras se realiza una consulta
- Varios usuarios conectados al mismo sistema
- Juegos y vehículos.

Programación Concurrente

Permite que una aplicación pueda realizar varias tareas al mismo tiempo. Para ello, se pueden generar varios hilos de ejecución o repartir las ejecuciones entre varios núcleos del CPU, consiguiendo ejecuciones simultaneas y paralelas de distintos hilos de ejecución.



Programación concurrente en PYTHON

Existen 3 librerías necesarias para utilizar la programación concurrente en Python.

- Threading: El propio sistema toma la decisión de la ejecución de las distintas partes del hilo.
- Asyncio: Igual a Threading, pero tendremos mas control en distintas partes del hilo.
- Multiprocessing: Reparte los hilos creados en varios núcleos del CPU, permitiendo ejecutar tareas de manera paralela, consumiendo recursos.

Ejemplo practico en PYTHON

Ejercicio1 > vehiculos.py > ...

```
1  import threading
2  import time
3
4  #Funcion que simulara la llegada de los vehiculos al estacionamiento
5  def llegada_vehiculo(vehiculo_id):
6      hora_llegada = time.strftime("%H:%M:%S", time.localtime())
7      print(f"Vehiculo {vehiculo_id} llego al estacionamiento a las {hora_llegada}!")
8      tiempo_espera = 5
9      time.sleep(tiempo_espera)
10     hora_registro = time.strftime("%H:%M:%S", time.localtime())
11     print(f"Vehiculo {vehiculo_id} registrado a las {hora_registro}!")
12
13 #Simulacion de llegada
14 def simulacion_llegada(cantidad_vehiculos):
15     for vehiculo_id in range(1, cantidad_vehiculos + 1):
16         threading.Thread(target=llegada_vehiculo, args=(vehiculo_id,)).start()
17         time.sleep(2)
18
19 #Simula la llegada de X cantidad de vehiculos
20 simulacion_llegada(3)
```

● Resultado:

```
Vehículo 1 llegó al estacionamiento a las 23:39:21
Vehículo 2 llegó al estacionamiento a las 23:39:23
Vehículo 3 llegó al estacionamiento a las 23:39:25
Vehículo 1 registrado a las 23:39:26
Vehículo 2 registrado a las 23:39:28
Vehículo 3 registrado a las 23:39:30
```

Aplicaciones concurrentes

Ventajas de la programación concurrente:

Eficiencia: La programación concurrente permite que una aplicación pueda realizar varias actividades simultáneamente.

Escalabilidad: Es capaz de gestionar cargas crecientes y adaptarse a ellas.

Gestión de las comunicaciones: La programación concurrente permite una gestión eficiente de los recursos de comunicación entre actividades, esta comunicación no implicara que toda la aplicación se detenga esperando alguna respuesta.

Flexibilidad: Las aplicaciones concurrentes tienen un diseño modular y estructurado, esto facilita la identificación y adaptación del modulo implicado por cambios en los requisitos de la aplicación.

Menor hueco semántico: Hay aplicaciones informáticas como por ejemplo hojas de calculo que requieren de varias actividades simultaneas, asi mismo los videojuegos, si se implantan como programas concurrentes resulta sencillo su diseño.

Aplicaciones concurrentes

Inconvenientes de la programación concurrente:

Programación delicada: Durante el desarrollo de aplicaciones concurrentes, pueden llegar a surgir varios problemas, por ejemplo, el problema llamado condición de carrera, que puede generar inconsistencias en el valor de las variables y atributos compartidos entre las actividades cuando estas los modifiquen, otro problema es el llamado interbloqueos.

Depuración compleja: En la aplicación concurrente puede ser complicado identificar y corregir errores, debido a que las diferentes partes de la aplicación pueden ejecutarse en diferentes secuencias en cada ejecución. Además las herramientas de depuración que se utilizan pueden no capturar adecuadamente los errores que ocurren debido a la naturaleza concurrente.

