

Ingeniero de Backend de Java: Entrevista

Java Core (20 puntos)

1. Comprensión de los principios de POO: Encapsulación, Herencia, Polimorfismo, Abstracción.
2. Generics en Java: Uso de parámetros de tipo, tipos limitados y generics comodín.
3. Multihilo en Java: Creación de hilos, ciclo de vida del hilo y comunicación entre hilos.
4. Gestión de memoria de la JVM: Heap, Stack, PermGen/Survivor spaces, algoritmos de recolección de basura.
5. Manejo de excepciones: Excepciones comprobadas y no comprobadas, bloques try-catch, finally y multi-catch.
6. Serialización en Java: Interfaz Serializable, serialización personalizada con writeObject y readObject.
7. Java Collections Framework: Interfaces List, Set, Map, Queue y sus implementaciones.
8. Expresiones lambda y interfaces funcionales: Uso de predicados, consumidores, proveedores y funciones.
9. API de Streams: Operaciones intermedias y terminales, streams paralelos y tuberías de streams.
10. API de Reflexión: Acceso a clases, métodos y campos en tiempo de ejecución, procesamiento de anotaciones.
11. Java IO vs NIO: Diferencias en manejo de archivos, I/O basado en canales y I/O no bloqueante.
12. API de Fecha y Hora de Java: Trabajo con LocalDate, LocalDateTime y Duration.
13. Redes en Java: Programación de sockets, conexiones URL y clientes HTTP.
14. Seguridad en Java: Criptografía, firmas digitales y prácticas de codificación segura.
15. Módulos de Java: Comprensión del JPMS (Java Platform Module System) y modularidad.
16. Enumeraciones en Java: Uso de enums, valores ordinales y métodos personalizados en enums.
17. Anotaciones en Java: Anotaciones integradas, anotaciones personalizadas y procesamiento de anotaciones.
18. Utilidades de concurrencia en Java: CountDownLatch, CyclicBarrier, Semaphore y Exchanger.
19. Fugas de memoria en Java: Causas, detección y estrategias de prevención.
20. Ajuste de rendimiento en Java: Opciones de JVM, herramientas de perfilado y técnicas de optimización de memoria.

Ecosistema Spring (20 puntos)

21. Contenedor IoC de Spring: Inyección de dependencias, ciclo de vida de beans y alcance.
22. Auto-configuración de Spring Boot: Cómo Spring Boot configura automáticamente los beans.
23. Spring Data JPA: Patrones de repositorio, operaciones CRUD y métodos de consulta.
24. Spring Security: Autenticación, autorización y seguridad de APIs REST.
25. Spring MVC: Métodos de controlador, mapeo de solicitudes y resolución de vistas.
26. Spring Cloud: Descubrimiento de servicios con Eureka, balanceo de carga con Ribbon.
27. Spring AOP: Programación orientada a aspectos, preocupaciones transversales y tipos de asesoramiento.
28. Spring Boot Actuator: Puntos finales de monitoreo, comprobaciones de salud y recolección de métricas.
29. Perfiles de Spring: Configuraciones específicas del entorno y activación de perfiles.
30. Dependencias de Spring Boot Starter: Uso de starters para simplificar la gestión de dependencias.
31. Spring Integration: Integración de diferentes sistemas, mensajería y adaptadores.
32. Spring Batch: Procesamiento por lotes, programación de trabajos y implementaciones de pasos.
33. Spring Cache: Estrategias de caché, anotaciones y gestores de caché.
34. Spring WebFlux: Programación reactiva, I/O no bloqueante y marcos de WebFlux.
35. Spring Cloud Config: Gestión centralizada de configuración para microservicios.
36. Spring Cloud Gateway: Patrones de gateway de API, enrutamiento y filtrado.
37. Pruebas en Spring Boot: Uso de @SpringBootTest, MockMvc y TestRestClient.
38. Spring Data REST: Exposición de repositorios como servicios RESTful.
39. Spring Cloud Stream: Integración con brokers de mensajes como RabbitMQ y Kafka.
40. Spring Cloud Sleuth: Rastreo distribuido y registro en microservicios.

