Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Programación de paralela y concurrente

Practica 1

Simulación de un sistema de Hospitalario

Autor:

Jorge Miguel Alvarado Reyes

No. Cuenta:

421010301

April 16, 2025

Contents

Objetivo

El objetivo de esta práctica es aplicar los paradigmas de programación paralela, concurrente y asincrónica mediante la simulación de un sistema hospitalario que imita el flujo real de atención médica en urgencias.

1 Diseño del flujo hospitalario

Antes de comenzar con la implementación del código, fue necesario realizar un recorrido mental del proceso de atención en urgencias, identificando los pasos clave que sigue un paciente desde su llegada al hospital hasta su egreso. Este análisis permitió establecer un flujo lógico que se representa en el siguiente diagrama:

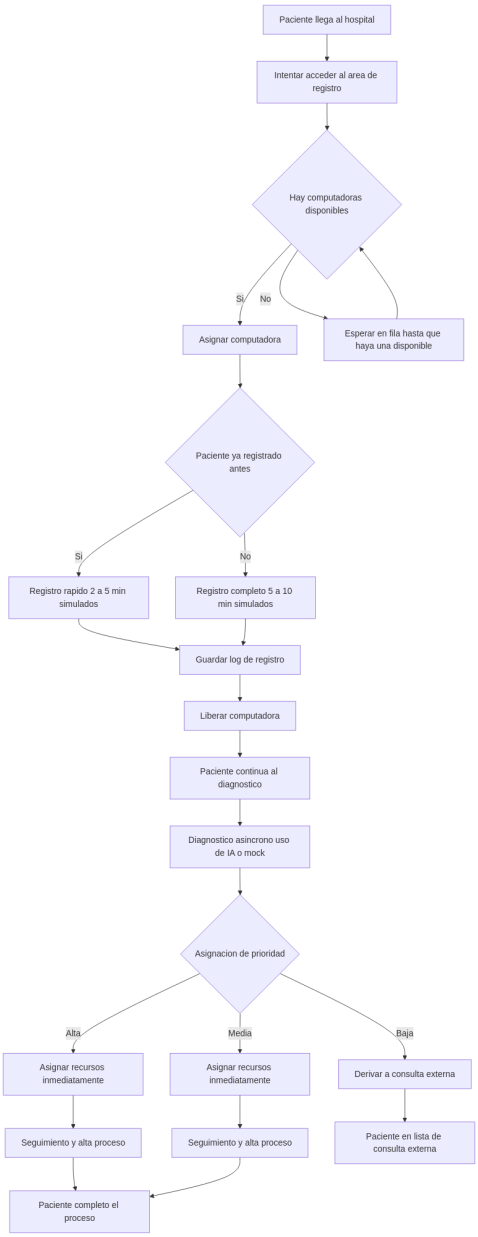

Figure 1: Diagrama de flujo del sistema hospitalario simulado

1.1 Estructura de un paciente

Cada paciente en la simulación es representado mediante una estructura que contiene la siguiente información:

- **ID:** Identificador único generado automáticamente.
- **Nombre:** Nombre del paciente.
- **Género:** Masculino, femenino u otro.
- **Edad:** Edad en años.
- **Raza:** Característica demográfica relevante para el diagnóstico.
- **Altura:** Estatura del paciente en metros.
- **Peso:** Peso del paciente en kilogramos.
- **Hábitos alimenticios:** Descripción general de su alimentación.
- **Antecedentes médicos:** Lista de condiciones médicas previas o enfermedades crónicas.
- **Registrado:** Indica si el paciente ya había sido registrado anteriormente en el sistema.
- **Síntomas:** Cadena de texto que describe lo que el paciente está sintiendo o experimentando al llegar al hospital (por ejemplo: "dolor de cabeza, fiebre y tos leve").

Esta estructura permite simular de forma realista distintos perfiles de pacientes y es utilizada a lo largo de todo el flujo hospitalario: registro, diagnóstico, asignación de recursos y alta.

2 Estructura del sistema

2.1 Organización modular del código

Se decidió separar la lógica de cada componente del sistema en módulos independientes, siguiendo el principio de responsabilidad única, donde cada módulo cumple una función específica.

