Preguntas de entrevista para ingeniero backend de Java

Java Core

1. ¿Cuáles son los cuatro principios principales de la POO en Java? Respuesta: Los cuatro principios principales son la Encapsulación, la Herencia, el Polimorfismo y la Abstracción. La Encapsulación oculta el estado interno de un objeto, la Herencia permite la herencia de clases, el Polimorfismo habilita la sobreescritura y sobrecarga de métodos, y la Abstracción proporciona una manera de representar características esenciales sin incluir detalles de fondo.

2. Explique el propósito de las genéricas en Java y proporcione un ejemplo. Respuesta: Las genéricas permiten que los tipos sean parametrizados, habilitando la reutilización de código y la seguridad de tipos. Por ejemplo, `ArrayList<T>` usa un parámetro de tipo `T` para almacenar elementos de cualquier tipo.

3. ¿Cómo se crea un hilo en Java y cuál es su ciclo de vida? Respuesta: Se puede crear un hilo extendiendo `Thread` o implementando `Runnable`. El ciclo de vida incluye los estados Nuevo, Ejecutable, Ejecutándose, Bloqueado, Esperando, Espera Temporizada y Terminado.

4. Describa las diferentes áreas de memoria gestionadas por la JVM. Respuesta: La JVM gestiona el Heap, el Stack, el Área de Métodos, el Stack de Métodos Nativos y el Registro del Contador de Programas. El Heap almacena objetos, mientras que cada hilo tiene su propio Stack para variables locales y llamadas de métodos.

5. ¿Cuál es la diferencia entre excepciones comprobadas y no comprobadas en Java? Respuesta: Las excepciones comprobadas deben ser declaradas o capturadas, mientras que las excepciones no comprobadas no se comprueban en tiempo de compilación. Ejemplos incluyen `IOException` para comprobadas y `NullPointerException` para no comprobadas.

6. ¿Cómo se implementa la serialización en Java y por qué es importante? Respuesta: La serialización se implementa implementando la interfaz `Serializable`. Es importante para guardar y restaurar el estado de un objeto, útil en redes y persistencia.

7. Compare `ArrayList` y `LinkedList` en el Framework de Colecciones de Java. Respuesta: `ArrayList` es adecuado para acceso y recorrido rápidos, mientras que `LinkedList` es mejor para inserciones y eliminaciones. `ArrayList` usa memoria contigua, mientras que `LinkedList` usa nodos con punteros.

8. ¿Qué son las expresiones lambda en Java y cómo se relacionan con las interfaces funcionales? Respuesta: Las expresiones lambda proporcionan una manera concisa de representar una interfaz de método (interfaces funcionales). Se utilizan para implementar interfaces funcionales como `Runnable` o `Comparator`.

9. Explique las operaciones clave disponibles en la API de Streams de Java. Respuesta: La API de Streams incluye operaciones intermedias (por ejemplo, `map`, `filter`) y operaciones terminales (por ejemplo, `forEach`, `collect`). Permiten operaciones de estilo funcional en colecciones.

10. ¿Cómo se usa la reflexión en Java para inspeccionar clases en tiempo de ejecución? Respuesta: La reflexión permite la inspección de clases, métodos y campos usando `Class.forName()`, `getMethods()` y `getField()`. Se usa para comportamiento dinámico y marcos.

