**Informe Técnico: Sistema Experto para Diagnóstico de Fallas en PC**

**Autor/es:** Luna Luciano, Sánchez Sergio y Stella Ventura.

**Materia:** Desarrollo de Sistemas de IA

### 1. Introducción

El presente informe detalla el diseño, desarrollo y funcionamiento de un sistema experto cuyo objetivo principal es asistir en el diagnóstico de fallas comunes de hardware y software en computadoras personales. Ante la complejidad y variedad de problemas que puede presentar un PC, se desarrolló una herramienta de software que aplica principios de inteligencia artificial para guiar al usuario hacia una posible causa del problema.

El proyecto cumple con los objetivos de crear una base de conocimiento con hechos (síntomas) y reglas (diagnósticos), e implementarla a través de un motor de inferencia en Python. Además, el sistema se ha diseñado con una arquitectura modular y flexible, ofreciendo múltiples interfaces de usuario y lógicas de diagnóstico intercambiables, demostrando así una solución robusta y escalable que va más allá de los requisitos iniciales.

### 2. Diseño de la Base de Conocimiento

La base de conocimiento es el corazón de cualquier sistema experto. En este proyecto, se compone de dos elementos fundamentales: la base de hechos y la base de reglas.

#### 2.1. Base de Hechos

La base de hechos define el "vocabulario" que el sistema entiende. Consiste en un conjunto de 7 síntomas observables en una PC, representados como valores booleanos (True si el síntoma está presente, False si no lo está):

* pc\_no\_enciende
* pantalla\_sin\_video
* sistema\_operativo\_lento
* ruidos\_extranos\_hdd
* periferico\_no\_funciona
* hace\_pitidos\_al\_arrancar
* mensajes\_de\_error\_os

#### 2.2. Base de Reglas (Estrategias Implementadas)

Para conectar los hechos con los diagnósticos, se implementaron y compararon dos estrategias distintas, demostrando un análisis profundo del diseño de sistemas expertos:

1. **Estrategia de Reglas Estrictas (SI... ENTONCES...)**: Este es el enfoque clásico, donde una regla solo se "dispara" si todas sus condiciones se cumplen. Por ejemplo: SI 'la PC no enciende' Y 'hace pitidos', ENTONCES el diagnóstico es 'Falla de RAM/Video'. Esta estrategia es transparente pero rígida.
2. **Estrategia de Puntuación por Pesos**: Este es un enfoque más avanzado que maneja la incertidumbre. Cada síntoma aporta una "puntuación" a diferentes diagnósticos. El sistema suma los puntos y presenta el diagnóstico más probable. Por ejemplo, el síntoma 'ruidos extraños en HDD' aporta **25 puntos** al diagnóstico de "Falla de Disco Duro", haciéndolo una pieza de evidencia casi definitiva.

### 3. Motores de Inferencia

El motor de inferencia es el componente lógico que procesa los hechos utilizando la base de reglas. El proyecto cuenta con dos implementaciones intercambiables:

* **Motor de Reglas (motor/logica\_reglas.py)**: Implementa un algoritmo de **encadenamiento hacia adelante**. Revisa secuencialmente las reglas if/elif y ejecuta la primera cuya condición se cumpla con los hechos proporcionados.
* **Motor de Puntuación (motor/logica\_pesos.py)**: Este motor calcula una puntuación para cada diagnóstico posible sumando los pesos de los síntomas activos. Luego, identifica el diagnóstico con la puntuación más alta. Si esta puntuación supera un umbral de confianza mínimo, se presenta como la causa más probable. Este enfoque es superior al manejar casos con información incompleta o ambigua.

### 4. Arquitectura del Software

El sistema fue diseñado con una arquitectura modular para garantizar la separación de responsabilidades, facilitando el mantenimiento y la escalabilidad futura.

* **Paquete motor/**: Contiene toda la lógica de negocio. Es completamente independiente de la interfaz.
* **Paquete gui/**: Contiene el código para la aplicación de escritorio (Tkinter).
* **Paquete templates/**: Almacena el archivo HTML para la interfaz web.
* **Archivos Raíz**: main.py actúa como lanzador, mientras que api\_server.py define el servidor web.

Esta estructura permitió implementar dos "motores" y dos "interfaces" que se pueden combinar, demostrando un alto nivel de "Complejidad y creatividad".

### 5. Interfaces de Usuario

Para maximizar la versatilidad, el sistema puede ser accedido de dos maneras:

1. **Aplicación de Escritorio (GUI)**: Al ejecutar python main.py y elegir la opción 1, se lanza una ventana nativa construida con Tkinter. El usuario puede seleccionar los síntomas mediante checkboxes y obtener un diagnóstico instantáneo en una ventana emergente.
2. **Servidor Web (API + Web)**: La opción 2 levanta un servidor FastAPI. Se puede interactuar con él de dos formas:
   * **Interfaz Web**: Abriendo http://127.0.0.1:8000 en un navegador, se muestra una página HTML interactiva.
   * **API REST**: El endpoint POST /api/diagnostico permite a otros programas enviar síntomas en formato JSON y recibir el diagnóstico, cumpliendo con la consigna original.

### 6. Limitaciones y Consideraciones Éticas

Es fundamental reconocer las limitaciones de este sistema para un uso responsable.

* **Limitaciones**:
  + La base de conocimiento, aunque mejorada, sigue siendo reducida y no cubre todas las fallas posibles.
  + El sistema no maneja la secuencia temporal de los eventos ni síntomas contradictorios.
  + El diagnóstico se basa únicamente en la información proporcionada y no puede realizar pruebas físicas.
* **Consideraciones Éticas**:
  + **No es un reemplazo profesional**: El sistema debe ser considerado una **herramienta de orientación preliminar**. Un diagnóstico incorrecto podría llevar a un usuario a realizar compras innecesarias o a intentar reparaciones que podrían dañar aún más el equipo.
  + **Responsabilidad**: Se debe comunicar claramente al usuario que el resultado es una **probabilidad** y no una certeza absoluta, y que la recomendación final siempre debe ser consultar a un técnico calificado.

### 7. Conclusión

El proyecto ha cumplido exitosamente con todos los objetivos planteados en la consigna, desarrollando un sistema experto funcional y bien estructurado. La decisión de implementar dos motores de inferencia y dos interfaces de usuario no solo demuestra el cumplimiento de los requisitos, sino una exploración profunda de diferentes arquitecturas y lógicas de la IA.

El sistema actual sienta una base sólida para futuras mejoras, como la implementación del desafío de Machine Learning para diagnósticos aún más precisos o la capacidad de guardar y cargar bases de conocimiento desde archivos externos, haciendo la herramienta aún más dinámica y potente.