Examen práctico de Analista de Desarrollo

Gerencia de Innovación

Nombre aspirante: Paulo Fernando Mérida Salazar

Teléfono: 59371710

Contestar las siguientes preguntas:

1. **Bases de datos:**
   - 1.1 ¿Qué hace un left join? **En el resultado se muestran todas las filas de la tabla izquierda y las filas que coinciden de la tabla derecha, en caso no exista coincidencia saldrá un valor nulo.**
   - 1.2 ¿Qué hace un inner join**? Devuelve las filas que coinciden en ambas tablas**.
   - **1.3** ¿Cuál es la diferencia entre el cast y convert**? Ambas se utilizan para convertir de un tipo de dato a otro, la diferencia es que cast es genérico de SQL y convert es especifico de SQL Server.**
   - 1.4 ¿Para qué sirve un índice? **Sirve para optimizar la búsqueda de datos en una tabla.**
   - **1.5** ¿Para qué sirve un Store Procedure? **Sirve para reutilizar código y mejorar el rendimiento.**
   - 1.6 ¿Cuál es la diferencia entre un índice Clusteriado y no clusteriado? **Un índice clusterizado determina el orden físico en que se almacenan los datos en el disco, mientras que un índice no clúster es una estructura separada que contiene punteros a los datos**
   - 1.7 ¿Para qué sirve una tabla temporal y como se declara? **Sirve para guardar datos de forma temporal durante la ejecución de una consulta o procedimiento. Se declara utilizando el símbolo # antes del nombre de la tabla**
   - 1.8 ¿Para qué sirve una tabla doble temporal y como se declara? **Se usa para almacenar datos temporales, pero solo visibles en la sesión actual y no en otras conexiones.**
   - 1.9 ¿Para qué sirve la función set dateformat dmy? **Configura el formato de interpretación de fechas para que SQL Server lea día/mes/año.**
   - 1.10 Escriba la sintaxis de un cursor:

```
DECLARE @MiVariable INT;

DECLARE MiCursor CURSOR FOR
  SELECT Id
  FROM MiTabla;
```

```
OPEN MiCursor;

FETCH NEXT FROM MiCursor INTO @MiVariable;

WHILE @@FETCH_STATUS = 0
BEGIN
   PRINT 'El valor es: ' + CAST(@MiVariable AS VARCHAR);
   FETCH NEXT FROM MiCursor INTO @MiVariable;
END

CLOSE MiCursor;
DEALLOCATE MiCursor;
```

1.11 ¿Diferencia entre un truncate table y un drop table? **Truncate table elimina todos los datos, pero mantiene la estructura de la tabla, mientras que drop table elimina la tabla completa junto con su estructura.**

1.12 ¿Indique una herramienta para ver el performance de un gestor de base de datos y que encontramos? **Para SQL Server se tiene SQL Server Management Studio que contiene la herramienta Activity Monitor en la que se puede encontrar los procesos activos y las consultas que se están ejecutando, el consumo del CPU, disco y memoria, los bloqueos y esperas entre transacciones, los índices que se van usando.**

2 **Backend:**

2.1 Explica cómo implementarías una estrategia de cacheo a nivel de aplicación sin utilizar servicios externos (Se puede apoyar de una base de datos relacional como Postgresql o sql server al ser un escenario común donde no se cuenta con la disponibilidad de usar servicios de alguna base de datos dictionary in memory como Redis):

Estrategia de cacheo sin servicios externos se podría crear una tabla en la base de datos para guardar resultados de consultas con su fecha y hora de generación. Antes de hacer la consulta pesada, se revisa si ya está guardada y si no ha caducado por ejemplo.