Ecosistema Spring

1. ¿Qué es el contenedor IoC de Spring y cómo funciona? Respuesta: El contenedor IoC gestiona beans y sus ciclos de vida. Usa inyección de dependencias para gestionar dependencias, reduciendo el acoplamiento.
2. Explique la auto-configuración de Spring Boot. Respuesta: La auto-configuración configura automáticamente beans en función de las dependencias de la ruta de acceso, simplificando la configuración y reduciendo el código boilerplate.
3. ¿Cómo simplifica Spring Data JPA el acceso a datos? Respuesta: Spring Data JPA proporciona repositorios con operaciones CRUD y métodos de consulta, abstrayendo las interacciones con la base de datos.
4. ¿Para qué se usa Spring Security? Respuesta: Spring Security proporciona mecanismos de autenticación y autorización, protegiendo aplicaciones de acceso no autorizado.
5. Describa el papel de Spring MVC en aplicaciones web. Respuesta: Spring MVC maneja solicitudes web, mapeando URLs a controladores y gestionando vistas y modelos para respuestas web.
6. ¿Qué es Spring Cloud y sus componentes principales? Respuesta: Spring Cloud proporciona herramientas para construir aplicaciones nativas de la nube, incluyendo descubrimiento de servicios (Eureka), interruptores de circuito (Hystrix) y pasarelas de API.
7. ¿Cómo mejora Spring AOP la funcionalidad de la aplicación? Respuesta: AOP permite separar preocupaciones transversales como el registro y la gestión de transacciones del código de negocio, usando aspectos y consejos.
8. ¿Qué es Spring Boot Actuator y qué hace? Respuesta: Actuator proporciona puntos finales para monitorear y gestionar aplicaciones, como comprobaciones de salud, métricas e información del entorno.
9. Explique el uso de perfiles de Spring. Respuesta: Los perfiles permiten diferentes configuraciones para diferentes entornos (por ejemplo, desarrollo, producción), habilitando configuraciones específicas del entorno.
10. ¿Cómo simplifican los starters de Spring Boot la gestión de dependencias? Respuesta: Los starters incluyen todas las dependencias necesarias para una funcionalidad específica, reduciendo la necesidad de gestionar dependencias manualmente.

Arquitectura de Microservicios

1. ¿Qué es el descubrimiento de servicios y por qué es importante? Respuesta: El descubrimiento de servicios automatiza el proceso de localización de servicios, esencial para entornos dinámicos y escalabilidad.
2. Explique el papel de una pasarela de API en microservicios. Respuesta: Una pasarela de API actúa como un punto de entrada único, enrutando solicitudes a los servicios apropiados, manejando seguridad y traducción de protocolos.
3. ¿Qué es el patrón Circuit Breaker y cómo ayuda? Respuesta: El Circuit Breaker evita fallos en cascada interrumpiendo solicitudes a servicios fallidos, permitiéndoles recuperarse.
4. Describa los principios de diseño de API RESTful. Respuesta: Los principios REST incluyen la sin estado, la arquitectura cliente-servidor, la cacheabilidad y la interfaz uniforme, asegurando APIs escalables y mantenibles.
5. ¿Qué es GraphQL y cómo difiere de REST? Respuesta: GraphQL es un lenguaje de consulta para APIs, permitiendo a los clientes solicitar exactamente lo que necesitan, reduciendo la sobrecarga y subcarga.
6. ¿Cómo se maneja la versionado de API en microservicios? Respuesta: El versionado se puede hacer a través de rutas de URL, encabezados o parámetros de consulta, asegurando compatibilidad hacia atrás y transiciones suaves.
7. Explique el patrón Saga en microservicios. Respuesta: Saga gestiona transacciones distribuidas a través de servicios, usando una serie de transacciones locales y compensaciones para fallos.
8. ¿Qué son las comprobaciones de salud en microservicios y por qué son importantes? Respuesta: Las comprobaciones de salud verifican la disponibilidad y el rendimiento de los servicios, cruciales para el monitoreo y la gestión de mallas de servicios.
9. Describa el desarrollo basado en contratos en microservicios. Respuesta: El desarrollo basado en contratos define APIs antes de la implementación, asegurando compatibilidad y desacoplamiento entre servicios.
10. ¿Cómo se implementa la limitación de tasa en microservicios? Respuesta: La limitación de tasa se puede implementar usando middleware o APIs como Spring Cloud Gateway, controlando las tasas de solicitud para prevenir abuso.