Aplicaciones reales

Programa de control del acelerador lineal.

Descripción del programa: El Therac-25 era un acelerador lineal utilizado en tratamientos de radioterapia para cáncer, desarrollado entre 1976 y 1982. En este acelerador, la gestión de errores se integró en el software en lugar de ser controlada por hardware, como en modelos anteriores.

Funcionamiento: El tratamiento se configuraba mediante la introducción de parámetros por parte del operador, como tipo de radiación, potencia, etc. El programa de control monitorizaba todos los componentes del sistema y detenía el tratamiento en caso de errores leves o graves.

Estructura del programa concurrente: El programa estaba compuesto por múltiples actividades, cada una gestionando diferentes elementos del sistema. Las actividades más importantes incluían la gestión de entrada, la gestión de los imanes y una actividad de pausa.

Error crítico: Debido a un error de programación, cuando finalizaba la actividad de pausa, el programa no consultaba si se habían solicitado cambios en la configuración del sistema. Esto causó que el estado real del sistema no coincidiera con la información mostrada al operador, lo que resultó en accidentes graves durante algunos tratamientos.

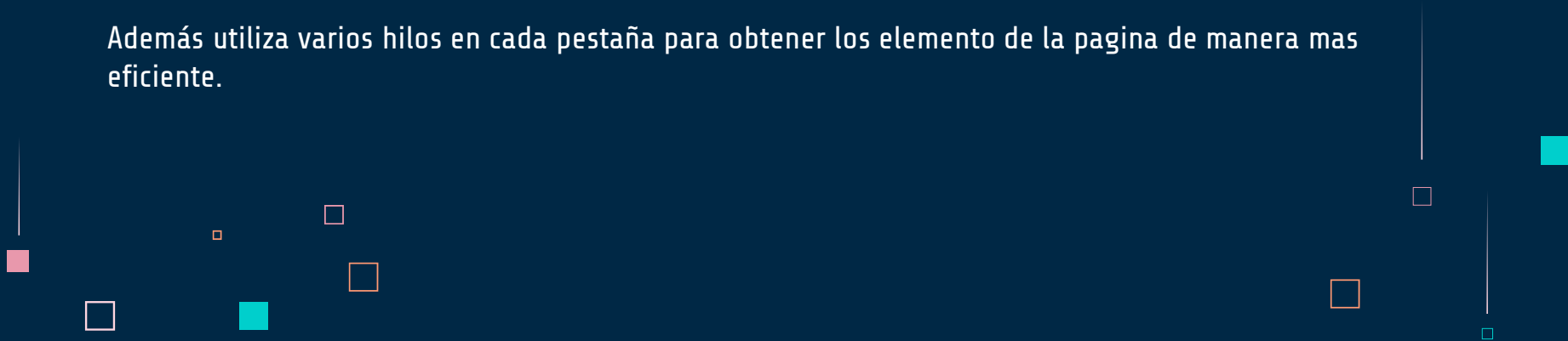
Aplicaciones reales

Navegador web.

El navegador web Chrome de Google, lanzado en el 2008 introdujo una arquitectura diferente a otros navegadores como Internet Explorer o Firefox, puesto que en Chrome cada pestaña abierta era respaldada por un proceso independiente, con esto si alguno de las pestañas fallaba las demás podían seguir funcionando con normalidad.

Mejóro el rendimiento optimizando su soporte para javascript, compilando y manteniendo el código generado para no tener que interpretarlo cada vez.

Además utiliza varios hilos en cada pestaña para obtener los elemento de la pagina de manera mas eficiente.



Problemas clásicos

Lectores-Escritores.

En este problema asumimos que existe un recurso compartido por varias actividades, algunas de ellas podrán modificar el estado del recurso, a estas actividades se les llamara escritores, mientras que aquellos procesos que solo lean sin poder cambiar el estado se les llamara lectores, tomemos al recurso como si fuese un fichero.