2.2 Explica el concepto de versionado de APIs y cómo lo implementarías.
Es la práctica de tener múltiples versiones coexistentes de una API para no romper aplicaciones que dependen de versiones antiguas. Se puede implementar usando rutas separadas por versiones y controladores separados por versión.
2.3 ¿Qué son los endpoints idempotentes y por qué son importantes?
Un endpoint es idempotente cuando realizar la misma operación varias veces produce el mismo resultado final que hacerlo una sola vez. Y son importantes porque ayuda a evitar efectos secundarios no deseados cuando hay reintentos
2.4 ¿Qué es la inyección SQL y cómo prevenirla?
Es una vulnerabilidad donde un atacante inserta código SQL malicioso en una consulta para alterar el comportamiento o acceder a datos no autorizados. Para prevenirlos se deben usar consultas parametrizadas y restringir permisos por usuario en la base de datos.
2.5 Explica cómo implementar autenticación y autorización en una aplicación backend (incluyendo como manejarías el almacenamiento seguro de contraseñas) para evaluar su conocimiento de encriptación simétrica y asimétrica.

Implementaría autenticación es algo indispensable para una aplicación también la autorización solo para aquellas personas que lo requieran, y encriptación en contraseñas o documentos sensibles

2.6 ¿Cómo identificarías y resolverías un cuello de botella en una aplicación backend?
Por medio de herramientas de monitoreo y también realizando pruebas de carga para identificar lentitud en consultas, llamadas externas o inclusive en el CPU. Para resolverlas lo mejor sería optimizar las consultas, usando por ejemplos índices o cache, también se podrían implementar colas para los procesos que son lentos.
2.7 Explica las diferencias entre concurrencia y paralelismo y cómo afectan el diseño de aplicaciones backend.

La concurrencia son varios procesos o tareas progresando a la vez y el paralelismo es la ejecución simultanea real de tareas, para ello se requieren múltiples núcleos o procesadores. La concurrencia ayuda a mejorar la capacidad de respuesta de nuestro backend y el paralelismo ayuda a mejorar el rendimiento total.

2.8 ¿Cómo manejarías la sincronización de hilos en un entorno multihilo para evitar condiciones de carrera? ***usar* locks, mutex *o* semáforos *para evitar que varios hilos modifiquen el mismo dato al mismo tiempo Utilizaría bloqueos para que solo un hilo pueda acceder a la vez y estructuras de datos thread-safe.***

2.9 ¿Cuáles son los problemas asociados con el uso de números de punto flotante en aplicaciones financieras? Proporcione ejemplos específicos.

Lo principal es que no representan exactamente todos los decimales y por lo mismo se encuentran errores de precisión. Por ejemplo en muchos lenguajes esta expresión: 0.1 + 0.2 == 0.3 es falsa.

2.10 ¿Cómo afecta el uso excesivo de bloques try-catch y la implementación de recursión profunda al rendimiento de una aplicación backend? En tu respuesta, analiza el impacto en el uso de la CPU y la memoria RAM, y explica los conceptos de ciencias de la computación que sustentan estos efectos.
Cada bloque agrega sobrecarga de ejecución y manejo de pila de excepciones y si se lanza una excepción, el sistema tiene que recorrer la pila lo que consume CPU.

Muchos try-catch: pueden volver el código más lento porque manejar excepciones cuesta recursos y la recursión profunda consume mucha memoria en la pila y puede saturar la CPU si no está controlada.

3. Desarrollo web y mobile

3.1 ¿Cuáles son las principales diferencias entre el Renderizado del Lado del Servidor (SSR) y el Renderizado del Lado del Cliente (CSR)? Analiza sus ventajas y desventajas en términos de rendimiento, SEO y experiencia de usuario.

Básicamente seria la carga inicial que se tiene en el servidor

3.2 Explica qué es el Critical Rendering Path en el contexto del desarrollo frontend y describe las técnicas avanzadas que utilizarías para optimizarlo, reduciendo el tiempo de carga percibido por el usuario.

es la secuencia de pasos para mostrar la página desde que llega el HTML hasta que se ve en pantalla.

3.3 Describe cómo implementarías un middleware personalizado en cualquier gestor de estados (Redux o Zustand) para manejar efectos secundarios complejos, como llamadas a APIs externas y gestión de autenticación, asegurando que el flujo de datos permanezca predecible y mantenible.

Mediante un intermediario que intercepta acciones antes de llegar al store lo que vendría a ser un middleware.

3.4 Explica las diferencias entre REST y GraphQL en el contexto del desarrollo frontend y describe cómo implementarías una solución híbrida que aproveche las ventajas de ambos enfoques.