Bases de Datos y Caché

1. ¿Qué son los joins SQL y cuándo se usan? Respuesta: Los joins SQL combinan registros de dos o más tablas basados en una columna relacionada, usados para recuperar datos a través de tablas relacionadas.

2. Explique las propiedades ACID en transacciones de bases de datos. Respuesta: ACID significa Atomicidad, Consistencia, Aislamiento y Durabilidad, asegurando un procesamiento de transacciones confiable.
3. ¿Qué es Redis y cómo se usa en caché? Respuesta: Redis es un almacén de valores clave en memoria usado para caché, proporcionando acceso rápido a datos frecuentemente usados.
4. Compare Redis y Memcached para caché. Respuesta: Redis soporta estructuras de datos y persistencia, mientras que Memcached es más simple y rápido para caché básica.
5. ¿Qué es el sharding en bases de datos y por qué se usa? Respuesta: El sharding particiona horizontalmente datos a través de múltiples bases de datos, usado para escalabilidad y rendimiento en grandes sistemas.
6. ¿Cómo simplifica Hibernate las interacciones con la base de datos? Respuesta: Hibernate es un marco ORM que mapea clases Java a tablas de bases de datos, simplificando operaciones CRUD.
7. Explique el agrupamiento de conexiones JDBC. Respuesta: El agrupamiento de conexiones reutiliza conexiones de bases de datos, mejorando el rendimiento al reducir la sobrecarga de creación de conexiones.
8. ¿Qué es una base de datos de series temporales y cuándo se usa? Respuesta: Las bases de datos de series temporales como InfluxDB almacenan datos con marca de tiempo, ideales para monitoreo, IoT y datos de sensores.
9. Describa los niveles de aislamiento de transacciones en bases de datos. Respuesta: Los niveles de aislamiento (No Committed Read, Committed Read, Repeatable Read, Serializable) definen cómo interactúan las transacciones entre sí.
10. ¿Cómo optimiza las estrategias de indexación en bases de datos? Respuesta: Elija índices basados en patrones de consulta, evite sobreindexar y use índices compuestos para consultas de múltiples columnas.

Concurrencia y Multihilo

1. ¿Qué es un deadlock en Java y cómo se puede evitar? Respuesta: Un deadlock ocurre cuando los hilos esperan indefinidamente unos a otros. Se puede evitar evitando esperas circulares y usando temporizadores.
2. Explique el marco de ejecución en Java. Respuesta: El marco de ejecución gestiona la ejecución de hilos, proporcionando grupos de hilos y programación de tareas.
3. ¿Cuál es la diferencia entre Callable y Runnable? Respuesta: Callable puede devolver un resultado y lanzar excepciones, mientras que Runnable no, haciendo a Callable más flexible para tareas que devuelven resultados.

4. Describa el modelo de memoria de Java. Respuesta: El modelo de memoria de Java define cómo los hilos acceden a variables, asegurando la visibilidad y el orden de las operaciones a través de procesadores.

5. ¿Qué es la palabra clave volatile en Java y cuándo debe usarse? Respuesta: Volatile asegura que los cambios en una variable sean visibles para todos los hilos, usado en entornos multihilo para prevenir problemas de caché.

6. ¿Cómo se previenen las condiciones de carrera en aplicaciones multihilo? Respuesta: Use sincronización, bloqueos o operaciones atómicas para asegurar acceso exclusivo a recursos compartidos.

7. Explique el concepto de un bloqueo de lectura-escritura. Respuesta: Los bloqueos de lectura-escritura permiten múltiples lectores o un solo escritor, mejorando la concurrencia permitiendo acceso compartido.

8. ¿Qué es un CountDownLatch y cómo se usa? Respuesta: CountDownLatch permite que un hilo espere a que un conjunto de hilos complete, usado para coordinar la ejecución de hilos.

9. Describa el estriado de bloqueos en Java. Respuesta: El estriado de bloqueos divide un bloqueo en múltiples partes (estriados), permitiendo acceso concurrente a diferentes partes, reduciendo la contención.