Arquitectura de Microservicios (20 puntos)

41. Descubrimiento de servicios: Cómo funcionan Eureka, Consul y Zookeeper.
42. Gateway de API: Patrones, enrutamiento y seguridad en gateways de API.
43. Circuit Breaker: Implementación de resiliencia con Hystrix, Resilience4j.
44. Arquitectura orientada a eventos: Origen de eventos, brokers de mensajes y manejadores de eventos.
45. Diseño de API RESTful: HATEOAS, diseño sin estado y restricciones REST.
46. GraphQL: Implementación de APIs GraphQL, definiciones de esquema y resolvers.

47. Comunicación entre microservicios: Comunicación síncrona vs asíncrona.
48. Patrón Saga: Gestión de transacciones distribuidas entre servicios.
49. Comprobaciones de salud: Implementación de sondas de actividad y preparación.
50. Desarrollo basado en contrato: Uso de Swagger para contratos de API.
51. Versionado de API: Estrategias para versionar APIs RESTful.
52. Limitación de tasa: Implementación de límites de tasa para prevenir abuso.
53. Patrones de Circuit Breaker: Implementación de retrocesos y reintentos.
54. Despliegue de microservicios: Uso de Docker, Kubernetes y plataformas en la nube.
55. Malla de servicios: Comprensión de Istio, Linkerd y sus beneficios.
56. Colaboración de eventos: Patrones Saga vs Coreografía.
57. Seguridad de microservicios: OAuth2, JWT y gateways de API.
58. Monitoreo y rastreo: Herramientas como Prometheus, Grafana y Jaeger.
59. Pruebas de microservicios: Pruebas de integración, pruebas de contrato y pruebas de extremo a extremo.
60. Base de datos por servicio: Gestión de datos y consistencia en microservicios.

Bases de Datos y Caché (20 puntos)

61. Uniones SQL: Uniones internas, externas, izquierdas, derechas y cruces.
62. Propiedades ACID: Atomicidad, Consistencia, Aislamiento, Durabilidad en transacciones.
63. Bases de datos NoSQL: Almacenes de documentos, almacenes clave-valor y bases de datos de grafos.
64. Caché Redis: Almacén de datos en memoria, estructuras de datos y opciones de persistencia.
65. Memcached vs Redis: Comparación de soluciones de caché.
66. Fragmentación de bases de datos: Particionamiento horizontal y balanceo de carga.
67. Frameworks ORM: Hibernate, MyBatis y especificaciones JPA.
68. Pool de conexiones JDBC: Implementaciones DataSource y ciclo de vida de conexiones.
69. Búsqueda de texto completo: Implementación de búsqueda en bases de datos como Elasticsearch.
70. Bases de datos de series temporales: InfluxDB, OpenTSDB para datos basados en tiempo.
71. Niveles de aislamiento de transacciones: No comprometido, comprometido, lectura repetible, serializable.

72. Estrategias de indexación: Índices B-tree, hash y compuestos.
73. Replicación de bases de datos: Configuraciones maestro-esclavo, maestro-maestro.
74. Copia de seguridad y recuperación de bases de datos: Estrategias para protección de datos.
75. Perfilado de bases de datos: Herramientas como SQL Profiler, registros de consultas lentas.
76. Modelos de consistencia NoSQL: Consistencia eventual, teorema CAP.
77. Migraciones de bases de datos: Uso de Flyway, Liquibase para cambios de esquema.
78. Estrategias de caché: Patrones cache-aside, read-through, write-through.
79. Invalidatez de caché: Gestión de expiración e invalidatez de caché.
80. Pool de conexiones de bases de datos: Configuraciones HikariCP, Tomcat JDBC.

