Prueba Técnica Desarrollador BackEnd

1. ¿En cuál/es de los siguientes lenguajes de programación tienes más de 1 año de experiencia?

a) Java
b) Javascript
c) Python
d) Ruby

Otros:

2. ¿Qué diferencias fundamentales tienen los conceptos de Scale Up vs Scale Out?

Scale up o escalar hacia arriba hacer referencia a incrementar los recursos en un solo servidor, mientras que scale Out que significa escalar hacia afuera apuesta por incrementar recursos en una arquitectura distribuida.

3. En tu experiencia, ¿microservicios lo definirías como un paradigma de programación, una arquitectura, un estilo de programación u otra cosa?

Los considero como una arquitectura o estilo de arquitectura que podemos usar para desarrollar aplicaciones

4. Siguiendo con microservicios, un gran desafío en este contexto ha sido el manejo de transacciones. En un sistema monolito es bastante simple delimitar una transacción pues esta ocurre en el mismo contexto. Supongamos que tenemos un microservicio que maneja personas y otro sus direcciones, ¿cómo manejarías la integridad en caso que la persona se registre correctamente pero las direcciones no pudieron guardarse?

En este desafío usaría un modelo de compensación que se encargaría de ejecutar operaciones que compensen las no ejecutadas.

Antes de debió implementar un registro de eventos para detectar el punto del fallo y compensar correspondientemente

5. Describa el patrón circuit breaker en microservicios. ¿Por qué es útil implementarlo?

Se usa en microservicios para mejorar la resistencia y la estabilidad del sistema al manejar fallas y errores de manera más elegante. Su implementación se inspira en el concepto de interruptor automático en la electricidad, que corta el suministro eléctrico cuando hay un problema para evitar daños mayores.

Es útil implementarlo ya que permite prevenir fallos en cascada, ya que al tener un servicio en falla se evita que se falle en cascada

6. Supongamos que tenemos el microservicio A y el microservicio B y existe una invocación sincrónica A a B. ¿Cómo invocarías a B sin tener que registrar la IP y puerto en A?

Utilizando patrones como Service Discovery y Load Balancing, podemos usar Eureka y un api Gateway

8. ¿Qué frameworks de pruebas de aceptación manejas?

He usado Selenium

9. Diseña una API REST que permita manejar el dominio de equipos de fútbol. Considere que se desea hacer operaciones CRUD sobre los clubs y los jugadores de un club. Supone todo lo que estime necesario para completar esta tarea. MER, URL, Response Codes, Payloads. *

> *Se debe enviar y/o compartir proyecto API (Puntos adicionales manejo de JPA, inclusión Swagger, manejo application.yml, Tokenización Endpoints)

10. ¿Para qué sirve OpenID Connect?

Nos sirve para implementar la autenticación y la autorización en un sistema, proporcionando una capa de seguridad y estandarización para la identificación de usuarios en aplicaciones web y móviles. Su uso es común en entornos donde la autenticación es de manera federada y se enfoca en interoperabilidad .

11. ¿Si tuvieras que diseñar una API desde cero, preferirías hacerla en un estilo REST o SOAP? ¿Por qué?

Definitivamente usuario REST, principalmente por la simplicidad y ligereza del formato en comparación SOAP, otra razón es por la capacidad instalada en la actualidad

12. ¿Qué características o ventajas provee el uso de un API Manager?

- ☂ Seguridad Centralizada
- ☂ Deployment Continuo
- ☂ Versionamiento de endpoints
- ☂ Protección DDOS
- ☂ Inyección de Headers
- ☂ Autodescubrimiento
- ☂ Revisión de Código
- ☂ Monitoreo

13. ¿Una clase abstracta puede ser considerada una implementación del patrón de diseño Facade? ¿Por qué?

No, una clase abstracta no se considera una implementación directa del patrón de diseño Facade. El patrón Facade proporciona una interfaz simplificada a un conjunto más grande de interfaces para facilitar el uso de un sistema. Una clase abstracta, por otro lado, generalmente se utiliza para definir una estructura común y compartir código entre clases derivadas, pero no necesariamente para simplificar la interfaz hacia un sistema más grande. El patrón Facade normalmente se implementa mediante una clase específica diseñada para proporcionar una interfaz más sencilla y unificada.

Integración

Las siguientes preguntas buscan evaluar tu conocimiento en Integración.

14. ¿Qué es aquello que es descrito por un documento WSDL? ¿Qué partes contiene?

Describe una interfaz de servicio web. Contiene información sobre:

Operaciones: Las funciones o métodos que el servicio web proporciona.

Tipos de Datos: Los tipos de datos utilizados en las operaciones, definidos utilizando

XML Schema.

Protocolo de Comunicación: Especifica cómo se comunicarán las operaciones, por ejemplo, utilizando SOAP sobre HTTP.