10. ¿Cómo se maneja la interrupción de hilos en Java? Respuesta: Los hilos pueden verificar el estado interrumpido y lanzar `InterruptedException`, permitiendo una terminación elegante.

Servidores Web y Balanceo de Carga

1. ¿Para qué se usa comúnmente Nginx? Respuesta: Nginx se usa como servidor web, proxy inverso, balanceador de carga y caché HTTP, conocido por su alto rendimiento y escalabilidad.

2. Explique la diferencia entre un balanceador de carga y un proxy inverso. Respuesta: Un balanceador de carga distribuye el tráfico a través de servidores, mientras que un proxy inverso reenvía solicitudes a servidores backend, a menudo proporcionando caché y seguridad.

3. ¿Qué es HAProxy y por qué se usa? Respuesta: HAProxy es un balanceador de carga y servidor proxy de alta disponibilidad, usado para gestionar y distribuir conexiones de red.

4. ¿Cómo se configura SSL/TLS en un servidor web? Respuesta: SSL/TLS se configura obteniendo certificados y configurando escuchas HTTPS, cifrando datos en tránsito.

5. ¿Qué es la caché del lado del servidor y cómo se implementa? Respuesta: La caché del lado del servidor almacena datos frecuentemente accedidos en memoria, implementada usando herramientas como Varnish o Redis para mejorar el rendimiento.

6. Explique la importancia del registro en servidores web. Respuesta: El registro ayuda a monitorear la actividad del servidor, solucionar problemas y auditar la seguridad, usando herramientas como ELK Stack para análisis.

7. ¿Cuáles son las mejores prácticas para la seguridad de servidores web? Respuesta: Las mejores prácticas incluyen usar encabezados de seguridad, mantener el software actualizado y configurar firewalls para proteger contra amenazas.

8. ¿Cómo se maneja la persistencia de sesiones en el balanceo de carga? Respuesta: La persistencia de sesiones se puede lograr usando sesiones pegajosas o replicación de sesiones, asegurando que las sesiones de usuario permanezcan consistentes.

9. ¿Qué es la descarga de SSL y por qué es beneficiosa? Respuesta: La descarga de SSL descifra el tráfico SSL/TLS en un balanceador de carga, reduciendo la carga del servidor y mejorando el rendimiento.

10. Describa el proceso de escalado horizontal de servidores web. Respuesta: El escalado horizontal implica agregar más servidores para manejar una carga aumentada, gestionado a través de balanceadores de carga y grupos de autoescalado.

CI/CD y DevOps

1. ¿Qué es GitOps y cómo difiere del CI/CD tradicional? Respuesta: GitOps trata la infraestructura como código, usando repositorios Git para gestionar configuraciones y despliegues, enfatizando definiciones declarativas.

2. Explique la estrategia de despliegue Blue/Green. Respuesta: La estrategia de despliegue Blue/Green implica ejecutar dos entornos idénticos, cambiando el tráfico al nuevo entorno tras un despliegue exitoso.

3. ¿Qué es una pipeline de Jenkins y cómo se configura? Respuesta: Una pipeline de Jenkins es una serie de pasos para construir, probar y desplegar software, definida en un Jenkinsfile usando sintaxis declarativa o de script.

4. ¿Cómo se implementa la integración continua en una pipeline CI/CD? Respuesta: La integración continua automatiza la construcción y prueba del código tras los commits, asegurando que el código siempre esté en un estado desplegable.

5. ¿Cuál es el papel de Docker en CI/CD? Respuesta: Los contenedores Docker proporcionan entornos consistentes para construir, probar y desplegar aplicaciones, asegurando la paridad a través de las etapas.

6. Explique el concepto de Infraestructura como Código (IaC). Respuesta: IaC gestiona la infraestructura usando código, permitiendo control de versiones, automatización y consistencia en la configuración de entornos.