Concurrencia y Multihilo (20 puntos)

81. Ciclo de vida del hilo: Nuevo, ejecutable, en ejecución, bloqueado, en espera, terminado.
82. Mecanismos de sincronización: Bloqueos, bloques sincronizados e intrínsecos.
83. Bloqueos reentrantes: Beneficios sobre bloques sincronizados, equidad y temporizadores.
84. Marco de ejecución: ThreadPoolExecutor, ExecutorService y configuraciones de pool de hilos.
85. Callable vs Runnable: Diferencias y casos de uso.
86. Modelo de memoria de Java: Visibilidad, relaciones happens-before y consistencia de memoria.
87. Palabra clave volatile: Asegurando la visibilidad de cambios de variables entre hilos.
88. Prevención de interbloqueos: Evitación y detección de interbloqueos.
89. Programación asíncrona: Uso de CompletableFuture para operaciones no bloqueantes.
90. ScheduledExecutorService: Programación de tareas con tasas y retrasos fijos.
91. Pools de hilos: Pools fijos, en caché y programados.
92. Franjas de bloqueo: Reducción de la contención de bloqueos con bloqueos franjeados.
93. Bloqueos de lectura-escritura: Permitiendo múltiples lectores o un solo escritor.
94. Mecanismos wait/notify: Comunicación entre hilos usando wait/notify.
95. Interrupción de hilos: Manejo de interrupciones y diseño de tareas interrumpibles.
96. Clases seguras para hilos: Implementación de patrones singleton seguros para hilos.
97. Utilidades de concurrencia: CountDownLatch, CyclicBarrier, Semaphore.
98. Características de concurrencia en Java 8+: Streams paralelos, marco fork-join.

99. Programación multicore: Desafíos y soluciones para procesamiento paralelo.

100. Volcados de hilos y análisis: Identificación de problemas con volcados de hilos.

Servidores Web y Balanceo de Carga (20 puntos)

101. Configuración de Apache Tomcat: Configuración de conectores, context.xml y server.xml.

102. Nginx como proxy inverso: Configuración de proxy_pass, servidores upstream y balanceo de carga.

103. HAProxy para alta disponibilidad: Configuración de failover y persistencia de sesión.

104. Seguridad del servidor web: Configuraciones SSL/TLS, encabezados de seguridad y reglas de firewall.

105. Algoritmos de balanceo de carga: Round Robin, Menos conexiones, Hash IP.

106. Caché del lado del servidor: Uso de Varnish, Redis o caches en memoria.

107. Herramientas de monitoreo: Uso de Prometheus, Grafana y New Relic para monitoreo de servidores.

108. Registro en producción: Registro centralizado con ELK stack o Graylog.

109. Escalado horizontal vs vertical: Comprensión de compromisos y casos de uso.

110. Ajuste de rendimiento del servidor web: Ajuste de hilos de trabajo, temporizadores de conexión y buffers.

111. Caché de proxy inverso: Configuración de encabezados de caché y expiración.

112. Pruebas de carga del servidor web: Herramientas como Apache JMeter, Gatling para pruebas de rendimiento.

113. Descarga SSL: Manejo de terminación SSL/TLS en el balanceador de carga.

114. Endurecimiento del servidor web: Mejores prácticas de seguridad y evaluaciones de vulnerabilidad.

115. Servicio de contenido dinámico vs estático: Optimización de configuraciones del servidor.

116. Clúster de servidores web: Configuración de clústeres para alta disponibilidad.

117. Autenticación del servidor web: Implementación de autenticación básica, digest y OAuth.

118. Formatos de registro del servidor web: Formatos de registro comunes y herramientas de análisis.

119. Límites de recursos del servidor web: Configuración de límites en conexiones, solicitudes y ancho de banda.

120. Copia de seguridad y recuperación del servidor web: Estrategias para recuperación ante desastres.

CI/CD y DevOps (20 puntos)