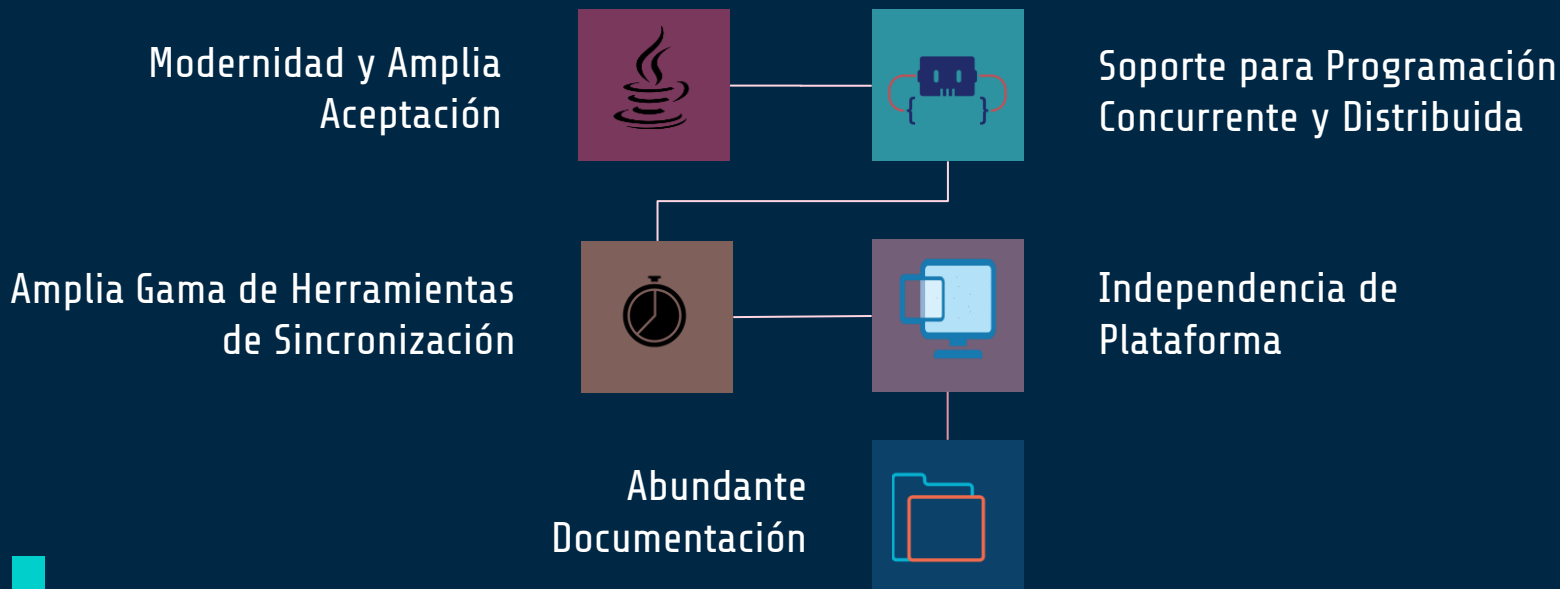
Para hacer un correcto uso se deben de poner algunas restricciones:

- Varios procesos lectores podrán acceder al recurso al mismo tiempo.
- Mientras que solo un proceso escritor podrá acceder a él. Cuando esto pase ningún otro proceso podrá tener acceso al recurso.



Programacion en Java

La tecnología Java es una elección ideal para entender el concepto de concurrencia y desarrollar ejemplos de programas concurrentes por varias razones:



Concurrencia en Java

- Cada hilo es una unidad de planificación, mientras que un proceso es la entidad que mantiene un programa en ejecución.
- En Java, el método `main()` en una clase define el hilo principal de ejecución de la aplicación, que se ejecuta utilizando una máquina virtual Java (JVM), creando un proceso compatible con el sistema operativo.
- Java permite la gestión de múltiples hilos de ejecución sin necesidad de bibliotecas adicionales.
- Los hilos pueden ser explícitos o implícitos, utilizados para tareas como la interfaz de usuario y la recolección de residuos.