7. ¿Cuáles son los beneficios de usar Kubernetes en CI/CD? Respuesta: Kubernetes orquesta aplicaciones contenerizadas, proporcionando escalabilidad, auto-reparación y capacidades de despliegue declarativo.

8. ¿Cómo se maneja el escaneo de seguridad en una pipeline CI/CD? Respuesta: Las herramientas de escaneo de seguridad como SonarQube o OWASP Dependency Check se integran en las pipelines para detectar vulnerabilidades tempranamente.

9. Describa el proceso de revertir un despliegue fallido. Respuesta: Los revertimientos se pueden automatizar usando control de versiones o herramientas CI/CD, volviendo a una versión conocida y estable tras un fallo.

10. ¿Cuál es la importancia de la gestión de entornos en DevOps? Respuesta: La gestión de entornos asegura la consistencia a través del desarrollo, pruebas y producción, reduciendo problemas específicos del entorno.

Patrones de Diseño y Mejores Prácticas

1. ¿Qué es el patrón Singleton y cuándo debe usarse? Respuesta: Singleton asegura que una clase tenga solo una instancia, útil para gestionar recursos compartidos como bases de datos o configuraciones.

2. Explique el patrón Factory y sus beneficios. Respuesta: El patrón Factory proporciona una interfaz para crear objetos sin especificar sus clases, promoviendo un acoplamiento flojo.

3. ¿Qué es el patrón Strategy y cómo promueve la flexibilidad? Respuesta: El patrón Strategy permite seleccionar un algoritmo en tiempo de ejecución, habilitando cambios de comportamiento flexibles sin modificar el código.

4. Describa los principios SOLID y su significancia. Respuesta: Los principios SOLID (Responsabilidad Única, Abierto/Cerrado, Sustitución de Liskov, Segregación de Interfaz, Inversión de Dependencia) guían el diseño para un código mantenible y escalable.

5. ¿Cómo mejora la inyección de dependencias la calidad del código? Respuesta: La inyección de dependencias reduce el acoplamiento externalizando la creación de objetos, haciendo el código más modular y testable.

6. ¿Qué es el origen de eventos y cómo difiere del almacenamiento de datos tradicional? Respuesta: El origen de eventos almacena una secuencia de eventos que describen cambios de estado, permitiendo la reconstrucción del estado y auditorías.

7. Explique el patrón de arquitectura CQRS. Respuesta: CQRS separa comandos (operaciones de escritura) y consultas (operaciones de lectura), optimizando para preocupaciones de escritura y lectura por separado.

8. ¿Cuáles son las mejores prácticas para la refactorización de código? Respuesta: Las mejores prácticas incluyen cambios pequeños e incrementales, mantener pruebas y usar herramientas para refactorizaciones automatizadas.

9. ¿Cómo se aseguran las prácticas de código limpio? Respuesta: Las prácticas de código limpio incluyen nombres significativos, adherencia a estándares y escritura de código auto-documentado.

10. ¿Cuál es la importancia del TDD (Desarrollo Guiado por Pruebas)? Respuesta: TDD implica escribir pruebas antes del código, asegurando que el código cumpla con los requisitos y mejorando la mantenibilidad a través de pruebas continuas.

Seguridad

1. ¿Qué es OAuth2 y cómo se usa para autorización? Respuesta: OAuth2 es un marco de autorización que permite a aplicaciones de terceros acceder a recursos sin compartir credenciales.

2. Explique JWT (JSON Web Tokens) y su papel en la seguridad. Respuesta: JWT proporciona una manera compacta y autónoma de transmitir información de manera segura entre partes, usada para autenticación e intercambio de información.

3. ¿Qué es RBAC y cómo simplifica el control de acceso? Respuesta: El Control de Acceso Basado en Roles asigna permisos a roles, simplificando la gestión de acceso del usuario asignando roles a usuarios.

