Agent Orchestration Engineers

Prueba Técnica

Prueba Técnica: Agent Orchestration Engineer (Semi Senior)

Introducción

Esta prueba técnica está diseñada para evaluar tus conocimientos y habilidades en el diseño, implementación y orquestación de sistemas de agentes de IA. Se evaluarán aspectos como arquitectura de sistemas multi-agente, patrones de comunicación, manejo de estados y contextos, y consideraciones de seguridad y escalabilidad.

Tiempo estimado: 90 minutos

Formato: Combinación de preguntas teóricas y ejercicios prácticos

Instrucciones

- Lee cuidadosamente cada sección antes de comenzar
- Proporciona respuestas claras y concisas
- En los ejercicios de código, incluye comentarios explicativos
- Puedes usar Python como lenguaje principal
- Envía tus respuestas en un repositorio Git con instrucciones de ejecución

Parte 1: Conocimiento Teórico (20 minutos)

1.1 Preguntas Conceptuales

- Explica la diferencia entre un sistema de agente único y un sistema multi-agente. ¿Cuándo utilizarías cada uno?
- Describe tres patrones comunes de comunicación entre agentes y sus casos de uso.
- ¿Cómo manejarías la consistencia del contexto en un sistema donde múltiples agentes están trabajando simultáneamente?

1.2 Arquitectura y Diseño

Dibuja un diagrama de arquitectura para un sistema que necesita coordinar tres agentes especializados:

- - Un agente de procesamiento de lenguaje natural
- - Un agente de búsqueda de información
- - Un agente de toma de decisiones

Incluye los flujos de comunicación y componentes principales.

Parte 2: Ejercicio Práctico (50 minutos)

2.1 Implementación de un Sistema Multi-Agente Básico

Desarrolla un sistema simple de orquestación de agentes que realice las siguientes tareas:

```
# Implementa un sistema con dos agentes que colaboren para:
# 1. Procesar un texto de entrada
# 2. Extraer información relevante
# 3. Tomar una decisión basada en reglas
class BaseAgent:
def __init__(self):
pass
def process(self, input_data):
pass
# Implementa las clases necesarias y su orquestación
```

Requisitos:

- Implementa la lógica de comunicación entre agentes
- Incluye manejo de errores y recuperación
- Implementa un sistema de logging para observabilidad
- Añade al menos un test unitario

2.2 Diseño de Escalabilidad (20 minutos)

Responde las siguientes preguntas sobre el sistema que implementaste:

¿Cómo modificarías tu implementación para manejar múltiples instancias de cada agente?

Describe cómo implementarías un sistema de caché para mejorar el rendimiento

Propón una estrategia para el manejo de fallos y recuperación

Criterios de Evaluación

Conocimiento Teórico (30%)

- Comprensión de conceptos fundamentales
- Claridad en las explicaciones
- Consideración de casos edge y limitaciones

Implementación Práctica (40%)

- Calidad y claridad del código
- Manejo de errores y casos edge
- Estructura y organización del código
- Implementación de patrones de diseño apropiados

Diseño y Escalabilidad (30%)

- Consideraciones de arquitectura
- Soluciones propuestas para escalabilidad
- Comprensión de trade-offs en el diseño

Entregables Esperados

Respuestas a las preguntas teóricas
Código fuente del ejercicio práctico
Tests unitarios
Documentación básica de la implementación
Respuestas a las preguntas de escalabilidad

Notas Adicionales

- Se valorará especialmente la claridad en la comunicación y documentación
- El código debe seguir buenas prácticas y estar bien documentado
- Se pueden usar librerías estándar, pero cualquier dependencia externa debe estar justificada