Creación y Gestión de Hilos

Definición de Hilos

Java permite crear hilos de ejecución de diferentes formas, ya sea implementando la interfaz Runnable o extendiendo la clase Thread. Estas alternativas ofrecen flexibilidad para adaptarse a las necesidades de cada proyecto.

Identificación de Hilos

Cada hilo en Java puede tener un nombre único que facilita su seguimiento y depuración. Además, es posible acceder al hilo de ejecución actual mediante la clase Thread, lo que permite a los programas obtener información valiosa sobre su entorno de ejecución.

1

2

3

Ciclo de Vida de los Hilos

Los hilos en Java pueden pasar por diferentes estados, como nuevo, preparado, en ejecución, bloqueado y terminado. Entender estos estados y cómo gestionarlos es crucial para escribir código concurrente robusto y eficiente.

```
import java.lang.Thread;

// Creamos una clase Corredor que extiende Thread para representar a cada corredor como un hilo
class Corredor extends Thread {
    private int numeroCorredor;

    public Corredor(int numeroCorredor) {
        this.numeroCorredor = numeroCorredor;
    }

    public void run() {
        // Cuando se inicia el hilo, se imprime un mensaje indicando que el corredor ha comenzado.
        System.out.println("Comienza el corredor " + numeroCorredor);
        long tiempoInicio = System.currentTimeMillis(); // Se registra el tiempo de inicio de la carrera.
        double tiempoTotal = 0;

        try {
            // Simula el avance del corredor en la carrera a través de un bucle.
            for (int i = 1; i <= 20; i++) {
                System.out.println("Corredor " + numeroCorredor + " - Metro " + i);

                // Se simula un tiempo de avance aleatorio entre 0.2 y 0.4 segundos.
                double tiempoAvance = (Math.random() * 0.2) + 0.2;
                tiempoTotal += tiempoAvance;

                Thread.sleep((long)(tiempoAvance * 1000));
            }
        } catch (InterruptedException e) {
            System.out.println("Corredor " + numeroCorredor + " interrumpido.");
        }
    }
}
```

Ejemplo práctico en Java

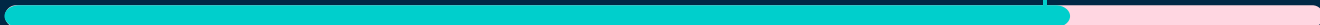

● Resultado:

```
Corredor 2 - Metro 17
Corredor 1 - Metro 18
Corredor 3 - Metro 17
Corredor 2 - Metro 18
Corredor 1 - Metro 19
Corredor 3 - Metro 18
Corredor 3 - Metro 19
Corredor 2 - Metro 19
Corredor 1 - Metro 20
Corredor 3 - Metro 20
El corredor 1 termin♦ la carrera en 11.1455870364305 segundos.
Corredor 2 - Metro 20
El corredor 3 termin♦ la carrera en 11.524116910921176 segundos.
El corredor 2 termin♦ la carrera en 12.066570058543423 segundos.
♦La carrera ha terminado!
```

Conclusiones



La programación concurrente es crucial en el desarrollo de software actual. Permite realizar múltiples tareas simultáneamente, aumentando así la eficiencia y la capacidad de respuesta del sistema. A través de ejemplos en Python y Java, resaltamos su importancia para crear aplicaciones más eficientes y responsivas en el exigente mundo digital.



Recomendaciones

- Aprender conceptos básicos de la programación concurrente.
- Reconocer sus ventajas y desafíos que tienen la programación concurrente.
- Explorar aplicaciones y problemas comunes.
- Practicar la creación y gestión de hilos.

Referencias Bibliográficas

Muñoz Escoí, F. D. (2013). Concurrencia y sistemas distribuidos: (ed.). Editorial de la Universidad Politécnica de Valencia. <https://elibro.net/es/lc/ulead/titulos/57365>



Gracias