REST devuelve datos predefinidos por otra parde GraphQL el cliente pide exactamente lo que necesita y en hibrito REST para lo estándar y GraphQL para situaciónes flexibles.

3.5 Explica los conceptos de Progressive Enhancement y Graceful Degradation en el desarrollo frontend y cómo los aplicarías para asegurar una experiencia de usuario consistente en diferentes navegadores y dispositivos.

Progressive Enhancement: empezar con lo básico y luego agregar funciones avanzadas si el navegador lo permite.

Graceful Degradation: crear algo completo y asegurarse que funcione aceptablemente en navegadores viejos.

3.6 Explica la importancia de la accesibilidad en el desarrollo y describe las mejores prácticas para asegurar que una aplicación web o mobile sea accesible para todos los usuarios, incluyendo aquellos con discapacidades o poca educación digital (usuarios de escasos recursos con dispositivos demasiado antiguos y con los que les cuesta interactuar).

que todos puedan usar la app, incluso personas con discapacidades o dispositivos limitados acompañado de buen contraste de colores, etiquetas ARIA, navegación por teclado, textos claros y simples.

3.7 Describe las Core Web Vitals (LCP, FID y CLS) y explica cómo optimizarías una aplicación frontend para cumplir con estos indicadores de rendimiento.

LCP: velocidad en mostrar el contenido principal FID: tiempo hasta que el usuario puede interactuar CLS: estabilidad visual Optimizar: imágenes comprimidas, cargar JS de forma diferida, reservar espacio para elementos.

3.8 En el contexto de desarrollar un chat frontend que interactúa con una inteligencia artificial (IA) a través de una API de un modelo de lenguaje grande (LLM), ¿qué tecnología utilizarías para manejar la comunicación en tiempo real: WebSockets o Server-Sent Events (SSE)? Detalla los beneficios de la tecnología seleccionada y explica cómo gestionarías la conexión, el envío y la recepción de mensajes

Para un chat con IA: usaría WebSockets porque permiten comunicación en tiempo real bidireccional y para el tema de la gestión abrir conexión al iniciar, enviar mensajes como JSON, escuchar respuestas y cerrar cuando no se use

4. Inteligencia Artificial

4.1 ¿Cuáles son los principales algoritmos de aprendizaje superficial (shallow learning) y en qué casos usarías cada uno?

Regresión lineal: predecir valores numéricos.

Regresión logística: clasificar en categorías.

Árboles de decisión: clasificaciones o predicciones con reglas.

KNN: clasifica según vecinos cercanos.

SVM: separar datos con una línea o hiperplano.

4.2 ¿Qué es la regresión lineal y cuándo la emplearías?
busca la recta que mejor se ajusta a los datos para predecir valores. Se usa cuando hay relación lineal entre variables.
4.3 ¿Cómo funciona un árbol de decisión y cuál es la diferencia entre criterio Gini y entropía?
Gini: mide impureza.

Entropía: mide desorden

4.4 ¿En qué se diferencian un SVM con kernel lineal y uno con kernel RBF?
separa datos con una línea recta. SVM con kernel RBF: transforma los datos para separarlos en espacios no lineales
4.5 ¿Qué efecto tiene limitar la profundidad de un árbol de decisión?
Limitar la profundidad de un árbol: evita sobreajuste pero puede perder precisión si es muy bajo.
4.6 ¿En qué se diferencian bagging y boosting, y cómo reduce cada uno el error del ensemble?
Bagging: entrena varios modelos en paralelo y promedia resultados.

Boosting: entrena modelos en secuencia corrigiendo errores del anterior.

4.7 ¿Cómo elegirías el número de clusters en K-Means usando el método del codo y la silueta?
busca el punto donde la mejora se estabiliza. Silueta: mide qué tan bien separados están los clusters.

4.8 Define la matriz de confusión y explica cómo se relacionan precisión, recall y la probabilidad posterior de Bayes.

Precisión: aciertos positivos / total positivos predichos.

Recall: aciertos positivos / total positivos reales.

Bayes: ayuda a calcular probabilidades condicionadas.