121. Pipeline de Jenkins como código: Escritura de Jenkinsfiles para pipelines CI/CD.

122. Contenerización con Docker: Creación de Dockerfile, builds multi-etapa y orquestación de contenedores.
123. Orquestación con Kubernetes: Despliegues, servicios, pods y estrategias de escalado.
124. Principios de GitOps: Uso de Git para gestión de infraestructura y configuración.
125. Herramientas de construcción Maven y Gradle: Gestión de dependencias, plugins y ciclo de vida de construcción.
126. Pruebas unitarias e integradas: Escritura de pruebas con JUnit, Mockito y TestNG.
127. Herramientas de cobertura de código: Uso de Jacoco para medir cobertura de código.
128. Análisis estático de código: Herramientas como SonarQube para comprobaciones de calidad de código.
129. Infraestructura como código (IaC): Uso de Terraform, CloudFormation para aprovisionamiento de infraestructura.
130. Despliegues Blue/Green: Minimización de tiempo de inactividad durante despliegues.
131. Despliegues Canary: Implementación gradual de nuevas características.
132. Pruebas automatizadas en pipelines CI: Integración de pruebas con etapas de construcción.
133. Gestión de entornos: Uso de Ansible, Chef o Puppet para gestión de configuración.
134. Mejores prácticas CI/CD: Integración continua, despliegue continuo y entrega continua.
135. Estrategias de retroceso: Implementación de retrocesos automatizados en fallos de despliegue.
136. Escaneo de seguridad: Incorporación de comprobaciones de seguridad como SAST, DAST en pipelines.
137. Pipelines CI/CD para microservicios: Gestión de pipelines para múltiples servicios.
138. Monitoreo de pipelines CI/CD: Alertas sobre fallos de pipeline y problemas de rendimiento.
139. Ecosistema de herramientas DevOps: Comprensión de herramientas como Docker, Kubernetes, Jenkins, Ansible.
140. CI/CD para aplicaciones nativas en la nube: Despliegue de aplicaciones en plataformas en la nube.

Patrones de Diseño y Mejores Prácticas (20 puntos)

141. Patrón Singleton: Implementación de singletons seguros para hilos.
142. Patrón Factory: Creación de objetos sin especificar la clase exacta.
143. Patrón Strategy: Encapsulación de algoritmos y cambio entre ellos.
144. Principios SOLID: Comprensión y aplicación de Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.

145. Inyección de dependencias: Reducción de acoplamiento e incremento de la mantenibilidad del código.
146. Patrón Event Sourcing: Almacenamiento de eventos para reconstruir el estado de la aplicación.
147. Arquitectura CQRS: Separación de responsabilidades de comandos y consultas.
148. Diseño para escalabilidad: Uso de escalado horizontal, particionamiento y balanceo de carga.
149. Técnicas de refactorización de código: Extracción de métodos, renombrado de variables y simplificación de condicionales.
150. Prácticas de código limpio: Escritura de código legible, mantenable y auto-documentado.
151. Desarrollo guiado por pruebas (TDD): Escritura de pruebas antes de la implementación.
152. Versionado de código: Uso de estrategias de ramificación de Git como GitFlow, Desarrollo basado en tronco.
153. Diseño para mantenibilidad: Uso de diseño modular, separación de preocupaciones.
154. Antipatrones a evitar: Clases Dios, código espagueti y acoplamiento fuerte.
155. Diseño para seguridad: Implementación de menor privilegio, defensa en profundidad.
156. Diseño para rendimiento: Optimización de algoritmos, reducción de operaciones de E/S.
157. Diseño para confiabilidad: Implementación de redundancia, tolerancia a fallos y manejo de errores.
158. Diseño para extensibilidad: Uso de plugins, extensiones y APIs abiertas.
159. Diseño para usabilidad: Aseguramiento de que las APIs sean intuitivas y bien documentadas.
160. Diseño para testabilidad: Escritura de código que sea fácil de probar y simular.