Ubicación del Servicio: La dirección URL o el punto de conexión del servicio web.

15. ¿Consideras el uso de una base de datos compartida un patrón de integración aceptable?

Sí, puede considerarse un patrón de integración aceptable en algunos casos. Este enfoque puede simplificar la integración y facilitar el intercambio de datos entre diferentes sistemas al tener una fuente de datos centralizada.

16. ¿Conoces y/o has aplicado algún EIP (Enterprise integration pattern)?. Cuéntanos tu experiencia.

NO

17. Favor menciona algunos frameworks de integración sobre los cuales tengas experiencia previa.

Apache Camel, Spring integration

18. Supongamos que tenemos un archivo que pesa 1TB, en cada una de sus líneas contiene un registro que debe usarse para invocar otro sistema. Describe tu enfoque para lograr este objetivo.

1. Paginar la carga del archivo
2. Usar técnicas de procesamiento concurrente o en paralelo
3. Gestionar los errores de datos
4. Optimizar recursos hardware y red

Java

Las siguientes preguntas buscan evaluar tu conocimiento general en Java. Si no lo manejas, responde "no" en la primera pregunta.

19. ¿Manejas Java?

- Si
- No

20. ¿Qué imprime por pantalla el siguiente código?

```
1  import java.util.ArrayList;
2  import java.util.List;
3  public class Test {
4   public static void main(String[] args) {
5    List < Integer > elements = new ArrayList < > ();
6    elements.add(50);
7    int firstElmnt = elements.get(1);
8    System.out.println(firstElmnt);
9   }
10 }
```

a) 50
b) null
c) 0
d) Se lanza un IndexOutOfBoundsException en tiempo de ejecución
e) Otros:

21. ¿Qué modicador de acceso permite dar acceso a un miembro de la clase dentro del mismo paquete y sus subclases?

a) private
b) protected
c) public
d) package-private

22. ¿Qué imprime por pantalla el siguiente código?

```java
public class Test {
 public static void main(String[] args) {
  int x = 10;
  int y = 2;
  try {
   for (int z = 2; z >= 0; z--) {
    int ans = x / z;
    System.out.print(ans + " ");
   }
  } catch (Exception e1) {
   System.out.println("E1");
  } catch (ArithmeticException e1) {
   System.out.println("E2");
  }
 }
}
```

a) E1
b) E2
c) 10 5 E1
d) Error de compilación

23. ¿Qué imprime por pantalla el siguiente código?

```java
StringBuilder s1 = new StringBuilder("Java");
String s2 = "Best";
s1.append(s2);
s1.substring(4);
int foundAt = s1.indexOf(s2);
System.out.println(foundAt);
```

a) -1
b) 3
c) 4
d) Error de compilación

24. ¿Cuáles de las siguientes características son propias de Java 8?

☂ Lambdas

- ☂ Default Methods
- ☂ Generic Collections
- ☂ Java Streams
- ☂ Java Time API
- ☂ Explicit GC calls
- ☂ SS Heap
- ☂ Futures

25. ¿Qué propósito tienen las instrucciones final, finally y finalize?

- **Final, se usa como Modificador, en variables, indica que el valor no puede ser cambiado, en métodos que no pueden ser sobre escritos y en clases que no pueden ser extendidas**
- **finally, encierra el código que siempre se va a ejecutar en un bloque try**
- **finalize, este se ejecuta antes del garbage collector, para que realice acciones de limpieza requeridas por el objeto**

Spring Boot

Las siguientes preguntas buscan evaluar tu conocimiento general en Spring Boot. Si no lo manejas, responde "no" en la primera pregunta.

26. ¿Conoces Spring Boot?

- 🌎 Si
- 🌎 No

27. ¿Cómo puedes saber que beans se están configurando en Spring Boot?

Con la anotación @Primary y el método dumpBeans() de la clase ApplicationContext.

28. ¿Qué diferencias tienen las anotaciones @Bean y @Component?

La anotación @Bean se utiliza para declarar métodos de producción de beans en una clase de configuración (@Configuration).

La anotación @Component se utiliza para indicar que una clase es un componente de Spring y debe ser escaneada y administrada automáticamente por el contenedor de Spring.

29. ¿Qué caso de uso tiene Spring Actuator?

Actuator se utiliza para proporcionar información sobre como está operando sobre una aplicación Spring Boot en ejecución

30. ¿Qué es un starter en el contexto de Spring Boot?

Un starter es una dependencia que simplifica la inclusión de dependencias comunes y configuraciones para desarrollar aplicaciones específicas

31. ¿Cómo solucionarías un problema de referencia circular en Spring Boot?

- **Uso de @Lazy**
- **Usar la inyección con constructores**

32. ¿Conoces algún componente de Netflix OSS o Spring Cloud?

a) Hystrix
b) Ribbon
c) Zuul
d) Eureka
e) Zipkin
f) Sleuth
g) Feign