4.9 ¿En qué consiste un modelo generativo estocástico como Naive Bayes y cómo difiere de un enfoque discriminativo?

usa probabilidad y asume independencia entre variables. Es generativo porque modela cómo se generan los datos, a diferencia de uno discriminativo que solo separa clases.

4.10 ¿Cuál es el propósito de la regularización L1 vs L2 y cómo afecta a los coeficientes del modelo?

elimina coeficientes pequeños dejándolos en 0, L2 reduce coeficientes grandes sin ponerlos en cero.

4.11 ¿Qué arquitecturas de redes (CNN, RNN, Transformer) se usan típicamente en visión, sonido y NLP, y qué ventajas ofrece cada una?

CNN: visión por computador.

RNN: secuencias como texto o audio.

Transformer: procesa texto y datos secuenciales más rápido y en paralelo.

4.12 ¿Cuál es el propósito de una función de activación y por qué ReLU es tan popular?

introduce no linealidad en redes neuronales. ReLU es popular porque es simple y rápida

4.13 ¿Cómo actúa el dropout para prevenir el sobreajuste en redes profundas?

apaga neuronas aleatorias durante el entrenamiento para evitar sobreajuste.

4.14 ¿En qué se diferencian las convoluciones 1D, 2D y 3D y sus aplicaciones típicas?

1D: señales o texto.

2D: imágenes.

3D: video o volúmenes.

4.15 Explica la atención escalada en Transformers y por qué su coste es $O(n^2)$.

compara cada palabra con todas las demás. Su coste es $O(n^2)$ porque analiza todas las combinaciones posibles.

4.16 ¿Podrías explicar detalladamente cómo se estructuran y se interrelacionan los procesos de propagación hacia adelante (forward propagation) y propagación hacia atrás (backpropagation) durante el entrenamiento de una red neuronal utilizando el algoritmo de descenso de gradiente? En tu explicación, incluye:

- ¿Qué es el descenso de gradiente y por qué es fundamental en el entrenamiento de redes neuronales?

Forward: pasa datos por la red y obtiene la salida.

Backward: ajusta pesos calculando el error.

Descenso de gradiente: actualiza los pesos para reducir el error.

4.17 ¿Qué papel juegan las operaciones de convolución en los modelos de visión por computador, y cómo se comparan con enfoques modernos como Vision Transformers? Además, ¿cuáles son las ventajas y limitaciones de utilizar servicios en la nube como AWS Rekognition para tareas de visión frente a modelos personalizados?
extraen características visuales. Vision Transformers usan atención en lugar de filtros.
En cuanto a rekognition es rápido y listo para usar, pero menos personalizable que modelos propios.
4.18 ¿Qué técnicas ofrece OpenCV para mejorar imágenes degradadas por ruido o baja iluminación antes de aplicar detección de bordes o segmentación, y cómo se comparan con enfoques basados en redes neuronales?
ofrece filtros de suavizado, ecualización de histograma. Las redes neuronales modernas pueden mejorar más pero requieren entrenamiento.
4.19 ¿Qué es la calibración de cámara en OpenCV, por qué es importante en visión estereoscópica o reconstrucción 3D, y cómo se realiza usando un patrón de tablero de ajedrez?
corrige distorsiones y obtiene parámetros de lente. Se hace con fotos de un tablero de ajedrez desde distintos ángulos.
4.20 ¿Cómo se puede utilizar visión por computador para estimar el estilo artístico de una imagen (por ejemplo, impresionismo, cubismo, arte pop), y qué técnicas como “Gram Matrix” o redes tipo VGG se utilizan en el proceso de “style transfer”? ¿Qué limitaciones tienen estos enfoques y qué avances modernos existen para lograr una transferencia de estilo más realista y eficiente?
usar *style transfer* con redes como VGG y Gram Matrix. Limitaciones: resultados poco naturales, tiempos largos. Avances: redes más rápidas como AdaIN o Diffusion Models.
4.21 ¿Qué retos presenta la implementación de visión por computador en dispositivos móviles o embarcados (como Raspberry Pi o Jetson Nano), y cómo se puede optimizar el uso de OpenCV para estos casos?
limitaciones de memoria y CPU. Optimizar usando versiones ligeras de modelos y procesamiento por lotes.
4.22 ¿Cómo influye el diseño de prompts (zero-shot, few-shot, chain-of-thought, tool-aware) en el comportamiento y capacidades de un LLM, y qué implicaciones tiene esto para la planificación, la interpretación de instrucciones complejas y la interacción con herramientas externas?