La estructura general del sistema es la siguiente:

```
hospital/
+-- .venv/                    # Entorno virtual de Python
+-- data/
|   +-- paciente.py           # Clase Paciente (estructura de persona real)
|   +-- pacientes.py          # Conjunto de pacientes
+-- diagnostico_ia.py         # Funcion de diagnostico con OpenIA
+-- diagnostico_mock.py       # Diagnóstico simulado para pruebas
+-- logger_datos.py           # Registro de eventos en archivos CSV
+-- main.py                   # Ejecuta el flujo de la simulación
+-- recursos.py               # Asignación de camas, doctores, etc.
+-- registro.py               # Flujo de registro de un paciente
+-- seguimiento.py            # Seguimiento y alta de pacientes
```

3 Implementación técnica

3.1 Uso de librerías

Para llevar a cabo la simulación del sistema hospitalario, se emplearon diversas librerías estándar de Python que permiten gestionar la concurrencia, el paralelismo y la asincronía, así como medir tiempos y manejar estructuras de datos compartidas:

- `time` — Utilizada para manejar los tiempos simulados dentro del sistema.
- `random` — Empleada para simular variabilidad en los tiempos de espera y decisiones dentro del flujo.
- `threading` — Permite la ejecución concurrente de múltiples pacientes simulados mediante hilos.
- `asyncio` — Utilizada para operaciones asincrónicas, como simulaciones de diagnósticos realizados por IA o APIs externas.
- `multiprocessing` — Empleada para ejecutar tareas en paralelo, como el procesamiento intensivo de diagnósticos en múltiples núcleos.
- `datetime` — Para registrar con precisión eventos temporales en los logs.
- `queue` — Utilizada para coordinar el acceso a recursos limitados (como computadoras de registro o camas).

3.2 Simulación del flujo hospitalario: `main.py`

El archivo `main.py` orquesta el flujo completo del sistema hospitalario. Cada paciente es procesado en un hilo independiente, simulando concurrencia real en un entorno hospitalario. A continuación se describe el flujo general implementado:

- Se define un modo de diagnóstico (`"mock"` o `"ia"`) que determina si se usa un diagnóstico simulado o uno basado en IA.
- Se crea una cola prioritaria para organizar a los pacientes según su nivel de urgencia.
- La función `flujo_paciente` representa el recorrido de un paciente a través del sistema:
  - Registro del paciente (bloque concurrente con `threading`).
  - Diagnóstico asincrónico usando `asyncio`.
  - En caso de prioridad `Alta` o `Media`, se asignan recursos y se realiza el seguimiento del paciente en un proceso separado (`multiprocessing`).
  - Para prioridad `Baja`, el paciente es derivado a consulta externa.
- El archivo principal inicia hilos para cada paciente y espera a que todos terminen antes de finalizar la simulación.

3.3 Registro concurrente de pacientes: `registro.py`

La función `registrar_paciente` simula la llegada y registro de un paciente en el sistema. Para este propósito, se utiliza un `BoundedSemaphore` que limita el acceso simultáneo a un máximo de 5 pacientes, representando cinco computadoras disponibles.

Además, se utiliza una `Queue` compartida para asignar dinámicamente el identificador de la computadora disponible. Este enfoque permite una gestión realista de recursos compartidos.

Durante el registro, se simula el tiempo de espera y el tiempo de atención, diferenciando entre registros rápidos (pacientes recurrentes) y registros completos (pacientes nuevos). Al finalizar, se guarda un log con los datos del evento.

3.4 Diagnóstico asincrónico simulado: `diagnostico_mock.py`

El módulo `diagnostico_mock.py` representa el diagnóstico médico realizado a cada paciente. Esta etapa fue diseñada como una función `async` para simular llamadas a una API de inteligencia artificial con latencia variable. Utiliza la librería `asyncio` para suspender la ejecución durante un tiempo aleatorio que representa la duración del análisis médico (entre 5 y 10 minutos simulados).

La función genera un diagnóstico aleatorio entre varias opciones posibles y clasifica la prioridad del paciente de acuerdo con el contenido del texto. Esta prioridad se utiliza más adelante para decidir si el paciente debe recibir atención inmediata o ser derivado a consulta externa.

También se registra el diagnóstico en un archivo CSV mediante la función `guardar_log()`.

3.5 Diagnóstico con Inteligencia Artificial (OpenAI)

Además del diagnóstico simulado, se implementó una variante que utiliza la API de OpenAI para generar un diagnóstico preliminar con base en las características del paciente. Esta opción representa un escenario más realista donde se consulta un modelo de lenguaje entrenado con información médica general.

Para la integración se empleó el cliente `AsyncOpenAI` del SDK oficial de OpenAI (`openai >= 1.0.0`), el cual permite realizar llamadas asincrónicas a modelos como `gpt-3.5-turbo`. El flujo del diagnóstico IA se adapta de forma dinámica a los datos del paciente, incluyendo síntomas, antecedentes médicos y características demográficas.