4. ¿Cómo se previenen los ataques de inyección SQL? Respuesta: Use declaraciones preparadas y consultas parametrizadas para separar código y datos, previniendo la ejecución de SQL malicioso.

5. ¿Qué es XSS (Cross-Site Scripting) y cómo se puede prevenir? Respuesta: XSS permite a los atacantes inyectar scripts en páginas web; se puede prevenir sanitizando entradas y salidas y usando encabezados de seguridad.

6. Explique la importancia del cifrado en la seguridad de datos. Respuesta: El cifrado protege la confidencialidad de los datos convirtiéndolos en un formato ilegible, asegurando que solo las partes autorizadas puedan acceder a ellos.

7. ¿Cuáles son las mejores prácticas para la codificación segura en Java? Respuesta: Las prácticas incluyen la validación de entradas, el uso de bibliotecas seguras y la adherencia a directrices de seguridad como OWASP.

8. ¿Cómo se implementan registros de auditoría en aplicaciones? Respuesta: Los registros de auditoría registran acciones de usuario y eventos del sistema, proporcionando visibilidad y responsabilidad para la seguridad y el cumplimiento.

9. ¿Qué es la autenticación de dos factores y por qué es importante? Respuesta: La autenticación de dos factores añade una capa extra de seguridad requiriendo dos formas de verificación, reduciendo los riesgos de acceso no autorizado.

10. Describa el papel de un Firewall de Aplicación Web (WAF). Respuesta: Un WAF protege aplicaciones web de ataques como inyección SQL y XSS filtrando y monitoreando el tráfico HTTP.

Ajuste de Rendimiento y Optimización

1. ¿Cómo se perfilan aplicaciones Java para problemas de rendimiento? Respuesta: Use herramientas de perfilado como VisualVM o JProfiler para analizar el uso de CPU, memoria y hilos, identificando cuellos de botella.
2. ¿Qué es la optimización de la recolección de basura y por qué es importante? Respuesta: La optimización de la recolección de basura ajusta los parámetros de la JVM para optimizar la gestión de memoria, reduciendo pausas y mejorando el rendimiento.
3. Explique las técnicas de optimización de consultas de bases de datos. Respuesta: Las técnicas incluyen indexación, reescritura de consultas y uso de planes de explicación para mejorar el rendimiento de las consultas.
4. ¿Qué estrategias de caché son efectivas en aplicaciones Java? Respuesta: Las estrategias incluyen caché local, caché distribuida (por ejemplo, Redis) y políticas de expiración de caché para equilibrar rendimiento y consistencia.
5. ¿Cómo se realiza la prueba de carga y estrés para aplicaciones? Respuesta: Use herramientas como JMeter o Gatling para simular altas cargas, identificando límites de rendimiento y cuellos de botella.
6. ¿Cuáles son las mejores prácticas para optimizar APIs RESTful? Respuesta: Las mejores prácticas incluyen minimizar la transferencia de datos, usar serialización eficiente y cachear respuestas para reducir la latencia.
7. ¿Cómo se reduce la latencia de red en sistemas distribuidos? Respuesta: Las técnicas incluyen usar CDNs, optimizar centros de datos y comprimir datos para minimizar el tiempo de transmisión.
8. ¿Qué es el dimensionamiento del grupo de conexiones y cómo se determinan los ajustes óptimos? Respuesta: El dimensionamiento del grupo de conexiones equilibra el rendimiento y el uso de recursos, determinado analizando patrones de tráfico y restricciones de recursos.
9. Explique la importancia del monitoreo y la alerta en la gestión del rendimiento. Respuesta: El monitoreo rastrea la salud y el rendimiento de la aplicación, mientras que la alerta asegura respuestas oportunas a los problemas, previniendo tiempos de inactividad.
10. ¿Cuáles son las señales comunes de cuellos de botella de rendimiento en aplicaciones Java? Respuesta: Las señales incluyen alto uso de CPU, fugas de memoria, tiempos de respuesta lentos e incremento de la actividad de recolección de basura.