El tipo de prompt afecta la capacidad del LLM para razonar o usar herramientas

4.23 ¿Qué define a un "agente" en el contexto de la IA y en qué se diferencia de un simple modelo de ML?

Un agente tiene memoria, herramientas y razonamiento; un modelo de ML solo recibe datos y predice.

4.24 ¿Qué impacto tiene el uso de etiquetas XML en los prompts sobre el comportamiento y la interpretación del modelo?

Etiquetas XML en prompts ayudan a estructurar y guiar mejor las respuestas.

4.25 Diseña un prompt largo "sandwich" (goal–persona–context–goal) para un LLM que genere un plan de proyecto de software, explicando cada capa.

Goal: objetivo.

Persona: rol que debe asumir.

Context: información relevante.

Goal: recordatorio final del objetivo.

4.26 ¿Cómo implementaría la autenticación y el control de acceso en un servidor MCP para proteger datos sensibles?

usar autenticación con tokens y roles de acceso.

4.27 ¿Qué técnicas de razonamiento puedes incorporar en un agente para que no solo actúe, sino que justifique sus decisiones (ej. CoT, ToT, SoT)?

Técnicas como *Chain-of-Thought* permiten que el agente explique su razonamiento paso a paso.

4.28 ¿Cuáles son los componentes básicos de un agente (perfil, memoria, razonamiento, herramientas, planificación y evaluación)?

perfil, memoria, razonamiento, herramientas, planificación y evaluación.

4.29 ¿Cómo funciona la arquitectura de un agente autónomo frente a un agente asistente o proxy?

Un agente autónomo actúa sin intervención; uno asistente o proxy depende de órdenes humanas.

4.30 ¿Cómo diseñarías un sistema multiagente en el que diferentes agentes con capacidades especializadas (como planificación, ejecución, búsqueda o memoria) colaboran para resolver una tarea compleja de forma autónoma? Describe cómo se comunican, se coordinan y qué arquitectura facilitaría su escalabilidad y robustez.

En un sistema multiagente, cada agente tiene una función y se comunican por mensajes, usando una arquitectura escalable como *blackboard*.

4.31 ¿Qué patrones de coordinación (broadcast, pipeline, blackboard) existen entre agentes y cuándo usar cada uno?
Broadcast: todos reciben el mensaje.

Pipeline: el resultado de uno pasa al siguiente.

Blackboard: comparten un espacio común.

4.32 ¿Cómo integras feedback humano en bucles de control para mejorar las decisiones de los agentes?
revisar resultados y corregirlos para que el agente aprenda.
4.33 ¿Cómo evalúas el rendimiento y la efectividad de un sistema multi-agente (KPIs, trazabilidad de decisiones)?
Evaluar con KPIs, seguimiento de decisiones y métricas de éxito.

4.34 Imagina que estás diseñando un sistema de inteligencia artificial para coordinar la respuesta a emergencias en una ciudad inteligente. ¿Cómo decidirías entre implementar

4.35 ¿Qué papel juegan los “agentic behavior trees” o los “policy planners” en la toma de decisiones de agentes complejos?
Los behavior trees permiten definir comportamientos jerárquicos y modulares. Los policy planners,, generan planes para obtener objetivos considerando diferentes escenarios y estados posibles

comportamientos organizados en árbol. Policy Planners: generan planes considerando escenarios.

4.36 ¿Qué criterios considerarías al elegir un framework para construir un sistema de agentes inteligentes (p. ej., autónomos, colaborativos, multimodales), y cuál elegirías entre opciones como TEN, Pipecat, LangGraph, Agno, Atomic Agents o Arklex, y por qué?
Elegir framework según facilidad, soporte, escalabilidad
4.37 ¿Qué ventajas ofrece el uso de *structured outputs* (por ejemplo, respuesta en JSON o Pydantic) en la interacción entre agentes o con herramientas externas?
Structured outputs facilitan el intercambio claro de datos entre agentes y herramientas

4.38 ¿Cómo implementarías un agente basado en modelos de lenguaje (LLM) que mejore su rendimiento mediante aprendizaje por refuerzo utilizando herramientas como Agent Reinforcement Trainer (ART)? Describe cómo integrarías componentes clave como la asignación de recompensas, la gestión de trayectorias y la actualización de modelos.
Recompensas por acciones correctas.