El modelo retorna un diagnóstico estructurado en tres secciones:

- **Diagnóstico:** Hipótesis clínica redactada en lenguaje técnico.
- **Prioridad:** Nivel de atención sugerido (`Alta`, `Media` o `Baja`).
- **Recomendación:** Acciones médicas sugeridas para el caso.

El siguiente fragmento muestra un ejemplo del mensaje enviado al modelo:

```
Eres un médico de urgencias en un hospital. Realiza un diagnóstico preliminar
con base en los siguientes datos del paciente: [...] Proporciona un diagnóstico
clínico inicial en lenguaje técnico y profesional. Incluye una prioridad de atención
y una recomendación.
```

Esta variante es controlada mediante la variable `MODO_DIAGNOSTICO` que permite cambiar entre `mock` e `ia`.

Se eligió la variante asincrónica ya que este tipo de operación puede tener latencia considerable. De este modo se evita bloquear la ejecución del resto del sistema hospitalario mientras se espera la respuesta del modelo.

3.6 Asignación de recursos hospitalarios

Una vez que un paciente recibe prioridad `Alta` o `Media`, es dirigido al área médica para ser atendido. En esta fase se modela la disponibilidad limitada de recursos mediante semáforos que controlan el acceso a camas y doctores:

- **Camas disponibles:** Se limita el acceso a un máximo de 10 pacientes simultáneamente mediante un semáforo.
- **Doctores disponibles:** Solo hay 5 doctores disponibles para consulta en paralelo.

El flujo general de esta etapa es el siguiente:

1. El paciente espera una cama disponible. Al obtenerla, permanece en observación un tiempo aleatorio (simulado entre 15 y 30 minutos reales).
2. Posteriormente, espera a ser atendido por un doctor. La consulta médica tiene una duración simulada de entre 20 y 40 minutos.
3. Al finalizar la atención, el paciente libera los recursos y se registra un log detallado con los tiempos de espera y duración de cada fase.

Demostración de ejecución

A continuación se muestra un ejemplo real del flujo completo que sigue un paciente dentro del sistema hospitalario. Este registro corresponde al caso de **Lucía Martínez**, quien fue atendida con prioridad alta debido a una crisis asmática.

```
[IngresoHospital][12:31:13] - Lucía Martínez ha ingresado al sistema.

[RegistroLlegada][12:31:13]
- Lucía Martínez llegó al área de registro y está esperando computadora...
[RegistroComienzo][12:31:13] - Lucía Martínez comenzó su registro en computadora 1 (esperó 0.00 s).
[RegistroFin] - Lucía Martínez terminó su registro (completo (nuevo)) en 0.54 s en computadora 1.

[12:31:13] Lucía Martínez está en diagnóstico (OpenAI)...
[IA] Diagnóstico para Lucía Martínez (12:31:15):
Diagnóstico: Crisis asmática aguda
Prioridad: Alta
Recomendación:
Administer broncodilatadores de acción rápida como salbutamol, oxigenoterapia
y evaluar la necesidad de corticosteroides sistémicos. Realizar monitoreo
continuo de la saturación de oxígeno y la función respiratoria.
Considerar hospitalización si no hay mejoría significativa.

[DiagnósticoListo] Lucía Martínez encolado con prioridad Alta
[Asignación] Lucía Martínez en atención inmediata.

[EsperaCama] - Lucía Martínez esperando cama...
[AsignacionCama][12:31:15] - Lucía Martínez fue asignado a una cama (esperó 0.00 s).
[EsperaDoctor] - Lucía Martínez esperando doctor...
[ConsultaInicio][12:31:17] - Lucía Martínez atendido por un doctor (esperó 0.00 s).
[ConsultaFin][12:31:20] - Lucía Martínez terminó su consulta.
[SalidaAreaMedica] - Lucía Martínez liberó cama y salió del área médica.

[ObservacionInicio][12:31:20] - Lucía Martínez está en observación post-consulta...
[Alta][12:31:23] - Lucía Martínez ha sido dado de alta tras 3.43 segundos de observación.

Lucía Martínez ha completado su proceso.

[Main] Todos los pacientes han sido atendidos o derivados a consulta externa.
```

Este ejemplo ilustra de forma detallada cómo se registran los eventos en tiempo real, permitiendo validar el comportamiento del sistema y analizar su desempeño bajo condiciones de concurrencia y procesamiento asincrónico.