Seguridad (20 puntos)

161. OAuth2 y JWT: Implementación de autenticación basada en tokens.
162. Control de acceso basado en roles (RBAC): Asignación de roles y permisos a usuarios.
163. Encabezados de seguridad: Implementación de Content Security Policy, X-Frame-Options.
164. Prevención de inyección SQL: Uso de declaraciones preparadas y consultas parametrizadas.
165. Protección contra Cross-Site Scripting (XSS): Sanitización de entradas y salidas.
166. Encriptación y desencriptación: Uso de AES, RSA para protección de datos.
167. Prácticas de codificación segura: Evitación de vulnerabilidades comunes como desbordamientos de búfer.
168. Implementación de auditorías: Registro de acciones de usuario y eventos del sistema.
169. Manejo de datos sensibles: Almacenamiento seguro de contraseñas con algoritmos de hash.

170. Cumplimiento de regulaciones: GDPR, PCI-DSS y leyes de protección de datos.
171. Implementación de autenticación de dos factores (2FA): Añadir una capa extra de seguridad.
172. Pruebas de seguridad: Pruebas de penetración, evaluaciones de vulnerabilidad.
173. Protocolos de comunicación segura: Implementación de SSL/TLS para encriptación de datos.
174. Gestión de sesiones segura: Gestión de tokens de sesión y temporizadores.
175. Implementación de firewalls de aplicaciones web (WAF): Protección contra ataques comunes.
176. Monitoreo y alertas de seguridad: Uso de herramientas como SIEM para detección de amenazas.
177. Mejores prácticas de seguridad en microservicios: Seguridad de comunicación entre servicios.
178. Implementación de CAPTCHA para protección contra bots: Prevención de ataques automatizados.
179. Seguridad en pipelines CI/CD: Escaneo de vulnerabilidades durante builds.
180. Implementación de seguridad por diseño: Incorporación de seguridad desde el inicio del proceso de desarrollo.

Ajuste de Rendimiento y Optimización (20 puntos)

181. Perfilado de aplicaciones Java: Uso de herramientas como JProfiler, VisualVM para análisis de rendimiento.
182. Ajuste de recolección de basura: Ajuste de parámetros de GC para rendimiento.
183. Optimización de consultas de bases de datos: Indexación, reescritura de consultas y uso de planes de explicación.
184. Estrategias de caché: Uso de caches distribuidos, mecanismos de invalidez de caché.
185. Pruebas de carga y estrés: Identificación de cuellos de botella de rendimiento.
186. Optimización de APIs RESTful: Reducción de tiempos de respuesta, minimización de transferencia de datos.
187. Reducción de latencia de red: Uso de CDNs, optimización de llamadas API.
188. Dimensionamiento de pool de conexiones: Determinación de tamaños óptimos de pool para bases de datos y conexiones.
189. Configuraciones de monitoreo y alertas: Uso de Prometheus, Grafana para monitoreo en tiempo real.
190. Identificación y resolución de cuellos de botella: Perfilado de uso de CPU, memoria e I/O.
191. Optimización de configuraciones de heap de Java: Configuración de tamaños de heap adecuados para diferentes entornos.

192. Reducción de pausas de recolección de basura: Uso de G1GC, ZGC para aplicaciones de baja latencia.
193. Optimización de E/S de disco: Uso de SSDs, configuraciones RAID y optimizaciones del sistema de archivos.
194. Caché vs almacenamiento: Decisión sobre cuándo almacenar datos en caché versus en una base de datos.
195. Optimización de registro: Reducción de sobrecarga de registro y gestión de volúmenes de registros.
196. Optimización de acceso concurrente: Uso eficiente de bloqueos y minimización de contención.
197. Perfilado de uso de memoria: Identificación de fugas de memoria y optimización de asignaciones de objetos.
198. Optimización de tamaños de pool de hilos: Equilibrio entre pocos y muchos hilos.
199. Optimización de estructuras de datos: Elección de las estructuras de datos adecuadas para casos de uso específicos.
200. Métricas y KPIs de rendimiento: Definición y seguimiento de indicadores clave de rendimiento para aplicaciones.