PROGRAMACIÓN PARALELA Y DISTRIBUIDA

SIMULACIÓN CONCURRENTE DE UNA COLMENA INTELIGENTE

PABLO JIMÉNEZ GIRARDEAU _ 16 de mayo de 2025

1. DESCRIPCIÓN DEL PROYECTO

Este proyecto consiste en la **simulación concurrente y distribuida** del comportamiento interno de una colmena de abejas, implementada en Python. La colmena está formada por distintos tipos de abejas (recolectoras, almacenadoras, nodrizas, defensoras y una reina) que trabajan de forma simultánea y coordinada para mantener el funcionamiento del ecosistema.

Cada tipo de abeja cumple una función específica y opera como un **hilo independiente**, comunicándose con otros roles mediante estructuras compartidas como colas y eventos. Además, el entorno se ve afectado por **eventos aleatorios** como cambios climáticos, ciclos día/noche y ataques externos, lo que obliga a la colmena a adaptarse dinámicamente.

2. ¿QUÉ PROBLEMA RESUELVE?

Este proyecto **simula la cooperación de múltiples agentes con roles distribuidos**, aplicando técnicas de concurrencia para:

2.1 Coordinar tareas interdependientes (como la recolección y almacenamiento de néctar).

2.2 Gestionar recursos limitados compartidos (celdas de néctar, larvas).

2.3 Responder de forma asíncrona a eventos externos como ataques o lluvia.

2.4 Redistribuir dinámicamente tareas según necesidades internas.

Resuelve el problema de **cómo diseñar un sistema reactivo y adaptable**, manteniendo la eficiencia en un entorno cambiante con múltiples hilos cooperando y compitiendo por recursos.

3. ¿CÓMO ESTÁ ESTRUCTURADO?

El proyecto se divide en los siguientes módulos:

colmena.py

Contiene la lógica central del sistema. Aquí se definen los recursos compartidos, mecanismos de sincronización y el estado global de la colmena (colas, contadores, estadísticas, eventos, etc.).

agentes.py

Define las clases de abejas como subclases de threading.Thread, cada una con su comportamiento especializado:

- Recolectora: visita flores y recolecta néctar.
- Almacenadora: toma el néctar de la cola y lo guarda en las celdas.
- Nodriza: alimenta larvas si hay néctar disponible.
- Defensora: patrulla y neutraliza ataques.
- Reina: gestiona desequilibrios y reemplazos.

eventos.py

Simula el entorno dinámico. Usa hilos para generar eventos aleatorios:

- Lluvia que reduce la calidad de las flores.
- Cambios día/noche que alteran el comportamiento de las abejas.
- Ataques de enemigos que deben ser respondidos por defensoras.

analisis.py

Registra métricas del sistema y genera informes con estadísticas sobre:

- Recolección y almacenamiento de néctar.
- Ataques y defensa.
- Alimentación de larvas.
- Cambios de rol y distribución de tareas.

main.py

Punto de entrada. Inicializa la colmena, lanza abejas, eventos y el analizador, y espera hasta que la simulación termine.

4. TÉCNICAS DE CONCURRENCIA, PARALELISMO O DISTRIBUCIÓN UTILIZADAS

✓ Hilos (threading.Thread)

Cada abeja y cada evento ambiental se ejecuta en su propio hilo, lo que permite que múltiples tareas ocurran de forma simultánea sin bloquearse entre sí.

✓ Exclusión mutua

Se utilizan:

- threading.Lock para proteger el acceso a celdas, estadísticas y larvas.
- threading.Semaphore para controlar la capacidad limitada de las celdas de néctar.

✓ Comunicación entre hilos

- queue

Queue permite pasar datos entre recolectoras y almacenadoras (cola de néctar).

- Otra queue.

Queue es usada para que las abejas comuniquen eventos a la reina.

✓ Eventos (threading.Event)

- Señalan ataques, condiciones climáticas (lluvia), cambios de día/noche y el fin de la simulación.
- Las abejas reaccionan solo cuando ciertos eventos están activos.

✓ Patrón productor-consumidor

- Recolectoras (productores) generan néctar y lo ponen en la cola.
- Almacenadoras (consumidores) lo extraen y lo almacenan si hay espacio.

✓ Redistribución dinámica de roles

La reina redistribuye tareas si detecta desequilibrios en la colmena o si una abeja muere o está inactiva.

5. CAPTURAS O EJEMPLOS DE EJECUCIÓN

Ejemplo de ejecución en consola

```
[Main] Colmena inicializada.
[Main] Reina iniciada.
[Main] 21 abejas lanzadas.
[Eventos] Iniciando simulación de eventos ambientales...
[Clima] Clima favorable. Calidad de flores: 1.12
[Ciclo] Cambio a noche
[Ataque] ¡Alerta! Ataque de avispa con intensidad 7.5
[Clima] Comienza a llover. Calidad de flores: 0.55
[Ataque] Ataque no neutralizado a tiempo
[Eventos] Fin de la simulación de eventos ambientales
[Main] Simulación finalizada. Generando informe...
```

Fragmento del informe generado automáticamente:

```
================================================
 INFORME DE RENDIMIENTO DE LA COLMENA
================================================

Tiempo total de simulación: 30.02 segundos (0.50 minutos)

--- PRODUCCIÓN DE MIEL ---
Néctar recolectado: 124 unidades
Néctar almacenado: 102 unidades
Tasa de recolección: 248.00 unidades/minuto
Eficiencia de almacenamiento: 82.26%

--- SEGURIDAD DE LA COLMENA ---
Ataques detectados: 3
Ataques neutralizados: 2
Eficiencia de defensa: 66.67%

--- REPRODUCCIÓN ---
Larvas alimentadas: 13
Tasa de alimentación: 26.00 larvas/minuto

--- ADAPTABILIDAD ---
Cambios de rol: 2

--- ESTADO FINAL ---
Celdas ocupadas: 13
Celdas libres: 87

--- ROLES PROMEDIO ---
recolectora: 8.12
almacenadora: 5.42
nodriza: 3.86
defensora: 2.57
```

6. INSTRUCCIONES DE EJECUCIÓN

- Requisitos: Python 3.x, sin librerías externas.
- Archivos necesarios: main.py, colmena.py, agentes.py, eventos.py, análisis.py
- Ejecutar con: python main.py. El informe se imprime en consola y se guarda como informe_colmena.js