Guardar trayectorias.

Ajustar pesos con base en rendimiento.

4.39 ¿Cómo garantizarías la reproducibilidad y auditabilidad de las decisiones de un agente que interactúa con APIs o herramientas externas?
Yo registraría cada interacción del agente con APIs o herramientas externas, como entradas, salidas, marcas de tiempo y contexto
Registrar todas las interacciones con APIs para auditoría.

4.40 ¿Qué es un tokenizer en un LLM y por qué es importante entender cómo tokeniza tu entrada y salida?
es la herramienta que divide el texto en unidades más pequeñas llamadas tokens, es importante porque afecta cómo el modelo procesa y genera texto, influye en la longitud máxima de entrada, el costo computacional y la precisión en la interpretación

divide texto en unidades pequeñas. Importante porque afecta coste y precisión.

4.41 ¿Qué estrategias puedes usar para reducir el número de tokens en un agente sin perder semántica?
podemos usar técnicas como resumir o condensar el texto, eliminar palabras innecesarias o redundantes, usar sinónimos más cortos, y simplificar frases manteniendo la esencia.

resumir, eliminar redundancias, usar frases más cortas

4.42 ¿Cuáles son los beneficios de incorporar un sistema de memoria persistente en agentes de inteligencia artificial, y cómo impacta en su capacidad para mantener contexto, personalizar respuestas y adaptarse a cambios en el tiempo? ¿Qué desafíos surgen al manejar hechos dinámicos, múltiples sesiones y datos empresariales en evolución, y cómo se pueden abordar con arquitecturas modernas de memoria? (ejemplo: Zep AI Agent Memory)

La memoria persistente mantiene contexto entre sesiones y personaliza respuestas. Desafíos: manejar datos cambiantes y grandes volúmenes.

4.43 Qué ventajas aporta el Model Context Protocol (MCP) frente a integraciones con function calling para los LLM?
MCP frente a function calling: MCP estandariza la comunicación y el contexto, haciéndolo más escalable.

4.44 ¿Cómo diseñarías un agente que lea documentos, extraiga imágenes relevantes y devuelva los nombres de los archivos junto con su contexto textual?
Diseñar agente lector de documentos: usar OCR o extracción de texto, buscar imágenes relacionadas, devolver archivo + contexto.

4.45 ¿Qué técnicas de caching y persistencia usarías para evitar reprocesar archivos ya vectorizados en agentes que consultan grandes volúmenes de datos?
guardar datos procesados en disco o base de datos para no recalcular.

4.46 ¿Cómo estructurarías y almacenarías documentos para ser usados en un sistema RAG?
guardarlos en formato vectorizado con metadatos para búsqueda rápida.

4.47 ¿Cómo combinarías búsqueda semántica con clasificación o reranking para mejorar resultados de RAG?.
Combinar búsqueda semántica con *reranking* mejora relevancia de resultados.

4.48 ¿Qué es un Deep Research Agent y en qué se diferencia de otros tipos de agentes de búsqueda?
Deep Research Agent consiste en investigar en profundidad sobre un tema.

4.49 ¿Qué es un Computer Use Agent y en qué se diferencia de otros tipos de agentes de búsqueda?
Computer Use Agent se utiliza para interactuar con interfaces como si fuera un usuario humano.

4.50 ¿Qué ventajas le ves a utilizar una librería como *Instructor* en lugar de hacer una llamada directa a una API de un LLM que devuelve respuestas en formato JSON definido?
La librería Instructor valida y estructura la salida del LLM automáticamente, ahorrando trabajo manual.

4.51 Explica qué es computación evolutiva
Yo defino computación evolutiva como técnicas inspiradas en la evolución natural para encontrar soluciones óptimas.